AIX バージョン 7.2

デバイスの管理

注記

本書および本書で記述する製品をご使用になる前に、241 ページの『特記事項』 を必ずお読みください。

本書は AIX バージョン 7.1 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典：
AIX Version 7.2
Device management

発行：
日本アイ・ビー・エム株式会社

担当：
トランスレーション・サービス・センター

目次

本書について

本書は、システムのバックアップおよび復元、物理ストレージおよび論理ストレージの管理、適切なページング・スペースのサイジングなどのタスクを実行する際のオプションの選択に影響する可能性がある全情報をユーザーおよびシステム管理者に提供します。 論理ボリューム、ストレージ、およびリソースの管理などのタスクを実行する方法についての全情報を記載してます。 システム・ユーザーは、コマンドの実行、プロセスの取り扱い、ファイルおよびディレクトリーの取り扱い、ならびに基本的な印刷などのタスクの実行方法を知ることができます。

ユーザーおよびシステム管理者に役立つその他のトピックには、ページング・スペースの作成およびサイズ変更、仮想メモリーの管理、システムのバックアップおよび復元、ハードウェアおよび疑似デバイスの管理、システム・リソース・コントローラー (SRC) の使用、ファイルの保護、ストレージ・メディアの使用、環境ファイルのカスタマイズ、ならびにシェル・スクリプトの作成が含まれます。 本書は、オペレーティング・システムと同梱の文書 CD にも記載されています。

強調表示

本書では、以下の強調表示規則を使用します。

太字	その名前がシステムによって事前に定義されているコマンド、サブルーチン、キーワード、ファイル、構造、ディレクトリー、およびその他の項目であることを示します。また、ボタン、ラベル、アイコンなどのように、ユーザーが選択するグラフィック・オブジェクトであることも示します。
イタリック	実際の名前や値がユーザーによって与えられるパラメーターであることを示します。
モノスペース	特定のデータ値の例、画面に表示されるものと似たテキストの例、 プログラマーとして書くと考えられるものと似たプログラム・コードの一部の例、 システムからのメッセージや実際に入力する情報であることを示します。

AIX 内のケース・センシティブ

AIX® オペレーティング・システムでは、すべてケース・センシティブとなっています。 これは、英大文字と小文字を区別するという意味です。 例えば、**ls** コマンドを使用するとファイルをリストできます。 LS と入力すると、システムはそのコマンドが「`is not found`」と応答します。 同様に、**FILEA**、**FiLea**、および **filea** は、同じディレクトリーにある場合でも、3 つの異なるファイル名です。 予期しない処理が実行されないように、常に正しい大/小文字を使用するようにしてください。

ISO 9000

当製品の開発および製造には、ISO 9000 登録品質システムが使用されました。

デバイスの管理

AIX で使用可能な、異なるデバイスを管理するためのコマンドを使用できます。 管理できるデバイスには、論理ボリューム・マネージャー、ファイルシステム、テープ・ドライブ、およびプリンターが含まれます。

新着情報

オペレーティング・システムおよびデバイスの 管理のトピック集に関する新規情報または大幅に変更された情報を一読してください。

新機能または変更内容を見分ける方法

この PDF ファイルでは、新規情報および変更情報の前後に改訂タグ (>| および |<) が示されている場合があります。

2018 年 2 月

以下の説明は、このトピック集に加えられた更新の要約です。

- 175 ページの『iSCSI オフロード・アダプターのセットアップ』のトピックで、iSCSI オフロード・アダプターのセットアップに関する説明を更新しました。

2017 年 12 月

以下の説明は、このトピック集に加えられた更新の要約です。

- 217 ページの『ストレージ・データ・キャッシングの制約』のトピックで、ストレージ・データ・キャッシングに関する制限を更新しました。

2017 年 10 月

以下の説明は、このトピック集に加えられた更新の要約です。

- 38 ページの『論理ボリューム・ストレージのスペース再利用のサポート』のトピックが追加されました。
- 214 ページの『USB デバイスのサポート』のトピックにおいて、AIX オペレーティング・システムでサポートされる USB デバイスに関する情報を更新しました。

2017 年 6 月

以下の説明は、このトピック集に加えられた更新の要約です。

- 224 ページの『キャッシュ統計情報のモニター』のトピックにおいて、統計情報モニターに関する情報を追加しました。
- 182 ページの『MPIO 可能なデバイスの管理』のトピックにおいて、**lsmpio** コマンドに関する情報を追加しました。

論理ボリューム・マネージャー

論理ボリューム・ストレージの設定と制御のための一連のオペレーティング・システムのコマンド、 ライブラリー・サブルーチン、 およびその他のツールを、論理ボリューム・マネージャー (LVM) と呼びます。

LVM は、ストレージ・スペースのより簡単で柔軟性のある論理 ビューと実際の物理 ディスクの間でデータをマッピングすることによって、 ディスク・リソースを制御します。 LVM は従来のディスク・デバイス・ドライバーより上位で作動するデバイス・ドライバー・コードのレイヤーを使用してこれを行います。

LVM は、論理ボリューム・デバイス・ドライバー (LVDD) と LVM サブルーチン・インターフェース・ライブラリーから成り立っています。 論理ボリューム・デバイス・ドライバー (LVDD) は、すべての入出力を管理し処理する疑似デバイス・ドライバーです。 これは、論理アドレスを物理アドレスに変換し、入出力要求を特定のデバイス・ドライバーに送信します。 *LVM* サブルーチン・インターフェース・ライブラリー には、 システムの論理ボリュームおよび物理ボリュームに対してシステム管理タスクを実行するために、 システム管理コマンドが使用する複数のルーチンが備えられています。

論理ボリューム・マネージャーの概念

論理ボリューム・マネージャーの使用を開始する前に、基本的な構造と用語について理解する必要があります。

オンに変更するプロセス

オンに変更するプロセスは、ボリューム・グループの使用準備ができていて、最新データが入っていることを確認するために LVM が使用するメカニズムの 1 つです。

varyonvg コマンドと **varyoffvg** コマンドは、ユーザーがシステムに定義したボリューム・グループをアクティブまたは非アクティブ (使用可能または使用不可) にします。 システムがボリューム・グループにアクセスするためには、その前にボリューム・グループをオンに変更 (vary on) しておく必要があります。 オンに変更するプロセス中に、LVM は管理データをボリューム・グループに定義された物理ボリュームから読み取ります。 ボリューム・グループ・ディスクリプター領域 (VGDA) とボリューム・グループ用ステータス領域 (VGSA) が入っているこの管理データは、ボリューム・グループのおのおのの物理ボリュームに保管されます。

VGDA には、ボリューム・グループ内の各論理ボリュームの論理区画への物理区画のマッピングを記述する情報だけでなく、 タイム・スタンプなどのその他の重要な情報が入っています。 オンに変更する操作をボリューム・グループに対して実行したときに、 どの物理区画が不整合であるか、およびどの物理ボリュームが欠落している (すなわち、使用不可またはアクティブでない) かなどの情報が VGSA に入っています。

オンに変更する操作がボリューム・グループ内に定義されている 1 つ以上の物理ボリュームにアクセスできないと、 このコマンドは、そのボリューム・グループに定義されているすべての物理ボリュームの名前とその状況を表示します。 これは、このボリューム・グループをオフに変更するかどうかを決定するのに役立ちます。

クォーラム (定足数)

クォーラム (定足数) は、ボリューム・グループの使用準備ができていて、最新データが入っていることを確認するために LVM が使用するいずれかのメカニズムです。

クォーラム (定足数) とは、アクティブなボリューム・グループ・ディスクリプター領域とボリューム・グループ用ステータス領域 (VGDA/VGSA) の数の必要数です。 ディスクに障害が発生した場合の VGDA/VGSA 領域のデータ保全性は、クォーラム (定足数) によって確保されます。 ボリューム・グループ内のどの物理ディスクにも、少なくとも 1 つの VGDA/VGSA があります。 ボリューム・グループが 1 つのディスクに作成された場合、そのボリューム・グループには、最初はそのディスクに存在する 2 つの VGDA/VGSA 領域があります。 ボリューム・グループが 2 つのディスクから構成される場合、1 つのディスクには 2 つの VGDA/VGSA 領域がありますが、もう 1 つのディスクには 1 つの VGDA/VGSA 領域があります。 ボリューム・グループが 3 つ以上のディスクから構成されている場合、 それぞれのディスクには 1 つの VGDA/VGSA が割り当てられます。

少なくともディスクの半分 (VGDA/VGSA を意味する) が LVM により読み取り不能であるとき、クォーラム (定足数) は失われます。 2 つのディスクから構成されるボリューム・グループでは、VGDA/VGSA を 1 つしか持たないディスクが失われても、3 つのうちの 2 つの VGDA/VGSA 領域にアクセスできるので、クォーラム (定足数) はまだ存在します。 2 つの VGDA/VGSA 領域を持つディスクが失われると、この説明は成り立たなくなります。 ボリューム・グループを構成するディスクの数が多くなるほど、1 つのディスクが障害を起こしたときにクォーラム (定足数) が失われる機会は少なくなります。

クォーラム (定足数) が失われると、ボリューム・グループはオフに変更 (vary off) されて、LVM からディスクにアクセスできなくなります。 これによって、そのボリューム・グループに対するそれ以後のディスク入出力を防ぎ、 物理的問題が生じたときにデータが失われないように、または書き込まれることがないようにします。 さらに、オフに変更した結果として、ハードウェア・エラーが発生してサービスを実行する必要があるということを、エラー・ログでユーザーに通知します。

クォーラム (定足数) が失われた場合でも、ボリューム・グループの操作を続行することが望ましい場合もあります。 このような場合、クォーラム (定足数) 検査はそのボリューム・グループに対してオフに変更されます。 このタイプのボリューム・グループは非クォーラム (定足数) ボリューム・グループ と呼ばれます。 非クォーラム (定足数) ボリューム・グループの最も一般的な例は、 論理ボリュームがミラーリングされている場合に起こります。 ディスクが失われたとき、使用不可でなくアクセス可能なディスク上に論理ボリュームのコピーが存在する場合には、データは失われません。 ただし、非クォーラム (定足数) ボリューム・グループにおいても、 ミラーリングされているかどうかに関係なく、データ (コピーも含む) が 1 つ以上の使用不可になったディスク上に存在しているという場合もあります。 このような場合には、ボリューム・グループをオンに変更 (vary on) したままにしておいても、データにアクセスできないことがあります。

ミラー・プール

ミラー・プールを使用するとボリューム・グループの物理ボリュームを分離したプールに分割できます。

ミラー・プールは、1 つ以上の物理ボリュームから成り立っています。 各物理ボリュームは、同時に 1 つのミラー・プールにのみ所属できます。 論理ボリュームを作成するとき、 作成される論理ボリュームの各コピーは 1 つのミラー・プールに割り当てられます。 あるミラー・プールに割り当てられた論理ボリューム・コピーは、そのミラー・プール内の 物理ボリュームから区画を割り振るのみです。 これは、論理ボリューム・コピーが使用できるディスクを制限する能力を提供します。 ミラー・プールがない場合、論理ボリュームの作成または拡張時に、どの物理ボリュームが割り当て用に使用されるかを制限する 唯一の方法は、マップ・ファイルを使用することです。 このようにして、ミラー・プールの使用によりこのプロセスを大きく簡素化します。 ミラー・プールは、**extendvg** コマンドまたは、**chpv** コマンドで作成できます。

新規のミラー・プールを作成するとき、ミラー・プール名を指定する必要があります。 ミラー・プール名は次の規則に準拠する必要があります。

- 英数字、_ (下線)、- (負符号)、または . (ピリオド) のみ使用できます。
- 15 文字以下でなければなりません。
- ボリューム・グループ内で固有でなければなりません。

ミラー・プールがボリューム・グループでいったん使用されると、ボリューム・グループはミラー・プールをサポートしないバージョンの AIX に最早インポートできません。 6.1.1.0 以前のすべての AIX は これに含まれます。 さらに、拡張並行モード LVM を持つミラー・プールを使用するために、クラスター内のすべてのノードはミラー・プールをサポートする必要があります。

ミラー・プールの制限

ミラー・プールの制限事項はミラー・プールの使用についてより厳しい制限事項を強化する ために使用できます。 ミラー・プールの 制限は、以下の 3 つの値のうちの 1 つを持つことができます。

off
ミラー・プールの制限が `off` に設定されると、ミラー・プールの使用に制限は課されません。 これはデフォルト値です。

on
ミラー・プールの制限が on に設定されると、ボリューム・グループ内で作成された各論理ボリュームはあるミラー・プールに割り当てる必要があります。

super
ミラー・プールの制限が `super` に設定されると、以下の制約事項が適用されます。

- ローカル物理ボリュームおよびリモート物理ボリュームは同一のミラー・プールに所属できない。

  注 : ローカル物理ボリュームおよびリモート物理ボリュームについて 詳しくは、HACMP/XD GLVM の資料を参照してください。
- 1 つのボリューム・グループ内で最大 3 つのミラー・プールが可能です。
- 各ミラー・プールは、ボリューム・グループ内の各論理ボリュームの少なくとも 1 つのコピーを含む必要があります。

地理的論理ボリューム・マネージャー

地理的論理ボリューム・マネージャー (GLVM) を使用すると、地理的に離れた場所のお客様のデータのミラー・コピーを管理できます。

GLVM では、重要なデータを遠隔地 の災害時回復サイトにミラーリングすることにより、お客様の業務の保護に役立ちます。 火災、洪水などの災害によりお客様の実動場所で のデータを破壊しそうなとき、お客様の災害時回復サイトでバックアップ・コピーを取ります。

データは、標準 TCP/IP ネットワークでミラーリングされます。 実動サイトおよび災害時回復サイトは同一の物理ネットワークに存在する必要はありません。 2 つのサイト間で ルーターとゲートウェイは使用可能です。 極端に長いディスク・ケーブルの代わりに、リモート・ディスクのアクセス用に TCP/IP ネットワークおよび リモート物理ボリューム (RPV) デバイス・ドライバーが使用されます。

ユーザーは、地理的に遠距離のディスクをリモート物理ボリュームとして構成し、次にこれらのリモート物理ボリュームを ローカル物理ボリュームで結合して地理的にミラーリングされたボリューム・グループを形成できます。 これらのボリューム・グループは、論理ボリューム・マネージャー (LVM) で 管理され、通常のボリューム・グループと同様に作業します。 GLVM は、同期リモート・ミラーリングと非同期リモート・ミラーリングの両方をサポートします。

非クォーラム (定足数) ボリューム・グループ

論理ボリューム・マネージャー (LVM) は、ボリューム・グループ記述領域 (VGDA) またはボリューム・グループ・ステータス領域 (VGSA) のクォーラム (定足数) に満たないときに、ボリューム・グループを自動的に非活動化します。 しかし、影響を受けていない VGDA/VGSA ペアが 1 つでもあれば、グループをオンラインのままにできるオプションを選択できます。 このオプションは、非クォーラム (定足数) ボリューム・グループ を生成します。

LVM は、 再アクティブ化の前に非クォーラム (定足数) ボリューム・グループ内のすべてのディスクにアクセスする必要があります。 これにより、VGDA および VGSA は必ず最新を維持します。

すべての論理ボリュームが少なくとも 2 つのコピーを持つようなシステムでは、非クォーラム (定足数) ボリューム・グループを作成できます。 ディスク障害が起こった場合、アクティブ・ディスクが 1 つでもあれば、 そのボリューム・グループはアクティブのままになります。

注 : ユーザー定義ボリューム・グループおよび **rootvg** ボリューム・グループは、両方とも非クォーラム (定足数) 状態で作動できますが、 ユーザー定義ボリューム・グループと **rootvg** ボリューム・グループでは、非クォーラム (定足数) ボリューム・グループとしてそれらを構成する方法およびハードウェア障害後のリカバリー方法が異なります。 正しい方法を該当するボリューム・グループに対して使用するようにしてください。

非クォーラム (定足数) ボリューム・グループを使用しているときでも、クォーラム (定足数) を失い、**errpt** コマンド出力に次のメッセージが表示される可能性があります。

```
QUORUM LOST,  VOLUME GROUP CLOSING LVM.（クォーラム（定足数）が失われ、
ボリューム・グループは LVM を終了する。）
```

このメッセージが発生するのは、すべての物理ボリュームが存在しない状態にあり、かつ LVM が自動的にボリューム・グループをオフに変更するときです。

このメッセージが QUORUM LOST（クォーラム（定足数）が失われた）と表示しているのは、ボリューム・グループ上のクォーラム (定足数) が使用不可化になることから 1 に対するクォーラム (定足数) 要件が減少するためです。**lsvg *vgname*** コマンドを使用すると、「QUORUM: （クォーラム （定足数）:）」フィールド内のクォーラム (定足数) 値を表示できます。 物理ボリュームが存在しない場合は、なおさらこの最小クォーラム (定足数) 要件に違反することになり、結果として、クォーラム (定足数) メッセージは失われ、ボリューム・グループは自動的にオフに変更されます。

ボリューム・グループの非クォーラム (定足数) 状況への移行

クォーラム (定足数) がない場合でもデータを継続的に使用可能にしておくために、ボリューム・グループを非クォーラム (定足数) 状況に変更することができます。

この手順は、次の構成をもったシステムでよく使用されます。

- 論理ボリュームがミラー化されている 2 つのディスクのボリューム・グループ

- 論理ボリュームが 1 回または 2 回ミラー化されている 3 つのディスクのボリューム・グループ

上記の状況にあるボリューム・グループが非クォーラム (定足数) 状況で動作できる場合は、ディスク障害が起こっても、ボリューム・グループの 少なくとも 1 つのディスクがアクティブである 限り、ボリューム・グループはアクティブのままになります。

非クォーラム (定足数) グループのリカバリーを可能にするには、必ず以下の作業を行います。

- システムで JFS または JFS2 のファイルシステムを 使用する場合は、JFS ログ論理ボリュームをミラーリングする。
- ミラーリングしたコピーを別々のディスクに置く。 構成が明確にわからない場合は、以下のコマンドを入力して、各論理区画の物理ロケーション (PV1、PV2、および PV3) をチェックします。 (コピーを別々のディスクに入れる場合、PV1、PV2、および PV3 カラムにそれぞれ異なる hdisk 番号が含まれていなければなりません。)

  ```
  lslv -m LVName
  ```

  論理ボリュームに同じディスク上にあるコピーだけしかない場合、そのディスクが使用不可 になると、そのボリューム・グループのクォーラム (定足数) 状況あるいは非クォーラム (定足数) 状況に関係なく、ユーザーはそのボリュームを使用できなくなります。

ユーザー定義ボリューム・グループおよび rootvg ボリューム・グループはどちらも、非クォーラム (定足数) 状況で動作できますが、それらの構成およびリカバリーの方式は異なります。

非クォーラム (定足数) ユーザー定義ボリューム・グループをアクティブにするには、そのボリューム・グループの物理ボリュームがすべてアクセス可能でなければなりません。そうでないと、アクティブになりません。非クォーラム (定足数) ボリューム・グループは、最後のディスクがアクセス不能になるまでオンラインにとどまっているため、アクティブにする時には各ディスクをアクセス可能にしておく必要があります。

重要: rootvg ボリューム・グループに関連したディスクが存在しないときは、存在しないディスクを修復できるまでは、システムを電源オンしないでください。 論理ボリューム・マネージャー (LVM) は、常に -f フラグを使用して非クォーラム (定足数) rootvg を強制的に活動化 (vary on) しますが、この操作には危険が伴います。 LVM が強制的に活動化を行う理由は、rootvg が活動化されない限りオペレーティング・システムを始動できないからです。つまり、LVM は、たとえアクセス可能なディスクが 1 つしかなくても、非クォーラム (定足数) rootvg の起動 (オンに変更) をするために、最後の試みを行います。

関連概念

アダプターまたは電源装置障害の場合の高可用性
アダプターや電源装置障害を防ぐには、ご使用の要件に応じて、次のうちいずれかの 1 つ以上を行ってください。

論理ボリューム・マネージャーの構成

論理ボリューム・マネージャー (LVM) は基本オペレーティング・システムと一緒に インストールされるので、それ以上の構成は必要ありません。 ただし、LVM がディスクを使用 できるようにするには、ディスクを構成し、物理ボリュームとして定義しておく必要があります。

関連タスク

アプリケーションのためのロー論理ボリュームの定義
ロー論理ボリューム とは、オペレーティング・システムまたはファイルシステムの 直接の制御下ではなく、アプリケーションの直接の制御下にある物理ディスク・スペースおよび論理ディスク・スペースの領域 のことです。例として、データベースまたは区画をあげることができます。

LVM 保守コマンドおよび高速パス

次の表には、LVM が制御するエンティティー (物理ボリュームおよび論理ボリューム、ボリューム・グループ、およびファイルシステム) の保守で必要な最も簡単なタスクが グループごとに示してあります。

表 *1.* 論理ボリュームおよびストレージの管理のタスク

タスク	*SMIT* 高速パス	コマンドまたはファイル
ボリューム・グループの活動化	**smit varyonvg**	

表 1. 論理ボリュームおよびストレージの管理のタスク (続き)		
タスク	***SMIT* 高速パス**	**コマンドまたはファイル**
データを持たないハード・ディスクの既存のボリューム・グループへの追加	**smit extendvg**	
データを持たないハード・ディスクの新規ボリューム・グループへの追加	**smit mkvg**	
論理ボリュームの追加注 1	**smit mklv**	
ボリューム・グループの追加	**smit mkvg**	
新規ボリューム・グループの追加と活動化	**smit mkvg**	
データ割り当てを使用するための論理ボリュームの変更	**smit chlv1**	
ボリューム・グループ名の変更注 2	1. **smit varyoffvg** 2. **smit exportvg** 3. **smit importvg** 4. **smit mountfs**	1. **varyoffvg** *OldVGName* 2. **exportvg** *OldVGName* 3. **importvg** *NewVGName* 4. **mount all**
自動活動化を使用するためのボリューム・グループの変更	**smit chvg**	
論理ボリューム・ポリシーの変更または設定	**smit chlv1**	
論理ボリュームの新規論理ボリュームへのコピー注 3	**smit cplv**	
同サイズの既存の論理ボリュームへの論理ボリュームのコピー**重要 1**	**smit cplv**	
より小さなサイズの既存の論理ボリュームへの論理ボリュームのコピー**重要 1** 注 3	SMIT は使用しないでください**重要 2**	1. 論理ボリュームの作成。 例: **mklv -y hdisk*N* vg00 4** 2. 新規論理ボリューム上に新規ファイルシステムを作成。 For example: **crfs -v jfs -d hdisk*N* -m /doc -A yes** 3. ファイルシステムのマウント。 例: **mount /doc** 4. 新規マウント・ポイントでのディレクトリーの作成。 例: **mkdir /doc/options** 5. ソースから宛先論理ボリュームへのファイルシステムの転送。 例: **cp -R /usr/adam/oldoptions/* ¥ /doc/options**
より大きなサイズの既存の論理ボリュームへの論理ボリュームのコピー**重要 1**	**smit cplv**	

表 1. 論理ボリュームおよびストレージの管理のタスク (続き)		
タスク	***SMIT* 高速パス**	**コマンドまたはファイル**
ボリューム・グループの非活動化	**smit varyoffvg**	
書き込み検査の使用可能化およびスケジューリング・ポリシーの変更	**smit chlv1**	
論理ボリュームの最大サイズの増加	**smit chlv1**	
論理ボリュームのサイズの増加	**smit extendlv**	
ボリューム・グループ別に論理ボリュームをすべてリスト	**smit lslv2**	
システム内の物理ボリュームをすべてリスト	**smit lspv2**	
ボリューム・グループをすべてリスト	**smit lsvg2**	
物理ボリュームの状況、論理ボリューム、または区画のリスト	**smit lspv**	
ボリューム・グループの内容をリスト	**smit lsvg1**	
論理ボリュームの状況またはマッピングのリスト	**smit lslv**	
データ割り当てをもつ、またはもたない論理ボリュームのミラーリング	**smit mklvcopy**	
取り外し可能ディスクの電源オフ	**smit offdsk**	常時取り外し可能フィーチャーにだけ使用可能
取り外し可能ディスクの電源オン	**smit ondsk**	常時取り外し可能フィーチャーにだけ使用可能
ボリューム・グループからのミラーリングの除去	**smit unmirrorvg**	
ボリューム・グループの除去	**smit reducevg2**	
ボリューム・グループの再編成	**smit reorgvg**	
ディスクの構成解除と電源オフ	**smit rmvdsk1** または **smit rmvdsk** 次に **smit opendoor**	

重要 :

1. この手順を使用して既存の論理ボリュームにコピーすると、ユーザーの確認を要求しないで、そのボリューム上のどんなデータも上書きしてしまいます。
2. より小さい論理ボリュームにより大きい論理ボリュームをコピーするときには、SMIT 手順または **cplv** コマンドを使用してはいけません。 使用すると、データの一部 (スーパーブロックなど) がより小さい論理ボリュームにコピーされないため、ファイルシステムが破壊される結果になります。

注 :

1. 論理ボリュームの作成後は、LVM 構造がその論理ボリュームを使用していないので、状態はクローズになります。 その論理ボリュームにファイルシステムがマウントされるか、論理ボリュームがロー I/O のためにオープンされるまで、状態はクローズのままです。
2. **rootvg** の名前の変更、インポート、エクスポートはできません。
3. 特定の論理ボリュームを複製するのに十分な直接アクセス記憶装置を持っている必要があります。

システムを使用可能にしたままのディスクの追加
以下の手順では、常時取り外し可能フィーチャーを使用してディスクをオンにし、構成する方法を説明しています。この機能を使用すると、システムを電源オフ せずにディスクを追加できます。

追加記憶装置用として、またはディスク障害を修正するために、ディスクを追加できます。 この機能は、特定のシステムでしか使用できません。

1. キャビネットの空きスロットにディスクをインストールします。 インストール手順の 詳細については、マシンの「サービス・ガイド」を参照してください。
2. コマンド・ラインに次の高速パスを入力して、新規ディスクを電源オンします。

```
smit ondsk
```

この時点では、ディスクがシステムに 追加されていますが、まだ使用可能にはなっていません。 次にどんな作業を行うかは、新規ディスクにデータが 含まれているかどうかによって異なります。

- ディスクにデータが含まれていない場合は、次のどちらかの作業を行って、ディスクを物理ボリュームとしてボリューム・グループに追加します。
  - ディスクを既存ボリューム・グループに追加するには、コマンド・ラインに次の高速パスを入力します。

    ```
    smit extendvg
    ```
  - ディスクを新規ボリューム・グループに追加するには、コマンド・ラインに次の高速パスを入力します。

    ```
    smit mkvg
    ```
- ディスクにデータが含まれている場合は、データをインポートします。

論理ボリュームの名前の変更
次の手順では、論理ボリュームのデータを失わずに論理ボリュームの名前を変更する方法 について説明します。

次の例では、論理ボリューム名を `lv00` から `lv33` に変更します。

1. 次のように入力して、論理ボリュームに関連したすべてのファイルシステムをアンマウントします。

   ```
   unmount /FSname
   ```

   FSname はファイルシステムの完全名です。

   注 :

   a. アンマウントしようとしているファイルシステムが 使用中の場合は、**unmount** コマンドは失敗します。 **unmount** コマンドを実行できるのは、オープンしているファイルシステムのファイルがなく、ユーザーの現行ディレクトリーがそのデバイス 上にない場合だけです。

   b. **unmount** コマンドの別の名前は **umount** です。 どちらの名前も使用することができます。
2. 以下のように入力して、論理ボリュームの名前を変更します。

   ```
   chlv -n NewLVname OldLVname
   ```

ここで、**-n** フラグは新しい論理ボリューム名 (*NewLVname*) を指定し、*OldLVname* は変更する名前です。 次に例を示します。

```
chlv -n lv33 lv00
```

注 : JFS または JFS2 のログの名前を変更した場合には、名前を変更したログ・デバイスを 使用するすべてのファイルシステムに対して **chfs** コマンドを実行するように、システムからプロンプト指示が出されます。

3. 次のように入力して、ステップ 8 ページの『1』 でアンマウントした ファイルシステムを 再マウントします。

```
mount /test1
```

この時点で、論理ボリュームの名前が変更され、論理ボリュームが使用可能になります。

論理ボリュームの別の物理ボリュームへのコピー

ファイルシステムの 保全性を保ちながら論理ボリュームを別の物理ボリュームにコピー する方法には、必要に応じて、複数の方法があります。

論理ボリュームまたは JFS を別の物理ボリュームにコピーする 方法は複数あります。 それぞれの目的に最適な方法をお選びください。

論理ボリュームのコピー

一番簡単な方法は、**cplv** コマンドを使用して元の論理ボリューム をコピーし、移動先の物理ボリュームに新規論理ボリュームを作成する方法です。

1. 論理ボリュームの使用を中止します。 該当する場合はファイルシステムをアンマウントし、論理ボリュームにアクセスするすべてのアプリケーションを 停止します。
2. 元の論理ボリュームのすべてのデータを収容する容量をもった物理ボリュームを 選択します。

   重要 : より大きな論理ボリュームからより小さな論理ボリューム へデータをコピーする場合は、データの一部 (スーパーブロックを 含む) が失われることにより、ファイルシステムが 壊れる可能性があります。
3. 次のコマンドを使用して、元の論理ボリューム (この例では、**lv00** という名前) をコピーし、新規論理ボリュームを作成します。

   注 : 次の **cplv** コマンドが新規論理ボリュームを作成し、ボリューム・グループがコンカレント・モード で vary on されている場合は、このコマンドは失敗します。

   ```
   cplv lv00
   ```
4. 該当する場合はファイルシステムをマウントし、アプリケーションを再始動して、論理ボリュームの使用を開始します。

この時点で、論理ボリューム・コピーが使用可能になります。

元の論理ボリュームを使用可能のままにした論理ボリュームのコピー

ご使用の環境で、元の論理ボリュームを継続して使用する必要がある場合は、以下の例で示すように、**splitlvcopy** コマンドを使用してその内容をコピーできます。

1. 次の SMIT 高速パスを入力して、論理ボリュームをミラーリングします。

   ```
   smit mklvcopy
   ```
2. 論理ボリュームの使用を中止します。 該当する場合はファイルシステムをアンマウントし、論理ボリュームにアクセスするすべてのアプリケーションを 停止するか、静止モードにします。

   重要 : 次のステップで **splitlvcopy** コマンドを使用します。 必ず、分割する前に論理ボリュームをクローズし、このコマンドを使用する前にデータをもったすべてのファイルシステムをアンマウントしてください。 オープンした論理ボリュームを分割すると、ファイルシステムが 壊れる

ことがあります。また、その論理ボリュームが複数のプロセス によって同時にアクセスされる場合に、元の論理ボリュームとそのコピーとの間の 整合性が失われることもあります。

3. root 権限で、次のコマンドを使用して、元の論理ボリューム (oldlv) を 新規論理ボリューム (newlv) にコピーします。

```
splitlvcopy -y newlv oldlv
```

-y で、新しい論理ボリューム名を指定します。 oldlv ボリュームに論理ボリューム制御ブロックがない場合は、**splitlvcopy** コマンドは正常に完了しますが、newlv ボリュームが論理ボリューム制御ブロックなしに作成された、という旨のメッセージが表示されます。

4. 該当する場合はファイルシステムをマウントし、アプリケーションを再始動して、論理ボリュームの使用を開始します。

この時点で、論理ボリューム・コピーが使用可能になります。

ロー論理ボリュームの別の物理ボリュームへのコピー

ロー論理ボリュームを別の物理ボリュームにコピーするには、次のステップを実行します。

1. 次のコマンドを実行して、ボリューム・グループ内の新規物理ボリュームに 論理ボリュームのミラー・コピーを作成します。

```
mklvcopy LogVol_name 2 new_PhysVol_name
```

2. 次のコマンドを実行して、新規ミラー・コピー内の区画を同期化します。

```
syncvg -l LogVol_name
```

3. 次のコマンドを使用して、論理ボリュームのコピーを物理ボリュームから除去 します。

```
rmlvcopy LogVol_name 1 old_PhysVol_name
```

この時点で、ロー論理ボリューム・コピーが使用可能になります。

ユーザー定義ボリューム・グループのファイルシステム・ログを専用ディスク上に作成

JFS または JFS2 のファイルシステム・ログ は、ファイルシステム・トランザクション・レコードのフォーマットされたリストです。このログを使用すると、トランザクションの完了前にシステムがダウンした場合にも、ファイルシステムの 保全性を保つことができます。ただし、データの保全性は必ずしも保証されません。

専用ディスクは、システムのインストール時に hd8 上に rootvg 用に作成されます。 次の手順を実行すると、他のボリューム・グループ用の別個のディスク上に JFS ログ を作成することができます。 JFS2 ログを作成する場合は、以下のように手順を変更する 必要があります。

- ログ・デバイス・タイプを jfs2log にする。
- **logform** コマンドで、JFS2 ログ・デバイスを指定する ための -V jfs2 オプションを使用する。
- **crfs** コマンドで、jfs の 代わりに jfs2 を指定する。

注 : JFS2 ログをファイルシステムとは別のディスクに配置しなければならないという要件はありません。唯一の要件は、ログ・デバイスをファイルシステムと 同じボリューム・グループに配置する必要があることです。この手順では、パフォーマンスを向上させるために、JFS2 ログを別のディスクに配置する必要があります。

ユーザー定義ボリューム・グループのファイルシステム ・ログ・ファイルを作成すると、特定の状況で、パフォーマンスを向上させることができます。例えば、NFS サーバーを使用していて、このサーバーのトランザクションを他のプロセスと競合させずに処理したいときなどです。

以下の手順を実行して、2 つの物理ボリューム (hdisk1 および hdisk2) をもつ ボリューム・グループ (fsvg1) を作成することもできます。 ファイルシステム (/u/myfs にマウントされる 256 メガバイトのファイルシステム) は hdisk2 に入り、ログは hdisk1 に入ります。JFS ログ・サイズのデフォルトは 4 メガバイトです。 /blv などのあまり使用されないプログラムは、ログと同じ物理ボリュームに入れてもパフォーマンスに影響を与えることはありません。

SMIT およびコマンド・ライン・インターフェースを 使用して、ユーザー定義ボリューム・グループの JFS ログを作成するには、以下のステップを実行します。

1. SMIT 高速パスを使用して、新しいボリューム・グループ (この例では、fsvg1) を追加します。

   ```
   smit mkvg
   ```

2. SMIT 高速パスを使用して、このボリューム・グループに 新しい論理ボリュームを追加します。

   ```
   smit mklv
   ```

3. 「**Add a Logical Volume (論理ボリュームの追加)**」画面で、以下のフィールドにデータを 入力します。次に例を示します。

   ```
   Logical Volumes NAME                    fsvg1log
   Number of LOGICAL PARTITIONS            1
   PHYSICAL VOLUME names                   hdisk1
   Logical volume TYPE                     jfslog
   POSITION on Physical Volume             center
   ```

4. フィールドの設定が完了したら Enter キーを押して変更を確定し、SMIT を終了します。
5. コマンド・ラインで次のコマンドを入力します。

   ```
   /usr/sbin/logform /dev/fsvg1log
   ```

6. 次のプロンプトがでたら、**y** を入力して、Enter キーを押します。

   ```
   Destroy /dev/fsvg1log
   ```

   プロンプトに表示はされていても、何も破棄されません。このプロンプトに 応えて y と入力すると、JFS ログがファイルシステム ・トランザクション を記録できるように、このログの論理ボリュームがシステムによってフォーマットされます。
7. SMIT 高速パスを使用して、別の論理ボリュームを追加します。

   ```
   smit mklv
   ```

8. ステップ 2 で使用したのと同じボリューム・グループ 名 (この例では `fsvg1`) を入力します。「Logical Volume (論理ボリューム)」画面で、以下のフィールドにデータを 入力します。 この論理ボリュームには、ステップ 3 のときとは 異なる物理ボリュームを指定することに注意してください。 次に例を示します。

   ```
   Logical Volumes NAME                    fslv1
   Number of LOGICAL PARTITIONS            64
   PHYSICAL VOLUME names                   hdisk2
   Logical volume TYPE                     jfs
   ```

   フィールドの設定が完了したら Enter キーを押して変更を確定し、SMIT を終了します。
9. 次のコマンド・シーケンスを実行して、新規論理ボリュームにファイルシステムを 追加 し、ログを指定し、さらに新規ファイルシステムをマウントします。

   ```
   crfs -v jfs -d LogVolName -m FileSysName -a logname=FSLogPath
   mount FileSysName
   ```

LogVolName は、ステップ 2 で作成 した論理ボリュームの名前です。*FileSysName* は、この論理ボリュームに マウントするファイルシステムの 名前です。*FSLogPath* は、ステップ 2 で作成した論理ボリュームの名前です。 次に例を示します。

```
crfs -v jfs -d fslv1 -m /u/myfs -a logname=/dev/fsvg1log
mount /u/myfs
```

10. ファイルシステムおよびログが正しく設定されたことを確認するために、次のコマンド (ボリューム・グループ名を置換) を入力します。

    ```
    lsvg -l fsvg1
    ```

    次の例のように、作成した両方の論理ボリュームが、それらのファイルシステム・タイプ と一緒に出力されます。

    ```
    LV NAME             TYPE    ...
    /dev/fsvg1log       jfslog  ...
    fslv1               jfs     ...
    ```

別々の物理ボリュームにある 2 つ以上の論理ボリュームを含んだ、ボリューム・グループが作成されました。それらの論理ボリュームの 1 つに、ファイルシステム・ログが入っています。

注 : 冗長性を提供するために、JFS2 ログ・デバイスに論理ボリューム・レベルでのミラーリングを提供することができます。ただし、ミラーリングを提供することは一般的な方法ではなく、必須でもありません。

ボリューム・グループのインポートまたはエクスポート

以下の表には、インポートおよびエクスポートを使用して、ユーザー定義ボリューム・グループ をあるシステムから別のシステムへ移動する方法が記載されています。 (rootvg ボリューム・グループは、エクスポートもインポートもできません。)

エクスポート手順では、システムからボリューム・グループの定義を除去します。 インポート手順は、ボリューム・グループを新規システムに導入するときに使用します。

以前はシステムと関連付けられていたが、エクスポートされていたボリューム・グループ を、システムに再導入するときにも、インポート手順を使用することができます。 追加するディスクをそれ自身のボリューム・グループ内に入れることによって、インポートおよびエクスポートを使用して、データを含む物理ボリュームを ボリューム・グループに追加することもできます。

重要 : importvg コマンドは、新規システム内に同じ名前の論理ボリュームが既に存在する場合は、インポートした論理ボリュームの名前を変更します。 **importvg** コマンドが論理ボリュームの名前を変更しなければならない場合は、標準エラーにエラー・メッセージ を印刷します。 競合がなければ、**importvg** コマンドは、/etc/filesystems ファイル内にファイル・マウント・ポイントおよびエントリーも作成します。

ボリューム・グループのインポート・タスクおよびエクスポート・タスク		
タスク	*SMIT* 高速パス	コマンドまたはファイル
ボリューム・グループのインポート	**smit importvg**	
ボリューム・グループのエクスポート	1. ボリューム・グループ内の論理ボリューム上のファイルシステムをアンマウントします。 **smit umntdsk** 2. ボリューム・グループをオフに変更します。 **smit varyoffvg** 3. ボリューム・グループをエクスポートします。 **smit exportvg**	

重要 : ページング・スペース・ボリュームがあるボリューム・グループは、そのページング・スペースがアクティブの 間はエクスポートできません。 アクティブなページング・スペースを持つボリューム・グループをエクスポートする 前に、次のコマンドを実行して、そのページング・スペースがシステム 初期化の際に自動的にはアクティブ化されないことを確認してください。

```
chps -a n paging_space name
```

その上で、システムをリブートし、そのページング・スペースがアクティブでなくなるようにします。

物理ボリュームの内容の移行

指定した 1 つ以上の論理ボリュームに属する物理区画を、ボリューム・グループ内の 1 つの物理ボリュームから、1 つ以上の別の物理ボリュームに 移動するには、以下の手順を実行します。 また、この手順を実行して、障害の発生したディスクを 修復や交換する前に、ディスクからデータを移動することもできます。この手順は、ルート・ボリューム・グループ内またはユーザー定義ボリューム・グループ内の 物理ボリュームに使用できます。

重要 : ブート論理ボリュームを物理ボリュームから移行するときに、ソースのブート・レコードを消去する必要があります。これをしないと、システムが停止する場合があります。 **bosboot** コマンド を実行するときには、以下の手順のステップ 4 で説明 されている **chpv -c** コマンドも実行する必要があります。

1. データを新規ディスクに移行したい場合は、以下のステップを実行します。 そうでない場合は、ステップ 2 から続けます。
   a. 次のように入力して、ディスクがシステムに 認識されており、使用可能になっていることを確認します。

   ```
   lsdev -Cc disk
   ```

   次のような出力が表示されます。

   ```
   hdisk0 Available 10-60-00-8,0  16 Bit  LVD  SCSI Disk Drive
   hdisk1 Available 10-60-00-9,0  16 Bit  LVD  SCSI Disk Drive
   hdisk2 Available 10-60-00-11,0 16 Bit  LVD  SCSI Disk Drive
   ```

   b. ディスクがリストされ、使用可能な状態である場合は、次のように入力して、ディスクが別のボリューム・グループに属していないことを確認します。

   ```
   lspv
   ```

   次のような出力が表示されます。

   ```
   hdisk0          0004234583aa7879       rootvg          active
   hdisk1          00042345e05603c1       none            active
   hdisk2          00083772caa7896e       imagesvg        active
   ```

   この例では、hdisk1 を宛先ディスクとして使用できます。3 番目のフィールドを見ると、このディスクがボリューム・グループによって使用されていないことが分かります。

   新規 ディスクがリストされないか、使用不可の場合は、ディスクまたは論理ボリューム・ストレージを構成する必要があります。

   c. 以下を入力して、新規ディスクをボリューム・グループに追加します。

   ```
   extendvg VGName diskname
   ```

   VGName はボリューム・グループの名前であり、*diskname* は新規ディスクの名前です。 この前のステップの例では、*diskname* を hdisk1 に置き換えます。

2. ソース物理ボリュームと宛先物理ボリュームが 同じボリューム・グループ内になければなりません。 両方の物理ボリュームが同じ ボリューム・グループにあるかどうかを判別するため、次のように入力します。

```
lsvg -p VGname
```

VGname はボリューム・グループの名前です。 ルート・ボリューム・グループの出力は次のようになります。

```
rootvg:
PV_NAME        PV STATE       TOTAL PPs   FREE PPs    FREE DISTRIBUTION
hdisk0         active         542         85          00..00..00..26..59
hdisk1         active         542         306         00..00..00..00..06
```

FREE PP の数に注意してください。

3. 移動したいソース・ディスクに対して、ターゲット・ディスク上に十分な余裕がある ことを確認します。

   a. 以下を入力して、ソース・ディスクにある物理区画の数を判別します。

   ```
   lspv SourceDiskName | grep "USED PPs"
   ```

   ここで *SourceDiskName* は、ソース・ディスクの名前です (例えば、hdisk0)。 次のような出力が表示されます。

   ```
   USED PPs:      159 (636 megabytes)
   ```

   この例では、移行を正常に完了するためには、宛先ディスクに 159 の FREE PP が必要になります。

   b. ソース・ディスクの USED PPs の数と宛先ディスクの FREE PPs の 数 (ステップ 2) を比較します。 FREE PPs の 数が USED PPs の数より大きい場合は、移行に十分なスペースがあります。

4. このステップは、rootvg ボリューム・グループ内のディスクからデータを移行する場合にのみ実行します。

   ユーザー定義ボリューム・グループ内のディスクからデータを移行する場合には、ステップ 5 に進みます。

   以下のように入力して、ソース・ディスクにブート 論理ボリューム (**hd5**) があるかどうかをチェックします。

   ```
   lspv -l SourceDiskNumber | grep hd5
   ```

   何も出力されない場合には、ブート論理ボリュームはソース・ディスクには存在しません。 ステップ 5 に進みます。

   次のような出力が表示された場合には、

   ```
   hd5            2   2   02..00..00..00..00   /blv
   ```

   次のコマンドを実行します。

   ```
   migratepv -l hd5 SourceDiskName DestinationDiskName
   ```

   宛先ディスクで **bosboot** コマンドを実行するように 警告するメッセージが表示されます。 **mkboot -c** コマンドも実行して、ソースのブート・レコードを消去する必要があります。 次のコマンド・シーケンスを入力します。

   ```
   bosboot -a -d /dev/DestinationDiskName
   bootlist -m normal DestinationDiskName
   mkboot -c -d /dev/SourceDiskName
   ```

5. 次の SMIT 高速パスを入力して、データを移行します。

   ```
   smit migratepv
   ```

6. 物理ボリュームをリストし、以前に調べたソース物理ボリュームを選択します。

7. 「**DESTINATION** physical volume (**移動先**の物理ボリューム)」フィールドに移動します。 デフォルトを受け入れると、ボリューム・グループ内の全物理ボリュームを転送できます。 デフォルト値でない場合には、移動する区画にとって十分なスペースを持つ 1 つ以上のディスクを選択します (ステップ 4 による)。
8. 必要であれば、この「**LOGICAL VOLUME (論理ボリューム)**」フィールドに属するデータのみを移動するフィールドに進み、論理ボリュームをリストし、選択します。 ソース物理ボリュームとして選択された物理ボリュームに存在する、指定された論理ボリュームに割り当てられた物理区画のみを移動します。
9. 物理区画を移動するために Enter キーを押します。

この時点で、データが新規 (宛先) ディスクに存在するようになります。 ただし、元の (ソース) ディスクはボリューム・グループに残ったままです。 このディスクがまだ信頼できるものであれば、ホット・スペア・ディスクとして 使用し続けることができます。 特にディスクに障害が発生した場合は、以下のステップの実行をお勧めします。

1. ボリューム・グループからソース・ディスクを削除するには、以下を入力します。

```
reducevg VGNname SourceDiskName
```

2. システムから物理的にソース・ディスクを削除するには、以下を入力します。

```
rmdev -l SourceDiskName -d
```

ディスクの構成

新規ディスクはさまざまな方法で構成できます。

次のどれかの方法で、新規ディスクを構成することができます。

- システムをシャットダウンし、電源をオフにできる場合は、方法 1 を使用します。可能な場合はいつでも、物理ディスクをシステムに 接続する際、システムをシャットダウンし、電源をオフにすることが望まれます。
- システムをシャットダウンできないが、サブクラス、タイプ、親名、接続先など、新規ディスクについての詳しい情報がわかっている場合は、メソッド 2 を使用する。
- システムをシャットダウンできず、ディスクの場所しか分からない場合は、メソッド 3 を使用する。

ディスクの構成が終われば、普通の使用法では、ディスクは使用可能になっています。ただし、論理ボリューム・マネージャーで使用する場合は、さらに物理ボリュームとして識別する必要があります。

メソッド 1

ディスクを接続する前にシステムをシャットダウンして電源をオフにできる場合は、以下の手順を実行します。

1. システムに添付されている資料に従って、システムに新規ディスクを物理的に接続し、次に、ディスクおよびシステムの電源をオンにします。
2. システム・ブート中に、構成マネージャー (**cfgmgr**) でディスクを自動的に構成します。
3. システム・ブートの後、root 権限がある状態で **lspv** コマンドをコマンド・ラインに入力し、新しいディスクの名前を探します。
   システムから、次に示すような項目のどちらかが戻されます。

```
   hdisk1  none                    none
```

   または:

```
   hdisk1  00005264d21adb2e        none
```

   最初のフィールドは、ディスクのシステム 割り当て名を示しています。 2 番目のフィールドには、存在すれば、物理ボリューム ID (PVID) が表示されます。 **lspv** 出力の中に新しいディスクがない場合は、「インストールおよび移行」を参照してください。

この時点で、システムでディスクを 使用できるようになります。しかし、LVM でディスクを使用するには、PVID が必要です。 新規ディスクが PVID を もっていない場合は、17 ページの『使用可能ディスクの物理ボリューム化』を参照してください。

メソッド 2

システムをシャットダウンできないが、新規ディスクについて次の情報が分かっている 場合は、次の方法を使用します。

- ディスクの接続方法 (サブクラス)
- ディスクのタイプ (タイプ)
- ディスクの接続先のシステム・アタッチメント (親名)
- ディスクの論理アドレス (接続先)

以下の手順を実行します。

1. システムに添付されている資料に従って、システムに新規ディスクを物理的に接続し、次に、ディスクおよびシステムの電源をオンにします。
2. ディスクを構成し、物理ボリュームとして使用できるようにするために、次の例のように、フラグを付けて **mkdev** コマンド を実行します。

   ```
   mkdev -c disk -s scsi -t 2200mb -p scsi3 ¥
   -w 6,0 -a pv=yes
   ```

   この例では、6 という SCSI ID をもち、0 という論理装置番号 の 2.2 GB のディスクを scsi3 SCSI バスに追加しています。 **-c** フラグでデバイスのクラスを定義しています。 **-s** フラグで サブクラスを定義しています。 **-t** フラグでデバイスのタイプを 定義しています。 **-p** フラグで、割り当てたい親デバイス名 を定義しています。 **-w** フラグでは、SCSI ID および論理装置番号 によってディスクの場所を指定しています。 **-a** フラグでは、デバイス属性値ペア pv=yes を指定しています。この属性で、ディスクを物理ボリュームにし、固有の物理ボリューム ID を もったブート・レコードをディスク上に書き込みます (まだブート・レコードがない場合)。

この時点で、ディスクは、使用可能デバイスおよび物理ボリュームの両方 として定義されます。 新規ディスクの項目をリストするには、コマンド・ラインに **lspv** コマンドを入力します。 **lspv** 出力の中に新しいディスクがない場合は、「インストールおよび移行」を参照してください。

メソッド 3

システムをシャットダウンできず、ディスクの場所しか分からない場合は、次の手順に従います。

1. システムに添付されている資料に従って、システムに新規ディスクを物理的に接続し、次に、ディスクおよびシステムの電源をオンにします。
2. システムで既に構成済みの物理ディスクを調べるために、コマンド・ライン に **lspv** コマンドを入力します。 **lspv** コマンドの詳細については、lspv コマンド のトピックを参照してください。 次のような出力が表示されます。

   ```
   hdisk0       000005265ac63976    rootvg
   ```

3. 構成マネージャーを実行するために、**cfgmgr** をコマンド・ラインに入力します。 構成マネージャーが、新規ディスクを含め、システムに新しく接続されたデバイスを 検出し、構成します。 **cfgmgr** コマンドの詳細については、**cfgmgr** を参照してください。
4. 新規ディスクが構成されたことを確認するために、もう一度 **lspv** コマンドを入力します。 出力は次のどちらかのようになります。

   ```
   hdisk1    none                 none
   ```

   または

   ```
   hdisk1   00005264d21adb2e     none
   ```

最初のフィールドは、ディスクのシステム 割り当て名を示しています。 2 番目のフィールドには、存在すれば、物理ボリューム ID (PVID) が表示されます。 **lspv** 出力の中に新しいディスクがない場合は、「インストールおよび移行」を参照してください。

この時点で、システムでディスクを 使用できるようになります。しかし、LVM でディスクを使用するには、PVID が必要です。 新規ディスクが PVID を もっていない場合は、17 ページの『使用可能ディスクの物理ボリューム化』を参照してください。

使用可能ディスクの物理ボリューム化

ディスクをボリューム・グループに割り当て、LVM が使用できるようにするためには、ディスクを物理ボリュームとして構成する必要があります。

物理ボリュームを構成するには、以下の手順に従います。

1. ディスクが、オペレーティング・システムに認識されていること、使用可能であること、オペレーティング・システムまたはアプリケーションによって 使用中でないことを確認します。 コマンド・ラインに **lspv** コマンドを入力します。 次のような出力が表示されます。

   ```
   hdisk1  none                    none
   ```

   次のことについて、出力を調べます。

   - コマンド出力に新規ディスク名が表示されない場合は、15 ページの『ディスクの構成』を参照してください。
   - 出力の 2 番目のフィールドにシステム 生成の物理ボリューム ID (PVID) (例 `00005264d21adb2e`) が表示される場合は、ディスクは既に物理ボリュームとして構成されており、この手順を完了する必要はありません。
   - 出力の 3 番目のフィールドにボリューム・グループ名 (`rootvg` など) が 表示される場合は、ディスクは使用中であり、この手順を実行するのに適切でないディスクです。

   新規ディスクが PVID をもっておらず、使用中でない場合は、次のステップを続けます。
2. 使用可能ディスクを物理ボリュームに変更するには、**chdev** コマンドをコマンド・ラインに入力します。 次に例を示します。

   ```
   chdev -l hdisk3 -a pv=yes
   ```

   -l フラグでディスクのデバイス名を指定しています。 **-a** フラグでは、デバイス属性値ペア `pv=yes` を指定しています。この属性で、ディスクを物理ボリュームにし、固有の物理ボリューム ID を もったブート・レコードをディスク上に書き込みます (まだブート・レコードがない場合)。

この時点で、ディスクは物理ボリュームとして定義されます。 新規ディスクの項目をリストするには、コマンド・ラインに **lspv** コマンドを入力します。

rootvg の PVID および VGID の変更

システムのブート・フェーズの間に、rootvg ボリューム・グループの物理ボリューム ID (PVID) およびボリューム・グループ ID (VGID) を変更できます。

rootvg の PVID と VGID を変更するには、 *sys0 dev ghostdev* 属性の値を 2 に設定して、システムをリブートします。 *sys0 device ghostdev* 属性は、 ビット単位のフラグです。

- *sys0 device ghostdev* 属性を設定して rootvg ボリューム・グループの PVID と VGID を変更するには、次のコマンドを入力します。

  ```
  chdev -l sys0 -a ghostdev=2
  ```

  注 : ipl_varyon コマンドによって rootvg にあるすべてのディスクの PVID と VGID が変更された後、*sys0 device ghostdev* 属性の値 2 は設定解除されます。 いずれかの rootvg ディスクの PVID を変更する **chdev** コマンドが失敗した場合、**ipl_varyon** コマンドから警告メッセージが送信され、rootvg での vary on を継続します。 rootvg のいずれかのディスクの PVID を変更する **chdev** コマンドが失敗し、次回リブート時に PVID と VGID を変更する場合は、 *sys0 device ghostdev* 属性を再度 2 に設定します。

- *ghostdev* 属性の値をリストするには、次のコマンドを入力します。

```
lsattr -E -l sys0 -a ghostdev
```

ミラーリングされたボリューム・グループ内の障害のある物理ボリュームの交換

次の手順で、ミラーリングされたボリューム・グループ内の障害のある物理ボリューム (PV) を交換します。 **replacepv** コマンドは、ほとんどの構成で障害のある PV を交換する方法を備えています。 **replacepv** コマンドを使用できない構成には、代わりの手順もあります。

ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

概要

障害のある PV を使用する論理ボリュームのすべてに、他の使用可能な PV 上で有効なコピーがありま す (考えられる例外は専用ダンプ論理ボリュームです)。

replacepv コマンドを使用した障害のある PV の交換

以下の前提条件のいずれかを満たすことができない場合は、代替手順を参照してください。

- 障害のある PV を含むボリューム・グループは rootvg ではない。
- 障害のある PV を含むボリューム・グループに交換 PV を追加できる (これは、`MAX PPs per PV` の ような PV サイズおよびボリューム・グループ特性によっては起こり得ないことがあります)。
- 交換 PV は、障害のある PV と同時にシステム内に構成できなければならない。
- 交換 PV の名前は、障害のある PV の名前と異なっていても構わない。
- 交換 PV のサイズは、少なくとも障害のある PV のサイズでなければならない。
- 障害のある PV を含むボリューム・グループは、スナップショット ・ボリューム・グループであって も、スナップショット・ボリューム・グループを持ってもなりません。

障害のある PV が `hdisk2` であり、交換 PV が `hdisk10` であると想定して、 以下のステップを行います。

1. 交換 PV がまだシステムにインストールされていない場合は、インストールに必要なステップを行います。 構成マネージャーを使用して新規 PV を定義するには、次のコマンドを 実行します。

   ```
   cfgmgr
   ```

   lspv コマンドを使用して、PV に割り当てられた名前を判別します。 この例の場合、新規 PV の名前 は `hdisk10` であるとします。
2. 障害のある PV をステップ 1 で定義されたものと交換するには、次のコマンドを実行します。

   ```
   replacepv hdisk2 hdisk10
   ```

   コマンドが実行 すると、`hdisk2` は `hdisk10` に交換され、`hdisk2` はボリューム・グループに割り当てら れなくなります。
3. 障害のある PV を定義解除するには、次のコマンドを実行します。

   ```
   rmdev -dl hdisk2
   ```
4. 障害のあるディスクをシステムから物理的に除去します。
5. 以下のステップを行って、手順が成功していたことを確認します。
   - すべての論理ボリュームが新規 PV に希望どおりにミラーリングされたことを検査するには、次のコマンドを実行します。

     ```
     lslv lvname
     ```

障害のある PV の影響を受けた各論理ボリュームの COPIES 属性を検査して、希望する数のコピーが現在存在することを確認します。 論理ボリュームのコピー数が希望数を下回る場合は、**mklvcopy** コマンドを使用して、追加のコピーを作成します。

- すべての論理ボリューム区画が同期化し、不整合区画がないことを確認するには、次のコマンドを実行します。

```
lspv hdisk10
```

交換 された PV の STALE PARTITIONS 属性を検査して、カウントがゼロであることを確認します。 不整合区画がある 場合は、**syncvg** コマンドを使用して、区画を同期化します。

ステップ 5 で、障害のある PV の交換手順は完了です。

replacepv コマンドを使用できない構成のときの障害のある PV の交換

障害のある物理ボリューム、hdisk0 と そのミラー、hdisk1 が、*yourvg* ボリューム・グループの一部であるとします。

1. 障害のある PV からミラー・コピーを除去するには、次のコマンドを実行します。

```
unmirrorvg yourvg hdisk0
```

2. PV の障害が rootvg で発生した場合は、次のコマンドを実行して、hdisk0 をブート・リストから除去します。

   注 : 構成が hdisk0 および hdisk1 以外のブート・デバイスを使用している場合は、それをコマンド構 文に追加します。

```
bootlist -om normal hdisk1
```

   このステップの場合、hdisk1 はブート可能デバイスを rootvg のままにしておく 必要があります。 このステップの完了後、hdisk0 が出力に表示されないことを確認します。

3. PV の障害が rootvg で発生した場合は、障害のある PV からすべての専用ダンプ・デバイスを再作成します。

   障害のある PV 上にあった専用ダンプ・デバイスがある場合 は、**mklv** コマンドを使用して、既存の PV 上に新規論理ボリュームを作成できます。 **sysdumpdev** コマンドを使用して、新規論理ボリュー ムを 1 次ダンプ・デバイスとして設定します。

4. 障害のある PV を定義解除するには、次のコマンドを実行します。

   注 : 障害のある PV がシステムのブートに使用される PV である場合は、ディスク装置エントリーを 除去すると、/dev/ipldevice ハード・リンクも除去されます。

```
reducevg yourvg hdisk0
rmdev -dl hdisk0
```

5. 障害のある PV が最近使用されたブート・デバイスの場合は、次のコマンドを実行して、ステップ 4 で除去された /dev/ipldevice ハード・リンクを再作成します。

```
ln /dev/rhdisk1 /dev/ipldevice
```

   PV 名の前のプレフィックス r に注意して ください。

   /dev/ipldevice ハード・リンクが再作成されたことを確認するには、次のコマンドを実行します。

```
ls /dev/ipldevice
```

6. 障害のあるディスクを交換します。
7. 新規 PV を定義するには、次のコマンドを実行します。

```
cfgmgr
```

cfgmgr コマンドは、交換 PV に PV 名を割り当てます。 割り当て られた PV 名は、これまで障害のある PV に割り当てられていた PV 名と同じであると思われます。 この例では、デバイス hdisk0 が 交換 PV に割り当てられているとします。

8. 新規 PV をボリューム・グループに追加するには、次のコマンドを実行します。

```
extendvg yourvg hdisk0
```

以下のエラー・メッセージを受けることがあります。

```
0516-050 Not enough descriptor space left in this volume group.
(このボリュームには十分なディスクリプター・スペースが残っていません。)
Either try adding a smaller PV or use another volume group.
(より小さい PV を追加するか、別のボリューム・グループを使用してみてください。)
```

このエラーを受けて PV をボリューム・グループに追加できない場合は、論理ボリュームをボリューム・グループに既に存在する別の PV にミラーリングするか、あるいはより小さい PV の追加を試みることができます。 いずれのオプションも不可能な場合 は、**chvg** コマンドを使用して、ボリューム・グループをビッグ・タイプ・ボリューム・グループまたはスケーラブル・タイプ・ボリューム・グループにアップグレード して、この制限をバイパスしてみることもできます。

9. ボリューム・グループをミラーリングします。

注 : mirrorvg コマンドは、以下の条件がすべて存在する場合は使用できません。

- ターゲット・システムが論理区画 (LPAR) である。
- ブート論理ボリューム (デフォルトでは、hd5) のコピーが、障害のある PV 上にある。
- 最後のコールド・ブート以降に、交換 PV のアダプターが LPAR に動的に構成された。

上の条件がすべて満たされている場合、以下のようにして、**mklvcopy** コマンドで論理ボリュームごとにミラー・コピーを再作成してください。

a. ブート論理ボリュームのコピーを作成して、それが一連の隣接する物理区画に割り振られてい るか確認します。
b. 残りの論理ボリュームのコピーを作成し、**syncvg** コマンドを使用してコピーを同期化します。
c. シャットダウンまたはリブート・コマンドを使用してリブートするのではなく、LPAR をシャ ットダウンし、それを活動化して、ディスクをブート可能にします。 このシャットダウンは即 時に行う必要はありませんが、システムは新規 PV からブートする必要があります。

そうでない場合は、新規 PV を次のコマンドとともに使用して、 ボリューム・グループ内に、論理ボリュームの新規コピーを作成します。

注 : mirrorvg コマンドは、デフォルトでクォーラムを使用不可にします。 rootvg の場合 は、**-m** オプションを使用して、新規論理ボリュームのコピーが作業ディスクと同じ方法で hdisk0 にマップされることを確認します。

```
mirrorvg yourvg hdisk0
```

10. ご使用の構成が一部の論理ボリュームのコピーを保持している場合は、次のコマンドによっ て、これらのコピーを再作成する必要がある場合があります。

```
mklvcopy -k
```

11. PV の障害が rootvg で発生した場合は、次のコマンドを実行して、ブート・レコードを初期化します。

```
bosboot -a
```

12. PV の障害が rootvg で発生した場合は、次のコマンドを実行して、ブート・リストを更新します。

注 : ご使用の構成が、hdisk0 および hdisk1 以外の ブート・デバイスを使用している場合は、それをコマンドに追加します。

```
bootlist -om normal hdisk0 hdisk1
```

13. 手順が成功したことを確認します。
    - すべての論理ボリュームが新規 PV にミラーリングされたことを検査するには、次のコマンドを実行します。

      ```
      lslv lvname
      ```

      障害のある PV の影響を受けた各論理ボリュームの COPIES 属性を検査して、希望する数のコピーが現在存在することを確認します。 論理ボリュームのコピー数が希望数を下回る場合は、**mklvcopy** コマンドを使用して、追加のコピーを作成します。
    - すべての論理ボリューム区画が同期化されていることを確認するには、次のコマンドを実行して、不整合区画がないことを検査します。

      ```
      lspv hdisk0
      ```

      交換 された PV の STALE PARTITIONS 属性を検査して、カウントがゼロであることを確認します。 不整合区画がある場合は、**syncvg** コマンドを使用して、区画を同期化します。

    PV の障害が rootvg で発生した場合は、次のステップを使用して、 この手順のその他の局面を確認します。
    - ブート・リストを確認するには、次のコマンドを実行します。

      ```
      bootlist -om normal
      ```
    - ダンプ・デバイスを確認するには、次のコマンドを実行します。

      ```
      sysdumpdev -l
      ```
    - ブート可能 PV のリストを確認するには、次のコマンドを実行します。

      ```
      ipl_varyon -i
      ```
    - /dev/ipl_device を確認するには、次のコマンドを実行します。

      ```
      ls -i /dev/rhdisk1 /dev/ipldevice
      ```

      ls コマンドの 出力の i ノード番号が、両方エントリーで同じであることを確認します。

    このステップで手順は完了します。

関連情報

Logical Volume Manager from A to Z: Introduction and Concepts

物理ボリュームが見つからないときの管理者への通知

AIX は、物理ボリュームがアクセス不能なときにエラーをログに記録しますが、エラーが検知不能になることがある環境があります。

例えば、物理ボリュームが、ミラーリングされるボリューム・グループの一部のときは、データの有効なコ ピーにはアクセスできているので、ユーザーは問題に気付きません。 このような場合は、ユーザーが自分た ちの作業に悪影響が出ていることに気付く前に、自動通知によってその問題のアラートを管理者に出すこと ができます。

以下の手順では、物理ボリュームが見つからないと宣言されたときの自動通知をセットアップする 方法を示します。 他の重大なエラーについても、 以下の手順を変更することによって追跡することができます。

ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

1. root 権限で、/etc/objrepos/errnotify ODM ファイルのバックアップ・コピーを作成します。

このバックアップ・コピーには任意の名前を付けることができます。次の例では、バックアップ・コピーに errnotify ファイル名と現在日付を付加しています。

```
cd /etc/objrepos
cp errnotify errnotify_current_date
```

2. 使い慣れたエディターで、次のスタンザを含む /tmp/pvmiss.add というファイルを作成します。

```
errnotify:
    en_pid = 0
    en_name = "LVM_SA_PVMISS"
    en_persistenceflg = 1
    en_label = "LVM_SA_PVMISS"
    en_crcid = 0
    en_type = "UNKN"
    en_alertflg = ""
    en_resource = "LVDD"
    en_rtype = "NONE"
    en_rclass = "NONE"
    en_method = "/usr/lib/ras/pvmiss.notify $1 $2 $3 $4 $5 $6 $7 $8 $9"
```

このトピックのすべてのステップを完了したら、エラー通知デーモンがこのスクリプトの $1 から $9 を自 動的に展開し、通知メッセージの中にエラー・ログ・エントリーの詳細情報を含めます。

3. 使い慣れたエディターで、以下の内容の /usr/lib/ras/pvmiss.notify という名前のファイルを作成します。

```
#!/bin/ksh
exec 3>/dev/console
print -u3 "?"
print -u3 - "-----------------------------------------------"
print -u3 "ALERT!  ALERT!  ALERT!  ALERT!  ALERT!  ALERT!"
print -u3 ""
print -u3 "Desc: PHYSICAL VOLUME IS MISSING. SEE ERRPT."
print -u3 ""
print -u3 "Error label: $9"
print -u3 "Sequence number: $1"
print -u3 "Error ID: $2"
print -u3 "Error class: $3"
print -u3 "Error type: $4"
print -u3 "Resource name: $6"
print -u3 "Resource type: $7"
print -u3 "Resource class: $8"
print -u3 - "-----------------------------------------------"
print -u3 "?"
mail - "PHSYICAL VOLUME DECLARED MISSING" root <<-EOF
-------------------------------------------
ALERT!  ALERT!  ALERT!  ALERT!  ALERT!  ALERT!
Desc: PHYSICAL VOLUME IS MISSING. SEE ERRPT.
Error label: $9
Sequence number: $1
Error ID: $2
Error class: $3
Error type: $4
Resource name: $6
Resource type: $7
Resource class: $8
-------------------------------------------
EOF
```

4. ファイルを保存し、エディターを終了します。
5. 作成したファイルに適切な許可を設定します。
   次に例を示します。

```
chmod 755 /usr/lib/ras/pvmiss.notify
```

6. 次のコマンドを入力して、ステップ 2 で作成した LVM_SA_PVMISS 定義を ODM に追加します。

```
odmadd /tmp/pvmiss.add
```

これで、LVM_SA_PVMISS エラーが発生すると、システムで /usr/lib/ras/pvmiss.notify スクリプトが実行されます。このスクリプトによって コンソールにメッセージが送信され、root ユーザーにメールも送信されます。

関連概念
論理ボリューム・ストレージ
論理ボリュームは、物理ボリューム上に存在する情報のグループです。

関連情報
odmadd コマンド

ミラーリングしたディスクのボリューム・グループからの分割
ミラーリングされたボリューム・グループの一貫性が、 潜在的なディスク障害のために損なわれるのを防ぐ上で、スナップショットのサポートが役立ちます。

スナップショット・フィーチャーを使用すると、ミラーリングした 1 つ、または複数のディスクを分割して、ボリューム・グループの信頼できる (LVM メタデータの観点から) 時刻指定バックアップとして使用 できます。また、必要な場合は、その分割したディスクをボリューム・グループに確実に再統合できます。以下の手順では、始めに、ボリューム・グループからミラーリングしたディスクを 分割し、次に、元のボリューム・グループに分割したディスクをマージします。 スナップショットの信頼性をさらに確実にするに は、ファイルシステムをアンマウントし、ロー論理ボリュームを使用するアプリケーションを既知の状態 (バックアップを使用する必要がある場合に、アプリケーションをそれからリカバリーできる状態) に しておく必要があります。

以下のいずれかが該当する場合には、ボリューム・グループは分割できません。

- ディスクが既に存在しない。
- 不整合でない最後の区画が分割したボリューム・グループ上に存在する。
- **splitvg** コマンドに強制フラグ (**-f**) を使用しない限り、不整合区画がボリューム・グループに存在する。

さらに、 スナップショット・フィーチャー (特に、**splitvg** コマンド) は、拡張コンカレント・モードまたは標準コンカレント・モードで使用できません。 分割した ボリューム・グループをコンカレントまたは拡張コンカレントの状態にすることはできません。また、分割ボリューム・グループおよび元のボリ ューム・グループの両方に対して、許容される変更の制限があります。 詳細は、**chvg** コマンドの説明をお読みください。

ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

1. ボリューム・グループが完全にミラーリングされていること、およびミラーが、このミラー・セットだけを含む 1 つまたは複数のディスクに存在することを確認します。
2. スナップショット・サポートを使用できるようにするため、次のコマンドを使用して、元のボリューム・グループ (`origVG`) を別の 1 つまたは複数の ディスクに分割します。

   ```
   splitvg origVG
   ```

   この時点で、元のボリューム・グループの信頼できる時刻指定バックアップができます。 ただし、分割したボリューム・グループ上の割り振りを変更できないことに注意してください。
3. 次のコマンドを使用して、分割したディスクを再びアクティブにし、元のボリューム・グループにマージします。

   ```
   joinvg origVG
   ```

   この時点で、分割したボリューム・グループが元のボリューム・グループに再統合されます。

関連概念
論理ボリューム・ストレージ
論理ボリュームは、物理ボリューム上に存在する情報のグループです。

関連情報
chvg コマンド
recreatevg コマンド
splitvg コマンド
Logical Volume Manager from A to Z: Introduction and Concepts

トラブルシューティング LVM

トラブルシューティングできる LVM の問題には、いくつかの共通タイプがあります。

ディスク・ドライブの問題のトラブルシューティング

この情報で、ディスク・ドライブ問題の診断および修正方法を指示します。

ディスク・ドライブの機構に障害があると疑われる場合は、以下の手順を実行して、ディスクの診断を行います。

1. root 権限で、コマンド・ラインに次の SMIT 高速パスを入力します。

```
smit diag
```

2.「**Current Shell Diagnostics (最新のシェル診断)**」を選択して、AIX Diagnostics (AIX 診断) ツールに入ります。
3.「Diagnostics Operating Instructions (診断操作指示)」画面を読んでから、Enter キーを押します。
4.「**Diagnostics Routines (診断ルーチン)**」を選択します。
5.「**System Verification (システム検査)**」を選択します。
6. リストをスクロールダウンし、テストしたいドライブを見付け、選択します。
7.「**Commit (コミット)**」を選択します。

診断結果に基づいて、ディスクの状態を判断できます。

- ディスク・ドライブに障害が発生していることが分かった場合は、ディスクからデータをリカバリーすることが一番大事なことです。 障害のあるディスクからデータをリカバリーする 方法として、移行をお勧めします。 以下の手順で、移行を正常に完了できないときに論理ボリュームの データをリカバリーまたは復元する方法について説明します。
- ドライブの障害時に、再フォーマットを行わずにドライブを修復できるのであれば、データは破損しません。
- ディスク・ドライブの再フォーマット、または置換が必要な場合は、置換する前に可能であればバックアップを作成し、ディスク・ドライブをボリューム・グループおよびシステム 構成から除去しておいてください。 単一コピー・ファイルシステムではデータが削除されることがあります。

ディスク・ドライブのスペース

ディスク・ドライブのスペースが不足した場合は、いくつかの方法で問題を解決することができます。 不必要なファイルを自動的に追跡して除去したり、特定のディレクトリーのユーザーによる使用を制限したり、または別のディスク・ドライブからスペースをマウントすることができます。

これらの作業を実行するには、root ユーザー、システム・グループ、または管理グループの権限が必要です。

ファイルシステムを自動的にクリーンアップするコマンド

不必要なファイルを除去してファイルシステムをクリーンアップするには 、**skulker** コマンドを使用します。

コマンド・ラインから次のように入力します。

```
skulker -p
```

skulker コマンドは、使用されていない不必要なファイルを、ファイルシステムから 定期的に除去するために使用します。 対象となるファイルは、/tmp ディレクトリー内のファイル、指定した日付よりも古いファイル、a.out ファイル、コア・ファイル、ed.hup ファイル などです。 **skulker** コマンドの詳細については、**skulker** を参照してください。

通常、**skulker** コマンドは、オフピーク時間帯に、**cron** コマンドが実行する アカウンティング手順の一部として毎日実行されます。

特定のディレクトリーに対するユーザーの制限

ディレクトリーへのアクセスを 制限し、ディスク使用率をモニターすることによって、ディスク・スペースを解放し、かつそれを空いたままにしておくことができるものと思われます。

1. 次のように入力して、特定のディレクトリーにアクセスできるユーザーを制限します。

   ```
   chmod 755 DirName
   ```

   このコマンドは、オーナー (root) に読み取り許可と書き込み許可を設定し、グループとその他に対して読み取り専用許可を設定します。 *DirName* は制限するディレクトリーの絶対パス名です。
2. 次の行を **/var/spool/cron/crontabs/adm** ファイルに追加することにより、個々のユーザーのディスク使用量をモニターします。

   ```
   0 2 * * 4 /usr/sbin/acct/dodisk
   ```

   この行で、**dodisk** コマンドが毎週木曜日 (4) の午前 2 時 (0 2) に実行されます。**dodisk** コマンドはディスク使用アカウンティングを開始します。このコマンドは通常、オフピーク時間帯に **cron** コマンドが実行するアカウンティング手順の一部として実行されます。

別のディスク・ドライブからのスペースのマウント
別のディスク・ドライブからのスペースをマウントすることによって、獲得するディスク・ドライブのスペースを増やせます。

以下の方法で、あるディスク・ドライブから別のディスク・ドライブにスペースを マウントすることができます。

- **smit mountfs** 高速パスを使用する。
- **mount** コマンドを使用する。 例:

  ```
  mount -n nodeA -vnfs /usr/spool /usr/myspool
  ```

  mount コマンドは、ファイルシステムを特定の場所で使用できるようにします。

再フォーマット設定なしのディスク・ドライブのリカバリー
障害のあるディスクを修復し、再フォーマットを行わずにこのディスクをシステムに 戻した場合には、ブート時に、システムに自動的に失効した物理区画を活動化させ、再同期化させることが可能です。 失効した物理区画 にはシステムが使用できないデータが入っています。

失効した物理区画が存在する疑いのある場合は、コマンド・ラインに次のように入力します。

```
lspv -M PhysVolName
```

ここで *PhysVolName* は物理ボリュームの名前です。 **lspv** のコマンド出力には、物理ボリューム上のすべての区画がリストされます。 以下に出力例の一部を示します。

```
hdisk16:112     lv01:4:2        stale
hdisk16:113     lv01:5:2        stale
hdisk16:114     lv01:6:2        stale
hdisk16:115     lv01:7:2        stale
hdisk16:116     lv01:8:2        stale
hdisk16:117     lv01:9:2        stale
hdisk16:118     lv01:10:2       stale
```

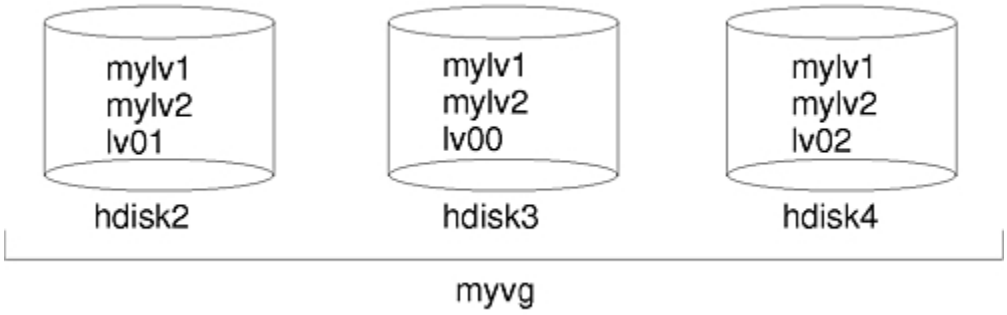
最初のカラムは物理区画を示し、2 番目のカラムは論理区画を示します。 失効した物理区画は 3 番目のカラムに表示がでます。

ディスク・ドライブを再フォーマット、または置換したリカバリー
再フォーマットまたは置換しなければならない障害のあるディスクから、データをリカバリーできます。

重要 : ディスク・ドライブの再フォーマットまたは置換を行う前に、障害のあるディスクからミラー化されていないファイルシステムへの 参照をすべて除去し、ボリューム・グループおよびシステム構成からディスクを除去します。行わないと、ODM (オブジェクト・データ・マネージャー) およびシステム 構成データベースで問題が発生します。 これらの重要なステップについての説明は、障害のあるディスクの交換または再フォーマット 設定前のもとの以下の手順に記載されています。

以下の手順では、*myvg* と呼ばれるボリューム・グループに、*hdisk2*、 *hdisk3*、および *hdisk4* と呼ばれる 3 つのディスク・ドライブが含まれるシナリオを使用します。 このシナリオでは *hdisk3* に障害が発生しま

す。 *hdisk2* にはミラー化 されていない論理ボリューム *lv01* および *mylv* 論理ボリューム のコピーが含まれています。 *mylv* 論理ボリュームはミラー化されており、3 つのコピーがあります。それぞれがディスクの物理区画を 2 つ使用します。 障害の発生する *hdisk3* には、*mylv* の 2 番目のコピー、および *lv00* と呼ばれるミラー化されていない論理ボリュームがあります。 最後に、*hdisk4* は、*mylv* の 3 番目のコピーと、*lv02* と呼ばれる論理ボリュームを含みます。 次の図は、このシナリオを示しています。



この手順は、以下の主要部分に分かれています。

- 障害のあるディスクを置換または再フォーマットする 前に、データを保護するために行う手順
- ディスクの置換または再フォーマットのために行う手順
- ディスクの置換または再フォーマット後の、データをリカバリーするために行う手順

障害のあるディスクを置換または再フォーマットする 前の手順:

1. root 権限でログインします。
2. 障害のあるドライブ上の論理ボリュームについて詳しく知らない場合は、作動中のディスクを使用して、障害のあるディスクの内容を表示します。
   例えば、*hdisk4* を使用して *hdisk3* を調べるには、コマンド・ラインに次のように入力します。

   ```
   lspv -M -n hdisk4 hdisk3
   ```

   lspv コマンドは、ボリューム・グループ内の物理ボリュームについての情報を表示します。 次のような出力が表示されます。

   ```
   hdisk3:1        mylv:1
   hdisk3:2        mylv:2
   hdisk3:3        lv00:1
   hdisk3:4-50
   ```

   最初のカラムは物理区画を示し、2 番目のカラムは論理区画を示します。 4 から 50 までの区画は空いています。
3. 可能であれば、故障したデバイス上にある単一コピーの全論理ボリュームのバックアップをとってください。 説明については、『ユーザー・ファイルまたはファイルシステムのバックアップ』を参照してください。
4. 単一コピー・ファイルシステムがある 場合は、ディスクからアンマウントしてください。
   (**lspv** コマンドの出力から単一コピー・ファイルシステムを 識別できます。 単一コピー・ファイルシステムとは、出力の中で、論理区画の数と物理区画の数が同じになっているファイルシステムです。) ミラー化されているファイルシステムをアンマウントする 必要はありません。

   シナリオでは、故障したディスク `hdisk3` 上の `lv00` は単一コピー・ファイルシステムです。 これをアンマウントするには、次のように入力します。

   ```
   unmount /dev/lv00
   ```

   ファイルシステムの 名前が分からない場合は、障害のあるディスクに `/etc/filesystems` ファイルが単独では存在しないものと想定し、コマンド・ラインに `mount` を入力し、マウント済みのすべてのファイルシステムをリストして、論理ボリュームに関連した名前を見つけます。`/etc/filesystems`

ファイルに **grep** コマンドを使用して、存在するなら、論理ボリュームに関連したファイルシステム名だけをリストすることもできます。 次に例を示します。

```
grep lv00 /etc/filesystems
```

出力は、次の例のようになります。

```
dev             = /dev/lv00
log             = /dev/loglv00
```

注意 :

a. アンマウントしようとしているファイルシステムが 使用中の場合、**unmount** コマンドは失敗します。**unmount** コマンドを実行できるのは、オープンしているファイルシステムのファイルがなく、ユーザーの現行ディレクトリーがそのデバイス 上にない場合だけです。

b. **unmount** コマンドの別の名前は **umount** です。 どちらの名前も使用することができます。

5. 次のような **rmfs** コマンドを入力し、故障した物理ボリュームから単一コピー・ファイルシステムをすべて除去します。

```
rmfs /FSname
```

6. 障害のあるディスク上のミラーリングされている論理ボリュームをすべて除去します。

注 : rootvg ボリューム・グループの中の物理ボリュームの hd5 と hd7 の論理ボリュームに対して、**rmlvcopy** を使用できません。 これらのコピーは 1 つしかないため、システムは、これらの論理ボリュームの除去を許可しません。

rmlvcopy コマンドは、各論理区画からコピーを除去します。 例えば、次のように入力します。

```
rmlvcopy mylv 2 hdisk3
```

hdisk3 上のコピーを除去することにより、*mylv* 論理ボリュームに属する各論理区画のコピーの数を、3 から 2 (1 つは *hdisk4* 上の、もう 1 つは *hdisk2* 上の) に減らすことができます。

7. 障害のあるディスクがルート・ボリューム・グループの一部であり、論理ボリューム hd7 を含んでいる場合は、コマンド・ラインに次のように入力して、1 次ダンプ・デバイス (hd7) を除去します。

```
sysdumpdev -P -p /dev/sysdumpnull
```

sysdumpdev コマンドは、稼働中のシステムの 1 次ダンプ・デバイスまたは 2 次ダンプ・デバイスの位置を変更します。 リブートが行われると、ダンプ・デバイスは元の位置に戻ります。

注 : DVD デバイスにダンプすることを選択できるようになりました。DVD をダンプ・デバイスに構成する方法について詳しくは、**sysdumpdev** を参照してください。

8. 次のコマンドを使用して、ディスク上のすべてのページング・スペースを除去します。

```
rmps PSname
```

ここで *PSname* は、削除されるページング・スペースの名前 (実際には、ページング・スペースが存在する論理ボリュームの名前) です。

rmps コマンドが成功しない場合は、**smit chps** 高速パスを使用して 1 次ページング・スペースを非活動化し、リブートしてから、この手順を続ける必要があります。活動中のページング・スペースが存在する 場合は、ステップ 10 の **reducevg** コマンドが失敗する場合があります。

9. **rmlv** コマンドを使用して、ファイルシステムを 含まない論理ボリュームなど、他のすべての論理ボリュームをボリューム・グループから除去します。
例えば、次のように入力します。

```
rmlv -f lv00
```

10. **reducevg** コマンド を使用して、ボリューム・グループから障害のあるディスクを取り外します。

例えば、次のように入力します。

```
reducevg -df myvg hdisk3
```

reducevg コマンドを実行できないか、このコマンドが成功しない場合は、ドライブを再フォーマットまたは置換した後に、ステップ 13 の 手順を使用すると、VGDA/ODM 情報をクリーンアップすることができます。

障害のあるディスクを置換または再フォーマットする 手順

11. 次のステップは、ディスクを再フォーマットするのか、置換するのかによって、また使用するハードウェアのタイプによっても異なります。
    - ディスク・ドライブを再フォーマットしたい場合は、以下の手順を実行します。
      a. root 権限で、コマンド・ラインに次の SMIT 高速パスを入力します。

      ```
      smit diag
      ```

      b.「**Current Shell Diagnostics (最新のシェル診断)**」を選択して、AIX Diagnostics tool (診断ツール) に入ります。
      c.「**Diagnostics Operating Instructions (診断操作指示)**」画面を読んでから、Enter キーを押します。
      d.「**Task Selection (タスク選択)**」を選択します。
      e. タスク・リストをスクロールダウンし、「**Format Media (フォーマット・メディア)**」を見付け、選択します。
      f. 再フォーマットしたいディスクを 選択します。 ディスクの再フォーマットを確認してから、ディスク上のすべての内容が消去されます。

      ディスクの再フォーマットが終わったら、ステップ 12 から続けます。
    - システムでホット・スワップ・ディスクがサポートされている場合は、31 ページの『システムを使用可能にしたままのディスク障害からのリカバリー』の手順を実行し、ステップ 13 から続けます。
    - システムでホット・スワップ・ディスクがサポートされていない場合は、以下のステップを実行します。
      – SMIT ファスト・パス **smit rmvdsk** を使用して、古いドライブの電源をオフにします。「Database (データベース)」フィールドの KEEP 定義を No に変更します。
      – 次の作業レベルのシステム・サポートと連絡を取り、ディスク・ドライブを置換します。

障害のあるディスクを置換または再フォーマットした後の手順:

12. 15 ページの『ディスクの構成』 および 17 ページの『使用可能ディスクの物理ボリューム化』 の手順に従います。
13. ディスクのフォーマット 前に古いボリューム・グループのディスクに **reducevg** コマンド を実行できない場合は (ステップ 10)、以下の手順を使用して VGDA/ODM 情報をクリーンアップできます。
    a. ボリューム・グループが、再フォーマットが行われた 1 つのディスクのみで構成されている場合は、次のように入力します。

    ```
    exportvg VGName
    ```

    VGName はボリューム・グループの名前です。
    b. ボリューム・グループが複数のディスクから構成されている場合には、コマンド・ラインで次のように入力します。

    ```
    varyonvg VGName
    ```

    ディスクが存在しないか、または使用できないというメッセージが表示され、新しい (再フォーマットした) ディスクがリストされます。 **varyonvg** メッセージ にリストされる新規ディスクの物

理ボリューム ID (PVID) に注意してください。 これは、存在しないディスクの名前と、PVNOTFND というラベルの間にある、16 文字の文字列です。 例:

```
hdisk3 00083772caa7896e PVNOTFND
```

次のように入力してください。

```
varyonvg -f VGName
```

これにより、存在しないディスクが、PVREMOVED ラベルを付けて表示されます。 次に例を示します。

```
hdisk3 00083772caa7896e PVREMOVED
```

次に次のコマンドを入力します。

```
reducevg -df VGName PVID
```

PVID は物理ボリューム ID (このシナリオでは 00083772caa7896e) です。

14. 新しいディスク・ドライブをボリューム・グループに追加するために、**extendvg** コマンドを使用します。
    例えば、次のように入力します。

    ```
    extendvg myvg hdisk3
    ```

15. 新しい (再フォーマットした) ディスク・ドライブに単一コピー論理ボリュームを再作成する ために、**mklv** コマンドを使用します。
    例えば、次のように入力します。

    ```
    mklv -y lv00 myvg 1 hdisk3
    ```

    この例では、lv00 論理ボリュームが、*hdisk3* ドライブ上に再作成されます。1 は、論理ボリュームがミラー化されないことを意味します。
16. 論理ボリュームにファイルシステムを 再作成するために、**crfs** コマンドを実行します。
    例えば、次のように入力します。

    ```
    crfs -v jfs -d lv00 -m /dev/lv00
    ```

17. 単一コピー・ファイルシステム・データをバックアップ・メディアからリストアするには、『ユーザー・ファイルをバックアップ・イメージからリストア』を参照してください。
18. 論理ボリュームのミラー化されたコピーを再作成するために、**mklvcopy** コマンドを実行します。
    例えば、次のように入力します。

    ```
    mklvcopy mylv 3 hdisk3
    ```

    この例では、*mylv* 論理ボリュームのミラー化された 3 番目の 区画を *hdisk3* 上に作成します。
19. 新しいミラーを、他のミラーのデータ (この例では *hdisk2* と *hdisk4* にある) と同期化 させるために、**syncvg** コマンドを実行します。
    例えば、次のように入力します。

    ```
    syncvg -p hdisk3
    ```

結果として、ミラーリングされたすべてのファイルシステムが 復元され、最新の状態になります。単一コピー・ファイルシステムをバックアップしておいた 場合には、これらも使用できる状態になります。 これでシステムを通常どおり使用できるようになります。

障害のあるディスク・ドライブからのリカバリーの例

ディスク・ドライブの障害からリカバリーするには、たどってきた道を戻る必要があります。つまり、ボリューム・グループを作成するためのステップをリストし、これを戻っていきます。

次の例は、この技法を示すものです。 ここでは、ミラー化された論理ボリュームの作成方法を示したあと、ディスクの障害時に、1 ステップずつ戻りながら、これを変更する方法を示しています。

注 : 以下の例は特別な場合を示しています。 これは、一般的なリカバリー手順の普遍的なプロトタイプとして作成されたものではありません。

1. システム・マネージャーの Jane が次のように入力して、**workvg** という 名前のボリューム・グループを hdisk1 上に作成しました。

   ```
   mkvg -y workvg hdisk1
   ```

2. 次のように入力して、このボリューム・グループにもう 2 つディスクを作成しました。

   ```
   extendvg workvg hdisk2
   extendvg workvg hdisk3
   ```

3. 3 つのコピーをもつ 40 MB の論理ボリュームを作成しました。

   各コピーは、**workvg** を構成する 3 つのディスクに それぞれ置かれます。 次のコマンドを使用しました。

   ```
   mklv -y testlv workvg 10
   mklvcopy testlv 3
   ```

ミラー化した workvg ボリューム・グループを作成した後に、hdisk2 に障害が発生しました。 そこで、リカバリーするために以下のステップを実行しました。

1. 次のように入力して、hdisk2 から論理ボリューム・コピーを除去しました。

   ```
   rmlvcopy testlv 2 hdisk2
   ```

2. 次のように入力して、ODM および VGDA を更新できるように、システムから hdisk2 を切り離しました。

   ```
   reducevg workvg hdisk2
   ```

3. 次のように入力して、置換の準備のために、システム 構成から hdisk2 を除去しました。

   ```
   rmdev -l hdisk2 -d
   ```

4. 次のように入力して、システムのシャットダウンを行いました。

   ```
   shutdown -F
   ```

5. ディスクを交換しました。 新規ディスクでは、以前の hdisk2 とは異なる SCSI ID が使用されていました。
6. システムをリブートしました。

   新しいディスクが使用されているため (システムはこのディスクに 新しい PVID があることを 認識します)、システムは空いている 最初の hdisk 名を選択します。 ステップ 3 で **-d** フラグを使用したため、hdisk2 という名前が空き、このため、システムは hdisk2 を新規ディスクの名前として選択しました。 **-d** フラグを使用しなかった場合は、新しい名前として hdisk4 が選択されることになります。
7. 次のように入力して、このディスクを **workvg** ボリューム・グループ に追加しました。

   ```
   extendvg workvg hdisk2
   ```

8. 次のように入力して、論理ボリュームのミラー化されたコピーを 2 つ作成しました。

   ```
   mklvcopy testlv 3
   ```

論理ボリューム・マネージャーが自動的に、論理ボリュームの 3 番目のコピーを新しい hdisk2 に置きました。

システムを使用可能にしたままのディスク障害からのリカバリー

ディスク障害は、常時取り外し可能フィーチャーを使用してリカバリーできます。

常時取り外し可能フィーチャーを使用したディスク障害からのリカバリーの手順は、そのほとんどが、25 ページの『再フォーマット設定なしのディスク・ドライブのリカバリー』での説明と同じです。異なる点を以下に示します。

1. ディスク上のファイルシステムをアンマウントするには、JFS または JFS2 のマウントの手順を使用します。
2. ディスクを、そのボリューム・グループおよびオペレーティング・システムから除去するには、48 ページの『データのないディスクの除去』手順を使用します。
3. 障害のあるディスクを新しいディスクと取り替える場合は、システムをシャットダウンする必要はありません。 以下の手順を順番に実行します。
   a) 35 ページの『論理ボリューム・ストレージ』
   b) 15 ページの『ディスクの構成』
   c) 25 ページの『ディスク・ドライブを再フォーマット、または置換したリカバリー』のステップ 13 に進みます。

ボリューム・グループが *1* つのディスクから構成されるときのディスクの交換

ボリューム・グループの一部として不良になるディスクにアクセスできる場合は、以下の手順を使用します。

- 13 ページの『物理ボリュームの内容の移行』

ディスクが不良で、そのディスクにアクセスできないときは、以下のステップを実行してください。

1. ボリューム・グループをエクスポートします。
2. ドライブを取り替えます。
3. バックアップ・メディアからデータを再作成します。

物理および論理ボリューム・エラー

トラブルシューティングできる物理ボリュームおよび論理ボリュームには、いくつかの共通するエラーがあります。

ホット・スポット問題

論理ボリュームへアクセス中にパフォーマンスの低下に気付いた場合は、論理ボリュームのホット・スポットのディスク入出力が多すぎる可能性があります。

詳細については60 ページの『論理ボリュームのホット・スポット管理』を参照してください。

LVCB の警告

LVCB に無効な情報が含まれていると、警告が生じます。

論理ボリューム制御ブロック (LVCB) は、論理ボリュームの最初のブロックです。 LVCB のサイズは、ボリューム・グループ内の物理ボリュームのブロック・サイズです。 この領域には、論理ボリュームの作成日、ミラーリングしたコピーの情報、および JFS のマウント可能ポイントなどの、重要な情報が含まれています。 LVCB を更新するには、LVM のアルゴリズムの一部として、特定の LVM コマンドが必要です。 有効であるかどうかを確認するために、古い LVCB が読み取られ、分析されます。 その情報が有効な LVCB 情報の場合は、LVCB が更新されます。 その情報が有効でない場合は、LVCB の更新は行われず、次のメッセージが表示されます。

```
Warning, cannot write lv control block data.
```

このメッセージが表示されるのは、ほとんどの場合、データベース・プログラムが JFS をバイパス し、ストレージ・メディアとしてロー論理ボリュームにアクセスするときです。 この場合は、データベースの情報がそのまま LVCB に書き込まれます。 ロー論理ボリューム の場合は、このようになっても致命的な結果にはなりません。 LVCB が上書きされた後も、以下の作業をすることができます。

- 論理ボリュームを拡張します。
- 論理ボリュームのミラー化されたコピーを作成します。
- 論理ボリュームを取り出します。
- 論理ボリュームをマウントするジャーナル・ファイルシステムを 作成します。

LVCB の削除には制限があります。 LVCB が削除されている論理ボリュームは、他のシステムに正常にインポートできない場合があります。 インポートの際に、LVM **importvg** コマンドが、ボリューム・グループにある定義済みの すべての論理ボリュームの LVCB をスキャンし、論理ボリュームに関する情報を調べます。 LVCB が存在しなくても、インポートされたボリューム・グループは、このボリューム・グループにアクセスする新規システムに論理ボリュームを 定義しています。このため、まだ、ロー論理ボリュームにアクセスすることができます。 ただし、通常、以下のことが起こります。

- JFS 情報がすべて失われ、関連するマウント・ポイントを新規システムにインポート できない。 この場合は、新規マウント・ポイントを作成する必要があります。また、ファイルシステムに保管されていたデータを使用できなくなる可能性があります。
- 論理ボリュームに関連した一部の非 JFS 情報を見付けることができない。 この場合は、ODM 情報を読み込むためにデフォルトの論理ボリュームの情報が使用されます。 したがって、**lslv** コマンドの出力のうちで、実際の論理ボリュームと矛盾するものがでる可能性があります。 元のディスクに 論理ボリューム・コピーが残っていると、ODM データベースに情報が正しく反映されません。 **rmlvcopy** コマンドおよび **mklvcopy** コマンド を実行して、すべての論理ボリューム・コピーを再作成し、ODM を同期化します。

物理区画の制限

論理ボリューム・マネージャー (LVM) の設計では、各論理区画は 1 つの物理区画 (PP) に マップされます。さらに、各物理区画はいくつかのディスク・セクターにマップされます。 LVM の設計では、LVM が追跡できるディスクあたりの物理区画の数は 1016 に制限されています。 ほとんどの場合、1016 個の追跡区画のすべてがディスクによって 使用されるわけではありません。

この制限を超えると、以下のようなメッセージが表示されます。

```
0516-1162 extendvg: Warning, The Physical Partition Size of PPsize requires the
creation of TotalPPs partitions for PVname.  The limitation for volume group
VGname is LIMIT physical partitions per physical volume.  Use chvg command
with -t option to attempt to change the maximum Physical Partitions per
Physical volume for this volume group.
```

ここで:

PPsize
1 MB から 1 GB (2 の累乗の形式) です。

Total PPs
PPsize が与えられているときの、ディスク上の物理区画の合計数です。

PVname
物理ボリュームの名前です。hdisk3 など。

VGname
ボリューム・グループの名前です。

LIMIT
1016 または 1016 の倍数です。

以下の場合に、この制限が強制されます。

1. **mkvg** コマンドを実行してボリューム・グループを作成するときに、ボリューム・グループのディスク上に、1016 個を超す物理区画の数を指定した。 この制限を回避するには、物理区画のサイズ (MB) を 1、 2、4 (デフォルト)、8、16、32、64、128、256、512、または 1024 の範囲から選択し、**mkvg -s** コマンド を実行してボリューム・グループを作成します。 あるいは、ディスクあたりの区画数 1016 の倍数 を可能にする適切なファクターを使用し、**mkvg -t** コマンドを実行して、ボリューム・グループを作成することもできます。
2. **extendvg** コマンドを実行して、既存ボリューム・グループに ディスクを追加するときに、新規ディスクが原因となって 1016 個の制限の違反となった。 これを解決するには、**chvg -t** コマンドを実行して、

ディスクあたり区画数 1016 の倍数を保持できるように、既存ボリューム・グループを変換します。 あるいは、新規ディスクを可能にする、より大きな区画サイズをもったボリューム・グループ を再作成するか、新規ディスク用のより大きな物理サイズで構成される スタンドアロン・ボリューム・グループを作成することができます。

区画制限と rootvg

インストール・コードによって、rootvg ドライブが 4 GB より大きいことが検出されると、使用可能な 1016 本のトラックにディスク全体の容量をマップできるように、**mkvg-s** の値が変更されます。 また、このインストールの変更から、rootvg に追加されるその他のすべてのディスクも、サイズにかかわらず、この物理区画サイズで定義されることも分かります。

区画制限と RAID システム

RAID (新磁気ディスク制御機構) を使用するシステムの場合、LVM によって使用される /dev/hdiskX の名前が、4 GB でない多くのディスクで構成される可能性があります。 この場合も、1016 個の制限が存在したままになっています。 LVM は、/dev/hdiskX を実際に構成する個々のディスクのサイズを 認識しません。 LVM は、/dev/hdiskX を実際に構成する物理ディスクに対してではなく、認識した /dev/hdiskX のサイズに対して 1016 個の制限を 課します。

デバイス構成データベースの同期

システムの誤動作によって、デバイス構成データベースが LVM と矛盾するようになる 場合があります。 デバイス構成データベースは LVM 情報によって同期できます。

デバイス構成データベースが LVM と矛盾すると、論理ボリューム・コマンドは、次のようなエラー・メッセージを生成します。

```
0516-322 The Device Configuration Database is inconsistent ...
```

または

```
0516-306 Unable to find logical volume LVname in the Device
Configuration Database.
```

(この場合、*LVname* と呼ばれる論理ボリュームは、通常使用可能です。)

重要 : ボリューム・グループまたは論理ボリュームの /dev エントリーは除去しないでください。 また、オブジェクト・データ・マネージャーを使用して、ボリューム・グループまたは論理ボリュームのデータベース・エントリーを変更できません。

デバイス構成データベースを LVM 情報と同期化するには、root 権限で、コマンド・ラインに次のように入力します。

```
synclvodm -v VGName
```

VGName は、同期化したいボリューム・グループの名前です。

ボリューム・グループ・エラーの修復

ボリューム・グループ・エラーの修復には、以下の方法を使用します。

importvg コマンドが正しく機能しない場合、デバイス構成データベースをリフレッシュしてください。

Varyon 障害の上書き

重要 : varyon 障害の上書きは通常の操作ではありません。これを行う前に、ハードウェア、ケーブル、アダプター、および電源などの、考えられる その他のすべての問題源を確認してください。 varyon 処理の際のクォーラム (定足数) 障害の上書きは、緊急の場合の最後の手段 (例えば、障害のあるディスクからデータを取り出す) です。 クォーラム (定足数) 障害を上書きすると、VGDA および VGSA の選択したコピーに含まれる管理データ のデータの保全性を保証できなくなります。

クォーラム (定足数) の不在を上書きすることにより、ボリューム・グループを強制的に varyon すると、この varyon 処理中に存在していないすべての物理ボリュームの PV STATE が、removed に変更されます。 これは、VGDA および VGSA のすべてのコピーが物理ボリュームから 除去されることを意味します。 この処

理の後では、クォーラム (定足数) の確認でこれらの物理ボリュームは対象外になります。また、ボリューム・グループに戻すまでは、これらの物理ボリュームをボリューム・グループ内で 活動化できなくなります。 varyonvg -f フラグ (クォーラム・ロスをオーバーライドするために使用される) は、ボリューム・グループがクォーラムを失っていない場合、 無視されます。

次の条件のどちらかまたは両方に該当する場合は、ボリューム・グループの使用可能ディスク上 のデータにアクセスできるようにするため、varyon 障害を上書きすることをお勧めします。

- 使用不可物理ボリュームの損傷が永続的なものと思われる。
- ボリューム・グループを最後にオンに変更したときに、現在アクセス可能な 物理ボリュームの少なくとも 1 つ (VGDA および VGSA の障害のないコピーも 入っている 必要がある) がオンラインの状態だったことを確認できる。 存在しない物理ボリュームを診断し、修復できるようになるまで、構成解除し、電源オフしてください。

1 つのディスクが存在しないか、またはディスクに障害が発生し、修復が必要となる 可能性がある場合には、クォーラム (定足数) がなくならないようにするために、以下の手順を実行します。

1. ボリューム・グループからそのボリュームを一時的に除去するために、次のように入力します。

   ```
   chpv -vr PVname
   ```

   このコマンドが完了すると、物理ボリューム *PVname* は、クォーラム (定足数) 検査の要素にはならなくなります。 ただし、2 つのディスクのボリューム・グループ の場合、2 つの VGDA/VGSA が入ったディスクに **chpv** コマンドを 実行すると、このコマンドは失敗します。 このコマンドを使用して、クォーラム (定足数) が失われるようにすることはできません。
2. 修復のためにディスクを取り外す必要がある場合は、システムを電源オフし、ディスクを取り外します。 (詳細については、24 ページの『ディスク・ドライブの問題のトラブルシューティング』を参照してください。) ディスクを修復し、システムにディスクを 戻した後、次のステップから継続します。
3. クォーラム (定足数) 検査のために、ボリューム・グループにとって、もう一度ディスクが使用可能になるためには、次のように入力します。

   ```
   chpv -v a PVname
   ```

   注 : chpv コマンドは、クォーラム (定足数) 検査による変更のためだけに使用します。 ディスク上のデータはディスクに残ったままになるので、ディスクをシステムに 戻さない場合は、そのデータを他のディスクに移動またはコピーする 必要があります。

VGDA の警告

既存ボリューム・グループに新規ディスクを追加するとき、または新規ボリューム・グループを作成するときに、問題が発生する場合があります。 LVM によって次のようなメッセージが出されます。

```
0516-1163 extendvg: VGname already has maximum physical volumes.  With the maximum
number of physical partitions per physical volume being LIMIT, the maximum
number of physical volumes for volume group VGname is MaxDisks.
```

ここで:

VGname
ボリューム・グループの名前です。

LIMIT
1016 または 1016 の倍数です。

MaxDisks
ボリューム・グループ内のディスクの最大数です。 例えば、ディスクあたりに 1016 個の 物理区画 (PP) がある場合は、*MaxDisk* は 32 です。2032 個が ある場合は、*MaxDisk* は 16 です。

image.data ファイルを変更し、「alternate disk installation (代替ディスクのインストール)」を使用するか、**mksysb** コマンドを使用してシステムを復元し、ボリューム・グループを 大きなボリューム・グループとして再作成することができます。 詳細については、「インストールおよび移行」を参照してください。

制限が 32 個のディスクより少なかった旧式の AIX の **rootvg** には、最大 VGDA に関するこの説明は当てはまりませんでした。 **mkvg -d** コマンドを使用すると、インストール・メニューで選択した ディスク数が

参照数として使用されました。このため、rootvg を作成したときに、より多くの空きディスク・スペースを確保することができました。 この **-d** の数は 1 個のディスクの場合 には 7 であり、追加ディスクを選択するたびに 1 つずつ増えます。 例えば、2 個のディスクを選択すると、数は 8 になり、3 個のディスクを選択すると、数は 9 になります。

論理ボリューム・ストレージ

論理ボリュームは、物理ボリューム上に存在する情報のグループです。

ディスク装置の管理には、構造階層を使用します。 物理ボリューム (PV) と呼ばれるディスク・ドライブには、個々それぞれに、`/dev/hdisk0` のような名前があります。 すべての使用中の物理ボリュームはボリューム・グループ (VG) に属します。 ボリューム・グループのすべての物理ボリュームは、同じサイズの物理区画 (PP) に分割されます。 スペース割り当ての目的で、それぞれの物理ボリュームは 5 つの領域 (**outer_edge**、**inner_edge**、**outer_middle**、**inner_middle**、**center**) に分割されます。 各領域の物理区画の数は、ディスク・ドライブの合計容量によって異なります。

各ボリューム・グループの中では、1 つ以上の論理ボリューム (LV) が定義されます。 論理ボリューム上のデータはユーザーからは連続しているように見えますが、 物理ボリューム上では不連続にすることができます。 これによって、ファイルシステム、ページング・スペース、およびその他の論理ボリュームについて、 サイズ変更や再配置を行ったり、複数の物理ボリュームにスパンしたり、内容を複製したりできるので、 データ・ストレージにおける柔軟性と可用性が高まります。

それぞれの論理ボリュームは、1 つ以上の論理区画 (LP) から構成されます。 各論理区画は少なくとも 1 つの物理区画に対応します。 論理ボリュームに対してミラーリングが指定されると、追加の物理区画が割り当てられ、 各論理区画の追加コピーが保管されます。 論理区画には連続した番号が付けられますが、基礎となる物理区画は必ずしも連続していたり、 隣接しているとは限りません。

論理ボリュームは、ページングなどの多くのシステム目的のために使用できますが、 それぞれの論理ボリュームは単一の目的のためだけに使用されます。 多くの論理ボリュームには、単一のジャーナル・ファイルシステム (JFS または JFS2) が入っています。 各 JFS はページ・サイズ (4 KB) のブロックのプールから構成されます。 データがファイルに書き込まれるとき、1 つ以上の追加ブロックがそのファイルに割り当てられます。 これらのブロックはお互いに、または前にファイルに割り当てられた他のブロックと、 連続していない場合があります。 ある特定のファイルシステムのフラグメント・サイズを 4 KB 未満 (512 バイト、1 KB、2 KB) に定義することも可能です。

インストール後のシステムには、システムを開始するために必要な論理ボリュームの基本セットと、 インストール・スクリプトに指定されたその他の論理ボリュームから構成される 1 つのボリューム・グループ (rootvg ボリューム・グループ) が存在します。 システムに接続したその他の物理ボリュームは、ボリューム・グループに追加することができます (**extendvg** コマンドを使用)。 物理ボリュームは、rootvg ボリューム・グループまたは別のボリューム・グループ (**mkvg** コマンドを使用して定義される) のいずれかに追加できます。 論理ボリュームは、コマンド、メニュー方式のシステム管理インターフェース・ツール (SMIT) を使用して調整できます。

関連タスク

物理ボリュームが見つからないときの管理者への通知
AIX は、物理ボリュームがアクセス不能なときにエラーをログに記録しますが、エラーが検知不能になることがある環境があります。

ミラーリングしたディスクのボリューム・グループからの分割
ミラーリングされたボリューム・グループの一貫性が、 潜在的なディスク障害のために損なわれるのを防ぐ上で、スナップショットのサポートが役立ちます。

ルート・ボリューム・グループのファイルシステムのサイズの 縮小

すべての ファイルシステムを最小サイズに縮小する一番簡単な方法は、バックアップから基本オペレーティング・システムを復元するときに、**SHRINK** オプションを「**yes**」に設定することです。

デバイスのインストール準備

システムへのデバイスのインストールは 、デバイスの接続場所の識別、デバイスの物理的な接続、および、構成マネージャー、または SMIT を使用してデバイスを構成することによって行います。

注 : 次の手順では、デバイスをインストールするためにシステムのシャットダウンが 必要になります。 すべてのデバイスがインストール時にシステムのシャットダウンを必要とするわけではありません。 個別のデバイスについてはデバイスに 付いている資料を参照してください。

このトピックでは、すべてのデバイスに共通するインストール・タスクについて説明します。 システムにインストールできるデバイスは多様なため、一般的な手順だけを説明します。 個別のデバイスについては、デバイスに付いているインストールの指示を参照してください。

1. **shutdown** コマンドを使用して、システム装置で実行されているアプリケーションをすべて停止し、そのシステム装置をシャットダウンします。
2. システム装置と、接続しているすべてのデバイスをオフにします。
3. システム装置と、接続しているすべてのデバイスのプラグを抜きます。
4. デバイスのセットアップおよびオペレーター・ガイドに記載されている手順を使用して、新しいデバイスをシステムに接続します。
5. システム装置と、接続しているすべてのデバイスの電源を入れます。
6. システム装置はオフにしたまま、接続しているすべてのデバイスをオンにします。
7. すべてのデバイスの電源オン自己診断テスト (POST) が完了したら、システム装置をオンにします。

構成マネージャーは接続されているデバイスを自動的に走査し、検出した新しいデバイスをすべて構成します。 新しいデバイスはデフォルトの属性を使用して構成され、カスタマイズされた構成データベースに記録された結果、**使用可能**状態になります。

SMIT の高速パス **smit dev** を使用して、デバイスを手動で構成できます。 デバイス属性をカスタマイズする必要がある場合、またはデバイスを自動的に構成できない場合は、デバイスに付いているデバイスの資料を参照して、個別の構成要件を調べてください。

読み取り/書き込み光ディスク・ドライブの構成

読み取り/書き込み光ディスク・ドライブには 2 つの構成方法があります。

読み取り/書き込み光ディスク・ドライブはシステムに 接続し、電源オンする必要があります。

メソッド 1

メソッド 1 は 2 つの方法のうち速い方のメソッドです。 それは指定された読み取り/書き込み光ディスク・ドライブを構成するだけです。 このメソッドを使用するためには、以下の情報を提供する必要があります。

項目	説明
サブクラス	ドライブの接続方法を定義します。
タイプ	読み取り/書き込み光ディスク・ドライブのタイプを指定します。
親の名前	ドライブが接続されるシステム・アタッチメントを指定します。
接続場所	ドライブの論理アドレスを指定します。

読み取り/書き込み光ディスク・ドライブを構成するために以下のコマンドを入力します。

```
mkdev -c rwoptical -s Subclass -t Type -p ParentName -w WhereConnected
```

以下は、SCSI ID 6、論理ユニット番号を持ち、3 番目の (scsi3) SCSI バスに接続されている読み取り/書き込み光ディスク・ドライブの例です。

```
mkdev -c rwoptical -s scsi -t osomd -p scsi3 -w 6,0 -a pv=yes
```

メソッド 2

メソッド 2 は構成マネージャーを使用して、現行の構成を検索し、新規デバイスを検出し、デバイスを自動的に構成します。 このメソッドは、読み取り/書き込み光ディスク・ドライブに関する情報が少ないときに使用します。

1. システム上で新規に検出されたすべてのデバイス (読み取り/書き込み光ディスク・ドライブを含め) を構成するために、構成マネージャーを使用します。

   ```
   cfgmgr
   ```

2. 現在構成されているすべての読み取り/書き込み光ディスク・ドライブの名前、ロケーション・コード、およびタイプをリストするために、以下のコマンドを入力します。

   ```
   lsdev -C -c rwoptical
   ```

3. 追加しようとするドライブのロケーションに一致するロケーション・コードを使用して、新規に構成される読み取り/書き込み光ディスク・ドライブの名前を確認します。

大量デバイスの構成

デバイスには、プリンター、ドライブ、アダプター、バス、およびエンクロージャーなどのハードウェア・コンポーネントのほかに、エラー・スペシャル・ファイルおよびヌル・スペシャル・ファイルなどの疑似デバイスなどもあります。 デバイス・ドライバーは、/usr/lib/drivers ディレクトリーにあります。

AIX がサポートできるデバイスの数は、システムによってそれぞれ異なり、 いくつかの重要な要因によって決まります。 以下に、デバイスをサポートするファイルシステムに 影響を与える要因を示します。

- 多数のデバイスを構成するには、ODM デバイス構成データベース内により多くの情報を入れるためのストレージが必要になります。 より多くのデバイス・スペシャル・ファイルも必要になる可能性があります。 その結果、ファイルシステムにはより多くのスペースとより多くの i ノードが必要になります。
- 一部のデバイスは、その他のデバイスより多くのスペースを ODM デバイス構成データベース内に必要とします。 使用されるスペシャル・ファイルまたは i ノードの数もデバイスによって異なります。 その結果、ファイルシステムに必要とされるスペース量および i ノードは、システム上のデバイスのタイプに応じて変わります。
- マルチパス入出力 (MPIO) デバイスには、非 MPIO デバイスより多くのスペースが必要です。 なぜなら、ODM 内にはそのデバイス自体に関する情報だけでなく、そのデバイスへの一つ一つのパスに関する情報も保管されるからです。 大まかなガイドラインとして、それぞれのパスがデバイスの 5 分の 1 のスペースをとるものと想定します。 例えば、5 つのパスを持つ 1 台の MPIO デバイスは、非 MPIO デバイス 2 台分と等しいスペースをとることになります。
- AIX は論理デバイスと物理デバイスの両方を、ODM デバイス構成データベースに組み込みます。 論理デバイスには、ボリューム・グループ、論理ボリューム、ネットワーク・インターフェースなどが 含まれます。 場合によっては、論理デバイスと物理デバイスとの関係が、サポートされるデバイスの合計数に多大な影響を与える可能性があります。 例えば、システムに接続されているそれぞれの物理ディスクごとに 2 つの論理ボリュームを持つボリューム・グループを定義した場合、その結果としてディスクごとに 4 つの AIX デバイスが存在することになります。 一方、それぞれの物理ディスクごとに 6 つの論理ボリュームを持つボリューム・グループを 定義した場合は、ディスクごとに 8 つの AIX デバイスが存在することになります。 そのため、ディスクの半数しか接続できなくなります。
- デバイスの属性をそれらのデフォルト設定から変更すると、結果的に ODM デバイス構成データベースが大きくなり、サポートできるデバイス数の減少につながる可能性があります。
- デバイスの数が増えれば増えるほど、より多くの実メモリーが必要となります。

AIX では、デバイスのサポートに次の 2 つのファイルシステムが 使用されます。

- RAM ファイルシステム。これは、ページング・スペースがなく、ディスク・ファイルシステムもマウントされていない環境でブート時に使用されます。 RAM ファイルシステムのサイズは 、システム・メモリー・サイズの 25% で、最大 128MB までです。 RAM ファイルシステムでは、1KB ごとに 1 つの i ノードが割り振られます。 AIX オペレーティング・システムの最小システム・メモリー要件は 256 MB で、これが 65536 個の i ノードを持つ 64 MB の最小 RAM ファイルシステム ・サイズに変換されます。 システム・メモリー・サイズが 512MB 以上の場合、RAM ファイルシステムはその 最大サイズである 128MB となり、i ノードの数は 131072 個となります。 接続されているデバイスのサポートに必要な RAM ファイ

ルシステム・スペースの量 または i ノードの数のいずれかが、その RAM ディスクに割り振られている数量を超えた場合は、システムをブートできない可能性があります。 この場合は、デバイスのいくつかを除去する必要があります。

- ディスク上のルート・ファイルシステム (rootvg) のスペースおよび i ノードは、その rootvg に未割り振りの区画がある限り増やすことができます。 最大の RAM ファイルシステム・サイズを使用した場合は、最大 25,000 の AIX デバイスを構成できる可能性があります。 この数には、物理デバイスと論理デバイスの両方が含まれます。 このセクションで言及したさまざまな要因によっては、 使用するシステムで構成可能なデバイスの数がこの数より多い場合や少ない場合があります。

注 : システム内のデバイスの数が多ければ多いほど、構成時間が長くなり、その分ブート時間が長くなります。

取り外し可能メディア・ドライブの追加

取り外し可能メディア・ドライブは追加できます。

以下の手順では、SMIT を使用して CD-ROM ドライブをシステムに追加します。 その他のタイプの取り外し可能メディア・ドライブについては、 別の高速パスを使用して追加します。ただし、どのタイプの場合も、一般的手順は同じです。 構成マネージャー、 または **mkdev** コマンドを 使用して取り外し可能メディア・ドライブを追加することもできます。

ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

1. CD-ROM ドライブをシステムに 追加するには、システムに付属している文書に従ってハードウェアを取り付けます。
2. root 権限で、次の SMIT 高速パスを入力します。

   ```
   smit makcdr
   ```

3. 次の画面で、サポートされているドライブの使用可能リストからドライブ・タイプを選択します。
4. 次の画面で、使用可能リストから親アダプターを 選択します。
5. 次の画面で、少なくとも、使用可能リストから接続アドレスを選択します。 この画面では、その他のオプショ ンを選択することもできます。 操作手順が終了したら、Enter キーを押します。これで SMIT により新規 CD-ROM ドライブが追加されます。

この時点で、新規 CD-ROM ドライブはシステムによって 認識されます。 読み取り/書き込み光ディスク装置を追加するには、**smit makomd** 高速パスを使用します。 テープ・ドライブを追加するには、**smit maktpe** 高速パス を使用します。

詳しくは、「コマンド・リファレンス 第 *3* 巻」で **mkdev** コマンドの説明を参照してください。

論理ボリューム・ストレージのスペース 再利用のサポート

7200-01 テクノロジー・レベル以降の AIX 7.2 では、論理ボリューム・マネージャー (LVM) はスペース再利用に対応する物理ボリュームのスペース再利用をサポートします。

LVM は、パーティション・スペースが使用されなくなり、ストレージ・サブシステムが割り振り済みスペースを再利用できることをディスク・ドライバーに通知し、ディスク・ドライバーはその内容をストレージ・サブシステムに通知します。 LVM はディスク・ドライバーの助けを借りて、物理ボリュームがスペース再利用に対応するかどうかを検出します。 LVM とファイル・システムの構成コマンド (例えば、`rmlv` コマンド、`rmlvcopy` コマンド、`chfs(shrink fs)` コマンド) は、パーティションが解放された後、それらのスペース再利用を開始します。 LVM は、`varyonvg` コマンドまたは `extendvg` コマンドの実行中、ボリュームをオープンする際に、物理ボリュームがスペース再利用に対応するかどうかを検出します。LVM は、ボリューム・グループがオンラインのときにも、その検出を試みます。 状態変更を検出するために物理ボリュームの再オープンが必要な場合、管理者はボリューム・グループに対して `varyoffvg` コマンドを実行してから、`varyonvg` コマンドを実行する必要があります。

AIX 7.2 テクノロジー・レベル 1 より前に作成されたボリューム・グループに空きパーティション・スペースがある場合がありますが、このスペースは自動再利用の対象になりません。 管理者は、これらの空きパーティション上にダミーの論理ボリュームを作成し、それを削除することで、そのスペースを再利用でき

ます。 ただし、AIX 7.2 テクノロジー・レベル 1 のインストール後に解放されたパーティションでは、スペースが自動的に再利用されます。

スペースを再利用するための LVM プロセスは、`rmlv` などのコマンドの実行完了後にバックグラウンドで実行されます。LVM プロセスがすべてのパーティションの 再利用プロセスを完了する前にシステムが異常終了した場合、パーティションは解放されますが、処理中のパーティションのスペースは再利用されません。 このようなシナリオが発生した場合、ダミーの論理ボリュームを作成し、それを削除することで、残りのパーティションのスペースを 再利用できます。

LVM プロセスでは、スペース再利用プロセスが処理中であっても、`varyoffvg` コマンドと `reducevg` コマンドのいずれの処理も遅延させません。スペース再利用プロセスは、そのプロセスが完了するのを待つのではなく破棄されます。

注 : コマンドは、ディスク・ドライバーに実行依頼された未処理のスペース再利用要求のみ待機します。

スペース再利用機能は、物理ボリュームから解放されたスペースを再利用するために、ストレージ・サブシステムから使用できます。 各ストレージ・サブシステムは再利用要求が特定の数の物理ブロックに合わせて調整されることを想定しており、物理ブロックの数はストレージ・サブシステムによって 異なります。そのため、場合によっては、再利用サイズがパーティションの 物理ブロックに合っていないために、そのパーティションのブロック (すべて、または一部) の再利用ができないことがあります。 一部のストレージ・サブシステムは LVM パーティション・サイズを超えるブロック・サイズの再利用をサポートしており、部分的なブロックの再利用はできません。 このシナリオでは、LVM は十分な連続した空きパーティションを蓄積できず、1 つの再利用要求さえ生成できない場合があります。 そのため、複数の LVM パーティションを削除した場合、ストレージ・サブシステムで同じ量のスペースを再利用できないことがあります。 **-r** オプションを指定した `lvmstat` コマンドを使用すると、LVM によって生成されるスペース再利用要求に関する情報を取得できます。

関連情報

varyoffvg コマンド

論理ボリューム・ストレージの概念

論理ボリューム (物理ボリュームにスパンできる) は、物理区画に割り当てられる論理区画から構成される。

以下の図で、基本的な論理ストレージの概念の間の関係を示します。

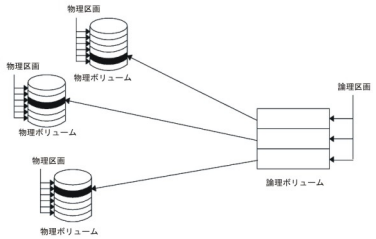


図 *1.* ボリューム・グループ

物理ボリューム

ディスクは物理ボリュームとして指定し、ボリューム・グループに割り当てる前に、使用可能な状態にしておく必要があります。

物理ボリュームには特定の構成情報と識別情報が書き込まれています。 この情報にはシステムに固有な物理ボリューム ID が含まれています。

論理装置番号 (LUN) に関連付けられた物理ボリュームに物理区画を追加することによって、RAID が LUN に追加できる追加スペースを LVM が利用できます。

ボリューム・グループ

ボリューム・グループ は、さまざまなサイズおよびタイプの 1 から 32 個の物理ボリュームの集合です。

大きなボリューム・グループは、1 から 128 個の物理ボリュームを持つ場合があります。 スケーラブルなボリューム・グループは、最大 1024 個の物理ボリュームを 持つことができます。 各物理ボリュームは、1 システムにつき 1 つのボリューム・グループにしか属せません。 1 つのシステム内に存在し得るアクティブ・ボリューム・グループの最大数は 255 です。

物理ボリュームをボリューム・グループに割り当てると、 その物理ボリュームのストレージ・メディア上の物理ブロックは、ユーザーがボリューム・グループ作成時に指定したサイズの物理区画に編成されます。

システムをインストールすると、そのシステムの始動に必要な論理ボリュームの基本セットと、 インストール・スクリプトに指定されたその他の論理ボリュームをすべて含む 1 つの ボリューム・グループ (rootvg と呼ばれるルート・ボリューム・グループ) が自動的に作成されます。 rootvg にはページング・スペース、ジャーナル・ログ、ブート・データ、ダンプ・ストレージがあり、 それぞれは固有の別個の論理ボリューム内にあります。 rootvg の属性は、ユーザー定義のボリューム・グループとは異なります。 例えば、rootvg はインポートまたはエクスポートをすることができません。 rootvg でコマンドまたは手順を実行する場合には、その固有の特性に精通していることが必要です。

ボリューム・グループは **mkvg** コマンドで作成します。 物理ボリュームをボリューム・グループに追加するには **extendvg** コマンドを使用し、物理ボリュームの変更されたサイズを利用するには **chvg** コマンドを使用し、 物理ボリュームをボリューム・グループから除去するには **reducevg** コマンドを使用します。 ボリューム・グループに使用するその他のコマンドには、リスト (**lsvg**)、除去 (**exportvg**)、インストール (**importvg**)、再編成 (**reorgvg**)、同期化 (**syncvg**)、使用可能化 (**varyonvg**)、使用不能化 (**varyoffvg**) を行うコマンドがあります。

小規模なシステムの場合には、システムに接続されたすべての物理ボリュームから構成されるボリューム・グループが 1 つあれば十分なこともあります。 しかし、各ボリューム・グループに個別にセキュリティー許可を持たせることができるため、 セキュリティー上の理由から複数のボリューム・グループを作成したい場合があります。 また、維持管理中のボリューム・グループ以外のグループをアクティブにしておくことができるため、 ボリューム・グループを複数に分けることによって維持管理が容易になります。 rootvg は常にオンラインである必要があるため、 システム操作に必要な最低数の物理ボリュームのみを入れるようにする必要があります。

migratepv コマンドを使用して、物理ボリュームから同じボリューム・グループ内の 他の物理ボリュームにデータを移動できます。 このコマンドによって、物理ボリュームを解放してボリューム・グループから除去できるようになります。 例えば、置き換えられることになっている物理ボリュームからデータを移動することができます。

物理ボリュームと論理ボリュームの限界を小さくして作成されたボリューム・グループは、 より多くの物理ボリュームおよび論理ボリュームを保持できるフォーマットに 変換することができます。 この操作には、ボリューム・グループ記述域 (VGDA) の拡張のために、 ボリューム・グループ内のすべての物理ボリュームに十分な空き区画が必要です。 必要な空き区画数は、現在の VGDA のサイズおよび物理区画サイズによって異なります。 VGDA はディスクのエッジに置かれ、連続するスペースが必要であるため、 ディスクのエッジに空き区画が必要です。 それらの区画がユーザー用として割り当てられていた場合は、 それらは同じディスク上のその他の空き区画に移行されます。 残りの物理区画は、VGDA 用に区画が減少したことを反映するために、番号が振り直されます。 この再番号付けにより、このボリューム・グループのすべての物理ボリュームでの論理区画から物理区画へのマッピングが変更されます。 リカバリー操作用に論理ボリュームのマッピングを保管している場合は、変換操作の完了後にマップを再度生成してください。 また、マップ・オプションを指定してボリューム・グループのバックアップを取り、 それらのマップを使用して復元しようとすると、区画番号が (縮小のため) 存在しなくなっているために、 復元操作が失敗することがあります。 マップ・オプションを使用する場合、 変換の前および変換直後にバックアップを取るようにお勧めします。 VGDA スペースがかなり増加しているため、どの VGDA 更新操作 (論理ボリュームの作成、論理ボリュームの変更、物理ボリュームの追加など) も、相当に時間がかかる恐れがあります。

物理区画

ボリューム・グループに物理ボリュームを追加すると、 物理ボリュームは物理区画 と呼ばれる、連続する等しいサイズを持つスペースの単位に区画化されます。 物理区画はストレージ・スペース割り当ての最小単位であり、物理ボリューム上の連続したスペースです。

物理ボリュームは、ボリューム・グループの作成時にのみ設定できる (例えば、**mkvg -s** コマンドを使用して) ボリューム・グループの物理区画サイズを継承します。 次の図は、物理ボリューム上の物理区画とボリューム・グループの間の関係を示しています。

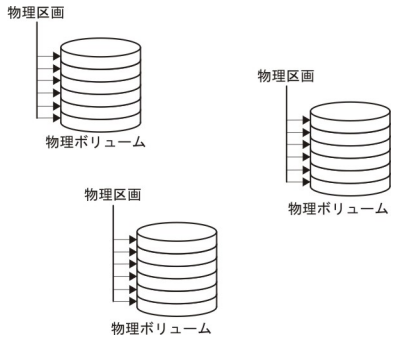


図 *2. 3* つの物理ボリュームから成るボリューム・グループ

論理ボリューム
ボリューム・グループを作成すると、そのボリューム・グループ内に論理ボリュームを作成することができます。

論理ボリューム は、不連続の物理区画に存在することも、 または複数の物理ボリューム上に存在することさえもできますが、 ユーザーとアプリケーションからは、論理ボリュームは単一の連続した拡張可能なディスク・ボリュームであるように見えます。 **mklv** コマンドを使用すれば、追加の論理ボリュームを作成することができます。 このコマンドを使用すれば、論理ボリュームの名前を指定し、論理ボリュームに割り当てる論理区画の数およびロケーションなどの特性を定義することができます。

論理ボリュームを作成後、**chlv** コマンドでその名前や特性を変更したり、**extendlv** コマンドでそれに割り当てた論理区画の数を増やすことができます。 作成時の論理ボリュームのデフォルトの最大サイズは、512 を超える数を指定しない限り、512 論理区画になります。 この制限を指定変更するには、**chlv** コマンドを使用します。

注 : 論理ボリュームの作成後、**lslv** コマンドを使用して参照できる特性 LV STATE はクローズされます。 これは、例えばその論理ボリュームにファイルシステムが 作成され、 その論理ボリュームがマウントされた場合などにオープンします。

論理ボリュームは、**cplv** コマンドでコピーしたり、**lslv** コマンドでリストしたり、あるいは、**rmlv** コマンドで除去することもできます。 また、**mklvcopy**、**rmlvcopy** コマンドによって、論理ボリュームが保持するコピー数を、それぞれ、増したり、 減らしたりすることもできます。 また、論理ボリュームは、ボリューム・グループを再編成するときに再配置することもできます。

このシステムでは、標準ボリューム・グループあたり最大 255 個 (大きなボリューム・グループの場合は最大 511 個、そしてスケーラブルなボリューム・グループの場合は最大 4095 個) の論理ボリュームを定義できますが、実際に定義できる数は、そのボリューム・グループに 対して定義されている物理ストレージの総量と、定義する論理ボリューム・サイズに よって決まります。

論理区画
論理ボリュームを作成するときに、論理ボリュームの論理区画 の数を指定します。

論理区画は、保持したいデータのインスタンスの数に応じて、1 つ、2 つ、または 3 つの物理区画から構成されます。 インスタンスを 1 と指定すると、論理ボリュームのコピーがただ 1 つしかないことを意味します (デフォルト)。 この場合には、1 つの論理区画から 1 つの物理区画への直接マッピングが行われます。 それぞれのインスタンスは、最初のものも含めてコピー と呼ばれます。 物理区画が位置する場所 (すなわち、互いに物理的にどの程度近くに配置されるか) は、論理ボリュームを作成するときにユーザーが指定するオプションによって決まります。

ファイルシステム
論理ボリュームはディスク・スペースの割り当てを物理区画のレベルまで下げて定義します。 よりきめの細かいデータ管理が、仮想メモリー・マネージャーまたはファイルシステムなどの 高水準のソフトウェア・コンポーネントによって行われます。 したがって、ディスクの設定の最終ステップが、ファイルシステムの作成です。

論理ボリュームごとに 1 つのファイルシステムを 作成することができます。 ファイルシステムを 作成するには、**crfs** コマンドを使用します。

論理ストレージ管理の制約
次の表は論理ストレージ管理の制約を示しています。

ボリューム・グループあたりの物理ボリュームのデフォルトの最大数は 32 (大きなボリューム・グループの場合は 128、そして スケーラブルなボリューム・グループの場合は 1024) ですが、**mkvg** コマンドを使用するとユーザー定義のボリューム・グループの場合の最大数を 設定できます。 しかし、rootvg の場合は、この変数はインストール時にシステムが自動的に最大に設定します。

カテゴリー	限界
ボリューム・グループ	• 32 ビット・カーネル用の 255 ボリューム・グループ • 64 ビット・カーネル用の 4096 ボリューム・グループ **注 :** 64 ビット・カーネル上の 装置テーブルは、アクティブ・メジャー番号を 1024 に制限します。 その結果、 アクティブ・ボリューム・グループの数は、1024 のボリューム・グループより小さいものに制限されます。
物理ボリューム	1 ボリューム・グループあたり (MAXPVS/ボリューム・グループ・ファクター)。 MAXPVS は、標準ボリューム・グループの場合は 32、大きなボリューム・グループの場合 は 128、そしてスケーラブルなボリューム・グループの場合は 1024。
物理区画	通常および大きなボリューム・グループ: 最大サイズ 1024 MB の物理ボリュームあたり (1016 x ボリューム・グループ・ ファクター)。 スケーラブルなボリューム・グループ: 最大サイズ 128 GB の 2097152 区画。 スケーラブルなボリューム・グループには、ボリューム・グループ・ファクター はありません。
論理ボリューム	ボリューム・グループあたりの MAXLVS。標準ボリューム・グループの場合は 255、大きなボリューム・グループの場合は 511、 そしてスケーラブルなボリューム・グループの場合は 4095。

物理ボリュームあたりの物理区画は 1016 個までという制約が適用される前にボリューム・グループを既に作成していた場合、 そのボリューム・グループ内の不整合区画 (最新のデータがもはや入っていないもの) は、そのボリューム・グループをサポートされる状態に変換しない限り、正しくトラッキングされません。 ボリューム・グループは、**chvg -t** コマンドによって変換できます。 ボリューム・グループ内で最大のディスクに対応するのに適したファクター値がデフォルトで選択されます。

例えば、9GB のディスクおよび 4MB の区画サイズを指定してボリューム・グループを作成した場合、 このボリューム・グループには約 2250 個の区画があります。 変換ファクター 3 (1016 * 3 = 3048) を使用すると、2250 個の区画がすべて正しくトラッキングされます。 これより高いファクターを指定して標準または大きなボリューム・グループを変換すると、最高 1016* ファクターの区画の、 より大きなディスクを含めることが可能になります。 また、小さな区画サイズを使用してより大きなディスクに対応するために、ボリューム・グループを作成する場合も、より大きなファクターを指定することができます。

こうした操作では、1 つのボリューム・グループに追加できるディスクの合計数が少なくなります。 追加できるディスクの新しい最大数は、MAXPVS/ファクターになります。 例えば、正規のボリューム・グループの場合にファクターを 2 にすると、ボリューム・グループ内のディスクの最大数は 16 (32/2) に減少します。 大きなボリューム・グループの場合にファクターを 2 にすると、ボリューム・グループ内のディスクの最大数は 64 (128/2) に減少します。

LVM デバイス・サイズの制限

以下の制限は、LVM の構造上の制限です。 LVM の不良ブロックの再配置が必要である場合、PV サイズは 128 GB を 超えることができません。 特定のストレージ・デバイスのサイズ 制限については、ストレージ・デバイスの資料を参照してください。

以下の制限は、64 ビットカーネル用です。

オリジナル VG
PV 制限: 1GB (PP) * 16256 (PPs/PV, factor=16) = 15.9 TB

LV 制限: 1GB (PP) * 32512 (PPs/VG) = 31.8 TB

大きい VG
PV 制限: 1GB (PP) * 65024 (PPs/PV, factor=64) = 63.5 TB

LV 制限: 1GB (PP) * 130048 (PPs/VG) = 127 TB

SVG
PV & LV 制限: 128GB (PP) * 2048K (PPs/PV) = 256 PB

以下の制限は、32 ビットカーネル用です。

すべての VG タイプ
PV 制限: < 1 TB

LV 制限: < 1 TB

論理ボリューム・ストレージの構成

論理ボリューム・ストレージについて、ボリューム・グループのミラーリング、論理ボリュームの定義、システム実行中のディスクの除去を行うことができます。

ボリューム・グループのミラーリング

以下のシナリオは、標準ボリューム・グループのミラーリング方法の説明です。

以下の手順では、SMIT (System Management Interface Tool) を使用して ルート・ボリューム・グループをミラーリングする方法を説明します。

(「**Volumes (ボリューム)**」コンテナーのボリューム・グループを選択し、次に「**Selected (選択)**」メニューから「**Mirror (ミラー)**」を選択します。) 経験豊富な管理者は、**mirrorvg** コマンドを使用することができます。

1. root 権限で、次の SMIT 高速パスを使用して、ボリューム・グループにディスクを 追加します。

   ```
   smit extendvg
   ```

2. 次の SMIT 高速パスを入力して、ボリューム・グループを新規ディスクに ミラーリングします。

   ```
   smit mirrorvg
   ```

3. 最初のパネルで、ミラーリングするボリューム・グループを選択します。
4. 次のパネルで、ミラーリング・オプションを定義するか、デフォルトを受け入れることができます。 必要な場合はオンライン・ヘルプを使用できます。

注 : SMIT パネルを完了し、「OK」をクリックするか、終了した場合、基礎コマンドが完了するまで、かなり長い時間がかかることがあります。 時間の長さは、エラー検査、ボリューム・グループ内の論理ボリュームのサイズと数、および新しくミラーリングした論理ボリュームの同期化にかかる時間によって 影響されます。

この時点で、論理ボリュームに対するすべての変更が、SMIT パネルでの指定に従って、ミラーリングされます。

ルート・ボリューム・グループのミラーリング

ここでは、ルート・ボリューム・グループ (rootvg) をミラーリングする方法について説明します。

注 : ルート・ボリューム・グループのミラーリングでは、システム管理の豊富な経験が必要です。 ミラーリングを正しく行わないと、システムがリブートできなくなる 場合があります。

以下のシナリオでは、rootvg は hdisk01 に含まれています。また、ミラーは hdisk11 と呼ばれるディスクに作成されます。

1. 次のコマンドで hdisk11 がブート・デバイスとして AIX によってサポートされている ことを確認します。

   ```
   bootinfo -B hdisk11
   ```

   このコマンドで 1 の値が戻される場合は、選択したディスクを AIX が ブートできます。 それ以外の値の場合は、hdisk11 で rootvg のミラーリングを行うことはできません。
2. 次のコマンドを使用して、rootvg を拡張して hdisk11 を 組み込みます。

   ```
   extendvg rootvg hdisk11
   ```

   次のエラー・メッセージ、

   ```
   0516-050 Not enough descriptor space left in this volume group, Either try
   adding a smaller PV or use another volume group.
   ```

   または次のようなエラー・メッセージが表示される場合は、

   ```
   0516-1162 extendvg: Warning, The Physical Partition size of 16 requires the
   creation of 1084 partitions for hdisk11. The limitation for volume group
   rootvg is 1016 physical partitions per physical volume. Use chvg command with
   the -t option to attempt to change the maximum physical partitions per Physical
   Volume for this volume group.
   ```

   以下の作業を試してください。

   - 既に rootvg に属している空のディスクに rootvg をミラーリングする。
   - より小さなディスクを使用する。
   - 以下の手順を実行して、rootvg がサポートする区画の最大数を変更する。

     a. 宛先ディスクに必要な物理区画数および rootvg によって現在サポートされている 物理区画の最大数に関するメッセージを検査する。

     b. **chvg -t** コマンドを使用して、rootvg で現在許可されて いる区画の最大数 (上記の例では 1016) を、宛先ディスクに必要な物理区画数 (上記の例 では 1084) より大きな数に増やす。 例:

     ```
     chvg -t 2 rootvg
     ```

     c. ステップ 2 の先頭の **extendvg** コマンドを再実行します。
3. 次のコマンドに示すように、正確なマッピング・オプションを指定して、rootvg をミラーリングします。

   ```
   mirrorvg -m rootvg hdisk11
   ```

   ボリューム・グループが rootvg の場合は、このコマンドでクォーラム (定足数) がオフになります。 正確なマッピング・オプションを指定しない場合は、ブート論理ボリューム hd5 の新規コピーが、隣接する区画によって構成されていることを検査する 必要があります。
4. 次のコマンドを使用して、すべてのブート・レコードおよびブート・デバイス を初期化します。

   ```
   bosboot -a
   ```

5. 次のコマンドで、ブート・リストを初期化します。

   ```
   bootlist -m normal hdisk01 hdisk11
   ```

   注:

   a. **bootlist** コマンドが、hdisk11 を代わりのブート・ディスクとして識別しても、hdisk01 が失敗すると、システムが hdisk11 をブート・デバイスとして使用する保証はありません。 その場合は、

製品メディアからブートし、「**maintenance (保守)**」を選択し、かつ障害のあるディスクを指定せずに **bootlist** コマンドを再実行しなければならない場合があります。

b. ハードウェア・モデルで **bootlist** コマンドがサポートされていない場合 でも、rootvg をミラーリングすることができます。ただし、元のディスクが使用できない場合は、代わりのブート・ディスクを選択してアクティブにする必要があります。

アプリケーションのためのロー論理ボリュームの定義

ロー論理ボリューム とは、オペレーティング・システムまたはファイルシステムの 直接の制御下ではなく、アプリケーションの直接の制御下にある物理ディスク・スペースおよび論理ディスク・スペースの領域 のことです。例として、データベースまたは区画をあげることができます。

ファイルシステムをバイパスすると、制御アプリケーション、特にデータベース・アプリケーションで、パ フォーマンスを改善することができます。 ただし、改善の程度は、データベースのサイズまたはアプリケー ションのドライバーなどの要因によって異なります。

注 : 必要に応じて、新規ロー論理ボリューム用の、文字またはブロックの特殊装置ファイルを アプリケーションに提供する必要があります。 アプリケーションは、オープン、読み取り、書き込みなどの実行時にこのデバイス・ファイルにリンクします。

重要 : 各論理ボリュームでは、最初のブロックに論理ボリューム制御ブロック (LVCB) があります。LVCB のサイズは、ボリューム・グループ内の物理ボリュームのブロック・サイズです。データは、物理ボリュームの 2 番目のブロックから始まります。ロー論理ボリュームでは、LVCB は 保護されません。 アプリケーションが LVCB を上書きすると、LVCB を正常に更新できるコマンドが失敗し、メッセージが出ます。論理ボリュームは引き続き正常に動 作します。また、LVCB の上書きは許容できるイベントですが、この上書きはお勧めできません。

以下の手順では、SMIT およびコマンド・ライン・インターフェースを 使用して、ロー論理ボリュームを定義します。

ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

1. root 権限で、次の SMIT 高速パスを入力して、ロー論理ボリュームを作成できる空き物理区画を見付けます。

   ```
   smit lspv
   ```

2. ディスクを選択します。
3. 2 番目の「ダイアログ」(状況) でデフォルトを採用し、「**OK**」をクリックします。
4. 「**FREE PPs (空き物理区画)**」フィールドの値に「**PP SIZE (物理区画サイズ)**」フィールドの値を掛け て、選択したディスクで使用できるロー論理ボリュームの合計メガバイト数を求めます。

   フリー・スペース量が十分でない場合は、使用可能な十分なフリー・スペースをもつ別のディスクを探し、選択します。
5. SMIT を終了します。
6. **mklv** コマンドを実行して、ロー論理ボリュームを作成します。

   次のコマンドは、38 個の 4-MB 物理区画を使用して、`lvdb2003` という名前のロー論理ボリュームを db2vg ボリューム・グループに作成します。

   ```
   mklv -y lvdb2003 db2vg 38
   ```

   システム生成名を使用する代わりに、論理ボリュームの名前を指定するには、**-y** フラグを使用します。

この時点で、ロー論理ボリュームが作成されます。 ボリューム・グループの内容をリストする場合は、デフォルト・タイプの jfs をもったロー論理ボリュームが表示されます。 論理ボリュームのこのタイプの項目はラベルにすぎません。 この項目は、ファイルシステムが ロー論理ボリューム用にマウントされていることを示すものではありません。

/dev/*rawLVname* をオープンし、 このロー・スペースを使用する方法については、アプリケーションの説明書を参照してください。

関連概念
論理ボリューム・マネージャーの構成
論理ボリューム・マネージャー (LVM) は基本オペレーティング・システムと一緒に インストールされるので、それ以上の構成は必要ありません。 ただし、LVM がディスクを使用 できるようにするには、ディスクを構成し、物理ボリュームとして定義しておく必要があります。

関連情報
mklv コマンド
Logical Volume Manager from A to Z: Introduction and Concepts

ルート・ボリューム・グループのミラーリング解除
ルート・ボリューム・グループはミラーリング解除できます。

重要 : ルート・ボリューム・グループのミラーリング解除には、システム管理の豊富な経験が必要です。 正しく終了しなかった場合は、システムがブートできなくなることもあります。

以下のシナリオで、ルート・ボリューム・グループは hdisk01 上にあり、hdisk11 にミラーリングされます。 この例では、hdisk11 上のミラーを除去します。 継続のためにブートしたディスクに関係なく、手順は同じです。

1. 次のコマンドを使用して、hdisk11 上のルート・ボリューム・グループをミラーリング解除します。

   ```
   unmirrorvg rootvg hdisk11
   ```

   unmirrorvg コマンドは、ルート・ボリューム・グループのクォーラム (定足数) に戻します。
2. 次のコマンドを使用して、ルート・ボリューム・グループからディスクを減らします。

   ```
   reducevg rootvg hdisk11
   ```
3. 次のコマンドを使用して、残ったディスクのブート・レコードを再初期化します。

   ```
   bosboot -a -d /dev/hdisk01
   ```
4. ミラーリングが解除されたディスクをリストから除去するために、次のコマンドを使用して、ブート・リストを変更します。

   ```
   bootlist -m normal hdisk01
   ```

ディスクはミラーリング解除されます。

システムを使用可能にしたままのディスクの除去
以下の手順では、常時取り外し可能フィーチャーを使用してディスクを除去する方法 について説明します。この機能を使用すると、システムをオフにせずにディスクを 除去 できます。 この機能は、特定のシステムでしか使用できません。

常時取り外し可能フィーチャーは、以下の作業を行うときに便利です。

- セキュリティーまたは保守の目的で、別個の非 rootvg ボリューム・グループにデータを 格納しているディスクを取り外す。
- ボリューム・グループからディスクを 永続的に取り外す。
- ディスク障害を訂正する。

データのあるディスクの除去
この手順は、データが入ったディスクを、システムをオフにせずに除去する場合に使用します。

取り外すディスクは、別個の非 rootvg ボリューム・グループ内になければなりません。 この手順は、ディスクを別のシステムに移動したい場合に実行します。

1. 取り外したいディスクに関連したボリューム・グループをリストするには、次のように入力します。

   ```
   smit lspv
   ```

次のような出力が表示されます。

```
PHYSICAL VOLUME:    hdisk2                  VOLUME GROUP:     imagesvg
PV IDENTIFIER:      00083772caa7896e VG IDENTIFIER
0004234500004c00000000e9b5cac262

PV STATE:           active
STALE PARTITIONS:   0                       ALLOCATABLE:      yes
PP SIZE:            16 megabyte(s)          LOGICAL VOLUMES:  5
TOTAL PPs:          542 (8672 megabytes)    VG DESCRIPTORS:   2
FREE PPs:           19 (304 megabytes)      HOT SPARE:        no
USED PPs:           523 (8368 megabytes)
FREE DISTRIBUTION:  00..00..00..00..19
USED DISTRIBUTION:  109..108..108..108..90
```

ボリューム・グループの名前が「VOLUME GROUP」フィールドにリストされます。 この例では、ボリューム・グループは imagesvg です。

2. ディスクが別個の非 rootvg ボリューム・グループにあることを確認するために、次のように入力します。

```
smit lsvg
```

次に、対象となるディスクに関連したボリューム・グループ (この例では imagesvg) を選択します。 次のような出力が表示されます。

```
VOLUME GROUP:   imagesvg                VG IDENTIFIER:
0004234500004c00000000e9b5cac262

VG STATE:       active                  PP SIZE:        16 megabyte(s)
VG PERMISSION:  read/write              TOTAL PPs:      542 (8672 megabytes)
MAX LVs:        256                     FREE PPs:       19 (304 megabytes)
LVs:            5                       USED PPs:       523 (8368 megabytes)
OPEN LVs:       4                       QUORUM:         2
TOTAL PVs:      1                       VG DESCRIPTORS: 2
STALE PVs:      0                       STALE PPs:      0
ACTIVE PVs:     1                       AUTO ON:        yes
MAX PPs per PV: 1016                    MAX PVs:        32
LTG size:       128 kilobyte(s)         AUTO SYNC:      no
HOT SPARE:      no
```

この例では、「TOTAL PVs」フィールドに、imagesvg に関連した 物理ボリュームは 1 つしか存在しないことが示されています。 このボリューム・グループのすべてのデータは hdisk2 に入っているので、この手順を実行して hdisk2 を取り外すことができます。

3. ディスクの論理ボリューム上のすべてのファイルシステムをアンマウントするには、次のように入力します。

```
smit umountfs
```

4. ボリューム・グループを非活動化するには、以下を入力します。

```
smit varyoffvg
```

5. ボリューム・グループをエクスポートするには、以下を入力します。

```
smit exportvg
```

6. ディスクを削除するには、以下を入力します。

```
smit rmvdsk
```

7. 取り外したいディスクの LED 表示を見て、 黄色の LED がオフ (点灯していない) に なっていることを確認します。
8. 物理的にディスクを取り外します。 取り外し手順の詳細については、マシンの「サービス・ガイド」を参照してください。

この時点で、ディスクがシステムから 物理的および論理的に取り外されます。 このディスクを永続的に取り外したままにする場合は、これで手順は完了です。

データのないディスクの除去

以下の手順では、データのないディスク、あるいは保持しておきたいデータの ないディスクを取り外す方法について説明します。

重要 : 以下の手順を使用すると、ディスク上にあるすべてのデータが消去されます。

1. ディスクの論理ボリューム上のすべてのファイルシステムをアンマウントするには、次のように入力します。

   ```
   smit umountfs
   ```

2. ボリューム・グループを非活動化するには、以下を入力します。

   ```
   smit varyoffvg
   ```

3. ボリューム・グループをエクスポートするには、以下を入力します。

   ```
   smit exportvg
   ```

4. ディスクを削除するには、以下を入力します。

   ```
   smit rmvdsk
   ```

5. 取り外したいディスクの LED 表示を見て、 黄色の LED がオフ (点灯していない) に なっていることを確認します。
6. 物理的にディスクを取り外します。

   取り外し手順の詳細については、マシンの「サービス・ガイド」を参照してください。

この時点で、ディスクがシステムから 物理的および論理的に取り外されます。 このディスクを永続的に取り外したままにする場合は、これで手順は完了です。

ファイルシステムを除去することによる論理ボリュームの除去

以下の手順では、JFS または JFS2 のファイルシステム、その関連論理ボリューム、/etc/filesystems ファイル内の関連スタンザを除去する方法、およびオプションとして、ファイルシステムがマウントされる マウント・ポイント (ディレクトリー) を除去する方法を説明します。

重要 : ファイルシステムを 除去すると、指定したファイルシステムおよび 論理ボリューム のすべてのデータが壊れます。

異なるタイプのファイルシステムがマウントされている 論理ボリューム、またはファイルシステム を含まない論理ボリュームを除去したい場合は、論理ボリュームのみを除去できます。

SMIT を使用してジャーナル・ファイルシステムを 除去するには、以下の手順を実行します。

1. 次の例のようなコマンドを使用して、論理ボリューム上にあるファイルシステムをアンマウントします。

   ```
   umount /adam/usr/local
   ```

   注 : umount コマンドは、使用中のデバイスでは使用できません。 何らかの理由でファイルがオープンされている場合や、ユーザーの現行ディレクトリーがそのデバイスにある 場合、そのデバイスは使用中です。
2. ファイルシステムを 除去するために、次の高速パスを入力します。

   ```
   smit rmfs
   ```

3.

1. 除去したいファイルシステムの 名前を選択します。
2. 「**Remove Mount Point (マウント・ポイントの除去)**」フィールドに進み、設定の変更に切り替えます。 **yes** を選択すると、基礎コマンド によって、ディレクトリーが空の場合、ファイルシステムがマウントされている マウント・ポイント (ディレクトリー) も削除されます。

3. Enter キーを押してファイルシステムを除去します。 ファイルシステムを除去するかどうかを 確認するために、SMIT がプロンプトを出します。
4. ファイルシステムを除去することを確認します。 ファイルシステムが 正常に 除去されると、SMIT がメッセージを表示します。

この時点で、ファイルシステム、そのデータ、およびその関連論理ボリュームが システムから完全に除去されます。

関連タスク

論理ボリュームのみの除去

この手順を実行して、異なるタイプのファイルシステムがマウントされている 論理ボリューム、またはファイルシステム を含まない論理ボリュームを除去します。

論理ボリュームのみの除去

この手順を実行して、異なるタイプのファイルシステムがマウントされている 論理ボリューム、またはファイルシステム を含まない論理ボリュームを除去します。

重要 : 論理ボリュームを除去すると、指定したファイルシステムおよび 論理ボリューム のすべてのデータが壊れます。

以下の手順では、論理ボリュームおよびすべての関連ファイルシステムを除去する 方法について説明します。 この手順を実行して、JFS でないファイルシステムまたは ファイルシステムを 含まない論理ボリュームを除去することができます。 以下の手順では、論理ボリュームを除去する方法を説明した後、 `/etc/filesystems` ファイルにある JFS でないファイルシステムのすべての スタンザを除去する方法を説明します。

SMIT を使用して論理ボリュームを除去するには、以下の手順を実行します。

1. 論理ボリュームにファイルシステムがない 場合は、ステップ 4 に進みます。
2. 次のように入力して、論理ボリュームに関連したすべてのファイルシステムをアンマウントします。

   ```
   unmount /FSname
   ```

   /FSname はファイルシステムの 絶対パス名です。

   注 :

   a. **unmount** しようとしているファイルシステムが 使用中の場合は、**unmount** コマンドは失敗します。 **unmount** コマンドを実行できるのは、オープンしているファイルシステムのファイルがなく、ユーザーの現行ディレクトリーがそのデバイス 上にない場合だけです。

   b. **unmount** コマンドの別の名前は **umount** です。 どちらの名前も使用することができます。
3. 知る必要のあるファイルシステムの 情報をリストするため、次の高速パスを入力します。

   ```
   smit lsfs
   ```

   リストの一部は次のとおりです。

   ```
   Name            Nodename   Mount Pt         ...
   /dev/hd3        --         /tmp             ...
   /dev/locallv    --         /adam/usr/local  ...
   ```
4. 2 番目の項目に 標準命名規則が使用されているとすれば、ファイルシステムは `/adam/usr/local` という名前となり、論理ボリュームは `locallv` となります。 これを確認するには、以下の高速パスを入力します。

   ```
   smit lslv2
   ```

   リストの一部は次のとおりです。

   ```
   imagesvg:
   LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT
   ```

```
hd3                 jfs        4     4     1    open/syncd    /tmp
locallv             mine       4     4     1    closed/syncd  /adam/usr/local
```

5. 論理ボリュームを除去するために、コマンド・ラインに次の高速パスを入力します。

```
smit rmlv
```

6. 除去したい論理ボリュームの名前を選択します。
7. 「**Remove Mount Point (マウント・ポイントの除去)**」フィールドに進み、設定の変更に切り替えます。**yes** を選択すると、ディレクトリーが存在し、さらにディレクトリーが空の場合、基礎コマンドで、ファイルシステムが マウントされているマウント・ポイント (ディレクトリー) も削除されます。
8. Enter キーを押して論理ボリュームを除去します。 論理ボリュームを除去するかどうかを 確認するために、SMIT がプロンプトを出します。
9. 論理ボリュームを除去することを確認します。 論理ボリュームが正常に 除去されると、SMIT がメッセージを表示します。
10. 論理ボリュームに非 JFS ファイルシステムがマウントされている 場合は、次のように、ファイルシステムを除去し、`/etc/filesystems` ファイル にある関連するスタンザを除去します。

```
rmfs /adam/usr/local
```

または、次のようにファイルシステム 名を使用することもできます。

```
rmfs /dev/locallv
```

この時点で、論理ボリュームが除去されます。 論理ボリュームに非 JFS ファイルシステム が含まれている場合は、そのシステムのスタンザも `/etc/filesystems` ファイル から取り除かれます。

関連タスク

ファイルシステムを除去することによる論理ボリュームの除去
以下の手順では、JFS または JFS2 のファイルシステム、その関連論理ボリューム、`/etc/filesystems` ファイル内の関連スタンザを除去する方法、およびオプションとして、ファイルシステムがマウントされる マウント・ポイント (ディレクトリー) を除去する方法を説明します。

RAID ボリューム・グループのサイズ変更

RAID を使用するシステムで、**chvg** および **chpv** のコマンド・オプション を使用すると、システムを中断することなく、RAID グループにディスクを 追加し、LVM が使用する物理ボリュームのサイズを増やすことができます。

注 :

1. ボリューム・グループが標準コンカレント・モードまたは拡張コンカレント・モードで活動化されているときには、この機能は使用できません。
2. 以下の手順を実行して rootvg ボリューム・グループのサイズを 変更することはできません。
3. 以下の手順を実行して、アクティブ・ページング・スペースをもったボリューム・グループ のサイズを変更することはできません。

ボリューム・グループが活動化されるときに (varyon)、ボリューム・グループ内のすべての ディスクのサイズが自動的に調べられます。 サイズの増加が検出されると、通知メッセージが出ます。

以下の手順では、RAID 環境内のディスクのサイズを 増加する方法について 説明します。

1. ディスク・サイズの増加を調べ、必要ならサイズ変更を行うために、次のコマンドを入力します。

```
chvg -g VGname
```

VGname はボリューム・グループの名前です。 このコマンドは、ボリューム・グループ内のすべてのディスクを調べます。 サイズが増加したディスクがあれば、このコマンドは、物理ボリュームに物理区画を追加しようとします。 必要な場合は、適切な 1016 の制限と乗数を判別し、ボリューム・グループが大きな ボリューム・グループに変換されます。

2. ボリューム・グループの LVM 不良ブロック再配置をオフにするには、次のように入力します。

```
chvg -b ny VGname
```

VGname はボリューム・グループの名前です。

ボリューム・グループ・ストラテジー

ストレージ・システムで最も多く発生するのはディスク障害であり、次いで、 アダプターと電源装置の障害が多く発生します。 ディスク障害からの保護措置には、論理ボリュームの構成が関係します。

アダプターおよび電源装置の故障から守るには、 特定のボリューム・グループに関して特殊なハードウェア構成を考慮してください。 このような構成には、2 つのアダプター、1 つのアダプターに少なくとも 1 つのディスク、複数のアダプターにわたるミラーリング、 および、非クォーラム (定足数) ボリューム・グループ構成が組み込まれます。 この構成にかかる追加の費用は、必ずしもすべてのサイトやシステムに適切であるとは限りません。 このような構成は、高可用性 (最後の瞬間まで使用可能) が優先されている場合にのみお勧めします。 構成によっては、高可用性によって、 最新のバックアップと現在のデータ入力との間で発生するハードウェア障害に対応することができます。 高可用性は、誤って削除されたファイルには適用されません。

個別のボリューム・グループの作成時

物理ボリュームを rootvg とは別のボリューム・グループに編成できる理由は、いくつかあります。

- より安全で容易な維持管理を行うため。
  - ユーザー・ファイルがオペレーティング・システムの次のような操作の途中で危険にさらされることがないように、 ユーザー・ファイルシステムをオペレーティング・システムから分離することができるので、オペレーティング・システムの更新、再インストール、および破損のリカバリーはユーザー・ファイルにとってより安全になります。
  - ユーザー・データを復元しなくても、オペレーティング・システムを更新または再インストールすることが可能なので、 維持管理が容易になります。 例えば、更新を行う前に、ファイルシステムをアンマウントし、 ユーザー定義のボリューム・グループをシステムから取り外すことができます。 そのグループを **varyoffvg** コマンドを使用して非アクティブにしてから、**exportvg** コマンドを使用してエクスポートします。 システム・ソフトウェアの更新後に、ユーザー定義のボリューム・グループを **importvg** コマンドを使用して再導入し、 次いで、そのファイルシステムを再マウントすることができます。
- 異なる物理区画サイズを使用するため。 同一ボリューム・グループ内の物理ボリュームは、すべて、 同じ物理区画サイズをもつ必要があります。 異なる物理区画サイズを持つ物理ボリュームを使用するには、 それぞれのサイズを別のボリューム・グループ内に入れます。
- 異なるクォーラム (定足数) 特性が必要になる場合があるため。 非クォーラム (定足数) ボリューム・グループの作成を必要とするファイルシステムを使用している場合は、そのデータに対して別のボリューム・グループを維持管理します。 その他のファイルシステムは、すべて、 クォーラム (定足数) に従って作動しているボリューム・グループ内に残しておく必要があります。
- セキュリティーのため。 例えば、夜間にボリューム・グループを取り外したい場合。
- システム相互間で物理ボリュームを切り替えるため。 複数のシステムからアクセスできるアダプターに、システムごとに別々のボリューム・グループを作成すると、物理ボリュームを、相互の通常の操作を中断せずに、そのアダプターでアクセスできるシステム 間で切り替えることができます (**varyoffvg**、**exportvg**、**importvg**、および **varyonvg** コマンドを参照)。

ディスク障害の場合の高可用性

ディスク障害に対する主な防御策には、例えばミラーリングを行うなどの、 論理ボリュームの構成設定があります。

ボリューム・グループに関する考慮事項は 2 次的とは言え、 ボリューム・グループごとの物理ボリュームの数が関係してくるので、かなりの経済的な影響があります。

- クォーラム (定足数) 構成。これは、デフォルトの設定で、ディスクのクォーラム (定足数) (51%) が存在する限り、 ボリューム・グループをアクティブに (オンに変更) 保持します。 多くの場合、ディスク障害

から防御するには、 ミラーリングされたコピーが入っているディスクが少なくとも 3 つはボリューム・グループ内に必要です。

- 非クォーラム (定足数) 構成は、ディスク上で VGDA が 1 つでも使用可能である限り、ボリューム・グループをアクティブ状態 (オンに変更) に保ちます。 この構成を使用した場合は、ディスク障害から防御するために、 ミラーリングされたコピーが入っているディスクはボリューム・グループ内で 2 つ必要とされるだけです。

それぞれのボリューム・グループに入れるディスクの数を決定する際、 データをミラーリングするための空きスペースについても計画する必要があります。 データのミラーリングと移動は、同一ボリューム・グループ内にあるディスク相互間でのみ行うことができるということを忘れないでください。 サイトでラージ・ファイルシステムを使用している場合は、後で、 ミラー先のディスク・スペースを見つけることが難しくなる場合があります。 論理ボリューム・コピーのディスク間設定と論理ボリュームのディスク内割り振りの可用性の意味に注意してください。

アダプターまたは電源装置障害の場合の高可用性

アダプターや電源装置障害を防ぐには、ご使用の要件に応じて、次のうちいずれかの 1 つ以上を行ってください。

- 同一または異なるシャーシ内にある 2 つのアダプターを使用します。 別々のシャーシにアダプターを 置くと、1 つのシャーシで電源装置の故障が発生しても、 両方のアダプターを失わずに済みます。
- それぞれが少なくとも 1 つのディスクに接続する 2 つのアダプターを使用します。 このような方法を用いると、ディスク A (アダプター A) 上の論理ボリュームとディスク B (アダプター B) 上の論理ボリュームとの間でクロス・ミラーリング が行われる (論理区画のコピーは、同じ物理ボリュームを共用できない) とすれば、ボリューム・グループ内のクォーラム (定足数) をなおも維持できるので、 いずれかのアダプター (別々のキャビネットにアダプターがある 場合は電源装置) が故障しても、防御できます。 つまり、これは、アダプター A に接続しているディスク上に常駐する論理ボリュームをアダプター B に接続しているディスクにコピーするということ 、また、同様に、 アダプター B に接続しているディスクに常駐している論理ボリュームをアダプター A に接続しているディスクにコピーするということも 意味します。
- 両方のアダプター上にあるすべてのディスクを、同じボリューム・グループ内に構成します。 これによって、アダプターが故障しても、あるいは、キャビネットが別々である場合は電源装置が故障しても、少なくとも 1 つの論理ボリューム・コピーがもとのままの状態で残ります。
- ボリューム・グループを非クォーラム (定足数) ボリューム・グループにします。 これによって、ボリューム・グループ内のいずれかのディスク上でボリューム・グループ・ディスクリプター領域 (VGDA) がアクセス可能である限り、 ボリューム・グループはアクティブのままになっていることができます。
- ボリューム・グループ内に 2 つのディスクがある場合は、 アダプター相互間でクロス・ミラーリングをインプリメントします。 それぞれのアダプター上で複数のディスクが使用できる場合は、 ダブル・ミラーリングをインプリメントします。 その場合、同じアダプターを使用しているディスク上にミラーリングされたコピーを 1 つ作成し、 別のアダプターを使用しているディスク上にも 1 つ作成します。

関連概念

ボリューム・グループの非クォーラム (定足数) 状況への移行

クォーラム (定足数) がない場合でもデータを継続的に使用可能にしておくために、ボリューム・グループを非クォーラム (定足数) 状況に変更することができます。

論理ボリューム・ストラテジー

ここで説明するポリシーは、サイトに適した可用性、パフォーマンス、および、コストの組み合わせを目指した論理ボリュームの使用法のストラテジーを設定する上で役に立ちます。

可用性 とは、関連するディスクに障害が起こったり、 そのディスクがアクセス不能になったとしても、そのデータにアクセスできることです。 データは、通常のシステム操作の間、別のディスクおよびアダプター上に作成されて維持されているそのデータのコピーを通じて、 アクセス可能の状態になっています。 ミラーリングやホット・スペア・ディスクの使用といった手法は、 データの可用性を確保するのに役立ちます。

パフォーマンス は、データのアクセスの平均スピードです。 書き込み検査およびミラーリングなどのポリシーは可用性を向上させますが、 システムの処理負荷を増やし、結果としてパフォーマンスを低下させます。 ミラーリングを行うと、論理ボリュームのサイズは 2 倍または 3 倍になります。 一般的に、可用性を高めるとパフォーマンスは低下します。 ディスクのストライピングは、パフォーマンスを向上させること

ができます。 ディスクのストライピングはミラーリングと一緒に使用できます。ディスク上の一部の論理区画に対するディスク入出力がシステム・パフォーマンスを著しく損なうほど多い場合に発生するホット・スポット問題を検出して改善できます。

ディスク上およびディスク間のデータ割り当てを制御すれば、 最大限のパフォーマンスを実現できるようにストレージ・システムを調整できます。 ストレージ・システムのパフォーマンスを 最大限まで高める方法の詳細については、「パフォーマンス・マネージメント」を参照してください。

以下のセクションを使用して、パフォーマンス、可用性、およびコストの間のトレードオフを評価してください。 可用性の増大はしばしばパフォーマンスを低下させ、その逆も真であることを忘れないでください。 ミラーリングがパフォーマンスを増大させることもあります。しかし、LVM が読み取り用に、最小のビジーなディスク上のコピーを選択します。

> **注:** ミラーリングは、 誤って削除したりソフトウェアの問題のために失われた個々のファイルの消失は防止できません。 これらのファイルは、従来のテープまたはディスクのバックアップからのみ復元できます。

ミラーリングまたはストライピングのための要件
論理ボリュームに保管するデータが、ミラーリング処理とディスク・スペースのコストに見合うだけの価値があるかどうかを判別します。 パフォーマンスが重要な大きな順次アクセス・ファイルシステムがある場合には、ディスクのストライピングを考慮する必要もあります。

パフォーマンスとミラーリングは必ずしも対立するわけではありません。 論理区画のさまざまなインスタンス (コピー) が別々の物理ボリュームにある (別々のアダプターに 接続されていることが望ましい) 場合、最もアクセス頻度の少ないディスク上のコピーを読み取るようにすれば、LVM での読み取りパフォーマンスを向上させることができます。 書き込みでは、すべてのコピーを更新する必要があるので、ディスクが異なるアダプターに 接続されていない限り、常に同じ費用がかかります。 読み取り操作では、コピーは 1 つしか読み取る必要がありません。

AIX LVM は、以下の RAID オプションをサポートします。

表 *2. RAID* 用の論理ボリューム・マネージャーのサポート

項目	**説明**
RAID 0	ストライピング
RAID 1	ミラーリング
RAID 10 または 0+1	ミラーリングおよびストライピング

ミラーリングはストレージ・システムの可用性を向上させますが、それは従来のテープ・バックアップ操作の代わりにはなりません。

rootvg のミラーリングは可能ですが、これを行う場合は、別個のダンプ論理ボリュームを作成してください。 ミラーリングされている論理ボリュームへのダンプは、ダンプに不整合が生じる可能性があります。また、デフォルトのダンプ・デバイスは 1 次ページング論理ボリュームなので、ページング論理ボリュームをミラーリングする場合は別個のダンプ論理ボリュームを作成してください。

通常、論理区画のデータが更新されるたびに、その論理区画を含むすべての物理区画は自動的に更新されます。 しかし、システムの誤動作や更新時に物理ボリュームが使用不可であったことが原因で、 物理区画が不整合 (最新データが入っていない) 状態になることがあります。 LVM は、現行データを最新の物理区画から最新ではない状態の区画にコピーすることによって、 最新ではない状態の区画を整合性のとれた状態にリフレッシュできます。 このプロセスはミラーリング同期 と呼ばれます。 リフレッシュはシステムの再始動時、物理ボリュームがオンラインに戻ったとき、または、ユーザーが **syncvg** コマンドを発行するときに行われます。

ブート論理ボリュームの物理区画構成に影響を与える変更を行う場合には、その変更後に **bosboot** コマンドを実行する必要があります。 つまり、ブート論理ボリュームのミラーリングの変更などのアクションには、**bosboot** が必要です。

ディスクへのミラーリングされた書き込みに関するスケジューリング・ポリシー
1 つの物理コピーしか持たないデータの場合には、論理ボリューム・デバイス・ドライバー (LVDD) が、論理読み取り要求アドレスまたは論理書き込み要求アドレスを物理アドレスに変換し、要求を処理する適切な物理デバイス・ドライバーを呼び出します。 この単一コピー、すなわち非ミラーリング・ポリシーは、書き込み要求に対して不良ブロックの再配置を行い、呼び出し側のプロセスにすべての読み取りエラーを戻します。

ミラーリング論理ボリュームを使用している場合には、次のディスク書き込みスケジューリング・ポリシーを、複数のコピーを持つ論理ボリュームに対して設定できます。

順次スケジューリング・ポリシー
これは、複数のコピー、すなわちミラーを、順番に書き込みます。 単一論理区画のミラーリング・コピーである複数の物理区画は、1 次、2 次、3 次のように指定されます。 順次スケジューリングでは、物理区画は順番に書き込まれます。 すなわち、システムは 1 つの物理区画への書き込み操作が完了するまで待ってから、次の書き込み操作を始めます。 すべてのミラーリング用の書き込み操作がすべて完了すれば、書き込み操作は完了します。

並列スケジューリング・ポリシー
これは、論理区画のすべての物理区画での書き込み操作を同時に開始します。 完了までに最長の時間がかかる物理区画に対する書き込み操作が終わった時点で、書き込み操作が完了します。 並列スケジューリング・ポリシーと一緒にミラーリングされた論理ボリュームを指定すると、 入出力読み取り操作のパフォーマンスが向上する可能性があります。なぜなら、複数のコピーがある場合、 システムはこの論理ボリューム用のディスクの中で最も使用頻度の低いディスクを選んで読み取り操作を実行できるからです。

順次読み取りスケジューリング・ポリシーを使った並列書き込み
これは、論理区画のすべての物理区画での書き込み操作を同時に開始します。 1 次読み取りコピーが常に最初に読み取られます。 その読み取り操作が失敗した場合は、次のコピーが読み取られます。 次のコピーの読み取り再試行の操作中、障害のある 1 次コピーは、ハードウェアを再配置して LVM によって修正されます。こうして、不良ブロックは今後アクセスされた場合に備えてパッチをあてられます。

ラウンドロビン読み取りスケジューリング・ポリシーを使った並列書き込み
これは、論理区画のすべての物理区画での書き込み操作を同時に開始します。 読み取りは、ミラーリングされた複数のコピーを前後に切り替えながら行われます。

不良ブロック・ポリシー
ボリューム・グループに不良ブロックの再配置が使用可能かどうかを指示します。 デフォルト値は *yes* です。 ボリューム・グループに対して値が yes に設定されている場合、 不良ブロックを再配置できます。 値が no に設定されている場合、 ポリシーは論理ボリューム設定を指定変更します。 値が変更された場合、すべての論理ボリュームは 直前の設定を使用して継続します。 値は、要求された入出力を再配置されたブロックに送信する必要があるかどうかを 指示します。 値が yes に設定されている場合、ボリューム・グループは不良ブロックの 再配置を許可します。 値が no に設定されている場合、不良ブロック割り振りは完了しません。 LVM は、ハードウェアの再配置が失敗したときのみ、ソフトウェア再配置を行います。 それ以外の場合は、LVM 不良ブロック再配置 (BBR) フラグは効果をもちません。

注 : ボリューム・グループおよび論理ボリュームに対する不良ブロック・ポリシー設定値 がいずれも *yes* に設定されて いない場合は、不良ブロックの再配置は使用不可になります。

論理ボリュームのミラー書き込み整合性ポリシー
ミラー書き込み整合性 (MWC) を ON にすると、システムまたはボリューム・グループが正しくシャットダウンされない場合に不整合になった論理区画を識別します。 ボリューム・グループをオンラインに戻すとき、論理区画を整合化するのにこの情報が使用されます。 これをアクティブ *MWC* と呼びます。

論理ボリュームがアクティブ MWC を使用している場合には、この論理ボリュームに対する要求は、MWC キャッシュ・ブロックがターゲット物理ボリューム上で更新されるまで、スケジューリング・レイヤー内に保留されます。 MWC キャッシュ・ブロックの更新が完了すると、物理データの書き込み操作要求が進行します。 書き込みを進行させるには、データが実際に置かれているディスクにのみこの MWC キャッシュ・ブロックがあらかじめ書き込まれなければなりません。

アクティブ MWC を使用すると、システム・パフォーマンスに悪影響があることがあります。 書き込み要求がアクティブであることを論理トラック・グループ (LTG) 内にロギングまたはジャーナリングするための

オーバーヘッドが原因で、このような悪影響が生じます。 ボリューム・グループ に使用できる LTG サイズは 128 K、256 K、512 K、1024 K、2 MB、4 MB、8 MB、および 16 MB です。

注 : LTG を 128K より大きくするには、ボリューム・グループに入っているディスクが、そのディスクのストラテジー・ルーチンから出されるこのサイズの入出力要求をサポートしていなければなりません。 LTG は、論理ボリューム内に収められている連続したブロックであり、LTG のサイズに合わせて位置設定されます。 このオーバーヘッドはミラーリング書き込みでのみかかります。

すべてのミラーへの書き込みが完了する前にシステムまたはボリューム・グループが破損した場合にのみ、ミラーどうしの互いのデータ整合性を確保する必要があります。 1 つのボリューム・グループ内のすべての論理ボリュームは MWC ログを共用します。 MWC ログは、各ディスクの外部エッジ上で維持されます。 また、ディスク上の MWC ログに論理ボリュームを近接させるため、アクティブ MWC を使用する論理ボリュームはディスクの外部エッジに置かれます。

MWC をパッシブに設定すると、論理ボリュームがオープンされたことがボリューム・グループでログに記録されます。 破損後にボリューム・グループをオンに変更すると、 論理ボリュームの自動同期が強制的に開始されます。 読み取られたブロックを、論理ボリューム内の他のミラーに伝搬する読み取りリカバリー・ポリシーのコピーが使われて、 強制同期が進行する際に整合性が保たれます。 このポリシーは、BIG ボリューム・グループ・タイプでのみサポートされます。

MWC を OFF にすると、システムまたはボリューム・グループの破損が起きた場合、ミラーリング論理ボリュームのミラーは不整合になってしまうことがあります。 ミラーの整合性を保つための自動保護機能はありません。 破損時に書き込みが未完了であった場合に、 その後ボリューム・グループをオンに変更すると、ミラーが不整合なデータを持っていることがあります。 破損後には、MWC を OFF にされたすべてのミラーリング論理ボリュームは、その論理ボリューム内のデータを使用する前に強制同期を実行する必要があります。 次に例を示します。

```
syncvg -f -l LTV name
```

ページング・スペースのように、論理ボリュームがオープンされている間のみ有効である内容をもつ論理ボリュームは強制同期の例外です。

書き込みという点では、ミラーリングされた論理ボリュームの場合もミラーリングされていない論理ボリュームの場合と同じです。 LVM が完全に書き込み要求を終了すると、LVM の下のすべてのドライブへのデータの書き込みが完了します。 書き込みに関する iodone が LVM から発行されるまで、その書き込みの結果は分かりません。 これが完了すると、破損からのリカバリーはもう必要ありません。 マシン破損時に完了 (iodone) していなかった書き込み中のすべてのブロックは、MWC の設定またはミラーリングされているかどうかとは無関係に、 チェックして再書き込みする必要があります。

ミラーリングされた論理ボリュームはミラーリングされていない論理ボリュームと同じなので、 最新データのようなものはありません。 データの妥当性を重視するアプリケーションではすべて、 論理ボリュームがミラーリングされていたかどうかに関係なく、 ボリューム・グループまたはシステムの 破損以前に書き込みが完了しなかった場合、 そのような未決または継続中の書き込みのデータの妥当性を確かめる必要があります。

アクティブおよびパッシブの MWC がミラーを整合化するのは、破損後に、1 つのミラーを取り込んでからそのデータを他のミラーに伝搬することによって、 ボリューム・グループがオンラインに変更される時点です。 このような MWC ポリシーでは、最新データをトラッキングしません。 アクティブ MWC は現在書き込み中の LTG しかトラッキングしないので、 どのミラーにも最新のデータが伝搬されるとは限りません。 パッシブの MWC は、破損の後には propagate-on-read モードに入ってミラーを整合化します。 破損後のデータの妥当性を判別するのは、LVM の上位のアプリケーションです。 LVM の観点では、破損の時点で未解決であった書き込みをアプリケーションが常にすべて再発行する場合には、 その書き込みが完了すれば、不整合であったかもしれないミラーでも整合した状態になります (ただし、破損の時点で未解決であった同じブロックが、その破損後に書き込まれる場合に限ります)。

注 : 破損の後には、使用前に MWC をオンにして強制同期を行うか、 あるいはパッシブの MWC をオンにして、JFS ログまたはファイルシステムの 入っているミラーリング論理ボリュームを同期する必要があります。

ディスク間割り振りポリシー

ディスク間割り振りポリシーは、論理ボリュームの物理区画が置かれるディスクの数を指定します。

論理ボリュームの物理区画は単一のディスク上に置かれることも、 ボリューム・グループのすべてのディスク全体にスプレッドされることもあります。 以下のオプションは、**mklv** および **chlv** コマンドと一緒に指定して、 ディスク間ポリシーを判断するのに使用します。

- Range オプションは、論理ボリュームの単一物理コピーに使用するディスクの数を決定します。
- Strict オプションによって、複数のコピーが同じ物理ボリュームを占有した場合の **mklv** 操作の成否を判別します。
- Super Strict オプションは、ある特定のミラーに割り振られている 区画は別のミラーからの区画と物理ボリュームを共用できないことを指定します。
- ストライピングされた論理ボリュームが持てるのは、最大範囲と super strict ディスク間ポリシーだけです。

論理ボリュームの単一コピーのディスク間設定

最小ディスク間設定 (`Range = minimum`) を選択すると、論理ボリュームに割り当てられた物理区画は、単一ディスク上に配置され、可用性が増します。 最大ディスク間設定 (`Range = maximum`) を選択すると、物理区画は複数のディスクに配置され、パフォーマンスが向上します。

ミラーリングされていない論理ボリュームの場合には、minimum の設定を使用すると可用性 (ハードウェア障害時のデータへのアクセス) が最も高くなります。 minimum の設定は、可能であれば 1 つの物理ボリュームにこの論理ボリュームのすべてのオリジナルの物理区画を収容することを指示します。 割り振りプログラムが複数の物理ボリュームを使用する必要がある場合には、最小数を使用して、他のパラメーターとの整合性を維持します。

最小数の物理ボリュームを使用することによって、 ディスク障害によるデータの損失リスクが軽減します。 単一物理コピーに複数の物理ボリュームを使用すると、そのリスクが増大します。 4 つの物理ボリュームにわたって存在する非ミラーリング論理ボリュームの方が、1 つの物理ボリュームに含まれる論理ボリュームよりも、1 つの物理ボリュームの障害によってデータを失う可能性が 4 倍高くなることになります。

以下の図は、最小ディスク間割り振りポリシーを図示しています。

図 *3.* 最小ディスク間割り振りポリシー

3 つのディスクがある。 1 つのディスクには 3 つの物理区画が含まれており、他の 2 つには物理区画がない。

他の制約を考慮に入れて、最大の設定では、可能な限り均等に、物理ボリュームの物理区画ができるだけ多くの物理ボリュームにスプレッドされます。 これはパフォーマンス指向のオプションです。物理区画をいくつかのディスクに分散することによって、論理ボリュームに対する平均アクセス時間を短縮できる傾向があるからです。 可用性を向上するため、最大設定はミラーリング論理ボリュームでのみ使用されます。

以下の図は、最大ディスク間割り振りポリシーを図示しています。

図 *4.* 最大ディスク間割り振りポリシー

3 つのディスクがある。そのいずれにも 1 つの物理区画がある。

これらの定義はまた、既存の論理ボリュームを拡張したりコピーするときにも 適用できます。 新規の物理区画の割り振りは、現在の割り振りポリシーと既に使用されている物理区画が存在するロケーションによって決定されます。

関連概念

論理ボリューム・コピーのディスク間設定

論理ボリュームの単一コピーのディスク上での割り振りは比較的単純です。

論理ボリューム・コピーのディスク間設定

論理ボリュームの単一コピーのディスク上での割り振りは比較的単純です。

ただし、ミラーリング・コピーを作成すると、そのコピーの割り振りはいくぶん複雑になります。 下図に、最初の論理ボリュームのインスタンスに対して最小および最大のディスク間 (Range) 設定を使用し、ミラーリング論理ボリュームのコピーに対して Strict 設定を使用した場合の例を表示します。

例えば、論理ボリュームのミラーリング・コピーが複数ある場合、 最小設定では、可能であれば、論理ボリュームの最初のインスタンスの入った物理区画を単一の物理ボリュームに割り振ります。 次に、Strict オプションの設定に応じて、追加の 1 つ以上 のコピーが、同一または別の物理ボリュームに割り振られます。 言い換えると、Strict オプションなどの他のパラメーターによる制約内で、アルゴリズムは可能な限り最小数の物理ボリュームを使用して、すべての物理区画を保持します。

`Strict = y` の設定は、 論理区画の各コピーが異なる物理ボリュームに配置されることを意味します。 `Strict = n` の設定は、コピーの配置場所が異なる物理ボリュームには制限されないことを意味します。 比較すると、`Super Strict` オプションの場合は、 ある特定のミラーの物理区画を、同じ論理ボリュームの別のミラーの物理区画と同じディスク上に 置くことは認められません。

注 : ボリューム・グループ内の物理ボリューム数が、選択した論理区画あたりのコピー数よりも少ない場合には、Strict を **n** に設定します。Strict が **y** に設定された場合は、論理ボリュームを作成しようとすると、エラー・メッセージが戻されます。

以下の図は、異なる Strict 設定を持つ最小ディスク間割り振りポリシーを図示しています。

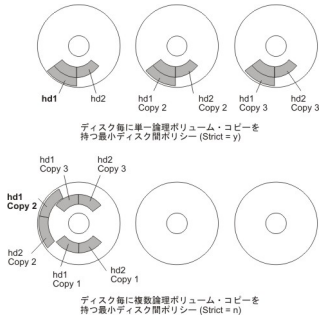


図 *5.* 最小ディスク間割り振りポリシー */Strict*

以下の図は、異なる *Strict* 設定を持つ最大ディスク間割り振りポリシーを図示しています。

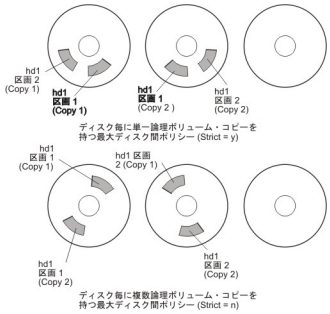


図 *6.* 最大ディスク間割り振りポリシー */Strict*

関連概念

論理ボリュームの単一コピーのディスク間設定

最小ディスク間設定 (`Range = minimum`) を選択すると、論理ボリュームに割り当てられた物理区画は、単一ディスク上に配置され、可用性が増します。 最大ディスク間設定 (`Range = maximum`) を選択すると、物理区画は複数のディスクに配置され、パフォーマンスが向上します。

論理ボリュームごとのディスク内割り振りポリシー

ディスク内割り振りポリシーの選択は、物理区画を配置できるディスクの 5 つの領域に基づいています。

ある特定の物理区画の位置が物理ボリュームのセンターに近ければ近いほど、平均シーク・タイムは短くなります。なぜなら、センターはディスク上の他のどの部分からでも最短の平均シーク距離にあるからです。

ファイルシステムのログは、オペレーティング・システムによって頻繁に使用されるため、物理ボリュームのセンターに割り振る候補としては最適と言えます。 逆にブート論理ボリュームはあまり使用されないので、物理ボリュームのエッジまたはミドルに割り振ります。

一般的な規則として、絶対的に入出力の回数が多いもの、または重要なアプリケーションの実行中の入出力の回数が多いものほど、論理ボリュームの物理区画を物理ボリュームのセンターに近づけて割り振る必要があります。

このルールには、次のような重要な例外が 1 つあります。 ミラーリング書き込み整合性 (MWC) の設定がオンになっているミラーリングされた論理ボリュームは、 システムが MWC データを書き込む場所である、外部エッジにあります。 ミラーリングが有効でない場合、MWC は適用されず、パフォーマンスには影響しません。

物理区画を配置できるディスクの 5 つの領域を、以下に示します。

1. 外部エッジ
2. 内部エッジ
3. 外部ミドル
4. 内部ミドル
5. センター

エッジの区画の平均シーク・タイムは最も遅く、その結果、アプリケーションが使用すると応答時間が長くなります。 センターの区画は最もシーク・タイムが短く、一般的に、アプリケーションが使用すると応答時間が最短になります。 しかしセンターには、物理ボリュームの区画は、他の領域よりも少ししか置けません。

割り振りポリシーの組み合わせ

互換性のないディスク間ポリシーとディスク内ポリシーを選択すると、 予期できない結果が生じます。

システムは 1 つのポリシーを他に優先させて物理区画を割り当てます。 例えば、ディスク内ポリシーにセンターを、ディスク間ポリシーに最小を選択した場合、 ディスク間ポリシーが優先されます。 システムは区画がセンター領域に入りきれない場合でも、 可能であれば 1 つのディスク上に論理ボリュームのすべての区画を配置します。 インプリメントする前に、選択したポリシーの対話を必ず理解しておいてください。

正確に割り当てるためのマップ・ファイルの使用

ディスク間ポリシーおよびディスク内ポリシーが提供するデフォルト・オプションがユーザーのニーズに対して十分でない場合には、 物理区画の正確な順番とロケーションを論理ボリュームに対して指定するために、 マップ・ファイルの作成を検討してください。

マップ・ファイルを作成するためには、SMIT、または **mklv -m** コマンドを使用できます。

例えば、rootvg の lv06 と呼ばれる 10 個の区画からなる論理ボリュームを、hdisk1 の区画 1 から 3、41 から 45、および 50 から 60 に作成するためには、以下のコマンド・ラインの手順を使用できます。

1. 使用しようとする物理区画に何も割り当てられていないことを確認するためには、次のように入力します。

   ```
   lspv -p hdisk1
   ```

2. 以下のような内容の、/tmp/mymap1 などのファイルを作成します。

   ```
   hdisk1:1-3
   hdisk1:41-45
   hdisk1:50-60
   ```

mklv コマンドは、マップ・ファイルに示された順番で物理区画を割り当てます。 **mklv** コマンドで指定した論理ボリューム全体を割り当てられるだけの十分な物理区画をマップ・ファイルに記述したことを確認してください。 (必要以上の物理区画をリストすることができます。)

3. 次のコマンドをタイプします。

```
mklv -t jfs -y lv06 -m /tmp/mymap1 rootvg 10
```

ストライピングされた論理ボリューム・ストラテジーの開発

ストライピングされた論理ボリュームは、アクセス頻度が高くパフォーマンスが重要な大きな順次ファイルシステムで使用されます。 ストライピングはパフォーマンスの向上を目的としています。

注 : ダンプ・スペースまたはブート論理ボリュームはストライピングされません。 ブート論理ボリュームは連続する物理区画でなければなりません。

lv07 という 12 区画のストライピングされた論理ボリュームを、hdisk1、hdisk2、および hdisk3 全体で 16KB のストライプ・サイズ (ストリップ・サイズにアレイ 内のディスク数を掛けると、ストライプ・サイズになる) を持つ VGName 内に作成するには、次のように入力します。

```
mklv -y lv07 -S 16K VGName 12 hdisk1 hdisk2 hdisk3
```

lv08 という 12 区画のストライピングされた論理ボリュームを、VGName 内の 3 つのディスク全体のストリップ・サイズが 8 KB の VGName 内に作成するには、次のように入力します。

```
mklv -y lv08 -S 8K -u 3 VGName 12
```

ディスク・ストライピングを使用してパフォーマンスを向上させる方法の詳細については、「パフォーマンス・マネージメント」を参照してください。

書き込み検査ポリシー

書き込み検査オプションを使用すると、すべての書き込み操作は、書き込みの成功を検査するためのその直後の読み取り操作によって検証されます。

書き込み操作が成功しなかった場合には、エラー・メッセージが表示されます。 このポリシーは可用性を向上させますが、読み取りに余分な時間が必要なためパフォーマンスは低下します。 論理ボリュームにおける書き込み検査ポリシーは、**mklv** コマンドを使用して論理ボリュームを作成するとき、または後で **chlv** コマンドによってそれを変更するときに、指定できます。

ホット・スペア・ディスク・ポリシー

ディスクを、ミラーリングされた論理ボリュームを持つボリューム・グループのホット・スペア・ディスクとして指定できます。

ホット・スペア・ディスクとして使用する ディスクを指定する時に、1 つ以上のディスクに障害が出始めたときに使用するポリシーを指定し、 さらに同期特性も指定できます。

ボリューム・グループに物理ボリュームを (ホット・スペア・ディスクとしてマークされるように) 追加する場合、そのディスクが少なくともそのボリューム・グループ内の既存の最小ディスクと同じ容量を持っていなければなりません。 このフィーチャーがインプリメントされると 、 ミラー書き込み整合性 (MWC) 書き込み障害によって物理ボリュームの欠落がマークされた場合、データがホット・スペア・ディスクに移行されます。

ホット・スペア・ディスク・サポートを使用可能にするためのコマンド **chvg** および **chpv** には、ご使用のサイトにおけるこのフィーチャーのインプリメント 方法に関するいくつかのオプションが用意されています。構文は次のとおりです。

```
chvg -hhotsparepolicy -ssyncpolicy VolumeGroup
```

ここで *hotsparepolicy* は、ディスクに障害が発生しているときに以下のポリシーの中のどれを使用するのかを決定します。

y
1 つの障害のあるディスクから 1 つのスペア・ディスクへ区画を自動的に移行します。 ホット・スペア・ディスクのプールから、その障害のあるディスクを置換するのに十分な大きさを持つディスクのうち、最小のものが使用されます。

Y
障害のあるディスクから区画を自動的に移行しますが、ホット・スペア・ディスク・プール全体が使用される場合があります。

n
自動的には移しません (デフォルト)。

r
このボリューム・グループ用のホット・スペア・ディスクのプールからすべてのディスクを 除去します。

syncpolicy 引数は、あらゆる不整合区画を自動的に同期させるかどうかを決定します。

y
自動的に不整合区画を同期化しようとします。

n
自動的に不整合区画を同期化しようとはしません。 (このオプションがデフォルトです。)

VolumeGroup 引数は、関連付けられているミラーリングされたボリューム・グループの名前を指定します。

論理ボリュームのホット・スポット管理
論理ボリュームで発生したホット・スポット 問題を識別して、システムの使用を妨げずにそれらの問題を改善できます。

ホット・スポット問題は、ディスク上の一部の論理区画に対するディスク入出力が システム・パフォーマンスを著しく損なうほど多い場合に発生します。

この問題の解決に向けた最初のステップは、それを識別することです。 デフォルトでは、システムは論理ボリュームの使用に関する統計情報を収集しません。 それらの統計情報の収集を可能にした後で、初めて **lvmstat** コマンドを実行すると、システムは前のシステム・リブート以降のカウンター値を表示します。 それ以降は、**lvmstat** コマンドを入力するたびに、直前の **lvmstat** コマンド以降の差分がシステムによって表示されます。

lvmstat コマンドの出力を解釈すれば、トラフィックが最も激しい論理区画を識別できます。 非常に頻繁に使用されている論理区画が 1 つの物理ディスク上に複数あり、それらを複数の 使用可能なディスクに分散させたい場合、 **migratelp** コマンドを使用してそれらの 論理区画を他の複数の物理ディスクに移動することができます。

以下の例では、統計情報の収集が使用可能にされており、基本的な統計情報を収集するために **lvmstat** コマンドが繰り返し使用されています。

```
# lvmstat -v rootvg -e
# lvmstat -v rootvg -C
# lvmstat -v rootvg
```

出力は次のようになります。

```
Logical Volume            iocnt      Kb_read     Kb_wrtn      Kbps
  hd8                         4            0          16      0.00
  paging01                    0            0           0      0.00
  lv01                        0            0           0      0.00
  hd1                         0            0           0      0.00
  hd3                         0            0           0      0.00
  hd9var                      0            0           0      0.00
  hd2                         0            0           0      0.00
  hd4                         0            0           0      0.00
  hd6                         0            0           0      0.00
  hd5                         0            0           0      0.00
```

上の出力では、すべてのカウンターがゼロにリセットされていることが示されています。 次の例では、 データは最初に /unix ディレクトリーから /tmp ディレクトリーにコピーされます。 **lvmstat** コマンドの出力には、rootvg のアクティビティーが 反映されています。

```
# cp -p /unix /tmp
# lvmstat -v rootvg

Logical Volume            iocnt      Kb_read     Kb_wrtn      Kbps
  hd3                       296            0        6916      0.04
  hd8                        47            0         188      0.00
  hd4                        29            0         128      0.00
  hd2                        16            0          72      0.00
  paging01                    0            0           0      0.00
  lv01                        0            0           0      0.00
  hd1                         0            0           0      0.00
  hd9var                      0            0           0      0.00
  hd6                         0            0           0      0.00
  hd5                         0            0           0      0.00
```

この出力には、**hd3** 論理ボリューム (/tmp ディレクトリーにマウント済み)、**hd8** (JFS ログ論理ボリューム)、**hd4** (/ (ルート))、**hd2** (/usr ディレクトリー)、および **hd9var** (/var ディレクトリー) のアクティビティーが示されています。 次の出力には、**hd3** および **hd2** に関する詳細が示されています。

```
# lvmstat -l hd3

Log_part    mirror#    iocnt    Kb_read    Kb_wrtn     Kbps
       1          1      299          0       6896     0.04
       3          1        4          0         52     0.00
       2          1        0          0          0     0.00
       4          1        0          0          0     0.00
# lvmstat -l hd2
Log_part    mirror#    iocnt    Kb_read    Kb_wrtn     Kbps
       2          1        9          0         52     0.00
       3          1        9          0         36     0.00
       7          1        9          0         36     0.00
       4          1        4          0         16     0.00
       9          1        1          0          4     0.00
      14          1        1          0          4     0.00
       1          1        0          0          0     0.00
```

ボリューム・グループに関する出力には、論理ボリュームのすべての入出力アクティビティーの 要約が示されます。 これは、入出力要求の数 (iocnt)、読み取りと書き込みのキロバイト数 (Kb_read と Kb_wrtn のそれぞれ)、および KB/秒単位での転送済みデータ (Kbps) に分かれています。 論理ボリュームに関する情報を要求した場合、受け取る情報は同じですが、論理区画ごとに別々に受け取ることになります。 論理ボリュームをミラーリングしている場合は、それぞれのミラー・ボリュームごとに統計情報を受け取ります。 すぐ上の出力例では、アクティビティーがまったくない 論理区画についての数行が省略されています。 iocnt 列では、出力は常に降順にソートされます。

migratelp コマンドは、パラメーターとして論理ボリュームの名前、論理区画の数 (**lvmstat** 出力に表示されるもの)、および特定のミラー・コピーのオプショナル 番号を使用します。 情報が省略されている場合は、最初のミラー・コピーが使用されます。 移動用のターゲット物理ボリュームを指定する必要があります。 その際、ターゲット物理区画番号も指定できます。 正常終了の場合、出力は次のようになります。

```
# migratelp hd3/1 hdisk1/109
  migratelp: Mirror copy 1 of logical partition 1 of logical volume
        hd3 migrated to physical partition 109 of hdisk1.
```

ホット・スポット・フィーチャーを使用可能にすると、論理ボリュームまたはボリューム・グループのいずれかについて、レポートと統計情報を定義し、その統計情報を表示させ、移行する論理区画を選択し、宛先となる物理区画を指定して、変更をコミットする前に情報を検証できます。

ボリューム・グループ・ポリシーの実装

使用するボリューム・グループ・ポリシーを決定したら、コマンド・ラインに **lspv** コマンドを入力して、現在の構成を分析してください。

標準構成は、同一のディスク・アダプターとその 他のサポート・ハードウェアに付加された複数の物理ボリュームの入った単一ボリューム・グループを提供します。 標準構成では、クォーラム (定足数) ボリュー

ム・グループを構成するディスクが多ければ多いほど、ディスク障害時にクォーラム (定足数) が残っている可能性が増大します。 非クォーラム (定足数) グループの場合は、最低 2 つのディスクでボリューム・グループを構成する必要があります。 ボリューム・グループ・ポリシーの変更を有効にするには、次のことを実行してください。

1. **lspv** コマンド出力を使用して、割り当て済み物理ボリュームと空き物理ボリュームをチェック。
2. 1 つ以上の物理ボリュームを追加して、クォーラム (定足数) を確保。
3. ボリューム・グループを非クォーラム (定足数) 状況に変更する
4. 高可用性を確保するために必要な場合にのみ、ハードウェアを再構成してください。 これについては、ご使用のシステムの「サービス・ガイド」を参照してください。

ページング・スペースおよび仮想メモリー

AIX では、仮想メモリーを使用して、システムで物理的に使用可能なメモリー以上のメモリーにアドレッシングします。

RAM 内またはディスク上のメモリー・ページの管理は、仮想メモリー・マネージャー (VMM) によって行われます。 仮想メモリーのセグメントは、ページ と呼ばれる単位に分割されます。 ページング・スペースは、論理ボリュームの一種で、仮想メモリー内に あるが現在はアクセスされていない情報が保管されるディスク・スペースを割り当てられています。 この論理ボリュームは、属性タイプがページングで、通常は単にページング・スペースまたはスワップ・スペース と呼ばれます。 システム内の RAM の空き容量が少ない場合は、メモリーを他のアクティビティーに 解放するために、最近使用されていないプログラムやデータがメモリーからページング・スペースに移されます。

ページング・スペースの概念

ページング・スペース は論理ボリュームの一種で、仮想メモリー内に あるが現在はアクセスされていない情報が保管されるディスク・スペースを割り当てられています。

この論理ボリュームは、属性タイプがページングで、通常は単にページング・スペースまたはスワップ・スペース と呼ばれます。 システム内の RAM の空き容量が少ない場合は、メモリーを他のアクティビティーに解放するために、最近使用されていないプログラムやデータがメモリーからページング・スペースに移されます。

もう 1 つのタイプのページング・スペースが使用できます。 このページング・スペースには、ページング・スペース・ストレージに NFS サーバーを使用するデバイスを介してアクセスすることができます。 NFS クライアントがこのページング・スペースにアクセスするには、NFS サーバーでファイルを作成し、そのファイルをそのクライアントにエクスポートしておく必要があります。 ファイル・サイズは、そのクライアントに使用するページング・スペース・サイズを表します。

必要なページング・スペースの量は、システム上で実行されるアクティビティーのタイプに 応じて異なります。 ページング・スペースが少なくなると、プロセスが失われる可能性があります。また、ページング・スペースが使い果たされると、システムが誤動作する可能性があります。 ページング・スペースの不足条件が検出されたら、追加のページング・スペースを定義してください。

論理ボリューム・ページング・スペースは、新規のページング・スペース論理ボリュームを作成するか、あるいは、既存のページング・スペース論理ボリュームのサイズを大きくすることで定義します。 NFS ページング・スペースのサイズを大きくするには、サーバー上の適切なアクションによって、サーバーにあるファイルを大きくする必要があります。

システムでページングに 使用できる合計スペースは、すべてのアクティブなページング ・スペース論理ボリュームのサイズを 合計したものになります。

ページング・スペース割り振りポリシー

PSALLOC 環境変数は、 遅延と早期のどちらのページング・スペース割り当てアルゴリズムを使用するのかを決定します。

AIX では、ページング・スペースの割り当てに 2 つのモードを使用します。 デフォルトは遅延です。早期ページング・スペース割り当てモードには、*PSALLOC* 環境変数の値を変更すれば切り替えられますが、このような変更を行う前に考慮しなければならない要因がいくつかあります。 早期割り当てアルゴリズムを

使用する場合、最悪のシナリオでは、 使用可能なページング・スペースをすべて使い果たしてしまってシステムを破損させる可能性があります。

遅延および早期ページング・スペース割り当ての比較

オペレーティング・システムは、*PSALLOC* 環境変数を使用して、メモリーおよびページング・スペース割り当てに使用するメカニズムを決定します。

PSALLOC 環境変数を設定しない場合、またはこの変数を `null` に設定したり早期以外の値に設定したりした場合、システムは、デフォルトの遅延割り当てアルゴリズムを使用します。

遅延割り当てアルゴリズムは、ディスク・リソースの有効使用を支援し、リソース管理に疎割り当てアルゴリズムを優先的に使用するアプリケーションをサポートします。このアルゴリズムでは、メモリー要求が出された時点ではページング・スペースの予約を行いません。ページング・スペースのディスク・ブロック割り当ては、要求されたページのページアウトが必要になるまで遅延されます。 一部のプログラムでは、大量の仮想メモリーを割り当ててから、 そのメモリーのほんの一部分だけを使用します。 このようなプログラムの例には、データ構造として疎ベクトルや疎マトリックスを使用する技術アプリケーションがあります。 遅延割り当てアルゴリズムは、オペレーティング・システム内のカーネルのようなリアルタイムの要求時ページング・カーネルの場合にも非常に効果的です。

このページング・スペースは、特に、ページングがほとんど行われない大きな実メモリーをもつシステムではまったく使用されない場合があります。遅延アルゴリズムでは、要求されたページのページアウトが必要になるまでページング・スペースの割り当てを遅らせます。その結果、無駄なページング・スペース割り当てはなくなります。この遅延割り当てにより、遅延アルゴリズムがシステムで 使用可能なスペースより多くのページング・スペースを割り当てようとする状況が生じる場合があります。 この状況をページング・スペースのオーバーコミットメントと呼びます。

オーバーコミットメントのシナリオ (ページング・スペースが使い尽くされ、ページをページアウトするためのページング・スペース・ディスク・ブロックの割り当てが試みられる状況) では、結果的に障害が発生します。 オペレーティング・システムは、ページング・スペースのオーバーコミットの影響を受けるプロセスを強制終了することによって、全面的なシステム障害の発生を回避します。 フリーのページング・スペースの量が不足していることをプロセスに通知するために、**SIGDANGER** シグナルが送信されます。 ページング・スペース状態がさらに危険な状態になると、**SIGDANGER** シグナルを受信していないプロセスが選択され、**SIGKILL** シグナルが送信されます。

PSALLOC 環境変数を使用して、早期割り当てアルゴリズムに切り替えられます。このアルゴリズムでは、実行中のプロセスへのページング・スペースの割り当ては、メモリーが要求された時点で行われます。 要求時に使用できるページング・スペースが不足していると、早期割り当てメカニズムは、メモリー要求に応じることができません。

PSALLOC 環境変数を早期に設定すると、その環境内でその時点で開始されたプログラムは (現在実行中のプロセスを除いて) すべて、早期割り当て環境内で実行します。 早期割り当て環境では、要求が出された時点で十分なページング・スペースを予約することができなかった場合、**malloc** サブルーチンや **brk** サブルーチンなどのインターフェースは 失敗します。

ページング・スペースの不足条件が発生しても、早期割り当て環境モードで実行されるプロセスには **SIGKILL** シグナルは送信されません。

PSALLOC 環境変数を早期に変更する方法は、変更を適用する範囲の広さによって異なります。

早期割り当て環境に切り替えると、以下のメモリー割り当てインターフェース・サブルーチンが影響を受けます。

- **malloc**
- **free**
- **calloc**
- **realloc**
- **brk**
- **sbrk**
- **shmget**
- **shmctl**

関連タスク

PSALLOC 環境変数の早期割り当てモードへの構成

オペレーティング・システムは、*PSALLOC* 環境変数を使用して、メモリーおよびページング・スペース割り当てに使用するメカニズムを決定します。

早期割り当てモード

早期割り当てアルゴリズムによって、メモリー割り当て要求で要求した量と同じだけのページング・スペースが保証されます。 したがって、システム・ディスクにおける正しいページング・スペースの割り当ては、効率的な操作を行うために重要です。

使用可能なページング・スペースがある一定のしきい値以下になると、新規プロセスは開始することができなくなり、現在実行中のプロセスもそれ以上のメモリーを確保できなくなる可能性があります。 デフォルトの遅延割り当てモードで実行中のプロセスはどれも、**SIGKILL** シグナル・メカニズムの影響を非常に受けやすくなります。さらに、オペレーティング・システム・カーネルでメモリー割り当てを要求することがあるため、使用可能なページング・スペースをすべて使い果たしてしまうと、システムの破損が生じる可能性があります。

システム全体で早期割り当てモードを使用する前に、そのシステムに適した量のページング・スペースを定義しておくことが非常に重要です。 早期割り当てモードに必要なページング・スペースは、ほとんどの場合、デフォルトの遅延割り当てモードに必要なページング・スペースよりも大きくなります。定義するページング・スペースの量は、システムの用途と実行するプログラムの内容によって決まります。 システムに適した妥当な比率を決定する場合、物理メモリーの量の 4 倍のページング・スペースを定義することから開始するとよいでしょう。

あるアプリケーションでは、早期割り当てモードで実行する場合に、非常に大量のページング・スペースを使用することがあります。 現在、AIXwindows サーバーは、アプリケーションが早期割り当てモードで実行する場合に 250 MB より大きいページング・スペースを必要とします。 どのアプリケーションの場合も、必要なページング・スペースは、そのアプリケーションの作成方法とその実行方法によって決まります。

ページング・スペースおよび処理メモリーの使用量を表示するすべてのコマンドとサブルーチンには、早期割り当てモードで割り当てられるページング・スペースが組み込まれます。 **lsps** コマンドは、**-s** フラグを使用してページング・スペースの割り当ての合計を表示しますが、これには、早期割り当てモードで割り当てられるページング・スペースも組み込まれます。

ページング・スペースのデフォルトのサイズ

デフォルトのページング・スペース・サイズは、次の規準に従って、AIX インストールのシステム・カスタマイズ・フェーズの際に決定されます。

- ページング・スペースは、少なくとも 64 MB を使用できる hd6 を除き、少なくとも 16 MB を使用できます。
- ページング・スペースは、合計のディスク・スペースの 20% までしか使用できません。
- 実メモリーが 256 MB より少ない場合のページング・スペースは、実メモリーの 2 倍の量になります。
- 実メモリーが 256 MB 以上の場合のページング・スペースは、512 MB になります。

ページング・スペース用のファイル、コマンド、およびオプション

/etc/swapspaces ファイルは、ページング・スペースおよび ページング・スペースの属性を指定します。

ページング・スペースは、**mkps** コマンド によって作成される際に /etc/swapspaces ファイルに追加され、**rmps** コマンド によって削除される際に /etc/swapspaces ファイルから削除されます。 ファイル内のページング・スペース属性は、**chps -a** コマンド または **chps -c** コマンドによって変更されます。 以前のフォーマット (ここで、チェックサム・サイズおよびスタンザ内の自動的スワップの属性はない) を 使用するファイルは引き続きサポートされます。 ページング・スペースのサイズが大きすぎる 場合は、**chps -d** コマンドを使用して リブートせずに、ページング・スペースから論理区画を減算することができます。

ページング・スペースは、次のコマンドを使用して管理します。

項目	説明
chps	ページング・スペースの属性を変更します。
lsps	ページング・スペースの特性を表示します。
mkps	追加ページング・スペースを追加します。 **mkps** コマンドは、ページング・スペース論理ボリュームの作成時に特定の 1 組のオプションを指定した **mklv** コマンドを使用します。 NFS ページング・スペースを作成するために、**mkps** コマンドは、 別のオプション・セットを指定して **mkdev** コマンドを使用します。 NFS ページング・スペースの場合、**mkps** コマンドには、NFS サーバーのホスト名、ならびに、そのサーバーからエクスポートされるファイルのパス名を指定する必要があります。
rmps	非アクティブ・ページング・スペースを除去します。
swapoff	システムをリブートせずに、1 つ以上のページング・スペースを非アクティブにします。 ページング・スペース内の情報は、アクティブなページング・スペース・エリアに移動されます。 次に、**rmps** コマンドを使って、 非アクティブにされたページング・スペースを削除することができます。
swapon	ページング・スペースをアクティブにします。 **swapon** コマンドは、初期のページング・スペース・デバイスをアクティブにするための 初期のシステム 初期設定で使用します。 初期設定の後の方のフェーズでは、他のデバイスが使用可能になった時点で、**swapon** コマンドを使用し、追加のページング・スペースをアクティブにして、 ページング・アクティビティーが複数のデバイスにわたって生じるようにします。

paging type オプションは、すべての論理ボリューム・ページング・スペースに必要です。

次のオプションは、論理ボリュームによってページング・パフォーマンスを最大化する場合に使用します。

- ディスクのミドルへの割り当て。これによって、ディスク・アームの移動が減少します。
- 複数のページング・スペースの使用。この場合、それぞれのページング・スペースは、別々の物理ボリュームから割り当てられます。

ページング・スペースの構成

SMIT について多くの構成タスクが実行可能です。 ページ・スペースおよび メモリー割り当ては、**PSALLOC** 環境変数により制御されます。

ページング・スペースの追加と活動化

ページング・スペースをシステムで 使用可能にするには、ページング・スペースを追加し、活動化する必要があります。

ページング・スペースの合計量は、しばしば試行錯誤により決定されます。一般的に使用される指針は、RAM サイズの 2 倍の数値をページング・スペースのターゲットとして使用することです。

SMIT インターフェースを 使用して、コマンド・ラインに以下のどれかの高速パスを入力します。

- 現行ページング・スペースをリストするには、`smit lsps` を入力する。
- ページング・スペースを追加するには、`smit mkps` を入力する。
- ページング・スペースを活動化するには、`smit swapon` を入力する。

ページング・パフォーマンスの向上

ページング・パフォーマンスを向上させるには、複数のページング・スペースを使用し、できる限り別々の物理ボリューム上に配置する必要があります。

ただし、複数のページング・スペースを同じ物理ボリュームに配置できます。 複数の物理ボリュームを使用できますが、システムに完全に精通していない限り、rootvg ボリューム・グループ内のディスクだけを選択するほうが賢明です。

PSALLOC 環境変数の早期割り当てモードへの構成

オペレーティング・システムは、*PSALLOC* 環境変数を使用して、メモリーおよびページング・スペース割り当てに使用するメカニズムを決定します。

デフォルト設定は `late` です。 以下の例で、*PSALLOC* 環境変数を早期に変更するさまざまな方法を示します。 選択する方式は、変更を適用したい範囲によって 異なります。

- シェル・コマンド・ラインに次のコマンドを入力する。

  ```
  PSALLOC=early;export PSALLOC
  ```

  このコマンドを使用すると、以後のすべてのコマンドが、早期割り当てモードで実行されるシェル・セッションから実行されるようになります。
- シェル・リソース・ファイル (`.shrc` または `.kshrc`) に 次のコマンドを追加する。

  ```
  PSALLOC=early;export PSALLOC
  ```

  このエントリーにより、ログイン・シェル以外の、ログイン・セッションのすべてのプロセスが、早期割り当てモードで実行されるようになります。 この方式を使用すると、**SIGKILL** シグナル・メカニズムからプロセスを保護することもできます。
- **putenv** サブルーチンをプログラム内に挿入して、*PSALLOC* 環境変数を早期に設定します。 この方式を採用すると、**exec** サブルーチン の次の呼び出し時に、早期割り当て動作が有効になります。

関連概念

遅延および早期ページング・スペース割り当ての比較

オペレーティング・システムは、*PSALLOC* 環境変数を使用して、メモリーおよびページング・スペース割り当てに使用するメカニズムを決定します。

ページング・スペースの 変更または除去

ページング・スペースの変更は SMIT によって容易に行われますが、ページング・スペースの除去はより危険です。

ページング・スペースの特性の変更は、コマンド・ラインに `smit chps` という SMIT 高速パスを入力して行うことができます。

ページング・スペースの除去は、より多くの危険性を伴う手順です。特に、除去したいページング・スペースが、hd6 などのデフォルトのページング・スペース の場合はなおさらです。 デフォルトのページング・スペースの除去には特別の手順が必要です。これらのページング・スペースは、システムを構成するシェル・スクリプトによって ブート時に活動化されているためです。 デフォルトのページング・スペースの 1 つを除去するためには、これらのスクリプトを変更し、新規のブート・イメージを作成する必要があります。

重要 : デフォルトのページング・スペースを間違って除去すると、システムの 再始動を妨げることがあります。 次の手順は、経験のあるシステム 管理者だけが実行できます。

既存のページング・スペースを除去するには、以下の手順を実行します。

1. root 権限で、コマンド・ラインに以下の SMIT 高速パスを入力して、ページング・スペースを非活動化します。

   ```
   smit swapoff
   ```
2. 削除しようとするページング・スペースがデフォルトのダンプ・デバイスの場合には、ページング・スペースを削除する前に、デフォルトのダンプ・デバイスを別のページング・スペースまたは論理ボリュームに変更する必要があります。

   デフォルトのダンプ・デバイスを変更するには、次のコマンドを入力します。

   ```
   sysdumpdev -P -p /dev/new_dump_device
   ```
3. 以下の高速パスを入力して、ページング・スペースを除去します。

   ```
   smit rmps
   ```

ページング・スペース割り当てモードのプログラミング・インターフェースの使用

ページング・スペース割り当てモードを制御するプログラミング・インターフェースでは、*PSALLOC* 環境変数を使用します。

常に所定のモードで (早期ページング・スペース割り当てを使用して、あるいは使用せずに) アプリケーションが実行されるようにするには、次のように処理します。

1. *PSALLOC* 環境変数の現在の状態を検査する場合は、**getenv** サブルーチンを使用します。
2. *PSALLOC* 環境変数の値がアプリケーションに必要な値でなければ、**setenv** サブルーチンを使用して、その環境変数の値を変更します。

   PSALLOC 環境変数の状態を検査するのは **execve** サブルーチンのみであるため、アプリケーションで受信したものと同じパラメーターと環境のセットを指定して、**execve** サブルーチンを呼び出します。 アプリケーションは、*PSALLOC* 環境変数の状態を再度調べて正しい値を検出したら、正常に続行します。
3. **getenv** サブルーチンで *PSALLOC* 環境変数の現在の状態が正しいことが明らかになれば、変更は不要です。

   アプリケーションは正常に続行します。

hd6 ページング・スペースの 再配置と削減

使用頻度の低いシステム内の他のディスクに強制的にページングとスワッピングを行って、ストレージ・システム・パフォーマンスを向上させるために、デフォルトのページング・スペースを縮小または移動できます。 デフォルトのページング・スペースを縮小または移動すると、hdisk0 上のディスク・スペースの節約にもなります。

ページング・スペースの移動もサイズの縮小も、その根拠は同じです。つまり、比較的使用頻度の低いディスクにページング・スペース・アクティビティーを 移動することです。 インストールのデフォルトでは、ページング論理ボリューム (hd6) を、使用頻度の高い / (ルート) および /usr ファイルシステムの 一部またはすべてが入っているドライブ hdisk0 上に作成します。 最小ディスク間割り振りポリシーを選択し、/ のすべて および大量の /usr が hdisk0 上にある場合、比較的使用頻度の低いディスクにページング・スペースを移動すると、パフォーマンスが著しく向上します。 最大ストレージ・ディスク間割り振りポリシーをインプリメントし、/ および /usr の両方が 複数の物理ボリュームにまたがって分散されている場合でも、hdisk2 (3 つのディスクを想定) に 含まれる、最も使用頻度の高いファイルシステムに 属する論理区画は少なくなる可能性があります。

次の手順は、hd6 ページング・スペースを縮小する方法と、同一ボリューム・グループ内での hd6 ページング・スペースの移動方法について記述しています。

hd6 ページング・スペースの 縮小

以下の手順では、**chps** コマンド を実行して、1 次ページング・スペース、および 1 次と 2 次のダンプ・デバイスなど、既存のページング・スペースを縮小します。

chps コマンドは **shrinkps** スクリプトを呼び出し、それが、システムをブート 不可能な状態にせずに、ページング・スペースを安全に縮小します。 特に、このスクリプトは以下のことを実行します。

1. 同じボリューム内に一時ページング・スペースを作成します。
2. その一時スペースに情報を移動します。
3. 同じボリューム内により小さなページング・スペースを新規作成します。
4. 元のページング・スペースを削除します。

chps コマンドが正常に完了するには、一時ページング・スペースを作成する十分な空きディスク・スペース (論理ボリュームに 割り当てられていない容量) が必要です。 一時ページング・スペースのサイズは、元のページング・スペース内のすべてのページアウトされたページを保持するために必要なサイズです。 1 次ページング・スペースの最小サイズは 32 MB です。 その他のページング・スペースの最小サイズは 16 MB です。

注 : 以下の手順で入出力エラーが発生する場合は、すぐにシャットダウンし、リブートする必要があります。

1. 次のコマンドを使用することにより、物理ボリュームにまたがる論理ボリュームとファイルシステムの分布をチェックします。

   ```
   lspv -l hdiskX
   ```

   hdiskX は物理ボリュームの名前です。
2. ページング・スペースのサイズを縮小するには、コマンド・ラインに次のように入力します。

   ```
   smit chps
   ```

注 : 1 次ページング・スペースはブート・レコード内にハードコーディングされます。 したがって、システムを再始動すると、常に 1 次ページング・スペースが起動します。 **chps** コマンドは 1 次ページング・スペースを非活動化できません。

操作構成の保守作業を優先して行ってください。 システム・チェックを行うと、ページング・スペースの縮小が即時に拒否されることがあります。 一時ページング・スペースの作成中にエラーが発生すると、手順が終了して、システムが元の設定に復帰します。 その他の問題が発生し、システム管理者の介入が必要になったり、場合によっては、即時にリブートしたりする必要が生じることがあります。 エラーによっては、一時ページング・スペースを削除できなくなることがあります。 この場合、通常、管理者による注意が必要になります。ただし、緊急ではありません。

重要 : shrinkps スクリプト内の **swapoff** コマンドによって、システムのバッキング・ページまたはユーザーのバッキング・ページで入出力エラーが検出された場合は、システムの破損の可能性を回避するために即時シャットダウンをお勧めします。 リブート時に一時ページング・スペースがアクティブになり、入出力エラーが発生したアプリケーションの停止と再始動の試行ができるようになります。 アプリケーションの停止および再始動に成功し、**swapoff** コマンドによる非活動化が完了したら、**mkps**、**swapoff**、および **rmps** コマンドを使用して縮小の手順を手動で完了し、必要なサイズのページング・スペースを作成し、一時ページング・スペースを削除します。

システムを再始動する前に入出力エラー状態にあった非活動化されたページング・スペースを削除 (**rmps** を使用) したり、再活動化 (**chps** を使用) しないでください。 ディスク・スペースが再使用されて、新たな問題が発生するリスクがあります。

同一ボリューム・グループ内での *hd6* ページング・スペースの移動

デフォルトのページング・スペースを hdisk0 から同一ボリューム・グループ内の別 のディスクに移動するには、システムのシャットダウンおよびリブートを 実行する 必要はありません。

root 権限で、次のコマンドを入力して、デフォルトの (hd6) ページング・スペース を hdisk0 から hdisk2 に移動します。

```
migratepv -l hd6 hdisk0 hdisk2
```

重要 : hd6 という名前を持つページング・スペースを、rootvg から別のボリューム・グループに移動させることはお勧めできません。なぜならば、ブート・プロセスの 2 番目のフェーズと、取り外し可能メディアからのブート時にルート・ボリューム・グループにアクセスするプロセスを含め、この名前は複数の場所でハードコーディングされているからです。 rootvg のページング・スペースのみが、ブート・プロセスの 2 番目のフェーズのときにアクティブになるので 、rootvg 内にページング・スペースがないと、システム・ブート・パフォーマンスに重大な影響を与えます。 ページング・スペースの大部分を他のボリューム・グループ上 に必要とする場合には、hd6 をできる限り小さく (物理メモリーと同サイズ) し、その後で、大きなページング・スペースを他のボリューム・グループ上に作成するほうが賢明です。

ページング・スペースのトラブルシューティング

ページング・スペースについての最も一般的な問題は、割り当てスペースの不足によって 生じるものです。

ページング・スペースの合計量は、しばしば試行錯誤により決定されます。一般的に使用される指針は、RAM サイズの 2 倍の数値をページング・スペースのターゲットとして使用することです。 ページング・スペースが少なくなると、プロセスが失われる可能性があります。また、ページング・スペースが使い果たされると、システムが誤動作する可能性があります。 次のシグナルおよびエラー情報は、ページング・スペース問題をモニターして解決するとき、あるいは予防するときに役立ちます。

オペレーティング・システムは、空きページング・スペース・ブロック数をモニターし、ページング・スペースの不足を検出します。 空きページング・スペース・ブロックの数 が、ページング・スペース警告レベル と呼ばれるしきい値を下回ると、システムは、**SIGDANGER** シグナルを送信することによって、すべてのプロセス (**kprocs**) を除く) にこの状態を通知します。 不足が続き、ページング・スペース *kill* レベルと呼ばれる 2 番目 のしきい値を下回ると、システムは、ページング・スペースの主要ユーザーであり、**SIGDANGER** シグナルのシグナル・ハンドラーをもたないプロセスに 対して、**SIGKILL** シグナルを送信します。 (**SIGDANGER** シグナルのデフォルト・アクションはシグナルを無視する ことです。) 空きページング・スペース・ブロック数がページング・スペース kill レベル を上回るようになるまで、システムは **SIGKILL** シグナルを送信し続けます。

注 : low_ps_handling パラメーターが 2 に 設定され (**vmo** コマンドの下で)、 どのプロセスも (**SIGDANGER** ハンドラーなしに) 強制終了しないことが分かった場合、システムは、**SIGDANGER** シグナル用のシグナル・ハンドラーをもつ、最も若いプロセスに **SIGKILL** シグナルを 送信します。

メモリーを動的に割り当てるプロセスは、**psdanger** サブルーチンで ページング・スペース・レベルをモニターするか、特殊な割り当てルーチンを使用することによって、十分なページング・スペースを確保することができます。 ページング・スペース kill レベル に達したときに、プロセスが終了しないようにするために、**disclaim** サブルーチンを使用できます。 これを行うには、**SIGDANGER** シグナルのシグナル・ハンドラーを定義し、プロセスのデータ領域とスタック領域および共有メモリー・セグメントで割り当てられている メモリーおよびページング・スペース・リソースを解放します。

次のようなエラー・メッセージが表示された場合は、ページング・スペースを増加する必要があります。

```
INIT: Paging space is low!
```

または

```
You are close to running out of paging space.
You may want to save your documents because
this program (and possibly the operating system)
could terminate without future warning when the
paging space fills up.
```

仮想メモリー・マネージャー

仮想メモリー・マネージャー (VMM) は、システムとそのアプリケーションによって 行われるメモリー要求を管理します。

仮想メモリーのセグメントは、ページ と呼ばれる単位に分割されています。 各ページは、実物理メモリー (RAM) に置かれているか、または必要になるまでディスクに保管されているかのいずれかです。 AIX では、仮想メモリーを使用して、システムで物理的に使用可能なメモリー以上のメモリーにアドレッシングします。 RAM 内またはディスク 上のメモリー・ページの管理は VMM によって行われます。

仮想メモリー・マネージャーにおける実メモリー管理

AIX では、仮想メモリーのセグメントはページと呼ばれる 4096 バイトの単位に分割されています。 実メモリーは 4096 バイトのページ・フレームに分割されています。

VMM は、次の 2 つの主要機能を備えています。

- ページ・フレームの割り当ての管理
- 現在 RAM 内にない (ページング・スペースに保管されている) か、またはまだ存在していない仮想メモリー・ページへの参照の解決

これらの機能を果たすために、VMM は使用可能なページ・フレームのフリー・リスト を維持しています。 また VMM は、現在 RAM 内にある仮想メモリー・ページのうち、どのページのページ・フレームをフリー・リストに再割り当てするのかを決定するためのページ置換アルゴリズムも使用します。 このページ置換アルゴリズムでは、永続セグメントと作業セグメントの存在、 再ページング、および VMM しきい値が考慮されます。

仮想メモリー・マネージャーの空きリスト
VMM は、ページ不在が発生した場合に VMM が使用する空き (未割り振り) ページ・フレームのリストを維持しています。

AIX は、フリー・リストに維持している少量を除いて、常に RAM のすべてを使用しようとします。 この少量の未割り振りページを維持するために、VMM はページ・アウト とページ・スチール を使用して、スペースを解放し、それらのページ・フレームをフリー・リストに再割り当てします。 どの仮想メモリー・ページのページ・フレームを再割り当てするかについての選択は、VMM のページ置換アルゴリズムを使用して行われます。

仮想メモリー・マネージャーにおける永続メモリー・セグメントと作業メモリー・セグメント
AIX では、タイプの異なるメモリー・セグメントはそれぞれ区別されます。 VMM を理解するには、作業セグメントと永続セグメントの違いを理解しておくことが重要です。

永続セグメント は、ディスク上に永続的な保管場所を持ちます。 データまたは実行可能プログラムが入っているファイルは永続セグメントにマップされます。 JFS ファイルまたは JFS2 ファイルがオープンされてアクセスされると、 そのファイルのデータは RAM にコピーされます。 VMM パラメーターにより、永続ページに割り当てられている物理メモリー・フレームを、 他のデータを保管するためにいつ上書きして使用するのかが制御されます。

作業セグメント は一時的なものであり、プロセスによって使用されている間だけ存在します。 作業セグメントは、ディスク上に永続的な保管場所を持ちません。 プロセスのスタック領域とデータ領域は、 作業セグメントと共用ライブラリー・テキスト・セグメントにマップされます。 作業セグメントのページも、実メモリーに保持できなくなった場合は、 ディスク上に保管場所を占める必要があります。 ディスク・ページング・スペースはこの目的のために使用されます。 プログラムの終了時には、そのプログラムのすべての作業ページが即時にフリー・リストに戻されます。

仮想メモリー・マネージャーにおける作動セグメントとページング・スペース
RAM 内の変更およびページアウト可能な作業ページには、 ページング・スペース内に対応するスロットが割り当てられます。

割り当てられたページング・スペースは、そのページのページアウトが必要となった場合にのみ使用されます。 ただし、ページング・スペースに割り当てられたページを別のページが使用することはできません。 そのスペースは、ある特定のページが仮想メモリー内に存在する限り、 そのページ用に予約されたままとなります。 永続ページはディスク上の元と同じ位置にページアウトされるので、RAM にある永続ページに対してはページング・スペースを割り振る必要がありません。

VMM には、早期 と遅延 の 2 種類のページング・スペース割り当てモードがあります。 早期割り振りポリシーは、作業ページに対するメモリー要求が出されるたびに、 ページング・スペースを予約します。 遅延割り振りポリシーは、作業ページが実際にメモリーからページアウトされる場合だけページング・スペースを割り当てるので、 システムのページング・スペース所要量が大幅に削減されます。

仮想メモリー・マネージャーのメモリー・ロード制御機能
ディスク上にある仮想メモリー・ページをプロセスが参照した場合に、 そのページが既にページアウトされているか、 まだ読み込まれていないと、その参照されたページをページインする必要があり、 使用可能な (空き) ページ・フレームの数が少ないと、 このページインのために 1 つ以上のページのページアウトが必要になることがあります。 VMM はページ置換アルゴリズムを使用して、 最近参照されていない、したがって近い将来参照される可能性の低いページ・フレームのスチールを試みます。

ページ置換が順調に行われると、 現在アクティブになっているすべてのプロセスのメモリー・ページは RAM 内に維持され、 アクティブでないプロセスのメモリー・ページがページアウトされます。 しかし、RAM がコミット過多の状態になると、ページアウトしたページが近い将来、 現在実行中のプロセスによって参照される可能性があるため、ページアウトするページの選択が困難になります。 その結果、すぐに参照される可能性のあるページがページアウトされ、 実際に参照されると再びページインされるといったことが起きる可能性があります。 RAM がコミット過多の状態のときには、スラッシング と呼ばれる、 連続したページインとページアウトが発生する可能性があります。 システムでスラッシングが起きている場合、 そのシステムはほとんどの時間を有用な命令の実行にではなく、 ページインとページアウトに費やすため、 どのアクティブ・プロセスにも目立った進行は行われません。 VMM は、システムでスラッシングが起きていることを検出してその状態の訂正を試みる、 メモリー関連の負荷を制御するためのアルゴリズムを備えています。

ファイルシステム

ファイルシステム とは、ファイルとディレクトリーの階層構造 (ファイル・ツリー) のことです。

この構造は、根が上に、枝が下にある逆向きの木に似ています。 このファイル・ツリーでは、ディレクトリーを使って、データとプログラムをグループに編成するので、 複数のディレクトリーとファイルを同時に管理することができます。

ファイルシステムは 1 つの論理ボリューム上に存在します。 すべてのファイルおよびディレクトリーは、論理ボリューム内のファイルシステムに属します。 その構造上、 タスクによっては、ファイルシステム 内の各ディレクトリーに対して実行するよりも、 ファイルシステム 全体に対して実行する方が効果的なものがあります。 例えば、ファイルシステム 全体のバックアップ、移動、または保護を行えます。 JFS ファイルシステムの時刻指定イメージまたは JFS2 ファイルシステムの時刻指定イメージ (スナップショット と呼ばれる) を作成できます。

注 : 論理ボリューム当たりの論理区画の最大数は、32,512 です。 ファイルシステムの 論理ボリュームの特性についての詳細は、 『**chlv** コマンド』を参照してください。

mkfs (make file system) コマンド、 またはシステム 管理インターフェース ・ツール (**smit** コマンド) は、ファイルシステムを 論理ボリューム上に作成します。

ファイルシステムをアクセス 可能にするには、ディレクトリー・マウント・ポイントにマウントしなければなりません。 複数のファイルシステムがマウントされると 、ディレクトリーの 構造が単一のファイルシステムのイメージを表すように作成されます。 これは、1 つのルートを持つ階層構造です。 この構造には、基本ファイルシステムとユーザーが 作成した他のファイルシステムが 含まれます。 **mount** コマンドを使用して、ローカルとリモートの両方のファイルシステムにアクセスできます。 これにより、ご使用のシステムからこのファイルシステムの 読み取りおよび書き込みが可能になります。 ファイルシステムのマウントおよびアンマウントを行うには、 通常、システム・グループのメンバーである 必要があります。 ファイルシステムは、`/etc/filesystems` ファイルに定義されている場合には、自動的にマウントできます。 ユーザーまたはプロセスがファイルシステムにアクセスしていない 限り、**umount** コマンドで、ローカルまたはリモート・ファイルシステムをアンマウントできます。 ファイルシステムのマウントの 詳細については、100 ページの『マウント』を参照してください。

AIX によって使用されるファイルシステムの 基本のタイプは、ジャーナル・ファイルシステム *(JFS)* と呼ばれます。 このファイルシステムでは、構造上の整合性を保つために、データベースのジャーナリング技法を利用します。 これにより、システムが異常停止したときに、ファイルシステムの 損傷を回避することができます。

AIX オペレーティング・システムは、ジャーナル・ファイルシステム (JFS) および拡張ジャーナル・ファイルシステム (JFS2) を含む複数のファイルシステム・タイプをサポートします。 ファイルシステム・タイプおよび各タイプの特性の詳細については、105 ページの『ファイルシステムのタイプ』を参照してください。

きわめて重要なシステム 管理作業の中には、ファイルシステムに関連したものがいくつかあります。代表的なものを次に示します。

- 論理ボリューム上でのファイルシステムに 対するスペースの割り振り
- ファイルシステムの 作成
- ファイルシステムのスペースがシステム ・ユーザーに使用できるようにする設定
- ファイルシステムのスペース 使用のモニター
- システム障害時のデータ損失の防止のためのファイルシステムのバックアップの 作成
- ファイルシステムを 整合性のある状態に保つための保守

以上の作業は、システム 管理者が実行するものです。

ファイルシステムの概念

お客様のファイルシステムを管理し、構成する前に、ファイル・ツリーの基本的な編成と内容を理解する必要があります。

ファイル・ツリーの編成および内容

ファイル・ツリーでは、同種の内容を持つファイルどうしがディレクトリーに編成されます。 このような編成のおかげで、ディレクトリーとファイルを簡単にリモート・マウントすることができます。

システム管理者は、これらのディレクトリーを構築ブロックとして使用して、1 つ以上 のサーバーから個々のディレクトリーをマウントして、 クライアント用に固有のファイル・ツリーを構築することができます。 ファイルやディレクトリーをリモートでマウントすれば、すべての情報をローカルで保持しておくよりも、 下記のような利点が得られます。

- ディスク・スペースが節約されます。
- システムを簡単に集中管理できるようになります。
- より安全な環境が提供されます。

ファイル・ツリーには、次のような特性があります。

- 同じハードウェア・アーキテクチャーのマシンが共用できるファイルは、/usr ファイルシステム 内にあります。
- スプール・ファイルやメール・ファイルなどのクライアントごとの可変ファイルは、/var ファイルシステムに置かれます。
- マニュアル・ページなどのアーキテクチャーに依存しない共用可能なテキスト・ファイルは、/usr/share ディレクトリーに置かれます。
- / (ルート) ファイルシステムには、システム操作のために重要なファイルおよびディレクトリーが入っています。 例えば、デバイス・ディレクトリー、システムの起動に使用されるプログラム、 ファイルシステムがルート・ファイルシステムにマウントされるときのマウント・ポイントなどが収められています。
- /home ファイルシステムは、 ユーザーのホーム・ディレクトリーのマウント・ポイントです。

ファイルシステムの構造

ファイルシステムとディレクトリーの 違いを理解しておくことは重要です。 ファイルシステムはファイルを収容するように割り振られた、ハード・ディスクのセクションです。 ハード・ディスクのこのセクションには、ディレクトリーの上にファイルシステムをマウントしてアクセスします。 ファイルシステムがマウントされると、これはエンド・ユーザーからは他のディレクトリーと同じように見えます。

しかし、ファイルシステムとディレクトリーには 構造上の違いがあるため、これらのエンティティー内のデータは別々に管理することができます。

オペレーティング・システムが初めてインストールされるときは、これは、以下の図に示されているように、ディレクトリー構造内にロードされます。

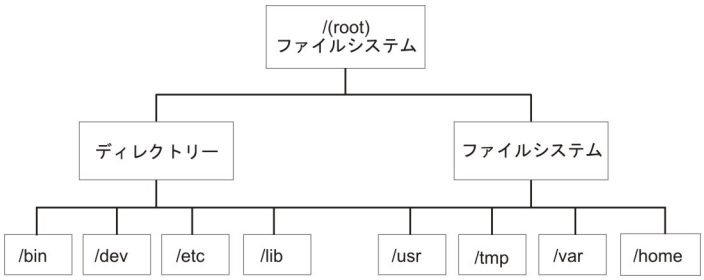


図 7. / (ルート) ファイルシステム・ツリー

右側のディレクトリー (/usr、/tmp、/var、および /home) はいずれもファイルシステムなので、ハード・ディスクの別々のセクションがそれぞれで使用するように割り振られています。 これらのファイルシステムはシステムの 始動時に自動的にマウントされるので、これらのファイルシステムと 左側にリストされているディレクトリー (/bin、/dev、/etc、および /lib) との違いはエンド・ユーザーにはわかりません。

スタンドアロン・マシンでは、デフォルトでは、次の各ファイルシステムが関連するデバイスに入っています。

/Device	/File System
/dev/hd1	/home
/dev/hd2	/usr
/dev/hd3	/tmp
/dev/hd4	/ (root)
/dev/hd9var	/var
/proc	/proc
/dev/hd10opt	/opt

ファイル・ツリーには、次のような特性があります。

- 同じハードウェア・アーキテクチャーのマシンが共用できるファイルは、/usr ファイルシステム 内にあります。
- 可変クライアント別ファイル (例えば、スプール・ファイルやメール・ファイル) は、/var ファイルシステムにあります。
- / (root) ファイルシステムには、 システム操作に必要なファイルとディレクトリーが入っています。例えば、以下に示すものが入っています。
  - デバイス・ディレクトリー (/dev)
  - ファイルシステムをルート・ファイルシステムにマウントする 場合のマウント・ポイント (例: /mnt)
- /home ファイルシステムは、ユーザーのホーム・ディレクトリーのマウント・ポイントです。
- サーバーの場合、/export ディレクトリーには、ページング・スペース・ファイル、クライアントごとの (共用されない) ルート・ファイルシステム 、ダンプ・ディレクトリー、ホーム・ディレクトリー、およびディスクレス・クライアント用の /usr/share ディレクトリーに加えて、エクスポートされた /usr ディレクトリーが入っています。
- /proc ファイルシステムには、システムの中のプロセスとスレッドの状態に関する情報が含まれています。
- /opt ファイルシステムには、アプリケーションなどのオプショナル ・ソフトウェアが含まれています。

以下に、/ (ルート) ファイルシステムのサブディレクトリーの 一部の内容について説明します。

項目	説明
/bin	/usr/bin ディレクトリーとのシンボリック・リンクです。
/dev	ローカル・デバイス用のスペシャル・ファイルの、デバイス・ノードが入っています。 /dev ディレクトリーには、テープ・ドライブ、プリンター、ディスク区画、および端末装置用のスペシャル・ファイルが収められます。
/etc	マシンごとに異なる構成ファイルが入っています。 例えば、次のようなものがあります。 • /etc/hosts • /etc/passwd
/export	リモート・クライアント用のサーバー上のディレクトリーおよびファイルが入っています。

項目	説明
/home	ユーザーのホーム・ディレクトリーが入っているファイルシステムの、マウント・ポイントとして使用されます。 /home ファイルシステムには、ユーザーごとのファイルおよびディレクトリーが入っています。 スタンドアロン・マシンでは、個別のローカル・ファイルシステムが、/home ディレクトリーの上にマウントされています。 ネットワークでは、複数のマシンからアクセスできるユーザー・ファイルが、サーバーに入っていることがあります。 この場合には、サーバーの /home ディレクトリーのコピーが、ローカルの /home ファイルシステムにリモートでマウントされます。
/lib	/usr/lib ディレクトリーに対するシンボリック・リンクです。これには lib*.a のフォーマットの名前の付いた、アーキテクチャーに依存しないライブラリーが入っています。
/sbin	マシンのブートおよび /usr ファイルシステムのマウントに必要なファイルが入っています。 ブート時に使用されるコマンドの大部分は、ブート・イメージの RAM ディスク・ファイルシステムに収められているため、ほとんどのコマンドは /sbin ディレクトリー内に常駐していません。
/tmp	システムが生成した一時ファイルが入っているファイルシステムの、 マウント・ポイントとして機能します。
/u	/home ディレクトリーへのシンボリック・リンクです。
/usr	変更されずに複数のマシンによって共用されるファイル (実行可能プログラムや ASCII 文書など) が入っているファイルシステムのマウント ・ポイントとして機能します。 スタンドアロン・マシンでは、/usr ディレクトリーの上に、個別のローカル・ファイルシステムをマウントします。 ディスクレスのマシンとディスク 容量の少ないマシンでは、/usr ファイルシステムの 上に、リモート・サーバーからディレクトリーをマウントします。
/var	マシンによって異なるファイルの、マウント・ポイントとして使用されます。 /var ファイルシステムは、入っているファイルが増大する傾向にあるので、ファイルシステムとして 構成されます。 例えば、それは /usr/tmp ディレクトリーに対するシンボリック・リンクで、そこには一時的な作業ファイルが含まれています。

ルート・ファイルシステム

ルート・ファイルシステムは、階層ファイル・ツリーの一番上に置かれます。 これには、システムのブートに使用されるデバイス・ディレクトリーとプログラムなども 含め、システム 操作に重要なファイルとディレクトリーが入れられます。 ルート・ファイルシステムには、マウント・ポイントも収められます。このマウント・ポイントにファイルシステムをマウントして、ルート・ファイルシステム 階層に接続することができます。

以下の図に、ルート・ファイルシステムのサブディレクトリーの 多くが示されています。

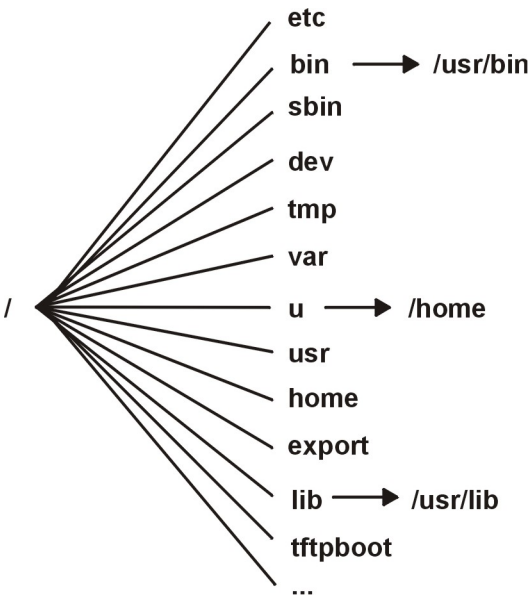


図 *8.* ルート・ファイルシステム

/ (ルート) ファイルシステムの一部のサブディレクトリーの内容を次にリストします。

項目	説明
/etc	マシンごとに異なる構成ファイルが入っています。 例えば、次のようなものがあります。 • /etc/hosts • /etc/passwd /etc ディレクトリーには、システム管理で通常使用するファイルが収められます。 前に /etc ディレクトリーに常駐していたコマンドの大部分は、現在では /usr/sbin ディレクトリーの中に常駐します。 ただし、互換性を保つために、/usr/sbin ディレクトリーには、一部の実行可能ファイルの位置とのシンボリック・リンクが収められています。 例えば、次のようなものがあります。 • /etc/chown は、/usr/bin/chown とのシンボリック・リンクです。 • /etc/exportvg は、/usr/sbin/exportvg とのシンボリック・リンクです。
/bin	/usr/bin ディレクトリーとのシンボリック・リンクです。 前の UNIX ファイルシステムでは、現在は /usr/bin ディレクトリーに常駐しているユーザー・コマンドが、/bin ディレクトリー内に入っていました。
/sbin	マシンのブートおよび /usr ファイルシステムのマウントに必要なファイルが入っています。 ブート時に使用されるコマンドの大部分は、ブート・イメージの RAM ディスク・ファイルシステムに収められているため、ほとんどのコマンドは /sbin ディレクトリー内に常駐していません。
/dev	ローカル・デバイス用のスペシャル・ファイルの、デバイス・ノードが入っています。 /dev ディレクトリーには、テープ・ドライブ、プリンター、ディスク区画、および端末装置用のスペシャル・ファイルが収められます。
/tmp	システム生成の一時ファイルが入れられるファイルシステムのマウント・ポイントとしての役割を果たします。 /tmp ファイルシステムは、空のディレクトリーです。
/var	マシンによって異なるファイルの、マウント・ポイントとして使用されます。 /var ファイルシステムは、 その中に収められているファイルに大きくなる傾向があることから、ファイルシステムとして構成されます。 詳しくは、78 ページの『/var ファイルシステム』を参照してください。

項目	説明
/u	/home ディレクトリーへのシンボリック・リンクです。
/usr	変更されることがなく、マシンで共用することができるファイル、例えば、 実行可能ファイルおよび ASCII 文書などが収められます。 スタンドアロン・マシンは、/usr ディレクトリーに、別個のローカル・ファイルシステムのルートをマウントします。 ディスクレス・マシンやディスク・リソースが制限されているマシンは、ディレクトリーをリモート・サーバーから /usr ファイルシステムにマウントします。 /usr ディレクトリーにマウントされるファイル・ツリーについて詳しくは、76 ページの『/usr ファイルシステム』を参照してください。
/home	ユーザーのホーム・ディレクトリーが入っているファイルシステムの、マウント・ポイントとして使用されます。 /home ファイルシステムには、ユーザーごとのファイルおよびディレクトリーが入っています。 スタンドアロン・マシンの場合、/home ディレクトリーは、ルート・ファイルシステムの /home ディレクトリーにマウントされるルートをもつ別個のファイルシステムの中に収められます。 ネットワーク内では、サーバーに、数種類のマシンからアクセスできるユーザー・ファイルが入れられることがあります。 このような場合、/home ディレクトリーのそのサーバーのコピーが、ローカル /home ファイルシステムにリモートでマウントされます。
/export	リモート・クライアント用のサーバー上のディレクトリーおよびファイルが入っています。 /export ディレクトリー下に常駐しているファイル・ツリーについて詳しくは、79 ページの『/export ディレクトリー』を参照してください。
/lib	/usr/lib ディレクトリーへのシンボリック・リンクです。 詳しくは、76 ページの『/usr ファイルシステム』を参照してください。
/tftpboot	ディスクレス・クライアントに使用されるブート・イメージとブート情報が収められます。

/usr ファイルシステム

/usr ファイルシステムには、複数のマシン間で共用することができる実行可能ファイルが収められます。

以下の図に、/usr ディレクトリーの主要なサブディレクトリーを示します。

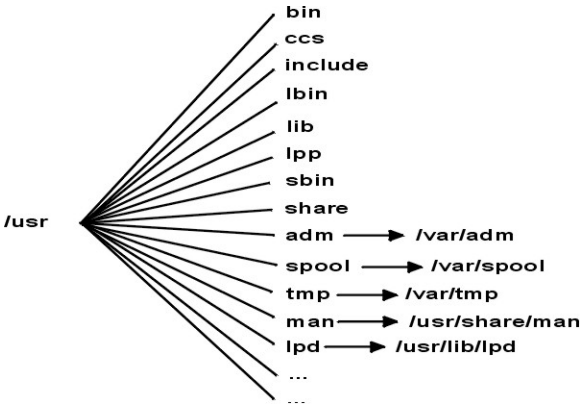


図 9. /usr ファイルシステム

スタンドアロン・マシンでは、/usr ファイルシステムは、(/dev/hd2 論理ボリューム内にある) 別個のファイルシステムです。 ディスクレス・マシンまたはディスク容量に制限のあるマシンでは、リモート・サーバーにあるディレクトリーが、ローカル /usr ファイルシステムに、読み取り専用許可でマウントされます。 /usr ファイルシステムには、読み取り専用のコマンド、ライブラリー、およびデータが収められます。

/usr/share ディレクトリーの内容を除き、/usr ファイルシステム内のファイルとディレクトリーは、同じハードウェア・アーキテクチャーのすべてのマシンで共用することができます。

/usr ファイルシステムには、次のディレクトリーが含まれます。

項目	説明
/usr/bin	通常のコマンドとシェル・スクリプトが収められます。 例えば、/usr/bin ディレクトリーには、**ls**、**cat**、および **mkdir** コマンドが入っています。
/usr/ccs	バンドルしない開発パッケージ・バイナリーが収められます。
/usr/include	組み込みファイルやヘッダー・ファイルが収められます。
/usr/lbin	コマンドのバックエンドである実行可能ファイルが収められます。
/usr/lib	これらのライブラリーには、lib*.a というフォームの名前が付けられます。/ (ルート) 内の /lib ディレクトリーは、/usr/lib ディレクトリーとのシンボリック・リンクであるため、以前は /lib ディレクトリー内に入れられていたファイルは、すべて、現在では /usr/lib ディレクトリーに収められています。 これには、互換性を保つ上で、いくつかの非ライブラリー・ファイルも含まれます。
/usr/lpp	オプションとしてインストールされたプロダクトが収められます。
/usr/sbin	システム管理に使用するユーティリティーが収められます。これには、システム管理インターフェース・ツール (SMIT) コマンドなどが含まれます。 前に /etc ディレクトリーに常駐していたコマンドの大部分は、現在では /usr/sbin ディレクトリーの中に常駐します。
/usr/share	異なるアーキテクチャーのマシン間で共用することができるファイルが収められます。 詳しくは、78 ページの『/usr/share ディレクトリー』を参照してください。

以下に、/var ディレクトリーへのシンボリック・リンクを示します。

項目	説明
/usr/adm	/var/adm ディレクトリーとのシンボリック・リンクです。
/usr/mail	/var/spool/mail ディレクトリーとのシンボリック・リンクです。
/usr/news	/var/news ディレクトリーとのシンボリック・リンクです。
/usr/preserve	/var/preserve ディレクトリーとのシンボリック・リンクです。
/usr/spool	/var/spool ディレクトリーとのシンボリック・リンクです。
/usr/tmp	/var/tmp ディレクトリーとのシンボリック・リンクです。/usr ディレクトリーは、数多くのノード間で共用される可能性があり、しかも読み取り専用であるためです。

以下に、/usr/share および /usr/lib ディレクトリーへのシンボリック・リンクを示します。

項目	説明
/usr/dict	/usr/share/dict ディレクトリーとのシンボリック・リンクです。
/usr/man	/usr/share/man ディレクトリーとのシンボリック・リンクです。
/usr/lpd	/usr/lib/lpd ディレクトリーとのシンボリック・リンクです。

/usr/share ディレクトリー

/usr/share ディレクトリーには、アーキテクチャーから独立した共用可能なテキスト・ファイルが収められます。 このディレクトリーの内容は、ハードウェア・アーキテクチャーに関係なく、すべてのマシンで共用することができます。

混合アーキテクチャー環境では、通常のディスクレス・クライアントは、独自の /usr ディレクトリーに 1 つのサーバー・ディレクトリーをマウントしてから、/usr/share ディレクトリーに別のディレクトリーをマウントします。 /usr/share ディレクトリーに属しているファイルは、別々にインストールできる 1 つ以上のパッケージに入れられます。 したがって、ノードは、サーバーを使用して /usr/share ディレクトリーを提供する一方で、依存している /usr ディレクトリーのその他の部分をローカルでインストールすることが可能です。

/usr/share ディレクトリー内の一部のファイルには、以下の図に示されているディレクトリーとファイルが含まれます。

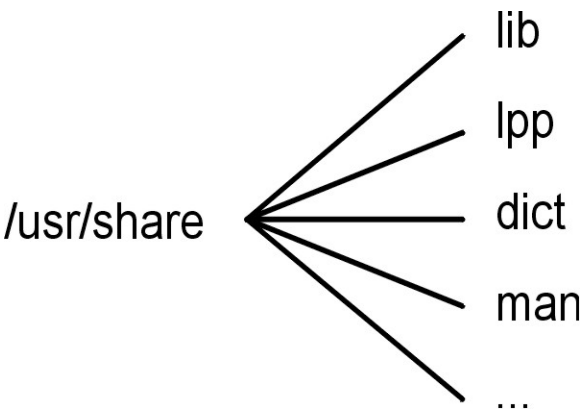


図 *10. /usr/share* ディレクトリー

この図は、/usr/share ディレクトリーの下にある /lib、/lpp、/dict、および /man などのディレクトリーを示しています。

/usr/share ディレクトリーには、次のものがあります。

項目	説明
/usr/share/man	マニュアル・ページが (ロードされている場合に) 収められます。
/usr/share/dict	スペル・ディクショナリーとその索引が収められます。
/usr/share/lib	terminfo、learn、tmac、 me、および macros を含む、アーキテクチャーから独立したデータ・ファイルが入ります。
/usr/share/lpp	オプションとしてシステム 上にインストールすることができるプロダクトに関するデータと情報が収められます。

/var ファイルシステム

/var ファイルシステムは、アカウンティング、メール、および印刷スプーラーなどの頻繁に使われるアプリケーションで使うサブディレクトリーとデータ・ファイルが収められるため、大きくなる傾向があります。

重要 : システム 上のアプリケーションが /var ファイルシステムを 非常によく使用する場合は、日常的に **skulker** コマンドを実行するか、またはこのファイルシステムのサイズを /var のデフォルトである 4MB よりも大きくしてください。

定期的なモニターが保証されている特定の /var ファイルは、/var/adm/wtmp と /var/adm/ras/errlog です。

モニターされる他の /var ファイルは、次のとおりです。

項目	説明
/var/adm/ras/trcfile	トレース機能がオンになっている場合。
/var/tmp/snmpd.log	**snmpd** コマンドがシステムで実行されている場合。

/var ディレクトリーの図に、/var ファイルシステムの一部のディレクトリーが示されています。

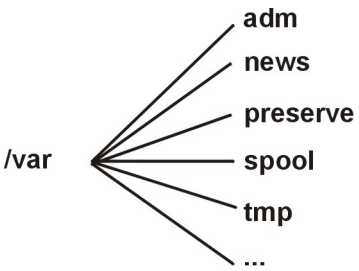


図 *11. /var* ディレクトリー

項目	説明
/var/adm	システム・ログ・ファイルとアカウンティング・ファイルが収められます。
/var/news	システム・ニュースが収められます。
/var/preserve	割り込まれた編集セッションから保持されているデータが収められます。 これは、前のリリースの /usr/preserve ディレクトリーとほぼ同じです。
/var/spool	電子メールなどのプログラムで処理中のファイルが収められます。これは、前のリリースの /usr/spool ディレクトリーとほぼ同じです。
/var/tmp	一時ファイルが収められます。これは、前のリリースの /usr/tmp ディレクトリーとほぼ同じです。 現在、/usr/tmp ディレクトリーは、/var/tmp とのシンボリック・リンクです。

/export ディレクトリー

/export ディレクトリーには、ディスクレス・コンピューター、データレス・コンピューター、またはディスク容量の少ないコンピューターなどのクライアントにエクスポートされるサーバー・ファイルが置かれています。

サーバーは、実行可能プログラムのパッケージ、ディスクレス・クライアント用のページング・スペース、ディスクレス・クライアントまたはディスク 容量の少ないクライアント用のルート・ファイルシステムなどのいくつかのタイプのディスク・スペースをエクスポートできます。 このようなディスク・スペースのファイル・ツリー内での標準的なロケーションが、/export ディレクトリーです。 /export ディレクトリーのサブディレクトリーの一部を、以下のリストに示します。

/exec
ディスクレス・クライアントが自身の /usr ファイルシステムにマウントするディレクトリーが 収められます。

/swap
ディスクレス・クライアントのリモート・ページング用のファイルが収められます。

/share
ディスクレス・クライアントが自身の /usr/share ファイルシステムにマウントするディレクトリーが収められます。

/root
ディスクレス・クライアントが自身の / （ルート）ファイルシステムにマウントするディレクトリーが収められます。

/home
ディスクレス・クライアントが自身の /home ファイルシステムにマウントするディレクトリーが 収められます。

/export ディレクトリーは、ディスクレス・コマンドのクライアント・リソースのデフォルトのロケーションです。 /export ディレクトリーは、サーバー上でクライアント・リソースが置かれている唯一のロケーションです。 クライアントはこれらのリソースを 自身のファイル・ツリーにマウントするため、各クライアントではこれらのリソースはファイル・ツリーの通常の場所にあるように見えます。 /export ディレクトリーの主なサブディレクトリーと、対応するクライアントのファイル・ツリー上のマウント・ポイントは次のようになります。

/export/root
このディレクトリーは、クライアントのルート (/) ファイルシステムにマウントされます。 クライアントのルート・ディレクトリーは、デフォルトでは /export/root ディレクトリー内にあり、クライアントのホスト名が付けられています。

/export/exec
SPOT (Shared Product Object Tree) ディレクトリーとも呼ばれます。 このディレクトリーは、クライアントの /usr ファイルシステムにマウントされます。 SPOT は /export/exec ディレクトリーに保管された /usr ファイルシステムの 1 バージョンであり、リリース・レベルを反映した名前が付けられています。 デフォルト名は RISCAIX です。

/export/share
このディレクトリーは、クライアントの /usr/share ディレクトリーにマウントされます。 このディレクトリーには、多くのアーキテクチャーで共用できるデータが収められています。 デフォルトのロケーションは /export/share/AIX/usr/share です。

/export/home
このディレクトリーは、クライアントの /home ファイルシステムにマウントされます。 このディレクトリーには、クライアントのホスト名ごとにまとめられているユーザー・ディレクトリーが収められています。 クライアントのホーム・ディレクトリーのデフォルトのロケーションは /export/home です。

/export/swap
ページング・ディレクトリーとも呼ばれます。 スタンドアロン・システムまたはデータレス・システムでは、ローカル・ディスクを使ってページングを行うことができます。ディスクレス・クライアントの場合、ページングはサーバーのファイルを 使用して実行されます。 このファイルはクライアントのホスト名が付けられたディレクトリー内にあり、デフォルトでは /export/swap ディレクトリーにあります。

/export/dump
スタンドアロン・システムはダンプ・デバイスとしてローカル・ディスクを使用します。ディスクレス・クライアントはサーバーのファイルをダンプ・デバイスとして使用します。 このファイルはクライアントのホスト名が付けられたディレクトリー内にあり、デフォルトでは /export/dump ディレクトリー内に置かれます。

microcode
このディレクトリーには物理デバイスのマイクロコードが備えられています。 デフォルトのロケーションは /export/exec/RISCAIX/usr/lib/microcode です。

JFS2 ファイルシステムの暗号化

AIX バージョン 6.1 を発端に、暗号化ファイルシステム (EFS) は JFS2 ファイルシステムでサポートされます。 EFS により、 ご使用のデータを暗号化し、鍵で保護されているデータへのアクセスを制御できます。

鍵は各ユーザーに関連付けられ、暗号法的に保護されている鍵ストアに保管されます。 ログインが正常に行われると、ユーザーの鍵はカーネルにロードされ、プロセス・クレデンシャルに関連付けられます。 EFS で保護されているファイルをオープンするには、プロセス・クレデンシャルはテストされます。 プロセスがファイル保護と一致する鍵を見つけた場合、プロセスはファイルの鍵およびファイル内容を暗号化解除します。

デフォルトでは、JFS2 ファイルシステムは EFS 対応ではありません。 JFS2 ファイルシステムは、すべてのデータが暗号化される前に EFS 対応になっている必要があります。 EFS 対応にすることについて詳しくは、「コマンド・リファレンス 第 *2* 巻」の **efsenable** コマンドを参照してください。

ファイルシステムの構成

ファイルシステムを追加または構成するときは、SMIT 高速パスのファイルシステム・コンテナーにあるオプションを選択できます。

SMIT 高速パスは、以下の表に記載されています。

表 3. 論理ボリュームおよびファイルシステムの管理のタスク	
タスク	***SMIT* 高速パス**
JFS または JFS2 の追加	**smit crfs**
既存論理ボリュームへの JFS2 の追加	**smit crjfs2lvstd**
定義済み論理ボリューム・メニューへの JFS の追加	論理ボリューム、次に **smit crjfslv** を作成する
JFS または JFS2 の属性の変更 注 1	**smit chfs**
ファイルシステムのサイズの 検査	**smit fs**
ファイルシステムのサイズの 増加	JFS: **smit chjfs** JFS2: **smit chjfs2**
ファイルシステムのサイズの 縮小	JFS2: **smit chjfs2**

注 : ファイルシステムのサイズの 縮小用の SMIT 高速パスは、JFS2 の場合のみです。

ファイルシステムの管理

ファイルシステムは完全なディレクトリー構造であり、 ルート・ディレクトリーおよびその下のサブディレクトリーとファイルを収めています。

ファイルシステムは 1 つの論理ボリューム内に収められます。 最も重要なシステム管理タスクの一部は、ファイルシステムを 対象としたものです。 代表的なものを次に示します。

- 論理ボリューム上でのファイルシステムに 対するスペースの割り振り
- ファイルシステムの 作成
- ファイルシステムのスペースがシステム ・ユーザーに使用できるようにする設定
- ファイルシステムのスペース 使用のモニター
- システム障害時にデータが損失しないようにファイルシステムのバックアップをとる 。
- ある時点におけるファイルシステムの 整合性のとれたブロック・レベルのイメージをキャプチャーするために、スナップショットを作成する。
- ファイルシステムの 整合状態を維持する。

以下に、ファイルシステムの 管理に役立つシステム 管理コマンドをリストします。

項目	説明
backup	ファイルシステムのフル ・バックアップ、または増分バックアップをとります。
chfs -a splitcopy	マウント済み JFS ファイルシステムのオンライン ・バックアップを作成します。
dd	あるデバイスから別のデバイスに直接データをコピーして、ファイルシステムのバックアップをとります。
df	ファイルシステムの 使用スペースと未使用スペースのサイズを報告します。
fsck	ファイルシステムを 検査し、不整合を修復します。
mkfs	指定された論理ボリュームに、指定されたサイズのファイルシステムを 作成します。
mount	ファイルシステム 内のファイルとディレクトリーにアクセスできるように、システム 全体の命名構造にファイルシステムを 結び付けます。
restore	ファイルをバックアップから復元します。

項目	説明
snapshot	JFS2 ファイルシステムのスナップショットを 作成します。
umount	システム 全体の命名構造からファイルシステムを除去し、ファイルシステム 内のファイルとディレクトリーにアクセスできないようにします。

ファイルシステムの使用可能なスペースを 表示する (df コマンド)

df コマンドを使用して、ファイルシステムの 合計スペースと使用可能なスペースについての情報を表示します。 **FileSystem** パラメーター には、ファイルシステムの 入っているデバイス名、ファイルシステムのマウント先のディレクトリー、ファイルシステムの 相対パス名のいずれかを指定します。

FileSystem パラメーターが指定されなかった場合には、**df** コマンドは、現在マウントされているすべてのファイルシステムの 情報を表示します。 ファイルまたはディレクトリーが指定された場合には、**df** コマンドは、そのファイルまたはディレクトリーがあるファイルシステムの 情報を表示します。

通常、**df** コマンドは、スーパーブロックに 入っているフリー・カウントを使用します。特定の例外条件のもとでは、これ らのカウントにエラーが生じる場合があります。 例えば、**df** コマンドの実行中に、ファイルシステムがア クティブに変更されると、空きカウントが不正確になる場合があります。

完全な構文については、「コマンド・リファレンス 第 *2* 巻」の **df** コマンドを参照してください。

注 : ネットワーク・ファイルシステム (NFS) などの一部のリモート・ファイルシステムでは、サーバーから情報が与えられない場合、ディスプレイ 上の使用可能なスペースを表すカラムがブランクのままになります。

df コマンドの使用例を以下に示します。

- マウント済みのファイルシステムすべてについての 情報を表示するには、次のように入力します。

  ```
  df
  ```

  /、/usr、/site および /usr/venus の 各ディレクトリーが、個別のファイルシステムに 常駐するようにシス テムが構成されている場合、**df** コマンドからの出力は次のようになります。

  ```
  Filesystem 512-blocks  free   %used   Iused %Iused   Mounted on
  /dev/hd4    20480     13780    32%      805   13%     /
  /dev/hd2   385024     15772    95%    27715   28%     /usr
  /dev/hd9var 40960     38988     4%      115    1%     /var
  /dev/hd3    20480     18972     7%       81    1%     /tmp
  /dev/hd1     4096      3724     9%       44    4%     /home
  ```

- 現行ディレクトリーが常駐するファイルシステムでの 使用可能なスペースを表示するには、次のように入力します。

  ```
  df .
  ```

ファイルシステム・コマンド

多くのコマンドが、タイプに関係なくファイルシステムを 操作するように設計されています。

/etc/filesystems ファイルは、 次のコマンドで操作できるファイルシステムのリストを 制御します。

項目	説明
chfs	ファイルシステムの 特性を変更します。
crfs	ファイルシステムを 追加します。
lsfs	ファイルシステムの 特性を表示します。
rmfs	ファイルシステムを 除去します。
mount	ファイルシステムを 使用できるようにします。

下記の 4 つのコマンドは、仮想ファイルシステムの 操作に使用します。 /etc/vfs ファイルには、下記のコマンドによって操作できるファイルシステム ・タイプの情報が備えられています。

項目	説明
chvfs	ファイルシステムのタイプの 特性を変更します。
crvfs	新しいファイルシステムのタイプを 追加します。
lsvfs	ファイルシステムのタイプの 特性をリストします。
rmvfs	ファイルシステムのタイプを 除去します。

異なるマシン上のファイルシステムの比較

異なるマシン上に存在する各ファイルシステムが 同一でなければならないときに、そのいずれかが損傷した と思われる場合は、各ファイルシステムを 比較することができます。

現行ホスト (このシナリオでは *orig_host* と言います) に常駐するファイルシステムの 属性を、リモート・ホスト上の同 じファイルシステムと 比較する方法の手順を以下に示します。

ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

1. リモート・ホストに root ユーザーとしてログインします。
   次に例を示します。

   ```
   tn juniper.mycompany.com


   AIX Version 6.1
   (C) Copyrights by IBM and by others 1982, 2002.
   login: root
   root's Password:
   ```

2. リモート・ホストの `.rhosts` ファイルを、使い慣れたエディターで編集し、root ユーザーがセキュア・リモート・コマンドを実行できるように スタンザを追加します。 次のフォーマットで、新規スタンザを作成します。

   ```
   orig_host root
   ```

   結果の `.rhosts` ファイルは、次のようになります。

   ```
   NIM.mycompany.com root
   nim.mycompany.com root
   host.othernetwork.com root
   orig_host.mycompany.com root
   ```

3. 変更を保存してリモート接続を終了します。
4. 使い慣れたエディターを使って、*orig_host* の root 権限で別のファイルを作成します。
   このシナリオでは、新しいファイルの名前を compareFS にします。 次に例を示します。

   ```
   vi compareFS
   ```

5. このファイルに以下のテキストを挿入します。ここで、*FSname* は比較するファイルシステムの 名前、*remote_host* は 比較ファイルシステムがあるホストの 名前です。

   ```
   FSname -> remote_host
   install -v ;
   ```

   注 : このファイルの **install** コマンド・ラインでは、 -v パラメーターとセミコロン (;) の間にスペースが必要です。

   例:

   ```
   /home/jane/* -> juniper.mycompany.com
   install -v ;
   ```

6. このファイルを保存してエディターを終了します。 次のステップでは、compareFS ファイルは、**rdist** コマンドの *distfile* として使用されます。

7. コマンド・プロンプトで次を入力します。

```
/usr/bin/rdist -f compareFS
```

または、比較の出力結果が大量になると思われるときは、出力をファイル名に送ります。 例:

```
/usr/bin/rdist -f compareFS > compareFS_output
```

ファイルシステム 間の違いが、出力としてリストされます。

ファイルシステムの保守

この表には、ファイルシステムの 保守で必要な最も簡単なタスクが グループごとに示してあります。

表 *4.* ファイルシステム保守タスク

タスク	***SMIT* 高速パス**	**コマンドまたはファイル**
名前によるファイルまたはディレクトリーのバックアップ	**smit backfile**	**backup** 注 1
JFS2 スナップショット・イメージの作成およびバックアップ	**smit backsnap**	**backsnap** 注 1
ディスク上のすべてのファイルシステムのリスト	**smit lsmntdsk**	
取り外し可能ディスク上のファイルシステムのリスト	**smit lsmntdsk**	
マウント済みファイルシステムのリスト	**smit fs**	
ファイルシステムのグループのマウント 注 5	**smit mountg**	**mount -t** *GroupName*
JFS または JFS2 のマウント 注 3	**smit mountfs**	**mount**
JFS2 スナップショットのマウント	**smit mntsnap**	**mount -v jfs2 -o snapshot** Device *MountPoint*
JFS または JFS2 の除去	**smit rmfs**	
JFS2 スナップショット の除去	**smit rmsnap**	**snapshot -d** *SnapshotDevice*
JFS2 ファイルシステムの 時刻指定スナップショットへの復帰	**smit rollbacksnap**	**rollback** [-s] [-v] [-c] *snappedFS snapshotObject*
ファイルシステムのアンマウント 注 4	**smit umountfs**	
取り外し可能ディスクのファイルシステムのアンマウント 注 4	**smit umntdsk**	
ファイルシステムのグループのアンマウント 注 5	**smit umountg**	**umount -t** *GroupName*
拡張されたジャーナル・ファイルシステム・クォータの管理	**smit j2fsquotas**	
クォータ管理を使用可能または使用不可にする	**smit j2enablequotas**	
クォータ制限適用の停止/再開	**smit j2enforcequotas**	**quotaon|off -v**
クォータ使用量のリスト	**smit j2repquota**	**repquota -v**

表 4. ファイルシステム保守タスク (続き)		
タスク	***SMIT* 高速パス**	**コマンドまたはファイル**
現行のディスク・ブロックおよびファイル使用率の統計の再計算	**smit j2quotacheck**	**quotacheck -v**
制限クラスの追加	**smit j2addlimit**	**j2edlimit -e**
制限クラスの特性の変更/表示	**smit j2changelimit**	
制限クラスをファイルシステムのデフォルトの制限にする	**smit j2defaultlimit**	
ユーザーまたはグループの制限クラスへの割り当て	**smit j2assignlimit**	
ファイルシステムの 制限クラスのリスト	**smit j2listlimits**	**j2edlimit -l '-u'**
制限クラスの除去	**smit j2removelimit**	

注 :

1. オプションについては、個々のコマンドを参照してください。
2. システムで重要な以下のファイルシステムの 名前は変更しないでください。論理ボリューム 4 (hd4) 上の / (ルート)、hd2 上の /usr、hd9var 上の /var、hd3 上の /tmp、および、hd5 上の /blv。 hd*n* 規則を使用する場合は、hd10 から開始してください。
3. マウントを実行する前に、88 ページの『ファイルシステムの 検査』手順を使用するか、**fsck** コマンドを実行して、ファイルシステムを 検査してください。
4. アンマウントが失敗した場合、ユーザーまたはプロセスがアンマウントしようとしたファイルシステム内のファイルがオープンしていることが考えられます。 **fuser** コマンドを使用すると、失敗の原因となったユーザーまたはプロセスを知ることができます。
5. ファイルシステムのグループとは 、**/etc/filesystems** ファイル内の type= ID に同じ値が設定されているファイルシステムの 集合です。

オンライン外部 JFS2 スナップショットからの 1 つ以上のファイルのリカバリー

オンライン外部 JFS2 スナップショットに完全なコピーがある場合、破壊されたファイルを 交換することができます。

次の手順に従って、オンライン外部 JFS2 スナップショット・イメージから 1 つ以上のファイルをリカバリーします。

この例のため、/home/aaa/myfile は、 /home ファイルシステムの 破損したファイルであると想定します。

1. 次のようなコマンドを使用してスナップショットのマウントを行います。

   ```
   mount -v jfs2 -o snapshot /dev/mysnaplv /tmp/mysnap
   ```

2. 次のようなコマンドを使用してスナップショットがあるディレクトリーに 変更します。

   ```
   cd /tmp/mysnap
   ```

3. 次のようなコマンドを使用して、破損したファイルを上書きするため、スナップショットから 正確なファイルをコピーします。

   ```
   cp aaa/myfile /home/aaa/myfile
   ```

前の例では、myfile という名前のファイルのみがコピーされます。 スナップショットから aaa ディレクトリー にすべてのファイルをコピーしたい場合、次のようなコマンドを使用します。

```
cp -R aaa /home/aaa
```

スナップショット・イメージを持つ、破損したファイルの置換の詳細な例については、「コマンド・リファレンス 第 *1* 巻」の **cp** または **cpio** コマンド説明を参照してください。

オンライン内部 JFS2 スナップショットからの 1 つ以上のファイルのリカバリー
オンライン内部 JFS2 スナップショットに完全なコピーがある場合、破壊されたファイルを 交換することができます。

次の手順に従って、オンライン内部 JFS2 スナップショット・イメージから 1 つ以上のファイルをリカバリーします。

この例のため、/home/aaa/myfile は、 /home ファイルシステムの破損したファイルであると想定します。

1. 次のようなコマンドを使用してスナップショットがあるディレクトリーに 変更します。

   ```
   cd /home/.snapshot/mysnap
   ```

2. 次のようなコマンドを使用して、破損したファイルを上書きするため、スナップショットから 正確なファイルをコピーします。

   ```
   cp aaa/myfile /home/aaa/myfile
   ```

   前の例では、myfile という名前のファイルのみがコピーされます。 スナップショットから aaa ディレクトリー にすべてのファイルをコピーしたい場合、次のようなコマンドを使用します。

   ```
   cp -R aaa /home/aaa
   ```

   スナップショット・イメージを持つ、破損したファイルの置換の詳細な例については、「コマンド・リファレンス 第 *1* 巻」の **cp** または **cpio** コマンド説明を参照してください。

CD-ROM および DVD ディスク上のファイルシステム
CD および DVD は自動的にはマウントされませんが、このフィーチャーは使用可能に できます。

このフィーチャーを使用可能にするには、**cdmount** コマンドを使用して、CDRFS または UDFS ファイルシステム をマウントします。例えば、次のように入力します。

```
cdmount cd0
```

次のコマンドを使用して、読み取り/書き込み UDFS を手動でマウントできます。

```
mount -V udfs DevName MtPt
```

DevName は DVD ドライブの名前、*MtPt* はファイルシステムのマウント ・ポイントです。

読み取り/書き込み光メディアでのファイルシステムの使用
CDRFS および JFS ファイルシステムは、読み取り/書き込み光メディアで使用できます。

CD-ROM ファイルシステム (CDRFS) は、CD-ROM に保管できるだけでなく、光メディアが書き込み保護されていれば、読み取り/書き込み光メディアにも保管できます。 次の表には、読み取り/書き込み光メディアで、CDRFS を追加、マウント、またはアンマウント する方法が示してあります。 ファイルシステムをマウントするときには、次の情報を指定する必要があります。

項目	説明
デバイス名	メディアを含むデバイスの名前を定義します。
マウント・ポイント	ファイルシステムのマウント 先となるディレクトリーを指定します。
自動マウント	システム 再起動時にファイルシステムを 自動的にマウントするかどうかを指定します。

光メディア上の **CDRFS**		
タスク	*SMIT* 高速パス	コマンドまたはファイル
CDRFS の追加 [1]	**smit crcdrfs**	1. ファイルシステムの追加: **crfs -v cdrfs -p ro -d***DeviceName* **-m** *MountPoint* **-A** *AutomaticMount* 2. ファイルシステムのマウント: **mount** *MountPoint*
CDRFS の除去 [2]	1. ファイルシステムのアンマウント: **smit umountfs** 2. **smit rmcdrfs**	1. ファイルシステムのアンマウント: **umount** *FileSystem* 2. ファイルシステムの取り外し: **rmfs** *MountPoint*

注:

- 読み取り/書き込み光メディアが書き込み保護されているかどうか確認してください。
- 読み取り/書き込み光メディアを取り外す場合は、まず、CDRFS ファイルシステムをアンマウントしておく必要があります。

JFS は、ハード・ディスクにあるものと同様の読み取り/書き込み ファイルシステムを 光メディアに提供します。 読み取り/書き込み光メディアに読み取り/書き込みファイルシステムを 作成またはインポートするには、システム 権限が必要であり (すなわち、ログインはシステム ・グループに属していなければなりません)、以下の情報を指定する必要があります。

ボリューム・グループ名
ボリューム・グループの名前を指定します。

デバイス名
読み取り/書き込み光ディスク・ドライブの論理名を指定します。

マウント・ポイント
ファイルシステムがマウントされるディレクトリーを 指定します。

自動マウント
システム 再起動時にファイルシステムを 自動的にマウントするかどうかを指定します。

注:

- 読み取り/書き込み光メディアに作成されたすべてのボリューム・グループは、このメディアだけに含まれていなければなりません。 ボリューム・グループは、1 つの読み取り/書き込み光ディスクを超えることはできません。
- 以前に作成したジャーナル・ファイルシステムにアクセスするときのボリューム ・グループ名は、ボリューム・グループを作成したときに使用した名前と一致させる必要はありません。

光メディア上の **JFS** に関するタスク		
タスク	*SMIT* 高速パス	コマンドまたはファイル

光メディア上の JFS に関するタスク		
JFS の追加	1. 光ディスクをドライブに挿入します。 2. ボリューム・グループの作成 (必要な場合): **smit mkvg** 3. ジャーナル・ファイルシステムの作成: **smit crfs**	1. 光ディスクをドライブに挿入します。 2. ボリューム・グループの作成 (必要な場合): **mkvg -f -y** *VGName* **-d 1** *DeviceName* 3. ジャーナル・ファイルシステムを作成します。**crfs -v jfs -g** *VGName* **-a size=***SizeFileSystem* **-m** *MountPoint* **-A** *AutomaticMount* **-p rw** 4. ファイルシステムのマウント: **mount** *MountPoint*
作成済み JFS へのアクセス 注 1	1. 光ディスクをドライブに挿入します。 2. ボリューム・グループのインポート: **smit importvg**	1. 光ディスクをドライブに挿入します。 2. ボリューム・グループのインポート: **importvg -y** *VGName DeviceName* 3. ファイルシステムのマウント: **mount** *MountPoint*
JFS の除去 注 2	1. ファイルシステムのアンマウント: **smit umountfs** 2. ファイルシステムの取り外し: **smit rmjfs**	1. ファイルシステムのアンマウント: **umount** *FileSystem* 2. ファイルシステムの取り外し: **rmfs** *MountPoint*

注 :

- ジャーナル・ファイルシステムを 含むメディアを挿入する場合は、この手順が必要になります。
- ジャーナル・ファイルシステムを取り外すと、そのファイルシステム 、および読み取り/書き込み光メディアに含まれるすべてのデータが破棄されます。

ファイルシステムの検査

ファイルシステムをマウントしたままシステムを 停止した場合、またはディスクが損傷を受けた場合には、ファイルシステムに 不整合が発生する可能性があります。 このような場合、ファイルシステムをマウントする前にファイルシステムを 検証することが大切です。

次の場合にも、ファイルシステムを 検証します。

- 動作不良の後; 例えば、ユーザーが、そのユーザーの許可 (uid) を 持つディレクトリーにディレクトリーを変更できない場合。
- エラーおよび復元の際に起こりうる問題を防止するため、ファイルシステムのバックアップをとる 前に。
- オペレーティング・システムのファイルのエラーがないことを確認するため、インストールまたはシステムのブート時に。

ユーザー定義のファイルシステムの検査

ユーザー定義のファイルシステムを検査するには、次のステップを実行します。

1. 検査するユーザー定義ファイルシステムをアンマウントします。
2. ファイルシステムのファイルに 対して書き込み許可をもっていることを確認します。もっていない場合は、修復プロンプトに「Yes(はい)」と答えても、損傷を受けたファイルを **fsck** で修復できません。

3. **smit fsck** 高速パスを使用して、「**Verify a File System (ファイルシステムの 検査)**」メニューにアクセスします。
4. 次のどちらかを行います。
   - 「**NAME of file system (ファイルシステム 名)**」フィールドに 検査したい個々のファイルシステムの 名前を指定するか
   - 「**TYPE of file system (ファイルシステムのタイプ)**」フィールド で、ジャーナル・ファイルシステム (JFS) などの、検査したい一般ファイルシステム・タイプを選択します。
5. 最も可能性の高い候補に検査を制限する場合は、「**FAST check? (確認を早く行う)**」フィールドで「Yes (はい)」を指定します。
   高速検査オプションでは、不整合のある可能性があるファイルだけが検査されます。例えば、過去のある時点で、システムが停止されたときにマウントされていた ファイルシステムなどです。
6. 「**SCRATCH file (スクラッチ・ファイル)**」フィールドに、検査対象でないファイルシステム 上の一時ファイルの名前を指定します。
7. ファイルシステム 検査を開始します。

***root* および */usr* ファイルシステムの検査**

/ または /usr ファイルシステムで **fsck** コマンドを実行する場合、/ (root) および /usr ファイルシステムは、実行中のシステムからはアンマウントできないので、システムをシャットダウンし、取り外し可能メディアからシステムをリブートする 必要があります。

以下の手順で、保守シェルから / および /usr ファイルシステムで **fsck** を実行する方法について説明します。

1. システムのシャットダウン
   (root アクセスが必要です)
2. インストール・メディアからブートします。
3. 「**Welcome (ウェルカム)**」メニューから、「**Maintenance (保守)**」オプションを選択します。
4. 「**Maintenance (保守)**」メニューからボリューム・グループにアクセスするオプションを 選びます。
5. rootvg ボリューム・グループを選択します。 選択したボリューム・グループに属する論理ボリュームのリストが表示されます。
6. **2** を選択し、ボリューム・グループにアクセスして、ファイルシステムのマウント 前にシェルを始動します。
   以下のステップでは、該当するオプションとファイルシステムのデバイス 名を使用して **fsck** コマンドを実行します。 **fsck** コマンドはファイルシステムの 整合性を検査し、対話形式でファイルシステムを修復します。 / (ルート) ファイルシステムのデバイスは 、/dev/hd4 であり、/usr ファイルシステムのデバイスは /dev/hd2 です。
7. / ファイルシステムを 検査するには、次のように入力します。

   ```
   $ fsck -y /dev/hd4
   ```

   -y フラグはあまり経験のないユーザーにお勧めします (**fsck** コマンドを参照)。
8. /usr ファイルシステムを 検査するには、次のように入力します。

   ```
   $ fsck -y /dev/hd2
   ```

9. rootvg のその他のファイルシステムを 検査するには、該当するデバイス名を指定して、**fsck** コマンドを入力します。 /tmp のデバイスは /dev/hd3 であり、/var のデバイスは /dev/hd9var です。
10. ファイルシステムの 検査を終了した後、システムをリブートします。

ルート・ボリューム・グループのファイルシステムのサイズの縮小

すべての ファイルシステムを最小サイズに縮小する一番簡単な方法は、バックアップから基本オペレーティング・システムを復元するときに、**SHRINK** オプションを「**yes**」に設定することです。

すべての ファイルシステムを最小サイズに縮小する一番簡単な方法は、バックアップから基本オペレーティング・システムを復元するときに、**SHRINK** オプションを「**yes**」に設定することです。 **SHRINK** オプ

ションと以下のシナリオを 相前後して使用することはできません。 以下の手順を実行した後に **SHRINK** オプションを **yes** に設定すると、 インストール・システムが /image.data ファイルをオーバーライドします。

このシナリオでは、選択した rootvg ファイルシステムのサイズを 手動で 縮小する方法を示します。 割り振 られたディスク・スペースの一部しか使用していないファイルシステムを 識別し、ファイルシステムが実際 に使用するスペース量に基づいて再割り振りを行います。こうすると、ルート・ボリューム・グループ用 に、より多くのスペースを解放します。 この手順の一部では、ボリューム・グループをバックアップし、修 正した割り振り量を適用して、オペレーティング・システムの再インストールを行います。

重要 : この手順では、基本オペレーティング・システムを シャットダウンして、再インストールする必要があります。 オペレーティング・システムを再インストールするときは、データや機能が失われないよう に、必ず、ワークロードへの影響が一番少ないダウン時間をスケジューリングします。 オペレーティング・システムを再インストールする前に、データおよびすべてのカスタマイズしたアプリケーションまたはボリ ューム・グループの信頼できるバックアップがあることを確認します。

ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

1. rootvg に含まれて いない すべてのファイルシステムのバックアップを 別々に作成します。 個別のバッ クアップにより、すべてのファイルシステムの 保全性を確実に保つことができます。
2. root 権限で、次のコマンドを入力することによって、ルート・ボリューム・グループのうちで、割り振りディスク・スペースを使用していないファイルシステムを 検査します。

   ```
   df -k
   ```

   -k フラグによって、ファイルシステムのサイズを KB 単位で表示します。 次のような結果が表示されます。

   ```
   Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
   /dev/hd4           196608      4976   98%     1944     2% /
   /dev/hd2          1769472    623988   65%    36984     9% /usr
   /dev/hd9var        163840     65116   61%      676     2% /var
   /dev/hd3            65536     63024    4%      115     1% /tmp
   /dev/hd1            49152      8536   83%      832     7% /home
   /proc                   -         -    -        -     -  /proc
   /dev/hd10opt        32768     26340   20%      293     4% /opt
   ```

   これらの結果を見ると、/usr にマウントされたファイルシステムで、多くの空きブロックがあり、使用 パーセントがかなり低いことが分かります。 /usr ファイルシステムに割り振られた区画数を減らすこと によって、多くのブロックを解放できると判断できます。
3. /etc/filesystems ファイルの内容を検査して、rootvg のすべてのファイルシステムがマウントされていることを確認します。 マウント されていないものがある場合、それらは再インストールされたシステムには組み込まれません。
4. 次のコマンドを入力して、/image.data ファイルを作成します。このファイルには、rootvg ボリューム・グループの、インストール手順に組み込まれるアクティブなファイルシステムのすべてがリストされます。

   ```
   mkszfile
   ```

5. 任意のエディターで /image.data ファイルをオープンします。
6. usr のテキスト・ストリングを探して、/usr ファイルシステムに関連した lv_data スタンザの場所を見付けます。

   このスタンザにある数を基本として、/usr ファイルシステムで減らすことのできる論理区画の数を判別します。 それぞれの追加論理区画の デフォルト・サイズは、/image.data ファイルの PP_SIZE エントリーに定義されています。 /image.data ファイルは次のように表示されます。

   ```
   lv_data:
       VOLUME_GROUP= rootvg
       LV_SOURCE_DISK_LIST= hdisk0
       LV_IDENTIFIER= 00042345d300bf15.5
   ```

```
    LOGICAL_VOLUME= hd2
    VG_STAT= active/complete
    TYPE= jfs
    MAX_LPS= 32512
    COPIES= 1
    LPs= 108
    STALE_PPs= 0
    INTER_POLICY= minimum
    INTRA_POLICY= center
    MOUNT_POINT= /usr
    MIRROR_WRITE_CONSISTENCY= on/ACTIVE
    LV_SEPARATE_PV= yes
    PERMISSION= read/write
    LV_STATE= opened/syncd
    WRITE_VERIFY= off
    PP_SIZE= 16
    SCHED_POLICY= parallel
    PP= 108
    BB_POLICY= relocatable
    RELOCATABLE= yes
    UPPER_BOUND= 32
    LABEL= /usr
    MAPFILE=
    LV_MIN_LPS= 70
    STRIPE_WIDTH=
    STRIP_SIZE=
```

この論理ボリュームに割り当てられた論理区画の数は 108 です (LPs=108)。

7. ステップ 2 の結果を参考にして、/usr ファイルシステムの既存データで必要となる論理区画の数を判別します。
   次のコマンドを使用すると、/usr ファイルシステムに固有の、既存のファイル・サイズを表示できます。

```
df -k /usr
```

/usr ファイルシステムに関してステップ 2 で受け取った結果の数 (KB 単位) は、再度表示されます。

```
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd2          1769472    623988   65%    36984     9% /usr
```

a) 割り振られた 1024-blocks の合計数からフリー・スペースの数量を引きます。

```
1769472 - 623988 = 1145484
```

b) このファイルシステムでの 将来の増加分を収容するのに必要なスペースの予測数量を追加します。この例では、計算結果に 200000 を追加します。

```
1145484 + 200000 = 1345484
```

c) 上の結果を (16*1024) バイト単位の論理区画サイズで割り、必要な論理区画の最小数を求めます。

```
1345484 / 16384 = 82.121826171875
```

この結果を切り上げて、必要な論理区画の数 (LPs=83) を再定義します。

8. image.data ファイルで、「LPs」フィールドを 108 から 83 に変更します。
9. /usr ファイルシステムに関連した fs_data スタンザを見つけます。
   fs_data スタンザは次のように表示されます。

```
fs_data:
    FS_NAME= /usr
    FS_SIZE= 3538944
    FS_MIN_SIZE= 2290968
    FS_LV= /dev/hd2
    FS_FS= 4096
    FS_NBPI= 4096
    FS_COMPRESS= no
    FS_BF= false
    FS_AGSIZE= 8
```

10. 物理区画サイズ (PP_SIZE) に 2 (物理区画が使用する 512 バイト・ブロックの数) および論理区画の 数 (LPs) を掛けて、ファイルシステム・サイズ (FS_SIZE) を計算します。
この例での値を使用して計算すると、次のようになります。

```
PP_SIZE * 512 blocks * LPs = FS_SIZE
16384 * 2 * 83 = 2719744
```

11. image.data ファイルで、「FS_SIZE」フィールドを 3538944 から 2719744 に変更します。
12. /usr ファイルシステムによって 使用される、現行データの実際のサイズに基づいて、最小のファイル システム・ サイズ (FS_MIN_SIZE) を以下のように計算します。
    a) 必要な最小区画数を計算します。
    この例での値を使用して計算すると、次のようになります。

    ```
    size_in_use (ステップ 7a を参照) / PP_SIZE = partitions
    1145484 / 16384 = 69.914794921875
    ```

    b) この区画数で必要となる最小サイズを計算します。
    上の計算結果を 70 に切り上げて計算すると、次のようになります。

    ```
    PP_SIZE * 512 blocks * partitions = FS_MIN_SIZE
    16384 * 2 * 70 = 2293760
    ```

13. image.data ファイルで、「FS_MIN_SIZE」フィールドを 2290968 から 2293760 に変更します。
14. 編集内容を保存し、エディターを終了します。
15. rootvg ボリューム・グループ内にない、すべてのファイルシステムをアンマウントします。
16. ユーザー定義のボリューム・グループがある場合は、以下のコマンドを入力して、それらをオフに変更 し、エクスポートします。

```
varyoffvg VGName
exportvg VGName
```

17. テープ・ドライブ内にテープがある場合は、以下のコマンドを入力して、完全システム・バックアップ を開始します。

```
mksysb /dev/rmt0
```

    このタイプのバックアップには、/image.data ファイルで指定したファイルシステム・サイズ情報が含 まれます。後から、新規ファイルシステム・サイズを指定してシステムを再インストールするときに、この情報を使用します。

    注 : このバックアップを開始するには、コマンド・ラインから **mksysb** コマンドを実行する必要があります。 SMIT などのシステム管理ツールを使用すると、バックアップで新規 image.data ファイルが作成され、変更した内容が上書きされます。
18. このバックアップを使用して、「**Install With Current System Settings (現行システム設定値を指定したインストール)**」オプションを指定して オペレーティング・システムを再インストールします。
    インストールの間、以下のオプションが適切に設定されていることを確認してください。
    - 「**Use Maps (マップの使用)**」が「**いいえ**」に設定されている
    - 「**Shrink the File Systems (ファイルシステムの縮小)**」が「**いいえ**」に設定されている

    インストール手順について詳しくは、システム・バックアップのインストールを参照してください。
19. オペレーティング・システムをインストールしたら、通常モードでシステムをリブートします。
    この時点で、/usr ファイルシステムのサイズが 変更されます。ただし、ユーザー定義のファイルシステムは 使用可能にはなっていません。
20. 次のコマンドを実行して、すべてのファイルシステムをマウントします。

```
mount all
```

    マウント済みのファイルシステムについて 「Device Busy」というメッセージが表示された場合は、それらのメッセージを無視してください。

この時点で、/usr ファイルシステムのサイズが 変更され、ルート・ボリューム・グループにより多くのフリ ー・スペースが割り当てられ、ユーザーのファイルシステムが使用可能になります。

関連概念

論理ボリューム・ストレージ
論理ボリュームは、物理ボリューム上に存在する情報のグループです。

関連情報

Creating a Root Volume Group Backup to Tape or File
/image.data ファイルの説明
mkszfile コマンド
mksysb コマンド

ファイルシステムのトラブルシューティング

お客様のファイルシステムに 発生するかもしれない基本的な問題に取り組むために、以下のトラブルシューティング方式を 使用してください。 トラブルシューティング 情報がお客様の問題を扱っていない場合、サービス担当員に連絡してください。

ユーザー定義ファイルシステムのオーバーフローの解決

オーバーフローしているユーザー定義ファイルシステムを 解決するには、以下の手順に従います。

1. 旧バックアップ・ファイルとコア・ファイルを削除します。 以下の例では、*.bak、.*.bak、a.out、core、*、または ed.hup ファイルをすべて除去します。

```
find / ¥( -name "*.bak" -o -name core -o -name a.out -o ¥
        -name "...*" -o -name ".*.bak" -o -name ed.hup ¥) ¥
        -atime +1 -mtime +1 -type f -print | xargs -e rm -f
```

2. ファイルが定期的にディスクをオーバーフローすること 防ぐために、**cron** プロセスの一部として **skulker** コマンドを実行して、不必要なファイル、または一時的なファイルを削除します。

   skulker コマンドは、 /tmp ディレクトリー内のファイル、指定された経過時間を超えたファイル、a.out ファイル、コア・ファイル、ならびに ed.hup ファイルをパージします。 それは、オフピーク時に、**cron** コマンドによって行われるアカウンティング手順 (アカウンティングがオンになっているとした場合) の一部として毎日実行します。

   cron デーモンは、指定された日時にシェル・コマンドを実行します。 **skulker** などの定期的にスケジュールされるコマンドは、crontab ファイルに含まれる指示に従って指定できます。 crontab ファイルを **crontab** コマンドによって実行依頼します。 crontab ファイルを編集するには、root ユーザー権限を持つ必要があります。

損傷したファイルシステムの解決

ファイルシステムのディレクトリー 構造の i ノードまたはスーパーブロック 情報が壊れると、ファイルシステムが壊れます。

この破損は、ハードウェア関連の不調、または i ノードやスーパーブロック 情報に直接アクセスするプログラムが壊れたことが原因である可能性があります。 (アセンブラーと C によって作成されたプログラムはオペレーティング・システムをバイパスし、ハードウェアに直接書き込むことができます。) 壊れたファイルシステムの症状の 1 つとして、特定のファイルシステム 内にあるデータをシステムが検索、読み取り、または書き込みできなくなることがあります。

損傷したファイルシステムを 解決するには、問題を診断し、それを修復する必要があります。 **fsck** コマンドは、低レベルの診断と修復を行います。

以下に、損傷したファイルシステムを 解決する手順を示します。

1. root 権限で、次のどちらかの SMIT 高速パスを使用して、損傷したファイルをアンマウント します。 **smit unmountfs** (ハード・ディスク・ドライブ上のファイルシステムの 場合) または **smit unmntdsk** (取り外し可能ディスク 上のファイルシステムの 場合)。

2. **fsck** コマンドを実行し、ファイルシステムの損傷部分を評価します。 次の例では、**fsck** コマンドは、/dev/myfilelv デバイスにあるアンマウントされたファイルシステムを 検査します。

```
fsck /dev/myfilelv
```

fsck コマンドは、整合がとれていないファイルシステムを 検査し、対話形式で修復します。 通常、ファイルシステムには整合性があり、**fsck** コマンドは、単にファイルシステム 上のファイル数、使用済みブロック数、ならびに未使用ブロック数を報告するのみです。 ファイルシステムに 不整合がある場合、**fsck** コマンドは、検出した不整合に関する情報を表示し、それらを修復するための許可を求めるプロンプトを出します。 **fsck** コマンドは、修復に際しては慎重で、有効なデータを失う結果になる可能性があるアクションを避けるようにします。 しかし、場合によっては、**fsck** コマンドが、損傷したファイルの破壊を推奨することがあります。 このコマンドの検査対象の不整合のリストについては、「コマンド・リファレンス 第 *2* 巻」の **fsck** コマンド説明を参照してください。

3. ファイルシステムを 修復できない場合は、バックアップから復元してください。

重要 : バックアップからファイルシステムを 復元すると、ディスク 上に以前に保管されていたすべてのファイルシステムが 破壊され、置き換えられます。

バックアップからファイルシステムを 復元するには、SMIT 高速パス **smit restfilesys** か、次の例に示す一連の コマンドを使用します。

```
mkfs /dev/myfilelv
mount /dev/myfilelv /myfilesys
cd /myfilesys
restore -r
```

この例では、**mkfs** コマンド により、/dev/myfilelv という名前のデバイス上に新規ファイルシステム が作成され、ボリューム・ラベル、ファイルシステム ・ラベル、開始ブロックが初期化されます。 **mount** コマンドにより、/dev/myfilelv が **myfilesys** の マウント・ポイントとして設定され、**restore** コマンドにより、バックアップからファイルシステムが 取り出されます。

ファイルシステムの 増分バックアップを使用してバックアップを 作成した場合は、バックアップ・レベルの小さいほうから順にバックアップ を復元する必要があります (例えば、0、1、2 の順)。 **smit restfilesys** を使用してファイルシステム 全体を復元するときには、ターゲット・ディレクトリー、復元デバイス (/dev/rfd0 以外)、および 1 回の入力操作で読み取るブロック数を入力します。

ファイルシステムのスーパーブロックの破損したマジック・ナンバーの修復

ファイルシステムのスーパーブロックが 損傷すると、ファイルシステムにアクセスできなくなります。 ファイルシステムのスーパーブロックの 破損した マジック・ナンバーは修復できます。

スーパーブロックの 損傷のほとんどは、修復することができません。 以下の手順では、マジック・ナンバー の破損により問題が引き起こされた場合、JFS ファイルシステムのスーパーブロックをどのように 修復するかについて説 明します。 JFS2 ファイルシステムの 1 次スーパーブロックが 壊れている場合は、**fsck** コマンドを使用し て、自動的に 2 次スーパーブロックをコピーし 、1 次スーパーブロックを 修復します。

以下のシナリオでは、/home/myfs は、 物理ボリューム /dev/lv02 上の JFS ファイルシステムであるとします。

ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

1. 次のコマンドを使用して、損傷している疑いのある /home/myfs ファイルシステムをアンマウントします。

```
umount /home/myfs
```

2. ファイルシステムの 損傷を確認するために、ファイルシステムに 対して **fsck** コマンドを実行します。次に例を示します。

```
fsck -p /dev/lv02
```

問題がスーパーブロックの 損傷の場合は、**fsck** コマンドは、次のどちらかのメッセージを戻します。

```
fsck: Not an AIXV5 file system
```

または

```
Not a recognized filesystem type
```

3. root 権限で、次の例のように、**od** コマンドを使用し、 ファイルシステムのスーパーブロックを 表示します。

```
od -x -N 64 /dev/lv02 +0x1000
```

この場合、-x フラグでは、出力は 16 進数形式で表示され、-N フラグは、 オフセット・パラメーター (+) から始まる 64 入力バイト分までをフォーマットするようシステムに 指示します。オフセット・パラメーターは、ファイルの 出力が開始されるファイル内のポイントを指定します。次に出力例を示します。

```
0001000  1234 0234 0000 0000 0000 4000 0000 000a
0001010  0001 8000 1000 0000 2f6c 7633 0000 6c76
0001020  3300 0000 000a 0003 0100 0000 2f28 0383
0001030  0000 0001 0000 0200 0000 2000 0000 0000
0001040
```

上記の出力で、0x1000 の壊れたマジック値 (1234 0234) に注意してください。ファイルシステムが 作成されたときに、すべてのデフォルトを用いた場合は、マジック・ナンバーは 0x43218765 になっているはずです。上書きされたデフォルトがある場合は、マジック・ナンバー は 0x65872143 になっているはずです。

4. **od** コマンドを使用して、正しいマジック・ナンバーの 2 次スーパーブロックを 検査します。 コマンド例とその出力結果を次に示します。

```
$ od -x -N 64 /dev/lv02 +0x1f000
001f000  6587 2143 0000 0000 0000 4000 0000 000a
001f010  0001 8000 1000 0000 2f6c 7633 0000 6c76
001f020  3300 0000 000a 0003 0100 0000 2f28 0383
001f030  0000 0001 0000 0200 0000 2000 0000 0000
001f040
```

0x1f000 にある正しいマジック値に注意してください。

5. 2 次スーパーブロックを 1 次スーパーブロックにコピーします 。 コマンド例および出力結果は、以下のようになります。

```
$ dd count=1 bs=4k skip=31 seek=1 if=/dev/lv02 of=/dev/lv02
dd: 1+0 records in.
dd: 1+0 records out.
```

6. **fsck** コマンドを使用して、2 次スーパーブロックの 使用によって生じた不整合ファイルをクリーンアップします。 次に例を示します。

```
fsck /dev/lv02 2>&1 | tee /tmp/fsck.errs
```

関連情報

fsck コマンド

od コマンド

ディスクのオーバーフロー

割り振られたスペースに入れられるファイルが多過ぎると、ディスクのオーバーフローが 起こります。 これは、多くの不必要なファイルを作成するランナウェイ・プロセスが原因で起こります。

問題を解決するために次の手順が使用できます。

注 : 自分自身以外のプロセスを除去するためには、root ユーザー権限を持つ必要があります。

問題プロセスの識別
この手順は、問題プロセスを特定する場合に使用します。

1. プロセス状態をチェックし、問題の原因と考えられるプロセスを識別するには、以下のように入力します。

   ```
   ps -ef | pg
   ```

   ps コマンドはプロセス状態を表示します。 **-e** フラグはすべてのプロセス (カーネル・プロセスを除く) についての情報を書き込み、**-f** フラグはプロセス作成時のコマンド名およびパラメーターを含め、プロセスの完全なリストを生成します。 **pg** コマンドは、出力が一度に 1 ページだけに表示されるようにします。このコマンドを使用すると、情報が、スクロールによって画面からすぐに消えないようになります。

   CPU 時間などのシステム・リソースを過度に使用しているシステム・プロセスまたはユーザー・プロセスを検査します。 **sendmail**、**routed**、および **lpd** などのシステム・プロセスが、ランナウェイに最もなりやすいシステム・プロセスです。
2. 予期したよりも多くの CPU を使用するユーザー・プロセスを検査するには、次のように入力します。

   ```
   ps -u
   ```
3. 各問題プロセスのプロセス ID を記録します。

プロセスの終了
問題プロセスは終了できます。

次の手順に従って、問題プロセスを終了します。

1. 次のように入力して、問題の原因となるプロセスを終了します。

   ```
   kill -9 PID
   ```

   ここで *PID* は問題プロセスの ID です。
2. プロセスが作成していたファイルを削除します。

   ```
   rm file1 file2 file3
   ```

   file1 file2 file3 は、プロセスに関連したファイルの 名前です。

プロセスを終了しないファイル・スペースの再利用
プロセスを終了せずにアクティブ・ファイルに割り振られていたブロックを再利用するには、別のコマンドの出力をそのファイルにリダイレクトします。 データ・リダイレクトはファイルを切り捨て、メモリーのブロックを再利用します。

アクティブ・ファイルをファイルシステムから 削除する場合、そのファイルに割り振られていたブロックは、ファイルをクローズするプロセスの結果、または、ファイルをオープンしたプロセスの終了のどちらかにより最後のオープン 参照が除去されるまで、割り振られたままです。 ランナウェイ・プロセスがファイルへの書き込みを行っていて、そのファイルが除去される場合には、ファイルに割り振られていたブロックは、プロセスが終了するまで解放されません。

例:

```
$ ls -l
total 1248
-rwxrwxr-x      1 web   staff   1274770 Jul 20 11:19 datafile
$ date > datafile
$ ls -l
total 4
-rwxrwxr-x      1 web   staff        29 Jul 20 11:20 datafile
```

date コマンドの出力が、datafile ファイルの直前の内容を置き換えました。 切り捨てられたファイルについて報告されるブロックは、1248> から 4 のサイズの差を反映します。ランナウェイ・プロセスがこ

の新規に切り捨てられたファイルに引き続き情報を追加すると、次の **ls** コマンドは、次のような結果を作成します。

```
$ ls -l
total 8
-rxrwxr-x       1 web   staff   1278866 Jul 20 11:21 datefile
```

datafile ファイルのサイズはランナウェイ・プロセスにより行われた追加を反映しますが、割り振られるブロックの数は小さなものです。 datafile ファイルはその中にホールを持つことになります。 ファイル・ホールとは、ディスク・ブロックが割り振られていないファイルの領域のことです。

/ (root) オーバーフロー
ルート・ファイルシステム (/) がいっぱいになったときは、以下について検査してください。

- 次のコマンドを使用して、/etc/security/failedlogin ファイルの内容を読み取ります。

  ```
  who /etc/security/failedlogin
  ```

  TTY の再生成が速過ぎる状態では、失敗するログイン・エントリーが作成される可能性があります。 出力の読み取りまたは保管後にファイルを消去するには、次のコマンドを実行します。

  ```
  cp /dev/null /etc/security/failedlogin
  ```

- /dev ディレクトリーを検査して、デバイス名が誤って入力されていないかどうかを調べます。「rmt0」と入力する ところを rmto とするなどのように、デバイス名を誤って入力した場合は、rmto という名前の /dev にファイルが 作成されます。 コマンドは、通常、ルート・ファイルシステム 全体がいっぱいになるまで進行してから、失敗します。 /dev はルート (**/**) ファイルシステムの 一部です。 デバイスでないエントリー (メジャー番号またはマイナー番号をもたない) を探します。 これを検査するには、次のコマンドを使用します。

  ```
  cd /dev
  ls -l | pg
  ```

  デバイス・ファイルには、通常ファイルのファイル・サイズを示す同じ位置に、コンマで区切られた 2 つの数があります。 例:

  ```
  crw-rw-rw-   1 root     system    12,0 Oct 25 10:19 rmt0
  ```

  次の例に示すように、ファイルの名前またはサイズの位置が無効デバイスを示している場合は、その関連ファイルを除去します。

  ```
  crw-rw-rw-   1 root     system   9375473 Oct 25 10:19 rmto
  ```

  注:

  – /dev ディレクトリーの有効デバイス名を除去しないようにしてください。 無効デバイスを示す 1 つの指標は、関連ファイル・サイズが 500 バイトより大きいことです。
  – システム監査が実行されている場合は、デフォルト /audit ディレクトリー がすぐにいっぱいになり、注意を促します。

- **find** コマンドを使用して、非常に大きな除去可能ファイルを検査します。 例えば、1MB より大きいルート (/) ディレクトリーのすべてのファイルを 見つけるには、次のコマンドを使用します。

  ```
  find / -xdev -size  +2048 -ls |sort -r -n +6
  ```

  このコマンドは、1 MB より大きなすべてのファイルを見付け、それらのファイルを 降順にソートして、一番大きなファイルが最初になるようにします。 この検索では、**-newer** などの、find コマンドのその他のフラグ も役に立ちます。 詳細については、**find** コマンド のコマンド説明を参照してください。

  注: ルート・ディレクトリーを検査する場合、/dev ディレクトリーにあるデバイスの 大きな数および小さな数 に、実際のファイルおよびファイル・サイズが交じります。 コンマで区切られた 大きな数および小さな数は、無視することができます。

ファイルを除去する場合は、まず、次のコマンドを使用して、ファイルが現在ユーザー・プロセスによって使用されていないことを確認します。

```
fuser filename
```

filename は、大きいと疑われるファイル の名前です。 除去のときにファイルがオープンしている 場合、ファイルは、ディレクトリー・リストからのみ除去されます。 ファイルをオープンしているプロセスが終了しないと、そのファイルに割り振られたブロックは解放されません。

/var ファイルシステムのオーバーフローの解決

/var ファイルシステムがいっぱいになったときは、以下のことを検査してください。

- find コマンドを使用して、/var ディレクトリーの 大きなファイルを見付けることができます。 例:

  ```
  find /var -xdev -size  +2048 -ls| sort -r  +6
  ```

  詳しくは、**find** コマンドの説明を参照してください。
- /var/tmp の古くなったファイルまたは 削除されずに残されたファイルを検査します。
- /var/adm/wtmp ファイルのサイズを検査します。このファイルは、すべてのログイン、rlogin、および telnet セッションを記録します。 システム・アカウンティングが実行されていないと、ログは無制限に大きくなります。 システム・アカウンティングは夜間にログを消去します。 /var/adm/wtmp を 消去または編集して、古い情報または不必要な情報を削除することができます。 ログを消去するには、次のコマンドを使用します。

  ```
  cp /dev/null  /var/adm/wtmp
  ```

  /var/adm/wtmp ファイルを編集するには、まず、次のコマンドでファイルを一時的にコピーします。

  ```
  /usr/sbin/acct/fwtmp < /var/adm/wtmp >/tmp/out
  ```

  /tmp/out ファイルを編集して不必要なエントリーを除去してから､次のコマンドで元のファイルを置き換えます。

  ```
  /usr/sbin/acct/fwtmp -ic < /tmp/out > /var/adm/wtmp
  ```
- 次の手順に従って、/var/adm/ras ディレクトリーのエラー・ログ を消去します。 手動でないと、エラー・ログを消去することはできません。

  注 : エラー・ログを消去する際には、**cp /dev/null** コマンドは絶対に使用しないでください。 ゼロの長さの errlog ファイルがあると、オペレーティング・システムのエラー・ログ機能が使用不可になります。バックアップからこのファイルを置き換える必要があります。

  1. 次のコマンドを使用して、エラー・デーモンを停止します。

     ```
     /usr/lib/errstop
     ```
  2. 次のどちらかのコマンドを使用して、エラー・ログ・ファイルを削除するか、別のファイルシステムに移動します。

     ```
     rm /var/adm/ras/errlog
     ```

     または

     ```
     mv /var/adm/ras/errlog filename
     ```

     filename は、移動する errlog ファイルの名前です。

     注 : エラー・ログ・ファイルを削除すると、エラー・データ・ヒストリーが削除されます。
  3. 次のコマンドを使用して、エラー・デーモンを再始動します。

     ```
     /usr/lib/errdemon
     ```

注: **cron** の次のエントリーを実行することによって、errlog を制限することも検討してください。

```
0 11 * * * /usr/bin/errclear -d S,O 30
0 12 * * * /usr/bin/errclear -d H 90
```

- このディレクトリーの trcfile ファイルが大きいかどうかを 検査します。 このファイルが大きくて、現在トレースが実行されていない場合は、次のコマンドを使用してファイルを削除することができます。

```
rm /var/adm/ras/trcfile
```

- ダンプ・デバイスが hd6 (これがデフォルト) に設定されている場合は、/var/adm/ras ディレクトリーにいくつかの vmcore* ファイルが存在する可能性があります。 それらのファイルの日付が古いか、それらを 保持したくない場合は、**rm** コマンドで削除することができます。
- キューイング・サブシステム・ファイルが含まれる /var/spool ディレクトリー を検査します。 次のコマンドを使用して、キューイング・サブシステムを 消去します。

```
stopsrc -s qdaemon
rm /var/spool/lpd/qdir/*
rm /var/spool/lpd/stat/*
rm /var/spool/qdaemon/*
startsrc -s qdaemon
```

- アカウンティング・レコードが含まれる /var/adm/acct ディレクトリー を検査します。 アカウンティングが実行されている場合は、このディレクトリーに 複数の大きなファイルが含まれている可能性があります。
- /var/preserve ディレクトリーを検査して、終了した **vi** セッションの有無を調べます。 一般に、これらのファイルを削除すると安全になります。 セッションをリカバリーしたい場合は、**vi -r** コマンドを使用して、リカバリー可能セッションをリストできます。 特定のセッションをリカバリーするには、**vi -r** *filename* を使用します。
- **su** コマンドの使用が試みられた回数、および各コマンドが成功したかどうかが記録されている、 /var/adm/sulog ファイルを変更します。 このファイルはフラット・ファイル であり、任意のエディターで表示して変更できます。 このファイルを削除した場合は、次に **su** コマンドを実行すると、このファイルを再作成できます。 **snmpd** デーモンからのイベントが記録されている /var/tmp/snmpd.log を変更します。 このファイルを 削除した場合は、**snmpd** デーモンを使用して再作成できます。

  注: 無制限に大きくならないように、/var/tmp/snmpd.log ファイル のサイズを制限することができます。 /etc/snmpd.conf ファイルを編集して、該当するセクションでサイズの数値 (バイト) を 変更します。

その他のファイルシステムの解決および一般の検索技法

-size フラグを指定した **find** コマンドを使用して、大きなファイルを探すか、ファイルシステムが 最近オーバーフローした場合は、 **-newer** フラグを使用して最近変更されたファイルを見つけます。

-newer フラグ用の検索対象ファイルを作成するには、次の touch コマンドを使用します。

```
touch mmddhhmm filename
```

ここで、*mm* は月、*dd* は日、*hh* は 24 時間形式の時間、*mm* は分、*filename* は **touch** コマンドで作成するファイルの名前です。

touch したファイルを作成した後に、次のコマンドを使用して、新規の大規模ファイル を見付けることができます。

```
find /filesystem_name -xdev -newer touch_filename -ls
```

次の例に示すように、**find** コマンドを使用して、24 時間以内に変更されたファイルを見付けることもできます。

```
find /filesystem_name -xdev -mtime 0 -ls
```

マウント

マウント は、ファイルシステム 、ファイル、ディレクトリー、デバイス、およびスペシャル・ファイルを、特定のロケーションで使用できるようにします。 これはファイルシステムをアクセス 可能にする唯一の方法です。

mount コマンドは、ファイルシステムを 指定されたディレクトリーに接続するように、オペレーティング・システムに指示します。

マウントしようとするファイルまたはディレクトリーへのアクセス 権とマウント・ポイントへの書き込み許可を持っていれば、ファイルやディレクトリーのマウントができます。 システム・グループのメンバーもデバイス・マウント (この場合、デバイスまたはファイルシステムはディレクトリーにマウントされる)、および、/etc/filesystems ファイルに記述されているマウントを実行することができます。 root ユーザー権限で操作中のユーザーは、デバイスとディレクトリーの 両方の名前をコマンド・ラインに指定すれば、任意にファイルシステムをマウントできます。 /etc/filesystems ファイルは、システムの 初期設定時に自動的に行うマウントを定義するのに使用されます。 システム 始動後のマウントには **mount** コマンドが使用されます。

マウント・ポイント

マウント・ポイント とは、新規ファイルシステム、ディレクトリー、またはファイルをアクセス可能にするディレクトリーまたはファイルのことです。 ファイルシステムまたはディレクトリーをマウントするためには、マウント・ポイントはディレクトリーである必要があります。

一般的に、ファイルシステム 、ディレクトリー、またはファイルは、空のマウント・ポイントにマウントされますが、それは必須というわけではありません。 マウント・ポイントとして機能するファイルまたはディレクトリーにデータが収められている場合には、そのデータは、別のファイルまたはディレクトリーによってマウントされている間はアクセスできません。 実際、そのようにマウントされたファイルまたはディレクトリーは、そのディレクトリーに前に入っていた内容を収容するようになります。 マウントがその上に行われた元のディレクトリーまたはファイルは、そのマウントが元に戻されると、再度アクセス可能になります。

ファイルシステムがディレクトリーにマウントされると 、マウントされたファイルシステムのルート ・ディレクトリーの権限は、マウント・ポイントの権限より高い優先度を持ちます。 1 つの例外は、その上にマウントが行われたディレクトリー内の .. (ドット・ドット) 親ディレクトリー・エントリーです。 オペレーティング・システムが新規ファイルシステムにアクセスするためには、マウント・ポイントの親ディレクトリー情報が使用可能である必要があります。

例えば、現行作業ディレクトリーが /home/frank である場合、コマンド **cd ..** は作業ディレクトリーを /home に変更します。 /home/frank ディレクトリーが、マウントされたファイルシステムのルートである場合、オペレーティング・システムは、**cd..**コマンドを成功させるために、/home/frank ディレクトリーで親ディレクトリーの情報を検索する必要があります。

親ディレクトリーの情報を必要とするすべてのコマンドが成功するためには、 ユーザーは、その上にマウントが行われたディレクトリー内に検索許可を持っている必要があります。 このようなディレクトリーが検索許可を与えないと、予期できない結果をもたらすことがあります。 特に、この検索許可は表示されないので注意が必要です。 よくある問題は **pwd** コマンドの失敗です。 その上にマウントが行われたディレクトリー内への検索許可を持っていないと、**pwd** コマンドは次のようなメッセージを戻します。

```
pwd: Permission denied
```

この問題は、その上にマウントが行われたディレクトリーへの検索許可を常に 111 以上に設定することで回避できます。

ファイルシステム、ディレクトリー、およびファイルのマウント

2 つのタイプのマウント、リモート・マウントとローカル・マウントがあります。 リモート・マウント は、データが通信回線で送信されるリモート・システム 上で行われます。 ネットワーク・ファイルシステム (NFS) などのリモート・ファイルシステムは、マウントの前に、ファイルがエクスポートされていることを必要とします。 ローカル・マウント はローカル・システムで行われるマウントです。

各ファイルシステムは、それぞれのデバイス (論理ボリューム) に関連付けられます。 ファイルシステムは、ユーザーが使用する前に、既存のディレクトリー構造 (ルート・ファイルシステム 、または既に接続済みの別のファイルシステム) に接続する必要があります。 **mount** コマンドがこの接続を行います。

同一のファイルシステム、ディレクトリー、またはファイルが複数のパスでアクセスできます。 例えば、1 つのデータベースがあり、 このデータベースを複数のユーザーが使用する場合、 同じデータベースを複数マウントすると便利な場合があります。 それぞれのマウントは、トラッキングとジョブの区別の目的で固有の名前とパスワードを持つ必要があります。 これは、同じファイルシステムを異なるマウント・ポイントにマウントすることで実現できます。 例えば、下記の例のように、/home/server/database から、 /home/user1、 /home/user2、および /home/user3 として指定されたマウント・ポイントにマウントできます。

```
/home/server/database        /home/user1
/home/server/database        /home/user2
/home/server/database        /home/user3
```

ファイルシステム、ディレクトリー、またはファイルは、シンボリック・リンクを使用して、さまざまなユーザーに使用可能にできます。 シンボリック・リンクは、**ln -s** コマンドを使って作成できます。 複数のユーザーを中央のファイルにリンクすると、ファイルに対するすべての変更が、ユーザーがそのファイルにアクセスするつど確実に反映されるようになります。

自動マウント制御

マウントは、システムの初期化時に、自動的に行われるように設定することができます。

2 つのタイプの自動マウントがあります。 最初のタイプは、システムのブートと実行に必要なマウントからなります。 これらのファイルシステムはブート・プロセスによって明示的にマウントされます。 /etc/filesystems ファイル内のそのようなファイルシステムのスタンザは、mount = automatic の指定を持ちます。 2 番目のタイプの自動マウントはユーザーが制御します。 /etc/rc スクリプトが **mount all** コマンドを発行すると、これらのファイルシステムがマウントされます。 ユーザー制御の自動マウントのスタンザは、/etc/filesystems に mount = true と指定されています。

/etc/filesystems ファイルは、自動マウントを制御します。 自動マウントは 1 時点に 1 マウント・ポイントずつ、階層的に行われます。 また、配列と再調整ができるように特定の順序で配置されます。 /etc/filesystems ファイルの詳細については、/etc/filesystems を参照してください。

/etc/filesystems ファイルは、マウントあたり 1 つずつ、スタンザに編成されます。 スタンザは、対応するファイルシステムの属性とそのマウント方法について記述しています。 システムは、/etc/filesystems ファイルに現れる順序でファイルシステムをマウントします。 以下は、/etc/filesystems ファイル内のスタンザの例です。

```
/:
 dev=/dev/hd4
 vol="root"
 mount=automatic
 check=false
 free=true
 vfs=jfs
 log=/dev/hd8
 type-bootfs

/home:
 dev=/dev/hd1
 vfs=jfs
 log=/dev/hd8
 mount=true
 check=true
 vol="/home"
 free=false

/usr:
 /dev=/dev/hd2
 vfs=jfs
 log=/dev/hd8
 mount=automatic
 check=false
 type=bootfs
 vol="/usr"
 free=false
```

マウントが行われる順序を制御するために、/etc/filesystems ファイルを編集できます。 1 つのマウントが成功しないと、/etc/filesystems ファイルに定義された後続のマウントのいずれかが続行されま

す。 例えば、/home ファイルシステムのマウントが失敗すると、/usr ファイルシステムに対するマウントが継続し、マウントされます。 マウントは、タイプミス、依存関係、またはシステム上の問題などの理由で失敗する場合があります。

ディスクレス・ワークステーションのマウント・セキュリティー

ディスクレス・ワークステーションは、サーバーから /dev ディレクトリーをマウントするために、リモート・マシン上にデバイス・スペシャル・ファイルを作成し、アクセスする機能を持つ必要があります。 サーバーはクライアント用のデバイス・スペシャル・ファイルとサーバー用のそれとの識別ができないために、サーバー上のユーザーがクライアントのデバイス・スペシャル・ファイルを使用して、サーバーの物理デバイスにアクセスできる場合があります。

例えば、**tty** に対する所有権は、**tty** を使用して自動的にユーザーに設定されます。 ユーザー ID がクライアントとサーバー上で同一でない場合には、サーバー上の権限のないユーザーが、サーバー上の別のユーザーが使用中の **tty** にアクセスできます。

クライアント上の特権のあるユーザーは、デバイス・スペシャル・ファイルを作成し、サーバー上の物理デバイスと突き合わせて、それらにアクセス特権を要求させないようにすることができます。 そうするとユーザーは、サーバー上の特権のないアカウントを使用し、新規のデバイス・スペシャル・ファイルを使用して、通常に保護されているデバイスにアクセスできるようになります。

同様のセキュリティー上の問題として、クライアントとサーバー上での **setuid** プログラムおよび **setgid** プログラムの使用の問題があります。 ディスクレス・クライアントは、通常の操作として、サーバー上で **setuid** プログラムと **setgid** プログラムの作成および実行ができる必要があります。 繰り返しますが、サーバーはサーバー用プログラムとクライアント用プログラムの識別はできません。

さらに、ユーザー ID およびグループ ID がサーバーとクライアントの 間で一致しない場合があるので、サーバー上のユーザーは、本来はサーバー上のユーザー用ではない機能を備えたプログラムを実行できることもあります。

問題は、**setuid** プログラムおよび **setgid** プログラムとデバイス・スペシャル・ファイルは、それらを作成したマシン上でしか使用できないことです。

解決策は、**mount** コマンドに対し、これらのオブジェクトの使用範囲を制限するセキュリティー・オプションを使用することです。 これらのオプションは、/etc/filesystems ファイル内のスタンザでも使用できます。

mount コマンドの nosuid オプションを使うと、このコマンドの結果としてマウントされたファイルシステムをとおしてアクセスされる **setuid** プログラムと **setgid** プログラムの実行が妨げられます。 このオプションは、別のホストによる使用のためにのみ特定のホスト上にマウントされる (例えば、ディスクレス・クライアントのためにエクスポートされる) ファイルシステムに対して使用されます。

mount コマンドの **nodev** オプションを使うと、このコマンドの結果としてマウントされたファイルシステムをとおしてアクセスされるデバイス・スペシャル・ファイルを使用するデバイスのオープンは妨げられます。 また、このオプションは、別のホストによる使用のためにのみマウントされる (例えば、ディスクレス・クライアントのためにエクスポートされる) ファイルシステムに対しても使用されます。

通常、サーバーのユーザーは、/export ディレクトリーへはアクセスできません。

/export/root ディレクトリーのエクスポート

/export/root ディレクトリーは、読み取り/書き込み許可でエクスポートする必要があり、しかも、サーバーの root ユーザーがアクセスできるようにしておく必要があります。 ただし、**mount** コマンドの次のようなオプションを使用して、このディレクトリーをマウントすることが必要になる場合があります。

項目	説明
nosuid	サーバーのユーザーがクライアントの **setuid** プログラムを実行できないようにします。
nodev	ユーザーがクライアントのデバイス・スペシャル・ファイルを使用してサーバーのデバイスにアクセスすることを防止します。

これらのオプションを使用して /export/root ディレクトリーをマウントする方法に代わって、サーバーで実行中のユーザーに /export/root ディレクトリーへのアクセスを許可しないという方法が使用できます。

/export/exec ディレクトリーのエクスポート

/export/exec ディレクトリーは、読み取り専用許可でエクスポートされ、しかも、root アクセスを提供する必要があります。 ただし、**mount** コマンドの次のようなオプションを使用して、このディレクトリーをマウントすることが必要になる場合があります。

項目	説明
nosuid	サーバーのユーザーがクライアントの **setuid** プログラムを実行できないようにします。 サーバーの /usr ディレクトリーをエクスポートしている場合は、**nousid** オプションを使用することはできません。
nodev	ユーザーがクライアントのデバイス・スペシャル・ファイルを使用してサーバーのデバイスにアクセスすることを防止します。

/export/share ディレクトリーのエクスポート

/export/share ディレクトリーは、読み取り専用許可でエクスポートされ、しかも、root アクセスを提供する必要があります。 通常、このディレクトリーには、データだけが (実行可能ファイルやデバイス・ファイルなしで) 入っているため、マウント・セキュリティー・オプションを使用する必要はありません。

/export/home ディレクトリーのエクスポート

ユーザーの /home ディレクトリーをマウントする方法はいくつかあります。

- クライアントの /home ディレクトリーに、/export/home/*Clienthostname* ディレクトリーをマウントすることができます。 この場合、クライアントは読み取り/書き込み許可を持ち、root ユーザーもアクセスができます。 システムのセキュリティーを 保証するには、**mount** コマンドに次のオプションを使用して、/export/home ディレクトリーをマウントしてください。

項目	説明
nosuid	サーバーのユーザーがクライアントの **setuid** プログラムを実行できないようにします。
nodev	ユーザーがクライアントのデバイス・スペシャル・ファイルを使用してサーバーのデバイスにアクセスすることを防止します。

- クライアントの /home ディレクトリーに、サーバーの /home ディレクトリーをマウントすることができます。 この場合、/home ディレクトリーは、root アクセスなしで、読み取り/書き込み許可でエクスポートされます。 システムのセキュリティーを 確保するには、**mount** コマンドの nosuid オプションと nodev オプションを使用して、サーバーおよびクライアントの 両方に /home ディレクトリーをマウントします。
- この代わりとして、サーバー上のそれぞれの /home/*UserName* ディレクトリーを、クライアント上の /home/*Username* ディレクトリーにマウントすることができます。この方法によって、ユーザーは、いろいろなマシンにログインすることが可能になり、しかも、自分のホーム・ディレクトリーにアクセスすることができます。 この場合、サーバーとクライアントの /home/*Username* ディレクトリーは、 両方とも、**mount** コマンドの nousid オプションと nodev オプションを使用してマウントされます。

/export/swap ディレクトリーのエクスポート

/export/swap/*Clienthostname* ファイルは、読み取り/書き込み許可と root アクセスを指定してエクスポートします。 セキュリティーの手段は必要ありません。 サーバーのユーザーは、/export/swap/*Clienthostname* ファイルにアクセスすることはできません。

ディスクレス・マウント
ディスクレス・ワークステーションのファイルシステムは、サーバーの /exports ディレクトリーからマウントされますが、ディスクレス・マシンから見ると、そのファイルシステムはスタンドアロン・マシン上のファイルシステムのように見えます。

以下に、サーバー・エクスポートとディスクレス・ワークステーションのマウント・ポイントとの間の関係を示します。

サーバー・エクスポート	ディスクレス・インポート
/export/root/*HostName*	/（root）
/export/exec/*SPOTName*	/usr
/export/home/*HostName*	/home
/export/share	/usr/share
/export/dump	ディスクレス・クライアントがダンプ・スペースとして使用。
/export/swap	ディスクレス・クライアントがリモート・ページング・スペースとして使用。

/export ディレクトリーの詳細については、79 ページの『/export ディレクトリー』を参照してください。

通常、サーバーのユーザーは、/export ディレクトリーへはアクセスできません。

/export/root ディレクトリーのエクスポート

/export/root ディレクトリーは、読み取り/書き込み許可でエクスポートする必要があり、しかも、サーバーの root ユーザーがアクセスできるようにしておく必要があります。 ただし、**mount** コマンドの次のようなオプションを使用して、このディレクトリーをマウントすることが必要になる場合があります。

項目	説明
nosuid	サーバーのユーザーがクライアントの **setuid** プログラムを実行できないようにします。
nodev	ユーザーがクライアントのデバイス・スペシャル・ファイルを使用してサーバーのデバイスにアクセスすることを防止します。

これらのオプションを使用して /export/root ディレクトリーをマウントする方法に代わって、サーバーで実行中のユーザーに /export/root ディレクトリーへのアクセスを許可しないという方法が使用できます。

/export/exec ディレクトリーのエクスポート

/export/exec ディレクトリーは、読み取り専用許可でエクスポートされ、しかも、root アクセスを提供する必要があります。 ただし、**mount** コマンドの次のようなオプションを使用して、このディレクトリーをマウントすることが必要になる場合があります。

項目	説明
nosuid	サーバーのユーザーがクライアントの **setuid** プログラムを実行できないようにします。サーバーの /usr ディレクトリーをエクスポートしている場合は、**nousid** オプションを使用することはできません。
nodev	ユーザーがクライアントのデバイス・スペシャル・ファイルを使用してサーバーのデバイスにアクセスすることを防止します。

/export/share ディレクトリーのエクスポート

/export/share ディレクトリーは、読み取り専用許可でエクスポートされ、しかも、root アクセスを提供する必要があります。 通常、このディレクトリーには、データだけが (実行可能ファイルやデバイ

ス・ファイルなしで) 入っているため、マウント・セキュリティー・オプションを使用する必要はありません。

/export/home ディレクトリーのエクスポート

ユーザーの /home ディレクトリーをマウントする方法はいくつかあります。

- クライアントの /home ディレクトリーに、/export/home/*Clienthostname* ディレクトリーをマウントすることができます。 この場合、クライアントは読み取り/書き込み許可を持ち、root ユーザーもアクセスができます。システムのセキュリティーを 保証するには、**mount** コマンドに次のオプションを使用して、/export/home ディレクトリーをマウントしてください。

項目	説明
nosuid	サーバーのユーザーがクライアントの **setuid** プログラムを実行できないようにします。
nodev	ユーザーがクライアントのデバイス・スペシャル・ファイルを使用してサーバーのデバイスにアクセスすることを防止します。

- クライアントの /home ディレクトリーに、サーバーの /home ディレクトリーをマウントすることができます。この場合、/home ディレクトリーは、root アクセスなしで、読み取り/書き込み許可でエクスポートされます。 システムのセキュリティーを 保証するには、**mount** コマンドの **nosuid** オプションと **nodev** オプションを使用して、サーバーおよびクライアントの 両方に /home ディレクトリーをマウントします。
- この代わりとして、サーバー上のそれぞれの /home/*UserName* ディレクトリーを、クライアント上の /home/*Username* ディレクトリーにマウントすることができます。この方法によって、ユーザーは、いろいろなマシンにログインすることが可能になり、しかも、自分のホーム・ディレクトリーにアクセスすることができます。 この場合、サーバーとクライアントの /home/*Username* ディレクトリーは、 両方とも、**mount** コマンドの **nousid** オプションと **nodev** オプションを使用してマウントされます。

/export/dump ディレクトリーのエクスポート

/export/dump/*Clienthostname* ディレクトリーは、読み取り/書き込み許可と root アクセスを指定してエクスポートします。 サーバーのユーザーは、/export/dump/*Clienthostname* ファイルにアクセスすることはできません。

/export/swap ディレクトリーのエクスポート

/export/swap/*Clienthostname* ファイルは、読み取り/書き込み許可と root アクセスを指定してエクスポートします。 セキュリティーの手段は必要ありません。 サーバーのユーザーは、/export/swap/*Clienthostname* ファイルにアクセスすることはできません。

ファイルシステムのタイプ

AIX は、複数のファイルシステム・タイプをサポートしています。

下記のようなファイルシステムがあります。

ジャーナル・ファイルシステム (JFS) または拡張ジャーナル・ファイルシステム (JFS2)

一連のファイルシステム・セマンティクス全体をサポートします。 これらのファイルシステムでは、構造上の整合性を保つために、データベースのジャーナリング技法を利用します。 これにより、システムが異常停止したときに、ファイルシステムの 損傷を回避することができます。

それぞれの JFS または JFS2 は、別々の論理ボリュームに存在します。 オペレーティング・システムは、初期化中にファイルシステムをマウントします。 この複数ファイルシステム 構成は、ファイル・ツリーの一部を分離して、 それに関する作業を行えるようにするので、バックアップ、復元、 および修復などのシステム 管理機能を行うのに有用です。

JFS は、ファイルシステム・コマンドのセット全体をサポートする基本的なファイルシステム・タイプです。

JFS2 は、ファイルシステム・コマンドのセット全体をサポートする基本的なファイルシステム・タイプです。

JFS と JFS2 の差は、JFS2 は大きなファイルと大きなファイルシステムをサポートするように 設計されているということです。

ネットワーク・ファイルシステム (NFS)

これは、ユーザーがリモート・コンピューター上に配置されているファイルやディレクトリーにアクセスして、 それらのファイルやディレクトリーをローカルにあるのと同じように使用できるようにする分散ファイルシステムです。 例えば、ユーザーはオペレーティング・システムのコマンドを使用して、リモート・ファイルとディレクトリーに対して、作成、除去、読み取り、書き込み、およびファイル属性の設定を行うことができます。

CD-ROM ファイルシステム (CDRFS)
CD-ROM の内容に、通常のファイルシステム・インターフェース (オープン、読み取り、クローズ) を介してアクセスできるようにします。

DVD-ROM ファイルシステム (UDFS)
これを使用すると、通常のファイルシステム・インターフェースを 通じて DVD の内容にアクセスできます。

JFS および JFS2
ジャーナル・ファイルシステム (JFS) および拡張ジャーナル・ファイルシステム (JFS2) は、基本オペレーティング・システム内に作成されます。 どちらのファイルシステム・タイプも、AIX 論理ボリューム・マネージャーが保管と検索に 使用する構造に対し、ファイル・データとディレクトリー・データをリンクします。

両者の差は、JFS2 の方は 64 ビット・カーネルとより大きなファイルを収容できるように設計されていることです。

以下のセクションでは、これらのファイルシステムについて 説明します。 特に断りがない限り、以下のセクションは JFS および JFS2 の両方に該当します。

***JFS* および *JFS2* 機能**
拡張ジャーナル・ファイルシステム (JFS2) は、既存のジャーナル・ファイルシステム (JFS) よりはるかに大きなファイルを保管できるファイルシステムです。

JFS または JFS2 のいずれかを実装するために選択することができます。 JFS2 は、 AIX 6.1 のデフォルトのファイルシステムです。

注 : JFS ファイルシステムとは 異なり、JFS2 ファイルシステムはディレクトリーのファイル ・タイプでは **link()** API を使用することはできません。 この制限のため、JFS ファイルシステムで 正しく作動するアプリケーションが JFS2 ファイルシステムで 失敗することがあります。

次の表は、JFS および JFS2 の機能の要約です。

機能	JFS2	JFS
フラグメント・サイズおよびブロック・サイズ	ブロック・サイズ (バイト): 512、1024、2048、4096 最大ファイルシステム・サイズ (テラバイト (TB)): 4、8、16、32	フラグメント・サイズ (バイト): 512、1024、2048、4096 最大ファイルシステム・サイズ (ギガバイト (GB)): 128、256、512、1024
最大ファイルシステム・サイズ	32 TB	1 TB
最小ファイルシステム・サイズ	16 MB	適用なし
最大ファイル・サイズ	16 TB	およそ 63.876 GB
i ノード数	動的 (ディスク・スペースによって限定される)	固定 (ファイルシステム 作成時に設定される)
ディレクトリー編成	B ツリー	リニア

機能	JFS2	JFS
圧縮	いいえ	はい
クォータ	はい	はい
エラー・ロギング	はい	はい

注 :

1. 最大ファイル・サイズおよび最大ファイルシステム・サイズは、32 ビット・カーネルで使用されるときは (1 TB - (物理区画サイズ)) に制限されます。 例えば、ボリューム・グループの物理区画サイズが 64 MB ならば、最大ファイルシステム・サイズは (1 TB - 64 MB) = (1048576 MB - 64 MB) = 1048512 MB です。 これは、32 ビット・カーネルの使用時に、論理ボリュームの最大サイズに基本的な制限があるためです。
2. JFS2 は、標準の AIX エラー・ロギング方式をサポートしています。 AIX エラー・ロギングについて詳しくは、「プログラミングの一般概念: プログラムの作成およびデバッグ」のエラー・ログの概要を参照してください。

JFS および *JFS2* ディスク・スペースの分割

多くの UNIX ファイルシステムは、 ファイルおよびディレクトリーの論理分割に使用される論理ブロックと等しいサイズの単位の、 連続するディスク・スペースしか割り振りません。 これらの割り振り単位は一般的にディスク・ブロック と呼ばれ、 単一のディスク・ブロックはファイルまたはディレクトリーの単一の論理ブロック内に入っているデータを保管するためにのみ使用されます。

比較的大きな論理ブロック・サイズ (例えば 4096 バイト) を使用し、論理ブロックと等しいサイズのディスク・ブロック割り振りを維持すると、1 回のファイルシステム 操作で実行しなければならないディスク入出力操作の数を減らす上で有利に働きます。 ファイルまたはディレクトリーのデータは、ディスク上では、多数の小さなディスク・ブロックではなく、少数の大きなディスク・ブロックの形で保管されます。 例えば、4096 バイト以下のサイズのファイルは、論理ブロック・サイズが 4096 バイトであれば、4096 バイトのディスク・ブロックを 1 つ割り振られます。 したがって、読み取りまたは書き込みの操作では、 ディスク上のデータにアクセスするためのディスク入出力操作は 1 回しか必要ありません。 論理ブロック・サイズが小さくて、同じ量のデータに対して複数の割り振りが必要な場合は、 データにアクセスするために複数のディスク入出力操作が必要になります。 また、論理ブロックが大きく、ディスク・ブロック・サイズが等しいということは、 大きなディスク・ブロックにはより大きなデータが入るため、 新規データをファイルやディレクトリーに追加するために実行する必要のあるディスク・スペースの割り振りアクティビティーを減らす上でも有利です。

しかし、ディスク・スペースの割り振り単位を論理ブロック・サイズに制限すると、小さなサイズのファイルやディレクトリーを数多く持つファイルシステムではディスク ・スペースが無駄になります。 ディスク・スペースの無駄は、論理ブロック分のディスク・スペースがファイルやディレクトリーの 部分論理ブロックに割り振られる場合に発生します。 部分論理ブロックには常に論理ブロック分のデータより少ない内容しか収容されないため、部分論理ブロックはそれに割り振られたディスク・スペースの一部分しか消費しません。 他のファイルやディレクトリーは既に割り振られているディスク・スペースにその内容を書き込むことができないため、残りの部分は未使用のまま残ります。 無駄なディスク・スペースの合計量は、多数の小さなファイルとディレクトリーを持つファイルシステムでは大きなものになる可能性があります。

ジャーナル・ファイルシステム (JFS) は、ディスク・スペースをフラグメント と呼ばれる割り振り単位に分割します。 拡張ジャーナル・ファイルシステム (JFS2) は、ディスク・スペースをブロック にセグメント化します。 目的は同じで、データを効率的に保管することです。

JFS フラグメントは、デフォルトのディスク割り振りサイズ (4096 バイト) より小さいサイズです。 フラグメントを使用すると、ファイルまたはディレクトリーの部分論理ブロックにデータがより効率的に保管されるので、無駄なディスク・スペースが最小限に抑えられます。 JFS フラグメント・サポートの機能の動作は、 バークレー・ソフトウェア・ディストリビューション (BSD) のフラグメント・サポートが提供する機能に基づいています。

JFS2 は、512、1024、2048、および 4096 の複数のファイルシステム ・ブロック・サイズをサポートします。 ブロック・サイズを小さくすれば、ファイルまたはディレクトリーの部分論理ブロックへのデータの保管がより効率化されるので、無駄なディスク・スペースが最小になります。 しかしブロック・サイズ

を小さくすると、操作時のオーバーヘッドがさらに増えることにもなります。 JFS2 のブロック・サイズは作成時に指定されます。 ファイルシステムごとにブロック・サイズはそれぞれ異なる場合がありますが、1 つのファイルシステムで使えるブロック・サイズは 1 つだけです。

JFS フラグメント
JFS では、ディスク・スペース割り振り単位はフラグメント と呼ばれます。これは、4096 バイトの論理ブロック・サイズよりも小さくできます。

4096 バイトよりも小さいフラグメントを使用すると、部分論理ブロック内に収容されるデータは、データを保持するのに必要な数のフラグメントのみを使用してより効率的に保管できます。 例えば、500 バイトしかない部分論理ブロックには 512 バイトのフラグメントを割り振ることができるので (512 バイトのフラグメント・サイズを想定して)、無駄なディスク・スペースを大幅に減らすことができます。 部分論理ブロックのストレージ要件が増えると、1 つ以上の追加のフラグメントが割り振られます。

ファイルシステムのフラグメント・サイズは作成時に指定されます。 ジャーナル・ファイルシステム (JFS) で使用できるフラグメント・サイズは 512、1024、2048、および 4096 バイトです。 フラグメント・サイズはファイルシステムごとに 異なる場合がありますが、1 つのファイルシステムでは 1 つのフラグメント・サイズしか使用できません。 フラグメント・サイズが異なっていても 1 つのシステム (コンピューター) に共存することはできるため、ユーザーは各ファイルシステムに最も適したフラグメント・サイズを選択できます。

JFS フラグメント・サポートは、連続するディスク・ブロックではなく、 連続するフラグメントとしてファイルシステムを扱います。 しかし、効率的なディスク操作を維持するために、多くの場合ディスク・スペースは 4096 バイト単位で割り振られ、ディスク・ブロックや割り振り単位は論理ブロックのサイズと同じままになります。 この場合のディスク・ブロックの割り振りは、4096 バイトの連続するフラグメントの割り振りとして扱うことができます。

操作上のオーバーヘッド (追加のディスク・シーク、データ転送、割り振りアクティビティー) およびディスク・スペースの使用率は、 ファイルシステムのフラグメント・サイズが小さくなるほど増大します。 オーバーヘッドの増大と使用可能なディスク・スペースの増大の間の最適バランスを維持するために、下記の要因が JFS フラグメント・サポートに適用されます。

- ファイルまたはディレクトリーの論理ブロックに対しては可能な限り、4096 バイトのフラグメントのディスク・スペースの割り振りが維持されます。
- 32KB 未満のサイズのファイルまたはディレクトリーの 部分論理ブロックにのみ、4096 バイト未満のフラグメントを割り振ることができます。

ファイルシステムのファイルやディレクトリーが 大きくなり、サイズが 32KB を超えると、部分論理ブロックに対して 4096 バイト未満のディスク・スペース割り振りを行う効果は少なくなります。 つまり、合計ファイルシステム・スペースに対するディスク・スペースの節約分の割合が少なくなるのに、小さなディスク・スペース割り振りを維持するための余分なパフォーマンス・コストは一定のままだからです。
4096 バイト未満のディスク・スペースの割り振りは、小さなファイルやディレクトリーで 使用した場合に最も効率的にディスク・スペースを使用できるため、32 KB 以上のファイルやディレクトリーの論理ブロックは常に 4096 バイトのフラグメントに割り振られます。 このような大きなファイルやディレクトリーに関連する部分論理ブロックもまた、4096 バイトのフラグメントに割り振られます。

JFS2 のブロック数
拡張ジャーナル・ファイルシステムは、ディスク・スペースをブロック にセグメント化します。 JFS2 は、512、1024、2048、および 4096 の複数のファイルシステム・ブロック・サイズをサポートします。

ファイルシステムごとにブロック・サイズはそれぞれ異なる場合がありますが、1 つのファイルシステムで使えるブロック・サイズは 1 つだけです。

ブロック・サイズを小さくすれば、ファイルまたはディレクトリーの部分論理ブロックへのデータの保管がより効率化されるので、無駄なディスク・スペースが最小になります。 しかしブロック・サイズを小さくすると、操作時のオーバーヘッドがさらに増えることにもなります。 また、デバイス・ドライバーは、ファイルシステムのブロック・サイズより小さいかまたは同じ大きさのディスク・ブロックをアドレス可能なものでなければなりません。

ディスク・スペースは、4096 バイト以外のブロック・サイズを持つファイルシステムではより 小さな単位で割り振られるため、ファイルやディレクトリーのサイズが 繰り返し拡張される場合、割り振りアクティビティーはより頻繁に行われることになります。 例えば、長さがゼロのファイルのサイズを 512 バイト拡

張する書き込み操作を行うと、ブロック・サイズが 512 バイトであると想定した場合、ファイルに対して 1 つのブロックが割り振られます。 もう一度 512 バイトが書き込まれたためにファイル・サイズが拡張された場合は、ファイルにブロックをもう 1 つ割り振る必要があります。 この例を 4096 バイトのフラグメントを持つファイルシステムで考えてみると、ディスク・スペースの割り振りは最初の書き込み操作の一部として 1 回行われるだけです。 最初の 4096 バイトのブロックの割り振りは 2 回目の書き込み操作で追加されたデータを保持するのに十分な大きさであるため、2 回目の書き込み操作では追加の割り振りアクティビティーは行われません。

ファイルシステムのブロック・サイズは、ファイルシステムの作成時にシステム管理インターフェース・ツール (SMIT)、または **crfs** コマンドや **mkfs** コマンドを使用して指定します。 どのファイルシステム・ブロック・サイズを選択するかは、ファイルシステムに収容されるファイルの予測サイズに基づいて決定する必要があります。

ファイルシステム・ブロック・サイズの値は、システム管理インターフェース・ツール (SMIT)、または **lsfs** コマンドを使用して識別できます。 アプリケーション・プログラムの場合は、**statfs** サブルーチンを使用してファイルシステムのブロック・サイズを識別することができます。

ブロックは、ディスク・スペース割り振りの基本単位としての役割を果たし、ファイルシステム内の各ブロックの割り振り状態は、ファイルシステムのブロック割り振りマップに記録されます。 4096 バイト未満のブロック・サイズを持つファイルシステムに対してブロック割り振りマップを保持するには、仮想メモリーおよびファイルシステムのディスク・スペースを増やす必要がある場合があります。

i ノードの可変数

ディスク・スペースを 4096 バイトより小さいサイズでセグメント化すると、ディスク・スペースの使用効率が最適化されますが、ファイルシステム内に保管できる小さなファイルやディレクトリーの数が増えます。

しかし、ディスク・スペースはファイルやディレクトリーで必要なファイルシステム・リソースの 1 つにすぎません。それぞれのファイルやディレクトリーにはディスク i ノードも必要となります。

JFS および *i* ノード

JFS を使用すると、必要なディスク i ノードの数がデフォルト数よりも多い、または少ない場合に、ファイルシステム内で作成されるディスク i ノードの数を指定できます。

ファイルシステム作成時のディスク i ノードの数は、*i* ノードあたりのバイト数 または *NBPI* と呼ばれる値で指定されます。 例えば、NBPI 値を 1024 にすると、ディスク i ノードは、ファイルシステムのディスク・スペースの 1024 バイトごとに作成されます。 もう 1 つの見方として、NBPI 値が小さい (例えば 512 など) 場合には i ノードの数が多くなり、NBPI 値が大きい (16,384 など) 場合には i ノードの数が少なくなります。

JFS ファイルシステムの場合、ファイルシステムに割り振られる割り振りグループ・スペースの各 NBPI バイトごとに、1 つの i ノードが作成されます。 ファイルシステムの i ノードの合計数によって、ファイルの合計数とファイルシステムの合計サイズが制限されます。 1 つの割り振りグループあたりの i ノード数すべてが割り振られたままですが、1 つの割り振りグループを部分的に割り振ることができます。 NBPI は、ファイルシステム内の i ノードの合計数に反比例します。

JFS は、すべてのファイルシステムを 16M (2^{24}) の i ノードに制限します。

使用できる NBPI 値は、割り振りグループ・サイズ (*agsize*) によって変わります。 デフォルトは 8MB です。 *agsize* が 8 MB の場合に使用できる NBPI 値は 512、1024、2048、4096、8192、および 16,384 です。 これより大きい *agsize* も使用できます。 使用できる *agsize* の値は 8、16、32、および 64 です。 使用可能な NBPI 値の範囲は、*agsize* が増えると増加します。 *agsize* が 2 倍の 16MB になると、NBPI 値の範囲もまた 2 倍、つまり 1024、2048、4096、8193、16384、および 32768 になります。

フラグメント・サイズおよび NBPI 値は、ファイルシステムの作成時にシステム管理インターフェース・ツール (SMIT)、または **crfs** コマンドや **mkfs** コマンドを使用して指定します。 フラグメント・サイズとファイルシステムに対して作成する i ノードの数は、ファイルシステムに収容されるファイルの予測数と予測サイズに基づいて決定されます。

フラグメント・サイズと NBPI 値は、システム管理インターフェース・ツール (SMIT)、または **lsfs** コマンドを使用して識別できます。 アプリケーション・プログラムの場合は、ファイルシステムのフラグメント・サイズを識別するには、`statfs` サブルーチンを使用してください。

JFS2 および *i* ノード
JFS2 では必要に応じて i ノードが割り振られます。

ファイルシステム 内にさらに i ノードを追加できるだけの余地がある場合は、追加の i ノードが自動的に割り振られます。 したがって、使用可能な i ノード数には、ファイルシステムそのもののサイズに 準じた制限があります。

***JFS* および *JFS2* のサイズ制限**
このファイルシステムの 作成時に、JFS の最大サイズを定義します。 JFS をどのようなサイズに定義するかは、いくつかの重大な問題に基づいて決定します。

JFS2 の推奨最大サイズは 16 TB です。 JFS2 の最小ファイルシステム ・サイズは 16 MB です。

4096 バイト未満の割り振り単位を使用するファイルシステムは、4096 バイトのデフォルトの 割り振り単位を使用するファイルシステムよりもかなり 少ないディスク・スペースしか必要としませんが、小さなフラグメントを使用するとパフォーマンス・コストが増大する可能性があります。

ファイルシステム 内の各フラグメント (JFS) またはブロック (JFS2) の割り振り状態は、 そのファイルシステムの割り振りマップに記録されます。 フラグメントまたはブロックのサイズが 4096 バイト未満のファイルシステムの場合、割り振りマップの保持により多くの仮想メモリーとファイルシステム ・ディスク・スペースが必要となる可能性があります。

フラグメント (JFS) またはブロック (JFS2) のサイズが 4096 バイト以外のファイルシステムでは、ディスク・スペースがより小さな単位で割り振られるため、 ファイルやディレクトリーのサイズが 繰り返し拡張された場合、割り振りアクティビティーの 発生頻度が高くなる可能性があります。 例えば、長さがゼロのファイルのサイズを 512 バイト単位で拡張する書き込み操作が 1 回実行されると、結果的に 512 バイトのフラグメントまたはブロック (どちらであるかはファイルシステム ・タイプによる) が 1 つ割り振られることになります。 もう一度 512 バイトの書き込みが行われたためにファイル・サイズがさらに拡張された場合は、 そのファイルにフラグメントまたはブロックをもう 1 つ割り振らなければなりません。 この例を 4096 バイトのフラグメントまたはブロックが使用されるファイルシステムに適用してみると、ディスク・スペースの割り振りは最初の書き込み操作の一部として 1 回行われるだけです。 最初の 4096 バイトの割り振りは 2 回目の書き込み操作で追加されたデータを保持するのに十分な 大きさであるため、2 回目の書き込み操作では追加の割り振りアクティビティーは 必要ありません。 ファイルを一度に 4096 バイト拡張すれば、割り振りアクティビティーを 最小限に抑えられます。

サイズに関連する問題の 1 つは、ファイルシステム ・ログのサイズです。

JFS では、ほとんどの場合、4MB のサイズで構成された共通ログを複数のファイルシステムが 使用します。 例えば初期のインストール後、ルート・ボリューム・グループ内のすべてのファイルシステムは、共通 JFS ログとして論理ボリューム hd8 を使用します。 デフォルトの論理ボリューム区画サイズは 4MB であり、デフォルトのログ・サイズは 1 区画であるため、ルート・ボリューム・グループは通常 4MB の JFS ログを収めています。 ファイルシステムが 2GB を超えるか、単一ログを使用するファイルシステム・スペースの合計量が 2 GB を超えると、デフォルトのログ・サイズでは 十分ではない場合があります。 いずれのケースでも、ファイルシステム ・サイズの増加に応じてログ・サイズは拡大されます。 ログ論理ボリュームのサイズを変更したときは、新しいスペースを使用する前に **logform** コマンドを実行して、そのログを再初期設定する必要があります。 JFS ログは、最大サイズ 256MB に制限されます。

単一の JFS ログがサポートできる結合ファイルシステムのサイズには 、実際的な制限があります。 ガイドラインとして、ファイルシステムの総容量を 1 つの JFS ログあたり 1 兆バイトまでに制限することをお勧めします。 このガイドラインを超えるか、または超えそうになった場合、あるいは (**fsck** コマンドによって呼び出される) **logredo** コマンドでメモリー不足エラーが発生した場合は、JFS ログをもう 1 つ追加して、その 2 つの JFS ログ・ファイル間で負荷を分担させるようにしてください。

JFS2 でもほとんどの場合、複数のファイルシステムも 1 つの共通ログを使用します。 ファイルシステムが 2GB を超えるか、単一ログを使用するファイルシステム・スペースの合計量が 2 GB を超えると、デフォルトのログ・サイズでは 十分ではない場合があります。 どちらの場合も、ファイルシステム ・サイズの増加に応じてログ・サイズを大きくするか、または JFS2 ログをもう 1 つ追加してその 2 つの JFS2 ログ・ファイル間で負荷を分担させるようにできます。

JFS のサイズの制限
最大の JFS サイズはファイルシステムの 作成時に定義されます。 その決定にからむ要因としては、NBPI、フラグメント・サイズ、および割り振りグループ・サイズがあります。

次の計算結果のうち、最小のものがファイルシステム・サイズの制限となります。

$$NBPI * 2^{24}$$

または

$$FragmentSize * 2^{28}$$

例えば、512 という NBPI 率を選択すると、ファイルシステム・サイズは 8GB (512 * 2^{24} = 8 GB) に制限されます。 JFS は、512、1024、2048、4096、8192、16384、32768、65536 および 131072 の NBPI 値をサポートします。

JFS は、すべてのファイルシステムを 16M (2^{24}) の i ノードに制限します。

ファイルシステムに割り振られる割り振りグループ・スペースの各 *NBPI* バイトごとに、1 つの i ノードが作成されます。 1 つの割り振りグループあたりの i ノード数すべてが割り振られたままですが、1 つの割り振りグループを部分的に割り振ることができます。 NBPI は、ファイルシステム 内の i ノードの合計数に反比例します。

JFS は、ファイルシステム・スペースを、i ノードとユーザー・データ用ディスク・ブロックのグループとに分割します。 これらのグループは割り振りグループと呼ばれます。 割り振りグループ・サイズはファイルシステムの 作成時に指定できます。 割り振りグループ・サイズは 8M、16M、32M、および 64M です。 それぞれの割り振りグループ・サイズは関連 NBPI 範囲を持っています。 その範囲は次の表によって定義されます。

メガバイト単位の 割り振りグループ・サイズ	使用可能な NBPI 値
8	512, 1024, 2048, 4096, 8192, 16384
16	1024, 2048, 4096, 8192, 16384, 32768
32	2048, 4096, 8192, 16384, 32768, 65536
64	4096, 8192, 16384, 32768, 65536, 131072

JFS は、それぞれ 512、1024、2048、および 4096 バイト単位の連続したディスク・スペースである 4 つのフラグメント・サイズをサポートします。 JFS は、フラグメント・アドレスを i ノードと間接ブロック内に 28 ビットの数字として保持します。 それぞれのフラグメントは、0 から (2^{28}) の数値でアドレス可能である必要があります。

JFS2 のサイズの制限
非常に大きなファイルが入っている、巨大な JFS2 ファイルシステムの方が、 小さなファイルが多数入っている JFS2 ファイルシステムよりも 維持管理が容易であることが、テストによって判明しています。 大きなファイルシステムに小さなファイルが多数入っている場合は、**fsck** コマンドおよびその他のファイルシステム保守タスクの実行に時間がかかります。

以下に、推奨サイズ制限を示します。

項目	説明
最大 JFS2 ファイルシステム・サイズ :	32TB
最大 JFS2 ファイル・サイズ :	16TB
最小 JFS2 ファイルシステム・サイズ	16MB

JFS のフリー・スペースのフラグメント化
JFS ファイルシステムの場合、4096 バイト未満のフラグメントを使用すると、 ディスクのフリー・スペースのフラグメント化が増大します。

例えば、それぞれが 512 バイトの 8 つのフラグメントに分割されたディスクの領域について考えてみましょう。 それぞれが 512 バイトを必要とする異なるファイルがディスクのこの領域の 1 番目、4 番目、5 番目、および 7 番目のフラグメントに書き込まれ、2 番目、3 番目、6 番目、および 8 番目のフラグメントが空いているとします。 2048 バイトのディスク・スペースを表す 4 つのフラグメントは空き状態ですが、4 つのフラグメント (または 2048 バイト) を必要とする部分論理ブロックはこれらの空きフラグメントには割り振られません。 1 つの割り振りの中のフラグメントは連続している必要があるからです。

ファイルまたはディレクトリーの論理ブロックに対して割り振られるフラグメントは連続している必要があるため、 使用可能なフリー・スペースの合計が操作を満足させるのに十分な大きさである場合でも、 フリー・スペースがフラグメント化されているため、 新しいディスク・スペースを要求するファイルシステム操作は失敗する場合があります。 例えば、長さがゼロのファイルを 1 論理ブロックだけ拡張する書き込み操作は、連続する 4096 バイトのディスク・スペースの割り振りが必要です。 ファイルシステムのフリー・スペースがフラグメント化され、32 の連続していない 512 バイトのフラグメントで合計 16 KB のフリー・ディスク・スペースがある場合、書き込み操作を満足させる 8 つの連続するフラグメント (4096 バイトの連続するディスク・スペース) を使用できないため、書き込み操作は失敗します。

管理が困難なほどフラグメント化されたフリー・スペースがある JFS ファイルシステムは、**defragfs** コマンドを実行してフラグメントを解消することができます。 **defrags** コマンドを実行すると、パフォーマンスにプラスの効果がもたらされます。

疎ファイル
ファイルとは、一つながりの複数の索引ブロックのことです。 ブロックは、i ノードから、それらが表すファイルの論理オフセットにマップされます。

データ・ブロックに対してマップされていない 1 つ以上の索引をもつファイルを、疎に割り当てられている または疎ファイル と呼びます。 疎ファイルにはサイズが関連付けられますが、 サイズ要件を満たすためにすべてのデータ・ブロックを割り当てられるわけではありません。 ファイルが疎に割り当てられているかどうかを確かめるには、**fileplace** コマンドを使います。 すると、ファイル内の現在未割り当て状態のすべてのブロックが示されます。

注 : ほとんどの場合、ファイルに割り当てられているデータ・ブロック数が、 そのサイズのファイルを収容するのに必要な数に一致しているかどうかを判断するのにも、**du** を使うことができます。 圧縮されたファイルシステムでは、疎に割り当てられていないファイルでも、 同じ動作を見せることがあります。

現在割り当てられている索引の外側のロケーションにまで進出してアプリケーションがファイルを拡張したときに、新たに割り当てられた索引のすべてが、書き込まれたデータによって占められたわけではない場合、疎ファイルが作成されます。 その新規のファイルのサイズは、ファイルへの最も遠い書き込みを反映しています。

割り当てが解除されているデータ・ブロックがあるファイルのセクションを読み取ろうとすると、 ゼロしか入っていないバッファーが戻されることになります。 データ・ブロックの割り当てを解除されているファイルのセクションに書き込もうとすると、必要なデータ・ブロックが割り当てられてからデータが書き込まれることになります。

このような動作は、ファイル操作やアーカイブのコマンドに影響を与えることがあります。 例えば、次のようなコマンドの場合、ファイルの疎割り当ては保持されません。

- **cp**
- **mv**
- **tar**
- **cpio**

注 : **mv** の場合、それが当てはまるのは、ファイルを別のファイルシステムに移動するときだけです。 同じファイルシステム内でファイルを移動すると、そのファイルは疎 (互いに隔たりあった) のままになります。

直前のコマンドからファイルをコピーまたは復元すると、 すべてのデータ・ブロックが割り当てられることになるので、 疎の特性が備わることはありません。 ただし、次のようなアーカイブ・コマンドの場合は、疎の特性が保持されるか、 またはファイルがアクティブで疎にされます。

- **backup**
- **restore**
- **pax**

疎ファイルはファイルシステムのリソースの 過剰使用につながる可能性があるので、 この種のファイルの使用と保守は慎重に行う必要があります。

JFS と大きなファイル

JFS ファイルシステム・タイプの大きなファイルを作成できます。

すべての JFS2 ファイルシステムがラージ・ファイルをサポートします。

ラージ・ファイルに使えるファイルシステムは、**crfs** コマンドと **mkfs** コマンドを使って作成できます。 両方のコマンドとも、ラージ・ファイルに使用可能なファイルシステムを指定するためにオプション (`bf=true`) を持っています。 これらのファイルシステムは、SMIT を使用しても作成できます。

ラージ・ファイルに使用可能なファイルシステムでは、4MB のファイル・オフセットの前に保管されるファイル・データは、4096 バイト・ブロック単位に割り振られます。 4MB のファイル・オフセットの後に保管されるファイル・データは、サイズが 128KB のラージ・ディスク・ブロックで割り振られます。 ラージ・ディスク・ブロックは、実際には 32 個の連続した 4096 バイト・ブロックです。

例えば、通常のファイルシステムでは、132 MB のファイルは 33K 個の 4KB ディスク・ブロック (それぞれが 1024 個の 4 KB ディスク・アドレスで埋められた 33 個の単一間接ブロック) を必要とします。 ラージ・ファイルに使用可能なファイルシステムでの 132MB のファイルは、1024 個の 4KB ディスク・ブロックと 1024 個の 128KB ディスク・ブロックから構成されます。 ラージ・ファイルの形状は、132MB のファイルに対し 2 つの単一間接ブロックしか必要としません。 ラージ・ファイル・タイプも通常のファイル・タイプも、ダブル間接ブロックが 1 つ必要です。

ラージ・ディスク・ブロックは、32 個の連続した 4KB ブロックを必要とします。 4MB を超えてラージ・ファイルに書き込みを行う場合、ファイルシステムに 32 個の未使用の連続した 4KB ブロックがないと、ファイル・オフセットが ENOSPC による障害を起こします。

注 : ファイルシステムには数千個の空きブロックがある場合もありますが、その内の 32 個が連続していない場合には、割り振りは失敗します。

defragfs コマンドは、より大きな連続する空きブロック領域を用意するために、ディスク・ブロックを再編成します。

すべての新規ディスク割り振りの初期化のために JFS が必要です。 JFS は、システムに最初のラージ・ファイル使用可能ファイルシステムをマウントするとき、初期ファイル割り振りをゼロにするために使用されるカーネル kproc 手順を始動します。 ラージ・ファイルに使用可能なファイルシステムが 正常にアンマウントされたとき、kproc 手順はそのまま残ります。

JFS のデータ圧縮

JFS は、フラグメント化された圧縮ファイルシステムをサポートしています。 これらのファイルシステムでは、論理ブロックを 4096 バイトのフル・ブロック・サイズよりも小さな 単位 (つまり「フラグメント」) でディスクに保管できるので、ディスク・スペースが節約できます。

JFS2 では、データ圧縮はサポートされません。

フラグメント化されたファイルシステムでは、ファイルの最後の 32 KB 以下の論理ブロックのみがこの方法で保管されるので、 フラグメント・サポートは多くの小さなファイルを持つファイルシステムでのみ有益です。 しかしデータ圧縮では、任意のサイズのファイルのすべての 論理ブロックを 1 つ以上 の連続するフラグメントとして保管できるようにします。 平均すると、データ圧縮では約 2 のファクターだけディスク・スペースを節約します。

ただし、フラグメントとデータ圧縮を使用すると、 ディスクのフリー・スペースがフラグメント化される可能性は高くなります。 論理ブロックに割り振られたフラグメントはディスク上で連続していなければなりません。 フリー・スペースがフラグメント化されているファイルシステムでは、 空きフラグメントの合

計数が論理ブロックの要求数を超えている場合であっても、 論理ブロックの割り振りのために、十分な連続したフラグメントを探すことは困難です。 JFS は、連続するフリー・スペースの量を増やすことでファイルシステムの「フラグメントを解消する」**defragfs** プログラムを提供して、フリー・スペースのフラグメント化を緩和します。 このユーティリティーは、フラグメント化されたファイルシステムと圧縮ファイルシステムで使用できます。 フラグメントやデータ圧縮によってかなりのディスク・スペースが節約され、フリー・スペースのフラグメント化の問題も管理しやすくなります。

現在の JFS のデータ圧縮はこのオペレーティング・システムの旧バージョンと互換性があります。 すべてのシステム・コールから構成される API は JFS の両バージョンで同じです。

JFS データ圧縮のインプリメンテーション
データ圧縮は、ファイルシステムを **crfs** または **mkfs** コマンドで作成するときに指定する、ファイルシステムの属性です。 SMIT を使用して、データ圧縮を指定できます。

重要 : ルート・ファイルシステム (/) は圧縮しないでください。 /usr ファイルシステムの圧縮はお勧めできません。なぜなら、**installp** は、更新および新規インストールの場合に、そのファイルシステムのサイズを 正確に計算できなければならないためです。

圧縮は、通常ファイルおよびこのようなファイルシステムにおける 長いシンボリック・リンクにのみ適用されます。 フラグメント・サポートは、圧縮されていないディレクトリーとメタデータに引き続き適用されます。 ファイルのそれぞれの論理ブロックは、ディスクに書き込まれる前に自動的に圧縮されます。 この方法による圧縮では、ランダム・シークと更新が容易になり、大きな単位でデータを圧縮するのに比べると、失われるフリー・ディスク・スペースの量は少なくなります。

圧縮後は、論理ブロックに必要なディスク・スペースは通常 4096 バイトより少なくなります。 圧縮された論理ブロックはディスクに書き込まれ、 保管に必要な連続するフラグメントの数のみが割り振られます。 論理ブロックが圧縮されない場合には、圧縮されないフォーマットでディスクに 書き込まれ、4096 バイトの連続するフラグメントが割り振られます。

lsfs -q コマンドは、圧縮についての現行値を表示します。 SMIT を使用して、データ圧縮を示すこともできます。

JFS データ圧縮の暗黙的動作
ファイルを書き込むプログラムは、書き込みの成功 (あるいはマップしたファイルの保管の成功) 後にスペース不足 (ENOSPC) 条件が発生することを予測していないため、論理ブロックがディスクに書き込まれるときにそのスペースが使用可能であることを保証する必要があります。

このために、ブロックを圧縮しない場合でも十分なディスク・スペースがあるように、最初の修正時に論理ブロックに 4096 バイトを割り振るようにしています。 4096 バイトの割り振りができない場合には、圧縮された論理ブロックを入れるのに十分なディスク・スペースがある場合でも、システムは ENOSPC や EDQUOT のエラー条件を戻します。 早まったスペース不足条件の報告は、ディスク・クォータの制限あたりで操作する場合や、ほぼ満ぱいとなったファイルシステムで 操作する場合に発生しやすくなります。

圧縮ファイルシステムは、次のような動作を見せることもあります。

- 最初に 4096 バイトが論理ブロックに割り振られるため、システム・コールによっては ENOSPC エラーまたは EDQUOT エラーを受け取る可能性があります。 例えば、古いファイルが mmap システム・コールを使用してマッピングされていることがあり、前に書き込まれたロケーションへの保管操作が ENOSPC エラーを生じさせることがあります。
- データ圧縮では、修正されたブロックがディスクに書き込まれるまで、このブロックにはフル・ディスク・ブロックが割り振られたままになります。 ブロックにフル・ブロックより小さい割り振りが既にコミットされている場合には、ブロックに結び付けられるディスク・スペースの大きさは 2 つの合計となり、以前の割り振りはファイル (i ノード) がコミットされるまで解放されません。 これは通常のフラグメントの場合です。 以前にコミットされた割り振りを持つ可能性のあるファイルの論理ブロックの数は、通常のフラグメントの場合は多くても 1 つですが、圧縮されている場合には最大でファイルの中のブロックの数と同じになります。
- 論理ブロックに対して以前にコミットされたリソースは、fsync システム・コールまたは sync システム・コールがアプリケーション・プログラムから実行されるまでは解放されません。
- stat システム・コールはファイルに割り振られているフラグメントの数を示します。 報告される数は、修正されているが書き込まれていないブロックに割り振られている 4096 バイト、および修正されていな

いブロックの圧縮されたサイズに基づいています。 stat システム・コールは、以前にコミットされたリソースはカウントしません。 stat システム・コールは、修正されたブロックが圧縮されない場合に、i ノードのコミット操作後に割り振られる正しいフラグメント数を報告します。 同様に、ディスク・クォータは現在の割り振りで満たされます。 ファイルの論理ブロックがディスクに書き込まれると、圧縮される場合には論理ブロックに割り振られるフラグメント数は減り、そのためにディスク・クォータと stat の結果は変化します。

JFS のデータ圧縮アルゴリズム
この圧縮アルゴリズムは、LZ の IBM® バージョンです。 一般的に、LZ アルゴリズムは、ある文字列の最初の出現箇所を識別するポインターと、その文字列の長さで 2 番目以降の同じ文字列を表すことによってデータを圧縮します。

圧縮プロセスの最初には文字列はまったく識別されていないため、少なくともデータの最初のバイトは、9 ビット (0, バイト) を必要とする「ロー (raw)」文字として表す必要があります。 指定された大きさ (例えば *N* バイト) のデータが圧縮されると、 圧縮プログラムは、*N* バイト内で、次の未処理バイトで始まる文字列と一致する最長の文字列を探します。 一致した最長文字列の長さが 0 または 1 である場合には、次のバイトは「ロー」文字としてエンコードされます。 それ以外の場合は、文字列は (ポインター、長さ) の組み合わせで表され、処理済みバイト数にはその長さが加えられます。 基本設計上、IBM LZ は、512、1024、または 2048 の *N* の値をサポートしています。 IBM LZ は (ポインター、長さ) ペアと、ロー・キャラクターのエンコードを指定します。 ポインターは log2 *N* のサイズの固定長フィールドであり、長さは可変長フィールドとしてエンコードされます。

JFS データ圧縮のパフォーマンス・コスト
データ圧縮はフラグメント・サポートの拡張機能であるため、フラグメントに関連したパフォーマンス・コストは、データ圧縮にも適用されます。

圧縮ファイルシステムは、パフォーマンスに下記の影響を与えます。

- データの圧縮と解凍には多大の時間が必要であるため、一部のユーザー環境では、圧縮ファイルシステムのユーザビリティーが制限される場合があります。
- ほとんどの UNIX 通常ファイルが書き込まれるのは 1 回のみですが、適宜更新されるものもあります。後者の場合のデータ圧縮では、論理ブロックの最初の修正時に 4096 バイトのディスク・スペースを割り振り、論理ブロックがディスクに書き込まれた後に再度ディスク・スペースを割り振る必要があるため、パフォーマンス・コストが追加されます。 この追加の割り振り処理は、非圧縮ファイルシステムの 通常ファイルでは必要ないものです。
- データ圧縮はプロセッサーのサイクル数を増やします。 ソフトウェア圧縮プログラムの場合、圧縮のサイクル数は 1 バイトあたり約 50 サイクルであり、解凍の場合は 1 バイトあたり 10 サイクルです。

JFS オンライン・バックアップおよび *JFS2* スナップショット
JFS ファイルシステムの時刻指定イメージまたは JFS2 ファイルシステムの時刻指定イメージを作成できます。 このイメージは、後でバックアップのために使用できます。 しかし、各ファイルシステム・タイプに対するこのイメージの要件と振る舞いには差異があります。

JFS ファイルシステムの 場合は、このファイルシステムのミラーリングされたコピーの 読み取り専用静的コピーを分割できます。 通常、ミラーリングされたコピーはオリジナルのファイルシステムが 更新されるたびに必ず更新されますが、この時刻指定コピーは変更されません。 これは、そのコピーが作成された時点に おける静的イメージのままに保たれます。 このイメージをバックアップに使用した場合、このイメージを作成するための手順を開始した時点以降に開始された変更は、そのバックアップ・コピー内には存在しない可能性があります。 したがって、分割が行われている間はファイルシステムのアクティビティーを 最小限にすることをお勧めします。 この分割の完了後に行われたあらゆる変更は、そのバックアップ・コピーには存在しないことになります。

JFS2 ファイルシステムの 場合、時刻指定イメージはスナップショット と呼ばれます。 スナップショットは静的なままで、スナップショットが作成された時点でオリジナルのファイルシステム (*snappedFS* と呼ばれる) が保持していたのと同じセキュリティー許可条件を保持します。 また、JFS2 スナップショットは、ファイルシステムをアンマウントまたは 静止しなくても作成できます。 JFS2 スナップショットは、 スナップショットをとったときに存在していたファイルまたはディレクトリー にアクセスする、または取り外し可能メディアにバックアップを 取るため、ファイルシステムのオンライン ・バックアップとして使用できます。 JFS2 スナップショットについて次のことに注意してください。

- システムがリブートされるとき、root (/) または /usr ファイルシステムのスナップショット ・イメージは上書きされます。 他のファイルシステムのスナップショットは 、リブートの前にファイルシステムのアンマウントにより保持されます。 AIX 5.2 (5200-01 適用) 内に 作成されたスナップショットはリカバリー可能です。 AIX 5.2 (5200-01 適用) で作成されたスナップショットを持つ JFS2 ファイルシステム上 で **fsck** または **logredo** を実行するとき、スナップショットは保持されます。 AIX 5.2 作成の スナップショットを持つ、手際良くアンマウントされたファイルシステムは 、AIX 5.2 (5200-01 適用) システムにいったんマウントされると リカバリー可能です。
- スナップショットを持つファイルシステムに 対して **defragfs** コマンドを実行することは、お勧めできません。 デフラグの実行中に移動される 各ブロックはスナップショットにもコピーしなければならず、時間がかかるのと同時に、スナップショット論理ボリューム内のスペースの無駄にもなります。
- スナップショットのスペースが 使い尽くされた場合、その snappedFS のすべてのスナップショットは削除されます。 この障害が発生すると、エラー・ログにエントリーが 書き込まれます。
- スナップショットへの書き込みが失敗すると、その snappedFS のすべてのスナップショットは削除されます。 この障害が発生すると、エラー・ログにエントリーが 書き込まれます。
- AIX 5.2 (5200-01 適用) システム上で作成またはアクセスされるスナップショットは、AIX 5.2 システム上ではアクセスできません。 ファイルシステムをマウントするためには、 これらのスナップショットを削除する必要があります。
- AIX 5.3 上にスナップショットを持つ JFS2 ファイルシステムは、AIX 5.2 (5200-01 適用) より前のすべてのリリースではアクセスできません。 システムを前のリリースに戻す場合、ファイルシステムへの アクセスを許可するために、まずスナップショットを削除する必要があります。

***JFS* オンライン・バックアップ**

JFS ファイルシステムの 時刻指定イメージを作成できます。このイメージは、 後でバックアップのために使用できます。

JFS ファイルシステムの 場合は、このファイルシステムのミラーリングされたコピーの 読み取り専用静的コピーを分割できます。 通常、ミラーリングされたコピーはオリジナルのファイルシステムが 更新されるたびに必ず更新されますが、この時刻指定コピーは変更されません。 これは、そのコピーが作成された時点に おける静的イメージのままに保たれます。 このイメージをバックアップに使用した場合、このイメージを作成するための手順を開始した時点以降に開始された変更は、そのバックアップ・コピー内には存在しない可能性があります。 したがって、分割が行われている間はファイルシステムのアクティビティーを 最小限にすることをお勧めします。 この分割の完了後に行われたあらゆる変更は、そのバックアップ・コピーには存在しないことになります。

***JFS2* スナップショット**

JFS2 ファイルシステムの 時刻指定イメージを作成できます。このイメージは、後でバックアップのために使用できます。

JFS2 ファイルシステム用の時刻指定イメージはスナップショット と呼ばれます。 スナップショットは静的なままで、スナップショットが作成された時点でオリジナルのファイルシステム (*snappedFS* と呼ばれる) が保持していたセキュリティー許可条件を保持します。 また、JFS2 スナップショットは、ファイルシステムをアンマウントまたは静止しなくても作成できます。 次のこと を行うために JFS2 スナップショットを使用できます。

- スナップショットをとったときに存在していたファイルまたはディレクトリー にアクセスするときに使用できます。
- 取り外し可能メディアにバックアップをとります。

2 つのタイプの JFS2 スナップショット (内部および外部) があります。 JFS2 外部スナップショットは、ファイルシステムからの別個の論理ボリュームで作成されます。 外部スナップショットは自身の固有の マウント・ポイントのファイルシステムから別個にマウントできます。

JFS2 内部スナップショットは、ファイルシステムと 同じ論理ボリューム内で作成され、ファイルシステムからのブロック数を割り振ります。 内部スナップショットは、スナップショット の JFS2 ファイルシステムの root 内の表に出ない .snapshot からアクセス可能 です。 JFS2 ファイルシステムは、ファイルシステムが作成されたとき内部スナップショットを サポートするように使用可能でなければなりません。

JFS2 スナップショットは、ファイルシステムの割り当て量のチェックをサポートしません。 割り当て量の状態を判別するために、スナップショットで **repquota** コマンドを使用する ことはできません。 ファイルシステム・イメージをスナップショット・イメージにロールバックした 場合、時刻指定割り当て量情報は保存されます。 JFS2 外部スナップショットおよび JFS2 内部 スナップショットに固有の次の考慮事項に注意してください。

- AIX 5.2 (5200-01 適用) システム 上で作成またはアクセスされる外部スナップショットは、AIX 5.2 システム上ではアクセスできません。 ファイルシステムをマウントする 前に、 これらのスナップショットを削除する必要があります。
- AIX 5.3 上にスナップショットを持つ JFS2 ファイルシステムは、AIX 5.2 (5200-01 適用) より前のすべてのリリースではアクセスできません。 システムを前のリリースに戻す場合、ファイルシステムへの アクセスを許可するために、まずスナップショットを削除する必要があります。
- デフラグの実行中に移動される各ブロックはスナップショットにもコピーしなければならず、 時間がかかるのと同時に、スナップショット論理ボリューム内のスペースの無駄にもなるので、外部 スナップショットをもつ JFS2 ファイルシステムに対する **defragfs** コマンド の実行は推奨されません。
- ある外部スナップショットによってスペースが 使い尽くされる、または外部スナップショットに障害が起こる場合、スナップショット は 無効であるというマークが付けられます。 スナップショットへのさらなるアクセスには障害が起こります。 これらの障害が発生すると、エラー・ログにエントリーが書き込まれます。

内部 JFS2 スナップショットの考慮事項:

- 内部スナップショットを持つ JFS2 ファイルシステム で **logredo** コマンドが実行されたとき、内部スナップショットは保存されます。
- 修理するため **fsck** コマンドが JFS2 ファイルシステムを 変更しなければならない 場合、内部スナップショットは除去されます。
- ある内部スナップショットによってスペースが 使い尽くされる、または内部スナップショットへの 書き込みに障害が起こる場合、スナップショット は 無効であるというマークが付けられます。 内部スナップショットへのさらなるアクセスには障害が起こります。 これらの障害が発生すると、エラー・ログにエントリーが書き込まれます。
- 内部スナップショットは、別々にマウント可能ではありません。 内部スナップショットが作成された 直後、ファイルシステムのルートの .snapshot ディレクトリー内の 内部スナップショットにアクセスできます。 その結果として、スナップショットの別個の マウント・ポイントをエクスポートする必要がなく、NFS サーバーを介して内部スナップショットに アクセスできます。
- 内部スナップショットは、AIX 6.1 以前 の AIX リリースとは互換性がありません。 内部 スナップショットをサポートするために作成された JFS2 ファイルシステム は、AIX の以前のリリースでは変更できません。
- 内部スナップショットをサポートするために 作成された JFS2 ファイルシステム は、拡張属性バージョン 2 をサポートするために使用可能でもあります。
- 内部スナップショットを持つ JFS2 ファイルシステムは、 データ管理アプリケーション・プログラミング・インターフェース (DMAPI) とともに使用することは できません。
- 内部スナップショットを持つ JFS2 に対して **defragfs** コマンドを使用すること はできません。
- .snapshot ディレクトリーは、**readdir()**システム呼び出しから戻され ません。 これによって、スナップショットへの意図的でないアクセスを防ぎます。 .snapshot ディレクトリーを指定した、**readdir()** システム呼び出しに依存するすべてのシステム呼び出し またはコマンドは、障害が起こります (例えば、.snapshot ディレクトリー の **/bin/pwd** コマンドおよび **getcwd()** システム呼び出しは、親ディレクトリーを見つけられません)。

互換性と移行

JFS ファイルシステムは、AIX 5.1 および AIX 5.2 内で完全に互換性があります。 このオペレーティング・システムの以前にサポートされていたバージョンは、 現行の JFS と互換性がありますが、デフォルト以外のフラグメント・サイズ、NBPI 値、または 割り振りグループ・サイズを使用しているファイルシステムの場合は、前のバージョンに移行する際に特別な注意を要することがあります。

注 : JFS ファイルシステムは、セクター・サイズが 4 KB のディスクではサポートされていません。そのため、ファイルシステムを作成したり、バックアップ操作を実行したりする場合は、ディスクのセクター・サイズが 4 KB ではないことを確認してください。

JFS2 ファイルシステムは、スナップショットを除き、AIX 5.1 および AIX 5.2 では 互換性がありますが、オペレーティング・システムの以前のバージョンの場合は互換性がありません。 スナップショットを使用している JFS2 ファイルシステムは、AIX 5.1 ではサポートされません。 **logredo** コマンドは必ずしも将来のリリース用に作成されたファイルシステムに対して実行されるわけではないので、AIX の以前のバージョンに戻る前にすべての JFS2 ファイルシステムのクリーン ・アンマウントを常に行うようにしてください。

注 : v2 フォーマットで作成された、または同フォーマットに変換された JFS2 ファイルシステムは、AIX の前のリリース上ではアクセスできません。

以下のリストでは、オペレーティング・システムの古いバージョンで作成されたファイルシステムの 場合に問題を起こす可能性のある側面について説明します。

JFS ファイルシステム・イメージ
デフォルト・フラグメント・サイズと 4096 バイトの NBPI 値、および 8 というデフォルトの割り振りグループ・サイズ (agsize) で作成されたあらゆる JFS ファイルシステム・イメージは、特別な移行操作をまったく行わなくても、AIX 4.3 およびこのオペレーティング・システムのそれ以降のバージョンで作成された JFS ファイルシステム ・イメージと交換可能です。

注: JFS2 スナップショット : AIX 5L バージョン 5.2 (5200-01 推奨メンテナンス・パッケージ適用) で作成またはアクセスされる JFS2 スナップショットは、以前のリリースではアクセスできません。 ファイルシステムをマウントするためには、 これらのスナップショットを削除する必要があります。

JFS ファイルシステム間のバックアップと復元
異なるブロック・サイズをもつ JFS ファイルシステムの間でバックアップ・シーケンスと復元シーケンスを実行することはできます。しかし、ソース・ファイルシステムのブロック・サイズがターゲット・ファイルシステムのブロック・サイズよりも小さい場合は、ディスクの使用率が高くなると、空きブロックの不足のために復元操作が失敗することがあります。 これは、ファイルシステム 全体のバックアップ・シーケンスと復元シーケンスに特に関係することであり、ターゲットのファイルシステムの 合計ファイルシステム・サイズがソースのファイルシステムの 合計サイズより大きい場合でも起こる可能性があります。

バックアップ・シーケンスおよび復元シーケンスは、圧縮ファイルシステムから非圧縮ファイルシステムに対して実行したり、異なるフラグメント・サイズを持つ圧縮ファイルシステム 間で実行することができますが、圧縮ファイルシステムのディスクの 使用率が増えたことが原因で、ディスク・スペースの不足のために復元操作が失敗することがあります。 これは、ファイルシステム 全体のバックアップ・シーケンスと復元シーケンスに特に関係することであり、ターゲットのファイルシステムの 合計ファイルシステム・サイズがソースのファイルシステムの 合計サイズより大きい場合でも起こる可能性があります。

JFS および JFS2 でのデバイス・ドライバーの制限
デバイス・ドライバーは、JFS ファイルシステムのフラグメント ・サイズまたは JFS2 の ブロック・サイズと同じかそれより小さいサイズのディスク ・ブロックをアドレッシングできなければなりません。例えば、ユーザー提供の RAM ディスク・デバイス・ドライバー上に JFS ファイルシステムが 作成された場合、512 バイトのフラグメントを持つファイルシステムを収めるためには、そのドライバーで 512 バイトのブロックを使用できる必要があります。 ドライバーがページ単位のアドレスしか行えない場合には、4096 バイトのフラグメント・サイズの JFS しか使用できません。

JFS の別の物理ボリュームへのコピー
JFS ファイルシステムは、ファイルシステムの 保全性を保ちながら、別の物理ボリュームにコピーできます。

以下のシナリオでは、ファイルシステムの 保全性を保ちながら、JFS ファイルシステムを 別の物理ボリュームへコピーする 方法について説明します。

ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

ファイルシステムの保全性を維持しながら、JFS を他の物理ボリュームにコピーするには、以下のようにします。

1. コピーしたいファイルシステムの 活動を停止します。 ファイルシステムを 使用しているアプリケーショ ンが静止していないか、ファイルシステムの 状態が分からない限り、バックアップ・データの内容を知ることはできません。
2. コマンド・ラインに次の SMIT 高速パスを入力して、論理ボリュームをミラーリングします。

   ```
   smit mklvcopy
   ```

3. 次のコマンドを実行して、ファイルシステムをコピーします。

   ```
   chfs -a splitcopy=/backup -a copy=2 /testfs
   ```

   -a フラグに **splitcopy** パラメーターを使用すると、コマンドはファイルシステムをミラーリングしたコピーを分離し、それを新規マウント・ポイントに読み取り専用でマウントします。 この処理を行うと、ファイルシステムのコピーに、バックアップに使用できる、ジャーナル処理された整合性のあるメタデータを提供できます。
4. ミラーリングしたコピーを別のマウント・ポイントに移動したい場合は、次の SMIT 高速パスを実行します。

   ```
   smit cplv
   ```

この時点で、ファイルシステムのコピーを 使用することができます。

CD-ROM ファイルシステムおよび UDF ファイルシステム

CD-ROM ファイルシステム (CDRFS) は、CD-ROM メディア、CD-RW メディア (書き込み保護の場合)、および DVD-ROM メディアに格納することができる、読み取り専用のローカル・ファイルシステムの 実装です。 CDRFS ファイルの最大サイズは、使用するメディアにかかわらず 2 GB です。 ユニバーサル・ディスク・フォーマット (UDF) ファイルシステムは、読み取り専用として DVD-ROM メディアに格納される、または DVD-RAM に読み取り/書き込み用として格納される可能性がある、書き込み可能なローカル・ファイルシステムの実装です。

CD はデフォルトで自動的にマウントされますが、このフィーチャーは使用不可にできます。 このフィーチャーが使用不可にされている場合は、**cdmount** コマンドを使用して CDRFS ファイルシステムをマウントします。

AIX は、以下の CDRFS ボリュームとファイル構造フォーマットをサポートしています。

タイプ	説明
ISO 9660:1988(E) 規格	CDRFS は ISO 9660 レベル 3 の交換と、レベル 1 のインプリメンテーションをサポートしています。
High Sierra Group 仕様	ISO 9660 に優先し、以前の CD-ROM との下位互換を提供します。
Rock Ridge Group プロトコル	ISO 9660 規格に完全準拠の ISO 9660 への拡張を指定し、System Use Sharing Protocol (SUSP) と Rock Ridge Interchange Protocol (RRIP) に基づく完全な POSIX ファイルシステム・セマンティクスを備え、他の UNIX ファイルシステムと同様に CD-ROM のマウント/アクセスを可能にします。
CD-ROM eXtended Architecture File Format (モード 2 フォーム 1 セクター・フォーマットのみ)	CD-ROM eXtended Architecture (XA) ファイル・フォーマットは、例えば、Photo CD など、CD-ROM ベースのマルチメディア ・アプリケーションで 使用される ISO 9660 への拡張を指定します。

すべてのボリュームとファイル構造フォーマットについて、下記の制約が適用されます。

- 単一ボリュームのボリューム・セットのみ

- 非インターリーブ・ファイルのみ

CDRFS は、物理セクター・フォーマット (CD-ROM モード 1 と CD-ROM XA モード 2 フォーム 1) とディスクのマルチセッション・フォーマット (最後のセッションのボリューム認識領域からのボリューム・ディスクリプター・セットをマッピングする) の透過性を提供する、基礎となる CD-ROM デバイス・ドライバーに依存しています。

注 : CDRFS をシステムからアンマウントしてからでないと 、CD-ROM メディアを取り外せません。

もう 1 つのファイルシステム・タイプ UDFS がサポートされるようになりました。これは、DVD-ROM メディアに保管される読み取り専用のファイルシステムです。 UDFS をシステムからアンマウントしてからでないと、メディアを取り外せません。 AIX で サポートされるのは、UDFS フォーマット・バージョン 1.50、2.00、および 2.01 です。

UDFS は、NFS を使用して、読み取り専用モードでエクスポートする必要があります。 NFS をマウントした UDFS への書き込みはサポートされません。

cdmount コマンドを使用して自動的に読み取り/書き込み UDFS をマウントするには、`cdromd.conf` ファイルを編集します。 読み取り/書き込み UDFS のマウントは、**mount** コマンドを使用して手動でも行えます。

ディレクトリー

ディレクトリー とは、ファイルやほかのディレクトリーにアクセスするために必要な情報のみが入っている独特なタイプのファイルです。 結果として、ディレクトリーの 占有するスペースは、ほかのタイプのファイルよりも少なくて済みます。

ファイルシステム は、ディレクトリーと、そのディレクトリー内のファイルのグループから構成されます。ファイルシステムは、普通、逆向きのツリーで表されます。 スラッシュ (/) 記号で表されるルート・ディレクトリーによって、ファイルシステムは定義され、このルート・ディレクトリーが、ファイルシステム・ツリー構造の図の上端になります。

ツリー構造の図では、ルート・ディレクトリーから各ディレクトリーが下方へブランチしており、これらのディレクトリーには、それぞれファイルとサブディレクトリーの両方を含めることができます。 ブランチにより、ディレクトリー構造を通してファイルシステム 内のすべてのオブジェクトにいたる 固有のパスが作成されます。

ディレクトリー には、ファイルのコレクションが保管されます。 多くの場合、このようなファイルのコレクションは、相互に関連しています。これらのコレクションをディレクトリー構造に保管することにより、編成が保持されます。

ファイル とは、読み取りと書き込みが可能なデータのコレクションです。 ファイルの内容は、作成されたプログラム、記述されたテキスト、獲得されたデータ、使用されるデバイスのいずれかです。 コマンド、プリンター、端末装置、通信文、アプリケーション・プログラムなどはすべて、ファイルに保管されます。これにより、ユーザーは、一貫した方法でシステムの 各種のエレメントにアクセスできます。またこのことにより、ファイルシステムの 柔軟性が高められます。

ディレクトリーは、ファイルとほかのディレクトリーをグループ 化して、ファイルシステムをモジュラー階層に編成し、さらに、ファイルシステム 構造に柔軟性と奥行きを与えます。

ディレクトリーには、ディレクトリー項目が入っています。 各項目には、ファイル名またはサブディレクトリー名と、索引ノード参照番号 (*i* ノード 番号) が含まれています。 スピードを早め、ディスク・スペースの使用を拡張するために、ファイルのデータは、コンピューターのメモリーのさまざまな位置に保管されています。 i ノード番号には、ファイルに関連する分散されたデータ・ブロックすべての位置を示すアドレスが入っています。 また、i ノード番号には、ファイルについてのその他の情報も記録されます。例えば、修正とアクセスの回数、アクセス・モード、リンク数、ファイル所有者、ファイル・タイプなどが記録されます。

コマンドの特殊セットでディレクトリーを制御します。 例えば、**ln** コマンドを使ってディレクトリー項目を作成すると、1 つのファイルの複数の名前を同一の i ノード番号にリンクすることができます。

多くの場合、ディレクトリーには、システムの 一部のユーザーしか使用してはならない情報が入っているので、ディレクトリーへのアクセスをプロテクトできます。 ディレクトリーの許可を設定して、そのディ

レクトリーへアクセスできるユーザーを制限することができます。また、ディレクトリー内の情報を変更できるユーザー (ただし、このようなユーザーが存在する場合) を決めることもできます。

ディレクトリーのタイプ

ディレクトリーは、オペレーティング・システム、システム管理者、ユーザーのいずれかによって定義されます。

システム定義のディレクトリーには、特定の種類のシステム・ファイル (コマンドなど) が入っています。ファイルシステムの階層では、システムの定義する / (ルート) ディレクトリーが先頭になります。 通常、/ (ルート) ディレクトリーには、次のような標準のシステム関連のディレクトリーが入っています。

項目	説明
/dev	入出力デバイスのためのスペシャル・ファイルが入っています
/etc	システム初期化とシステム管理のためのファイルが入っています
/home	システム・ユーザーのためのログイン・ディレクトリーが入っています
/tmp	一時的なファイルで、指定の日数後に自動的に削除されるファイルが入っています
/usr	lpp、include およびその他のシステム・ディレクトリーが入っています
/usr/bin	ユーザーの実行可能プログラムが入っています

ログイン・ディレクトリー、ホーム・ディレクトリー ($HOME) などのいくつかのディレクトリーは、システム管理者によって定義され、カスタマイズされます。 オペレーティング・システムにログインした時点では、ログイン・ディレクトリーが現行ディレクトリーとなります。

ユーザーが作成するディレクトリーは、ユーザー定義 のディレクトリーと呼ばれます。 このようなディレクトリーにより、使用するファイルを編成し保守することができます。

ディレクトリー編成

ディレクトリーには、ファイルかサブディレクトリー、またはその両方が入っています。 サブディレクトリー とは、ディレクトリー内にあるディレクトリーのことです。サブディレクトリーが入っているディレクトリーは、親ディレクトリー と呼ばれます。

各ディレクトリーに、それ自体が作成された親ディレクトリーを表すエントリー .. (2 つのドット) と、そのディレクトリー自体を表すエントリー . (ドット) があります。 多くのディレクトリー・リストでは、このようなファイルは隠されています。

ディレクトリー・ツリー
ディレクトリーのファイルシステム 構造は、複雑になりがちです。ファイルとディレクトリーの構造を、できる限り単純な構造に保つようにしてください。ファイルとディレクトリーを作成するときには、簡単に識別できるような名前を付けてください。 これにより、ファイルでの作業が容易になります。

親ディレクトリー
/ (ルート) を除く各ディレクトリーは、1 つの親ディレクトリーをもっており、子ディレクトリーをもつことができます。

ホーム・ディレクトリー
システムはユーザーがログインすると、ホーム・ディレクトリー またはログイン・ディレクトリーと呼ばれるディレクトリーに導きます。 このようなディレクトリーは、各ユーザー用に、システム管理者によって設定されます。 ホーム・ディレクトリーは、そのユーザーのパーソナル・ファイルのリポジトリーです。 通常、ユーザーが自分のために作成するディレクトリーは、ホーム・ディレクトリーのサブディレクトリーとなります。 任意の時点でホーム・ディレクトリーに戻るには、プロンプトのとき **cd** コマンドを入力して、Enter キーを押します。

作業ディレクトリー
作業は、常にディレクトリー内で行われます。 現在作業を行っているディレクトリーは、現在の ディレクトリーまたは作業 ディレクトリーと呼ばれます。 **pwd** (Present Working Directory) コマンドは、作業ディレクトリーの名前を報告します。 作業ディレクトリーを変更するには、**cd** コマンドを使用します。

ディレクトリーの命名規則
各ディレクトリーの名前は、そのディレクトリーが保管されているディレクトリー内で、固有でなければなりません。 これにより、各ディレクトリーは、ファイルシステムで固有のパス名を持つことになります。

ディレクトリーは『ファイル命名規則』で説明するファイルの場合と同じ命名規則に従って命名されます。

ディレクトリーの省略形
省略形は、特定のディレクトリーを指定する便利な方法です。

次に、省略形のリストを示します。

省略形	意味
.	現行作業ディレクトリー
..	現行作業ディレクトリーの 1 つ上のディレクトリー (現行ディレクトリーの親)。
~	ユーザーのホーム・ディレクトリー。 (Bourne シェルの場合は該当しません。 詳しくは、『Bourne シェル』を参照してください。)
$HOME	ユーザーのホーム・ディレクトリー。 (すべてのシェルの場合。)

ディレクトリーのパス名
各ファイルとディレクトリーは、ファイルシステム・ツリー構造内で、パス名 と呼ばれる固有のパスを使って検索されます。 パス名は、ファイルシステム 内でのディレクトリーまたはファイルの 位置を示します。

注 : パス名の長さは、1023 文字を超えることはできません。

ファイルシステムでは、次の種類のパス名を使用します。

項目	説明
絶対パス名	/ (ルート) ディレクトリーからパスをトレースします。 絶対パス名は、必ずスラッシュ (/) 記号で始まります。
相対パス名	現行ディレクトリーから、その親ディレクトリーまたはサブディレクトリーおよびファイルまでのパスをトレースします。

絶対パス名は、/ (ルート) ディレクトリーから下方へ向かう、完全なディレクトリー名またはファイル名です。 絶対パス名を指定すると、現在、ファイルシステムのどの 位置で作業しているかにかかわらず、必ずディレクトリーまたはファイルを検索できます。 絶対パス名は、ルート・ディレクトリーを表すスラッシュ (/) 記号で始まります。 パス名 /A/D/9 が、9 の絶対パス名です。先頭のスラッシュ (/) は / (ルート) ディレクトリーを表し、これが検索の開始位置になります。 パス名の残りの部分では、A、D、そして最後に 9 の順に検索先を指定します。

9 という名前のファイルが 2 つ存在し得ます。これは、ファイルまでの絶対パス名が各ファイルに付くことによって、ファイルシステム 内での固有の名前にしてしまうからです。 パス名 /A/D/9 と /C/E/G/9 では、9 という名前の 2 つの固有のファイルを示します。

絶対パス名とは異なり、相対パス名では、現行作業ディレクトリーを基準にして、ディレクトリーまたはファイルを指定します。 相対パス名の場合は、ドット・ドット (..) 表記は、ファイルシステムの 階層の中を上方へ移動することを意味します。 ドット・ドット (..) は、親ディレクトリーを表します。 相対パス名では現行ディレクトリーからのパスを示すため、パス名の先頭がスラッシュ (/) になることはありません。 相対パス名は、現行ディレクトリー内のファイルの名前、またはファイルシステム 内で現行ディレクトリーのレベルより上または下にあるファイルまたはディレクトリーのパス名を示すのに使用されます。 D が現行ディレクトリーである 場合、10 にアクセスするための相対パス名は F/10 です。 しかし、絶対パス名は常に /A/D/F/10 です。 また、3 にアクセスするための相対パス名は ../../B/3 です。

また、ドット (.) 表記を使用して、現行ディレクトリーの名前を表すことができます。 ドット (.) 表記が通常使用されるのは、現行ディレクトリー名を読み取るプログラムの実行時です。

ディレクトリーを作成する (mkdir コマンド)

Directory パラメーターで指定された、1 つ以上のディレクトリーを作成するには、**mkdir** コマンドを使用します。

各新規ディレクトリーには、それぞれ標準項目であるドット (.) およびドット・ドット (..) が入っています。 **-m** *Mode* フラグ を使用すると、新規ディレクトリーの許可を指定できます。

ディレクトリーを作成するとき、そのディレクトリーは、ファイルシステムの別の位置への絶対パス名を指 定しない限り、現行ディレクトリー、つまり作業ディレクトリー内に作成されます。

mkdir コマンドの使用例を以下に示します。

- 現行作業ディレクトリーに Test という名前の新規ディレクトリーを作成し、デフォルトの許可を与えるには、次のように入力します。

  ```
  mkdir Test
  ```

- 以前に作成した /home/demo/sub1 ディレクトリーに Test という名前のディレクトリーを 作成し、rwxr-xr-x の許可を与えるには、次のように入力します。

  ```
  mkdir -m 755 /home/demo/sub1/Test
  ```

- /home/demo/sub2 ディレクトリーに Test という名前のディレクトリーを作成し、デフォルトの許可を 与えるには、次のように入力します。

  ```
  mkdir -p /home/demo/sub2/Test
  ```

  この **-p** フラグ は、/home、/home/demo、および /home/demo/sub2 の各ディレクトリーがすでに存在しない場合に、 これらのディレクトリーを作成します。

完全な構文については、「コマンド・リファレンス 第 *3* 巻」の **mkdir** コマンドを参照してください。

ディレクトリーを移動または名前変更する (mvdir コマンド)

mvdir コマンドを使用して、ディレクトリーを移動または名前変更します。

mvdir コマンドの使用例を以下に示します。

- ディレクトリーを移動するには、次のように入力します。

  ```
  mvdir book manual
  ```

  これにより、manual という 名前のディレクトリーが存在する場合には、book ディレクトリーを、manual ディレクトリーの下に移動します。 manual という 名前のディレクトリーが存在しない場合は、book ディ レクトリーが manual という名前に変更されます。

- ディレクトリーを移動して名前変更するには、次のように入力します。

  ```
  mvdir book3 proj4/manual
  ```

  manual と いう名前のディレクトリーがすでに存在している場合は、book3 およびその内容は proj4/manual に 移動されます。 言い換えれば、book3 は、proj4/manual のサブディレクトリーに成ります。 manual が存在しない場合は、これにより book3 ディレクトリーは、proj4/manual に名前変更されます。

完全な構文については、「コマンド・リファレンス 第 *3* 巻」の **mvdir** コマンドを参照してください。

現行ディレクトリーを表示する (pwd コマンド)

現行ディレクトリーの絶対パス名 (/ (ルート) ディレクトリーからのパス名) を 標準出力に書き込むには、**pwd** コマンドを使用します。

各ディレクトリーは、スラッシュ (/) で区切られます。 / (ルート) ディレクトリーは、先頭のスラッシュ (/) で表されます。また、最後のディレクトリー名が現行ディレクトリーです。

例えば、現行ディレクトリーを表示するには、次のように入力します。

```
pwd
```

現行 ディレクトリーの絶対パス名が、次のように表示されます。

```
/home/thomas
```

別のディレクトリーへ移動する (cd コマンド)

現行ディレクトリーから別のディレクトリーへ移動するには、**cd** コマンドを使用します。 指定するディレクトリーには、実行 (検索) の許可を 持っている必要があります。

Directory パラメーターを指定しない場合には、**cd** コマンドは、ログイン・ディレクトリー (**ksh** 環境と **bsh** 環境では $HOME、**csh** 環境では $home) へ移動します。 指定されたディレクトリー名が 絶対パス名であれば、このディレクトリーが現行ディレクトリーになります。 絶対パス名は、/(root) ディレクトリーを示すスラッシュ (/)、現行ディレクトリーを示すドット (.)、または親ディレクトリーを示すドット・ドット (..) から始まります。 ディレクトリー名 が絶対パス名でなければ、**cd** コマンドは、*$CDPATH* シェル変数 (または *$cdpath* **csh** 変数) に 指定されたパスのうちいずれか 1 つに対して、相対的なディレクトリー名を検索します。 この変数の構文は、*$PATH* シェル変数 (または *$path* **csh** 変数) の構文と同一で、同じようなセマンティクスを持っています。

cd コマンド の使用例を以下に示します。

- ホーム・ディレクトリーに移動するには、次のように入力します。

  ```
  cd
  ```

- /usr/include ディレクトリーに移動するには、次のように入力します。

  ```
  cd /usr/include
  ```

- ディレクトリー・ツリー内で 1 レベル下の sys ディレクトリーに移動するには、次のように入力します。

  ```
  cd sys
  ```

  現行ディレクトリー が /usr/include で、sys というサブディレクトリーが入っている場合、 /usr/include/sys が現行ディレクトリーと なります。

- ディレクトリー・ツリー内で 1 レベル上に移動するには、次のように入力します。

  ```
  cd ..
  ```

  スペシャル・ファイル名 であるドット・ドット (..) は、現行ディレクトリーのすぐ上のディレクトリー、つまりその親ディレクトリーを表します。

完全な構文については、「コマンド・リファレンス 第 *1* 巻」の **cd** コマンドを参照してください。

ディレクトリーをコピーする (cp コマンド)

SourceFile または *SourceDirectory* パラメーターによって指定されたファイルま たはディレクトリーの内容を、*TargetFile* または *TargetDirectory* パラメーターによって指定されたファイ ルまたはディレクトリーにコピーするには、**cp** コマンドを使用します。

TargetFile として指定されたファイルが存在する場合には、ファイルの元の内容はコピーにより上書きされます。 複数の *SourceFile* をコピーする場合には、 ターゲットはディレクトリーでなければなりません。

SourceFile のコピーをディレクトリーに置くには、*TargetDirectory* パラメーターに対して、既存のディレクトリーへのパスを指定してください。 このパスの最後に新しいファイル名を指定しない限り、ファイルのそれぞれの名前は、 ディレクトリーへコピーされてからもファイルに 保持されます。 また、**cp** コマンドは、**-r**、**-R** のいずれかのフラグを指定すると、複数のディレクトリー全体を、ほかの複数のディレクトリーへコピーします。

cp コマンドの 使用例を以下に示します。

- /home/accounts/customers/orders ディレクトリーの中のすべてのファイルを /home/accounts/customers/shipments ディレクトリーに コピーするには、次のように入力します。

```
cp /home/accounts/customers/orders/* /home/accounts/customers/shipments
```

  これにより、 ディレクトリーではなく、ファイルが、orders ディレクトリーから、shipments ディレクトリーにコピーされます。
- あるディレクトリーを、そのすべてのファイルとサブディレクトリーと 共に、別のディレクトリーにコピーするには、次のように入力します。

```
cp -R /home/accounts/customers /home/accounts/vendors
```

  これに より、customers ディレクトリーが、そのすべてのファイル、サブディレクトリー、およびそれら サブディレクトリーの中のファイルとともに、vendors ディレクトリーへコピーされます。

完全な構文については、「コマンド・リファレンス 第 *1* 巻」の **cp** コマンドを参照してください。

ディレクトリーの内容を表示する (ls コマンド)

ディレクトリーの内容を表示するには、**ls** コマンドを使用してください。

ls コマンドは、指定された各 *Directory* の内容または指定された各 *File* の名前とともに、フラグで指定されたその他の情報を標準出力に書き出します。 *File* または *Directory* が指定されなかった場合には、**ls** コマンドは、現行のディレクトリーの内容を表示します。

デフォルトでは、**ls** コマンドは、ファイル名のアルファベット順に全情報を表示します。 コマンドが root 権限をもつユーザーによって実行された場合は、デフォルトで **-A** フラグが使用されて、(.) ドットおよび (..) ドット・ドット以外のすべての項目のリストが表示されます。 ドット (.) で始まるファイルを含むすべてのファイル項目を表示するには、**ls -a** コマンドを使用してください。

出力は、以下の方法でフォーマットできます。

- **-l** フラグを使用して、1 行に 1 エントリーずつリストします。
- **-C** または **-x** のいずれかのフラグを指定して、複数のカラムにエントリーをリストします。 tty への出力の場合には、**-C** フラグがデフォルトのフォーマットです。
- **-m** フラグを指定して、コンマで区切られて一列になった複数のエントリーをリストします。

出力行における文字位置の番号を決めるため、**ls** コマンドは、*$COLUMNS* 環境変数を使用します。 この変数が設定されていない場合には、コマンドは、terminfo ファイルを読み取ります。 **ls** コマンドが、このどちらの方法でも文字位置の番号を決定できないときには、デフォルト値の 80 を使用します。

-e と **-l** フラグが指定された場合に表示される情報は、次のように解釈されます。

各項目の先頭文字は、次のうちのいずれかです。

項目	説明
d	項目はディレクトリーです
b	項目はブロック・スペシャル・ファイルです
c	項目はキャラクター・スペシャル・ファイルです
l	項目はシンボリック・リンクです
p	項目は先入れ先出し (FIFO) パイプ・スペシャル・ファイルです
s	項目はローカル・ソケットです
-	項目は通常のファイルです。

2 文字目以降の 9 文字は、3 文字ごとに 3 つのセットに分けられます。 最初の 3 文字は、ファイルまたはディレクトリーの所有者の許可を示します。 次の 3 文字は、グループ内のほかのユーザーの許可を示します。 最後の 3 文字は、そのファイルに対するアクセス許可を持つその他のユーザーの許可を示します。 各

セットの 3 文字は、ファイルに対する読み取り、書き込み、実行の許可を示しています。 ディレクトリーに対して実行の許可を持っていると、そのディレクトリー内の指定のファイルを検索できます。

許可は、次のように示されます。

項目	説明
r	読み取り権限が認可されている
t	ディレクトリーやファイルの所有者以外の人が、ディレクトリーへの書き込み許可を持っていても、ディレクトリーのファイルの削除や名前変更は所有者しか行えない
w	書き込み (編集) 権限が認可されている
x	実行 (検索) 権限が認可されている
-	対応する権限が認可されていない

-e フラグが指定された場合に表示される情報は、11 番目のキャラクターが追加されるという点を除けば、**-l** フラグが指定された場合と同じです。このキャラクターは、次のように解釈します。

項目	説明
+	ファイルに拡張セキュリティー情報が含まれていることを示します。 例えば、ファイルに、そのモードの拡張 ACL、TCB、または TP 属性が含まれていることがあります。
-	ファイルに拡張セキュリティー情報が含まれていないことを示します。

ディレクトリーの各ファイルのサイズがリストされる場合には、**ls** コマンドによって、間接ブロックを含めたブロックの合計カウント数が表示されます。

以下の例をご確認ください。

- 現行ディレクトリーにあるすべてのファイルのリストを表示するには、次のように入力します。

  ```
  ls -a
  ```

  これにより、以下のものを含む、すべてのファイルがリストされます。

  – ドット (.)
  – ドット・ドット (..)
  – その他のファイル (名前がドット (.) で始まるものも、そうでないものもある)
- 詳細な情報を表示するには、次のように入力します。

  ```
  ls -l chap1 .profile
  ```

  これにより、chap1 と .profile の詳細な情報が追加された、長いリストが表示されます。
- ディレクトリーの詳細な情報を表示するには、次のように入力します。

  ```
  ls -d -l . manual manual/chap1
  ```

  これにより、. と manual ディレクトリーと、 manual/chap1 ファイルの、長いリストが表示されます。 **-d** フラグを指定しない場合は、ディレクトリー自体に関する詳細情報ではなく、. および manual ディレクトリー内のファイルがリストされます。

完全な構文については、「コマンド・リファレンス 第 *3* 巻」の **ls** コマンドを参照してください。

ディレクトリーを削除または除去する (rmdir コマンド)

Directory パラメーターによって指定されたディレクトリーを、システムから 除去するには、**rmdir** コマンドを使用します。

ディレクトリーは、除去する前に空にして置く必要があり (含めることができるのは . と .. のみ)、かつその親ディレクトリーの書き込み許可が必要です。 **ls -a***Directory* コマンドを使って、ディレクトリーが空であるかどうかを検査してください。

rmdir コマンドの使用例を以下に示します。

- ディレクトリーを空にし、除去するには、次のように入力します。

```
rm mydir/* mydir/.*
rmdir mydir
```

  これにより、mydir の内容が除去され、次に空になったこのディレクトリーが除去されます。 **rm** コマンドは、ドット (.) ディレクトリーと、ドット・ドット (..) ディレクトリーの除去についてのエラー・メッセージを表示します。その後、**rmdir** コマンドが . と .. ディレクトリー、およびそのディレクトリー自体を除去します。

  注: rm mydir/* mydir/.* は、 まずドット以外で始まる名前のファイルを除去し、次にドットで始まる名前のファイルを除去することに注意してください。 **ls** コマンドで **-a** フラグを使用しないと、 ドットで始まるファイル名のリストは表示されません。

- /tmp/jones/demo/mydir ディレクトリー、およびその下のすべてのディレクトリーを除去するには、次のように入力します。

```
cd /tmp
rmdir -p jones/demo/mydir
```

  これにより、/tmp ディレクトリーから jones/demo/mydir ディレクトリーが除去されます。 ディレクトリー を除去するときに、ディレクトリーが空でなかった場合、またはこのディレクトリーに対する書き込 み許可がなかった場合には、コマンドは終了し、適切なエラー・メッセージが出されます。

完全な構文については、「コマンド・リファレンス 第 *4* 巻」の **rmdir** コマンドを参照してください。

ディレクトリーの内容を比較する (dircmp コマンド)

Directory1 と *Directory2* のパラメーターで指定された 2 つのディレクトリーを比較し、これらの内容についての情報を 標準出力に書き込むには、**dircmp** コマンドを使用します。

dircmp コマンドは、まず各ディレクトリーのファイル名を比較します。 両方に同じファイル名が含まれて いた場合、**dircmp** コマンドは、両方のファイルの内容を比較します。

dircmp コマンドは、出力で、各ディレクトリーに固有のファイルを リストします。 次に、両方のディレクトリー内の名前が同じで内容が異なるファイルをリストします。 また、フラグが指定されなかった 場合には、両方のディレクトリー内の内容と名前がまったく同一のファイルもリストします。

dircmp コマンドの使用例を以下に示します。

- proj.ver1 ディレクトリーと proj.ver2 ディレクトリーにあるファイルの相違点の要約を表示するには、次のように入力します。

```
dircmp proj.ver1 proj.ver2
```

  これに より、proj.ver1 ディレクトリーと proj.ver2 ディレクトリーの相違点の要約が表示されます。 要約では、片方のディレクトリーまたはもう一方のディレクトリーにのみ存在するファイルと、両方のディレクトリーに存在するファイルとが、別個にリストされます。 両方の ディレクトリーに存在するファイルがあった場合、**dircmp** コマンドは、この 2 つのコピーが同一であるかどうかを表示します。

- proj.ver1 ディレクトリーと proj.ver2 ディレクトリーにあるファイルの相違点の詳細を表示するには、次のように入力します。

```
dircmp -d -s proj.ver1 proj.ver2
```

-s フラグ では、同一のファイルについての情報を抑止します。 **-d** フラグでは、両方のディレクトリーに存在する異なる各ファイルを、**diff** リストに表示します。

完全な構文については、「コマンド・リファレンス 第 *2* 巻」の **dircmp** コマンドを参照してください。

ファイルシステムとディレクトリーに関するコマンドのまとめ
以下に、ファイルシステムおよびディレクトリー 用のコマンド、ディレクトリー処理プロシージャー用のコマンド、 ならびにディレクトリー省略語のリストを示します。

表 *5.* ファイルシステムに関するコマンドのまとめ

項目	説明
df	ファイルシステム 上のスペースに関する情報を報告します

表 *6.* ディレクトリーの省略形

項目	説明
.	現行作業ディレクトリー
..	現行作業ディレクトリーの 1 つ上のディレクトリー (親ディレクトリー)
~	ユーザーのホーム・ディレクトリー。 (Bourne シェルの場合は該当しません。 詳しくは、『Bourne シェル』を参照してください。)
$HOME	ユーザーのホーム・ディレクトリー。 (すべてのシェルの場合。)

表 *7.* ディレクトリー処理プロシージャー用のコマンド要約

項目	説明
cd	現行ディレクトリーを変更します
cp	ファイルまたはディレクトリーをコピーします
dircmp	2 つのディレクトリーとその中にあるファイルの内容を比較します
ls	ディレクトリーの内容を表示します
mkdir	1 つ以上の新規ディレクトリーを作成します
mvdir	ディレクトリーを移動します (ディレクトリーの名前を変更します)
pwd	作業ディレクトリーのパス名を表示します
rmdir	ディレクトリーを除去します

ワークロード・マネージャー

ワークロード・マネージャー (WLM) は、スケジューラー仮想メモリー・マネージャー (VMM)、およびディスク入出力サブシステムがリソースをプロセスに 割り当てる方法に対するシステム管理者の制御が増加する設計になっています。

WLM を使用すれば、さまざまなクラスのジョブが相互に妨害し合うことを防止し、さまざまなユーザー・グループの要件に応じてリソースを割り当てることが可能になります。

重要 : WLM を有効に使用するには、既存システムのプロセスおよびパフォーマンスについての 幅広い知識が必要です。 システム管理者が極端な値または不正確な値 で WLM を構成すると、パフォーマンスが著しく低下します。

WLM は、主に、ラージ・システムでの使用を目的としています。 ラージ・システムは、サーバー統合によく使用されます。 このサーバー統合では、数多くの異なるサーバー・システム (プリンター、データベース、一般ユーザー、トランザクション処理システムなど) からのワークロードが単一のラージ・システムに結合され、システムの維持管理のコストが削減されます。 これらのワークロードは、相互に妨害し合う場合が多くあり、しかも、 それぞれの目標とサービスの契約が異なります。

また、WLM では、非常に異なるシステム動作をもつユーザー群を分離します。 これによって、特定の動作 (例えば、対話式ジョブまたは低 CPU 使用ジョブ) をするワークロードのリソースが、その他の動作 (例えば、バッチ・ジョブまたは高メモリー使用ジョブ) をするワークロードによって使われて足りなくなることを防ぐことができます。

さらに、WLM がアカウンティング・サブシステムに接続すると、ユーザーは、ユーザーまたはグループごとの標準アカウンティングに加えて、WLM クラスごとにリソース使用状況のアカウンティングを行うこともできます。

ワークロード・マネージメントの概念

WLM を使用すれば、ジョブに対してさまざまなサービス・クラスを作成し、 これらのクラスに属性を指定することができるようになります。

これらの属性では、クラスに割り当てられる CPU、物理メモリー、 およびディスク入出力スループットの最小と最大の量を指定します。 WLM は次に、システム管理者が提供するクラス割り当て規則を使用して、ジョブを自動的にクラスに割り当てます。 その割り当て規則は、一組のプロセス属性の値に基づいたものです。 システム管理者または特権ユーザーは、自動割り当てを指定変更して手動でジョブをクラスに割り当てることもできます。

ワークロード・マネージメントの用語

ワークロード・マネージメントに関連する共通用語は、この表にリストおよび説明を記載します。

項目	説明
クラス	クラス とは、プロセスおよびそれに関連したスレッドの集合です。 クラスには、一連のリソース制限値とターゲット Share があります。 クラス は、サブクラスとスーパークラスの 両方を指します。
スーパークラス	スーパークラス とは、サブクラスが関連付けられたクラスのことです。 プロセスは、同時にサブクラスにも属さない限り、スーパークラスに属することはできません。 スーパークラスには、どのプロセスをスーパークラスに割り当てるかを決める一連のクラス割り当て規則が備えられています。 またスーパークラスには、スーパークラスに属するプロセスが使用できるリソース量を決める一連のリソース制限値とリソース・ターゲット Share も備えられています。 これらのリソースは、サブクラスのリソース制限値とリソース・ターゲット Share に基づいて、サブクラス間で分割されます。
サブクラス	サブクラス は、ただ 1 つのスーパークラスに関連付けられたクラスです。 サブクラス内のどのプロセスも、やはりそのスーパークラスのメンバーになります。 サブクラスは、スーパークラスが使用できるリソースにしかアクセスすることはできません。 サブクラスには、スーパークラスに割り当てられたどのプロセスが自身に属するかを判断するための一連のクラス割り当て規則が備えられています。 またサブクラスには、そのサブクラス内のプロセスが使用できるリソースを判断するための一連のリソース制限値とリソース・ターゲット Share も備えられています。 この一連のリソース制限値とリソース・ターゲット Share は、スーパークラスが使用できるリソース (スーパークラスのターゲット) のうちのどれを、そのサブクラス内のプロセスが使用できるかを示します。 WLM の管理は、SMIT、または WLM コマンド・ライン・インターフェースを使って行うことができます。
分類メカニズム	分類メカニズム は、どのプロセスをどのクラス (スーパークラスまたはスーパークラス内のサブクラス) に割り当てるかを決める一連のクラス割り当て規則です。
クラス割り当て規則	クラス割り当て規則 は、特定クラス (スーパークラスまたはスーパークラス内のサブクラス) への割り当てを決める一連のプロセス属性内の値を示します。

項目	説明
プロセス属性値	プロセス属性値 とは、プロセス属性用にプロセスに備えられた値のことです。 プロセス属性には、ユーザー ID、グループ ID、およびアプリケーション・パス名などの属性が含まれます。
リソース制限値	リソース制限値 とは、一連のリソースの使用率値用に WLM で維持される一連の値のことです。 この制限は、**setrlimit** サブルーチンで指定されるリソース制限とはまったく別個のものです。
リソース・ターゲット Share	リソース・ターゲット *Share* とは、あるクラス (サブクラスまたはスーパークラス) が使用できるリソースの Share のことです。 この Share を、同じレベルと Tier 内の他のクラス (サブクラスまたはスーパークラス) Share と一緒に使って、そのレベルと Tier の複数のクラスに対するリソースの任意の分配が決定されます。
リソースの使用率の値	リソースの使用率の値 とは、システム内で 1 つまたは一連のプロセスが現在使用しているリソース量のことです。 1 つまたは一連のどちらのプロセスであるかは、プロセス・リソース・コレクションの有効範囲で判別されます。
リソース・コレクションの有効範囲	リソース・コレクションの有効範囲 とは、 リソースの使用率が収集され、しかもリソース制限値が適用されるレベルのことです。 それは、クラス内の各プロセスのレベル、 各ユーザーが所有するクラス内の各プロセスを総合したレベル、 またはクラス内の各プロセスを総合したレベルのいずれかになります。 現在サポートされている唯一の有効範囲は後者です。
プロセスのクラス属性	プロセスのクラス属性 とは、割り当てられた先のクラス (サブクラスおよびスーパークラス) に基づいてプロセスに与えられる一連の属性のことです。
クラス権限	クラス権限 とは、クラスに対する操作またはクラス内のプロセスとスレッドの操作をどのユーザーおよびグループが実行できるかを指示する一連の規則です。 これには、クラスに対するプロセスの手動割り当てや、スーパークラスのサブクラスの作成のための権限も含まれます。
クラス Tier	クラス *Tier* の値とは、すべてのクラスについて望ましいリソース制限階層内での該当クラスの位置のことです。 Tier 内のすべてのクラスのリソース制限 (リソース・ターゲットを含む) が満たされてはじめて、下位の Tier クラスにリソースが提供されます。 スーパークラスとサブクラスの 両方のレベルで Tier が設けられています。 スーパークラスには、その Tier に基づいてリソースが提供されます。 スーパークラス内では、そのクラスでの Tier 値に基づいてリソースがサブクラスに与えられます。 つまり、スーパークラスの Tier が、リソース分配における主要な決定要因となり、サブクラスの Tier は、スーパークラス内のさらに細分化された分配の決定要因となります。

ワークロード・マネージメントのクラス

システム管理者は WLM を使って、クラスを定義してから、それぞれのクラスごとに、一連の属性とリソース制限を定義することができます。

プロセスは システム管理者が指定した基準に基づいてクラスに割り当てられます。 リソースに対する権利と制限は、クラス・レベルで規定されます。 このようにして、サービス・クラスが定義され、アプリケーションの各クラスのリソース使用率が規制されるので、それぞれかなり異なったリソース使用パターンを有する複数のアプリケーションが 1 つのサーバーを共用しても、互いに妨害しあうことはありません。

WLM では、次の 2 つのレベルに分けてクラス階層がサポートされます。

- システムのリソースは、各スーパークラスのリソース 権に応じてそれぞれのスーパークラスに 分配されます。 システム管理者がリソース権を定義します。
- 次にスーパークラスは、サブクラスをもつことができます。 各サブクラスに与えられたリソース権に応じて、スーパークラスに割り当てられたリソースが各サブクラスに分配されます。

- システム管理者は、各スーパークラスのサブクラスの 管理を、スーパークラス 管理者、またはスーパークラス管理者グループに委任することができます。
- WLM は、最大 69 個のスーパークラス (ユーザー定義クラスは 64 個) と 1 スーパークラスあたり 64 個のサブクラス (ユーザー定義クラスは 61 個) をサポートします。
- 企業での要件に合わせてシステム管理者は、スーパークラスのみを 使っても、またはスーパークラスとサブクラスを使ってもかまいません。

注 : WLM についてのここでの記述を通じて、クラス という用語はスーパークラスとサブクラスの 両方に適用されます。 特定タイプのクラスのみについて記述する場合は、 そのタイプが明示的に示されます。

ワークロード・マネージメントのクラスへのプロセスの割り当て
プロセスは、システム管理者が指定したクラス割り当て規則を使用してクラスに割り当てられます。 分類基準のベースになるのは、ユーザー ID、グループ ID、アプリケーション・ファイルの名前、プロセスのタイプ、およびアプリケーション・タグなどの一連のプロセス属性の値です。

プロセスを割り当てるスーパークラスの決定には、一連の定義済み規則を使用します。 当該スーパークラスにサブクラスが定義されている場合は、 どのサブクラスをどのプロセスに割り当てるか決定するにあたり、 当該スーパークラスに使用される一連の規則が別にあります。 このような自動割り当てプロセスでは、スーパークラスとサブクラスの 両方の `inheritance` 属性も考慮に入れられます。

自動クラス割り当てが行われるのは、プロセスで **exec** サブルーチンを呼び出した場合です。 プロセスで、分類のために使用されるプロセス属性を変更できるサブルーチンを使用すると、クラス割り当ては再評価されます。 その例には、**setuid**、**setgid**、**setpri**、および **plock** サブルーチンがあります。

該当する権限をもつユーザーは、この自動クラス割り当てに加えて、個々のスーパークラスまたはサブクラスにプロセスまたはプロセス・グループを手動で割り当てることもできます。

関連概念
クラス属性
WLM クラスのすべての属性をリストする。

リソース制御
WLM では、リソースを 2 つの方法 (つまり、使用可能なリソースのパーセントか、または合計リソース使用率) で管理できます。

パーセントで制御できるリソースには、次のものがあります。

- クラス内のタイプ SCHED_OTHER のスレッドによるプロセッサー使用状況。 これは、クラス内の各スレッドが使用するすべてのプロセッサー・サイクルの合計です。 固定優先順位スレッドは調整できません。 したがって、変更もできず、 プロセッサー使用率ターゲットを超えることができます。
- クラス内のプロセスの物理メモリー使用状況。 これは、クラス内のプロセスに属するすべてのメモリー・ページの合計になります。
- クラスのディスク入出力帯域幅。 これは、クラスからアクセスされる各ディスク・デバイス上のクラス内のスレッドが実行するすべての入出力の帯域幅 (512 バイト・ブロック/秒の単位)。

合計使用率を基にして制御できるリソースは、2 つのカテゴリー (クラス合計またはプロセス合計) のどちらかに当てはまります。 クラス合計カテゴリーに当てはまるものとしては、次のものがあります。

クラス内のプロセス数
これは、1 つのクラス内で同時にアクティブになっているプロセスの数です。

クラス内のスレッド数
これは、1 つのクラス内で同時にアクティブになっているスレッドの数です。

クラス内のログイン数
これは、1 つのクラス内で同時にアクティブになっているログイン・セッションの数です。

プロセス合計カテゴリーに当てはまるものとしては、次のものがあります。

合計 CPU 時間
これは、単一プロセスについての合計累積 CPU 時間です。

合計ディスク入出力
これは、単一プロセスについてのディスク入出力の合計累積ブロックです。

合計接続時間
1 つのログイン・セッションがアクティブでいられる時間の合計です。

リソース指定
WLM ではシステム管理者は、それぞれのリソース・タイプごとに独自のクラス別リソース権を指定することができます。

以下のように指示すれば、そのような権利を指定することができます。

- さまざまなタイプのリソースを使用するためのターゲット。 そのターゲットは、Share を使って指定します。 Share は、異なるクラスどうしの相対使用量で指定します。 例えば、2 つのクラスがそれぞれ CPU の 1 および 3 の Share を持っていて、しかもその時点でこれらのクラスだけがアクティブである場合、その CPU 調整のために WLM で使用されるパーセント目標は、それぞれ 25% および 75% になります。 各 Tier 内のクラスのターゲット・パーセントは、その Tier 内のアクティブ Share の数およびその Tier で使用可能なリソースの量に基づいて計算されます。
- 上下限値。 これらの値は、使用可能なリソースの合計のパーセントで指定します。 WLM では、次の 2 種類の上限値がサポートされます。
  - ソフト最大値。 これは、リソースの競合が起きた場合に使用可能にできるリソースの最大量を示します。 競合が起きていない場合、つまり他に誰もリソースを必要としていなければ、 この最大値を超えてもかまいません。
  - ハード最大値。これは、リソースに競合が起きるかどうかには関係なく 使用可能にできるリソースの最大量を示します。 ただし、固定優先順位スレッドの場合は、 これと同じ規則には制約されず、したがって制限を超えることができます。
- 制限合計。 制限合計は厳密に適用されます。 あるプロセスがその使用量の制限合計を超えた場合、そのプロセスは終了させられます。 あるクラスがそのクラスの制限合計のいずれかに達している場合、 そのリソースの別のインスタンスが結果として作成される操作はすべて失敗します。

ほとんどの場合、ソフト最大値はリソース権が満たされ、適用するに十分な値です。 ハード最大値を使用すると、結果的にシステム・リソースが使用されないことになる可能性があります。 なぜなら、これは当該リソースに競合が発生していない場合であっても厳密に適用されるからです。 ハード最大値を低すぎる値に設定すると、システムまたはアプリケーションのパフォーマンスに 多大な影響を与える可能性があるので、これを使用する場合は注意深い配慮が必要です。 制限合計も注意深く使用する必要があります。なぜなら、 プロセスが終了したり、意図したように働かなくなる場合があるからです。

アクティブ・モードでは、WLM はアクティブ・クラスをそれぞれのターゲットに近い状態に保とうとします。 各種の制限の値に関する制約はほとんどないので、 すべてのクラスに適用されるどの制限についても、合計が 100% をはるかに超えても構いません。 このケースでは、すべてのクラスがアクティブになっている場合、 すべてのクラスで制限に達することはありません。 WLM は、システム内のスレッドが属するクラスの制限とターゲットを基準にして、 そのクラスの実行状況に応じて非固定優先順位スレッドのスケジューリング優先順位を調整することにより、 プロセッサー使用量を規制します。 このアプローチにより、非常に短い間隔 (例えば、10 ms) での プロセッサー使用量ではなく、ある期間での平均プロセッサー使用量が保証されます。

例えば、クラス A が唯一のアクティブ・クラスで、そのプロセッサー最小が 0%、 プロセッサー・ターゲットが 60 Share であるとすると、このクラスはプロセッサーを 100% 獲得できます。 プロセッサー最小制限値が 0% で、プロセッサー・ターゲットが 40 Share のクラス B がアクティブになった場合、クラス A のプロセッサー使用率は 60% まで漸次減少し、クラス B のプロセッサー使用率は 0 から 40% に増加します。 数秒のうちに、システムはプロセッサー使用率がそれぞれ 60% と 40% の状態で安定します。

この例では、クラス間にメモリー競合がないと仮定しています。 通常の作動条件下では、プロセッサーおよびメモリーに設定された制限は相互に依存します。 例えば、あるクラスのメモリー使用率の最大限度がそのクラスの実効ページ・セットと比較して低過ぎる場合は、 そのクラスはターゲットのプロセッサー割り当てまたは最小のプロセッサー割り当てに すら達しないことがあります。

ある特定のアプリケーションのセットに関するクラス定義とクラス制限を細かく調整するのを援助するために、WLM には現在各クラスが使用中のリソースの量を表示する **wlmstat** レポート作成ツールが用意されています。 システムのモニター用として、グラフィカルな表示ツールである **wlmmon** も用意されています。

ワークロード・マネージャーの仮想メモリー制限

ワークロード・マネージャー (WLM) 仮想メモリー制限により、管理者は、クラスまたはプロセスに仮想メモリー制限を施すことによって、過剰ページングによるシステムの 性能低下またはシステム障害を予防できます。

制限を超過すると、WLM は、以下のいずれかを行ってアクションを取ります。

- 制限を超過した WLM クラス下のすべてのプロセスを強制終了する
- WLM クラスの使用量が制限を超過する原因となったプロセスのみを強制終了する
- プロセス制限を超過したプロセスを強制終了する

仮想メモリー制限は、すべてのユーザー定義のクラス、ユーザー定義スーパークラスの 下のすべてのデフォルト・サブクラス、およびデフォルトのスーパークラスに 指定できます。

アカウンティングの 目的からは、WLM の合計クラスまたはプロセス使用量を判別する際に WLM が仮想メモリーとして考慮するのは、以下に限られます。

- ヒープ
- ローダー初期化データ、BSS、共用ライブラリー、および専用ロード済みセグメント
- UBLOCK および mmap 領域
- 大きく、固定されたユーザー・スペース・ページ

管理者は、クラスまたはクラス内の各プロセスに WLM 仮想メモリー制限を指定できます。 クラス制限を超過すると、WLM は、**vmenforce** クラス属性が、それぞれ `class` に設定されているか、あるいは `proc` に設定されているかによって、そのクラスに割り当てられたすべてのプロセスを強制終了するか、制限超過の原因となったプロセスを強制終了します。 デフォルトの動作は、制限超過の原因となったプロセスのみの強制終了です。 プロセスの仮想メモリーの使用量が制限を上回ると、プロセス制限は終了します。

ワークロード・マネージャーの操作のモード

WLM は、クラスあたりのパーセント、クラスあたりの合計、またはプロセスあたりの合計としてリソース消費を調整するのに使用できます。 リソース・タイプの規制は、すべて WLM をアクティブ・モードで実行して使用可能にできます。

オプションで、新規および既存のプロセスをクラス分けし、各種クラスのリソース使用量を、この使用量の調整を試みずにモニターする、WLM のモードを開始できます。 このモードはパッシブ・モード と呼ばれます。

パッシブ・モードは、新しいシステムで WLM を構成して、 分類と割り当ての規則を検証し、WLM がプロセッサーおよびメモリーの割り当てを規制しない 場合における各種クラスのリソース使用率のベースラインを確立するのに使用できます。 これは、ビジネス上の要求に合わせるために、リソース Share とリソース制限 (必要な場合) を どのように適用して、重要なアプリケーションは優先的に、重要度が低い作業は制限をするかという 基準をシステム管理者に提供します。

規制の対象にするリソースがプロセッサー時間だけである場合は、WLM を プロセッサーについてはアクティブ・モードで実行し、 プロセッサー以外のリソースについてはパッシブ・モードで実行するようにできます。 このモードは *CPU* のみ モードと呼ばれます。 クラス別パーセントは規制するが、合計リソース・タイプの方はどちらも規制しない場合は、クラス別合計またはプロセス別合計 (あるいはその両方) については合計リソースのアカウンティングと 規定を使用不可にできます。 すべてのモードには、リソース・セット・バインディングを使用不可にするオプションが用意されています。

ワークロード・マネージャーの動的制御

WLM がアクティブになった後、現在の構成のどのパラメーターでもいつでも変更することができます。例えば、クラスの属性、リソースの Share および制限、割り当て規則の変更と、新規クラスの追加、または既存のクラスの削除などを行うことができます。

それには、以下の方法のいずれかを行います。

- 現在アクティブな 構成 (シンボリック・リンク `/etc/wlm/current`) によって指し示されているディレクトリー) の属性ファイルを変更し、**`wlmcntrl`** コマンドを使用して新規パラメーターを使用する。
- 別の一連のパラメーターを使って別の構成を作成してから WLM を更新して、新しい構成のパラメーターをロードし、それを現行構成にする。

- WLM コマンド・ライン・インターフェース (**mkclass**、**chclass**、および **rmclass** コマンド) を使って、現在アクティブな 構成のパラメーターの一部を変更する。
- アプリケーションから WLM API を使用して、現在アクティブになっている 構成のパラメーターの一部を変更する。

構成セット を使用して、 1 日のうちの指定した時刻に自動的に新しい構成に切り替わるようにすることができます。 構成セットを使用すれば、管理者は使用する一連の構成とそれぞれの構成がアクティブになる時間範囲を指定できます。

モニター・ツール

WLM 統計情報を表示し、WLM 操作をモニターするには、WLM コマンドを使用します。

- **wlmstat** コマンド。これはテキスト指向のコマンドであり、統計情報をテキスト (WLM で管理されるすべてのリソース・タイプのクラス別リソース使用率のパーセント) で表示します。
- **wlmmon** コマンド。クラス別のリソース使用率と WLM 規定をグラフィックで表示します。
- **wlmperf** コマンド。 これは、Performance Toolbox と一緒に使うオプショナルのツールであり 、 長時間の記録や再生などの別の機能を備えています。

ワークロード・マネージャーにおける後方互換性

wlmstat コマンドのデフォルト出力はスーパークラスのみをリストしますが、これは旧バージョンのものに似ています。

例:

```
# wlmstat
                CLASS  CPU  MEM  DKIO
        Unclassified    0    0   0
           Unmanaged    0    0   0
             Default    0    0   0
              Shared    0    2   0
              System    2   12   0
              class1    0    0   0
              class2    0    0   0
#
```

一部のスーパークラスに 対して WLM 管理者がサブクラスを定義した場合は、そのサブクラスがリストされます。 例:

```
# wlmstat
                CLASS  CPU  MEM  DKIO
        Unclassified    0    0   0
           Unmanaged    0    0   0
             Default    0    0   0
              Shared    0    2   0
              System    3   11   7
              class1   46    0   0
    class1.Default     28    0   0
     class1.Shared      0    0   0
        class1.sub1    18    0   0
              class2   48    0   0
#
```

この出力は、**ps** コマンドを使用した場合と同じです。 サブクラスを持たないスーパークラス内のプロセスの場合、**ps** コマンドは、そのプロセスのクラス名としてスーパークラス 名を示します。

クラス別のアカウンティング

AIX アカウンティング・システム・ユーティリティーを 使用すれば、 さまざまなシステム・リソースの使用をユーザー、グループ、または WLM クラスごとに収集し、報告することができます。

プロセスのアカウンティングがオンの場合は、そのプロセスの終了時に、 オペレーティング・システムはプロセス・リソース使用率に関する統計情報をアカウント・ファイルに記録します。 このアカウンティング・レコードには、プロセスが属する WLM クラスの名前を表す 64 ビットの数字キーが含まれます。

アカウンティング・サブシステムは、スペースを節約するために、全 34 文字のクラス名の代わりに 64 ビット・キーを使用します (さもなければ、その変更により、アカウンティング・レコードのサイズは事実上 2 倍になります)。 アカウンティング・コマンドを実行してプロセス当たりのデータを抽出すると、このキ

ーは、上記のルーチンを使用して再度クラス名に変換されます。 この変換には、現在 WLM 構成ファイル内にあるクラス名が使用されます。 したがって、プロセスの終了時に、アカウンティング・レコードが書き込まれた時刻と、アカウンティング・レポートが実行される時刻の間にクラスが削除された場合は、このキーに対応するクラス名は検出できず、このクラスは「Unknown」として表示されます。

アカウンティング期間に削除されたクラスのリソース使用率の正確なレコードをとるために、 以下のいずれかを実施してください。

- クラスを削除する代わりに、クラス名をクラス・ファイル内に保持し、 ルール・ファイルからクラスを削除して、どのプロセスもそのクラスに割り当てることができないようにする。 その後、アカウンティング期間の終わりにアカウンティング・レポートが生成されれば、 このクラスを削除することができます。
- あるいは、クラスが属する構成からクラスを削除し、 その期間のアカウント・レコードが生成されるまで、クラス名を「ダミー」構成 (アクティブにされることのない構成) 内のクラス・ファイルに保持する。

ワークロード・マネージャー を管理する

システム管理者が ワークロード・マネージャー (WLM) を使用すると、スケジューラーと仮想メモリー・マネージャー (VMM) がリソースをプロセスに割り当てる方法 をより細かく制御できるようになります。 WLM を使用すると、いろいろなクラスのジョブが互いに干渉しあうのを防ぎ、それぞれのユーザー・グループの要件 に基づいてリソースを割り当てることができます。

WLM を使用すれば、 ジョブのさまざまなサービス・クラスを作成すること、 および、それらのクラスに属性を指定することができます。 これらの属性は、クラスに割り当てられる CPU、物理メモリー、およびディスク入出力スループットの最小と最大の量を指定します。 WLM は次に、システム管理者が提供するクラス割り当て規則を使用して、 ジョブを自動的にクラスに割り当てます。 その割り当て規則は、一組のプロセス属性の値に基づいたものです。 システム管理者または特権ユーザーは、自動割り当てを指定変更して手動でジョブをクラスに割り当てることもできます。

WLM は、基本オペレーティング・システムの一部であり、基本オペレーティング・システムと一緒にインストールされますが、オプションのサービスです。 ワークロード・マネージャーは、ご使用のシステム環境に合わせて構成し、使用したいときに開始し、WLM サービスを中止または終了したいときに停止する必要があります。

このセクションでは、ご使用するサイトに適したクラスおよびルールで、WLM を構成するための手順について説明します。また、予期しないリソース消費動作のトラブルシューティングを 行うための方法についても説明します。

重要 : このセクションのタスクでは、お客様が WLM の概念を熟知していることを前提としています。 WLM を有効に使用するには、既存システムのプロセスおよびパフォーマンスについての 幅広い知識が必要です。 システム管理者が極端な値または不正確な値 で WLM を構成すると、パフォーマンスが著しく低下します。

ワークロード・マネージャーの開始とシャットダウン

WLM はオプションのサービスであり、開始および停止する必要があります。

WLM を開始または停止するには、システム管理インターフェースの SMIT の どちらかを使用することをお勧めします。

- SMIT を使用して WLM を開始または停止するには、`smit wlmmanage` 高速パスを使用します。

上記のオプションの主要な相違は、パフォーマンスに現れます。 SMIT では、3 通りの方法で WLM を開始または停止できます。

current session
このオプションを指定して WLM の停止を要求すると、WLM はこのセッションだけで 停止し、次のリブートでは再始動します。 このオプションを指定して始動を要求すると、WLM はこのセッションだけで始動し、次のリブートでは再始動しません。

next reboot
このオプションを指定して WLM の停止を要求すると、WLM はこのセッションでは 実行され続けます。しかし、次のリブートでは再始動しません。 このオプションを指定して始動を要求すると、WLM はこのセッションでは使用できませんが、次のリブートで再始動します。

both
このオプションを指定して WLM の停止を要求すると、WLM はこのセッションだけで 停止されます。 また、次のリブートでも再始動しません。 このオプションを指定して始動を要求すると、WLM はこのセッションだけで始動します。また、次のリブート で再始動します。

wlmcntrl コマンドを使用することもできますが、**wlmcntrl** コマンドで WLM を開始または停止できるのは、現行セッションでのみです。 コマンド・ライン・インターフェースを 使用し、マシンをリブートしたときに変更を有効にしておきたい場合は、/etc/inittab ファイルを編集する必要があります。

WLM は、クラスあたりのパーセント、クラスあたりの合計、またはプロセスあたりの合計としてリソース消費を調整するのに使用できます。 WLM をアクティブ・モードで実行することによって、すべてのリソース・タイプを調整することができます。 オプションで、新規および既存のプロセスをクラス分けし、各種クラスのリソース使用量を、この使用量の調整を試みずにモニターする、WLM のモードを開始できます。 このモードは、パッシブ・モードと呼ばれます。 調整したいリソースが CPU 時間だけの場合は、CPU 用に WLM をアクティブ・モードで実行し、その他のすべてのリソースについてパッシブ・モードで実行することができます。 このモードは *CPU* のみ モードと呼ばれます。

WLM が始動される前にシステムに存在していたすべてのプロセスは、新しく読み込まれた割り当て規則に従ってクラス分けされ、WLM によってモニターされます。

ワークロード・マネージャーの属性
SMIT、WLM コマンド・ライン・インターフェースを 使用するか、フラット ASCII ファイルを作成することにより、WLM 構成の属性を指定することができます。 SMIT インターフェースでは、WLM コマンドを使用して、属性ファイル と呼ばれる、同じフラット ASCII ファイルに 情報を記録します。

一組の WLM 属性ファイルによって WLM 構成を定義します。 さまざまな構成のワークロード・マネージメントを定義することによって、複数の属性ファイルセットを作成することできます。 これらの構成は **/etc/wlm** のサブディレクトリーに入っています。 *Config* 構成のスーパークラスを 記述した WLM の属性ファイルは、/etc/wlm/Config 内のファイルの *classes*、*description*、*limits*、*shares* および *rules* です。 したがって、この構成のスーパークラス *Super* のサブクラスを記述した属性ファイルは、/etc/wlm/Config/Super ディレクトリー内のファイルの *classes*、*limits*、*shares* および *rules* です。 root ユーザーだけが WLM を始動または停止でき、構成を別の構成に切り替えることができます。

属性ファイルの名前は以下のとおりです。

項目	説明
classes	クラス定義
description	構成説明テキスト
groupings	属性値グループ
limits	クラス制限
shares	クラス・ターゲット share
rules	クラス割り当て規則

WLM プロパティー・ファイルをサブミットするコマンド **wlmcntrl** と、その他の WLM コマンドによって、ユーザーは、WLM プロパティー・ファイルの代替ディレクトリー名を指定できます。 これにより、デフォルト WLM 属性ファイルを変更しなくても、ユーザーは WLM 属性を変更できます。

シンボリック・リンクの /etc/wlm/current は、現行構成ファイルのあるディレクトリーを指しています。 指定した構成または構成セットで WLM を始動させるときに、**wlmcntrl** コマンドを使用してこのリンクを更新してください。 オペレーティング・システムと一緒に出荷されているサンプル構成ファイルは、 /etc/wlm/standard にあります。

属性値グループの作成

属性値をグループ化し、rules ファイルの単一値でそれらの値を 表すことができます。 これらの属性値グループ は、WLM 構成ディレクトリー内の groupings ファイルで定義されます。

デフォルトでは、構成には groupings ファイルはありません。 このファイルを作成するためのコマンド・インターフェースまたは 管理インターフェース はありません。 属性値グループを生成し使用するには、以下の手順を実行します。

1. root 権限で、次の例に示すように、適切な構成ディレクトリーに変更します。

```
cd /etc/wlm/MyConfig
```

2. 任意のエディターを使用して、groupings という名前のファイル を編集します。
   例:

```
vi groupings
```

3. 次の形式で、属性およびそれらの関連値を定義します。

   attribute = *value*, *value*, ...

   すべての値をコンマで 区切る必要があります。 スペースは無意味です。 範囲およびワイルドカードを使用できます。 例:

```
trusted = user[0-9][0-9], admin*
nottrusted = user23, user45
shell = /bin/?sh, ¥
        /bin/sh, ¥
        /bin/tcsh
rootgroup=system,bin,sys,security,cron,audit
```

4. ファイルを保管します。
5. クラスの選択基準内で属性グループを使用するには、rules ファイル を編集します。

   属性グループ名の前には、対応する値を組み込むためにドル記号 ($) を付けるか、値を除外するために感嘆符 (!) を付ける必要があります。 感嘆符は、グループのメンバー (ステップ 137 ページの『3』) では使用できません。また、この rules ファイルのグループの 前で使用できる唯一の修飾子です。 次の例では、アスタリスク (*) はコメント行を示しています。

```
*class  resvd  user                  group       application       type  tag
classA  -      $trusted,!$nottrusted  -           -                 -     -
classB  -      -                     -           $shell,!/bin/zsh  -     -
classC  -      -                     $rootgroup  -                 -     -
```

6. ファイルを保管します。

この時点で、分類ルールに属性値グループが組み込まれます。 ルールの構文解析を行う場合、 エレメントが $ で始まるときは、groupings ファイル内でそのエレメントが検索されます。 エレメントの構文が無効か、groupings ファイルが存在しない場合は、警告メッセージが表示され、その他のルールの処理が続行されます。

時間基準の構成セットの作成

専門の構成セットを作成し、指定の構成を有効にしたい場合に、セット内の各構成を日および時刻に割り当てることができます。

時刻に基づいた 構成セット と呼ばれるこれらのセットは、標準構成とはまったく 異なりますが、互換性はあります。 **wlmcntrl -u** コマンドを使用すると、必要に応じて構成セットと標準構成を切り替えることができます。

構成セットを使用する場合は、指定した既存の構成を、通常、特定の時間範囲に関連付けます。 特定の時刻には 1 つの構成しか使用できないため、指定する各時間範囲は固有のものでなければなりません。時間範囲を、オーバーラップまたは 重複させることはできません。

指定した構成が時間範囲外になったため、別の構成を使用する必要がある場合は、**wlmd** デーモンが WLM に警告を出します。 root ユーザーだけが これらの時間範囲を管理できます。これらの範囲は、.times と呼ばれる ASCII ファイルの構成セット・ディレクトリー内で指定されます。

時刻に基づく構成セットを作成するには、以下の手順を実行します。

1. root 権限で、構成セット・ディレクトリーを作成し、次にそのディレクトリーに変更します。
   例:

   ```
   mkdir /etc/wlm/MyConfigSet
   cd /etc/wlm/MyConfigSet
   ```

2. 任意のエディターを使用して、構成セットの **.times** ファイル を作成し、次の形式で構成および時間範囲を指定します。

   ```
   ConfigurationName:
               time = "N-N,HH:MM-HH:MM"
   ```

   または

   ```
   ConfigurationName:
               time = -
   ```

   (no time value specified)

   N は、0 (日曜) から 6 (土曜) までの範囲の曜日を表す数表示であり、*HH* は、00 (午前 0 時) から 23 (午後 11 時) の範囲の時刻を表し、さらに *MM* は、00 から 59 の範囲の分を表します。 日だけ、またはなにも指定しないこともできます。 分の値が 00 であれば、24 という時間の値は、一日の終了時刻として有効です。 特定の構成で、時間範囲の代わりにダッシュ (-) を 入力する場合は、他の構成の時間範囲が無効の場合にその構成が使用されます。 時間範囲を付けない場合は、1 つの構成しか指定できません。

   次に例を示します。

   ```
   conf1:
       time =
   conf2:
       time = "1-5,8:00-17:00"
   conf2
       time = "6-0,14:00-17:00"
   conf3
       time = "22:00-6:00"
   ```

3. 新規構成セットで WLM を更新する場合は、**wlmcntrl -u** コマンドを使用します。
   例:

   ```
   wlmcntrl -u /etc/wlm/MyConfigSet
   ```

この時点で、WLM の現在の構成が時刻に基づく新規の構成セットになります。

confsetcntrl コマンド および **lswlmconf** コマンド を実行して、構成セットを作成および操作することもできます。 次に例を示します。

conf1 のデフォルト構成で confset1 構成セット を作成するには、次のコマンドを実行します。

```
confsetcntrl -C confset1 conf1
```

conf2 を confset1 に追加し、毎日、午前 8 時から 午後 5 時までそれをアクティブ構成にするには、次のコマンドを実行します。

```
confsetcntrl -d confset1 -a conf2 "0-6,08:00-17:00"
```

この構成セットをアクティブ構成にするには、次のコマンドを実行します。

```
wlmcntrl -d confset1
```

リソース・セットの作成

CPU に関する限り、リソース・セット (rsets) を使用するのは、ワークロードを互いに分離するための効果的な方法です。 2 つの異なるワークロードを 2 つのクラスに分け、各クラスに CPU の異なる サブセットを与えることにより、2 つのワークロードが互いに CPU リソースを 要求して競合しないようにすることができます。ただし、依然として、物理メモリーおよび入出力帯域幅を要求して競合することはあります。

リソース・セットを作成する一番簡単な方法は、SMIT インターフェース (**smit addrsetcntl** 高速パス) または **mkrset** コマンドを使用する方法です。

説明のために、以下の例では、4 ウェイ・システムでリソース・セットを作成および 命名する各ステップについて説明します。 この目的では、プロセッサー 0 から 2 を含む リソース・セットを作成し、それを WLM 構成で使用して、スーパークラスのすべての プロセスを、これら 3 つのプロセッサーに制限します。

1. root 権限で、次のコマンドを使用して、使用可能な構築ブロック (リソース・セット の作成元) を表示します。

   ```
   lsrset -av
   ```

   この例の出力を以下に示します。

   ```
   T  Name              Owner   Group   Mode    CPU  Memory  Resources
   r  sys/sys0          root    system  r-----    4   98298  sys/sys0
   r  sys/node.00000    root    system  r-----    4   98298  sys/sys0
   r  sys/mem.00000     root    system  r-----    0   98298  sys/mem.00000
   r  sys/cpu.00003     root    system  r-----    1       0  sys/cpu.00003
   r  sys/cpu.00002     root    system  r-----    1       0  sys/cpu.00002
   r  sys/cpu.00001     root    system  r-----    1       0  sys/cpu.00001
   r  sys/cpu.00000     root    system  r-----    1       0  sys/cpu.00000
   ```

   この出力では、**sys/sys0** はシステム 全体 (この場合、4 ウェイ SMP) を 表します。 WLM クラスが `rset` 属性を指定しない場合は、これが、プロセスがアクセス可能なデフォルト・セットになります。
2. 次の SMIT 高速パスを使用して、リソース・セットを作成および命名します。

   ```
   smit addrsetcntl
   ```

   この例では、フィールドを以下のように埋めてください。

   Name Space
   admin

   Resource Set Name
   proc0_2

   Resources
   リストから、メモリーおよび CPU 0 から CPU 2 まで (sys/cpu.00000 から sys.cpu.00002 まで) に対応する行を選択します。

   All other fields
   リストから選択します。

   フィールドの入力を終え、SMIT を終了すると、`admin/proc0_2` rset が `/etc/rsets` に作成されます。
3. 新規 rset を使用するには、次の SMIT 高速パスを使用して、これをカーネル・データ構造に追加します。

   ```
   smit reloadrsetcntl
   ```

   このメニューでは、データベースの再読み込みのオプションとして、「`now` (即時)」、「`at next boot` (次のブート時)」または「`both` (両方)」を選択することができます。 新規リソース・セットを使用するのはこれが始めてですから、「`both` (両方)」を 選択して、この rset が、今すぐ、および各リブートの後でロードされるようにします。 (既存 rset を変更した場合は、「`now` (即時)」を選択します。)
4. 次の SMIT 高速パスを使用して、新規 rset を WLM クラスに追加します。

   ```
   smit wlmclass_gal
   ```

   クラスを選択してから (この例では、**super1**)、「**Resource Set (リソース・セット)**」フィールドの 使用可能なリストから **admin/proc0_2** を選択します。選択を行い、 SMIT を終了すると、ディスク上の `classes` ファイルが 変更されています。
5. 次のどちらかを行います。

- WLM が実行中であれば、次の SMIT 高速パスを使用して、構成を更新します。

```
smit wlmupdate
```

- WLM が実行中でなければ、次の SMIT 高速パスを使用して開始します。

```
smit wlmstart
```

6. クラスに関する新規リソース・セットの効果をモニターします。例:
   a) クラス **super1** で 90 の CPU ループ (無限ループを実行するプログラム) を開始します。
   b) コマンド・ラインに `wlmstat` を入力します。この例の出力を以下に示します。

```
                CLASS CPU MEM BIO
       Unclassified   0   0   0
          Unmanaged   0   0   0
            Default   8   0   0
             Shared   0   0   0
             System   0   0   0
             super1  75   0   0
             super2   0   0   0
   super2.Default     0   0   0
    super2.Shared     0   0   0
      super2.sub1     0   0   0
      super2.sub2     0   0   0
```

   この出力を見ると、無制約であれば CPU を 100% 必要としたであろう 90 の CPU 制約プロセスは、リソース・セットによって CPU 0 から 2 で実行されるように制限されたため、CPU を 75% しか使用していないことが分かります。
   c) プロセス (その PID で識別される) がアクセスできるリソース・セットを確認するために、次の SMIT 高速パスを使用します。

```
smit lsrsetproc
```

   調べたいプロセスの PID を 入力するか、リストから選択します。 次の出力は、1 つのループ・プロセスに関する 出力です。

```
    CPU  Memory  Resources
      3   98298  sys/mem.00000 sys/cpu.00002 sys/cpu.00001 sys/cpu.00000
```

   ただし、`rset` 属性が指定されていないクラスからのプロセスは、`Default` リソース・セットを使用します。このリソース・セットには、排他的リソース・セットに含まれているプロセッサー以外のすべてのプロセッサーが含まれています。どのクラスにも属さないプロセスは、`System` クラス (ルート・プロセスの場合) または `Default` クラス (非ルート・プロセスの場合) を使用します。これらのどちらのクラスにも、リソース・セットが定義されている場合があります。

   次の出力は **init** プロセスからのものです。このプロセスは、リソース・セットを指定しないクラスの中にあります。

```
    CPU  Memory  Resources
      4   98298  sys/sys0
```

この時点で、リソース・セットが存在し、WLM 内の 1 つ以上のクラスによって使用されるようになります。

注 : WLM は、現在、**bindprocessor** サブルーチン割り当て または別の rset アタッチメントをもつプロセスに、rset アタッチメントを設定しません。 他のアタッチメントが存在しなくなると、WLM は rset を自動的に割り当てます。

注 : リソース・セットは、`Default` クラスおよび `System` クラスを含むすべての WLM クラスに対して作成できます。

ワークロードを統合するためのワークロード・マネージャー構成

ワークロード・マネージャー (WLM) を使用すると、システムでジョブが 使用する リソースを制御することができます。

デフォルトの WLM 構成テンプレートが、インストール済みのすべての AIX オペレーティング・システムに存在します。 以下の手順では、WLM 構成テンプレートを更新して、共用サーバー上に リソース管理ポリシーをインプリメントします。 更新した構成は、テストの開始ポイントとして使用できます。 WLM の具体的な構成方法は、使用する環境のワークロードおよびポリシーの要件によって異なります。

注 :

1. WLM を有効に使用するには、既存システムのプロセスおよびパフォーマンスについての 幅広い知識が必要です。 ワークロードに適応できる構成を開発するには、テストおよびチューニングを繰り返し行うことが必要になるでしょう。 極端な値や不正確な値 を使用して WLM を構成すると、システム・パフォーマンスが著しく低下する場合があります。
2. プロセスの種別属性 (例えば、ユーザー名、グループ名、または アプリケーション名) を事前に 1 つ以上知っていると、WLM の構成プロセスは簡単になります。 リソースの現在の使用状況を把握していない場合は、**topas** などのツールを使用して、1 次リソース・ユーザーのプロセスを識別し、得られた情報を、クラスおよびルールを定義するための開始ポイントとして使用します。
3. 以下のシナリオでは、ユーザーが 129 ページの『ワークロード・マネージメントの概念』 で 説明されているワークロード・マネージャーの基本概念を理解しているものとします。

WLM 構成ファイルは、`/etc/wlm/`*ConfigurationName* ディレクトリーにあります。 各スーパークラスの個々のサブクラスは、`/etc/wlm/`*ConfigurationName/SuperClassName* という名前の 構成ファイルで定義されています。これらのファイルの詳細については、「ファイル参照」を参照してください。

次の手順では、2 つの別個の部門サーバーのワークロードを、より大規模な 1 つのサーバーに統合します。この例では構成ファイルを編集しますが、SMIT (**smit wlmconfig_create** 高速パス) を使用して、構成することもできます。 要約すると、この手順では以下のことを行います。

1. 統合したいアプリケーションのリソース 要件を識別する。 これにより、大規模サーバーに 移動できるアプリケーションの数を知ることができます。
2. 統合したワークロードのテストを開始するために、tier、リソース share および制限 を定義する。
3. 期待した結果を得られるまで、構成を微調整する。

ここで解説する情報は AIX の特定バージョンを使用してテストされたものです。したがって、その内容は使用される AIX のバージョンおよびレベルによってかなり異なることがあります。

ステップ 1. アプリケーション要件の識別

このシナリオでのワークロードは、データベース・サーバーで典型的にみられるものです。 ジョブは、以下の一般的なカテゴリーに分類されるものとします。

Listeners
ここに含まれるジョブは、ほとんどの時間スリープしていて、要求に応えて定期的にウェイクアップするプロセスです。 これらのプロセスは、リソースをそれほど消費しませんが、応答時間が重要になります。

Workers
ここに含まれるジョブは、ローカル側から、または、リモート側からの要求に応えて作業を行うプロセスです。 これらのプロセスは、多くの CPU 時間およびメモリーを使用します。

Reporters
ここに含まれるジョブは、自動化タスクを実行するプロセスです。 これらのプロセスでは、多くの CPU 時間またはメモリーが必要となりますが、応答時間が遅くても問題ありません。

Monitors
ここに含まれるジョブは、通常、システムまたはアプリケーションの 状態を検査するために、定期的に実行されるプロセスです。 これらのプロセスは、大量のリソースを使用しますが、時間は短時間です。

Commands
ここに含まれるジョブは、システム・ユーザーが任意の時間に実行するコマンドまたは その他のアプリケーションです。 それらのリソース要件は予測することができません。

上記の作業以外に、スケジュール化されたジョブは、次のどちらかのカテゴリーに 分類されます。

SysTools
ここに含まれるジョブは、自動化タスクを実行するプロセスです。 これらのジョブは、システム 操作では重要でありませんが、定期的であり、特定の制約された時間内に実行される必要があります。

SysBatch
ここに含まれるジョブは、頻繁には実行されないプロセスです。また、システム 操作では重要でなく、一定時間内に終了する必要もありません。

構成を作成する最初のステップは、クラスおよびルールを定義することです。 以下の手順では、上記にリストした一般ジョブ・カテゴリーを使用してクラスを定義します。 次の手順を使用してください。

1. 次のコマンドを実行して、/etc/wlm ディレクトリー内 に MyConfig という名前の新規構成を作成します。

   ```
   mkdir /etc/wlm/MyConfig
   ```

2. 次のコマンドを実行して、テンプレート・ファイルを /etc/wlm/MyConfig ディレクトリー にコピーします。

   ```
   cp -pr /etc/wlm/template/* /etc/wlm/MyConfig
   ```

3. スーパークラスを 作成するには、任意のエディターを使用し、/etc/wlm/MyConfig/classes ファイルを編集して以下の内容を含めます。

   ```
   System:

   Default:

   DeptA:

   DeptB:

   SysTools:

   SysBatch:
   ```

   まず始めに、それぞれの部門ごとに 1 つのスーパークラスを 定義します (2 つの部門が サーバーを共用するため)。 SysTool スーパークラスおよび SysBatch スーパークラスは、上記の一般カテゴリーで説明した、スケジュールされたジョブを処理します。 System スーパークラス および Default スーパークラスは、常に定義されます。

4. MyConfig ディレクトリー内に、DeptA および DeptB の それぞれのスーパークラスごとに 1 つのディレクトリーを作成します。 (構成を作成する際には、サブクラスをもつ各スーパークラスに 1 つのディレクトリーを作成 する必要があります。) 以下の手順では、各部門のスーパークラスに 5 つのサブクラス (各作業カテゴリーに 1 つ) を定義します。

5. ジョブの各一般カテゴリー用サブクラスを作成するには、/etc/wlm/MyConfig/DeptA/classes ファイルおよび /etc/wlm/MyConfig/DeptB/classes ファイルを編集して、以下の項目を含めます。

   ```
   Listen:

   Work:

   Monitor:

   Report:

   Command:
   ```

   注 : classes ファイルの内容は、それぞれのスーパークラスごとで 異なる場合があります。

   クラスを指定した後に、以下のステップで、スーパークラスおよびサブクラス のレベルで、プロセスを分類する際に使用する分類ルールを作成します。 簡単にするために、すべてのアプリケーションが既知の場所から実行され、ある 1 つの部門からのすべてのプロセスが deptAUNIX グループの下で実行され、その他の部門からのプロセスが deptB UNIX グループの下で実行されるものとします。

6. スーパークラス割り当てルールを作成するために、/etc/wlm/MyConfig/rules ファイルを編集して、以下の項目を含めます。

```
DeptA - - deptA - -
DeptB - - deptB - -
SysTools - root,bin - /usr/sbin/tools/* -
SysBatch - root,bin - /usr/sbin/batch/* -
System - root - - -
Default - - - - -
```

   注 : 同一のアプリケーション の複数のインスタンスを実行することができ、すべての種別属性 (タグ以外) が同じ場合は、**wlmassign** コマンド または **wlm_set_tag** サブルーチン を使用し、インスタンスを別々のクラスに割り当てることによって、それらのインスタンスを区別してください。

7. より具体的なサブクラス・ルールを作成するため、次の内容をもった /etc/wlm/MyConfig/DeptA/rules ファイル および /etc/wlm/MyConfig/DeptB/rules ファイルを作成します。

```
Listen - - - /opt/myapp/bin/listen* -
Work - - - /opt/myapp/bin/work* -
Monitor - - - /opt/bin/myapp/bin/monitor -
Report - - - /opt/bin/myapp/report* -
Command - - - /opt/commands/* -
```

8. 各クラスのリソース消費動作を判別するため、次のコマンドを実行して、WLM をパッシブ・モードで開始します。

```
wlmcntrl -p -d MyConfig
```

   WLM をパッシブ・モードで開始したら、まず、各アプリケーションを別々に実行し、個別アプリケーションのリソース要件の詳細な見通しを得ることができます。 次に、すべてのアプリケーションを同時に実行して、すべてのクラス間での相互作用をより正確に 確認することができます。

アプリケーション・リソース要件を確認する別の方法として、統合するアプリケーションの存在する個別のサーバー上で、まず WLM をパッシブ・モードで 実行することもできます。 この方法の欠点は、大規模システムで構成を再作成 しなければならない点、およびリソースの所要パーセンテージが、大規模システムで異なる可能性がある点です。

ステップ 2. Tier、Share、および制限の定義

WLM の構成とは、リソース管理ポリシーをインプリメントすることです。 WLM をパッシブ・モードで実行すると、特定のワークロードに対して、リソース管理ポリシーが合理的であるかどうかを判別するための情報を得ることができます。 これを行えば、tier、share、および制限を定義して、リソース管理ポリシーに基づいてワークロードを調整できるようになります。

このシナリオでは、以下の要件を満たしているものとします。

- System クラスは最高の優先順位をもっており、いつでも、一定パーセントの システム・リソースを利用することが保証されている。
- SysTools クラスは、いつでも一定のパーセントのリソースを利用できるが、DeptA および DeptB で実行されているアプリケーションに大きな影響がでるほど多くの リソースは利用できない。
- SysBatch クラスは、システム上のその他の作業に干渉することはない。
- DeptA は使用可能なリソース (つまり、share をもったクラスが使用できるリソース) の 60% を 受け取り、DeptB は 40% を受け取る。 DeptA および DeptB の中では、以下のとおりです。
  - Listen クラス内のプロセスは、短い待ち時間で要求に応答しなければならないが、多くのリソースを消費することはできない。
  - Work クラスはほとんどのリソースを消費できなければならない。 Monitor クラス および Command クラスは、一定のリソースを消費できるが、Work クラスより多くのリソース は消費できない。
  - Report はその他の作業に干渉することはない。

以下の手順では、tier、share、および制限を定義します。

1. スーパークラス tier を作成するには、任意のエディターを使用し、/etc/wlm/MyConfig/classes ファイルを編集して以下の内容を含めます。

```
System:

Default:

DeptA:
        localshm = yes
        adminuser = adminA
        authuser = adminA
        inheritance = yes

DeptB:
        localshm = yes
        adminuser = adminB
        authuser = adminB
        inheritance = yes

SysTools:
        localshm = yes

SysBatch:
        tier = 1
        localshm = yes
```

   SysBatch スーパークラスは tier 1 に置きます。このクラスには、システム上の他の作業に干渉してほしくない、非常に低い優先順位のジョブが含まれるからです。 (tier を指定しないと、クラスは tier 0 のデフォルト設定になります。) 各部門のスーパークラスの 管理は、adminuser および authuser の属性で定義します。 DeptA および DeptB で継承属性を使用可能にします。 継承属性をもったクラスで 開始されたすべての新規プロセスは、そのクラスのまま変わりません。

2. 各ジョブ・グループのサブクラス tier を作成するには、/etc/wlm/MyConfig/DeptA/classes ファイル および /etc/wlm/MyConfig/DeptB/classes ファイルを編集して、以下の内容を含めます。

```
Listen:

Work:

Monitor:

Report:
        tier = 1
Command:
```

3. スーパークラスに 初期 share を割り当てるには、/etc/wlm/MyConfig/shares ファイルを編集して以下の内容を含めます。

```
DeptA:
        CPU = 3
        memory = 3

DeptB:
        CPU = 2
        memory = 2
```

   CPU に合計 5 share を割り当てることにしたので、DeptA プロセスは、CPU の 合計リソース 5 share のうちから 3 share (60%) を利用できます。DeptB プロセスは、5 share のうちから 2 share (40%) を利用できます。 SysTools、System、および Default の 各クラスには share を割り当てなかったので、それらの消費目標値は、アクティブな share 数 の影響を受けません。このため、それらのクラスは、制限に達するまでは、DeptA および DeptB より高い優先順位でリソースにアクセスできます。 SysBatch クラスだけは tier 1 のスーパークラスであるため、share を 1 つも割り当てませんでした。したがって、share を割り当てることはできません。 SysBatch クラスのジョブが消費できるのは、tier 0 のどのクラスによっても使用されない リソースだけです。

4. サブクラスに初期 share を割り当てるには、/etc/wlm/MyConfig/DeptA/shares ファイルおよび /etc/wlm/MyConfig/DeptB/shares ファイルを編集して、以下の内容を含めます。

```
Work:
        CPU = 5
```

```
        memory = 5

Monitor:
        CPU = 4
        memory = 1

Command:
        CPU = 1
        memory = 1
```

Listen クラスには share を割り当てなかったので、このクラスがリソースを要求した場合は、スーパークラスの中では最も高い優先順位でリソースにアクセスします。Work クラスのリソース要件が最大なので、このクラスに最大数の share を割り当てました。それぞれ、クラスの監視された動作および相対的な重要度に基づいて、Monitor および Command の各クラス に share を割り当てました。Report クラスだけは tier 1 のサブクラスであるため、share を割り当てませんでした。したがって、share を割り当てることはできません。Report クラスのジョブが消費できるのは、tier 0 のサブクラスに使用されないリソースだけです。

この例の以下のステップでは、share を割り当てなかったクラスに制限を割り当てます。(share をもったクラスに制限を割り当てることもできます。 詳細については、154 ページの『ワークロード・マネージャーによるリソース管理』を参照してください。)

5. スーパークラスに制限を割り当てるには、`/etc/wlm/MyConfig/limits` ファイルを編集して以下の内容を含めます。

```
Default:
        CPU = 0%-10%;100%
        memory = 0%-10%;100%

SysTools:
        CPU = 0%-10%;100%
        memory = 0%-5%;100%

System:
        CPU = 5%-50%;100%
        memory = 5%-50%;100%
```

System、SysTools、および Default の各クラスが、システム上のその他の作業に著しく 干渉しないようにするために、これらのクラスにソフト最大制限を割り当てました。 System クラスには、CPU およびメモリーに関して、最小制限を割り当てました。このクラスには、システム 操作で重要になるプロセスが含まれていること、また、このクラスが保証されたリソース量を消費できなければならないからです。

6. サブクラスに制限を割り当てるには、`/etc/wlm/MyConfig/DeptA/limits` ファイルおよび `/etc/wlm/MyConfig/DeptB/limits` ファイルを 編集して以下の内容を含めます。

```
Listen:
        CPU = 10%-30%;100%
        memory = 10%-20%;100%

Monitor:
        CPU = 0%-30%;100%
        memory = 0%-30%;100%
```

注 : 制限は各サブクラス・ファイルで異なる場合があります。

Listen および Monitor の各クラスが、同じスーパークラス内のその他のサブクラスに著しく 干渉しないようにするために、これらのクラスにソフト最大制限を割り当てました。 特に、Work クラスが、クラス内のジョブを処理するためのリソースを利用できない場合は、システムがジョブ実行の要求を受け続ける状況は望ましくありません。 また、応答時間を速くするために、Listen クラスに最小制限を割り当てました。 メモリーの最小制限によって、このクラスによって使用されるページが ページ置き換えによってスチールされることがなくなります。このため、実行時間が高速になります。 CPU の最小制限によって、プロセスを実行できるときに、それらのプロセスが、スーパークラスの中で CPU リソースに対する最高の優先順位アクセス権 をもてるようになります。

ステップ 3. ワークロード・マネージャー構成の微調整

1. **wlmstat** コマンドを使用してシステムをモニターし、WLM による規制がゴールに沿ったものかどうか、および一部のアプリケーションが必要以上にリソースを獲得している一方で、不当にリソースを奪われているアプリケーションがないかどうかを検証します。 この場合、Share を調整し、WLM をリフレッシュします。
2. Share、制限、および Tier 番号をモニターして調整しながら、 一部またはすべてのスーパークラスについて、 サブクラスの管理を委任するかどうかを決定します。 次に、管理者はサブクラスの Share、制限、および Tier 数をモニターしてセットアップすることができます。

それぞれのサブクラスの管理者は、 それぞれのスーパークラスのサブクラスについてこのプロセスを 繰り返すことができます。 相違点は、パッシブ・モードでは、サブクラス・レベルのみで WLM を実行することはできないという点だけです。 サブクラス構成とチューニングは、WLM を使用してアクティブ・モードで実行しなければなりません。 スーパークラスのユーザーおよびアプリケーションに 影響を与えないようにする方法の 1 つには、サブクラスの Tier 数と、Share および制限にデフォルト値 (Share には '-' (ハイフン)、最小に 0%、ソフトおよびハード最大 (%) に 100%) を使用して開始することがあります。 これらの設定値を指定した場合、WLM がサブクラス間のリソース割り当てを調整することはありません。

詳細情報

- ワークロード・マネージャー
- ワークロード・マネージメント
- 「パフォーマンス・マネージメント」のワークロード・マネージメントの診断
- ファイル参照 におけるファイル classes、limits、rules、および shares の説明。
- **topas**、**wlmassign**、**wlmcheck**、**wlmcntrl**、および **wlmstat**。
- WLM サブルーチンの説明、特に **wlm_set_tag**

クラス

ワークロード・マネージャーは、サービスのクラスを定義し、それらのクラスのそれぞれにリソースを割り当てることにより、システム・リソース割り当ての制御に役立ちます。

各クラスは、そのクラスのリソース権とその他の動作を決める一連の属性を持ちます。 システム上のすべてのプロセスは、あるサービス・クラスに分類されます。 そのため、各プロセスにはそのクラスに限定されたリソース権と動作が適用されます。 クラスへのプロセスの割り当ては、手動割り当てを使用して手動で行われるか、 またはユーザー定義の分類規則に従って自動的に行われるかのいずれかです。

WLM は、2 つのレベルのクラス (スーパークラス およびサブクラス) をサポートします。 スーパークラスには、使用可能なシステム・リソースに基づいたリソース権が与えられます。サブクラスには、それらが関連付けられているスーパークラスの資格を基準にしたリソース権が与えられます。 オプションとして、スーパークラスのプロセスをよりきめ細かく制御できるようにサブクラスを定義できます。 また、スーパークラスに対して admin または admin を指定して、サブクラスを定義する責任を委任することもできます。

スーパークラス・レベルおよびサブクラス・レベルの場合はともに、SMIT、またはコマンド・ライン・インターフェースを 使用して、クラス、リソースの Share および制限、ならびに規則を定義できます。 アプリケーションは WLM API を使用できます。 構成定義は、WLM 属性ファイル と呼ばれる一連のテキスト・ファイルに保持されます。

クラス名は 16 文字までの長さであり、大文字および小文字と数字および下線 (_) だけを使うことができます。 ある特定の WLM 構成では、各スーパークラス 名は固有でなければなりません。 各サブクラス名は、そのスーパークラス 内では固有でなければなりませんが、その他のスーパークラスのサブクラス 名とは一致しても構いません。 各サブクラスを一意的に識別できるように、サブクラスの完全名は、 例えば *Super.Sub* のように、 ドットで区切ったスーパークラス 名とサブクラス名から構成されます。

スーパークラス

システム管理者は最大 64 のスーパークラスを定義することができます。

さらに、以下のような 5 つのスーパークラスが 自動的に作成されます。

***Default* スーパークラス**
これはデフォルトのスーパークラスであり、常に定義されます。 特定のスーパークラスに自動的に割り当てられることのない非ルート・プロセスはすべて、この Default スーパークラスに割り当てられます。 特定の割り当て規則を指定すれば、その他のプロセスを *Default* スーパークラスに割り当てることもできます。

***System* スーパークラス**
規則によって特定のクラスに対してまだ割り当てられていなければ、このスーパークラスにはすべての特権付き (ルート) プロセスが割り当てられます。 このスーパークラスはまた、カーネル・メモリー・セグメント、およびカーネル・プロセスに属するメモリー・ページも収集します。 System スーパークラス用に特定の割り当て規則を指定すれば、その他のプロセスをこのスーパークラスに割り当てることもできます。 このスーパークラスのメモリーの最小値は、デフォルトで 1% です。

***Shared* スーパークラス**
このスーパークラスは、複数のスーパークラス内のプロセスが共用するメモリー・ページを受け取ります。 それには、共有メモリー領域内のページと、複数のスーパークラス (または別のスーパークラスのサブクラス) 内のプロセスで使われているファイル内のページが含まれます。 いずれも同じ 1 つのスーパークラス (または同じスーパークラスのサブクラス) に属する複数のプロセスで使われている共有メモリーとファイルは、そのスーパークラスに関連付けられます。 ページが Shared スーパークラスに置かれるのは、別のスーパークラスのプロセスが共有メモリー領域またはファイルにアクセスした場合のみです。 このスーパークラスに対しては、物理メモリー Share および上下限値だけを適用することができます。 このスーパークラスには、他のリソース・タイプ、サブクラス、または割り当て規則用の Share または制限を指定できません。 同じスーパークラスの異なるサブクラスのプロセスによって共用されるメモリー・セグメントが *Shared* サブクラスに分類されるか、その元のサブクラスにとどまるかは、元のサブクラスの **localshm** 属性の値によって決まります。

***Unclassified* スーパークラス**
これは、未分類プロセスのメモリー割り当てです。 WLM の始動時に存在していたプロセスは、ロードされる WLM 構成の割り当て規則に従って分類されます。 この初期分類時に、各プロセスに付加されているすべてのメモリー・ページは、そのプロセスが属する スーパークラス (共用されていないか、または同じスーパークラス内のプロセスにより共用されている場合) か、 別のスーパークラス内のプロセスで共用されている場合は、*Shared* スーパークラスのいずれかに チャージされます。

ただし、この分類時にどのプロセスにも (したがってどのクラスにも) 直接結び付けられないいくつかのページがあり、 このメモリーは *Unclassified* スーパークラスにチャージされます 。 このメモリーの大半はその後、プロセスからアクセスされたときか、または WLM の始動後に解放されてプロセスに対して再割り当てされたときに、正しく再分類されます。 *Unclassified* スーパークラスにはプロセスはありません。 このスーパークラスに対しては、物理メモリーの Share および制限を適用することができます。 このスーパークラスには、他のリソース・タイプ、サブクラス、または割り当て規則用の Share または制限を指定できません。

***Unmanaged* スーパークラス**
Unmanaged と名前が付けられた、特殊なスーパークラスは常に定義されます。 このクラスにはプロセスは割り当てられません。 この クラスは、WLM によって管理されないシステム内の、すべてのピンされたページのメモリー使用率を集計します。 `waitproc` プロセスの CPU 使用率は、システムが 100% の使用率であると見えること を防ぐため、どのクラスでも累積されません。 このスーパークラスには、どのリソース・タイプ、サブクラス、または指定された割り当て規則用の Share または制限を指定できません。

サブクラス
システム管理者またはスーパークラス管理者は、61 個までのサブクラスを定義することができます。

それ以外に、*Default* および *Shared* の 2 つのサブクラスが常に定義されます。

***Default* サブクラス**
これはデフォルトのサブクラスであり、常に定義されます。 スーパークラスの特定のサブクラスに自動的に割り当てられることのないプロセスはすべて、この *Default* サブクラスに割り当てられます。 また、特定の割り当て規則を指定すれば、他のプロセスを *Default* サブクラスに割り当てることもできます。

***Shared* サブクラス**
このサブクラスは、スーパークラスの 複数のサブクラス内のプロセスが使用するすべてのメモリー・ページを受け取ります。 それには、共有メモリー領域内のページと、同じスーパークラスの 複数のサブクラスのプロセスで使われているファイル内のページが含まれます。 すべて同じ 1 つのサブクラスに属する複数のプロセスで使われている共有メモリーとファイルは、そのサブクラスに関連付けられます。 ページがスーパークラスの *Shared* サブクラスに置かれるのは、同じスーパークラスの 別のサブクラスのプロセスが共有メモリー領域またはファイルにアクセスした場合のみです。 *Shared* サブクラスにはプロセスはありません。 このサブクラスに対しては、物理メモリーの Share および制限だけを適用することができます。他のリソース・タイプまたは割り当て規則用の Share または制限は指定できません。 同じスーパークラスの異なるサブクラスのプロセスによって共用されるメモリー・セグメントが *Shared* サブクラスに分類されるか、その元のサブクラスにとどまるかは、元のサブクラスの `localshm` 属性の値によって決まります。

クラス属性
WLM クラスのすべての属性をリストする。

クラス名 (Class Name)
16 文字までの長さにすることができ、大文字および小文字と数字および下線 (_) だけを使うことができます。

Tier
それぞれのクラスへのリソース割り当てを優先順位付けするのに使用する 0 から 9 の番号。

継承 (Inheritance)
子プロセスが親からクラス割り当てを継承するかどうかを指定します。

localshm
あるクラスに属すメモリー・セグメントが共用クラスに移行されないようにします。

管理者 (adminuser、admingroup、authgroup) (スーパークラスのみ)
スーパークラスの 管理を委任します。

権限 (authuser、authgroup)
プロセスをクラスに手動で割り当てる権利を委任します。

リソース・セット (rset)
ある特定のクラスからアクセスできる一連のリソースを CPU (プロセッサー・セット) 単位で制限します。

delshm
最後のプロセス参照が仮想メモリー制限のために強制終了した場合に、共有メモリー・セグメントを削除します。

vmeforce
クラスがその仮想メモリー制限に達したときに、クラス内のすべてのプロセスを強制終了するか、問題のプロセスのみを強制終了するかを示します。

io_priority
クラスに分類されたスレッドにより発行された、入出力要求に割り当てられた優先順位を指定します。この優先順位は、デバイス・レベルでの入出力バッファーを優先順位付け するために使用されます。ストレージ・デバイスが入出力優先順位をサポートしない場合、優先順位は無視されます。 有効な入出力優先順位は、0 から 15 に及びます。

関連概念
ワークロード・マネージメントのクラスへのプロセスの割り当て
プロセスは、システム管理者が指定したクラス割り当て規則を使用してクラスに割り当てられます。 分類基準のベースになるのは、ユーザー ID、グループ ID、アプリケーション・ファイルの名前、プロセスのタイプ、およびアプリケーション・タグなどの一連のプロセス属性の値です。

***Tier* 属性**
Tier は、システム・リソースが WLM クラスに割り当てられる際の順序を表します。

管理者は、クラスを最大 10 個までの Tier に定義できます。Tier には 0 から 9 の番号が振られますが、0 が最高、つまり最も重要な Tier です。 Tier 0 が使用可能なリソースの量は、使用可能なすべてのシステム・リソースです。 優先順位が低い (番号が大きい) Tier が使用できるリソースの量は、それよりも優先順位が

高いいずれの Tier も使用していないリソースの量です。 クラスのターゲット使用量パーセントは、それが属す Tier 内のアクティブ Share の数およびその Tier が使用可能なリソースの量に基づいています。 Tier 0 は、それ自体に使用可能なリソースを常に持つことが保証されている唯一の Tier であるため、システム操作に必須のプロセスは、この Tier 内のクラスに分類することをお勧めします。 クラスにどの Tier 値も設定しなかった場合は、そのクラスは Tier 0 に入れられます。

Tier は、スーパークラス・レベルとサブクラス・レベルの両方で指定できます。 スーパークラスの Tier は、それぞれのスーパークラスに 対するリソース割り当ての優先順位を指定するのに使われます。 サブクラスの Tier は、同じスーパークラスのそれぞれのサブクラスに 対するリソース割り当ての優先順位を指定するのに使われます。 別々のスーパークラスのサブ Tier 間はまったくの無関係です。

継承 *(Inheritance)* 属性
クラスの継承属性は、そのクラス内のプロセスを、そのプロセスの分類属性のいずれかが変更されたときに、自動的に再分類する必要があるかどうかを示します。

fork サブルーチンで新しいプロセスを作成すると、そのプロセスは、継承が可能になっているかどうかとはかかわりなく、自動的にその親のクラスを継承します。 ただし、親プロセスがタグを持っていて、その **inherit tag at fork** がオフに設定されており、さらに親のクラスに対するクラス継承がオフになっている場合は例外です。 この場合、子プロセスは分類規則に従って再分類されます。

あるクラスに対して継承が可能になっていない場合、 分類規則の中で使用されているプロセス属性を変更するサービスのどれかを呼び出すと、 その後そのクラス内のすべてのプロセスは、分類規則に従って自動的に分類されます。 そのような呼び出しの中で最も一般的なのは exec サブルーチンですが、 分類を変更する可能性のあるその他のサブルーチンとしては、 setuid、setgid、 plock、setpri、および wlm_set_tag があります。 継承が可能になっている場合、そのプロセスは分類規則に基づく再分類の対象とはならず、 現行のクラスにとどまります。 手動割り当ては、継承よりも優先順位が高いため、 継承が可能になっているクラス内のプロセスを再分類するのに使用できます。

inheritance 属性の指定値は、yes か no のどちらかです。 指定されないと、クラスの継承は可能にはなりません。

この属性は、スーパークラスおよびサブクラスの 両方のレベルで指定することができます。 特定のスーパークラスのサブクラスの場合、次のようになります。

- スーパークラス・レベルとサブクラス・レベルの両方で継承属性を yes に設定すると、そのサブクラス内のプロセスの子は同じサブクラスに留まります。
- 継承属性をスーパークラスに 対しては yes に、サブクラスに対しては no (または未指定) に設定 すると、サブクラス内のプロセスの子は同じスーパークラスに留まり、そのスーパークラスの 割り当て規則に従って、そのスーパークラスのいずれかのサブクラスに 分類されます。
- 継承属性をスーパークラスに 対しては no (または未指定) に、サブクラスに対しては yes に設定 すると、サブクラス内のプロセスの子は、そのスーパークラスの 自動割り当て規則に従うことになります。
  - プロセスは、同じスーパークラス内の規則によって分類された場合、サブクラス内にとどまります (サブクラスの割り当て規則には実行依頼されません)。
  - プロセスが別のスーパークラス内のスーパークラス 規則によって分類された場合、新しいスーパークラスのサブクラス割り当て規則が用いられて、プロセスが割り当てられる新しいスーパークラスのサブクラスが判別します。
- スーパークラスとサブクラスの 両方の継承属性を no (または未指定) に設定すると、サブクラス内の プロセスの子は、標準自動割り当てに従うことになります。

***localshm* 属性**
localshm 属性は、スーパークラスおよびサブクラスの 両方のレベルで指定することができます。

localshm 属性を使用するのは、あるクラスに属するメモリー・セグメントが他のクラスの プロセスにアクセスされたとき、*Shared* スーパークラスまたは サブクラスに移行しないようにするためです。 この属性の考えられる値は、yes または no です。 yes の値は、このクラスの共有メモリー・セグメントが このクラスのローカル側にとどまり、該当する *Shared* クラスに移行してはならないことを意味します。 no の値は、この属性が指定されていない場合のデフォルトです。

メモリー・セグメントはページ・フォールトで分類されます。 セグメントが作成されると、 そのセグメントには *Unclassified (*未分類*)* スーパークラスに属するとのマークが付けられます。 セグメントの最初のページ・フォールトが起こったときに、 このセグメントは障害の起きているプロセスと同じクラスに分類されます。 後で、このセグメント・ページとは異なるクラスに属するプロセスがこのセグメントでページ・フォールトを起こすと、WLM はそのセグメントを適切な *Shared* クラス (スーパークラスまたはサブクラス) に再分類する必要があるかどうか考慮します。 障害のあるプロセスとセグメントが別々のスーパークラスに属する場合は、以下のいずれかが起きます。

- セグメントのスーパークラスで `localshm` 属性が `yes` に設定されている場合、セグメントはその現在のスーパークラスにとどまります。 セグメントの サブクラスで `localshm` 属性が `yes` に設定されている場合、そのセグメントはそのときのサブクラスに留まります。 スーパークラスの `localshm` 属性が `yes` に設定されていても、そのスーパークラスのサブクラス 属性が `no` に設定されている場合は、セグメントは現在のスーパークラスの *Shared* サブクラスに入れられます。
- セグメントのスーパークラスで `localshm` 属性が `no` に設定されている場合、 そのセグメントは *Shared* スーパークラスに 入れられます。 これがデフォルトのアクションです。

障害のあるプロセスとセグメントが同じスーパークラスの別々のサブクラスに所属し、セグメントのサブクラスで `localshm` 属性が `yes` に設定されている場合、セグメントはその現在のクラス (スーパークラスおよびサブクラス) にとどまります。 それ以外の場合、セグメントはスーパークラスの *Shared* サブクラスに入れられます。

もちろん、障害のあるプロセスとセグメントが同じクラス (同じスーパークラスの 同じサブクラス) に属する場合は、`localshm` 属性の値にかかわらず、セグメントの再分類は行われません。

管理者属性
スーパークラス 管理をユーザーまたはユーザーのグループに委任する場合は、`adminuser` および `admingroup` 属性を使用します。

注 : これらの属性はスーパークラスでのみ 有効です。

`adminuser` 属性は、スーパークラスでの 管理タスクの実行を許可されるユーザーの名前 (`/etc/passwd` にリストされたときの) を指定します。 `admingroup` 属性は、スーパークラスでの 管理タスクの実行を許可されるユーザーのグループの 名前 (`/etc/group` にリストされたときの) を指定します。

属性ごとに 1 つの値 (ユーザーまたはグループ) しか指定できません。 どちらか片方を指定しても、両方とも指定しても、またはどちらも指定しなくてもかまいません。 そのユーザーまたはグループは、以下を行う権限を持つことになります。

- サブクラスの作成と削除
- サブクラスの属性およびリソースの Share および制限の変更
- サブクラスの割り当て規則の定義、除去、または変更
- スーパークラスのアクティブな WLM 構成のリフレッシュ (更新)

許可属性
`authuser` および `authgroup` 属性はすべてのクラスで有効です。 このクラスは、クラス (スーパークラスまたはサブクラス) にプロセスを手動で割り当てる許可を受けるユーザーまたはグループを指定するのに使われます。

プロセス (またはプロセス・グループ) をスーパークラスに 手動で割り当てると、そのスーパークラスの割り当て規則が使われて、スーパークラスのどのサブクラスに 各プロセスが割り当てられるかが決められます。

属性ごとに 1 つの値 (ユーザーまたはグループ) しか指定できません。 どちらか片方を指定しても、両方とも指定しても、またはどちらも指定しなくてもかまいません。

リソース・セット属性
rset と呼ばれるリソース・セット属性は、 任意のクラスに指定できます。 その値は、システム 管理者によって定義されたリソース・セットの名前になります。

rset 属性は、システム 上で使用できる CPU リソースのサブセット (プロセッサー・セット) を表します。 デフォルトは「system」であり、この場合、システム 上で使用可能なすべての CPU リソースにアクセスでき

るようになります。 ただ 1 つ、*rset* をサブクラスに対して指定すると、そのセット内の CPU のセットは当該スーパークラスが 使用できる CPU のサブセットでなければならないという制限があります。 詳しくは、**mkrset** コマンドを参照してください。

注 : Tier 0 にないクラスへのリソース・セットの割り当ては慎重に検討してください。 なぜなら、優先順位の低い Tier はそれより優先順位の高い Tier が使用していないリソースにしか アクセスできないため、Tier 0 でないクラスをシステム 上の CPU の、あるサブセットに制限してしまうと、それらの CPU で使用可能な CPU 時間がない場合に CPU 欠乏状態になる可能性があるからです。

ワークロード・マネージャーにおけるプロセス分類

WLM では、プロセスは次の 2 つの方法のいずれかで分類できます。

- プロセスの分類属性が変更されたときに、割り当て規則を使用してプロセスが自動的に割り当てられる。 WLM がアクティブ・モードで稼働している間は、 この自動割り当ては常に有効です (オフにはできません)。 これが最も一般的なプロセス分類方法です。
- 選択したプロセスまたはプロセス・グループは、プロセスと ターゲット・クラスの両方に対する必須権限を持つユーザーが、手動でクラスに割り当てが可能。 手動割り当ては、 直接または SMIT を通じて呼び出す WLM コマンドを使用して、あるいは、 WLM アプリケーション・プログラミング・インターフェースの機能を使用するアプリケーションから行えます。 この手動割り当てを実行すると、 自動割り当てと継承は指定変更されます。

ワークロード・マネージャーにおける自動クラス割り当て

クラスに対するプロセスの自動割り当てでは、WLM 管理者によって指定された一連のクラス割り当て規則を使います。

割り当て規則は次の 2 つのレベルに分かれます。

- 所定のプロセスをどのスーパークラスに割り当てるかを判別するのに使われる WLM 構成レベルの一連の割り当て規則。
- さらに、サブクラスが定義されている各スーパークラスには、スーパークラスのどのサブクラスにそのプロセスを割り当てればよいかを判断するのに使われる一連の割り当て規則が備えられています。

どちらのレベルの割り当て規則も、一連のプロセス属性の値に基づいています。 この属性には次のものがあります。

- プロセス・ユーザー ID
- プロセス・グループ ID
- 実行されるアプリケーション (プログラム) のパス名
- プロセスのタイプ (例えば、32 ビット または 64 ビット)
- プロセス・タグ

タグは、文字列として定義されるプロセス属性で、アプリケーションが WLM API を使用してプログラム別に設定できます。

属性が変更されるたびに、それらのプロセス属性の値がクラス割り当てルール・ファイル (名前は `rules`) 内に記述されている値のリストと比較され、分類が行われます。 この比較により、どの規則がそのプロセス属性の現在値と一致するかが判別されます。

クラス割り当て規則は、以下のフィールドを含むテキスト・ストリングで、各フィールドは 1 つ以上のスペースで区切られます。

項目	説明
Name	このフィールドには、ルール・ファイルのレベル (スーパークラスまたはサブクラス) に対応するクラス・ファイル内に定義されているクラスの名前が入っていなければなりません。 クラス名には、大文字と小文字、数字、および下線のみを使うことができ、16 文字以内の長さにすることができます。 Unclassified、Unmanaged、および Shared のシステム定義クラスに対して割り当て規則を指定することはできません。

項目	説明
Reserved	このフィールドは今後の拡張に備えて予約されています。 その値はハイフン (-) でなければなりません。これは必須フィールドです。
User	このフィールドには、ハイフン (-) または少なくとも 1 つの有効なユーザー名 (/etc/passwd ファイルに定義されているもの) のいずれかを入れることができます。 このリストは、コンマ (,) で区切られた 1 つ以上の名前で構成されます。 名前の前に感嘆符 (!) を付ければ、特定のユーザーをクラスから除外することができます。 完全な Korn シェル・パターン・マッチング構文を使えば、一連のユーザー名に一致するパターンを指定することができます。 有効なユーザー名が 1 つもない場合、規則は無視されます。
Group	このフィールドには、ハイフン (-) または少なくとも 1 つの有効なグループ名 (/etc/group ファイルに定義されているもの) のいずれかを入れることができます。 このリストは、コンマ (,) で区切られた 1 つ以上の名前で構成されます。 名前の前に感嘆符 (!) を付ければ、特定のグループをクラスから除外することができます。 完全な Korn シェル・パターン・マッチング構文を使えば、一連のグループ名に一致するパターンを指定することができます。 有効なグループ名が 1 つもない場合、規則は無視されます。
Application	このフィールドには、ハイフン (-) を入れるか、 またはアプリケーション・パス名のリストを入れることができます。 これは、クラスに組み込まれているプロセスによって実行されるアプリケーション (プログラム) のパス名です。 アプリケーション名は、絶対パス名になるか、またはパス名に一致する Korn シェル・パターンになります。 このリストは、コンマ (,) で区切られた 1 つ以上のパス名で構成されます。 名前の前に感嘆符 (!) を付けると、特定のアプリケーションを除外することができます。 ロード時に、リストにあるアプリケーションが少なくとも 1 つ見つからなければなりません。見つからない場合、規則は無視されます。 この理由のために当初は無視された規則が後から有効になることがあります。 それは、このリストに含まれているアプリケーションを 1 つ以上含んでいるファイルシステムがマウントされた場合です。
Type	このフィールドには、ハイフン (-) を入れるか、またはプロセス属性のリストを入れることができます。 この属性に使える値は次のとおりです。 • **32bit**: プロセスが 32 ビット・プロセスの場合 • **64bit**: プロセスが 64 ビット・プロセスの場合 • **plock**: メモリーをピンするのに **plock** サブルーチンをプロセスから呼び出した場合 • **fixed**: プロセスが固定優先順位プロセス (SCHED_FIFO または SCHED_RR) の場合 fixed タイプはプランニング目的にのみ記述されています。 WLM は、固定優先順位のプロセスまたはスレッドによるプロセッサーの使用は 規制しません。 そのため、固定優先順位プロセスがクラス内の他のプロセスを 侵害する可能性があるため、これらのジョブを分離 できるようにこの分類属性が提供されています。 この属性は、また、このような プロセスの使用量について報告するためにも使用できます。 type フィールドの値は、上記の属性の 1 つ以上をプラス記号 (+) で区切って組み合わせることができます。 32bit と 64bit は、排他的であり、一緒に指定することはできません。
Tag	このフィールドには、ハイフン (-) を入れるか、またはアプリケーション・タグのリストを入れることができます。 アプリケーション・タグは、30 字以内の英数字の文字列です。 このリストは、コンマで区切られた 1 つ以上のアプリケーション・タグで構成されます。

User、Group、Application、および Tag 属性は、属性値のグループであることもあります。

プロセスを作成する (fork する) と、そのプロセスは親と同じクラス内にとどまります。 分類に使われるプロセス属性のうちのいずれかを変更する可能性のあるシステム・コールをその新規プロセスが発行すると、再分類が行われます。そのような呼び出しには、*exec*、*setuid* (およびこれに関連した呼び出し)、*setgid* (およびこれに関連した呼び出し)、*setpri*、および *plock* があります。

WLM ではプロセスを分類するために、トップレベルのルール・ファイルが調べられて、アクティブな構成が確かめられ、どのスーパークラスにそのプロセスが属すればよいかが判断されます。 ファイル内の各規則ごとに、その規則内に指定されている値と値リストに突き合わせて、プロセス属性の現行値が調べられます。 規則は、ファイル内に現れた順番どおりに調べられます。 一致するものが見つかると、規則の最初のフィールドに示されているスーパークラスにそのプロセスが割り当てられます。 次に同じやり方でスーパークラスのルール・ファイルが調べられ、プロセスをどのサブクラスに割り当てればよいかが判断されます。

プロセスがいずれかの規則に一致するには、その属性のおのおのが、その規則内の対応フィールドに一致しなければなりません。 以下に、ルール・ファイルのフィールド内の値に属性値が一致するかどうかを判断するのに使われる基準のリストを示してあります。

- ルール・ファイル内のフィールドにハイフン (-) の値が入っていると、それに対応するプロセス属性のどの値にも一致します。
- *type* 以外のすべての属性の場合、プロセス属性の値が、ルール・ファイルのリスト内の値のうちの 1 つ (ただし、除外されていない、すなわち、! が前に付いていない値) に一致すれば、一致したことになります。
- *type* 属性の場合、規則内の値のうちの 1 つが、プラス記号 (+) で区切られた複数の値で構成されていれば、その特性が上記のすべての値に一致する場合にのみプロセスに一致します。

スーパークラスおよびサブクラスの 両方のレベルで、ルール・ファイルに現れた順序で規則が 1 つずつたどられ、プロセスが一致した最初の規則に対応するクラス内のプロセスが分類されます。 したがって、ルール・ファイル内の規則の順序はきわめて重要です。 ルール・ファイルを作成または変更する場合は慎重に行うようにしてください。

ワークロード・マネージャーにおける手動クラス割り当て

プロセスまたはプロセスのグループは、SMIT、または **wlmassign** コマンドを使用することにより、スーパークラスまたはサブクラス (あるいはその両方) に手動で割り当てることができます。

詳しくは、 **wlmassign** を参照してください。 アプリケーションは API 関数 **wlm_assign** を使用してプロセスを割り当てることができます。

クラスにプロセスを手動で割り当てたり、既存の手動割り当てを取り消したりするには、 ユーザーに適切なレベルの特権がなければなりません。 手動割り当ては、スーパークラス・レベルまたはサブクラス・レベルのいずれか、あるいはこの両方で割り当てたり取り消したりできます。 この割り当ては、プログラミング・インターフェースのフラグと 、WLM 管理ツールで使用されるコマンド・ライン・インターフェースの一連のオプションによって指定します。 したがって、プロセスを手動で割り当てることができるのは、スーパークラスのみ、サブクラスのみ、またはスーパークラスとそのスーパークラスのサブクラスになります。 後者のケースでは、同時に (単一コマンドまたは API 呼び出しを使用して) あるいは別の時に異なるユーザーによって割り当てが二重になされる可能性があります。

割り当ては非常に柔軟に行えますが、混乱を招く可能性もあります。 以下の 2 つの例で、考えられるケースを考慮します。

例 1: プロセスの最初の割り当て

システム管理者は、*Process1* を *superclassA* から *superclassB* へ手動で割り当てます (スーパークラス・レベルのみの割り当て)。 WLM は *superclassB* のサブクラスについての自動割り当て規則を使用し、プロセスが最終的に割り当てられるサブクラスを判別します。 *Process1* は *superclassB.subclassA* に割り当てられ、「superclass only」割り当てのあるものとしてフラグが付けられます。

適切な特権を持つユーザーは、*Process2* を現在のクラス *superclassA.subclassA* から、同じスーパークラスの新しいサブクラス、*superclassA.subclassB* に割り当てます。 *Process2* は、その新しいサブクラスに割り当てられ、「subclass only」割り当てのあるものとしてフラグが付けられます。

superclassB のサブクラスの WLM 管理者は、*Process1* を *superclassB* の別のサブクラス、 *subclassC* に手動で割り当て直します。 *Process1* は *superclassB.subclassC* に分類され、スーパークラスおよびサブクラス・レベルの両方の割り当てがあるものとしてフラグが付きます。

例 2: 手動割り当ての再割り当てまたは取り消し

サブクラス・レベルでの手動割り当ての再割り当ておよび取り消しは、あまり複雑ではなく、 影響はそのサブクラス・レベルの割り当てに限定されます。

システム管理者が *Process2* をリソースを追加したスーパークラスに入れたいと思っており、*Process2* を手動で *superclassC* に割り振ることにしたとします。 例 1 で、*Process2* では、 *superclassA* の *subclassB* に、「subclass only」割り当てを指定して手動で割り当てられています。 *Process2* は異なるスーパークラスに割り当てられるため、以前の手動割り当ては無意味になり、取り消されます。 これで *Process2* は、*superclassC* に 対して「superclass only」で手動割り当てがされています。継承は行われないので、自動割り当て規則が使用されて、*superclassC* のサブクラスに割り当てられます。

ここで、システム管理者は、*Process1* から *superclassB* への手動割り当てを終了することを決めます。 *Process1* の「superclass level」での手動割り当てが取り消されます。 継承は行われないので、トップレベルの自動割り当て規則が使用されて、*Process1* にはスーパークラスが割り当てられます。

規則が変更されていない場合、*Process1* は *superclassA* に割り当てられ、*superclassB.subclassC* へのサブクラス・レベルの手動割り当ては無意味になり取り消されます。

なんらかの理由でトップレベル規則が *Process1* を *superclassB* に割り当てる場合、 *superclassB.subclassC* へのスーパークラス・レベル割り当ては依然として有効です。 *Process1* には、これで「subclass only」手動割り当てがされました。

ワークロード・マネージャーへの更新

WLM を更新すると (**wlmcntrl -u** コマンドによる)、更新済み構成が分類規則の新しいセットをロードできます。

これが起こると、多くの場合、この新しい規則を使用してプロセスが再分類されます。 WLM は、手動で割り当てされたプロセス、または継承が可能になっているクラスのプロセスは再分類 しません。ただし、それらのプロセスのクラスが新しい構成に存在しない場合は除きます。

ワークロード・マネージャーのセキュリティーに関する考慮事項

クラスにプロセスを割り当てたり、以前の手動割り当てを取り消したりするには、プロセスおよびターゲット・クラスの両方でユーザーに権限がなければなりません。

これらの制約は以下の規則のように説明できます。

- root ユーザーは、任意のクラスに任意のプロセスを割り当てられます。
- 指定されるスーパークラスのサブクラスで 管理特権を持つユーザー (つまり、ユーザーまたはグループ名が、スーパークラスの属性 `adminuser` および `admingroup` で指定されるユーザーまたはグループ名に一致するユーザー) は、このスーパークラスのサブクラスの 1 つから、スーパークラスの 他のサブクラスに任意のプロセスを割り当て直すことができます。
- ユーザーは、自分自身のプロセス (同じ実ユーザーまたは実効ユーザー ID に関連付けられたプロセス) を、自身が手動割り当て特権を持っているサブクラスに手動で割り当てることができます (つまり、ユーザーまたはグループ名が、スーパークラスまたはサブクラスの 属性 `authuser` および `authgroup` で指定されるユーザーまたはグループ名に一致します)。

手動割り当てを変更したり終了したりするには、ユーザーは、少なくとも最後に手動割り当てを発行した人と同じレベルの特権を持っていなければなりません。

ワークロード・マネージャーによるリソース管理

WLM は、システム上でアクティブなスレッドとプロセスによるリソース 使用率を、クラスごとにモニターして調整します。 WLM で管理するそれぞれのリソース・タイプのクラス別に、 最小または最大の制限値を設定することができます。 さらに、クラス別に、それぞれのリソースごとにターゲット値を設定することもできます。

このターゲットは、該当クラスのジョブに最適なリソースの量を表します。 スーパークラス・レベルの Share および制限は、システムで使える各リソースの合計量を指します。 サブクラス・レベルでは、Share

および制限は、 サブクラスが属するスーパークラス (スーパークラス・ターゲット) で使用可能な各リソースの量を指します。 クラス階層は、システム・リソースを複数のユーザー・グループ (スーパークラス) に分割して、そのリソース Share の管理をスーパークラス管理者に代行させるための手段です。 その後、各スーパークラス管理者は、サブクラスを作成してから、 それらのサブクラスに対するリソース権利を定義すれば、 該当量のリソースをグループ内のそれぞれのユーザーに再分配することができます。

ワークロード・マネージャーにおけるリソース・タイプ
WLM は、次の 3 つのタイプのリソースを、その使用量のパーセントを基にして管理します。

項目	説明
クラス内のスレッドの CPU 使用状況	これは、クラス内の各スレッドが使用するすべての CPU サイクルの 合計になります。
クラス内のプロセスの物理メモリー使用状況	これは、クラス内のプロセスに属するすべてのメモリー・ページの合計になります。
クラスのディスク入出力帯域幅	これは、クラスからアクセスされる各ディスク・デバイス上のクラス内のスレッドが実行するすべての入出力の帯域幅 (512 バイト・ブロック/秒の単位)。

WLM は 1 秒ごとに、直前の 1 秒間におけるそれぞれのリソースのクラス別使用率を、次のように、使用可能な合計リソースのパーセントとして計算します。

- CPU の場合、毎秒使える合計 CPU 時間は、1 秒にシステムの CPU 数を乗算したものになります。 例えば 8 way SMP で、直前の 1 秒間にあるクラスのすべてのスレッドで費やされた CPU 時間が 2 秒であったとすると、それは 2/8 = 25% を表します。 WLM が調整に使用するパーセントは、このような「瞬間的な」秒当たりのリソース使用率の数秒間にわたる減衰平均値です。
- 物理メモリーの場合、任意の時点で処理に使える物理メモリーの合計量は、 システム上に物理的に存在するメモリー・ページの合計数から、埋め込みページ数を引いた数になります。 埋め込みページは WLM によって管理されません。 そのようなページは、メモリー使用率を調整するためにクラスから取り出して別のクラスに与えることができないからです。 クラスのメモリー使用率とは、システムで使用できるページ数に対する、 クラス内のすべてのプロセスによって所有されている非埋め込みメモリー・ページ数をパーセントで表した比率のことです。
- ディスク入出力の場合の主な問題は、デバイスで、 意味のある使用可能な帯域幅を判別することにあります。 ディスクが 100% 使用中であると、複数のアプリケーションがランダム 入出力を行っている場合よりも、1 つのアプリケーションが順次入出力を行っているほうが、秒あたりのブロック数単位のそのスループットは大きく変わります。 順次入出力の場合に測定された最大スループットを (デバイスで使用可能な入出力帯域幅の値として) 使って、ランダム入出力のもとでのデバイスの使用率パーセントを計算すると、そのデバイスは実際には 100% の使用率のときでも、20% の使用率しかないように見えることがあります。

クラス別のディスク使用率のより正確かつ信頼できるパーセントを得るために、WLM では、ディスク・ドライバーによって示される統計情報 (**iostat** コマンドで表示されます) が使われます。その場合、直前の 1 秒間にデバイスが使用中になっていた時間のパーセントが、それぞれのディスク・デバイスごとに示されます。 WLM では、デバイスにアクセスしたすべてのクラスが 直前の 1 秒間にそのデバイスで読み書きした合計ブロック数がカウントされ、また、各クラスによっていくつのブロックが読み書きされたかと、デバイスの使用パーセントはどのくらいかもカウントされます。 次に WLM は、各クラスが費やしたディスク・スループットのパーセントを計算します。

例えば直前の 1 秒間に読み書きされた合計ブロック数が 1000 であって、デバイスは 70% 使用中になっていたとすると、100 個のブロックを読み取りおよび書き込みしたクラスは、ディスク帯域幅の 7% を使用したことになります。 CPU 時間 (別の交換可能リソース) と同様に、WLM がディスク入出力の調整に使用する値も、秒あたりのパーセントの数秒間にわたる減衰平均値です。

ディスク入出力リソースの場合、クラスから個別にアクセスされる各ディスク・デバイスに Share および制限が適用されます。 調整は各デバイスごとにそれぞれ単独で行われます。 つまり、クラスがある 1 つのデバイスでその権利が終了して、そのデバイスでの入出力が調整される一方で、別のディスクではその権利がまだ有効であり、この別のデバイスでの入出力は制約を受けないということです。

WLM は、合計使用量を基にしたリソースのアカウンティングと 規定をサポートします。 この方法で規制できるリソースには、クラス合計とプロセス合計という 2 つのタイプがあります。

クラス合計
クラス別制限は、クラス内のプロセス数、スレッド数、およびログイン・セッション数について指定できます。 これらの制限は、クラス内に任意の時点に存在する可能性のある各リソースの絶対数として指定します。 これらの制限は厳密に適用されます。クラスが上記のいずれかのリソースの制限に達した場合、そのリソースの別のインスタンスを作成しようとしても失敗します。 そのクラスがそのリソースについて指定されている制限を下回るまで、そのクラスのどのプロセスで実行しても、操作は引き続き失敗します。

プロセス合計
プロセス別制限は、CPU 時間、ディスク入出力のブロック、および 1 回のログイン・セッションでの接続時間の総量について指定できます。 これらの制限はクラス・レベルで指定されますが、クラス内の各プロセスに個別に適用されます (それぞれのプロセスがこの量を使用できるということです)。 これらの使用量値は累積されるため、プロセスがそのライフタイムの 間に使用したそれぞれの特定のリソースの総量を表します。 リソースのいずれかについてプロセスが指定された制限合計を超えると、そのプロセスは終了させられます。 このプロセスには SIGTERM シグナルが送られますが、プロセスがこのシグナルを受け取った後 5 秒間の猶予期間が経過しても終了しなかった場合には SIGKILL シグナルが送られます。 ログイン・セッションが指定された接続時間制限の 90% に達すると、警告メッセージが制御端末装置に書き出され、そのセッションがすぐに終了するという警告がユーザーに通知されます。

ワークロード・マネージャーにおけるターゲット Share
クラスのターゲット (または必要な) リソース使用量のパーセントは、 そのクラスが特定のリソースに対して持っている Share の数によって決定されます。

Share は、あるクラスがその Tier 内のそれ以外のクラスを基準にした場合の、獲得する特定のリソースの量を表します。 特定のリソースについてのあるクラスのターゲット・パーセントは、そのクラスの Share 数を Tier 内のアクティブ Share 数で割っただけのものです。 同時に制限も使用されている場合は、ターゲットは [minimum, soft maximum] の範囲に制限されます。 計算されたターゲットがこの範囲外であった場合は、ターゲットは適切な上限値/下限値に設定されます (『リソース制限』を参照してください)。 アクティブ Share 数は、中に少なくとも 1 つのアクティブ・プロセスを持っているすべてのクラスの Share を合計した数です。 アクティブ Share 数は動的であるため、ターゲットも動的です。 あるクラスが Tier 内での唯一のアクティブ・クラスである場合は、ターゲットはその Tier で使用可能なリソース量の 100% になります。

例えば、Tier 0?A には 3 つのアクティブ・スーパークラス 0—A、B、 および C—with が属していて、ある特定のリソースについてのそれぞれの Share が 15、10、および 5 であったとすると、 ターゲットは次のようになります。

ターゲット (A) = 15/30 = 50%
ターゲット (B) = 10/30 = 33%
ターゲット (C) = 5/30 = 17%

しばらくしてクラス B が非アクティブ (アクティブなプロセスがない) 状態になった場合、クラス A とクラス C のターゲットは自動的に次のように調整されます。

ターゲット (A) = 15/20 = 75%
ターゲット (C) = 5/20 = 25%

これで分かるように、Share は自己調整されるパーセントを表すものです。 Share を使用すれば、リソースがクラスに平等に分配されるように、あるクラスがアクティブになったときにはそれ以外のクラスからそのクラスに、非アクティブになったときにはそのクラスからそれ以外のクラスに、リソースが割り当てられるようにできます。

高い柔軟性を持たせるために、クラスに対する Share 数は 1 から 65535 の間の任意の数にできます。 Share はスーパークラスにもサブクラスにも 指定できます。 スーパークラスの 場合は、Share は同じ Tier 内の他のすべてのアクティブ・スーパークラスを基準にしたものになります。 サブクラスの場合は、その Share は同じ Tier 内の同じスーパークラス内の他のすべてのアクティブ・サブクラスを基準にしたものになります。 あるスーパークラスに属すサブクラスの Share は、別のスーパークラスに属すサブクラスの Share とは無関係です。

場合によっては、あるクラスのターゲットをアクティブ Share の数とは無関係にした方がよいことがあります。 Share の数として値「-」を指定すれば、そのようにできます。 この場合、そのクラスは当該リソー

スに関しては規制から外されます。これは、そのクラスは Share を持たず、そのターゲットはアクティブ Share の数に依存しないことを意味します。 そのターゲットは「tier が使用可能なリソース - その Tier 内の他のすべてのクラスの mins の合計」に設定されます。 すると、このターゲットまたは実際の使用量 (そのどちらか少ない方) が同じ Tier 内のその他のクラスの使用可能量から差し引かれます。

例えば、ある特定のリソースについてのクラス A、B、C、および D の Share がそれぞれ「-」、200、 150、および 100 であるとします。 すべてのクラスがアクティブで、クラス A がそのリソースの 50% を使用している場合は、次のようになります。

ターゲット (A) = 規制なし = 100%
ターゲット (B) = 200/450 * 使用可能量 = 44% * 50% = 22%
ターゲット (C) = 150/450 * 使用可能量 = 33% * 50% = 17%
ターゲット (D) = 100/450 * 使用可能量 = 22% * 50% = 11%

クラス A が規制なしで、使用可能リソースの 50% を使用しているため、 それ以外のクラスに使用可能な分は 50% だけです。 そのため、それらのクラスのターゲットはこのパーセントを基にして計算されます。クラス A は常にそのターゲット (100%) を下回ることになるので、 それぞれのターゲットと同じかまたはそれを上回る、同じ Tier 内の他のすべてのクラスより常に高い優先順位を持つことになります (詳細については、160 ページの『ワークロード・マネージャーにおけるクラス優先順位』を参照してください)。

注 : あるリソースについてある特定のクラスを規制外にするということは、そのクラスを上位の Tier に入れることと同じではありません。 下にリストされている動作は、(同じ Tier において) 規格外クラスにあてはまりますが、より上位の Tier にクラスが置かれている場合はあてはまりません。

- Share はリソース単位で定義されるため、クラスを 1 つ以上のリソースの場合は規制から外し、その他のリソースの場合は規制することができます。
- 同じ Tier 内のその他のクラスの最小値は、順守されます。 上位の Tier は、下位の Tier 内で指定されている最小値は守りません。
- Share を持つクラスに最小限度が指定されていなかったとしても、 規制外クラスの使用量は Share を持つクラスにある程度依存します。 なぜなら、規制外クラスは、その Tier で使用可能なリソースの一部については競合しているからです。 ある特定のワークロードではどのように動作するのかを調べるには、なんらかの実験を行う必要があります。

Share の数が指定されていない場合は、デフォルト値の「-」が使用され、 そのクラスは当該リソースについては規制外となります。 WLM の最初のバージョンでは、未指定の場合のデフォルト Share 値は 1 であったことに注意してください。

Share は、すべてのリソース・タイプについて、クラス別に指定します。 Share は **share** ファイルのスタンザ内に指定します。 例:

```
shares
classname:
   CPU      =   2
   memory   =   4
   diskIO   =   3
```

ワークロード・マネージャーにおけるリソース制限の指定

相対的なリソース権を定義するための Share の使用に加え、WLM には個々のクラスに対してリソース制限を指定するための機能が用意されています。 リソース制限を使用すると、管理者がリソース割り当てをより細かく制御できるようになります。 これらの制限は、パーセントとして指定され、そのクラスが入っている Tier で使用可能なリソースを基準とした量を指定します。

パーセント・ベースの規定には、次の 3 つのタイプの制限があります。

最小
これは、クラスに対して使用可能にしなければならない最小リソース量です。 実際のクラス使用量がこの値を下回る場合は、 そのクラスにはそのリソースに対する最高優先順位のアクセス権が与えられます。 使用できる値は 0 から 100 で、0 がデフォルトです (未指定の場合)。

ソフト最大
これは、リソースに対する競合がある場合にクラスが使用できる最大リソース量です。 クラス使用量がこの値を超えた場合は、そのクラスには Tier 内での最低の優先順位が与えられます。 同じ Tier 内の

他のクラスとの間にリソースに対する競合がない場合は、 そのクラスには必要なだけ使用することが許されます。 使用できる値は 1 から 100 で、100 がデフォルトです (未指定の場合)。

ハード最大
これは、競合がまったくない場合にクラスが使用できる、最大リソース量です。 クラスがこの制限に達した場合は、それ以降、 その使用量のパーセントがこの制限を下回るまで、 このクラスはそのリソースを使用できなくなります。 使用できる値は 1 から 100 で、100 がデフォルトです (未指定の場合)。

リソース制限値は、それぞれのクラスのスタンザ内のリソース・タイプ別に、 リソース制限値ファイルの中に指定されます。 これらの制限値は、最小値からソフト最大値の範囲として指定します。 この場合、ハイフンで区切り、ホワイト・スペースは無視します。 ハード最大値は、指定される場合には、ソフト最大値の後に続き、おのおのがセミコロン (;) で区切られます。 それぞれの制限値の直後に、パーセント記号 (%) を付けます。

以下に、ルール・ファイルの使用例を示してあります。

- グループ acct3 のユーザー joe が `/bin/vi` を実行すると、プロセスはスーパークラス acctg に入れられます。
- グループ dev のユーザー sue が `/bin/emacs` を実行すると、そのプロセスはスーパークラス devlt (グループ ID が一致します) に入れられますが、editors サブクラスには分類されません。ユーザー sue はそのクラスから除外されているからです。 このプロセスは devlt に入れられます。 デフォルト
- DB 管理者が `oracle` というユーザー ID と `dbm` というグループ ID を使用して、データベース DB1 にサービスを提供するために `/usr/sbin/oracle` を開始すると、そのプロセスはデフォルトのスーパークラスに分類されます。 そのプロセスは、そのタグを `_DB1` に設定してはじめて、スーパークラス db1 に割り当てられます。

制限値は、すべてのリソース・タイプについて、クラス別に `limits` ファイルのスタンザ内に指定します。次に例を示します。

```
shares
classname:
   CPU     =   0%-50%;80%
   memory  =   10%-30%;50%
```

この例では、ディスク入出力には制限が設定されていません。 システム・デフォルトを使用すると、これは次のように変換されます。

```
diskIO   =   0%-100%;100%
```

上記の例ではすべて、記載されているスーパークラスおよびサブクラスでは 継承属性は yes に設定されていないと仮定しています。 そうでない場合、新規のプロセスは単にその親からスーパークラスまたはサブクラスを継承します。

WLM では、リソース制限値に関して次の制約だけが設けられます。

- 下限値は、ソフト最大値より小か等しくなければなりません。
- ソフト最大値は、ハード最大値より小か等しくなければなりません。
- Tier 内のすべてのスーパークラスの 最小の合計は 100 を超えてはなりません。
- Tier 内のある特定のスーパークラスのすべてのサブクラスの 最小の合計は 100 を超えてはなりません。

ハード・メモリー制限を設定されたクラスがその制限値に達し、さらにページを要求するときは、VMM ページ置換アルゴリズム (LRU) が開始され、制限値に達したクラスからページを「スチールする」ので、 そのページ数は、新しいページが提供される前にハード最大値よりも少なくなります。 これは正しい動作ですが、多くの空きページが使用可能な状態であっても行われることがある余分なページング・アクティビティーによって、 システムの一般パフォーマンスが影響を受けます。 どのクラスに対しても、ハード・メモリーの最大値を強制する前に、他のクラスのメモリー最小値を使用することをお勧めします。

最小値を下回っているクラスは Tier 内での最高の優先順位を持つため、最小値の合計は、同じ Tier 内のそれ以外のクラスのリソース所要量に基づいて、妥当なレベルに保つ必要があります。

Tier 内の下限値の合計は 100 以下でなければならないという制約は、最高優先順位の Tier 内のクラスは、その下限値以内のリソースをいつでも獲得できることを意味します。 WLM では、クラスが実際に下限値に

達するとは限りません。 これは、クラス内のプロセスがそれぞれのリソースを使用する方法、ならびに、そのとき有効なその他の制限に応じて異なります。 例えば、クラスは、十分なメモリーを取得することができないために、その最小の CPU 使用権を得られない場合があります。

物理メモリーで、メモリーの制限を設定すると、クラス内のプロセスのメモリー・ページに対して何らかの保護が設けられます (少なくとも、最高優先順位の Tier 内のクラスの場合は)。 アクティブなすべてのクラスがそれぞれの最小値を下回っていて、しかもそのクラスのうちの 1 つがさらにページを要求しない限り、その最小値を下回っているクラスからページをスチールしてはなりません。 最高位の Tier 内のクラスがその最小値を下回っているときに、 クラスからページをスチールしてはなりません。 したがって、対話式ジョブのクラスにメモリーの最小値を設定すれば、連続したアクティブ化の間にページがすべてスチールされることがないようにする (メモリーが不足している場合でも) ことができ、それによって応答時間が改善されます。

重要 : ハード最大値を使用する場合は、適切に使用しないと、 システムまたはアプリケーションのパフォーマンスに多大な影響を与える可能性があります。 ハード制限を課すと、結果的にシステム・リソースが使用されない場合があるため、 ほとんどの場合ではソフト最大値の方がより適切です。 ハード最大値の 1 つの用途として、上位 Tier の使用量を制限してリソースの一部を下位の Tier が使用できるようにすることが考えられますが、 リソースを必要とするアプリケーションを上位の Tier に入れる方がよりよい解決策です。

制限合計をしきい値ファイルで指定できます。このファイルでの値と単位の要約を以下の表に示します。

表 *8.* ワークロード・マネージャーのリソース制限

リソース	許可される単位	デフォルトの単位	最大値	最小値
totalCPU	s、m、h、d、w	s	2^{30} – 1s	10 s
totalDiskIO	KB、MB、TB、PB、EB	KB	(2^{63} – 1) * 512/1024 KB	1 MB
totalConnectTime	s、m、h、d、w	s	2^{63} – 1 s	5 m
totalProcesses	–	–	2^{63} – 1	2
totalThreads	–	–	2^{63} – 1	2
totalLogins	–	–	2^{63} – 1	1

注 : 単位指定子には大文字小文字の区別はありません。 s = 秒、m = 分、h = 時間、d = 日、w = 週、KB = キロバイト、MK = メガバイトなど。

制限スタンザの例は、以下のとおりです。

```
BadUserClass:
     totalCPU = 1m
     totalConnectTime = 1h
```

制限合計は、上の表の任意の値を使用して指定できますが、次の制限があります。

- 指定する場合、totalThreads の値は最小でも totalProcesses の値でなければならない。
- totalThreads を指定して totalProcesses を指定しないと、totalProcesses に対する制限は totalThreads の値に設定されます。

制限合計は、スーパークラス・レベルおよびサブクラス・レベルで指定できます。 制限の検査時には、サブクラスの制限の方がスーパークラスの制限より前に検査されます。 この 2 つの制限が両方とも指定されている場合は、どちらか値の小さい方が適用されます。 指定されているサブクラスの制限が、関連付けられているスーパークラスの 制限より大きい場合は、構成のロード時に警告が発行されますが、ロードは行われます。 このことは、クラス制限合計に関して重大な意味を持ちます。なぜなら、この制限は 絶対値である (スーパークラスを基準としたものではない) ため、あるスーパークラスで 使用可能なリソースを 1 つのサブクラスがすべて使用してしまう可能性があるからです。 指定されないと、すべての制限合計のデフォルト値は「-」(制限なしの意味) です。 デフォルトでは、クラスおよびプロセスのアカウンティングと規定は、WLM の実行中は使用可能にされます。 **wlmcntrl** コマンドの **-T [class|proc]** オプションを使用すると、合計のアカウンティングと規定を使用不可にできます。

ワークロード・マネージャーにおけるクラス優先順位

WLM は、各クラスにそれぞれのリソースごとの優先順位を与えたうえで、クラスにリソースを割り当てます。

クラスの優先順位は動的なもので、そのクラスの Tier、Share、制限、および現在の使用量に基づいています。 最高の優先順位を持つクラスには、いつでも優先的にリソースへのアクセスが許可されます。 最高のレベルでは、Tier はクラス優先順位の互いに重複しない範囲を表します。 Tier 0 のクラスは、ハード最大値を超えない限り、常に Tier 1 のクラスより高い優先順位を持つことを保証されます。

クラス優先順位を決定する際に、WLM は以下の優先順位で制約を行います (最初のものが最高で、下に行くほど優先順位が下がります)。

ハード制限
クラス使用量があるリソースのハード最大値を超えた場合、 そのクラスにはそのリソースに関しては可能な範囲内での最低の優先順位が与えられ、 その使用量がこの制限を下回るまでアクセスを拒否されます。

Tier
ハード制限がない場合、クラスの優先順位は、そのクラスが属す Tier に許可される最低優先順位から最高優先順位までの範囲に限られます。

ソフト制限
あるリソースに関し、クラス使用量がソフト最大値の最小値を下回っている場合、そのクラスにはその Tier 内で最高の優先順位が与えられます。 クラス使用量がソフト最大値を上回っている場合は、そのクラスにはその Tier 内で最低の優先順位が与えられます。

共有
これらは、それぞれのリソースごとにクラス使用量のターゲットを計算するのに使用されます。 クラスの優先順位は、クラス使用量がターゲットを下回るにつれて上がり、 ターゲットを上回るにつれて下がります。 ソフト制限の方が優先順位が高いため、クラスの優先順位はソフト制限に基づいて決定されることに注意してください。

Share と制限は両方とも、各クラスおよび各リソースに使用できますが、クラスに対してどちらか一方のみを使用した方が、容易に結果を予測できます。

排他使用のプロセッサー

排他使用のプロセッサー・リソース・セット (XRSET) によって、 管理者は、重要な作業のためのリソースを確保することができます。 XRSET は名前付きのリソース・セットで、その中に含まれているすべての CPU の動作を変更します。 一度 CPU が排他的と指定されると、その CPU は明示的にそこに送信されたプログラムだけを実行します。

XRSET の作成

XRSET を作成するには root ユーザーである必要があります。 **mkrset** コマンドを使用して sysxrset ネームスペース内に リソース・セットを作成します。 例えば、コマンド **mkrset -c 1-3 sysxrset/set1** では、CPU 1、2 および 3 用に XRSET が作成されます。また、rs_registername() サブルーチンを使用して XRSET を作成することもできます。

XRSET がシステムに定義済みであるかどうかの判別

lsrset -v -n sysxrset コマンドを使用すると、システムに定義されている すべての XRSET が表示されます。 (これを行うためのプログラミング API は現在のところありません。)

XRSET の削除

XRSET を削除するには root ユーザーである必要があります。 XRSET は **rmrset** コマンドで削除されます。 rs_discardname() サブルーチンによっても、XRSET を削除することができます。

システムのリブート

システムをリブートすると 、設定されていたすべての XRSET がレジストリーから除去され、どれも無効になります。

XRSET を使用する作業の指定

作業が排他使用プロセッサーを使うのに適格であるとマークする方法は、複数あり ます。 排他使用プロセッサーを含むリソース・セットを指定するには、**attachrset** と **execrset** のコマンドが使用できます。 排他使用プロセッサーを含んでいるリソース・セットを WLM クラスに関連付ける ことができます。 このような WLM クラスに分類された作業は、リソース・セット内に 指定された排他使用プロセッサーを使用します。

Bindprocessor および _system_configuration.ncpus での XRSET の使用

作業を排他使用プロセッサーで実行させるために bindprocessor を使用することはできません。 リソース・セット・ベースで接続した場合のみ、作業を排他使用プロセッサーで 実行させることができます。

システム構成における CPU の数 (**_system_configuration.ncpus** フィールド) は、XRSET が作成されても変わりません。 システムには依然として NCPU が存在します。

プログラムで NCPU に対する bindprocessor システム・コールを行うと、XRSET 内の CPU は EINVAL エラーになって失敗します。 **bindprocessor** コマンドの照会オプションで戻される ID は、どれもバインドできます。 照会オプション (bindprocessor -q) は、有効なバインド ID のみを戻し、排他的 CPU に関連付けられている CPU は戻しません。

例えば、システムで 10 CPU がオンラインになっていて、その中の 3 つが XRSET に入っているとすると、0 から 6 までのバインド ID で CPU を bindprocessor にすることは成功します。 7 から 9 のバインド ID での CPU の bindprocessor には、EINVAL エラーが戻されます。

動的 CPU 再構成操作での XRSET の使用

一般に、動的 CPU 再構成は排他使用プロセッサーによる影響を受けません。 ただし、XRSET を作成したり、それらのプロセッサーへ作業を割り当てすると、CPU を 除去できなくなることがあります。 システムに動的に追加された CPU は、汎用プロセッサーまたは 排他使用プロセッサーのいずれかとしてシステムに参加します。 動的に追加された CPU が排他使用としてシステムに入るのは、システムに 参加するときに、その論理 CPU ID を含む XRSET がある場合です。

ワークロード・マネージャーにおけるリソース・セット

WLM はリソース・セット (つまり *rsets*) を使用して、ある特定のクラスのプロセスが使用できるシステムの物理リソースを、 あるサブセットだけに制限します。 WLM では、管理される物理リソースは、メモリーとプロセッサーです。 有効なリソース・セットは、メモリーと少なくとも 1 つのプロセッサーから構成されます。

システム管理者は、SMIT を使用して、システム上で使用可能なリソースのサブセットが含まれているリソース・セットを定義し、 それに名前を付けることができます。 次に、WLM 管理インターフェースを 使用して、root ユーザーまたは指定されたスーパークラス 管理者はそのリソース・セットの名前を WLM クラスの rset 属性として使用できるようになります。 それ以降、この WLM クラスに割り当てられているすべてのプロセスは、 そのリソース・セットのプロセッサーのいずれかに対してのみディスパッチされるようになり、CPU リソースのワークロードが効率よく分けられます。

現在のすべてのシステムでは、メモリー・ドメインは 1 つだけで、それをすべてのリソース・セットが共用しているため、この方法によってメモリー内のワークロードが物理的に分けられることはありません。

ワークロード・マネージャーにおけるリソース・セット・レジストリー

rset レジストリー・サービスを使用すると、 システム管理者はリソース・セットを定義して名前を付け、それらを他のユーザーまたはアプリケーションが使用できるようにできます。

名前の衝突が起こるリスクを緩和するために、レジストリーは 2 レベルの命名方式をサポートしています。 リソース・セットの名前は、*name_space*/*rset_name* という形式になります。 *name_space* および *rset_name* は、それぞれ 255 文字までの長さが可能で、両方とも大文字小文字が区別され、 使用できるのは英大文字と小文字、数字、下線、およびピリオド (.) だけです。 **sys** の *name_space* は、オペレーティング・システムによって 予約済みであり、システムのリソースを 表す rset の定義に使用されます。

rset 定義名は、レジストリー・ネーム・スペース内では固有です。 既存の rset 定義と同じ名前を使用して新しい rset 定義をレジストリーに追加すると、既存の定義がその新しい定義で置き換えられて、 適切

な許可と特権が与えられます。 SMIT を使用してリソース・セットを作成、変更、および削除し、 メモリー内の rset データベースを更新できるのは root だけです。

各 rset 定義は、それに関連付けられているオーナー (ユーザー ID)、グループ (グループ ID)、およびアクセス権を持っています。 これらは rset 定義の作成時に指定され、アクセス制御のために存在します。 ファイルの場合と同様に、読み取りおよび/または書き込み許可が認可されているかどうかを定義する、 オーナー、グループ、およびその他のものに対する別々のアクセス権が存在します。 読み取り許可があると、rset 定義を検索できます。 一方、書き込み許可があると、rset 定義を変更または除去できます。

システム管理者が定義した rset 定義は、/etc/rsets スタンザ・ファイル内に保持されます。このファイルのフォーマットの記述はありません。 そのため、このファイル・フォーマットが変更された場合に将来起こる可能性のある互換性の問題を未然に防ぐために、rset の操作は SMIT のインターフェースを 通じて行う必要があります。 WLM クラス定義の場合と同様に、rset 定義をカーネルのデータ構造にロードしてからでなければ、それらを WLM で使用することはできません。

ワークロード・マネージャーの設定

クラス定義、クラス属性、Share および制限、 および自動クラス割り当て規則を入力するには、SMIT、または WLM コマンド・ライン・インターフェースを 使用します。 これらの定義および規則は、やはりテキスト・エディターを使って作成または変更できるプレーン・テキスト・ファイル内に保持します。

これらのファイル (*WLM* 属性ファイル と呼ばれる) は、/etc/wlm のサブディレクトリーに保持されます。スーパークラスとそれに関連したサブクラスを記述した一連のファイルで、WLM 構成を定義します。 WLM **Config** 構成のファイルは /etc/wlm/Config 内にあります。 このディレクトリーには、スーパークラス用の WLM パラメーターの定義が入ります。 それらのファイルには、description、classes、shares、limits、および rules という名前が付けられます。 このディレクトリーにはまた、サブクラス定義を保管するスーパークラスの 名前のついたサブディレクトリーも 置かれることがあります。 例えば、WLM **Config** 構成のスーパークラス *Super* の場合、スーパークラス *Super* のサブクラス用の属性ファイルはディレクトリー /etc/wlm/Config/Super に入っています。 それらのファイルには、description、classes、 shares、 limits、および rules という名前が付けられます。

システム管理者によって WLM 構成が定義されたら、**smit wlmmanage** 高速パス、または **wlmcntrl** コマンドを使用して、それをアクティブ構成にできます。

属性ファイルの複数のセットを定義して、ワークロード・マネージメントのさまざまな構成を定義することができます。 通常、これらの構成は、/etc/wlm のサブディレクトリーの中に収められます。 シンボリック・リンク /etc/wlm/current は、現在の構成ファイルが収められているディレクトリーを指します。このリンクは、WLM が指定されたセットの構成ファイルを使用して開始されたときに、**wlmcntrl** コマンドによって更新されます。

ワークロード・マネージャー構成のアプリケーションの要件

構成定義の最初のフェーズでは、ユーザーやユーザーの計算に関するニーズについて理解し、 さらにシステム上のアプリケーションとそれらのリソースに 対するニーズ、およびビジネス上の要件に ついて理解している必要があります (例えば、どのタスクが重要で、どのタスクの優先順位を低くしてもよいかなど)。このような理解に基づいて、スーパークラスを定義し、次にサブクラスを定義します。

優先順位の設定は、WLM がご使用の編成で実行する機能に依存します。 サーバー統合の場合、読者は既にアプリケーション、ユーザー、およびリソースの要件を知っており、ステップの一部をスキップするか、あるいは短くすることができるでしょう。

WLM を使用すると、ユーザーまたはグループ、アプリケーション、タイプ、タグ、 あるいはこれらの属性の組み合わせでプロセスを分類することができます。 WLM ではクラス間のリソース使用率が調整されるので、 システム管理者は、リソース使用率のパターンが同じアプリケーションおよびユーザーを、 同じクラスにグループ化する必要があります。 例えば、通常は CPU 時間をほとんど消費しないが応答時間は高速にする必要がある対話式ジョブを、 通常は CPU およびメモリーを大量に消費するようなバッチ・タイプ・ジョブから分離したい場合があるかもしれません。 これは、作業量の大きいデータ・マイニングの照会から OLTP タイプのトラフィックを 分離する必要があるデータベース環境でも同じです。

このステップは、SMIT、またはコマンド・ライン・インターフェースを 使用して行います。 最初の何度かは、SMIT を使用するとよいでしょう。 これにより、スーパークラスの 定義およびその属性の設定を含む、最初の WLM 構成を作成するステップを実行できます。 最初のパスでは、属性の一部を設定し、その他の

部分はデフォルト値のままにしておくことができます。 これは、リソース Share および制限の場合も同じです。 これらのクラス特性はすべて、後で動的に変更できます。

その後、パッシブ・モードで WLM を始動し、ご使用の種別を確認し、 ご使用のアプリケーションのリソース使用率パターンを調べることができます。

wlmcheck コマンドを使用するか、対応する SMIT メニューを使用してご使用の構成を検証します。 次に、新しく定義した構成で、パッシブ・モードで WLM を始動します。 WLM は既存のすべてのプロセス (およびここで作成されるすべてのプロセス) を分類し、さまざまなクラスの CPU、メモリー、およびディスク入出力使用率の統計情報のコンパイルを開始します。 WLM は、リソース使用状況の調整は試みません。

各種プロセスが、システム管理者が期待する適切なクラスに分類されていることを確認します (**ps** コマンドの **-o** フラグを使用)。 プロセスの一部が望んだように分類されていない場合、割り当て規則を調整するか、 クラスの一部の継承ビットを設定して (新しいプロセスが親と同じクラスに入るようにしたい場合)、WLM を更新します。 この処理を繰り返して、第 1 レベルの種別 (スーパークラス) について満足するまで調整してください。

パッシブ・モードで WLM を実行し WLM をリフレッシュする (常にパッシブ・モード) ことによるリスクは低く、 オーバーヘッド演算も低く、通常のシステム操作を妨げることなく実動システムで安全に実行できます。 WLM をアクティブにし、リフレッシュするには、**wlmcntrl** コマンドをコマンド・ラインから、あるいは SMIT から呼び出して使用します。

パッシブ・モードで WLM を実行し、**wlmstat** コマンドを使用して統計情報を収集します。 **wlmstat** コマンドを一定の時間間隔で使用すると、使用可能なリソースの合計のパーセントで、 スーパークラスのクラスごとのリソース使用率を表示することができます。 これにより、長期間にわたってご使用のシステムをモニターでき、 メイン・アプリケーションのリソース使用効率を検討することができます。

ワークロード・マネージャーにおける Tier、Share、および制限

WLM をパッシブ・モードで実行して収集したデータとビジネス・ゴールを使用して、 それぞれのスーパークラスに与える Tier 番号、および各種のクラスに対するリソース別の Share の配分方法を決定します。

一部のクラスについては、最小値または最大値の定義が必要となる場合があります。 ご使用のリソース割り当てのゴールに到達するように、Share および Tier 数を調整してください。 Share だけでは解決できない場合に備えて制限を予約します。 また、サブクラスを追加する必要に迫られる場合もあります。

- 通常はリソース使用量は少ないものの、 外部イベントによりアクティブにされる 際には即時の応答時間が必要なアプリケーションには、 下限を使用します。 メモリーが少なくなっている状況で対話式ジョブを実行する際の問題として、 非アクティブ状態のときにページがスチールされることがあります。 メモリー最小限度は、クラスが Tier 0 の場合に、 対話式ジョブのページの一部を保護するのに使用できます。
- リソースを集中的に使用する優先順位の低いジョブを含めるには、上限を使用します。 他の理由でご使用のシステム・リソースを分割しないかぎり、ハード最大値は、 メモリーなどの更新できないリソースの大半に意味のあるものになります。 これは、高い優先順位のクラスで最初のクラスが使用したページが必要な場合に、 データをページング・スペースに書き出すのに必要な時間が原因です。 CPU 使用の場合、Tier およびソフト最大値を使用して、優先順位の高いクラスが即時に CPU 時間を割り当てられるかどうかを確認することができます。

サブクラスのパラメーターを作成して調整する際には、システムの動作が意図したものとなるまで、 ある特定のスーパークラスのサブクラス (他のスーパークラスのユーザーやアプリケーションに 影響を 与えないもの) のみを対象として WLM のリフレッシュを行えます。

また、ビジネスのニーズに応じて、異なるパラメーターを使用して別の構成も定義できます。 これを実行する際には、既存の構成をコピーしたり変更したりして時間を節約することができます。

ワークロード・マネージャー構成の微調整

wlmstat コマンドを使用してシステムをモニターし、WLM による規制がゴールに沿ったものかどうか、および一部のアプリケーションが必要以上にリソースを獲得している一方で、 不当にリソースを奪われているアプリケーションがないかどうかを検証します。 この場合、Share を調整し、WLM をリフレッシュします。

Share、制限、および Tier 番号をモニターして調整しながら、 一部またはすべてのスーパークラスについて、 サブクラスの管理を委任するかどうかを決定します。 次に、管理者はサブクラスの Share、制限、および Tier 数をモニターしてセットアップすることができます。

それぞれのサブクラスの管理者は、 それぞれのスーパークラスのサブクラスについてこのプロセスを 繰り返すことができます。 相違点は、パッシブ・モードでは、サブクラス・レベルのみで WLM を実行することはできないという点だけです。 サブクラス構成とチューニングは、WLM を使用してアクティブ・モードで実行しなければなりません。 スーパークラスのユーザーおよびアプリケーションに 影響を与えないようにする方法の 1 つには、サブクラスの Tier 数と、Share および制限にデフォルト値 (Share には '-' (ハイフン)、最小に 0%、ソフトおよびハード最大 (%) に 100%) を使用して開始することがあります。 これらの設定値を指定した場合、WLM がサブクラス間のリソース割り当てを調整することはありません。

ワークロード・マネージャーのトラブルシューティング

現在の構成で期待した動作が得られない場合は、WLM の構成の調整が必要になる可能性があります。

wlmstat コマンドを使用して、各クラスの消費量の値をモニターすることができます。 このデータを収集し、分析することにより、構成をどのように変更する必要があるかを判別することができます。 構成の更新後、**wlmcntrl -u** コマンドを使用して、アクティブな WLM の構成を更新します。

以下の指針を参考すると、構成の変更方法を決めることができます。

- ある tier のアクティブな share 数が時間の経過とともに大きく変動する場合は、1 つのクラスに対してリソースの share を与えないようにします。こうすると、クラスは、アクティブな share 数から独立した消費量ターゲットをもつことができます。 この手法は、リソースに対して高い優先順位のアクセスを必要とする重要なクラス に役立ちます。
- 特定量のリソースに対してアクセスを保証する必要がある場合は、最小の制限を指定します。 この手法は、それほど多くのリソースを消費しないが、外部イベントに対して素早く応答する必要のある、対話式ジョブに役立ちます。
- リソースに対するアクセスを制限する必要があるが、share では適切に制御できない場合は、最大の制限を指定します。 たいていの場合、ソフト最大制限を指定するのが適切な方法ですが、厳格な強制を行うために、ハード最大制限を使用することもできます。 ハード最大制限を指定すると、システム・リソースが 浪費される場合がありますし、また、メモリーの調節に使用するとページングの活動が 増加する場合もあります。このため、他のクラスに対して 最小制限を指定してからハードの最大制限を指定してください。
- あまり重要でないジョブがより重要なジョブに干渉する場合は、あまり重要でないジョブを低い tier に置きます。 この手法を使用すると、あまり重要でない ジョブが低い優先順位をもつことができるため、より重要なジョブの実行中に、使用可能なリソースを要求して競合しません。
- クラスがリソースの消費量ターゲットに到達しない場合は、別のリソースについての競合によってこの条件が引き起こされていないかどうかを検査します。 それが原因だった場合は、競合しているリソースに対するクラスの割り当てを 変更します。
- クラス内のプロセスの動作またはリソース消費量が大きく変動する場合は、より細かな制御ができるように、クラスの数をふやします。 また、重要なアプリケーション ごとに、クラスを分割することをお勧めします。
- 分析の結果、ある 1 つのクラスが必要とするリソース量が、別のクラスの消費量に依存することが分かった場合は、それに応じて、リソースの再割り当てを行います。 例えば、ClassZ が必要とするリソース量が、ClassA が処理できる作業要求数に依存す る場合は、ClassZ に必要なリソース量を提供するために、ClassA が十分なリソースにアクセスできるよう保証する必要があります。
- 1 つ以上のアプリケーションが、適切に動作できる十分なリソースを受け取れない 状態が続いている場合は、システムのワークロードを減らす以外に方法はありません。

注 : スーパークラスの *adminuser* を定義すると、WLM 管理者に必要とされる作業量を減らせます。 最高レベルの構成のテストと 調整が終わったら、特定の要求に合わせて、スーパークラス adminusers が 以後の変更 (サブクラスの作成および構成など) を行うことができます。

ワークロード・マネージャー API

アプリケーションは、/usr/lib/libwlm.a ライブラリー内の一連のルーチン、WLM API を使用して、WLM 管理者が WLM コマンド・ライン・インターフェースを 使用して実行できるすべてのタスクを実行できます。

サポートされるファイルシステムのタイプを 次に示します。

- クラスの作成、変更、または削除
- クラス属性またはリソース Share および制限の変更
- クラスの除去
- 手動によるプロセスのクラスへの割り当て
- WLM 統計情報の検索

アプリケーションは、API を使用して、タグと呼ばれるアプリケーション定義の分類属性を設定できます。システム管理者から提供される (アプリケーション・ユーザー・ドキュメンテーションを介して) 一連の値を使ってこのタグを設定すると、同じアプリケーションの複数のインスタンスを個々に区別することができます。 したがって、異なるクラスを異なるリソース権によって分類することができます。

さらに、**wlm_set_tag** ルーチンを使用すると、アプリケーションでアプリケーション・タグをセットアップし、**fork** または **exec** で子プロセスがこのタグを継承すべきかどうかを指定することができます。 スレッドは、アプリケーション・タグに **wlm_set_thread** タグを割り当てることもできます。 スレッドのアプリケーション・タグは、**fork**、**exec** または **pthread_create** サブルーチン間で継承できます。 このライブラリーは、32 ビットまたは 64 ビットのマルチスレッド・アプリケーションをサポートします。

アプリケーション・タグ
アプリケーション・タグは文字列で、プロセスまたはスレッドの自動分類の分類基準の 1 つとして使用されます (`rules` ファイルを使用)。 このタグは基本的に、*user*、*group*、*application*、および *type* などのシステム定義の基準に加えて、アプリケーション定義の分類基準を提供します。

アプリケーション・プロセスまたはスレッドがそのタグを設定すると、現在アクティブな WLM 構成で有効なスーパークラスおよびサブクラス 規則を使用して、即時に分類し直されます。 それから、WLM は、新しいタグを含むすべてのプロセス属性を使用して、一致するものを探す割り当て規則を検討します。

有効にするには、このタグは 1 つ以上の割り当て規則に表示されていなければなりません。 各種タグのフォーマットおよび使用方法を、アプリケーションごとに、当該アプリケーションの管理ドキュメンテーションに明確に指定する必要があります。これは、WLM 管理者がそのフォーマットおよび使用法を十分認識し、割り当て規則にあるタグのさまざまな値を使用して、同じアプリケーションの異なるインスタンスを識別できるようにするためです。

ユーザーによって、これらのプロセスを分類するために使用したいアプリケーション・プロセスの特性について、異なる要件を持っている可能性があるため、アプリケーションで、タグの作成に使用できる構成またはランタイム属性のセットを提供することをお勧めします。 アプリケーション管理者は、このタグのフォーマットをアプリケーションに 指定することができます。 タグ、および WLM タグのフォーマットを指定する構文に使用できる属性はアプリケーションに依存しており、属性の提供はアプリケーション・プロバイダーの役割です。

例えば、データベース・サーバーのインスタンスは、実行するデータベース (*db_name*) および 指定されるユーザーが接続する TCP ポート (*port_num*) を判別します。 管理者は、以下の優先順位のどれでも取得できます。

- 異なるデータベースにアクセスするプロセスについて異なるクラスを作成し、それぞれのクラスに異なるリソース資格を付与する。
- 異なる起点からのリモート要求を扱うプロセスを分離し、種別属性としてポート番号を使用する。
- それぞれのデータベースごとに 1 つのスーパークラスを 作成し、それぞれのスーパークラスにあるポート番号ごとにサブクラスを作成する。

これらの異なるニーズに応える方法の 1 つは、タグの内容およびフォーマットを指定することです。 この例では、タグを構成ファイルにあるアプリケーションか、あるいは `WLM_TAG=$db_name` や `WLM_TAG=$db_name_$port_num` などの ランタイム・パラメーターに渡せると想定しています。

アプリケーションはそのタグを設定する際に、タグを子が継承するどうかを指定して、アプリケーションの特定のインスタンスによって作成されるすべてのプロセスが、同じクラスに分類されるようにすることができます。 タグ継承の設定は、アプリケーション・タグの最も一般的な使用方法です。

以下は、アプリケーション・タグの使用方法の例です。 この例では、データベースのタグはデータベース名と同じになっています。 2 つの異なるデータベースで実行するサーバーの 2 つのインスタンスは、2 つの異なるタグ、例えば、db1 と db2 をセットアップします。

システム管理者は 2 つの異なるクラス dbserv1 および dbserv2 を作成し、タグを使用してこれらのクラスに 2 つのデータベース・サーバー (および、タグ継承が使用される場合にはそれらの子すべて) を分類できます。 そうすると、事業の目的にしたがって、それぞれのクラスに異なるリソース使用権利を付与することができるようになります。

対応する割り当て規則は以下に類似したものになります。

```
* class   resvd  user  group     application     type  tag
*
dbserv1     -    -     dbadm     /usr/sbin/dbserv  -   db1
dbserv2     -    -     dbadm     /usr/sbin/dbserv  -   db2
```

アプリケーション・プログラミング・インターフェース・タイプ
以下に、ワークロード・マネージャー (WLM) アプリケーション・プログラミング・インターフェース ・タイプを示します。

クラス管理 API

WLM API は、以下の機能を持つアプリケーションを提供します。

- 指定される WLM 構成の既存のクラス (wlm_read_classes) の名前および特性を照会する。
- 所定の WLM 構成に新規クラスを作成し、クラスの各種の属性 (Tier、Inheritance など) の値と、CPU、物理メモリー、ブロック入出力など、WLM によって管理されるリソースの Share および制限を定義する (wlm_create_class)。
- 指定される WLM 構成の既存のクラスの特性を変更する。これには、クラス属性およびリソース Share および制限を含みます (wlm_change_class)。
- 指定の構成の既存のクラスを削除する (wlm_delete_class)。

変更は、指定される WLM 構成の属性ファイルにのみ適用されます。 オプショナルで、構成名として空文字列を指定することにより、 メモリー内のクラスにのみ変更を適用して、アクティブ構成の状態を直ちに更新することができます。

API 呼び出しを実行するには、呼び出し側に、次のように、コマンド・ライン・インターフェースまたは SMIT インターフェースに 必要とされるのと同じレベルの特権が必要です。

- どのユーザーでも、クラス名および特性を読み取ることができます。
- root ユーザーのみが、スーパークラスを 作成、変更、あるいは削除できます。
- root ユーザーまたは指定されるスーパークラス 管理者 (スーパークラス 属性が adminuser または admingroup) のみが、所定のスーパークラスのサブクラスを 作成、変更、あるいは削除できます。

WLM の管理が、WLM 管理者によりコマンド・ラインおよび管理ツールを使用して行われ、かつアプリケーションにより API を介しても行われる場合は、多少注意を要します。 両方のインターフェースは、スーパークラスおよびサブクラス 名に同じネーム・スペース、およびスーパークラスおよびサブクラスの合計数を共用します。

さらに、API がメモリー内の WLM データを直接変更 (例えば、新規クラスを作成) した場合、WLM 管理者は、自分では作成していないクラスが **wlmstat** などのコマンドの出力に表示されるまで、このことには気付きません。 システム管理者が WLM を更新する際に、この API を使用するアプリケーションを混乱させるような競合を避けるため、WLM 属性ファイルに定義されていない API で作成されたクラスが、メモリー内のデータから自動的に除去されることはありません。 これらは、wlm_delete_class ルーチンによるか、**rmclass** コマンド (直接、あるいはシステム管理者による SMIT を介して呼び出される) によって明示的に除去されるまでは、効力を維持します。

WLM API は、以下の機能を持つアプリケーションも 提供します。

- wlm_set 機能を使用して WLM の操作のモードを照会または変更する
- WLM の現在の状況を照会する
- WLM を停止する
- アクティブ・モードとパッシブ・モードを切り替える
- **rset** 割り当てをオンおよびオフにする

- wlm_load ルーチンを使用して、現在の構成または代替の構成で WLM を始動または更新する
- wlm_assign ルーチンを使用して、プロセスまたはプロセスのグループをクラスに割り当てる

この API には、対応する **wlmcntrl** コマンドおよび **wlmassign** コマンドと同じレベルの特権が必要です。

- どのユーザーも WLM の状態を照会できます。
- root ユーザーのみが、WLM の操作モードを変更できます。
- root ユーザーのみが、構成全体の更新またはリフレッシュを実行できます。
- root または許可スーパークラス管理者 (adminuser または admingroup) は、指定されるスーパークラスのサブクラスについて WLM を更新できます。
- root、許可ユーザー (authuser または authgroup で指定される)、あるいは許可スーパークラス管理者 (adminuser または admingroup) は、スーパークラスまたはサブクラスにプロセスを 割り当てることができます。

WLM 統計情報 API

WLM API ルーチンおよび wlm_get_bio_stats は、**wlmstat** コマンドにより表示される WLM 統計情報へのアプリケーション・アクセスを提供します。

WLM 分類 API

wlm_check ルーチンによって、ユーザーは、ある特定の WLM 構成のクラス定義および割り当て規則を確認できます。 API ルーチン wlm_classify を使用すると、アプリケーションは、指定された一連の属性を持つプロセスがどのクラスに分類されるかを判別できます。

バイナリー互換性
将来、データ構造に変更が生じる場合のバイナリー互換性を備えるために、API 呼び出しごとにパラメーターとしてバージョン番号が渡されます。

これによって、ライブラリーは、アプリケーションが作成された際のデータ構造のバージョンを判別できます。

ワークロード・マネージャーの分類、規則、および制限の例

プロセスを分類する方法は数種類あり、これらはすべて、同時に作動します。

最もフレキシブルな構成に備えるために、 トップダウンで最も厳密な最初の突き合わせアルゴリズムを使用します。 プロセスのグループ分けは、プログラムの名前に特殊な大文字小文字を持つユーザー別に、または、 あるユーザーに特殊な大文字小文字を持つパス名別に、あるいは、 他の任意の方法で編成することができます。

ワークロード・マネージャー割り当て規則の例
この例で、構成 *Config* (/etc/wlm/Config/rules ファイル) のトップレベルのルール・ファイルを示します。

```
* This file contains the rules used by WLM to
* assign a process to a superclass
*
* class resvd user    group   application     type  tag
db1      -      -       -    /usr/bin/oracle*      _DB1
db2      -      -       -    /usr/bin/oracle*      _DB2
devlt    -      -      dev         -             -    -
VPs      -   bob,ted    -         -             -    -
acctg    -      -     acct*       -             -    -
System   -    root      -         -             -    -
Default  -      -       -         -             -    -
```

以下に、/etc/wlm/Config/devlt/rules ファイル内の devlt スーパークラスの ルール・ファイルの例を示します。

```
* This file contains the rules used by WLM to
* assign a process to a subclass of the
* superclass devlt
```

```
*
* class  resvd   user    group  application         type       tag
hackers   -    jim,liz     -       -                 -         -
hogs      -       -        -       -              64bit+plock    -
editors   -     !sue       -   /bin/vi,/bin/emacs    -          -
build     -       -        -   /bin/make,/bin/cc     -          -
Default   -       -        -       -                 -          -
```

注 : アスタリスク (*) は、ルール・ファイルで使用されるコメント文字です。

以下に、このルール・ファイルの使用例を示します。 以下の例では、記述されているスーパークラス およびサブクラスは、継承属性が yes に設定されていないと想定しています。 継承が可能になっていたとすると、 新規のプロセスはその親プロセスからスーパークラスまたはサブクラスを 継承します。

- グループ acct3 のユーザー joe が **/bin/vi** を実行すると、プロセスはスーパークラス acctg に入れられます。
- グループ dev のユーザー sue が **/bin/emacs** を実行すると、そのプロセスはスーパークラス devlt (グループ ID が一致) に入れられますが、 このユーザーはそのクラスから除外されているため、editors サブクラスには分類されません。 このプロセスは、デフォルトで devlt に入れられます。
- oracle というユーザー ID と dbm というグループ ID を持つデータベース管理者が DB1 データベースに対して **/usr/sbin/oracle** を開始すると、そのプロセスは Default スーパークラスに分類されます。そのプロセスは、そのタグを _DB1 に設定してはじめて、スーパークラス db1 に割り当てられます。

Share および制限を伴うワークロード・マネージャー・クラスの例
この例の場合は、クラス A、B、C、および D の Share がそれぞれ 3、2、1、および 1 であるとします。

クラス A、C、および D がアクティブであるとすると、計算されたターゲットは次のようになります。

ターゲット (A) = 3/5 = 60%
ターゲット (C) = 1/5 = 20%
ターゲット (D) = 1/5 = 20%

テスト中に、クラス A のアプリケーションはリソースの 50% を使用できるときに十分なパフォーマンスを示すことが分かった場合は、その分を差し引いた残りの 50% のリソースを他のクラスで使用できるようにするとよいでしょう。 これを行うには、このリソースについて、 クラス A に 50% のソフト最大を指定します。 現在の計算済みターゲットである 60% はこの制限をオーバーしているので、これは引き下げられて、このソフト最大値に調整されます。 これが行われた場合、クラス A のターゲットまたは実際の使用量 (どちらか少ない方) が使用可能リソース量から減算されます。 このクラスのターゲットはその制限 (その Share ではない) によって制約されているため、このクラスの Share もアクティブ Share 数から減算されます。 クラス A の現在の使用量が 48% であるとすると、各ターゲットは今度は次のようになります。

ターゲット (A) = 3/5 = 60%、softmax = 50、= 50%
ターゲット (C) = 1/2 * (100 - 48) = 26%
ターゲット (D) = 1/2 * (100 - 48) = 26%

しばらくして、すべてのクラスがアクティブになり 、ターゲットは再度自動的に次のように調整されます。

ターゲット (A) = 3/7 = 42%
ターゲット (B) = 2/7 = 28%
ターゲット (C) = 1/7 = 14%
ターゲット (D) = 1/7 = 14%

CPU 制限を伴うワークロード・マネージャー・クラスの例
この例では、各クラスが、与えられている CPU すべてを使用するという前提で、CPU 割り当てを検査します。

A と B の 2 つのクラスが同じ Tier 内にあるとします。 A の CPU 制限は [30% - 100%] です。 B の CPU 制限は [20% - 100%] です。 これら双方のクラスが実行中であり、十分な CPU を使用している場合、WLM は、最初に、これらのクラスが両方とも、各秒の最小パーセント (数秒から出された平均値) を確保するようにします。 次に、WLM は、CPU ターゲット Share 値に従って、残りの CPU サイクルを分配します。

A と B の CPU ターゲット Share がそれぞれに 60% と 40% である場合、A と B の CPU 使用率は、それぞれ、60% と 40% で安定します。

3 つ目のクラス C が追加されたとします。 このクラスは CPU 制約のジョブのグループで、使用可能な CPU の半分 (またはそれ以上) を使用して実行する必要があります。 クラス C の制限は、[20% - 100%] であり、CPU ターゲット Share は 100% であるものとします。 C が A および B と同じ Tier 内にある場合は、C が開始すると、A と B の CPU 割り当ては急激に減少し、これら 3 つのクラスの CPU 使用率は、それぞれ、30%、20%、および 50% で安定します。 この場合のそれぞれのターゲットも、A と B に関しては最小値になります。

優先順位がより高い可能性のある他のジョブも実行中である場合に、 システム管理者は、バッチ・ジョブで CPU の 50% を消費することは望みません。 この例のような状態では、C をより優先順位の低い Tier 内に置きます。 これによって C は、A と B が必要な分を受け取った後に残った CPU を受け取ります。 この例の場合、A と B はそれぞれ CPU の 100% を使用することができるため、C の受け取る CPU 時間はありません。 しかし、ほとんどの場合、優先順位の高い Tier 内の A と B は対話式ジョブやトランザクション指向ジョブであり、 これらのジョブが常に CPU のすべてを使用するわけではありません。 したがって、C は、CPU の Share をある程度受け取ります。 ただし、この Share について、優先順位がそれと同じかそれよりも低い Tier 内の他のクラスとの競合が生じます。

メモリー制限を伴うワークロード・マネージャー・クラスの例
この例では、可変メモリー・ターゲットを持つプロセスのグループへのメモリー割り当てについて検査します。

次の 3 つのプロセスのグループを実行する必要があるものとします。 つまり、使用されるときは必ず実行する必要のある対話式プロセスのグループ (PEOPLE)、常にバックグラウンドで実行されるバッチ・ジョブ (BATCH1)、および 2 番目の、毎晩実行される、より重要なバッチ・ジョブ (BATCH0) の 3 つです。

PEOPLE には、20% という最小メモリー、50 のメモリー・ターゲット Share、およびクラスの Tier の値 1 が指定されています。20% の最小制限により、このクラスのデスクトップ・アプリケーションは、ユーザーがキーボードに触れるとかなり迅速に再開することが保証されます。

BATCH1 は、50% の最小メモリー、50 のメモリー・ターゲット Share、および Tier の値 3 が指定されています。

BATCH0 は、80% の最小メモリー、50 のメモリー・ターゲット Share、および Tier の値 2 が指定されています。

クラス PEOPLE と BATCH1 には、合計で 70 というメモリー最小制限があります。 通常の操作 (BATCH0 が実行されていない) の場合、この 2 つのクラスは両方とも、それぞれに予約されているメモリーをすべて取得することができます。これら 2 つのクラスは、別々の Tier 内にある場合でも、マシンに残されているメモリーを約半分ずつの割合で共用します。 午前 0 時になって BATCH0 が開始すると、メモリーの最小合計値は 150 に達します。 WLM は、上位 Tier 内のプロセスが終了するまで、最下位 Tier の最小要件を無視します。 BATCH0 は、BATCH1 の 50% 予約のメモリーを使用しますが、PEOPLE の 20% 予約のメモリーは使用しません。 BATCH0 が終了すると、Tier 3 のプロセスのメモリー予約は再度受け入れられ、システムは、通常のメモリー・バランスを持つ状態に戻ります。

ワークロード・マネージャーのコマンド

WLM には、システム管理者のさまざまな機能の実行を可能にするコマンドが用意されています。

サポートされるファイルシステムのタイプを 次に示します。

- スーパークラスとサブクラスの 作成、変更、および削除。使用するコマンドは、**mkclass**、**chclass**、および **rmclass** です。 これらのコマンドは、ファイルのクラス、*Share*、および制限 を更新します。
- WLM の始動、停止、および更新。使用するコマンドは、**wlmcntrl** です。
- **wlmcheck** コマンドを使用して、ある特定の構成用の WLM プロパティー・ファイルを確認し、ある特定のセットの属性を持つプロセスがどのクラス (スーパークラスまたはサブクラス) に割り当てられているかを判別します。
- **wlmstat** (ASCII) コマンドを使用して、クラスごとのリソース使用状況をモニターします。 **svmon** コマンドや **topas** コマンドで始動されるような ほとんどのパフォーマンス・ツールは、WLM クラスを考慮し、新規コマンド・ライン・オプションを使用して クラス別および Tier 別の統計情報を提供する拡張機能を備えています。

- **ps** コマンドのフラグによって、ユーザーは、プロセスとそのアプリケーション・タグがどのクラスにあるかを表示できます。 また、**ps** コマンドでは、ある特定のスーパークラスまたはサブクラスに 属しているすべてのプロセスのリスト表示も可能です。
- 割り当て規則を管理するためのコマンド・ライン・インターフェースはありません。 SMIT 管理ツール、またはテキスト・エディターを使用する必要があります。

デバイス・ノード

デバイスは、ノード と呼ばれるクラスターに編成されます。 各ノードはデバイスの論理的サブシステムであり、そこでは下位レベルのデバイスが上位レベルのデバイスに依存するという親子関係になっています。

例えば、システム・ノードはすべてのノードの中で最高であり、システム内のすべての物理デバイスから構成されています。 システム・デバイスはノードの頂点にあり、その下にバスとアダプターがあり、これらは上位レベルのシステム・デバイスに依存する関係を持っています。 階層の一番下には他のデバイスが接続されていないデバイスがあります。 これらのデバイスは階層内のその上にあるすべてのデバイスに依存する関係を持っています。

親子の依存関係は、ノードを構成するすべてのデバイスを構成するために、起動時に使用されます。 構成はトップ・ノードから下方に向かって行われ、上位レベルのデバイスと依存関係をもつデバイスはその上位レベルのデバイスが構成されるまで構成されません。

AIX オペレーティング・システムは、マルチパス入出力 (MPIO) フィーチャーをサポートします。 デバイスが MPIO が可能なデバイス・ドライバーを備えている場合は、この階層内で複数の親を持てる ので、そのデバイスと、あるマシンとの間、マシン上のある論理区画との間で同時に 複数の通信パスを持つことが可能です。

デバイス・クラス

デバイスを管理するには、許可されているデバイス接続をオペレーティング・システムに認識させる必要があります。 オペレーティング・システムは、デバイスを階層的に次の 3 つのグループに分類します。

これらのグループは以下のとおりです。

- 機能クラス
- 機能サブクラス
- デバイス・タイプ

機能クラス は、同じ機能を実行するデバイスで構成されます。 例えば、プリンターは機能クラスを構成します。 機能クラスはある種のデバイス類似性に従ってサブクラスに分類されます。 例えば、プリンターはシリアルまたはパラレル・インターフェースをもっています。 シリアル・プリンターは 1 つのサブクラスであり、パラレル・プリンターは別のサブクラスです。 デバイス・クラスはそのモデルとメーカーによって分類されます。

デバイス・クラス は、オペレーティング・システムに対して有効な親子関係の接続を定義します。 階層は、起こり得る子接続ロケーションのおのおのに接続が可能な、起こり得るサブクラスを定義します。 例えば、RS-232 8 ポート・アダプターという 用語は、RS-232 サブクラスに属するデバイスだけがアダプターの 8 ポートのどれかに接続できることを示しています。

デバイス・クラスとそれらの階層依存関係は、 オブジェクト・データ・マネージャー (ODM) デバイス構成データベース内で維持管理されます。

デバイス構成データベースおよびデバイスの管理

デバイス情報は、デバイス構成データベースを構成する事前定義データベースまたはカスタマイズ済みデータベースの中に収められます。

事前定義データベースには、システムによってサポートされる 可能性のあるすべてのデバイスに関する構成データが収められています。 階層デバイス・クラス情報はこのデータベースに収容されます。 カスタマイズ済みデータベースには、システム内で現在定義および構成されているすべてのデバイスに関する構成データが収められています。 システムに現在接続されている各デバイスのレコードが保管されます。

構成マネージャーは、システムの起動時と実行時にシステム上のデバイスを自動的に構成するプログラムです。 構成マネージャーは、このプロセス時に事前定義データベースとカスタマイズ・データベースからの情報を使用し、そのあとでカスタマイズ・データベースを更新します。

マルチパス入出力 (MPIO) 機能は、デバイスが発見されたパスとは無関係に、固有のデバイス ID (UDID) を使用して MPIO が可能な各デバイスを識別します。 UDID は、デバイス構成データベースに保管されます。デバイスが発見されると、 このデータベース内の UDID が検査されて、そのデバイスが新規のものか、 既存のデバイスへの別のパスが見つかったのかが判別されます。 1 つのデバイスに対して複数のパスが検出された場合、 特定の要求に対してどのパスを使用するのかは、 デバイス・ドライバーまたは Path Control Manager カーネル・エクステンションが決定します。

SMIT、またはオペレーティング・システム・コマンドを使用して、 デバイスの削除、追加、または構成などのデバイス管理タスクを行えます。

デバイスの状態

システムに接続されるデバイスは、次の 4 つの状態のうちのいずれかに該当します。

システムに接続されるデバイスは、次の状態のうちのいずれかに該当します。

項目	説明
Undefined	デバイスはシステムに 認識されておりません。
Defined	デバイスの特定の情報はカスタマイズ・データベースに記録されますが、システムでは使用できません。
Available	定義されたデバイスはオペレーティング・システムに結合されているか、 さもなければ、その定義されたデバイスは構成に含まれています。
Stopped	デバイスは使用不能ですが、デバイス・ドライバーには引き続き認知されています。

tty デバイスとプリンターが 交替で同じ tty コネクターを使用する場合、tty デバイスとプリンターは、ともに、そのデバイス構成データベース内の同じ親とポート上に定義されます。 これらのデバイスを構成するときは、一度に 1 つだけしかできません。 tty コネクターが構成されるときは、 プリンター固有セットアップ情報はそれが再び構成されるまで残されています。 デバイスは除去されず、定義された状態にあります。 デバイスを定義された状態に維持すると、現在使用中でないデバイスに関するカスタマイズ情報は、最初に使用可能にされるまで、または一時的にシステムから除去されている間残されています。

デバイスに使用されるデバイス・ドライバーが存在する場合、そのデバイスは、そのデバイス・ドライバーを介して使用可能にすることができます。

ある種のデバイス、特に TCP/IP 疑似デバイスは、停止状態を必要とします。

デバイスのロケーション・コード

ロケーション・コード とは、CPU ドロワーまたはシステム 装置から、アダプター、シグナル・ケーブル、または非同期分配ボックス (存在する場合) を経由して、デバイスやワークステーションに至るパスのことです。 このコードは、物理デバイスを識別するもう 1 つの方法です。

ロケーション・コードは、デバイスのタイプによって 異なる 4 つまでの情報のフィールドで構成されます。これらのフィールドは、ドロワー、スロット、コネクター、およびポートを表します。 これらのフィールドのそれぞれは 2 つの文字で構成されます。

ドロワーのロケーション・コードはドロワー・フィールドでのみ構成され、単なる 2 文字のコードです。アダプターのロケーション・コードはドロワー・フィールドとスロット・フィールドで構成され、*AA-BB* というフォーマットを持っています。ここで、*AA* はドロワーのロケーションに対応し、*BB* はアダプターが入るスロットとバスを指します。 その他のデバイスは、フォーマット *AA-BB-CC* または *AA-BB-CC-DD* のロケーション・コードを持っています。ここで *AA-BB* は、デバイスを接続するアダプターのロケーション・コードであり、*CC* は、デバイスを接続するアダプターのコネクターに 対応し、*DD* は、ポート番号または SCSI デバイス・アドレスに対応します。

アダプターのロケーション・コード

アダプターのロケーション・コードは、フォーマット *AA-BB* という 2 つの対の数字で構成されます。 ここで、*AA* はアダプターを 含むドロワーのロケーション・コードを識別し、*BB* はカードを含む入出力バスおよびスロットのロケーション・コードを識別します。

AA フィールドの値 **00** は、アダプターが、システムのタイプによって、CPU ドロワーまたはシステム装置に接続されていることを意味します。 **AA** フィールドがそれ以外の値であるときは、 カードが入出力拡張ドロワーに接続されていることを意味します。 この場合には、*AA* の値は、非同期拡張アダプターを 含む CPU ドロワーの入出力バス番号とスロット番号を識別します。 最初の数字は、0 で標準入出力バスに対応する入出力バスを識別し、1 はオプショナルの 入出力バスに対応します。 2 番目の数字は指示された入出力バス上のスロット番号を識別します。

BB フィールドの最初の数字はアダプター・カードを含む入出力ボードを識別します。 カードが CPU ドロワーまたはシステム装置にある場合には、 この数字は標準入出力バスに対しては 0 であり、オプショナルの入出力バスに対しては 1 です。 カードが入出力拡張ドロワーにある場合には、この数字は 0 です。 2 番目の数字は、カードを含む指示された入出力バス上のスロット番号 (または入出力拡張ドロワーのスロット番号) を識別します。

標準入出力ボードを識別するには、ロケーション・コード 00-00 を使用します。

例:

項目	説明
00-05	標準入出力ボードのスロット 5 のアダプター・カードを識別し、そのアダプター・カードは、システムのタイプによって、CPU ドロワーまたはシステム装置のいずれかに接続されています。
00-12	アダプターがオプショナルの 入出力バスのスロット 2 で識別され、それは CPU ドロワーに接続されています。
18-05	入出力拡張ドロワーのスロット 5 に接続されているアダプター・カードを識別します。 このドロワーは、CPU ドロワーのオプション入出力バスのスロット 8 に取り付けられた、非同期拡張アダプターに接続されています。

プリンターおよびプロッターのロケーション・コード

ロケーション・コード 00-00-S1-00 または 00-00-S2-00 は、プリンター、プロッター、または tty デバイスが、標準入出力ボードのシリアル・ポート s1 または s2 に接続されていることを示します。 ロケーション・コード 00-00-0P-00 は、並列プリンターが標準入出力ボードのパラレル・ポートに接続されていることを示します。

その他のロケーション・コードは、プリンター、プロッター、または tty デバイスが、標準入出力ボード以外のアダプター・カードに接続されていることを示します。 これらのプリンター、プロッター、および tty デバイスに対しては、ロケーション・コードのフォーマットは *AA-BB-CC-DD* です。ここで、*AA-BB* は制御アダプターのロケーション ・コードを示します。

項目	説明
AA	AA フィールドの値 **00** は、システムのタイプによって、アダプター・カードが CPU ドロワーまたはシステム装置にあることを示します。 **AA** フィールドのその他の値は、カードが入出力拡張ドロワーに接続されていることを示します。この場合には、最初の数字は入出力バスを指し、2 番目の数字は、入出力拡張ドロワーが接続されている非同期拡張アダプターを 含む、CPU 内のバス上のスロット番号を指します。
BB	**BB** フィールドの最初の数字は、アダプター・カードを含む入出力バスを識別します。 カードが CPU ドロワーまたはシステム装置にある場合には、 この数字は標準入出力バスに対しては 0 であり、オプショナルの入出力バスに対しては 1 です。 カードが入出力拡張ドロワーにある場合には、この数字は 0 です。 2 番目の数字は、カードを含む入出力バス上のスロット番号 (または入出力拡張ドロワーのスロット番号) を識別します。

項目	説明
CC	**CC** フィールドは、非同期分配ボックスが接続されているアダプター・カード上のコネクターを識別します。 可能な値は、01、02、03、および 04 です。
DD	**DD** フィールドは、プリンター、プロッター、または tty デバイスが接続されている非同期分配ボックス上のポート番号を識別します。

tty のロケーション・コード

ロケーション・コード 00-00-S1-00 または 00-00-S2-00 は、tty デバイスが標準入出力シリアル・ポート s1 または s2 に接続されていることを示します。

その他のロケーション・コードは、標準入出力ボード以外のアダプター・カードに接続されている tty デバイスを指します。 これらのデバイスの場合、ロケーション・コードのフォーマットは *AA-BB-CC-DD* です。 ここで *AA-BB* は制御アダプター・カードのロケーション・コードを示します。

項目	説明
AA	AA フィールドの値 **00** は、システムのタイプによって、アダプター・カードが CPU ドロワーまたはシステム装置にあることを示します。 **AA** フィールドのその他の値は、カードが入出力拡張ドロワーにあることを示します。 この場合には、最初の数字は入出力バスを識別し、2 番目の数字は、入出力拡張ドロワーが接続されている非同期拡張アダプターを 含む、CPU ドロワー内のバス上のスロット番号を識別します。
BB	**BB** フィールドの最初の数字は、アダプター・カードを含む入出力バスを識別します。 カードが CPU ドロワーまたはシステム装置にある場合には、 この数字は標準入出力バスに対しては 0、 オプショナルの入出力バスに対しては 1 です。 カードが入出力拡張ドロワーにある場合には、この数字は 0 です。 2 番目の数字は、カードを含む入出力バス上のスロット番号 (または入出力拡張ドロワーのスロット番号) を識別します。
CC	**CC** フィールドは、非同期分配ボックスが接続されているアダプター・カード上のコネクターを識別します。 可能な値は、01、02、03、および 04 です。
DD	**DD** フィールドは、tty デバイスが接続されている非同期分配ボックス上のポート番号を識別します。

SCSI デバイスのロケーション・コード

以下は SCSI デバイスのロケーション・コードです。

このロケーション・コードは、以下を含めてすべての SCSI デバイスに適用されます。

- CD-ROM
- ディスク
- イニシエーター・デバイス
- 読み取り/書き込み光ディスク・ドライブ
- 磁気テープ
- ターゲット・モード

ロケーション・コードのフォーマットは AA-BB-CC-S,L です。**AA-BB** フィールドは、SCSI デバイスを制御する SCSI アダプターのロケーション・コードを識別します。

項目	説明
AA	AA フィールドの値 **00** は、システムのタイプによって、制御アダプター・カードが CPU ドロワーまたはシステム装置にあることを示します。

項目　説明

BB　**BB** フィールドは、カードを含む入出力バスとスロットを識別します。 最初の数字は入出力バスを識別します。 それは、標準入出力バスに対しては 0 で、 オプショナルの 入出力バスに対しては 1 です。 2 番目の数字はカードを含む指示された入出力バス上のスロットです。 BB フィールドの値 **00** は、SCSI コントローラーを指します。

CC　**CC** フィールドは、デバイスが接続されているカードの SCSI バスを識別します。 単一 SCSI バスのみを提供するカードの場合、このフィールドは 00 に設定されます。 そうでない場合は、値 00 はカードの内部 SCSI バスに接続されたデバイスを指し、値 01 はカードの外部 SCSI バスに接続されたデバイスを指します。

S,L　**S,L** フィールドは、SCSI デバイスの SCSI ID と論理ユニット番号 (LUN) を識別します。 値 S は SCSI ID を指し、値 L は LUN を指します。

Dials/LPFKeys のロケーション・コード

グラフィックス入力アダプター 接続の Dials/LPFKeys デバイスに対しては、ロケーション・コードのフォーマットは *AA-BB-CC* です。

個々のフィールドは次のように解釈されます。

項目　説明

AA　AA フィールドの値 **00** は、システムのタイプによって、制御アダプター・カードが CPU ドロワーまたはシステム装置にあることを示します。

BB　**BB** フィールドは、カードを含む入出力バスとスロットを識別します。 最初の数字は入出力バスを識別します。 それは、標準入出力バスに対しては 0 で、 オプショナルの 入出力バスに対しては 1 です。 2 番目の数字はカードを含む指示された入出力バス上のスロットです。

CC　**CC** フィールドは、デバイスが接続されているカード・コネクターを指します。 その値は、デバイスがカード上のポート 1 に接続されているか、ポート 2 に接続されているかによって、01 または 02 のいずれかです。

注 : 直列接続の Dials/LPFKeys デバイスにはロケーション・コードがありません。 これらのデバイスは tty に接続されると見なされるからです。 tty デバイスは Dials/LPFKeys 定義のときユーザーが指定します。

マルチプロトコル・ポート・ロケーション・コード

マルチプロトコル・ポートのロケーション・コードのフォーマットは、*AA-BB-CC-DD* です。ここで、*AA-BB* はマルチプロトコル・アダプター・カードのロケーション・コードを指します。

個々のフィールドは次のように解釈されます。

項目　説明

AA　AA フィールドの値 **00** は、システムのタイプによって、マルチプロトコル・アダプター・カードが CPU ドロワーまたはシステム装置にあることを示します。

BB　**BB** フィールドは、カードを含む入出力バスとスロットを識別します。 最初の数字は入出力バスを識別します。 それは、標準入出力バスに対しては 0 で、 オプショナルの 入出力バスに対しては 1 です。 2 番目の数字はカードを含む指示された入出力バス上のスロットです。

CC　**CC** フィールドは、マルチプロトコル分配ボックスが接続されているアダプター・カード上のコネクターを示します。 その値は常に 01 です。

DD　**DD** フィールドは、マルチプロトコル分配ボックス上の物理ポート番号を識別します。 可能な値は、00、01、02、および 03 です。

iSCSI オフロード・アダプターのセットアップ

iSCSI オフロード・アダプターのセットアップは、アダプターの 構成とターゲットの追加または更新を伴います。 これらの指示は、iSCSI オフロード・アダプターにのみ適用されます。 iSCSI ソフトウェア・イニシエーターの構成については、『iSCSI ソフトウェア・イニシエーターとソフトウェア・ターゲット』を参照してください。

AIX における iSCSI アダプターの構成

iSCSI アダプターの 構成は、非常に単純かつ明快な作業です。

1. AIX コマンド・プロンプトで、**smit iscsi** と入力します。
   iSCSI 画面が表示されます。
2. iSCSI 画面から「**iSCSI Adapter (iSCSI アダプター)**」を選択します。
   「iSCSI Adapter (iSCSI アダプター)」画面が表示されます。
3. 「iSCSI Adapter (iSCSI アダプター)」画面から「**Change / Show Characteristics of an iSCSI Adapter (iSCSI アダプターの 特性の変更/表示)**」を選択します。
   「Change / Show Characteristics of an iSCSI Adapter (iSCSI アダプターの 特性の変更/表示)」画面が表示されます。
4. 構成する iSCSI アダプターをリストから 選択します。
   以下の例のような構成画面が表示されます。

```
                                                [Entry Fields]

iSCSI Adapter                                   ics0

Description                                     iSCSI Adapter

Status                                          Available

Location                                        10-60

Max. number of commands to queue to adapter    [200]      +                        +#

Maximum Transfer Size                          [0x100000]     +

Discovery Filename                             [/etc/iscsi/targetshw]
Discovery Policy                               [file]
Automatic Discovery Secrets Filename           [/etc/iscsi/autosecret]
Adapter IP Address                             [10.1.4.187]
Adapter Subnet Mask                            [255.255.255.0]

Adapter Gateway Address                        [10.1.4.1]

Apply change to DATABASE only                   no        +
```

注 : 特定のフィールドの 目的に関して質問がある場合は、そのフィールド上にカーソルを置き、**F1** を押してヘルプを得ます。

フラット・ファイル・ディスカバリーを使用する場合は、「**Discovery Policy (ディスカバリー・ポリシー)**」フィールドに「file （ファイル)」と入力します。 ODM ディスカバリーを使用する場合は、「**Discovery Policy (ディスカバリー・ポリシー)**」フィールドに「odm」と入力します。DHCP 発見の iSCSI ターゲットの場合は、「**Discovery Policy (ディスカバリー・ポリシー)**」フィールドに「slp」と入力します。

iSCSI ターゲットのフラット・ファイルの更新

フラット・ファイルは、iSCSI ターゲットの構成に使用する静的構成ファイルです。 そのデフォルト・ファイル名は /etc/iscsi/targetshw です。

フラット・ファイル内のすべての関係する iSCSI ターゲット・ディスカバリー属性を、明示的に指定する必要があります。

関連情報

targets File

静的に発見された iSCSI ターゲットの ODM への追加

ODM ディスカバリー・ポリシーを iSCSI アダプターまたは iSCSI ソフトウェア・イニシエーターのどちらかに使用している場合、iSCSI ドライバーは iSCSI ターゲット記述を ODM から取得します。

ODM 内の iSCSI ターゲット情報を取り扱う場合は、AIX コマンドまたは SMIT を使用できます。**chiscsi**、**lsiscsi**、**mkiscsi**、および **rmiscsi** コマンドは、iSCSI ターゲット情報の変更、表示、追加、および ODM からの除去を行います。

SMIT を使用して、静的に発見された 1 つの iSCSI ターゲットを ODM に追加するには、次のようにします。

1. AIX コマンド・プロンプトで、**smit iscsi** と入力します。
   iSCSI 画面が表示されます。
2. iSCSI 画面から「**iSCSI Target Device Parameters in ODM (ODM の iSCSI ターゲット・デバイス・パラメーター)**」を選択します。
   「iSCSI Target Device Parameters in ODM (ODM の iSCSI ターゲット・デバイス・パラメーター)」画面が表示されます。
3. iSCSI 画面から「**Add an iSCSI Target Device in ODM (iSCSI ターゲット・デバイスの ODM への追加)**」を選択します。
   「Add an iSCSI Target Device in ODM (iSCSI ターゲット・デバイスの ODM への追加)」画面が表示されます。
4. 構成する iSCSI デバイスをリストから選択します。 このリストには、iSCSI オフロード・アダプター (「ics」で始まるデバイス) と iSCSI ソフトウェア・イニシエーター (「iscsi」で始まるデバイス) がどちらも含まれている場合があります。
5. 構成する iSCSI デバイスをリストから選択します。 このリストには、iSCSI オフロード・アダプター (「ics」で始まるデバイス) と iSCSI ソフトウェア・イニシエーター (「iscsi」で始まるデバイス) がどちらも含まれている場合があります。
6. 以下のフィールドに該当する情報を入力します。
   以下に例を示します。

```
[Entry Fields]

iSCSI Adapter                                       ics0
iSCSI Target Name                                   [iqn.com.ibm.tgt:0]
IP Address of iSCSI Target                          [10.1.4.25]
Port Number of iSCSI Target                         [3260]
Password                                            [my password]
User Name                                           [user name]
```

   注 : 特定のフィールドの目的に関して質問がある場合は、そのフィールド上にカーソルを置き、**F1** を押してヘルプを得ます。

フラット・ファイルから静的に発見された iSCSI ターゲットの ODM への追加

フラット・ファイルの情報は、SMIT を使用して ODM にインポートできます。

1. AIX コマンド・プロンプトで、**smit iscsi** と入力します。
   iSCSI 画面が表示されます。
2. iSCSI 画面から「**iSCSI Target Device Parameters in ODM (ODM の iSCSI ターゲット・デバイス・パラメーター)**」を選択します。
   「iSCSI Target Device Parameters in ODM (ODM の iSCSI ターゲット・デバイス・パラメーター)」画面が表示されます。
3. iSCSI 画面から「**Add an iSCSI Target Device in ODM (iSCSI ターゲット・デバイスの ODM への追加)**」を選択します。
   「Add an iSCSI Target Device in ODM (iSCSI ターゲット・デバイスの ODM への追加)」画面が表示されます。
4. iSCSI 画面から「**Add iSCSI Target Device Data in ODM from a File (iSCSI ターゲット・ターゲット・デバイスのファイルから ODM への追加)**」を選択します。
   「Add iSCSI Target Device Data in ODM from a File (iSCSI ターゲット・デバイス・データのファイルから ODM への追加)」画面が表示されます。
5. 構成する iSCSI アダプターをリストから選択します。

選択した iSCSI アダプターの「Add iSCSI Target Device Data in ODM from a File (iSCSI ターゲット・デバイス・データのファイルから ODM への追加)」画面が表示されます。

6. 以下のフィールドに該当する情報を入力します。
以下に例を示します。

```
                                                [Entry Fields]
iSCSI Protocol Device                           iscsi3
iSCSI Group                                    [static]      +
Filename of iSCSI Targets                      [/etc/iscsi/targetshw]      +
```

注 : 特定のフィールドの目的に関して質問がある場合は、そのフィールド上にカーソルを置き、**F1** を押してヘルプを得ます。

PCI ホット・プラグ管理

オペレーティング・システムの稼働中に、新しい PCI ホット・プラグ・アダプターを使用可能な PCI スロットに挿入できます。

これは、現在インストールされているものと同じタイプの別のアダプターである場合も、異なるタイプの PCI アダプターの場合もあります。 新しいリソースは、オペレーティング・システムを再始動しなくても、オペレーティング・システムやアプリケーションで使用可能になります。 ホット・プラグ・アダプターを追加する理由としては、次のようなことが考えられます。

- ユーザーの既存のハードウェアおよびファームウェアに別の機能を追加する。
- PCI アダプターにより提供される機能が必要ではなくなったシステムからこれらのアダプターを移行する。
- PCI アダプターを含むオプショナルのハードウェア・サブシステムの初期構成、およびオペレーティング・システムのインストールおよび始動の後に使用可能になったアダプター・カードに新しいシステムをインストールする。

注 : PCI ホット・プラグの置換または追加操作を使用して、あるいは、動的論理区画を使用してアダプターを追加する場合、**bootlist** コマンドを使用してブート・デバイスとして指定する際に、このアダプターとその子デバイスを使用できない可能性があります。 マシンを再始動して、すべての潜在的なブート・デバイスをオペレーティング・システムに認知させなければならないこともあります。 既にブート・リストにリスト済みのアダプターを同一タイプのアダプターと置き換えた場合、そのアダプターはやはり有効なブート・デバイスです。

また、システムのシャットダウンや電源をオフにすることなく、問題または障害のある PCI ホット・プラグ・アダプターを取り外したり、同じタイプの別のアダプターに交換することもできます。 アダプターを交換する場合は、タイプが同じなので、既存のデバイス・ドライバーがそのアダプターをサポートします。そのアダプターの下にあるデバイスに関するデバイスの構成および構成情報が、その置き換えるデバイス用に保持されます。 アダプターを取り替える理由としては、以下のことが考えられます。

- 問題の判別を助けるため、あるいは障害のある FRU を分離するために一時的にカードを置換する。
- 欠陥、障害のある、あるいは断続的に障害が起きるアダプターを、正常に機能するカードと取り替える。
- HACMP またはマルチパス入出力構成内の障害のある予備アダプターを取り替える。

ホット・プラグ・アダプターを取り外すと、そのアダプターによって提供されるリソースも、オペレーティング・システムやアプリケーションから使用できなくなります。アダプターを除去する理由には、以下のものが含まれます。

- 既存の入出力サブシステムを除去する。
- 必要ではなくなったアダプターまたは障害のあるアダプター、および置換カードが使用できないアダプターを除去する。
- 除去元のシステムで機能が必要でなくなった場合に、アダプターを別のシステムに移行する。

ホット・プラグ・デバイスを除去あるいは置換するには、その前にそれを構成から削除する必要があります。 関連付けられたデバイス・ドライバーは、デバイスに割り振られていたシステム・リソースをすべて

解放しなければなりません。 これには、メモリー割り振りを外して解放し、割り込みおよび EPOW ハンドラーの定義を解除し、DMA およびタイマー・リソースを解放し、さらに必要な他のステップを実行することを含みます。 割り込み、バス・メモリー、およびバス入出力がデバイスで使用不可になっていることをドライバーで確認することも必要です。

アダプターの取り外しの前後には、システム管理者は以下のタスクを実行する必要があります。

- デバイスを使用しているアプリケーション、デーモン、またはプロセスを終了および復元する。
- ファイルシステムをアンマウントして再マウントする。
- デバイス定義を除去あるいは再作成し、使用中のデバイスを解放するために必要な他の操作を実行する。
- システムをサービスのためにセーフ状態にする。
- 必要な任意のデバイス・ドライバーを入手し、インストールする。

除去および置換操作は、指定されるスロットに接続されているデバイスが構成解除され、 定義済み状態にならない限り失敗します。 これは、**rmdev** コマンドを使用して実行できます。定義済み状態にアダプターを入れる前に、 このアダプターを使用しているアプリケーションをすべてクローズしてください。 そうしないと、コマンドは失敗します。**rmdev** コマンドの詳細については、**rmdev** を参照してください。

場合によっては、以下のタスクも行うことができます。

- 挿入、除去、または置換する PCI ホット・プラグ・アダプターを準備する。
- ホット・プラグ操作に関係するスロットまたは PCI アダプターを識別する。
- PCI ホット・プラグ・アダプターを除去または挿入する。

注 : ホット・プラグ操作中は、オブジェクト・データ・マネージャー (ODM) はロックされたままです。 そのため、ODM を必要とするその他のタスクはハングまたは失敗することがあります。 また、他のノードで開始されたクラスター全体の構成変更は、個別のクラスターにおいてハングまたは失敗することがあります。 したがって、ホット・プラグ操作が完了するまでは、そのようなタスクを実行しないでください。

重要 : PCI ホット・プラグ管理は、システムの電源を切ったりオペレーティング・システムを再始動したりせずに PCI アダプターを追加、除去、および置換する機能を提供していますが、 ホット・プラグ・スロットにあるすべてのデバイスをこの方法で管理できるというわけではありません。 例えば、rootvg ボリューム・グループを構成するハード・ディスクまたはこれが接続されている入出力コントローラーは、 オペレーティング・システムの実行に必要なものであるために、 システムの電源を切らないままで除去したり置換したりすることはできません。rootvg ボリューム・グループがミラーリングされている場合、**chpv** コマンドを使用してディスクをオフラインにすることができます。 rootvg ボリューム・グループが、 マルチパス入出力 (MPIO) 使用可能で複数の入出力コントローラーに接続されている 1 つ以上のディスクに常駐する場合、それらの入出力コントローラーの 内の 1 つをシステムをリブートすることなしに除去 (または置換) することができます。 この状態では、 除去される (または、置換される) 入出力コントローラーからのパスはすべて、アダプター上で **rmdev -R** コマンドを使用して構成解除する必要があります。 これにより、パスとアダプターが 構成解除されます。 これで、ホット・プラグ管理に進むことができます。 PCI ホット・プラグ・アダプターを除去または挿入しようとする前に、「*PCI Adapter Placement Reference*」(ホット・プラグをサポートするシステム装置に付属する) を参照し、ご使用のアダプターがホット・スワップ可能かどうかを判別してください。 アダプターのインストール方法および取り外し方法については、ご使用のシステム装置の資料を参照してください。

PCI ホット・プラグ・スロット情報の表示

ホット・プラグ・アダプターを追加、取り外し、または取り替える前に、PCI ホット・プラグ・スロットに関する情報を表示することができます。

以下の情報を表示できます。

- マシン内のすべての PCI ホット・プラグ・スロットのリスト
- 使用可能な空きスロットがあるかどうか
- 現在使用中のスロット
- 特定のスロットの特性 (スロット名、説明、コネクターのタイプ、接続デバイスの名前など)

SMIT やシステム・コマンドを使用できます。 これらのタスクを行うには root ユーザーとしてログインする必要があります。

SMIT 高速パス手順

1. システム・プロンプトで `smit devdrpci` と入力してから Enter キーを押します。
2. SMIT ダイアログを使用して、作業を完了します。

作業を完了するための追加情報を取得するには、SMIT ダイアログの F1 ヘルプ・キーを選択します。

コマンド・プロシージャー

ホット・プラグ・スロットや接続されているデバイスに関する情報を表示するには、以下のコマンドを使用します。

- lsslot コマンド。すべての PCI ホット・プラグ・スロットとその特性をリスト表示します。
- **lsdev** コマンド。システムにインストールされているすべてのデバイスの 現在の状態を表示します。

PCI 通信アダプターの構成解除

以下は、PCI 通信アダプターを 構成解除するプロセスの概要の説明です。 これには、イーサネット、トークンリング、FDDI、および ATM アダプターも 含まれています。

ご使用のアプリケーションが TCP/IP プロトコルを使用している場合には、ネットワーク・インターフェース・リストからアダプターの TCP/IP インターフェースを 除去してからでなければ、アダプターを 定義済み状態にすることはできません。 **netstat** コマンドを使用して、ご使用のアダプターが TCP/IP に構成されているかどうかを判別し、ご使用のアダプターのアクティブ ・ネットワーク・インターフェースを 確認してください。 **netstat** コマンドについては、**netstat** を参照してください。

イーサネット・アダプターには 2 つのインターフェース、標準イーサネット (en*X*) または IEEE 802.3 (et*X*) があります。 *X* は、ent*X* アダプター 名で使用されているのと同じ番号です。 一度に 1 つのインターフェースしか、TCP/IP を使用することはできません。例えば、イーサネット・アダプター `ent0` は en0 および et0 インターフェースを 持つことができます。

トークンリング・アダプターのインターフェースは Token-ring (tr*X*)、1 つだけです。 *X* は、tok*X* アダプター名と同じ数です。 例えば、トークンリング・アダプター `tok0` には tr0 インターフェースがあります。

ATM アダプターの atm インターフェースは ATM (at*X*)、1 つだけです。 *X* は、atm*X* アダプター 名と同じ数です。 例えば、ATM アダプター `atm0` には at0 インターフェースがあります。 ただし、ATM アダプターでは、複数のエミュレートされたクライアントを単一アダプターで 実行することができます。

ifconfig コマンドは、ネットワークからインターフェースを 除去します。 **rmdev** コマンドは、デバイス定義をカスタマイズ・デバイス・オブジェクト・クラスに保持して、PCI デバイスを構成解除します。 アダプターが定義済みの状態に入れられたならば、**drslot** コマンドを使用してこのアダプターを 除去することができます。

デバイス定義をカスタマイズ・デバイス・オブジェクト・クラスに保持しながら、PCI バス `pci1` の子とその下の他のすべてのデバイスを構成解除するには、次のように入力します。

```
rmdev  -p pci1
```

システムは、以下のようなメッセージを表示します。

```
rmt0 Defined
hdisk1 Defined
scsi1 Defined
ent0 Defined
```

PCI ホット・プラグ・アダプターの取り外しまたは取り替え

オペレーティング・システムをシャットダウンしたり、システムの電源をオフにしなくても、システム装置から PCI ホット・プラグ・アダプターを取り外したり取り替えることができます。 アダプターを取り外すと、オペレーティング・システムおよびアプリケーションでは、そのアダプターが提供するリソースが使用できなくなります。

アダプターを取り外す前に、そのアダプターを 構成解除する必要があります。

以下は、PCI ホット・プラグ・アダプターを取り外す手順です。 SMIT またはシステム・コマンドを使用してこれらの作業を完了できます。 これらのタスクを行うには root ユーザーとしてログインする必要があります。

アダプターを 同じ種類の他のアダプターと 取り替えた場合は、取り替えられたアダプターの 構成情報が維持され、この情報が取り替えたカードと比較されます。 取り替えられたアダプターの 既存のデバイス・ドライバーは、取り替えたアダプターをサポートしている 必要があります。

SMIT 高速パス手順

1. システム・プロンプトで「`smit devdrpci`」と入力してから Enter キーを押します。
2. SMIT ダイアログを使用して、作業を完了します。

作業を完了するための追加情報を取得するには、SMIT ダイアログの F1 ヘルプ・キーを選択します。

コマンド・プロシージャー

以下のコマンドを使用して、ホット・プラグ・スロットや接続されたデバイスに関する情報を表示したり、PCI ホット・プラグ・アダプターを取り外すことができます。

- **lsslot** コマンド。すべての PCI ホット・プラグ・スロットおよびその特性のリストを表示します。 このコマンドの使用法については、コマンド・リファレンス 第 *3* 巻の **lsslot** を参照してください。
- **lsdev** コマンド。システムにインストールされているすべてのデバイスの 現在の状態を表示します。 このコマンドの使用法については、コマンド・リファレンス 第 *3* 巻の **lsdev** を参照してください。
- **drslot** コマンド。ホット・プラグ・アダプターを取り外しできるようにホット・プラグ・スロットを準備します。 このコマンドの使用法については、「コマンド・リファレンス 第 *2* 巻」の **drslot** を参照してください。

PCI ホット・プラグ・アダプターの追加

PCI ホット・プラグ・アダプターをシステム 装置の空きスロットに追加すると、オペレーティング・システムをリブートしなくても、オペレーティング・システムおよびアプリケーションが 新しいリソースを使用できるようになります。 アダプターは、現在インストールされているアダプター ・タイプ、または 別のアダプター・タイプのものでも 構いません。

以下は、新規の PCI ホット・プラグ・アダプターの 追加手順です。

重要 : PCI ホット・プラグ・アダプターを 追加する前に、ホット・プラグをサポートするシステム 装置に付属していた 「*PCI* アダプター インストール・ガイド」を参照して、お手持ちのアダプターがホット・プラグ可能かどうかを確認してください。 アダプターのインストール 方法および取り外し方法については、ご使用のシステム 装置の資料を参照してください。

新規の PCI ホット・プラグ・アダプターを 追加するには、次の作業を実行します。

- マシンの空きスロットを見つけ、識別する。
- そのスロットをアダプター 用に構成するための準備をする。
- 必要に応じて、デバイス・ドライバーをインストールする。
- 新規アダプターを 構成する。

SMIT やシステム・コマンドを使用できます。 これらのタスクを行うには root ユーザーとしてログインする必要があります。

注 : システムにホット・プラグ・アダプターを 追加するときに、**bootlist** コマンドを使用してそのアダプターと子デバイスをブート・デバイスとして指定しようとしても、指定できない場合があります。 オペレーティング・システムにすべての 使用可能なブート・デバイスを認識させるには、システムをリブートする必要があります。

SMIT 高速パス手順

1. システム・プロンプトで「`smit devdrpci`」と入力してから Enter キーを押します。
2. SMIT ダイアログを使用して、作業を完了します。

作業を完了するための追加情報を取得するには、SMIT ダイアログの F1 ヘルプ・キーを選択します。

コマンド・プロシージャー

以下のコマンドを使用して、PCI ホット・プラグ・スロットや接続されているデバイスに関する情報を表示したり、PCI ホット・プラグ・アダプターを 追加することができます。

- **lsslot** コマンド。すべてのホット・プラグ・スロットとその特性をリスト表示します。 このコマンドの使用法については、コマンド・リファレンス 第 *3* 巻の **lsslot** を参照してください。
- **lsdev** コマンド。システムにインストールされているすべてのデバイスの 現在の状態を表示します。 このコマンドの使用法については、コマンド・リファレンス 第 *3* 巻の **lsdev** を参照してください。
- **drslot** コマンド。ホット・プラグ・アダプターを追加または取り外すためにホット・プラグ・スロットの準備を行います。 このコマンドの使用法については、「コマンド・リファレンス 第 *2* 巻」の **drslot** を参照してください。

マルチパス入出力 (Multiple Path I/O)

マルチパス入出力 (Multiple Path I/O (MPIO)) を使用すると、1 つ以上 の物理接続、あるいはパス を使用して、デバイスを個別に検出することができます。

パス制御モジュール (PCM) はパス管理機能を提供します。

MPIO 可能デバイス・ドライバーは、複数のタイプのターゲット・デバイスを制御することができます。 PCM は、1 つ以上の特定デバイスをサポートすることができます。 したがって、1 つのデバイス・ドライバーを、各ターゲット・デバイスへの複数のパスにわたる入出力を制御する複数の PCM にインターフェースさせることができます。

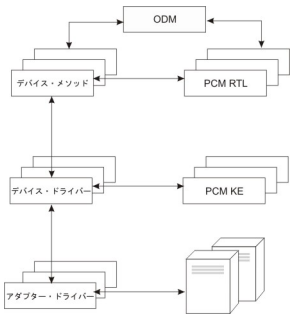


図 *12. MPIO* コンポーネントの対話

デバイスが MPIO を利用するためには、オブジェクト・データ・マネージャー (ODM) 内のデバイスのドライバー、メソッド、および事前定義属性を、 複数のパスの検出、構成、および管理をサポートするために変更する必要があります。 パラレル SCSI ディスクおよびファイバー・チャネル・ディスクのデバイス・ドライバーおよび そのデバイス・メソッドは MPIO ディスク装置をサポートします。 iSCSI ディスク装置が MPIO デバイスとしてサポートされます。 ファイバー・チャネル・テープ・デバイス・ドライバーとそのデバイス・メソッドは、MPIO テープ・デバイスをサポートします。 また、ODM 内の一部のデバイスの事前定義属性が MPIO のために変更されました。

AIX PCM は、PCM RTL 構成モジュールと PCM KE カーネル・エクステンションから構成されます。 PCM RTL は、デバイス・メソッドが、PCM KE が必要とする追加の PCM KE デバイス固有の属性またはパス ODM 属性を検出できるようにするランタイム・ロード可能モジュールです。 PCM RTL はデバイス・メソッドによってロードされます。 PCM RTL 内の 1 つ以上のルーチンは、PM KE 変数を初期設定または変更する特定の操作を実行するためにアクセスされます。

PCM KE は、MPIO インターフェースをサポートする 任意のデバイス・ドライバーにパス制御の管理機能を提供します。 PCM KE は、デバイス構成に応じてパスを検出し、 その情報をデバイス・ドライバーに伝えます。 各 MPIO 可能デバイス・ドライバーは、その直接の親 (複数の場合もある) からパスをデバイスに追加します。 種々のパスにわたる入出力の維持およびスケジューリングは、PCM KE によって提供され、MPIO 可能デバイス・ドライバーにははっきりと認識されません。

PCM KE は複数の経路指定アルゴリズムを提供することができます。 このアルゴリズムはユーザーが選択できます。 PCM KE は、 任意の入出力要求に関する最適パスの判別および選択に使用できる情報の収集にも役立ちます。 PCM KE は、種々の基準 (ロード・バランシング、接続速度、接続障害など) に基づいて最適パスを選択することができます。

AIX PCM には、以下のことを行うためのヘルス検査機能があります。

- パスを検査し、どのパスが入出力の送信に現在使用可能であるかを判別する。
- 一時的なパス障害 (例えば、デバイスへのケーブルが 除去され、 その後で再接続された) が原因で、障害があると以前にマークされたパスを使用可能にする。
- フェイルオーバーが起こった場合に使用されることになる現在未使用のパスを検査する (例えば、アルゴリズム属性値が `failover` の場合は、 ヘルス検査が代替パスをテストすることができます)。

AIX デフォルト PCM を使用してすべてのディスク・デバイスとテープ・デバイスを検出および構成 できるわけではありません。 AIX デフォルト PCM は 2 つのパス制御モジュールで構成されています。うち 1 つはディスク装置の管理用、もう 1 つはテープ・デバイスの管理用です。 ご使用のデバイスが検出されない場合は、デバイス・ベンダーに問い合わせて、PCM がご使用のデバイスで使用可能であるかどうかを判別してください。

MPIO 可能なデバイスの管理

マルチパス I/O (MPIO) フィーチャーは、フェイルオーバーを行うためにデバイスへの代替パスを 定義するのに使用できます。

フェイルオーバー とは、デバイスの信頼性および可用性を高める パス管理アルゴリズムのことです。システムは自動的に特定の入出力パスの障害を 検出し、入出力の経路を代替パスに切り替えるからです。 すべての SCSI SCSD ディスク・ドライブ は、自動的に MPIO デバイスとして構成され、ファイバー・チャネル・ディスク・ドライブの選択番号 は、MPIO Other ディスクとして構成できます。 デバイス・ドライバー と AIX での MPIO インプリメンテーションとに互換性があれば、その他のデバイスもサポート されます。

MPIO は BOS のインストールの一部としてインストールされ、構成されます。 これ以外の構成を行う必要はありません。ただし、SMIT、またはコマンド・ライン・インターフェースを 使用して、デバイス (またはデバイス・パス) を追加、除去、再構成、使用可能化、および使用不可化することができます。 以下のコマンドを使用して、MPIO パスを管理することができます。

mkpath
ターゲット・デバイスへのパスを追加します。

rmpath
ターゲット・デバイスへのパスを除去します。

chpath
ターゲット・デバイスへのパスの属性または運用状況を変更します。

lsmpio
マルチパス入出力 (MPIO) 記憶デバイスに関する状況、構成、および統計といった情報を表示します。

lspath
ターゲット・デバイスへのパスに関する情報を表示します。

Multipath I/O (MPIO) のような複数のポートを持つディスクで BOS インストールまたは `mksysb` インストールを実行する場合、そのインストールの間、ポートがアクティブになっている必要があります。

MPIO デバイスとしての SCSI デバイスのケーブル接続

SCSI デバイスは、MPIO 可能デバイスとして構成すると、最大 2 つまでのアダプターをサポートします。

MPIO デバイスとしてパラレル SCSI デバイスをケーブルで 接続するには、次の例のような簡易構成を使用します。 以下は設定に必要な最小限の構成作業ですが、追加の構成作業が必要になる場合があります。

1. 電源オフして、2 つの SCSI アダプターをインストールします。
2. デバイスを両方の SCSI アダプターにケーブルで 接続します。
3. システムの電源をオンにします。

4. 片方のアダプターの 設定を固有の SCSI ID に変更します。 デフォルトでは SCSI アダプターの SCSI ID は「7」です。各 ID は固有でなければならないので、片方のアダプターを別の ID (例えば、「6」) に変更します。
5. **cfgmgr** コマンドを実行します。
6. 構成を検査するためにコマンド・ラインで次のように入力します。

```
lspath -l hdiskX
```

ここで、*X* は新規に構成したデバイスの論理番号です。 コマンド出力結果は 2 つのパスとその状況を表示します。

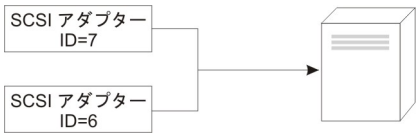


図 *13. MPIO SCSI* デバイスのケーブル構成

図は 2 つの SCSI アダプターから 同一のデバイスへの配線を示しています。

MPIO デバイスとしてのファイバー・チャネル・デバイスのケーブル接続
ファイバー・チャネル・デバイスは複数のアダプターにケーブルで 接続できます。ソフトウェアでの制限はありません。

MPIO デバイスとしてファイバー・チャネル・デバイスをケーブルで 接続するには、次の例のような簡易構成を使用します。 以下は設定に必要な最小限の構成作業ですが、追加の構成作業が必要になる場合があります。

1. 電源オフして、2 つのファイバー・チャネル・アダプターをインストールします。
2. アダプターをスイッチまたはハブにケーブル 接続します。
3. デバイスをスイッチまたはハブにケーブル 接続します。
4. システムの電源をオンにします。
5. 構成を検査するためにコマンド・ラインで次のように入力します。

```
lspath -l hdiskX
```

ここで、*X* は新規に構成したデバイスの論理番号です。 コマンド出力結果は、インストール済みの各アダプターごとに 1 つのパスとその状況を表示します。

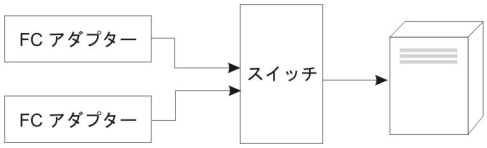


図 *14. MPIO* ファイバー・チャネル・デバイスのケーブル構成

MPIO デバイスの構成

MPIO 可能デバイスを構成する場合には、非 MPIO デバイスと同じコマンドを使用してください。

cfgmgr、**mkdev**、**chdev**、**rmdev** および **lsdev** の各コマンドは、MPIO デバイス・インスタンスの管理をサポートし、その属性を表示します。 MPIO デバイス・インスタンスは、そのデバイス・インスタンスに関連したパス・インスタンスをもっています。 コマンド **mkpath**、**chpath**、**rmpath**、および **lspath** は、パス・インスタンスを管理し、それらの属性を表示します。

パス・インスタンスは、デバイスを構成解除せずに MPIO デバイスへの追加または MPIO デバイスからの除去を行うことができます。

MPIO デバイスは追加の属性を保持できますが、すべての MPIO デバイスがサポートしなければならない必須属性は、`reserve_policy` および `algorithm` です。 `reserve_policy` 属性は、 デバイスのオープン時にデバイス・ドライバーがインプリメントする 予約方式のタイプを決定します。 この属性を使用して、

同一システム上でも別のシステム上でも、 他のアダプターからのデバイス・アクセスを制限することができます。 algorithm 属性は、デバイス用に構成された複数のパスにわたる入出力の管理に PCM が使用する方式を定義します。

サポートされるマルチパス・デバイス

AIX デフォルト PCM は、devices.common.IBM.mpio.rte ファイルセットで定義されたディスク・デバイスとテープ・デバイスのセットをサポートします。

AIX ディスク PCM またはテープ PCM によってサポートされないデバイスの場合は、デバイス・ベンダーが、そのデバイスを MPIO 可能デバイスとして認識するために必要な ODM、PCM、および他のサポート・コードで事前定義された属性を提供する必要があります。

どのディスク・デバイスが AIX ディスク PCM によってサポートされているかを判別するには、次のスクリプトを実行します。

```
odmget -qDvDr=aixdiskpcmke PdDv | grep uniquetype | while read line
do
      utype=`echo $line | cut -d'"' -f2`
      dvc=`odmget -q"uniquetype=$utype AND attribute=dvc_support" PdAt`
      echo $dvc | grep values | cut -d'"' -f2
done
```

どのテープ・デバイスが AIX ディスク PCM によってサポートされているかを判別するには、次のスクリプトを実行します。

```
odmget -qDvDr=aixtapepcmke PdDv | grep uniquetype | while read line
do
      utype=`echo $line | cut -d'"' -f2`
      dvc=`odmget -q"uniquetype=$utype AND attribute=dvc_support" PdAt`
      echo $dvc | grep values | cut -d'"' -f2
done
```

このスクリプトの出力には、AIX デフォルト PCM によってサポートされる固有のデバイス・タイプのリストが 表示されます。 AIX ディスク PCM によってサポートされる 3 つのデバイス・タイプは、自己構成パラレル *SCSI* ディスク (disk/scsi/scsd)、*MPIO other FC* ディスク (disk/fcp/mpioosdisk)、および *MPIO other iSCI* (disk/iscsi/mpioosdisk) です。 AIX テープ PCM によってサポートされるデバイス・タイプは *MPIO other FC* テープ (tape/fcp/mpioost) です。

MPIO other FC ディスクと *MPIO other FC* テープ・デバイスは、それぞれ他のファイバー・チャネル・ディスクとファイバー・チャネル・テープのサブセットです。 デバイスが *MPIO other FC* デバイスとしてサポートされるのは、ベンダー提供の ODM 事前定義属性が存在せず、かつデバイスが AIX デフォルト PCM のいずれかと一緒に作動することが認証済みである場合に限られます。 この認証は、デバイスが *MPIO other FC* デバイスとして構成されている場合にすべてのデバイス機能がサポートされることを保証するものではありません。

MPIO other iSCSI ディスクは、他の iSCSI ディスクのサブセットです。 ベンダー提供の ODM 事前定義属性が存在せず、デバイスが AIX PCM と一緒に作動することが認証済みである場合にのみ、デバイスは *MPIO other iSCSI* ディスクとしてサポートされます。 この認証は、デバイスが *MPIO other iSCSI* ディスクとして構成されている場合にすべてのデバイス機能がサポートされることを保証するものではありません。

どのデバイスが *MPIO other FC* ディスクとしてサポートされているかを判別するには、次のスクリプトを実行します。

```
odmget -quniquetype=disk/fcp/mpioosdisk PdAt | grep deflt | cut -d'"' -f2
```

どのデバイスが *MPIO other FC* テープとしてサポートされているかを判別するには、次のスクリプトを実行します。

```
odmget -q "uniquetype=tape/fcp/mpioosdisk AND attribute=mpio_model_map PdAt | grep deflt | cut -d'"' -f2
```

どのデバイスが *MPIO other iSCSI* ディスクとしてサポートされているかを判別するには、次のスクリプトを実行します。

```
odmget -quniquetype=disk/iscsi/mpioosdisk PdAt | grep deflt | cut -d'"' -f2
```

このスクリプトの出力は、 デバイスのベンダーおよびモデルが入っている照会データのリストを表示します。

システム上にインストールされるすべての MPIO 可能デバイスを表示するには、 次のスクリプトを実行します。

```
odmget -qattribute=unique_id PdAt | grep uniquetype | cut -d'"' -f2
```

このスクリプトの出力には、AIX デフォルト PCM とベンダー提供の PCM によってサポートされる固有の MPIO 可能デバイス・タイプのリストが表示されます。

MPIO デバイス属性

以下の属性は、マルチパス・デバイスによってのみサポートされます。 これらの属性は、SMIT、またはコマンド (特に **lsattr** および **chdev** コマンド) を使用して表示または変更することができます。

いくつかのマルチパス入出力 (MPIO) デバイス属性では、属性の並行更新が使用可能になっています。それらの属性値はディスクがオープンされて 使用中であるときでも更新でき、新しい値は即時に有効になります。一部の属性、特に reserve_policy 属性では、属性を変更できるタイミングや属性に指定できる新しい値について、制限が存在する場合があります。例えば、あるディスクがオープンされていて 、クラスター・リポジトリー・ディスクとして現在使用中である場合、そのディスクに対して予約ポリシーを設定しようとすると、他のクラスター・ノードがリポジトリーへのアクセスを失うことになるため、その試みは AIX オペレーティング・システムによってブロックされます。

すべての MPIO デバイスがサポートしなければならない必須デバイス属性は、reserve_policy です。 一般に、マルチパス・デバイスは PR_key_value デバイス属性も持っています。 マルチパス・デバイスは、追加のデバイス固有の属性をもつことができます。 その他のデバイス固有の属性は、次のとおりです。

FC3_REC
デバイスが、ファイバー・チャネルのクラス 3 を使用するエラー・リカバリーをオンにする必要があるかどうかを指定します。この機能を使用可能にすると、ファイバー・チャネルに関連する特定のタイプのファブリック・エラーについて、エラー検出とエラー・リカバリーの効率が向上します。この属性は、限られた一連のデバイスでのみ使用できます。この属性は、以下の値を持つことができます。

true
ファイバー・チャネルのクラス 3 を使用するエラー・リカバリーを使用可能にします。

false
ファイバー・チャネルのクラス 3 を使用するエラー・リカバリーを使用不可にします。

reserve_policy
デバイスのオープン 時に予約方式を使用するかどうかを定義します。 この値は次のとおりです。

no_reserve
予約方式をデバイスに適用しません。 デバイスは他のイニシエーターによってアクセスされる可能性があり、 それらのイニシエーターは他のホスト・システム上にある可能性があります。

single_path_reserve
SCSI2 予約方式をデバイスに適用します。 これは、このデバイスが予約方式を発行したイニシエーターによってのみアクセスできることを意味します。 このポリシーにより、同一ホストまたは他のホスト上の他のイニシエーターはこのデバイスにアクセスできなくなります。 このポリシーでは、SCSI2 予約ポリシーを使用して単一イニシエーター (パス) にデバイスをロックします。 他のいずれかのパスを使用して経路指定されたコマンドは、予約上の矛盾を起こします。

複数のパス間でコマンドを交替させるパス選択アルゴリズムを使用すると、single_path_reserve 値を選択したときにスラッシングが生じる可能性があります。 例えば、デバイス固有の PCM が複数のパスにわたって入出力を分散する値に設定される必須属性をもっているものと想定します。 single_path_reserve が有効なときには、 ディスク・ドライバーは、バス・デバイス・リセット (BDR) を出してから、 前の予約を破棄する次のコマンドを送信するための新規パスを使用して予約を発行する必要があります。 異なるパスが選択されるたびに、BDR の送信およびターゲット・デバイスへの予約の発行というオーバーヘッドが原因で、 スラッシングと、パフォーマンスの低下が生じます。 (AIX PCM は、ユーザーが、 スラッシングを引き起こす可能性があるアルゴリズムを選択することは認めません。)

PR_exclusive
デバイスのオープン時に SCSI3 永続予約、排他ホスト方式を適用します。 PR_key_value 属性値は、ホスト・システムごとに固有のものでなければなりません。 PR_key_value 属性を使用して、他のホスト・システムのイニシエーターがデバイスにアクセスできないようにします 。

PR_shared
デバイスのオープン時に SCSI3 永続予約、共用ホスト方式を適用します。 PR_key_value 値は、ホスト・システムごとに固有のものでなければなりません。 他のホスト・システムのイニシエーターは、デバイスにアクセスするためには、 登録が必要です。

PR_key_value
これは、デバイスがいずれかの永続予約ポリシー (PR_exclusive または PR_shared) をサポートする場合にのみ必要です。

パス制御モジュール属性

デフォルトの AIX デフォルト・パス制御モジュール (PCM) に加えて、デバイス固有の PCM がデバイス・ベンダーによって提供されることがあります。 ユーザー変更可能属性のセットは、デバイス・ベンダーによって定義されます。 デバイス固有の PCM は、デバイスとパスの属性をもつことができます。

AIX デフォルト PCM のデバイス属性は以下のとおりです。

algorithm
入出力がデバイス用の複数のパスにわたって分散されるときに使用する方式を決定します。 アルゴリズム属性は下記の値をもつことができます。

注 : 一部のデバイスは、これらの値のサブセットのみをサポートします。

failover
すべての入出力操作を単一のパスに送信します。パスに、障害あり、または使用不可のマークが付けられると、使用可能な次のパスが、すべての入出力操作を送信するために選択されます。このアルゴリズムは、path priority 属性の昇順の値に基づいて、使用可能なすべてのパスを順序付きのリストで管理します。各入出力操作に、最小の path priority 値を持つ有効なパスが選択されます。

round_robin
複数の使用可能なパスにわたって、入出力操作を分散させます。アクティブ・パスとパッシブ・パスを持つデバイス、および優先パスと非優先パスを持つデバイスの場合は、入出力操作にパスのサブセットだけが使用されます。パスに、障害あり、または使用不可のマークが付けられると、そのパスは入出力操作の送信に使用されなくなります。入出力操作は、path priority 属性に基づいて分散されます。より高い path priority 値を持つパスは、より多くの入出力操作の割り当てを受けます。

shortest_queue
複数の使用可能なパスにわたって、入出力操作を分散させます。 アクティブ・パスとパッシブ・パスを持つデバイス、および優先パスと非優先パスを持つデバイスの場合は、入出力操作にパスのサブセットだけが使用されます。このアルゴリズムは、round_robin アルゴリズムによく似ています。 ただし、shortest_queue アルゴリズムは、各パス上で保留中の入出力操作の数に基づいて入出力操作を分散させます。現在保留中の入出力操作の数が最も少ないパスが、次の操作に選択されます。アルゴリズムが shortest_queue に設定されていると、path priority 属性は無視されます。

hcheck_mode
ヘルス検査機能の使用時にどのパスを検査すべきかを決定します。 この属性は、以下のモードをサポートします。

enabled
使用可能状態にあるパスを介して **healthcheck** コマンドを送信します。このモードでは、**healthcheck** コマンドは使用不可状態および欠落状態にあるパスを介しては送信されません。

failed
障害あり状態のパスを介して **healthcheck** コマンドを送信します。このモードでは、**healthcheck** コマンドは使用可能状態、使用不可状態、および欠落状態にあるパスを介しては送信されません。

nonactive
(デフォルト) デバイスへのアクティブな 入出力がないパス (障害あり状態および使用可能状態のパスを含む) を介して **healthcheck** コマンドを送信します。このモードでは、**healthcheck** コマンドは使用不可状態および欠落状態にあるパスを介しては送信されません。

hcheck_interval
デバイス用の複数のパスに関してヘルス検査を実行する頻度を定義します。 この属性は、0 から 3600 秒までの範囲をサポートします。 値 0 が選択されると、 ヘルス検査は使用不可になります。

注: ヘルス検査は、ディスクが何らかのプロセスによって開かれたが、まだ閉じられていない場合にのみ実行されます。 ディスクを開いているエンティティーがない 場合、パス制御モジュールは、そのデバイスの **hcheck_interval** 属性がゼロ以外の値に設定されていてもパスを検査しません。

dist_tw_width
「時間ウィンドウ」の期間を定義します。 これは、分散エラー検出アルゴリズムがエラー付きで戻される入出力を累積する時間フレームです。 dist_tw_width 属性の計測単位は、 ミリ秒です。 この属性値を下げると、取られた各サンプルの 期間が減少し、 入出力エラーの小さなバーストに対する アルゴリズムの感度が減少します。 この属性値を増加すると、エラーの小さなバーストに対するアルゴリズムの感度とパスの失敗の可能性が増加します。

dist_err_percent
ローパフォーマンスのためにパスが失敗するまでにパス上で許可されているエラーを持つ「時間ウィンドウ」の パーセントを定義します。 dist_err_percent の範囲は、0 から 100 です。 属性をゼロ (0) に設定すると、分散エラー検出アルゴリズムは使用不可になります。 デフォルトの設定値は ゼロです。 分散エラー検出アルゴリズムは、デバイスをアダプターに 接続しているファブリックをエラー用にサンプリングします。 アルゴリズムは、エラーのあるサンプルのパーセンテージを 計算し、計算値が dist_err_percent 属性値より大きい場合はパスを失敗させます。

以下に、AIX PCM のパス属性を示します。

path priority
パスのリストに関するアルゴリズム・ポリシーの動作を変更します。

アルゴリズム属性値が failover の場合は、各パスはリスト内に保持されます。 このリスト内の順序は、最初に選択されるパス、および、 パスに障害が生じた場合は次に選択されるパスを決定します。 この順序は、パス優先順位属性の値によって決定されます。 優先順位 1 が最も高い優先順位です。 複数のパスが同じ優先順位の値をもつことができますが、 すべてのパスが同じ値をもつ場合は、選択は各パスがいつ構成されたかに基づいて行われます。

アルゴリズム属性値が round_robin の場合、path priority アルゴリズムは各パスに優先順位値を割り当てます。入出力操作のパスは、パス優先順位に応じて選択されます。したがって、より高い優先順位値を持つパスは、より多くの入出力操作に選択されます。すべてのパス優先順位が同じである場合、パスは均等に選択されます。

cntl_hcheck_int

ストレージ・ファブリック・トランスポート障害が検出されると、コントローラー・ヘルス検査シーケンスが開始されます。 **cntl_delay_time** 属性は、コントローラー・ヘルス検査シーケンスがアクティブになるときの最大期間 (秒数) を決定します。 コントローラー・ヘルス検査コマンドが、使用可能なパスを検出して正常に完了すると、コントローラー・ヘルス検査シーケンスは終了し、入出力の再開を許可します。 コントローラー・ヘルス検査シーケンスの最後に正常なパスが検出されていない場合、障害の起きたパスが戻されたことをデバイス・ヘルス・チェッカーが検出するまでは、そのデバイスへの保留中の入出力および後続の入出力はすべて失敗します。

コントローラー・ヘルス検査シーケンスがアクティブの 間は、**cntl_hcheck_interval** 属性は、次のセットのコントローラー・ヘルス検査コマンドが出されるまでの時間の長さ (秒数) を示します。 0 (使用不可) に設定されていない限り、**cntl_hcheck_interval** 属性は、**cntl_delay_time** より小さくなければなりません。 **cntl_delay_time** または **cntl_hcheck_interval** のどちらかが 0 に設定されている場合、この機能は使用不可です。

cntl_delay_time

ストレージ・ファブリック・トランスポート障害が検出されると、コントローラー・ヘルス検査シーケンスが開始されます。 **cntl_delay_time** 属性は、コントローラー・ヘルス検査シーケンスがアクティブ

になる最大期間 (秒数) を決定します。 コントローラー・ヘルス検査コマンドが、使用可能なパスを検出して正常に完了すると、コントローラー・ヘルス検査シーケンスは終了し、入出力の再開を許可します。 コントローラー・ヘルス検査シーケンスの最後に正常なパスが検出されていない場合、障害の起きたパスが戻されたことをデバイス・ヘルス・チェッカーが検出するまでは、そのデバイスへの保留中の入出力および後続の入出力はすべて失敗します。

コントローラー・ヘルス検査シーケンスがアクティブの 間は、**cntl_hcheck_interval** 属性は、次のセットのコントローラー・ヘルス検査コマンドが出されるまでの時間の長さ (秒数) を示します。 0 (使用不可) に設定されていない限り、**cntl_hcheck_interval** 属性は、**cntl_delay_time** より小さくなければなりません。 **cntl_delay_time** または **cntl_hcheck_interval** のどちらかが 0 に設定されている場合、この機能は使用不可です。

timeout_policy
コマンドのタイムアウトおよびトランスポート・エラーに対して、PCM の動作を調整します。 **timeout_policy** が **fail_path** または **disable_path** のいずれかに設定されていると、 マルチパス I/O (MPIO) デバイスへの一部のパスに断続的なストレージ・エリア・ネットワーク (SAN) のファブリック上の問題が発生した場合、性能低下が改善される可能性があります。 **timeout_policy** 属性は下記の値をもつことができます。

retry_path
パスにおける最初のコマンド・タイムアウトの発生は、直ちにパス障害の原因となることはありません。 トランスポートの問題でパスに障害が発生し、ヘルス・チェックによってリカバリーした場合、リカバリーされたパスはただちに使用できます。

fail_path
コマンド・タイムアウトの最初の発生でパスに障害が発生しました (このパスがパス・グループの最後のパスではないことを前提)。 トランスポートの問題で障害が発生したパスがリカバリーされた場合、そのパスで障害が発生することなく一定の時間が経過するまで、そのパスは読み取りまたは書き込みの入出力操作に使用できません。 このフィーチャーが有効な場合、読み取りまたは書き込み入出力がトランスポート・エラーからリカバリーされたパスに経路指定されるまでに、一定の遅延が発生する可能性があります。

>|

fail_ctrl
この設定によって、MPIO は優先コントローラーから非優先コントローラーに、より素早く切り替わります。 この設定を有効にしないと、MPIO が優先コントローラーから非優先コントローラーに切り替わる前に、優先コントローラーへのパスすべてでエラーが発生します。 この設定を有効にすると、優先コントローラーへの 2 つのパスでエラーが発生した場合に、MPIO は切り替わります。 この設定は、**fail_path** 設定に似ています。

|<

disable_path
コマンド・タイムアウトの最初の発生でパスに障害が発生しました (このパスがパス・グループの最後のパスではないことを前提)。 トランスポートの問題で障害が発生したパスがリカバリーされた場合、そのパスで障害が発生することなく一定の時間が経過するまで、そのパスは読み取りまたは書き込みの入出力操作に使用できません。 一定期間に引き続き複数のコマンド・タイムアウトが発生する場合、このパスは無効である可能性があります。 無効なパスは、次のアクションの 1 つを実行するまで無効の (使用できない) ままです。つまり、**chpath** コマンドを実行して無効なパスを有効にする、影響を受けたディスクを再構成する、またはシステムをリブートするかのいずれかです。

SAN 複製属性

AIX のマルチパス 入出力 (MPIO) をインストールする必要があり、デバイスは AIX パス制御モジュール (PCM) を使用している必要があります。 これらの属性は、 ストレージ・サブシステムが 提供する設定および機能に依存しています。

次の AIX 属性は、 ストレージ・サブシステムを 使用して複製される論理装置番号 (LUN) の動作と関連します。 これらの属性は、すべてのストレージ・サブシステムまたはマイクロコード ・レベルでサポートされていない可能性があります。 PowerHA® SystemMirror® のようなクラスタリング・ソフトウェアまたは高可

用性ソフトウェアをインストールして、クラスター内のノード間でストレージ・エリア・ネットワーク (SAN) の複製管理を調整します。 次の属性は、仮想入出力サーバー (VIOS) では変更できません。

san_rep_cfg
AIX オペレーティング・システムにおいて、対等通信リモート・コピー (PPRC) を使用する同期デバイスをどのように定義し、構成するかを決定します。 ディスク・インスタンスの unique_id 値 は、この属性によって影響を受け、属性値が変更されるとこの値も変更される場合があります。 **san_rep_cfg** 属性 は、ストレージ・サブシステム 上の PPRC デバイスの状態を変更しません。 属性は下記の値をもつことができます。

none [デフォルト]
PPRC を使用する同期デバイス内で、LUN を別の論理ディスク・インスタンスとして構成します。

new
PPRC を使用する同期デバイスを、単一の論理インスタンスとして定義し、構成します。 このデバイスは、既存のディスク・インスタンスが PPRC デバイスのどの LUN とも一致しない場合のみ 定義され、構成されます。

new_and_existing
PPRC を使用する同期デバイスを、単一の論理インスタンスとして定義し、構成します。 PPRC デバイスを表す論理ディスク・インスタンスが存在しない場合、 新しいディスク・インスタンスが定義されます。

migrate_disk
PPRC を使用する同期デバイスを、単一の hdisk インスタンスとして定義し、構成します。 また、このデバイス用に選択された論理ディスク・インスタンスを使用します。 この操作は、**san_rep_device** 属性が supported または detected のいずれかに設定されたデバイス上でのみサポートされます。 ターゲット・デバイスの **san_rep_device** 属性が supported に設定されている場合、 この操作が動作するのは、ディスクが最後に構成された後で SAN の複製がストレージ上に設定されている場合のみです。 デバイスがクラスター内のリポジトリー・ディスクとして使用されていない場合、ディスク上でこの操作がサポートされるのは、ディスクがオープンされており 、AIX オペレーティング・システムで使用さている場合です。 影響を受けるディスクの unique_id 値は、PPRC デバイスの ID を反映して更新されます。

revert_disk
PPRC 論理ディスク・インスタンスを使用している既存の同期デバイスを、非 PPRC デバイスの hdisk インスタンスに構成します。 この操作は、**san_rep_device** 属性が yes に設定されているデバイス上でのみサポートされます。 ターゲット・ディスク・インスタンスの論理デバイス名と特殊ファイルは変更されずにそのままで残されます。 SAN 複製デバイス用の 1 次 (ソース) LUN は、復元された hdisk インスタンス用に使用されます。 1 次 (ソース) LUN が見つからないまたはホストに対して不明の場合、2 次 (宛先) LUN は復元された hdisk インスタンス用に使用されます。 デバイスがクラスター内のリポジトリー・ディスクとして使用されていない場合、ディスク上でこの操作がサポートされるのは、ディスクがオープンされており 、AIX オペレーティング・システムで使用さている場合です。 影響を受けるディスクの unique_id 値は、LUN ID を反映して更新されます。

none 値、**new** 値、および **new_and_existing** 値は、 同じオブジェクト・データ・マネージャー (ODM) 固有タイプのすべてのデバイスに 対して、動作が更新されることを意味します。 **chdef** コマンドは、**none**、**new**、および **new_and_existing** の値を設定するために使用されます。 **chdef** コマンドは、指定した ODM 固有タイプのすべてのデバイスに 対して、 属性のデフォルト値を更新します。 また、**chdef** コマンドも、 すでに定義された既存のデバイス・インスタンスの属性値を更新します。 すでに構成済みのデバイスの場合、 **chdef** コマンドを実行すると、システムがリブートされるまで、またはそれらの デバイスが構成解除され再構成されるまで、変更は反映されません。 **chdef** コマンドには、クラス、サブクラス、および属性のタイプが必要です。 **san_rep_cfg** 属性の ODM 固有タイプを判断するには、lsattr -l hdisk# -F class,subclass,type コマンドを使用します。 次に例を示します。

```
lsdev -l hdisk0 -F class,subclass,type
disk,fcp,aixmpiods8k
chdef -a san_rep_cfg=none -c disk -s fcp -t aixmpiods8k
chdef -a san_rep_cfg=new -c disk -s fcp -t aixmpiods8k
chdef -a san_rep_cfg=new_and_existing -c disk -s fcp -t aixmpiods8k
```

migrate_disk 値および **revert_disk** 値は、 単一で指定されたデバイス・インスタンスの動作を更新することを意味します。 **chdev** コマンドは、指定されたデバイスの **migrate_disk** 値または **revert_disk** 値のいずれかを設定するために使用する必要があります。 **chdev** コマンドは、指定されたデバイスの値のみを更新します。 次に例を示します。

```
chdev -a san_rep_cfg=migrate_disk -l hdisk0
chdev -a san_rep_cfg=revert_disk -l hdisk0
```

san_rep_device
論理ディスク・インスタンスが SAN 複製デバイスとして定義されていることを示します。 この属性はディスク構成時に設定されます。また、ディスクが構成された後でデバイスの状態が変更されると、失効することがあります。 属性は下記の値をもつことができます。

no
デバイスは、AIX オペレーティング・システムで SAN 複製デバイスとして構成されていません。 このデバイスは、SAN 複製デバイスとして構成されるための要件を満たしていません。

supported
デバイスは、AIX オペレーティング・システムで SAN 複製デバイスとして構成されていません。 このデバイスは、SAN 複製デバイスとして構成されるための要件を満たしています。 ただし、現在はストレージ・サブシステム 上の SAN 複製デバイスとしてセットアップされていません。

detected
デバイスは、AIX オペレーティング・システムで SAN 複製デバイスとして構成されていません。 AIX オペレーティング・システムは、このデバイスが SAN 複製デバイスとして構成されるための要件を満たしており、現在、ストレージ・サブシステム 上の SAN 複製デバイスとして設定されていることを検出しました。

yes
デバイスは、AIX オペレーティング・システムで SAN 複製デバイスとして構成されています。

通信アダプターの取り外し

ホット・プラグ・アダプターを取り外したり、取り替えたりするには、その前にアダプターを 構成解除しなければなりません。

通信アダプターを 構成解除するには、次のタスクを実行します。

- 取り外しまたは取り替えようとするアダプターを 使用しているすべてのアプリケーションを終了する
- そのアダプターに 接続されているすべてのデバイスが識別され、停止していることを確認する
- 現在使用中のすべてのスロット、あるいは特定のアダプターに 占有されているスロットをリストする
- アダプターのスロットの 位置を確認する
- ネットワーク・インターフェース ・リストのインターフェース 情報を表示し、削除する
- アダプターを 使用不可にする

以下の手順で通信アダプターを 構成解除するには、**root** としてログインする必要があります。

イーサネット、トークンリング、FDDI、および ATM アダプターの構成解除

イーサネット、トークンリング、FDDI、または ATM アダプターを 構成解除するには、次のステップを実行します。

1. `lsslot -c pci` と入力して、システム装置内のすべてのホット・プラグ・スロットをリストし、その特性を表示する。
2. 次の例に示されている、該当する SMIT コマンドを入力してインストール済みのアダプターをリストし、システム装置内のすべてのデバイスの現在の状態を表示する。

項目	説明
smit lsdenet	イーサネット・アダプターをリストする
smit lsdtok	トークンリング・アダプターをリストする
smit ls_atm	ATM アダプターをリストする

次の命名規則は、さまざまなタイプのアダプターに 使用されています。

項目	説明
名前	**アダプターのタイプ**
atm0、atm1、...	ATM アダプター
ent0、ent1、...	イーサネット・アダプター
tok0、tok1、...	トークンリング・アダプター

3. 構成解除するアダプターを 使用しているすべてのアプリケーションを終了する。
   この手順を進めるには、システム上でネットワーク・ダンプ・ロケーションを使用不可にする必要があります。 ネットワーク・ダンプ・ロケーションを探し、使用不可にするには以下を実行します。
   a) コマンド・ラインから次のとおり入力する。

   ```
   smit dump
   ```

   b)「**Show Current Dump Devices** (現行ダンプ・デバイスの表示)」を選択する。
   c) すべての構成済みダンプ・デバイスがネットワーク・ロケーションを表示しているかどうかを検査する。
   表示していなければ、SMIT を終了させてステップ 4 へ進む。 ダンプ・デバイスをローカル・ロケーションに変更するには、「**取り消し**」を選択するか、または F3 キーを押して、次のステップに進む。
   d) 1 次ダンプ・デバイスがネットワーク・ロケーションを表示している場合は、「**Change the Primary Dump Device (1 次ダンプ・デバイスの変更)**」を選択してから、「**Primary dump device (1 次ダンプ・デバイス)**」フィールドでローカル・ロケーションを入力して、ローカル・ロケーションに変更する。
   e) 2 次ダンプ・デバイスがネットワーク・ロケーションを表示している場合は、「**Change the Secondary Dump Device (2 次ダンプ・デバイスの変更)**」を選択してから、「**Secondary dump device (2 次ダンプ・デバイス)**」フィールドでローカル・ロケーションを入力して、ローカル・ロケーションに変更する。
   f) 完了したら、「**了解**」をクリックするか、Enter (キー) を押す。
4. `netstat -i` と入力し、構成されているインターフェースすべてのリストを 表示し、アダプターが TCP/IP 用に構成されているかどうかを判別する。 次のような出力が表示されます。

   ```
   Name  Mtu    Network   Address         Ipkts  Ierrs  Opkts Oerrs Coll
   lo0   16896  link#1                     076      0    118     0    0
   lo0   16896  127       127.0.0.1        076      0    118     0    0
   lo0   16896  ::1                        076      0    118     0    0
   tr0    1492  link#2    8.0.5a.b8.b.ec   151      0    405    11    0
   tr0    1492  19.13.97  19.13.97.106     151      0    405    11    0
   at0    9180  link#3    0.4.ac.ad.e0.ad    0      0      0     0    0
   at0    9180  6.6.6     6.6.6.5            0      0      0     0    0
   en0    1500  link#5    0.11.0.66.11.1   212      0      1     0    0
   en0    1500  8.8.8     8.8.8.106        212      0      1     0    0
   ```

   トークンリング・アダプターは 1 つのインターフェースしか 持つことができません。 イーサネット・アダプターは、2 つのインターフェースを 持つことができます。 ATM アダプターは 複数のインターフェースを持つことができます。
5. 次の例に示されている、該当する **ifconfig** コマンドを入力し、ネットワーク・インターフェース・リストからインターフェースを 削除する。

項目	説明
`ifconfig en0 detach`	標準のイーサネット・インターフェースを 削除する
`ifconfig et0 detach`	IEEE 802.3 イーサネット・インターフェースを 削除する
`ifconfig tr0 detach`	トークンリング・インターフェースを 削除する
`ifconfig at0 detach`	ATM インターフェースを 削除する

6. 次の例に示されている、該当する **rmdev** コマンドを入力してアダプターを 構成解除し、そのデバイス定義をカスタマイズ・デバイス・オブジェクト・クラスに保持 する。

項目	説明
`rmdev -l ent0`	イーサネット・アダプターを 構成解除する
`rmdev -l tok1`	トークンリング・アダプターを 構成解除する
`rmdev -l atm1`	ATM アダプターを 構成解除する
`rmdev -p pci1`	PCI Bus の子とそれに関連する他のすべてのデバイスを構成解除する。ただし、カスタマイズ・デバイス・オブジェクト・クラスでのデバイス定義は保存しておく。

注 : アダプターを 構成解除し、カスタマイズ・デバイス・オブジェクト・クラスのデバイス定義を削除するには、**-d** フラグを指定して **rmdev** コマンドを使用することができます。

重要 : 目的がアダプターを取り外すことであり、それを置き換えることがない限り、ホット・プラグ操作には、**rmdev** コマンドに **-d** フラグを使用しない でください。

WAN アダプターの構成解除

WAN アダプターを 構成解除できます。

WAN アダプターを 構成解除するには、以下のようにします。

1. `lsslot -c pci` と入力して、システム装置内のすべてのホット・プラグ・スロットをリストし、その特性を表示する。
2. 次の例に示されている、該当する SMIT コマンドを入力してインストール済みのアダプターをリストし、システム 装置内のすべてのデバイスの現在の状態を表示する。

項目	説明
`smit 331121b9_ls`	2 ポート・マルチプロトコル WAN アダプターをリストする
`smit riciophx_ls`	ARTIC WAN アダプターをリストする

次の命名規則は、さまざまなタイプのアダプターに 使用されています。

Name	アダプターのタイプ
dpmpa	2 ポート・マルチプロトコル・アダプター
riciop	ARTIC960 アダプター

3. 構成解除するアダプターを 使用しているすべてのアプリケーションを終了する。

   この手順を進めるには、システム 上でネットワーク・ダンプ・ロケーションを使用不可にする必要があります。 ネットワーク・ダンプ・ロケーションを探し、使用不可にするには以下を実行します。

   a) コマンド・ラインから次のとおり入力する。

   ```
   smit dump
   ```

   b)「**Show Current Dump Devices** (現行ダンプ・デバイスの表示)」を選択する。

   c) すべての構成済みダンプ・デバイスがネットワーク・ロケーションを表示しているかどうかを検査する。

   表示していなければ、SMIT を終了させて、下記のステップ 5 に進めます。 ダンプ・デバイスをローカル・ロケーションに変更するには、「**取り消し**」を選択するか、または F3 キーを押して、次のステップに進む。

   d) 1 次ダンプ・デバイスがネットワーク・ロケーションを表示している場合は、「**Change the Primary Dump Device (1 次ダンプ・デバイスの変更)**」を選択してから、「**Primary dump device (1 次ダン**

プ・デバイス)」フィールドでローカル・ロケーションを入力して、ローカル・ロケーションに変更する。

e) 2 次ダンプ・デバイスがネットワーク・ロケーションを表示している場合は、「**Change the Secondary Dump Device (2 次ダンプ・デバイスの変更)**」を選択してから、「**Secondary dump device (2 次ダンプ・デバイス)**」フィールドでローカル・ロケーションを入力して、ローカル・ロケーションに変更する。

f) 完了したら、「**了解**」をクリックするか、Enter (キー) を押す。

4. 次の表のコマンドを使用して、これらのアダプターのデバイス・ドライバーとエミュレーター・ポートを構成解除し、取り外す。

Name	**2 ポート・マルチプロトコル・アダプター**
`smit rmhdlcdpmpdd`	デバイスを構成解除する
`smit rmsdlcscied`	SDLC COMIO エミュレーターを構成解除する

Name	**ARTIC960Hx PCI アダプター**
`smit rmtsdd`	デバイス・ドライバーを構成解除する
`smit rmtsdports`	MPQP COMIO エミュレーション・ポートを削除する

IBM 4 ポート 10/100 Base-TX イーサネット PCI アダプターの構成解除

4 ポート 10/100 Base-TX イーサネット PCI アダプターには 4 つのイーサネット・ポートがあり、アダプターを取り外すには、その前に各ポートを構成解除しなければなりません。

1. 「`lsslot -c pci`」と入力して、システム装置内のすべてのホット・プラグ・スロットをリストし、その特性を表示する。
2. 「`smit lsdenet`」と入力して、PCI サブクラスのデバイスをすべてリストする 。
   次のようなメッセージが表示されます。

```
ent1  Available 1N-00 IBM 4-Port 10/100 Base-TX Ethernet PCI Adapter (23100020) (Port 1)
ent2  Available 1N-08 IBM 4-Port 10/100 Base-TX Ethernet PCI Adapter (23100020) (Port 2)
ent3  Available 1N-10 IBM 4-Port 10/100 Base-TX Ethernet PCI Adapter (23100020) (Port 3)
ent4  Available 1N-18 IBM 4-Port 10/100 Base-TX Ethernet PCI Adapter (23100020) (Port 4)
```

3. 構成解除するアダプターを 使用しているすべてのアプリケーションを終了する。
   この手順を進めるには、システム上でネットワーク・ダンプ・ロケーションを使用不可にする必要があります。 ネットワーク・ダンプ・ロケーションを探し、使用不可にするには以下を実行します。

   a) コマンド・ラインから次のとおり入力する。

```
smit dump
```

   b) 「**Show Current Dump Devices** (現行ダンプ・デバイスの表示)」を選択する。

   c) すべての構成済みダンプ・デバイスがネットワーク・ロケーションを表示しているかどうかを検査する。
   表示していなければ、SMIT を終了させてステップ 4 へ進む。 ダンプ・デバイスをローカル・ロケーションに変更するには、「**取り消し**」を選択するか、または F3 キーを押して、次のステップに進む。

   d) 1 次ダンプ・デバイスがネットワーク・ロケーションを表示している場合は、「**Change the Primary Dump Device (1 次ダンプ・デバイスの変更)**」を選択してから、「**Primary dump device (1 次ダンプ・デバイス)**」フィールドでローカル・ロケーションを入力して、ローカル・ロケーションに変更する。

   e) 2 次ダンプ・デバイスがネットワーク・ロケーションを表示している場合は、「**Change the Secondary Dump Device (2 次ダンプ・デバイスの変更)**」を選択してから、「**Secondary dump device (2 次ダンプ・デバイス)**」フィールドでローカル・ロケーションを入力して、ローカル・ロケーションに変更する。

   f) 完了したら、「**了解**」をクリックするか、Enter (キー) を押す。

4. netstat -i と入力し、構成されているインターフェースすべてのリストを 表示し、アダプターが TCP/IP 用に構成されているかどうかを判別する。
   次のような出力が表示されます。

```
Name  Mtu    Network   Address         Ipkts  Ierrs  Opkts Oerrs Coll
lo0   16896  link#1                      076      0    118     0    0
lo0   16896  127       127.0.0.1         076      0    118     0    0
lo0   16896  ::1                         076      0    118     0    0
tr0    1492  link#2    8.0.5a.b8.b.ec    151      0    405    11    0
tr0    1492  19.13.97  19.13.97.106      151      0    405    11    0
at0    9180  link#3    0.4.ac.ad.e0.ad     0      0      0     0    0
at0    9180  6.6.6     6.6.6.5             0      0      0     0    0
en0    1500  link#5    0.11.0.66.11.1    212      0      1     0    0
en0    1500  8.8.8     8.8.8.106         212      0      1     0    0
```

   イーサネット・アダプターは、2 つのインターフェース (例えば、et0 と en0) を持つことができます。
5. **ifconfig** コマンドを使用して、ネットワーク・インターフェース・リストから各インターフェースを削除する。
   例えば、「iconfig en0 detach」と入力して、標準のイーサネット・インターフェースを 削除し、「iconfig et0」と入力して、IEEE 802.3 インターフェースを 削除します。
6. **rmdev** コマンドを使用してアダプターを 構成解除し、そのデバイス定義はカスタマイズ・デバイス・オブジェクト・クラスにそのまま保持する。
   例えば、「rmdev -l ent0」と入力します。

   注 : アダプターを 構成解除し、カスタマイズ・デバイス・オブジェクト・クラスのデバイス定義を削除するには、**-d** フラグを指定して **rmdev** コマンドを使用することができます。

   重要 : 目的がアダプターを取り外すことであり、それを置き換えることがない限り、ホット・プラグ操作には、**rmdev** コマンドに **-d** フラグを使用しない でください。

ATM アダプターの構成解除

アダプターを取り外すには、その前に、エミュレートされた LAN の各デバイスを構成解除しなければなりません。

従来の IP および LAN エミュレーション・プロトコルは、ATM アダプターで 実行することができます。 LAN エミュレーション・プロトコルを使用すると、ATM ネットワークを使用した、エミュレートされた LAN のインプリメンテーションを使用可能にできます。 エミュレートされた LAN は、Ethernet/IEEE 802.3、トークンリング /IEEE 802.5、または、MPOA (MultiProtocol Over ATM) のいずれでもかまいません。

LAN インターフェースを 削除するには、以下のようにします。

1. lsslot -c pci と入力して、システム装置内のすべてのホット・プラグ・スロットをリストし、その特性を表示する。
2. 「smit ls_atm」と入力して、すべての ATM アダプターをリストする 。
   次のようなメッセージが表示されます。

```
.
.
atm0 Available 04-04 IBM PCI 155 Mbps ATM Adapter (14107c00)
atm1 Available 04-06 IBM PCI 155 Mbps ATM Adapter (14104e00)
```

3. 「smit listall_atmle」と入力して、アダプター上の、エミュレートされた LAN のすべてのクライアントをリストする。
   次のようなメッセージが表示されます。

```
ent1 Available  ATM LAN Emulation Client (Ethernet)
ent2 Available  ATM LAN Emulation Client (Ethernet)
ent3 Available  ATM LAN Emulation Client (Ethernet)
tok1 Available  ATM LAN Emulation Client (Token Ring)
tok2 Available  ATM LAN Emulation Client (Token Ring)
```

   すべての ATM アダプター上で、エミュレートされた複数のクライアントを実行することができます。
4. 「smit listall_mpoa」と入力して、アダプター上の、エミュレートされた LAN のすべてのクライアントをリストする。

次のようなメッセージが表示されます。

```
mpc0 Available     ATM LAN Emulation MPOA Client
```

atm0 と atm1 は、物理 ATM アダプターです。 mpc0 は、エミュレートされた MPOA クライアントです。 ent1、ent2、ent3、tok1、および tok2 は、エミュレートされた LAN クライアントです。

5. 「entstat」と入力して、クライアントが実行されているアダプターを判別する。
次のようなメッセージが表示されます。

```
-------------------------------------------------------------
ETHERNET STATISTICS (ent1) :
Device Type: ATM LAN EmulationATM Hardware Address: 00:04:ac:ad:e0:ad
.
.
.
ATM LAN Emulation Specific Statistics:
------------------------------------
Emulated LAN Name: ETHelan3
Local ATM Device Name: atm0
Local LAN MAC Address:
.
.
```

6. 構成解除するアダプターを 使用しているすべてのアプリケーションを終了する。
この手順を進めるには、システム上でネットワーク・ダンプ・ロケーションを使用不可にする必要があります。 ネットワーク・ダンプ・ロケーションを探し、使用不可にするには以下を実行します。
   a) コマンド・ラインから次のとおり入力する。

   ```
   smit dump
   ```

   b) 「**Show Current Dump Devices** (現行ダンプ・デバイスの表示)」を選択する。
   c) すべての構成済みダンプ・デバイスがネットワーク・ロケーションを表示しているかどうかを検査する。
   表示していなければ、SMIT を終了させてステップ 7 へ進む。 ダンプ・デバイスをローカル・ロケーションに変更するには、「**取り消し**」を選択するか、または F3 キーを押して、次のステップに進む。
   d) 1 次ダンプ・デバイスがネットワーク・ロケーションを表示している場合は、「**Change the Primary Dump Device (1 次ダンプ・デバイスの変更)**」を選択してから、「**Primary dump device (1 次ダンプ・デバイス)**」フィールドでローカル・ロケーションを入力して、ローカル・ロケーションに変更する。
   e) 2 次ダンプ・デバイスがネットワーク・ロケーションを表示している場合は、「**Change the Secondary Dump Device (2 次ダンプ・デバイスの変更)**」を選択してから、「**Secondary dump device (2 次ダンプ・デバイス)**」フィールドでローカル・ロケーションを入力して、ローカル・ロケーションに変更する。
   f) 完了したら、「**了解**」をクリックするか、Enter (キー) を押す。
7. **rmdev -l** *device* コマンドを使用して、以下の順序でインターフェースを 構成解除する。
   - エミュレートされたインターフェース = en1、et1、en2、et2、tr1、tr2 ...
   - エミュレートされたインターフェース = ent1、ent2、tok1、tok2 ...
   - Multiprotocol Over ATM (MPOA) = mpc0
   - ATM アダプター = atm0
8. SCSI アダプター scsi1 とその children のすべてを構成解除するには次のように入力する。 (ただし、カスタマイズ・デバイス・オブジェクト・クラスでのデバイス定義は保存しておく。)

```
rmdev  -R scsi1
```

システムは、次のようなメッセージを表示します。

```
        rmt0 Defined
        hdisk1 Defined
        scsi1 Defined
```

9. SCSI アダプターそのものでなく、SCSI アダプターの children scsi1 のみを構成解除するには次のように入力する。 (ただし、カスタマイズ・デバイス・オブジェクト・クラスでのデバイス定義は保存しておく。)

```
rmdev  -p scsi1
```

システムは、次のようなメッセージを表示します。

```
        rmt0 Defined
        hdisk1 Defined
```

10. デバイス定義をカスタマイズ・デバイス・オブジェクト・クラスに保持しながら、PCI バス pci1 の子とその下の他のすべてのデバイスを構成解除するには、次のように入力します。

```
rmdev  -p pci1
```

システムは、次のようなメッセージを表示します。

```
         rmt0 Defined
         hdisk1 Defined
         scsi1 Defined
         ent0 Defined
```

ストレージ・アダプターの構成解除

ストレージ・アダプターを取り外したり、置き換えたりするには、その前にそのアダプターを 構成解除しなければなりません。

これらのタスクを行うには root ユーザーとしてログインする必要があります。

以下の手順を実行して、SCSI およびファイバー・チャネルのストレージ・アダプターを 構成解除します。

ストレージ・アダプターを 構成解除するには、次のタスクを実行します。

- 取り外し、置き換え、あるいは移動するアダプターを 使用しているすべてのアプリケーションを終了する
- ファイルシステムをアンマウントする
- そのアダプターに 接続されているすべてのデバイスが識別され、停止していることを確認する
- 現在使用中のすべてのスロット、あるいは特定のアダプターに 占有されているスロットをリストする
- アダプターのスロットの 位置を確認する
- 親および子のデバイスを使用不可にする
- アダプターを 使用不可にする

SAS アダプター、SCSI アダプター、NVMe アダプター、およびファイバー・チャネル・アダプターの 構成解除

ストレージ・アダプターは、通常、ディスクやテープ・ドライブなどのメディア・デバイスの親デバイスです。 親を取り外すには、それに接続されているすべての子を取り外すか、定義状態に置く必要があります。

SCSI およびファイバー・チャネルのアダプターを 構成解除するには、次のステップを実行します。

1. 構成解除するアダプターを 使用しているすべてのアプリケーションを終了する。
2. 「lsslot -c pci」と入力して、システム装置内のすべてのホット・プラグ・スロットをリストし、その特性を表示する。
3. 「lsdev -C」と入力して、システム装置内のすべてのデバイスの現在の状態をリストする。

4. 「umount」と入力して、このアダプターを使用して以前にマウントされたファイルシステム、ディレクトリー、またはファイルをアンマウントする。 追加情報については、JFS または JFS2 のマウントを参照してください。
5. 「rmdev -l adapter -R」と入力して、アダプターを使用不可にする。

重要 : ホット・プラグ操作には、**rmdev** コマンドに **-d** フラグを使用しない でください。使用した場合、構成が削除されます。

非同期アダプターの構成解除

非同期アダプターは構成解除できます。

これらのタスクを行うには root ユーザーとしてログインする必要があります。

以下に、非同期アダプターを構成解除する際のステップを示します。

非同期アダプターを取り外したり、置き換えたりするには、その前にアダプターを構成解除しなければなりません。 非同期アダプターを構成解除するには、次のタスクを実行します。

- 取り外し、置き換え、あるいは移動するアダプターを使用しているすべてのアプリケーションを終了する
- そのアダプターに接続されているすべてのデバイスが識別され、停止していることを確認する
- 現在使用中のすべてのスロット、あるいは特定のアダプターに占有されているスロットをリストする
- アダプターのスロットの位置を確認する
- 親および子のデバイスを使用不可にする
- アダプターを使用不可にする

手順

非同期アダプターを置き換えたり、取り外したりするには、その前に、そのアダプターと、そのアダプターによってコントロールされるすべてのデバイスを構成解除しなければなりません。 デバイスを構成解除するには、それらのデバイスを使用しているプロセスをすべて終了しなければなりません。 次のステップを実行します。

1. 構成解除するアダプターを使用しているすべてのアプリケーションを終了する。
2. 「lsslot -c pci」と入力して、システム装置内のすべてのホット・プラグ・スロットをリストし、その特性を表示する。
3. 「lsdev -C -c tty」と入力して、使用可能なすべての tty デバイスと、システム装置内のすべてのデバイスの現在の状態をリストする。
4. 「lsdev -C -c printer」と入力して、アダプターに接続されているすべてのプリンターとプロッターをリストする。
5. **rmdev** コマンドを使用して、アダプターを使用不可にする。

重要 : ホット・プラグ操作には、**rmdev** コマンドに **-d** フラグを使用しない でください。使用した場合、構成が削除されます。

入出力デバイスのトラブルシューティング

デバイス障害の原因は判別できます。

デバイス・ソフトウェアの検査

次の方法でデバイス・ソフトウェアの問題を解決します。

- エラー・ログの検査
- すべてのデバイスのリスト作成
- デバイスの状態の検査
- デバイスの属性の検査

- デバイスの属性の変更
- 他のアプリケーションでデバイスを 使用
- 新規デバイスの定義

エラー・ログの検査

エラー・ログを調べて、デバイス、そのデバイスのアダプター 、またはそのデバイスを使用しているアプリケーションに関するエラーが記録されているかどうかを確認します。 この検査の実行手順については、エラー・ログ機能を参照してください。 手順に従って実行を行った後に、ここに戻るようにしてください。

デバイスの障害は解決しましたか?

これまでの方法で問題を訂正できなかった場合は、次のステップ (**デバイスのリスト**) に進んで、すべてのデバイスをリストしてください。

デバイスのリスト

lsdev -C コマンドを使用して、定義済みまたは使用可能なデバイスをすべてリストします。 このコマンドは、システムに接続されているすべてのデバイスの特性を表示します。

デバイスがデバイス・リストにある場合は、次のステップ (**デバイスの状態の検査**) に進んで、デバイスの状態を検査してください。

デバイスがデバイス・リストにない場合は、新規デバイスを定義してください (下記の**新規デバイスの定義**を参照)。

デバイスの状態の検査

lsdev -C コマンドで生成されたリストから、問題のデバイスを見つけます。 デバイスが使用可能な状態かどうかを検査します。

デバイスが使用可能なな状態の場合は、次のステップ (**デバイス属性の検査**) に進んで、デバイス属性を検査してください。

デバイスが使用可能なな状態でない場合は、新規デバイスを定義してください (下記の**新規デバイスの定義**を参照)。

デバイス属性の検査

lsattr -E -l *DeviceName* コマンドを使用して、デバイスの属性をリストします。

lsattr コマンドは、システムに接続されているデバイスの属性の特性と、可能な属性値を表示します。正しい設定値については、そのデバイスの資料を参照してください。

デバイス属性の設定が正しい場合は、下記の**別のアプリケーションによるデバイスの 使用** を参照してください。

デバイス属性の設定が正しくない場合は、次のステップ、**デバイス属性の変更**に進んでください。

デバイス属性の変更

chdev -l *Name* **-a** *Attribute*=*Value* コマンドを使用して、デバイスの属性を変更します。 このコマンドを実行する前に、「コマンド・リファレンス 第 *1* 巻」を参照してください。

chdev コマンドは、**-l** *Name* フラグで指定されたデバイスの特性を変更します。

属性を変更してもデバイスの障害が訂正されない場合は、次のステップ、**別のアプリケーションによるデバイスの使用**に進んでください。

別のアプリケーションによるデバイスの使用

デバイスを他のアプリケーションで使用してみます。 デバイスが他のアプリケーションで正しく動作する場合は、問題の原因は、前のアプリケーションにある可能性があります。

デバイスが他のアプリケーションで正しく動作する場合は、問題の原因は、前のアプリケーションにある可能性があります。 ソフトウェアのサービス技術員に問題を報告してください。

デバイスが他のアプリケーションでも正しく動作しなかった場合は、次のステップ、**新規デバイスの定義**に進んでください。

新規デバイスの定義

注 : mkdev コマンドを使用するには、root ユーザー権限を所有しているか、またはセキュリティー・グループのメンバーでなければなりません。

mkdev コマンドを使用して、システムにデバイスを 追加します。

mkdev コマンドは、新規のデバイスを定義し、使用可能にしたり、定義済みのデバイスを使用可能にします。 **-c**、**-s**、および **-t** フラグを適切に組み合わせて使用することによって、定義済みのデバイスを一意的に識別することができます。 このコマンドを実行する前に、「コマンド・リファレンス 第 *3* 巻」を参照してください。

デバイスを定義しても障害が解決しない場合は、作業を中止してサービス技術員に障害を報告するか、診断プログラムを使用してデバイスをテストしてください。

デバイス接続の検査

デバイス接続を検査するには、次のステップを実行します。

1. コンセントを介して電気が流れていることを確認する。
2. デバイスの電源ケーブルが、デバイスとコンセントに正しく接続されていることを確認する。
3. デバイスのシグナル・ケーブルが、デバイスとシステム 装置に正しく接続されていることを確認する。
4. SCSI デバイスの場合は、SCSI ターミネーターが正しく接続されており、SCSI アドレスが正しく設定されていることを確認する。
5. 通信デバイスの場合には、デバイスが通信回線に正しく接続されていることを確認する。
6. デバイスの電源がオンになっていることを確認する。

配線と構成手順、および、さらに詳しいトラブルシューティング 情報については、デバイスの資料を参照してください。

このトピックのステップで問題が修正できない 場合、次のステップに行きます。

アダプター取り外し問題の解決

rmdev コマンドを使用してアダプターを 構成解除する際にデバイスがオープンになっていると 、エラー・メッセージを受け取ることがあります。

rmdev コマンドを使ってアダプターを 構成解除するときに、次のようなメッセージが表示される場合は、デバイスがオープンになっていることを 示しており、取り外すか置き換えようとしているアダプターに、アプリケーションがアクセスし 続けていることが考えられます。

```
#rmdev -l ent0
Method error (/usr/lib/methods/ucfgent):
        0514-062
    Cannot perform the requested function because the
    specified device is busy.
```

この問題を解決するには、そのアダプターを 使用し続けているアプリケーションを特定して、それをクローズしなければなりません。 該当するアプリケーションには、次のものがあります。

- TCP/IP
- SNA
- OSI
- IPX/SPX
- Novell NetWare
- Streams

- 汎用データ・リンク制御 (GDLC)
  - IEEE イーサネット DLC
  - トークンリング DLC
  - FDDI DLC

システム・ネットワーク・アーキテクチャー・アプリケーション

アダプターを 使用し続けている SNA アプリケーションには以下のものがあります。

- DB2®
- TXSeries® (CICS® & Encina)
- DirectTalk
- MQSeries®
- HCON
- ADSM

ストリーム・アプリケーション

アダプターを 使用している可能性のあるストリーム・ベースのアプリケーションの 一部を以下に示します。

- IPX/SPX
- Novell NetWare V4 および Novell NetWare Services 4.1
- このオペレーティング・システムの接続と NetBios

WAN アダプターで 実行されるアプリケーション

WAN アダプターを 使用している可能性のあるアプリケーションを以下に示します。

- SDLC
- Bisync
- ISDN

TCP/IP アプリケーション

インターフェース・レイヤーを使用している TCP/IP アプリケーションはすべて、**ifconfig** コマンドで切り離すことができます。 これにより、TCP/IP を使用するアプリケーションはタイムアウトになり、インターフェースがダウンしたという 警告がユーザーに出されます。 アダプターの 追加または置き換えの後、**ifconfig** コマンドを実行してインターフェースを 接続すると、アプリケーションが再開します。

デバイスの作動可能状態の検査

デバイスが作動可能状態かどうかを調べるために検査できます。

デバイスが作動可能状態になっているかどうかを判別するには、次の手順を実行します。

1. デバイスの作動可能表示がオンになっていることを確認する。
2. テープ、ディスケット、光ディスク装置などの取り外し可能メディアが正しく挿入されていることを確認する。
3. プリンターやプロッターのリボン、用紙、およびトナーが切れていないことを確認する。
4. デバイスに書き込もうとしている場合は、書き込み先のメディアが書き込み可能になっていることを確認する。

これらの確認によって、デバイスの障害は解決しましたか? デバイスの作動可能状態の検査によって障害を訂正できなかった場合は、次のステップに進んでください。

デバイスの診断

デバイスの障害の有無を判別する場合は、ご使用のハードウェア診断を実行します。

ハードウェア診断でデバイスの障害が見つからない場合は、デバイスのソフトウェアを 検査してください。 デバイスが診断テストに合格した場合は、そのデバイスが使用するシステム・ソフトウェアで作動す

る方法に問題がある可能性があります。 障害を再現できる場合は、ソフトウェア・サービス部門に問題を報告してください。

ターゲット・デバイス構成

cfgmgr コマンドは、接続オプションとしての **-c** フラグとともに、限られた有効範囲での入出力デバイスのターゲット構成に使用できます。

FC デバイスおよび FCoE デバイスのターゲット構成

cfgmgr -c オプションは、ファイバー・チャネル (FC) アダプターおよび Fibre Channel over Ethernet (FCoE) アダプターを使用したターゲット構成に使用されます。

cfgmgr コマンドは、接続オプションとしての **-c** フラグとともに、限られた有効範囲でのデバイス構成に使用できます。 FC アダプターおよび FCoE アダプターの場合、構文は以下のとおりです

```
cfgmgr -l fscsi0 -c "parameter=val[,parameter=val,...]"
```

接続フィルター・ストリングを使用することにより、以下の 1 つ以上のパラメーターを使用して、デバイス・ディスカバリーの有効範囲を制限できます。

表 9. ***cfgmgr -c*** フラグのパラメーター

パラメーター名	説明
ww_name	ターゲット・デバイスの World Wide Port Name
node_name	ターゲット・デバイスの World Wide Node Name
scsi_id	Fibre Channel Protocol (FCP) ストレージ・デバイスの Small Computer System Interface (SCSI) ID へマップされる、ターゲット・デバイスの N_Port ID
lun_id	論理装置番号 (LUN)

例えば、次のコマンドは、World Wide Port Name が 0x5001738000330191 であるストレージ・ターゲット・ポートに、**lun_id** が 0x1000000000000 である単一の LUN を構成します。

```
# cfgmgr -l fscsi0 -c "ww_name=0x5001738000330191,lun_id=0x1000000000000"
```

このスキャンは、`fscsi0` ホスト・アダプター・ポートについてのみ発生します。

注意 :

- パラメーター値の中の先行文字 `0x` は、オプションです。
- すべてのパラメーターを 16 進数として表す必要があります。

次の例では、パラメーターが 1 つだけ指定されています。

```
# cfgmgr -l fscsi0 -c "lun_id=0x1000000000000"
```

このコマンドは、ストレージ・エリア・ネットワーク (SAN) 上にあるすべてのストレージ・デバイス・ポートをスキャンし、この LUN が存在する各 SAN ターゲット・ポートに対して、この単一の論理装置を構成します。

接続フィルター・パラメーターのガイドラインと規則

接続フィルター・パラメーターを使用するときは、以下の点を考慮してください。

- FC デバイスおよび FCoE デバイスのターゲット構成は、スイッチ接続環境にのみ適用されます。ターゲット・ポートに直接接続される接続ストリングを指定した場合、接続は失敗し、子デバイスが検出できないことを示すメッセージが表示されます。

- **-c** フラグは、有効範囲を一度に 1 つの fscsiX デバイスだけに制限する、**cfgmgr** コマンドの **-l** フラグを一緒に指定した場合にのみサポートされます。
- **cfgmgr** コマンドの **-c** フラグに、接続ストリングとして **-?** を **-v** フラグとともに指定した場合は、使用方法に関する情報が表示されます。
- 重複するパラメーターを指定した場合 (例えば、**lun_id** を 2 回リストした場合)、結果はエラーになります。デバイスは検出されません。
- 重複しない限り、**lun_id**、**scsi_id**、**ww_name**、および **node_name** の各パラメーターの任意の組み合わせを指定できます。構成する LUN、ターゲット、またはストレージのノードを一意的に識別するためには、パラメーターを 1 つか、なるべくなら 2 つ指定する必要があります (もっと多く指定することもできます)。以下のリストは、LUN、ターゲット、またはストレージのノードを一意的に識別するために必要なパラメーター、またはパラメーターの組み合わせを示しています。
  - **ww_name** パラメーターと **lun_id** パラメーターは、構成するターゲット・ポート上の LUN を一意的に識別します。
  - **scsi_id** パラメーターと **lun_id** パラメーターは、構成するターゲット・ポート上の LUN を一意的に識別します。
  - **node_name** パラメーターと **lun_id** パラメーターは、特定のストレージ・ノードのすべてのターゲット・ポート用に 1 つの LUN を構成します。 これらのパラメーターがターゲット・ポートを構成できるのは、すべてのターゲット・ポートが同じ **node_name** パラメーターを持つ場合に限られます。これは、一部のストレージ・デバイスに当てはまります。
  - **ww_name** パラメーターは、特定のターゲットのすべての LUN を構成します。
  - **node_name** パラメーターは、特定のストレージ・ノードのすべてのターゲット・ポートを構成します (すべてのターゲット・ポートが同じ **node_name** パラメーターを持つ場合に限られます。これは、一部のストレージ・デバイスに当てはまります)。
  - **lun_id** パラメーターは、その fscsi デバイスから可視であるすべてのターゲット・ポート上に 1 つの LUN を構成します。
- 3 つ以上のパラメーターを指定した場合、デバイス構成コードは、その追加情報を使用してデバイスの位置を検証します。 指定したパラメーター値のいずれかが、SAN 上で報告された値と競合する場合、コマンドは失敗し、デバイスは構成されません。

テープ・ドライブ

ここで説明するシステム管理機能は、テープ・ドライブに関連します。

これらの機能の多くは、ご使用のシステムのデバイスに関する情報が入っているデバイス構成データベースから、情報の変更または獲得を行います。 デバイス構成データベースは、システム上でサポートされる可能なかぎりのタイプのデバイスに関する情報を収めている事前定義構成データベースと、現在システム上にある特定のデバイスの情報を収めているカスタマイズ構成データベースとからなっています。 オペレーティング・システムがテープ・ドライブまたはその他のデバイスを利用する場合は、デバイスをカスタマイズ構成データベースで定義し、デバイス・タイプを事前定義の構成データベースで定義している必要があります。

テープ・ドライブの属性

これらのテープ・ドライブ属性は、システムのニーズに 合わせて調整できます。

これらの属性は、SMIT、またはコマンド (特に **lsattr** および **chdev** コマンド) を使用して表示または変更することができます。

各タイプのテープ・ドライブは、すべての属性のサブセットだけを使用します。

各属性に関する一般的な情報

ブロック・サイズ
ブロック・サイズ属性は、テープの読み取り、あるいはテープへの書き込みに使用するブロック・サイズを示します。 データは、データ・ブロック単位でテープに書き込まれ、ブロックとブロックの間に

はレコード間ギャップがあります。 ラージ・レコードは、フォーマット設定されていないテープへの書き込み時に便利です。 それは、そのテープ全体に生じるレコード間ギャップの数が少なくなって、より多くのデータを書き込むことが可能になるからです。 **0** の値は、可変長ブロックを示します。 許容値およびデフォルト値は、テープ・ドライブによって異なります。

デバイス・バッファー
デバイス・バッファー属性を (**chdev** コマンドを使用して) mode=yes に設定すると、データがテープ・ドライブのデータ・バッファーに転送された後でアプリケーションに書き込み完了が通知されますが、必ずしも、実際にデータがテープに書き込まれた後に通知されるとは限りません。 mode=no を指定すると、実際にデータがテープに書き込まれた後に書き込み完了がアプリケーションに通知されます。 この属性が値 mode=no に設定されている場合、ストリーム・モードで読み取りや書き込みを行うことはできません。 デフォルト値は mode=yes です。

値として mode=no を指定すると、テープ・ドライブの速度は低下しますが、電源異常やシステム障害が発生した場合でも完全度の高いデータを保持することができるため、メディア終端条件をより適切に処理することが可能になります。

拡張ファイル・マーク
拡張ファイル・マーク属性 (**chdev** コマンドの場合は extfm 属性) を値 no に設定すると、ファイル・マークの書き込み時に、必ず、正規のファイル・マークがテープに書き込まれます。 この属性を値 yes に設定すると、拡張ファイル・マークが書き込まれます。 テープ・ドライブの場合、この属性をオンに設定することができます。 デフォルト値は no です。例えば、8 mm のテープ・ドライブの拡張ファイル・マークには 2.2MB 分のテープが使用され、書き込みには最大 8.5 秒を要します。 正規のファイル・マークには 184KB が使用され、書き込みには約 1.5 秒を要します。

追加モードで 8 mm テープを使用する場合は、エラーを減少させるため、反転操作の後にファイル・マークで正しく位置決めできるように拡張ファイル・マークを使用してください。

緩み取り
緩み取り 属性 (**chdev** コマンドの場合は ret 属性) を ret=yes に設定すると、テープ・ドライブは、テープの挿入時やドライブのリセット時に必ずテープの緩み取りを自動的に行います。 緩み取り とは、テープ全体のテンションを一定にするために、テープの終わりまで巻き終えてから、再度、テープの先頭まで巻き戻すことを意味します。 テープを緩み取り すると、エラーを少なくすることができますが、このアクションには数分かかる場合があります。 値 ret=no を指定すると、テープ・ドライブは自動的にテープの緩み取りを行いません。 デフォルト値は yes です。

#1 に設定中の記録密度および #2 に設定中の記録密度
#1 に設定中の記録密度 (**chdev** コマンドの場合は density_set_1 属性) は、スペシャル・ファイル /dev/rmt*、/dev/rmt*.1、/dev/rmt*.2、および /dev/rmt*.3 を使用している場合のテープ・ドライブの書き込み密度を設定します。 #2 に設定中の記録密度 (**chdev** の場合は density_set_2 属性) は、スペシャル・ファイル /dev/rmt*.4、/dev/rmt*.5、/dev/rmt*.6、および /dev/rmt*.7 を使用している場合のテープ・ドライブの書き込みの密度値を設定します。 詳しくは、213 ページの『テープ・ドライブ用のスペシャル・ファイル』を参照してください。

密度設定値は、**0** から **255** の範囲の 10 進数で表します。 ゼロ (**0**) という設定値を指定すると、テープ・ドライブのデフォルトの密度が選択されます。 これは、通常、ドライブの高密度設定値です。 許可されている特定値とその意味は、テープ・ドライブのタイプによって異なります。 これらの属性は、テープ・ドライブによってサポートされているさまざまな密度で作成されたテープを、テープ・ドライブが読み取る機能には影響しません。 普通、#1 に設定中の記録密度をそのテープ・ドライブで可能な最高密度に設定し、#2 に設定中の記録密度をそのテープ・ドライブで可能な 2 番目に高い密度に設定します。

予約サポート
予約属性 (**chdev** コマンドの場合は res_support 属性) を使用するテープ・ドライブの場合、値 res_support=yes を指定すると、テープ・ドライブは、それがオープンされている間、SCSI バス上に予約されます。 複数の SCSI アダプターが磁気テープ装置を共用している場合、これによって、そのデバイスがオープンしている 間 1 つのアダプターだけがアクセスするようにできます。 SCSI テープ・ドライブの中には、予約または解放コマンドをサポートしていないものがあります。 この属性に値が事前定義されているため、予約または解放コマンドを常にサポートする SCSI テープ・ドライブもあります。

可変長ブロック・サイズ
可変長ブロック・サイズ属性 (**chdev** コマンドの場合は var_block_size 属性) は、テープ・ドライブが可変長レコードの書き込み時に必要とするブロック・サイズを指定します。 一部の SCSI テープ・ドライブでは、可変長レコードを書き込む場合でも、ゼロ以外のブロック・サイズをそのモード選択データに指定する必要があります。 可変長レコードを指示する場合は、ブロック・サイズ属性を **0** に設定します。 これが必要条件か否かを判別するには、特定のテープ・ドライブの『SCSI 仕様』を参照してください。

データ圧縮
データ圧縮属性 (**chdev** コマンドの場合は compress 属性) を compress=yes に設定すると、テープ・ドライブは圧縮モードになります。 ただし、これはそのドライブがデータを圧縮できる場合に限ります。 圧縮モードになると、ドライブはデータを圧縮フォーマットでテープに書き込むので、1 つのテープにより多くのデータが収められます。 この属性を no に設定すると、テープ・ドライブは、ネイティブ・モード (非圧縮モード) での書き込みを強制されます。 この属性を設定しても、読み取り操作には何の影響もありません。 デフォルト設定値は yes です。

オートローダー
オートローダー属性 (**chdev** コマンドの場合は autoload 属性) を autoload=yes に設定すると、ドライブにオートローダーが装備されている場合は、オートローダーがアクティブになります。 この場合に、他のテープがそのローダーで使用可能であれば、読み取りまたは書き込み操作がそのテープを最後まで進めると、それらの操作は自動的に次のテープに継続されます。 1 つのテープ・カートリッジだけに限られるテープ・ドライブ・コマンドはこの影響を受けません。 デフォルト設定値は yes です。

再試行遅延
再試行遅延属性は、コマンドが失敗してから、そのコマンドを再発行するまでにシステムが待つ秒数を設定します。 システムは、最高 4 回まで失敗したコマンドを再実行することができます。 この属性は、タイプが OST のテープ・ドライブにのみ適用されます。 デフォルト設定値は 45 です。

読み取り/書き込みのタイムアウト
読み取り/書き込みのタイムアウト属性または READ/WRITE の最大遅延属性は、読み取りまたは書き込みコマンドが完了するまでの時間としてシステムが許容できる最大秒数を設定します。 この属性は、タイプが OST のテープ・ドライブにのみ適用されます。 デフォルト設定値は 144 です。

テープ交換でのエラー・リターン
テープ交換またはリセット時の戻りエラー属性が設定されている場合、テープ・ドライブがリセットされるか、テープが交換されると、オープン時にエラーが戻されます。 テープ・ドライブに対する直前の操作の結果、クローズ時のテープが、テープ開始点をすぎたところに位置づけられたままになっている可能性があります。 戻されるエラーは -1 で、errno は EIO に設定されます。 アプリケーションに戻されると、エラー状態はクリアされます。 また、テープ・ドライブ自体を再構成すると、エラー状態はクリアされます。

2.0 GB 4 mm テープ・ドライブ (タイプ 4mm2gb) の属性
以下は 2.0 GB 4 mm テープ・ドライブ (タイプ 4mm2gb) の属性です。

ブロック・サイズ
デフォルト値は 1024 です。

デバイス・バッファー
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

固定値を持つ属性
テープ・ドライブが 2GB 4 ミリ・テープ装置として構成されている場合は、緩み取り、予約のサポート、可変長ブロック・サイズ、#1 に設定中の記録密度、および #2 に設定中の記録密度の各属性に、変更不可能な値が事前定義されています。 密度設定値は、テープ・ドライブが常に 2.0GB モードで書き込みを行うために事前定義されます。

4.0GB 4mm テープ・ドライブ (タイプ 4mm4gb) の属性
以下は 4.0GB 4mm テープ・ドライブ (タイプ 4mm4gb) の属性です。

ブロック・サイズ
デフォルト値は 1024 です。

デバイス・バッファー
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

#1 に設定中の記録密度および #2 に設定中の記録密度
ユーザーはこのドライブの密度設定値を変更することはできません。つまり、そのデバイスは、次のようにして、インストールされているデジタル・データ記憶装置 (DDS) のメディア・タイプに応じてそれ自身を自動的に再構成します。

メディア・タイプ	デバイス構成
DDS	読み取り専用。
DDS ||||	2.0GB モードでの読み取り/書き込みのみ。
DDS2	いずれかの密度での読み取り。4.0GB モードのみの書き込み。
非 DDS	サポートされていません。カートリッジはイジェクトされます。

データ圧縮
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

固定値を持つ属性
テープ・ドライブが 4.0 GB 4 mm テープ・ドライブとして構成されている場合、緩み取り、予約サポート、可変長ブロック・サイズ、#1 に設定中の記録密度、および #2 に設定中の記録密度の各属性には、変更できない事前定義値があります。

2.3GB 8mm テープ・ドライブ (タイプ 8mm) の属性
以下は 2.3 GB 8 mm テープ・ドライブ (タイプ 8mm) の属性です。

ブロック・サイズ
デフォルト値は 1024 です。 これより値が小さいと、テープに保管されるデータの量が減ってしまいます。

デバイス・バッファー
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

拡張ファイル・マーク
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

固定値を持つ属性
テープ・ドライブが 2.3GB 8 ミリ・テープ機構/装置として構成されている場合は、緩み取り、予約のサポート、可変長ブロック・サイズ、データ圧縮、#1 に設定中の記録密度、および #2 に設定中の記録密度の各属性に、変更不可能な値が事前定義されています。 密度設定値は、テープ・ドライブが常に 2.3GB モードで書き込みを行うために事前定義されます。

5GB 8mm テープ・ドライブ (タイプ 8mm5gb) の属性
以下は 5GB 8mm テープ・ドライブ (タイプ 8mm5gb) の属性です。

ブロック・サイズ
デフォルト値は 1024 です。 テープが 2.3GB モードで書き込まれる場合、値がこれより小さいと、テープに保管されるデータの量が減ってしまいます。

デバイス・バッファー
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

拡張ファイル・マーク
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

#1 に設定中の記録密度と #2 に設定中の記録密度
以下の設定値が適用されます。

設定値	意味
140	5GB モード (圧縮可能)
21	5GB モード非圧縮テープ

設定値	意味
20	2.3GB モード
0	デフォルト (5.0GB モード)

デフォルト値は、#1 に設定中の記録密度の場合は 140 で、#2 に設定中の記録密度の場合は 20 です。#1 に設定中の記録密度または #2 において、値が 21 であると、ユーザーは 5GB モードで非圧縮テープの読み取りまたは非圧縮テープへの書き込みを許可されます。

データ圧縮
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

固定値を持つ属性
テープ・ドライブが 5.0 GB 8 mm テープ・ドライブとして構成されている場合、緩み取り、予約サポート、および可変長ブロック・サイズの各属性には、変更できない事前定義値があります。

20000MB 8mm テープ・ドライブ (自己構成) の属性
以下は 20000MB 8mm テープ・ドライブ (自己構成) の属性です。

ブロック・サイズ
デフォルト値は 1024 です。

デバイス・バッファー
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

拡張ファイル・マーク
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

#1 に設定中の記録密度と #2 に設定中の記録密度
このドライブは 20.0GB フォーマットでデータ・カートリッジの読み取り、あるいはデータ・カートリッジへの書き込みを行うことができます。 読み取りコマンドの実行中、このドライブは、どのフォーマットがテープに書き込まれるかを自動的に決定します。 書き込み中は、密度設定値が、どのデータ・フォーマットがテープに 書き込まれるかを決定します。

以下の設定値が適用されます。

設定値	意味
39	20GB モード (圧縮可能)
0	デフォルト (20.0GB モード)

#1 に設定中の記録密度と #2 のデフォルト値は 39 です。

データ圧縮
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

固定値を持つ属性
テープ・ドライブが 20.0 GB 8 mm テープ・ドライブとして構成されている場合、緩み取り、予約サポート、および可変長ブロック・サイズの各属性には、変更できない事前定義値があります。

35GB テープ・ドライブ (タイプ 35gb) の属性
以下は 35 GB テープ・ドライブ (タイプ 35gb) の属性です。

ブロック・サイズ
IBM 7205 モデル 311 のスループットは、ブロック・サイズに依存します。 このドライブに推奨できる最小ブロック・サイズは、32 KB です。ブロック・サイズが 32 KB に満たない場合、データ転送速度 (バックアップまたはリストアの時間) が制限されます。以下の表は、コマンドごとの推奨ブロック・サイズを示したものです。

サポートされるコマンド	デフォルト・ブロック・サイズ (バイト)	推奨ブロック・サイズ
BACKUP	32K または 51.2K (デフォルト)	バックアップが名前によるか否かによって、32 K または 51.2 K のいずれかを使用します。変更は必要ありません。
TAR	10K	512 KB ブロック・サイズと示している資料にエラーがあります。**Blocking** パラメーターを **-N64** に設定。
MKSYSB	**BACKUP** を参照してください。	**MKSYSB** コマンドは **BACKUP** コマンドを使用します。 変更は必要ありません。
DD	該当なし	**Blocking** パラメーターを **bs=32K** に設定。
CPIO	該当なし	**Blocking** パラメーターを **-C64** に設定。

注 : ブロック・サイズを選択する場合には、容量およびスループットを知っておく必要があります。 ブロック・サイズが小さい場合、パフォーマンスには大きな影響を与えますが、容量にはほとんど影響しません。 推奨サイズより小さいブロック・サイズを使用すると、2.6GB フォーマット (密度) と 6.0GB フォーマット (密度) の容量に影響を与えます。 例えば、1024 バイトのブロック・サイズを使用して 32 GB のデータをバックアップするには、約 22 時間かかります。32 KB のブロック・サイズを使用して同じ 32 GB のデータをバックアップする場合は、約 2 時間で済みます。

デバイス・バッファー
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

拡張ファイル・マーク
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

#1 に設定中の記録密度と #2 に設定中の記録密度
次の表は、IBM 7205-311 テープ・ドライブに対してサポートされているデータ・カートリッジ・タイプと密度設定値 (10 進法と 16 進法) を示します。 リストア操作 (読み取り) を実行するとき、テープ・ドライブは、書き込まれた密度と一致するように自動的に密度を設定します。バックアップ (書き込み) 操作を行うときは、使用するデータ・カートリッジと一致するように密度を設定する必要があります。

サポートされるデータ・カートリッジ	固有容量	圧縮データ容量	SMIT 密度設定値	16 進数の密度設定値
DLTtape III	2.6GB	2.6GB (圧縮なし)	23	17h
	6.0GB	6.0GB (圧縮なし)	24	18h
	10.0GB	20.0GB (ドライブのデフォルト)	25	19h
DLTtapeIIIxt	15.0GB	30.6GB (ドライブのデフォルト)	25	19h
DLTtapeIV	20.0GB	40.0GB	26	1Ah
	35.0GB	70.0GB (ドライブのデフォルト)	27	1Bh

注 : データ・カートリッジに関して、サポートされていない固有容量を要求すると、そのドライブは、デフォルトとして、そのドライブにロードされているデータ・カートリッジでサポートされている最高の容量値をとります。

データ圧縮
実際の圧縮は、書き込まれるデータのタイプによって異なります (上記の表を参照してください)。この圧縮データ容量については、圧縮率 2:1 が想定されています。

固定値を持つ属性
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

150 MB 1/4 インチ・テープ・ドライブ (タイプ 150mb) の属性

以下は 150 MB 1/4 インチ・テープ・ドライブ (タイプ 150mb) の属性です。

ブロック・サイズ
デフォルトのブロック・サイズは 512 です。 可変長ブロックでは、これ以外に有効な唯一のブロック・サイズは 0 です。

デバイス・バッファー
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

拡張ファイル・マーク
1/4 インチ・テープへの書き込みは、テープ開始点 (BOT)、あるいはブランク・テープが検出された後にのみ行われます。 そのテープにデータがある場合は、BOT の場合を除き、そのデータには上書きできません。 書き込んでから巻き戻されたテープにデータを追加する場合は、次のファイル・マークが検出されるまで前送りする必要があります。これによって、システムはエラーを戻します。 書き込みを再開できるのは、この後に限られます。

緩み取り
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

#1 に設定中の記録密度と #2 に設定中の記録密度
以下の設定値が適用されます。

設定値	意味
16	QIC-150
15	QIC-120
0	デフォルト (QIC-150)、または使用システムによる最新の密度設定値。

デフォルト値は、#1 に設定中の記録密度の場合は 16 で、#2 に設定中の記録密度の場合は 15 です。

固定値を持つ属性
テープ・ドライブが 150MB 1/4 インチ・テープ機構/装置として構成されている場合は、拡張ファイル・マーク、予約のサポート、可変長ブロック・サイズ、およびデータ圧縮の各属性に、変更不可能な値が事前定義されています。

525MB 1/4 インチ・テープ・ドライブ (タイプ 525mb) の属性

以下は 525MB 1/4 インチ・テープ装置 (タイプ 525mb) の属性です。

ブロック・サイズ
デフォルトのブロック・サイズは 512 です。 これ以外の有効なブロック・サイズは、0 (可変長ブロックの場合) と 1024 です。

デバイス・バッファー
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

拡張ファイル・マーク
1/4 インチ・テープへの書き込みは、テープ開始点 (BOT)、あるいはブランク・テープが検出された後にのみ行われます。 そのテープにデータがある場合は、BOT の場合を除き、そのデータには上書きできません。 書き込んでから巻き戻されたテープにデータを追加する場合は、次のファイル・マークが検出されるまで前送りする必要があります。これによって、システムはエラーを戻します。 書き込みを再開できるのは、この後に限られます。

緩み取り
1/4 インチ・テープへの書き込みは、テープ開始点 (BOT)、あるいはブランク・テープが検出された後にのみ行われます。 そのテープにデータがある場合は、BOT の場合を除き、そのデータには上書きできません。 書き込んでから巻き戻されたテープにデータを追加する場合は、次のファイル・マークが

検出されるまで前送りする必要があります。これによって、システムはエラーを戻します。 書き込みを再開できるのは、この後に限られます。

#1 に設定中の記録密度と #2 に設定中の記録密度
以下の設定値が適用されます。

設定値	意味
17	QIC-525*
16	QIC-150
15	QIC-120
0	デフォルト (QIC-525)、または使用システムによる最新の密度設定値。

* QIC-525 は、1024 のブロック・サイズをサポートする唯一のモードです。

デフォルト値は、#1 に設定中の記録密度の場合は 17 で、#2 に設定中の記録密度の場合は 16 です。

固定値を持つ属性
テープ・ドライブが 525 MB 1/4 インチ・テープ・ドライブとして構成されている場合、拡張ファイル・マーク、予約サポート、可変長ブロック・サイズ、およびデータ圧縮の各属性には、変更できない事前定義値があります。

1200MB 1/4 インチ・テープ装置 (タイプ 1200mb-c) の属性
以下は 1200MB 1/4 インチ・テープ装置 (タイプ 1200mb-c) の属性です。

ブロック・サイズ
デフォルトのブロック・サイズは 512 です。 これ以外の有効なブロック・サイズは、0 (可変長ブロックの場合) と 1024 です。

デバイス・バッファー
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

拡張ファイル・マーク
1/4 インチ・テープへの書き込みは、テープ開始点 (BOT)、あるいはブランク・テープが検出された後にのみ行われます。 そのテープにデータがある場合は、BOT の場合を除き、そのデータには上書きできません。 書き込んでから巻き戻されたテープにデータを追加する場合は、次のファイル・マークが検出されるまで前送りする必要があります。これによって、システムはエラーを戻します。 書き込みを再開できるのは、この後に限られます。

緩み取り
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

#1 に設定中の記録密度と #2 に設定中の記録密度
以下の設定値が適用されます。

設定値	意味
21	QIC-1000*
17	QIC-525*
16	QIC-150
15	QIC-120
0	デフォルト (QIC-1000)、または使用システムによる最新の密度設定値。

* QIC-525 と QIC-1000 だけが、1024 ブロック・サイズをサポートしているモードです。

デフォルト値は、#1 に設定中の記録密度の場合は 21 で、#2 に設定中の記録密度の場合は 17 です。

固定値を持つ属性
テープ・ドライブが 1200 MB 1/4 インチ・テープ・ドライブとして構成されている場合、拡張ファイル・マーク、予約サポート、可変長ブロック・サイズ、およびデータ圧縮の各属性には、変更できない事前定義値があります。

12000 MB 4 mm テープ・ドライブ (自己構成) の属性

以下は 12000 MB 4 mm テープ・ドライブ (自己構成) の属性です。

ブロック・サイズ

IBM 12000 MB 4 ミリ・テープ装置のスループットは、ブロック・サイズに依存します。 このドライブに推奨できる最小ブロック・サイズは、32 KB です。ブロック・サイズが 32 KB に満たない場合、データ転送速度 (バックアップまたはリストアの時間) が制限されます。以下の表は、コマンドごとの推奨ブロック・サイズを示したものです。

サポートされるコマンド	デフォルト・ブロック・サイズ (バイト)	推奨
BACKUP	32K または 51.2K (デフォルト)	バックアップが名前によるか否かによって、32 K または 51.2 K のいずれかを使用します。変更は必要ありません。
TAR	10K	512 KB ブロック・サイズと示している資料にエラーがあります。**Blocking** パラメーターを **-N64** に設定。
MKSYSB	BACKUP を参照してください。	**MKSYSB** コマンドは **BACKUP** コマンドを使用します。変更は必要ありません。
DD		**Blocking** パラメーターを bs=32K に設定。
CPIO		**Blocking** パラメーターを -C64 に設定。

注 : ブロック・サイズを選択する場合には、容量およびスループットを知っておく必要があります。 ブロック・サイズが小さい場合、パフォーマンスには大きな影響を与えますが、容量にはほとんど影響しません。

デバイス・バッファー

この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

拡張ファイル・マーク

この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

#1 に設定中の記録密度と #2 に設定中の記録密度

次の表は、IBM 12000 MB 4 ミリ・テープ装置に使用される、サポートされるデータ・カートリッジ・タイプと密度設定値 (10 進法と 16 進法) を示します。 復元 (読み取り) 操作を実行するとき、テープ・ドライブは、書き込まれた密度と一致するように自動的に密度を設定します。 バックアップ操作 (書き込み) を行う場合、使用するデータ・カートリッジと一致するように密度を設定する必要があります。

サポートされるデータ・カートリッジ	固有容量	圧縮データ容量	SMIT 密度設定値	16 進数の密度設定値
DDS III	2.0 GB	4.0 GB	19	13h
DDS2	4.0 GB	8.0 GB	36	24h
DDS3	12.0 GB	24.0 GB	37	25h

注 : データ・カートリッジに関して、サポートされていない固有容量を要求すると、そのドライブは、デフォルトとして、そのドライブにロードされているデータ・カートリッジでサポートされている最高の容量値をとります。

データ圧縮

実際の圧縮は、書き込まれるデータのタイプによって異なります (上記の表を参照してください)。この圧縮データ容量については、圧縮率 2:1 が想定されています。

固定値を持つ属性
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

13000MB 1/4 インチ・テープ・ドライブ (自己構成) の属性

以下は 13000 MB 1/4 インチ・テープ・ドライブ (自己構成) の属性です。

ブロック・サイズ
デフォルトのブロック・サイズは 512 です。 これ以外の有効なブロック・サイズは、0 (可変長ブロックの場合) と 1024 です。

デバイス・バッファー
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

拡張ファイル・マーク
1/4 インチ・テープへの書き込みは、テープ開始点 (BOT)、あるいはブランク・テープが検出された後にのみ行われます。 そのテープにデータがある場合は、BOT の場合を除き、そのデータには上書きできません。 書き込んでから巻き戻されたテープにデータを追加する場合は、次のファイル・マークが検出されるまで前送りする必要があります。これによって、システムはエラーを戻します。 書き込みを再開できるのは、この後に限られます。

緩み取り
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

#1 に設定中の記録密度と #2 に設定中の記録密度
以下の設定値が適用されます。

設定値	意味
33	QIC-5010-DC*
34	QIC-2GB*
21	QIC-1000*
17	QIC-525*
16	QIC-150
15	QIC-120
0	デフォルト (QIC-5010-DC)*

* QIC-525、QIC-1000、QIC-5010-DC、および QIC-2GB だけが、1024 ブロック・サイズをサポートしているモードです。

デフォルト値は、#1 に設定中の記録密度の場合は 33 で、#2 に設定中の記録密度の場合は 34 です。

固定値を持つ属性
テープ・ドライブが 13000 MB 1/4 インチ・テープ・ドライブとして構成されている場合、拡張ファイル・マーク、予約サポート、および可変長ブロック・サイズの各属性には、変更できない事前定義値があります。

1/2 インチ 9 トラック・テープ・ドライブ (タイプ 9trk) の属性

以下は 1/2 インチ 9 トラック・テープ・ドライブ (タイプ 9trk) の属性です。

ブロック・サイズ
デフォルトのブロック・サイズは 1024 です。

デバイス・バッファー
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

#1 に設定中の記録密度と #2 に設定中の記録密度
以下の設定値が適用されます。

設定値	意味
3	6250 ビット/インチ (bpi)

設定値	意味
2	1600 bpi
0	以前に使用された書き込み密度であればいずれでも

デフォルト値は、#1 に設定中の記録密度の場合は 3、#2 に設定中の記録密度の場合は 2 に設定されます。

固定値を持つ属性
テープ・ドライブが 1/2 インチ 9 トラック・テープ・ドライブとして構成されている場合、拡張ファイル・マーク、緩み取り、予約サポート、可変長ブロック・サイズ、およびデータ圧縮の各属性には、変更できない事前定義値があります。

3490e 1/2 インチ・カートリッジ (タイプ 3490e) の属性
以下は 3490e 1/2 インチ・カートリッジ (タイプ 3490e) の属性です。

ブロック・サイズ
デフォルトのブロック・サイズは 1024 です。 このドライブの特長はデータ転送速度が高速であるということであり、効率的な操作を行うためにはブロック・サイズが重要になります。 大きなブロック・サイズによって、操作速度を大幅に向上させることができるので、通常は、可能な限り最大のブロック・サイズを使用するようにしてください。

注 : ブロック値を大きくすると、そのシステム上の他のプログラムと互換性がなくなる可能性があります。 このことが発生した場合、プログラムの実行中に次のエラー・メッセージを受け取ります。

```
A system call received a parameter that is not valid.
```

デバイス・バッファー
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

圧縮
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

オートローダー
このドライブの特徴は、テープ・シーケンサー、つまり、オートローダーを備えていて、カートリッジ・ローダーに対して一連のテープ・カートリッジのロードと排出を順次行うことです。 この機能を正しく作動させるには、フロントパネル・スイッチを AUTO の位置にセットし、オートローダー属性を yes に設定する必要があります。

その他の SCSI テープ (タイプ ost) の属性
以下は、その他の SCSI テープ (タイプ ost) の属性です。

ブロック・サイズ
システム・デフォルトは 512 ですが、これは、使用しているテープ・ドライブのデフォルトの ブロック・サイズに調整する必要があります。 一般的な値は 512 と 1024 です。 8 mm と 4 mm のテープ・ドライブでは、通常 1024 を使用するので、ブロック・サイズ属性が 512 に設定されたままになっている場合は、スペースが無駄になります。 一部のドライブでは、0 で可変ブロック・サイズを示します。

デバイス・バッファー
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

拡張ファイル・マーク
この属性に関する一般的な情報は、このテープ・ドライブ・タイプに適用されます。

#1 に設定中の記録密度と #2 に設定中の記録密度
デフォルト値はいずれの設定値の場合も 0 です。 他の値とその意味は、テープ・ドライブによって異なります。

予約サポート
デフォルト値は no です。ドライブが予約/解放コマンドをサポートしている場合は、この値を yes に設定することができます。 これらのコマンドをサポートしているかどうかわからない場合は、no に設定しておくと安全です。

可変長ブロック・サイズ
可変長ブロック・サイズのデフォルト値は 0 です。ゼロ以外の値は、基本的に、1/4 インチ・カートリッジ (QIC) ドライブで使用されます。 詳細については、特定のテープ・ドライブの 『SCSI 仕様書』を参照してください。

再試行遅延
この属性は、タイプが ost のテープ・ドライブにのみ適用されます。

読み取り/書き込みタイムアウト
この属性は、タイプが ost のテープ・ドライブにのみ適用されます。

固定値を持つ属性
テープ・ドライブが他の SCSI テープ・ドライブとして構成されている場合は、拡張ファイル・マーク、緩み取り、およびデータ圧縮の各属性に、変更不可能な値が事前定義されています。

MPIO テープ属性
MPIO がサポートする磁気テープ装置には、MPIO 装置属性の下にリストされた追加属性があります。

テープ・ドライブ用のスペシャル・ファイル

オペレーティング・システムが認識している各テープ・ドライブに関連するスペシャル・ファイルが、いくつかあります。

テープ上のファイルに対する書き込みおよび読み取りは、rmt スペシャル・ファイルを使用して行います。これらのスペシャル・ファイルは、/dev/rmt*、/dev/rmt*.1、/dev/rmt*.2 から /dev/rmt*.7 です。 rmt* はテープ・ドライブの論理名です。例えば、rmt0、rmt1、などです。

テープ・ドライブに関連したいずれか 1 つのスペシャル・ファイルを選択して、そのテープ・ドライブに関した入出力操作の実行方法を選択します。

項目	説明
密度	テープ・ドライブの #1 に設定中の記録密度と #2 に設定中の記録密度のいずれで書き込むかを選択することができます。これらの密度設定値の値は、そのテープ・ドライブの属性の一部です。 通常、#1 に設定中の記録密度は、そのテープ・ドライブに使用できる最高密度に設定され、#2 に設定中の記録密度は、そのテープ・ドライブに使用できる 2 番目に高い密度に設定されるため、#1 に設定中の記録密度を使用するスペシャル・ファイルは高密度であると言われ、#2 に設定中の記録密度を使用するスペシャル・ファイルは低密度であると言われる場合がありますが、この表現は必ずしも正しいとは限りません。 テープから読み取りを行っているときは、この密度設定値は無視されます。
クローズ時の巻き戻し	テープ・ドライブに関連するスペシャル・ファイルがクローズされるときに、そのテープを巻き戻すかどうかを選択できます。 クローズ時に巻き戻しを選択すると、テープは、ファイルのクローズ時にテープの先頭の位置に合わせられます。
オープン時の緩み取り	ファイルのオープン時にテープの緩み取りを行うか否かを選択することができます。 緩み取りとは、テープを終わりまで巻いてから、テープの最初まで巻き戻して、エラーを少なくすることを意味します。 オープン時の緩み取りを選択すると、 オープン・プロセス中に、テープが最初まで巻き戻されます。

以下の表は、rmt スペシャル・ファイルの名前と、それらの特性を示したものです。

スペシャル・ファイル	クローズ時の巻き戻し	オープン時の緩み取り	密度設定
/dev/rmt*	はい	いいえ	#1
/dev/rmt*.1	いいえ	いいえ	#1
/dev/rmt*.2	はい	はい	#1

スペシャル・ファイル	クローズ時の巻き戻し	オープン時の緩み取り	密度設定
/dev/rmt*.3	いいえ	はい	#1
/dev/rmt*.4	はい	いいえ	#2
/dev/rmt*.5	いいえ	いいえ	#2
/dev/rmt*.6	はい	はい	#2
/dev/rmt*.7	いいえ	はい	#2

テープ・ドライブ rmt2 のテープに 3 つのファイルを書き込むとします。 1 番目のファイルはテープの先頭に置かれ、2 番目のファイルは 1 番目のファイルの後ろに置かれ、3 番目のファイルは 2 番目のファイルの後ろに置かれます。 次にリストしているスペシャル・ファイルは、その順序で、テープへの書き込みに使用することができます。

1. /dev/rmt2.3
2. /dev/rmt2.1
3. /dev/rmt2

これらの特定のスペシャル・ファイルは、次のような理由で選択されます。

- /dev/rmt2.3 が 1 番目のファイルとして選択されているのは、このファイルには、オープン時の緩み取りがあり、1 番目のファイルがテープの開始点に位置づけられるからです。 クローズ時の巻き戻しは、次の入出力操作がこのファイルの終了点から開始するので、選択されません。 このファイルが、1 番目のファイルのオープン時に既に開始点に位置づけられている場合は、/dev/rmt2.1 ファイルを 1 番目のファイルとして使用した方が、テープの緩み取りにかかる時間が省けるので、速く処理できます。
- /dev/rmt2.1 は、オープン時の緩み取りもクローズ時の巻き戻しも選択されていないため、2 番目のファイルとして選択されます。 ファイルのオープン時かクローズ時に、テープの開始点に位置づけられなければならない理由はありません。
- /dev/rmt2 は、3 番目のファイルが 2 番目のファイルの後にくるのでオープン時の緩み取りが不要なために、3 番目すなわち最後のファイルとして選択されます。 3 番目のファイルの後にテープへの書き込みを行う予定がないために、クローズ時の巻き戻しが選択されています。 次にこのテープを使用するときは、そのテープの開始点から開始します。

特定の rmt スペシャル・ファイルを選択してテープ操作を制御するほかに、**tctl** コマンドを使用してテープ操作を制御することもできます。

USB デバイスのサポート

AIX オペレーティング・システムは、USB デバイスをサポートしています。

AIX オペレーティング・システムは、以下のタイプのサード・パーティー USB デバイスおよび IBM USB デバイスをサポートしています。

- フラッシュ・ドライブ
- ディスク・ドライブ
- 光ディスク・ドライブ (Blu-ray、DVD、および CD-ROM)
- テープ・ドライブ
- キーボード
- マウス
- スピーカー

注 : AIX オペレーティング・システムは、USB プリンターをサポートしていません。

一部のサード・パーティー USB デバイスは、AIX オペレーティング・システムによって認識されない可能性があります。 例えば、電源システム USB ポートから得られる電流が不十分である場合が考えられます。そのため、AIX オペレーティング・システムは、考えられるすべてのサード・パーティー USB デバイスをサポートするわけではありません。

USB フラッシュ・ドライブのサポート

AIX 5.3 テクノロジー・レベル 5300-09 および AIX 6.1 テクノロジー・レベル 6100-02 からは、USB フラッシュ・ドライブがサポートされています。

このデバイスのサポートは、次のデバイス・パッケージに組み込まれています。

```
devices.usbif.08025002
```

AIX による USB フラッシュ・ドライブのサポートは、業界標準の OEM USB フラッシュ・ドライブのサンプルで検証されています。AIX USB ホスト・コントローラーのデバイス・ドライバーは、USB 2.0 をサポートします。USB フラッシュ・ドライブは、`usbms0` や `usbms1` などの論理名で構成され、ロー・スペシャル・ファイルとブロック・スペシャル・ファイルの両方を提供します。例えば、`usbms0` のロー・スペシャル・ファイルは `/dev/rusbms0`、ブロック・スペシャル・ファイルは `/dev/usbms0` です。AIX バージョン 5.3 テクノロジー・レベル 5300-11 よりも前、および AIX バージョン 6.1 テクノロジー・レベル 6100-04 よりも前では、USB フラッシュ・ドライブは `/dev/flashdrive0` として構成されていました。

これらのドライブでは、国際標準化機構 (ISO) ファイルシステム (読み取り専用 ISO 9660) がサポートされています。**tar** コマンド、**cpio** コマンド、またはアーカイブのバックアップあるいはリストアを 使用することにより、ドライブ上にシステム・バックアップを作成することができます。また、**dd** コマンドを使用すると、ISO イメージをドライブに追加することができます。

AIX オペレーティング・システムでは、USB フラッシュ・ドライブのプラグ・アンド・プレイはサポートされていません。AIX ユーザーがフラッシュ・ドライブを利用できるようにするには、root ユーザーがドライブをシステム USB ポートに接続し、次のコマンドを実行する必要があります。

```
cfgmgr -l usb0
```

重要 : ポートからフラッシュ・ドライブを取り外すときは注意してください。取り外す前にドライブが正しくクローズされていなかったりアンマウントされていなかったりした場合、ドライブ上のデータが破壊される可能性があります。

ドライブを取り外した後、root ユーザーが次のコマンドを実行するまでは、ドライブはオブジェクト・データ・マネージャー (ODM) で使用可能状態のままとなります。

```
rmdev -l usbms<i>n</i>
```

ドライブが使用可能状態の場合は、システムに再接続して、再マウントまたは再オープンすることができます。ユーザーに対してオープンされたままの状態でドライブがシステム USB ポートから切断された場合、そのドライブは、ユーザーがクローズして再オープンするまで再使用可能になりません。

USB ブルーレイ・ドライブの読み取り専用サポート

AIX バージョン 6.1 (6100-06 テクノロジー・レベル適用) およびそれ以降は、USB 接続ブルーレイ・ドライブを認識して構成します。

この機能は、次のデバイス・パッケージに組み込まれています。

```
devices.usbif.08025002
```

ブルーレイ・メディアを読み取る AIX オペレーティング・システムの機能は、業界標準の相手先商標製造会社 (OEM) の USB ブルーレイ・ドライブのサンプルで 検証されています。

USB ブルーレイ・ドライブは、論理名 (`cd0` や `cd1` など) を使用して構成されます。これらのドライブは、ロー・スペシャル・ファイルとブロック・スペシャル・ファイルの両方を提示します。例えば、`cd0` のロー・スペシャル・ファイルは `/dev/rcd0`、ブロック・スペシャル・ファイルは `/dev/cd0` です。

読み取り専用機能は、国際標準化機構 (ISO) ファイルシステム (読み取り専用 ISO 9660)、ユニバーサル・ディスク・フォーマット (UDF) ファイルシステム (バージョン 2.01 以前) および標準の光メディア・アクセス・コマンド (**dd** や **tar** など) に対して提供されています。

AIX オペレーティング・システムでは、USB ブルーレイ・ドライブにある CD、DVD、またはブルーレイ・メディアへの書き込み操作はサポートされません。(ドライブが書き込み可能である場合) 書き込み操作は禁止されませんが、IBM は、書き込み操作時に検出される問題をサポートしません。

AIX オペレーティング・システムでは、USB ブルーレイ・ドライブのプラグ・アンド・プレイはサポートされていません。AIX ユーザーから USB ブルーレイ・ドライブが利用できるようにするには、root ユーザーがそのドライブをシステムの USB ポートに接続し、以下のコマンドを実行する必要があります。

```
cfgmgr -l usb0
```

ドライブが取り外された後、root ユーザーが次のコマンドを実行するまで、ドライブはオブジェクト・データ・マネージャー (ODM) データベースで使用可能状態のままとなります。

```
rmdev -l cd*n*
```

ドライブが使用可能状態の場合は、システムに再接続することができます。ユーザーに対してオープンされたままの状態でドライブがシステム USB ポートから切断された場合、クローズして再オープンするまでそのドライブを使用できません。

ストレージ・データのキャッシング

ストレージ・データのサーバー・サイド・キャッシングは、キャッシュ・デバイスでサポートされています。

キャッシュ・デバイスには、以下のいずれかのタイプのデバイスを使用できます。

- サーバー接続のフラッシュ・デバイス。例えば、サーバーに組み込まれたソリッド・ステート・ドライブ (SSD)。
- シリアル接続 SCSI (SAS) コントローラーを使用してサーバーに直接接続されたフラッシュ・デバイス。
- ストレージ・エリア・ネットワーク (SAN) 内のフラッシュ・リソース。

AIX オペレーティング・システムでは、デバイスをキャッシュ・デバイスとして使用するためには、そのデバイスがフラッシュ・デバイスとして識別されている必要があります。 AIX オペレーティング・システムは、「SCSI ブロック・デバイス特性重要プロダクト・データ (VPD) (SCSI Block Device Characteristics Vital Product Data (VPD))」ページの「**メディアの回転速度 (MEDIUM ROTATION RATE)**」フィールドを使用して、デバイスがフラッシュ・デバイスであるかどうかを判断します。 デバイスがこのページをサポートしておらず、`0001h Non-rotating medium` 以外の値を「**メディアの回転速度 (MEDIUM ROTATION RATE)**」フィールドに書き込む場合、そのデバイスをキャッシュ・デバイスとして使用することはできません。

ストレージ・データ・キャッシングの概念

ワークロードの実行中、ストレージ・データを動的にキャッシュ (キャッシングを開始または停止) できます。 キャッシング操作を実行するために、ワークロードを非アクティブ状態にする必要はありません。

キャッシングの概念を説明するために、以下の用語が使用されます。

キャッシュ・デバイス
キャッシュ・デバイスは、キャッシング用のソリッド・ステート・ドライブ (SSD) またはフラッシュ・ディスクです。

キャッシュ・プール
キャッシュ・プールは、ストレージ・キャッシング専用のキャッシュ・デバイスのグループです。

キャッシュ・パーティション
キャッシュ・パーティションは、キャッシュ・プールから作成される論理的なキャッシュ・デバイスです。

ターゲット・デバイス
ターゲット・デバイスは、キャッシュされるストレージ・デバイスです。

単一のキャッシュ・パーティションを使用して、1 つ以上のターゲット・デバイスをキャッシュできます。ターゲット・デバイスをキャッシュすると、そのデバイス・ブロックに対するすべての読み取り要求はキャッシング・ソフトウェアにルーティングされます。 キャッシュ内に特定のブロックが見つかると、入出力要求はキャッシュ・デバイスで処理されます。 要求されたブロックがキャッシュ内に見つからない場合、または要求が書き込み要求である場合、入出力要求はターゲット・デバイスに戻ります。

ストレージ・データ・キャッシングの利点

ストレージ・データをサーバー・サイドでキャッシュすると、特にストレージ・サブシステムが輻輳している場合に、仮想化密度を向上できます。

ストレージ・データ・キャッシングには以下の利点があります。

待ち時間
ソリッド・ステート・ドライブ (SSD) ストレージの待ち時間は短いため、分析とトランザクションのワークロードの照会応答時間が短縮されます。 サーバー・サイド・キャッシングを使用すると、トランザクション・ワークロードの平均待ち時間を半分に短縮できます。

スループット
SSD ストレージではスループットが向上するため、オンライン・トランザクション処理 (OLTP) ワークロードのトランザクション速度が高まります。

書き込みスループット
ストレージ・エリア・ネットワーク (SAN) が輻輳している環境では、キャッシュとして使用されるフラッシュ・ドライブが読み取り要求の多くの割合をオフロードできます。 読み取り要求がオフロードされると、SAN の書き込みスループットを向上でき、SAN はより多くのクライアントとホストに効率的に対応できます。

メモリー占有スペースの削減
フラッシュ・キャッシュ・ドライブを構成すると、メモリー占有スペースが少ない場合でも一部のワークロードは実行可能です。

ストレージ・データ・キャッシングの制約

キャッシング機能を使用する場合の制約と追加の構成要件を、確実に理解しておいてください。 キャッシュする必要があるターゲット・デバイスに関するアプリケーションの制約についても考慮する必要があります。

ストレージ・データのキャッシングに関する以下の制約を考慮してください。

- キャッシング・ソフトウェアは読み取り専用キャッシュとして構成されます。つまり、フラッシュ・ソリッド・ステート・ドライブ (SSD) では読み取り要求のみが処理されます。 書き込み要求はすべて、元のストレージ・デバイスによって処理されます。
- ストレージ・デバイスに書き込まれるデータは、自動的にキャッシュに取り込まれるわけではありません。キャッシュ内のブロックに対する書き込み操作が実行される場合、そのキャッシュ内の既存のデータには無効のマークが付けられます。 同じブロックは、そのブロックへのアクセスの頻度と最新性に応じて、キャッシュに再び現れます。
- キャッシング・ソフトウェアが各読み取りブロックのメタデータを管理するので、各 AIX 論理区画 (LPAR) に追加のメモリーが必要です。キャッシングが有効になっている LPAR では最小で 4 GB のメモリーが必要です。
- キャッシング・ソフトウェアはローカルの読み取りパターンに基づいてキャッシュ内にデータをロードし、キャッシュ・エントリーをローカルで無効にします。 ターゲット・デバイスは複数の LPAR によって同時に共有してはなりません。ターゲット・デバイスは、Oracle Real Application Clusters (RAC)、DB2 pureScale®、General Parallel File System (GPFS) などのクラスター化ストレージに含めることはできません。高可用性クラスターに含まれるターゲット・デバイスをキャッシュできるのは、アクセス権限により、ターゲット・デバイスでのデータの読み取りや書き込みを一度に 1 つのホストが行い、キャッシングがアクティブ・ノードでのみ有効になるように指定されている場合のみです。
- キャッシュ・ディスクは AIX LPAR と仮想入出力サーバー (VIOS) LPAR のどちらにもプロビジョンできます。キャッシュ・デバイスは共有できません。

- キャッシング・ソフトウェアは、ターゲット・デバイスをオープンしてターゲット・デバイスへの入出力要求をインターセプトする必要があります。 キャッシングの開始後にワークロードでターゲット・デバイスを排他的にオープンする必要が生じた場合、その排他的オープン操作は失敗します。 こういった場合には、キャッシングを停止し、ワークロードの開始後に再開する必要があります。 排他的オープン操作の例として、ターゲット・ディスクに対する物理ボリューム ID (PVID) の設定があります。
- ディスクをターゲット・デバイスとして使用する場合、そのディスクの **reserve_policy** 属性は `single_path` に設定してはなりません。
- ターゲット・デバイスのキャッシング操作が開始されると、キャッシュ・エンジンのロジックはキャッシュ内へのデータのプロモーションを遅延させます。 この遅延は、ターゲット・デバイスに対する未処理のすべての入出力操作 (キャッシング操作の開始前に発行されているもの) をキャッシング操作の開始前に完了させるために必要です。 正確な遅延時間は、ターゲット・ディスクの使用可能なパスの数と **rw_timeout** 属性 (ある場合) に基づいて、内部で計算されます。 内部で計算された時間をユーザー定義の時間によってオーバーライドする必要がある場合は、`/etc/environment` ファイル内の *DEFAULT_IO_DRAIN_TIMEOUT_PD* 環境変数をカスタム・タイムアウト値 (秒数) に設定できます。

ストレージ・データ・キャッシングのコンポーネント

キャッシング・ソフトウェアはキャッシュ管理コンポーネントとキャッシュ・エンジン・コンポーネントで構成されます。

キャッシュ管理

ストレージ・データ・キャッシングは **cache_mgt** コマンドを使用して管理できます。このコマンドは AIX オペレーティング・システムおよび仮想入出力サーバー (VIOS) で使用できます。 **cache_mgt** コマンドを使用すると、以下のタスクを実行できます。

- キャッシュ・プールを作成し、パーティション化する。
- キャッシュ・パーティションを、仮想 SCSI (vSCSI) デバイスとしてターゲット・デバイスまたは AIX 論理区画 (LPAR) に割り当てる。
- キャッシング操作を開始および停止する。

キャッシュ・エンジン

キャッシュ・エンジンは、キャッシング・ソフトウェアの最も重要な部分です。キャッシュ・エンジンは、ストレージ内のどのブロックをキャッシュする必要があるか、およびデータをキャッシュと 1 次ストレージのどちらから取得する必要があるかを決定します。

キャッシング・アルゴリズムは「読み取り時に取り込み (populate-on-read)」メカニズムに基づきます。このメカニズムでは、空間的局所性のあるデータ (最近読み取られた他のブロックの付近にあるデータ) をキャッシュに取り込みます。 このキャッシング・アルゴリズムでは、キャッシュが空の場合、より迅速にデータがキャッシュに取り込まれます。

キャッシュ内のすべてのブロックがモニターされ、それらの読み取り頻度がチェックされて、ヒート・マップが生成されます。 ヒート・マップでは、アクセスの頻度と最新性の両方が考慮されます。取り込まれたデータでキャッシュが満杯になると、新規ブロックがキャッシュ内の最もコールドな (古い) ブロックよりウォームである (新しい) 場合に限り、キャッシュに新規エントリーが追加されます。 最もコールドなブロックはキャッシュから除去され、新規エントリーが追加されます。

積極的な取り込みを行うとウォームアップ時間を短縮できるため、使用可能になったキャッシュはすぐに有効になります。 ヒート・マップに基づく除去ポリシーでは、キャッシングが動的に行われ、変化するワークロード・パターンに合わせて調整されます。

ストレージ・データ・キャッシングの構成

AIX オペレーティング・システムでは、フラッシュ・デバイスのサーバー・サイド・キャッシングは複数の異なる構成でサポートされます。これらの各構成は、キャッシュ・デバイスが AIX 論理区画 (LPAR) にプロビジョンされる方法が異なります。

AIX オペレーティング・システムでは、サーバー・サイド・キャッシングで以下のモードがサポートされます。

- 専用モード

- 仮想モード
- N_Port ID Virtualization (NPIV) モード

専用モードでのストレージ・データのキャッシング
専用モードでは、キャッシュ・デバイスは AIX 論理区画 (LPAR) に直接プロビジョンされます。

必ずキャッシュ・プールを作成してから、キャッシュ・デバイス上にキャッシュ・パーティションを作成します。専用キャッシュ・デバイスに作成できるキャッシュ・パーティションは 1 つのみです。キャッシュ・パーティションを使用すると、AIX LPAR 上で任意の数のターゲット・デバイスをキャッシュできます。キャッシュ・デバイスはこの LPAR 専用であるため、LPAR の移動はできません。LPAR を別のサーバーにマイグレーションする必要がある場合は、キャッシングを手動で停止し、キャッシュ・デバイスを構成解除してからマイグレーションを行う必要があります。

次の図は、専用キャッシュ・デバイスでの AIX LPAR 上のキャッシング構成の例を示しています。

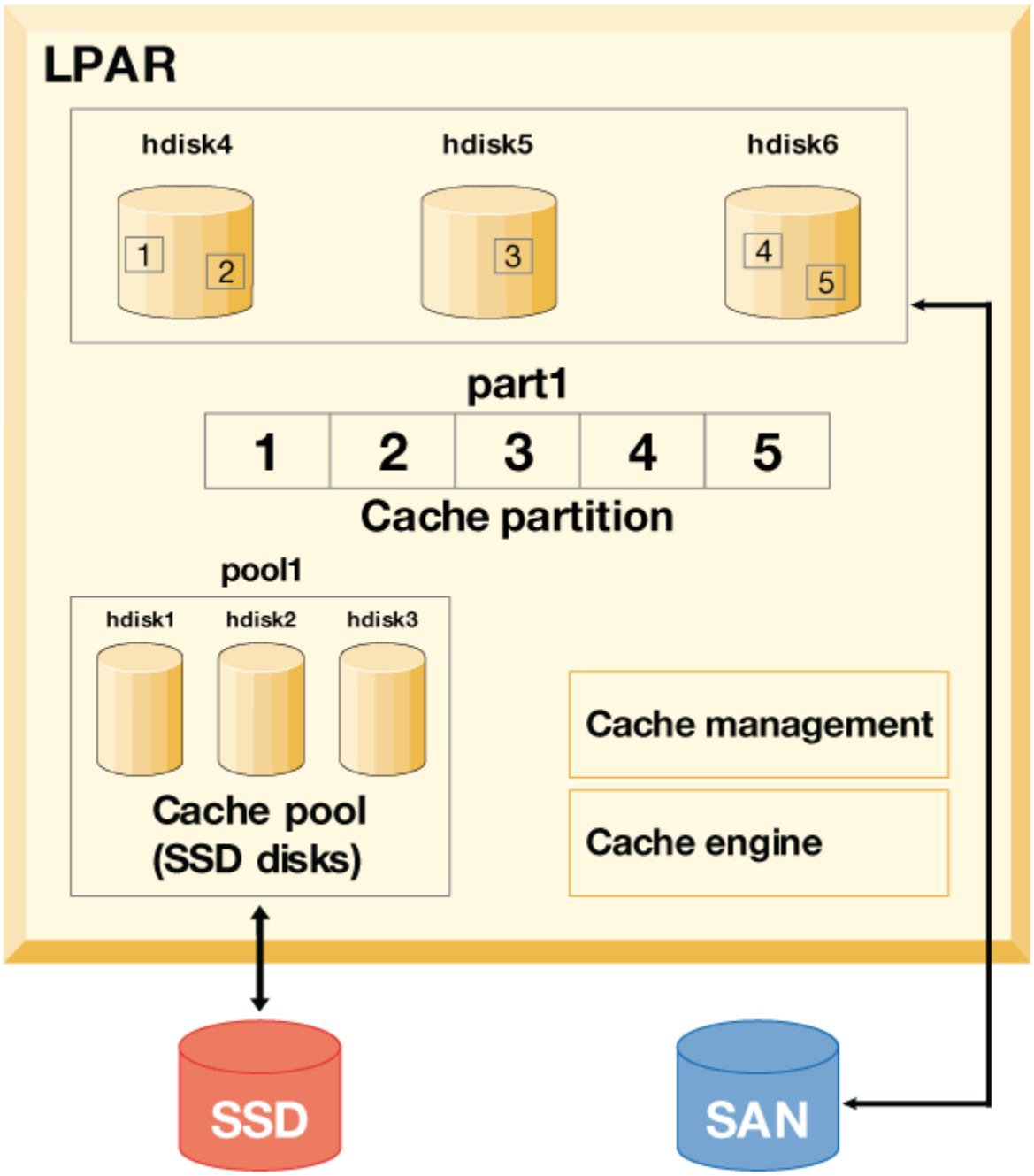


図 *15.* ストレージ・データ・キャッシング*:* 専用キャッシュ・デバイスの構成

キャッシュ・デバイスが hdisk1、hdisk2、および hdisk3 であり、ターゲット・デバイスが hdisk4、hdisk5、および hdisk6 であるとします。ターゲット・デバイスのキャッシングを開始およびモニターするには、以下のステップを実行します。

1. SSD ストレージ上にキャッシュ・プールを作成します。

```
# cache_mgt pool create -d hdisk1,hdisk2,hdisk3 -p cmpool0
```

2. そのキャッシュ・プールから 80 MB のキャッシュ・パーティションを作成します。

```
# cache_mgt partition create -p cmpool0 -s 80M -P part1
```

3. そのキャッシュ・パーティションを、キャッシュするターゲット・ディスクに割り当てます。

```
# cache_mgt partition assign -t hdisk4 -P part1
# cache_mgt partition assign -t hdisk5 -P part1
# cache_mgt partition assign -t hdisk6 -P part1
```

4. ターゲット・デバイスのキャッシングを開始します。

```
# cache_mgt cache start -t hdisk4
# cache_mgt cache start -t hdisk5
# cache_mgt cache start -t hdisk6
```

5. キャッシュ・ヒットの統計をモニターします。

```
# cache_mgt monitor get -h -s
```

仮想モードでのストレージ・データのキャッシング

仮想モードの場合、キャッシュ・デバイスは仮想入出力サーバー (VIOS) に割り当てられます。

仮想モードの場合、キャッシュ・プールは VIOS 上に作成されます。その後、キャッシュ・プールは VIOS 上でパーティションに分割されます。 各キャッシュ・パーティションは仮想ホスト (vhost) アダプターに割り当てることができます。AIX 論理区画 (LPAR) 上でキャッシュ・パーティションがディスカバーされると、そのキャッシュ・パーティションをターゲット・デバイスのキャッシングに使用できます。 キャッシュ・デバイスが仮想であるため、このキャッシュ・パーティションは別のサーバーにマイグレーションできます。 マイグレーション前に、ソース LPAR でキャッシングが自動的に停止されます。 キャッシング・ソフトウェアがインストールされていて、ターゲット VIOS でキャッシュ・プールが使用可能な場合、マイグレーション操作の一環として、ターゲット VIOS 上に同じサイズのキャッシュ・パーティションが動的に作成されます。 マイグレーション中、そのキャッシュ・パーティションが LPAR から使用できるようになります。 マイグレーションが完了すると、宛先 LPAR でキャッシングが自動的に開始されます。 この場合、キャッシングは空の (データが取り込まれていない) 状態で開始されます。

次の図は、仮想モードでのキャッシング構成の例を示しています。この場合、キャッシュ・デバイスは VIOS LPAR 上にあり、ターゲット・デバイスは AIX LPAR 上にあります。

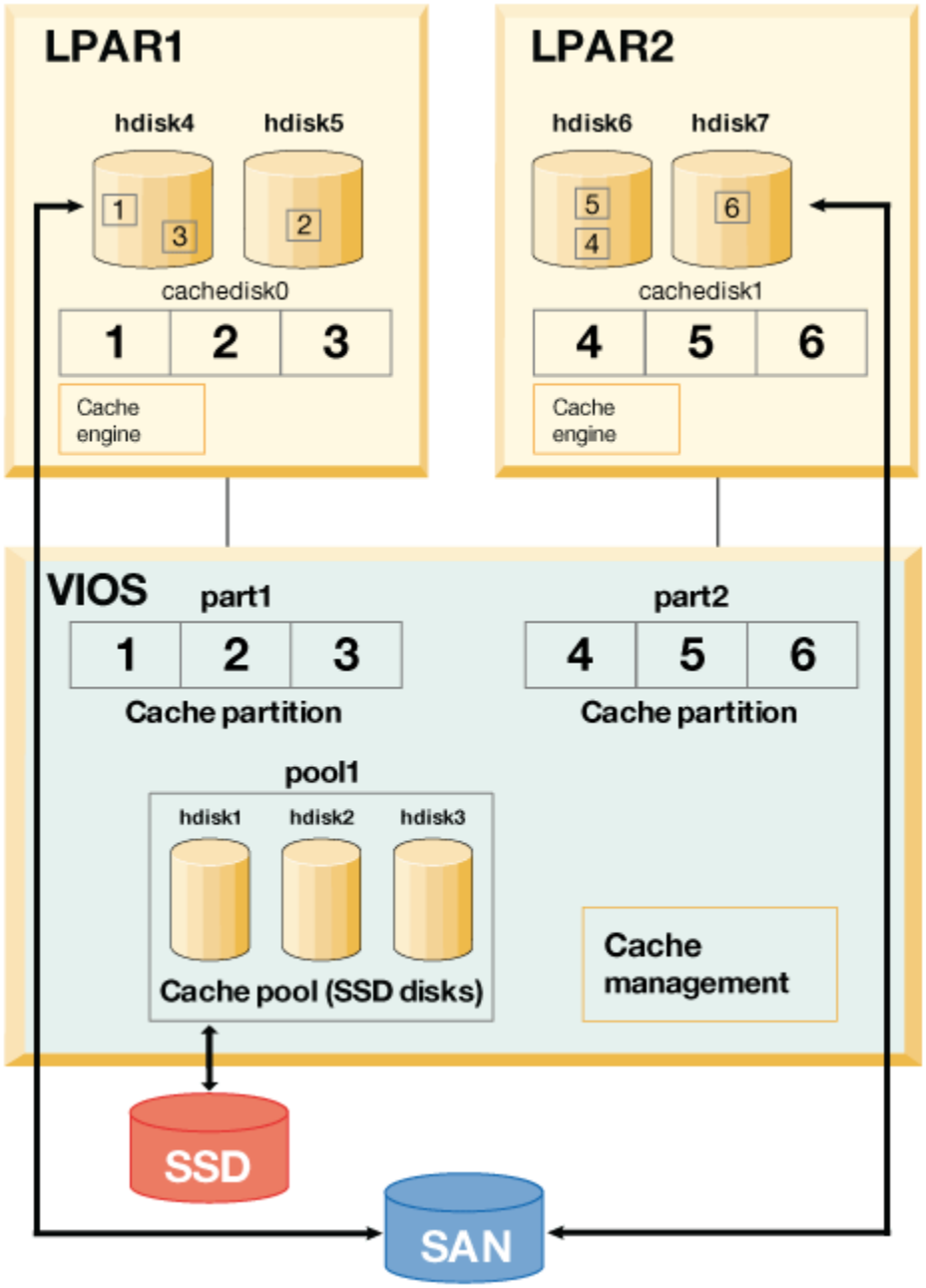


図 *16.* ストレージ・データ・キャッシング*:* 仮想キャッシュ・デバイスの構成

キャッシュ・デバイスが hdisk1、hdisk2、および hdisk3 (VIOS LPAR 上)、ターゲット・デバイスが hdisk4 および hdisk5 (AIX LPAR 上) であるとします。ターゲット・デバイスのキャッシングを開始およびモニターするには、以下の手順を実行します。

1. VIOS LPAR で、SSD ストレージを使用してキャッシュ・プールを作成します。

```
# cache_mgt pool create -d hdisk1,hdisk2,hdisk3 -p cmpool0
```

2. VIOS LPAR で、そのキャッシュ・プールから 80 MB のキャッシュ・パーティションを作成します。

```
# cache_mgt partition create -p cmpool0 -s 80M -P part1
```

3. VIOS LPAR で、そのパーティションを仮想ホスト・アダプターに割り当てます。

```
# cache_mgt partition assign -P part1 -v vhost0
```

4. AIX LPAR で、そのキャッシュ・パーティションをターゲット・デバイスに割り当てます。

```
# cache_mgt partition assign -t hdisk4 -P cachedisk0
# cache_mgt partition assign -t hdisk5 -P cachedisk0
```

5. AIX LPAR でターゲット・デバイスのキャッシングを開始します。

```
# cache_mgt cache start -t hdisk4
# cache_mgt cache start -t hdisk5
```

6. AIX LPAR でキャッシュ・ヒットの統計をモニターします。

```
# cache_mgt monitor get -h -s
```

NPIV モードでのストレージ・データのキャッシング

このモードでは、AIX 論理区画 (LPAR) 上でキャッシュ・デバイスを仮想ファイバー・チャネル (N_Port ID Virtualization) デバイスとして使用できます。

必ずキャッシュ・プールを作成してから、AIX LPAR 上にキャッシュ・パーティションを作成します。AIX LPAR に作成できるキャッシュ・パーティションは 1 つのみです。キャッシュ・パーティションを使用すると、AIX LPAR 上で任意の数のターゲット・デバイスをキャッシュできます。キャッシュ・デバイスはストレージ・エリア・ネットワーク (SAN) から使用可能であるため、この LPAR は別のサーバーにマイグレーションできます。 キャッシュ・デバイスは宛先システムに表示されるようにする必要があります。 キャッシュ操作はマイグレーション・プロセス中に続行可能であり、マイグレーションの完了後にキャッシュが取り込まれます。

NPIV モードでのターゲット・デバイスのキャッシングは専用モードでストレージ・データをキャッシュする場合と同じですが、キャッシュ・デバイスが SAN から使用できる点が異なります。 ただし、ターゲット・デバイスをキャッシュする手順は専用モードでストレージ・データをキャッシュする場合と同じです。

ストレージ・データ・キャッシュの管理

キャッシュが構成されている場合でも、時間がたつとキャッシング要件が変わる可能性があります。キャッシュするワークロードを新規に追加する必要が生じる場合があります。 変化する要件を満たすために、追加のキャッシュ・デバイスを含むようにキャッシュ・プールを拡張するか、既存のプールに新規キャッシュ・パーティションを作成するか、または既存のパーティションのサイズを 増やすことができます。

以下の例を利用して、キャッシング構成を管理できます。

- キャッシュ・デバイスをプールに追加するには、次のコマンドを入力します。

```
# cache_mgt pool extend -p pool1 -d hdisk4 -f
```

  -f フラグを指定すると、ディスク (hdisk4) に既存のボリューム・グループが含まれている場合に、そのディスクの既存の用途がオーバーライドされます。
- サイズ 100 MB の新規ワークロード用のパーティションを作成するには、次のコマンドを入力します。

```
# cache_mgt partition create -p pool1 -s 100M -P part2
```

- 既存のパーティションのサイズを 20 MB 増やすには、次のコマンドを入力します。

```
# cache_mgt  partition extend -p part1 -s 20M
```

高可用性に関する考慮事項

キャッシュされるターゲット・デバイスが高可用性クラスター内で管理されているリソース・グループに含まれる場合、フェイルオーバー操作を適切に計画する必要があります。

キャッシングをアクティブにできるノードは一度に 1 つのみです。フェイルオーバー・イベントを開始する前に、元のシステムでキャッシング操作が無効であることを必ず確認してください。 代替システムへのフェイルオーバーが完了したら、キャッシング・ソフトウェアを手動で有効にする必要があります。

キャッシング・ソフトウェアを有効にするには、以下のステップを実行します。

1. 元のシステムでキャッシングを停止します。

```
# cache_mgt cache stop -t hdisk2
```

2. フェイルオーバー・リカバリーが完了したら、新規システムでキャッシングを開始します。

```
# cache_mgt cache start -t hdisk2
```

キャッシュ統計情報のモニター

ターゲット・デバイスごとのキャッシュ統計情報は、**cache_mgt monitor get** コマンドで表示できます。

例えば、キャッシュされるターゲット・デバイスが hdisk1 のみであれば、**cache_mgt** コマンドの出力は次の例のようになります。

```
# cache_mgt monitor get -h -s
ETS Device I/O Statistics -- hdisk1
Start time of Statistics -- Mon Mar 27 07:10:41 2017
-----------------------------------------------------
 Read Count:                                152125803
 Write Count:                                79353626
 Read Hit Count:                                  871
 Partial Read Hit Count:                           63
 Read Bytes Xfer:                       10963365477376
 Write Bytes Xfer:                       4506245999616
 Read Hit Bytes Xfer:                         48398336
 Partial Read Hit Bytes Xfer:                  5768192
 Promote Read Count:                        2033078104
 Promote Read Bytes Xfer:               532959226494976
```

cache_mgt コマンドは、以下のキャッシュ・メトリックを表示します。

Read Count
すべてのアプリケーションによってターゲット・デバイスに対して実行された読み取り操作の総数。 データがキャッシュ・デバイスに存在する場合、この値には、キャッシュ・デバイスに対する読み取り操作も含まれます。 この値は個別の読み取り要求の総数であって、読み取り要求のサイズ (バイト・カウント) を示すわけではありません。

Write Count
ターゲット・デバイスに対して実行された書き込み操作の総数。 この値は個別の書き込み要求の総数であって、書き込み要求のサイズ (バイト・カウント) を示すわけではありません。

Read Hit Count
ターゲット・デバイスに対して実行された、完全読み取りヒット である読み取り操作の総数。 完全読み取りヒット とは、読み取り要求がキャッシュ・デバイスによって完全に満たされる読み取り操作インスタンスです。 **Read Hit Count** の値は個別の読み取りヒット要求の総数であって、読み取り要求のサイズ (バイト・カウント) を示すわけではありません。 この値は、**Read Count** の値に含まれます。

Partial Read Hit Count
ターゲット・デバイスに対して実行された、部分読み取りヒット である読み取り操作の総数。 部分読み取りヒット とは、読み取り要求がキャッシュ・デバイスによって部分的に満たされる読み取り操作インスタンスです。 キャッシュ・デバイスに存在しない残りのデータは、ターゲット・デバイスから取得する必要があります。 **Partial Read Hit Count** の値は個別の読み取り要求の合計カウントであって、読み取り要求のサイズ (バイト・カウント) を示すわけではありません。 この値は、**Read Count** の値に含まれます。

Read Bytes Xfer
アプリケーションからターゲット・デバイスに対して実行されたすべての読み取り要求において、転送されたバイト数の合計。 この値は、完全読み取りヒット のデータ、部分読み取りヒット のデータ、およびターゲット・デバイスから取得されたデータの合計バイト・カウントを表します。

Write Bytes Xfer
アプリケーションからターゲット・デバイスに対して実行されたすべての書き込み要求において、転送されたバイト数の合計。

Read Hit Bytes Xfer
完全読み取りヒット のインスタンスの間に読み取られたバイト数の合計。

Partial Read Hit Bytes Xfer
部分読み取りヒット のインスタンスの間に読み取られたバイト数の合計。

Promote Read Count
データがキャッシュ・デバイスに取り込まれるときに、ターゲット・デバイスに対して実行された読み取りコマンドの総数。 この値は、データがキャッシュ・デバイスに取り込まれるときのインスタンスの数を示すわけではありません。ターゲット・ディスクへのデータの最大転送サイズが要求サイズより小さい場合、データをキャッシュ・デバイスに取り込むための 1 つの要求が複数の読み取り操作に分割される可能性があるためです。

Promote Read Bytes Xfer
データがキャッシュ・デバイスに取り込まれるときに、ターゲット・デバイスから読み取られたバイト数の合計。

ログイン名、システム ID、パスワード

オペレーティング・システムが、ユーザーに正しい環境を提供するためには、そのユーザーが誰であるかをシステムに知らせる必要があります。

ユーザーは、自分自身をオペレーティング・システムに認識させるために、ログイン名 (ユーザー ID またはユーザー名ともいいます) とパスワード を入力してログインします。 パスワードは、セキュリティーの 1 つの形です。 他人にログイン名を知られても、パスワードを知られなければ、ユーザーのシステムにログインできません。

システムをマルチユーザー・システムとして設定する場合は、許可ユーザーそれぞれがそのシステム上のアカウント、パスワード、ログイン名を持つことになります。 オペレーティング・システムは、各ユーザーのリソース使用をトラックします。 これを、システム・アカウント といいます。 各ユーザーには、システムのストレージ・スペースにファイルシステム と呼ばれる、専用領域が与えられます。 ユーザーがログインしたとき、システムには他に数多くのファイルがあっても、ファイルシステムにはそのユーザーのファイルしか含まれていないかのように見えます。

1 つのシステム上で、複数の有効なログイン名を持つことができます。 ログイン名を別のログイン名に変えたい場合でも、システムからログアウトする 必要はありません。 ログアウトすることなく、異なるログイン名を別のシェル内で同時に使用したり、同じシェル内で連続して使用できます。 また、システムが他のシステムに接続されたネットワークの一部である場合、ユーザーがログイン名を持っているシステムであれば、他のどのシステムにもログインできます。 これをリモート・ログイン といいます。

オペレーティング・システムでの作業が終了したら、ログアウトしてファイルやデータの安全を確保します。

オペレーティング・システムにログインする

オペレーティング・システムを使用するには、システムが実行中で、ユーザーがログインしていることが必要です。 オペレーティング・システムにログインすると、システムはユーザーを認識し、ユーザー環境を設定できるようになります。

システムは、1 日のうちの一定の時間、および週内の一定の曜日にしかログインできないように設定されている場合があります。 許可されている時間外にログインしようとすると、アクセスが拒否されます。 ユーザーのログインの時間は、システム管理者が検証できます。

ログイン・プロンプトのときログインします。 オペレーティング・システムにログインすると、自動的にユーザーのホーム・ディレクトリー (ログイン・ディレクトリー ともいいます) から始まることになります。

システムがオンになったら、システムにログインしてセッションを 開始します。

1. 「**login:**」プロンプトの後に、ログイン名を次のように入力します。

   ```
   login: LoginName
   ```

   例えば、ログイン名が denise であれば、次のように入力します。

   ```
   login: denise
   ```

2. 「**password:**」プロンプトが表示されたら、パスワードを入力します。

(パスワードを入力しても画面には表示されません。)

```
password: [your password]
```

パスワード・プロンプトが表示されない場合は、パスワードが定義されていません。オペレーティング・システムで作業を始めることができます。

マシンがオンになっていない場合は、ログインの前に以下の操作を行います。

1. 各接続デバイスの電源スイッチをそれぞれオンに設定します。
2. 電源スイッチをオン (**I**) に設定して、システム装置を始動します。
3. 3 桁コードの表示を確認します。 自己診断テストがエラーなしで完了すると、この 3 桁表示はブランクとなります。

注意すべきエラーが発生した場合は、3 桁のコードが表示されたままになり、システム装置が停止します。エラー・コードとリカバリーの詳細については、システム管理者に問い合わせます。

自己診断テストが正常に完了すると、以下のようなログイン・プロンプトが画面に表示されます。

```
login:
```

ログイン後は、オペレーティング・システムのセットアップ方法に応じて、コマンド・ライン・インターフェース (シェル) かグラフィカル・インターフェース (例えば、AIXwindows または CDE) のいずれかでシステムが始動します。

パスワードまたはユーザー名の構成について質問がある場合は、システム管理者に問い合わせてください。

複数回ログインする (login コマンド)

複数のプロジェクトで作業していて、別々のアカウントを維持したい場合、複数の同時ログインを行うことができます。 同じログイン名を使って、または異なるログイン名を使って、システムにログインすることができます。

注 : システムごとに、随時アクティブにできる最大数のログインがあります。 この数は、ユーザーのライセンス契約によって決まり、インストールによっても異なります。

例えば、 denise1 として既にログオンしていて、別のログイン名が denise2 の場合、プロンプトで次のように入力します。

```
login denise2
```

「**password:**」プロンプトが表示されたら、パスワードを入力します。 (パスワードを入力しても画面には表示されません。) これで、システム上で 2 つのログインを実行していることになります。

完全な構文については、「コマンド・リファレンス 第 *3* 巻」の **login** コマンドを参照してください。

システム上で別のユーザーになる (su コマンド)

su (ユーザーの切り替え) コマンドを使って、(ユーザーのログイン名がわかっている場合に) あるセッションに関連付けられたユーザー ID を変更できます。

例えば、ユーザー joyce に切り替えたい場合は、プロンプトで次のように入力します。

```
su joyce
```

「**password:**」プロンプトが表示されたら、ユーザー joyce のパスワードを入力します。 これで、ユーザー ID は joyce になりました。 パスワードがわからない場合は、この要求は拒否されます。

ユーザー ID が joyce であることを確認するには、**id** コマンドを使用します。

ログイン・メッセージを抑止する

正常にログインすると、**login** コマンドは、ログイン当日のメッセージ、このユーザーがログインに成功したときと失敗したときの最後の日付と時刻、確認情報 (通常の場合パスワード) の最後の変更以降にログインに失敗した回数の合計などのメッセージを表示します。 ホーム・ディレクトリーに .hushlogin ファイルを組み込むことによって、これらのメッセージを抑止できます。

ホーム・ディレクトリーのプロンプトのとき、次のコマンドを入力します。

```
touch .hushlogin
```

touch コマンドは、.hushlogin という空のファイルが存在しない場合は、それを作成します。 次回にログインしたとき、すべてのログイン・メッセージの出力が抑止されます。 当日のメッセージだけを残して、他のログイン・メッセージは抑止しておくようにシステムに指示することができます。

オペレーティング・システムからログアウトする (exit コマンドと logout コマンド)

オペレーティング・システムからログアウトするには、システム・プロンプトで次のいずれかを行います。

ファイルの終わり制御キー・シーケンス (Ctrl-D キー) を押します。

または

「exit」と入力します。

または

「logout」と入力します。

ログアウトすると、システムが「**login:**」プロンプトを表示します。

ユーザー ID の表示

指定されたユーザーのシステム ID を表示するには、**id** コマンドを使用します。 システム ID は、システムに対してユーザーおよびユーザー・グループを識別する数字です。

id コマンドは、該当する場合は、以下の情報を表示します。

- ユーザー名と実ユーザー ID
- ユーザーのグループの名前と実グループ ID
- ユーザーの補足グループの名前と補足グループ ID

例えば、プロンプトに次のように入力します。

```
id
```

システムは次のような情報を表示します。

```
uid=1544(sah) gid=300(build) euid=0(root) egid=9(printq) groups=0(system),10(audit)
```

この例では、ユーザーは、ユーザー名が ID 番号 1544 の sah、1 次グループ名が ID 番号 300 の build、実効ユーザー名が ID 番号 0 の root、実効グループ名が ID 番号 9 の printq、2 つある補足グループ名がそれぞれ ID 番号 0 と 10 の system と audit です。

例えば、プロンプトに次のように入力します。

```
id denise
```

システムは次のような情報を表示します。

```
uid=2988(denise) gid=1(staff)
```

この例では、ユーザー denise は、ユーザーの ID 番号が 2988、グループの ID 番号が 1 の 1 次グループ名 staff のみを持ちます。

完全な構文については、「コマンド・リファレンス 第 *3* 巻」の **id** コマンドを参照してください。

ログイン名を表示する方法 (whoami コマンドと logname コマンド)

同時に複数のログインを行っている場合、これらのログイン名、特にその時点で使用しているログイン名が分からなくなることがあります。 この情報の表示には、**whoami** および **logname** コマンドを使用できます。

whoami コマンドを使用する方法

使用されているログイン名を判別するには、プロンプトに次のように入力します。

```
whoami
```

システムは次のような情報を表示します。

```
denise
```

この例では、使用されているログイン名は denise です。

完全な構文については、「コマンド・リファレンス 第 *6* 巻」の **whoami** コマンドを参照してください。

who am i コマンドを使用する方法

who コマンドの変形の 1 つである **who am i** コマンドを使用すると、ログイン名、端末装置名、ログイン時刻を表示できます。 プロンプトに次のように入力してください。

```
who am i
```

システムは次のような情報を表示します。

```
denise   pts/0   Jun 21 07:53
```

この例では、ログイン名は denise、端末装置の名前は pts/0 で、このユーザーがログインしたのは 6 月 21 日の午前 7 時 53 分です。

完全な構文については、「コマンド・リファレンス 第 *6* 巻」の **who** コマンドを参照してください。

logname コマンドを使用する方法

who コマンドのもう 1 つの変形である **logname** コマンドでは、**who** コマンドの場合と同じ情報が表示されます。

プロンプトに次のように入力してください。

```
logname
```

システムは次のような情報を表示します。

```
denise
```

この例では、ログイン名は denise です。

オペレーティング・システム名を表示する方法 (uname コマンド)

オペレーティング・システムの名前を表示するには、**uname** コマンドを使用します。

例えば、プロンプトに次のように入力します。

```
uname
```

システムは次のような情報を表示します。

```
AIX
```

この例では、オペレーティング・システム名は AIX です。

完全な構文については、「コマンド・リファレンス 第 *5* 巻」の **uname** コマンドを参照してください。

使用中のシステム名を表示する方法 (uname コマンド)

ユーザーがネットワーク上にいて、自分のシステム名を表示するには、**-n** フラグを付けた **uname** コマンドを使用します。 ネットワークは、システム名によって、ユーザーのシステムを識別します。システム名は、ログイン ID とは異なります。

例えば、プロンプトに次のように入力します。

```
uname -n
```

システムは次のような情報を表示します。

```
barnard
```

この例では、システム名は barnard です。

完全な構文については、「コマンド・リファレンス 第 *5* 巻」の **uname** コマンドを参照してください。

ログインしているすべてのユーザーの 表示

現在ローカル・システムにログインしているすべてのユーザーに 関する情報を表示するには、**who** コマンドを使用します。

ログイン名、システム名、ログインの日付と時刻などの情報が表示されます。

注 : このコマンドは、ローカル・ノード上のログイン・ユーザーの識別のみを行います。

現在ローカル・システム・ノードを使用しているユーザーについての情報を表示するには、次のように入力します。

```
who
```

システムは次のような情報を表示します。

```
joe   lft/0 Jun 8 08:34
denise   pts/1 Jun 8 07:07
```

この例では、ユーザー joe が端末装置 lft/0 を使用しており、6 月 8 日の午前 8 時 34 分にログインしています。

who コマンドを参照してください。

パスワード

固有のパスワードを持つことで、ファイルに関してある程度のシステム・セキュリティーが確保されます。

システムには、アカウントごとに対応するパスワードがあります。 セキュリティーは、許可されていない人がシステムへのアクセスを取得したり、他のユーザーのファイルを許可なく変更することを防ぐための、コンピューター・システムの重要な部分です。 また、セキュリティーによって、使用できるコマンドやアクセスできるファイルについて、一部のユーザーに独占的な特権を認可できます。 保護のために、一部のシステム管理者は、一定のコマンドやファイルに限って、ユーザーにアクセスを許可します。

パスワードの指針

ユーザーは、固有のパスワードを持つ必要があります。 パスワードは共有できません。 パスワードは、会社の他のすべての資産を保護するのと同様に保護してください。 パスワードを作成するときは必ず、推測しにくいものにしてください。ただし、自分が忘れないように書き留めておく必要があるほど、難しいものにはしないでください。

わかりにくいパスワードを使用することで、ユーザー ID が保護できます。 自分の名前や生年月日のような、個人情報をもとにしたパスワードは適しません。 よく使われる一般的な言葉も簡単に推測される可能性があります。

6 文字以上で、アルファベット以外の文字も含まれているのがよいパスワードです。 見慣れない言葉の組み合わせや、意図的にスペルを間違えた言葉もよい選択です。

注 : 覚えにくく、書き留めておく必要があるパスワードは、良いパスワードではありません。

パスワードを選択するときは、以下の指針に従ってください。

- ユーザー ID をパスワードとして使用しないでください。 ユーザー ID を逆にしたり、2 回繰り返したり、少し変えて使用するのもよくありません。
- パスワードは再使用しないでください。 システムは、パスワードの再使用を拒否するようセットアップされていることがあります。
- 人名をパスワードとして使用しないでください。
- オンラインのスペルチェック辞書に載っている可能性のある言葉は、パスワードとして使用しないでください。
- 長さが 6 文字未満のパスワードは使用しないでください。
- わいせつな言葉は使用しないでください。わいせつな言葉は、パスワードを推測する際に最初に試される言葉に入っています。
- 書き留めておかなくても済むように、覚えやすいパスワードを使用してください。
- 文字と数字の両方を使用し、大文字と小文字の両方を入れたパスワードを使用してください。
- パスワードには、2 つの言葉を数字で区切って使用してください。
- 発音できるパスワードを使用してください。 その方が覚えやすくなります。
- パスワードは書き留めておかないようにしてください。 どうしても書き留めておかなければならない場合は、鍵を掛けたキャビネットなどの物理的に安全な場所に保管してください。

パスワードの変更 (passwd コマンド)

パスワードを変更するには、**passwd** コマンドを使用します。

1. プロンプトに次のように入力してください。

   ```
   passwd
   ```

   まだパスワードを持っていない場合は、ステップ 2 をスキップしてください。
2. 次のプロンプトが表示されます。

   ```
   Changing password for UserID
   UserID's Old password:
   ```

   この要求により、システムを離れている間に他の無許可のユーザーにパスワードを変更されてしまわないようにします。 現在のパスワードを入力して、Enter キーを押します。
3. 次のプロンプトが表示されます。

   ```
   UserID's New password:
   ```

   新規パスワードを入力し、Enter キーを押します。
4. 次のプロンプトが表示され、新規パスワードを再入力するように要求されます。

   ```
   Enter the new password again:
   ```

   この要求により、再作成できない、誤って入力した文字列をパスワードに設定してしまわないようにします。

完全な構文については、「コマンド・リファレンス 第 *4* 巻」の **passwd** コマンドを参照してください。

パスワードのヌル化 (passwd コマンド)

ログインのたびにパスワードを入力したくない場合は、パスワードをヌル (ブランク) に設定します。

パスワードをヌルに設定するには、次のように入力します。

```
passwd
```

新規パスワードの入力を求めるプロンプトが表示されたら、Enter か Ctrl-D キーを押します。

passwd コマンドは、パスワードの再入力を要求しません。 ヌルのパスワードを検証するメッセージが表示されます。

詳細と完全な構文については、**passwd** コマンドを参照してください。

ログイン名、システム ID、パスワードに関するコマンドのまとめ

ログイン名、システム ID、およびパスワードを操作するためのコマンドが用意されています。

ログインとログアウト・コマンド

項目	説明
login	セッションを開始します
logout	すべてのプロセスを停止します
shutdown	システム操作を終了します
su	セッションに関連するユーザー ID を変更します
touch	ファイルのアクセス時刻および変更時刻を更新するか、または空のファイルを作成します

ユーザーおよびシステムの識別コマンド

項目	説明
id	指定されたユーザーのシステム識別を表示します
logname	ログイン名を表示します
uname	現在のオペレーティング・システムの名前を表示します
who	現在ログインしているユーザーを識別します
whoami	ユーザーのログイン名を表示します

パスワード・コマンド

項目	説明
passwd	ユーザーのパスワードを変更します

Common Desktop Environment

Common Desktop Environment (CDE)を使用すれば、ネットワークに接続されたデバイスやツールがどこにあるかを知る必要なしに、それらにアクセスできます。 オブジェクトを単にドラッグしドロップするだけで、アプリケーション間でデータを交換できます。

これまでは複雑なコマンド・ライン構文を必要としたタスクの多くが、より容易に、しかもどのプラットフォームでも同じように実行できるようになります。 例えば、中央でアプリケーションを構成し、ユーザーに分配することができます。 システム管理者は、サポートしているユーザーに対して、アプリケーションのセキュリティー、可用性、およびインターオペラビリティーも中央で管理できます。

注 : Common Desktop Environment (CDE) 1.0 ヘルプ・ボリューム、Web ベースの資料、およびハードコピー・マニュアルでは、デスクトップのことを、CDE、AIXwindows デスクトップ、CDE 1.0、またはデスクトップと呼ぶ場合があります。

デスクトップ自動開始を使用可能化する方法および使用不可にする方法

システムをオンにするときに、CDE を自動的に開始するようにシステムをセットアップすると 、より便利な場合があります。

これは、System Management Interface Tool (SMIT) を使用するか、またはコマンド・ラインから行うことができます。

前提条件

デスクトップ自動開始を使用可能化または使用不可にするには、root ユーザー権限が必要です。

デスクトップ自動開始を使用可能化または使用不可にする方法を決める場合は、次の表を参照します。

タスク	SMIT 高速パス	コマンドまたはファイル
デスクトップの自動開始の使用可能化 [1]	**smit dtconfig**	**/usr/dt/bin/dtconfig -e**
デスクトップの自動開始の使用不可 [1]	**smit dtconfig**	**/usr/dt/bin/dtconfig -d**

注 : この作業を終了後、システムを再始動します。

手動による CDE の開始

CDE を手動で開始する場合は、この手順を使用します。

1. root としてシステムにログインします。
2. コマンド・ラインで次のように入力します。

```
/usr/dt/bin/dtlogin -daemon
```

「**Desktop Login (デスクトップ・ログイン)**」画面が表示されます。 ログインすると、デスクトップ・セッションが始動されます。

手動による CDE の停止

ログイン・マネージャーを手作業で停止すると、ログイン・マネージャーが始動したすべての X サーバーおよびデスクトップ・セッションは停止されます。

1. 端末装置エミュレーター・ウィンドウを開き、root としてログインします。
2. 次のように入力して、ログイン・マネージャーのプロセス ID を入手します。

```
cat /var/dt/Xpid
```

3. 次のように入力して、ログイン・マネージャーを停止します。

```
kill -term process_id
```

デスクトップ・プロファイルの変更

デスクトップにログインする 場合、シェル環境ファイル (`.profile` または `.login`) を自動的には読み取りません。 デスクトップは、ログインする前に X サーバーを実行します。したがって、`.profile` ファイルまたは `.login` ファイルに用意されている機能は、デスクトップのログイン・マネージャーが提供する必要があります。

ユーザー特有の環境変数は、*/Home Directory/* `.dtprofile` に設定されます。 このファイルのテンプレートは `/usr/dt/config/sys.dtprofile` にあります。 デスクトップにだけ適用される変数とシェル・コマンドを `.dtprofile` に入れます。 `.dtprofile` の終わりに行を追加して、シェル環境ファイルを組み込みます。

システム全体の環境変数は、ログイン・マネージャー構成ファイルに設定できます。 環境変数の構成の詳細については、「*Common Desktop Environment 1.0: Advanced User's and System Administrator's Guide*」を参照してください。

Common Desktop Environment 用ディスプレイおよび端末装置の追加および除去

Common Desktop Environment 用ディスプレイおよび 端末装置は、追加および除去できます。

ログイン・マネージャーは、単一ローカル・ビットマップまたはグラフィックス・コンソールが装備されているシステムから始動できます。 ただし、その他の多くの状態においても可能です (下図を参照)。 Common Desktop Environment は以下のものから始動できます。

- ローカル・コンソール
- リモート・コンソール
- ネットワーク上のホスト・システムで実行中のビットマップ・ディスプレイおよびキャラクター・ディスプレイの X ターミナル・システム

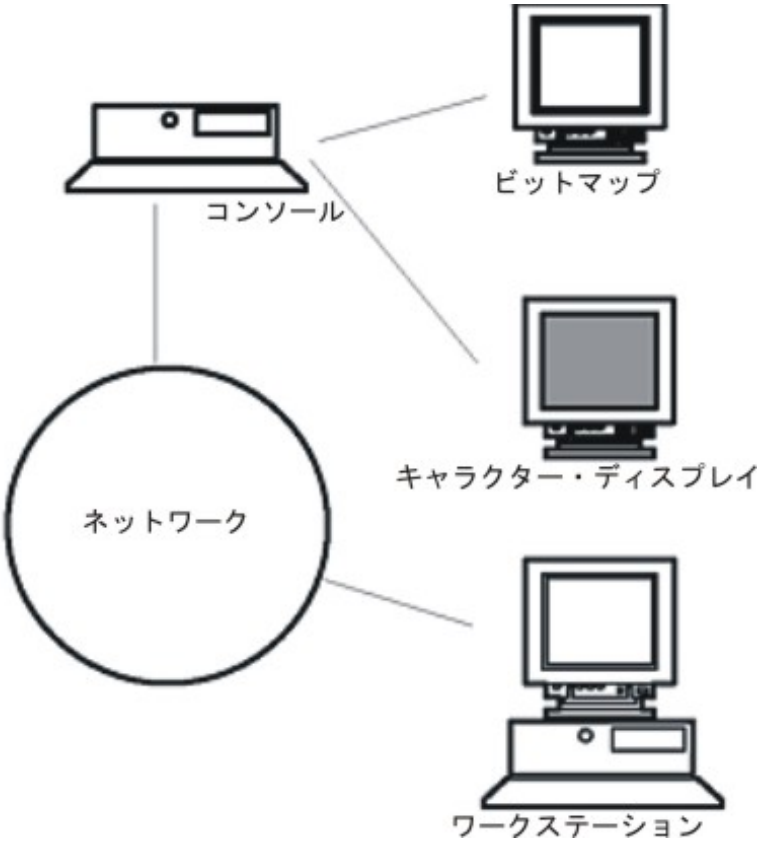


図 *17. CDE* インターフェース・ポイント

X 端末システムは、ディスプレイ・デバイス、キーボード、および X サーバーのみを実行するマウスから構成されています。 Common Desktop Environment を含むクライアントは、ネットワーク上の 1 つ以上のホスト・システムで稼働します。 クライアントからの出力は、X ターミナル・ディスプレイに送信されます。

以下のログイン・マネージャー構成タスクによって、多数の構成のサポートが可能です。

- ローカル・ディスプレイを除去する方法
- ASCII 端末装置またはキャラクター・ディスプレイ 端末装置を追加する方法

ワークステーションを X 端末装置として使用するには、コマンド・ラインで次のように入力します。

```
/usr/bin/X11/X -query hostname
```

X ターミナルとして作動するワークステーションの X サーバーは、以下のことができなければなりません。

- XDMCP および **-query** コマンド・ライン option のサポート。
- 端末装置ホストに対する xhost アクセス権の提供 (/etc/X*.hosts の中で)。

ローカル・ディスプレイを 除去するには、そのエントリーを /usr/dt/config ディレクトリーの Xservers ファイルから除去します。

キャラクター・ディスプレイ 、または *ASCII* 端末 は、端末装置がビットマップ・デバイスではない場合の構成です。

ビットマップ・ディスプレイがない 場合に ASCII コンソールまたはキャラクター・ディスプレイ ・コンソールを追加するには、次のステップを実行します。

1. /etc/dt/config/Xservers ファイルがない場合、/usr/dt/config/Xservers ファイルを /etc/dt/config ディレクトリーにコピーします。
2. Xservers ファイルを /etc/dt/config にコピーする必要がある場合、/etc/dt/config/Xconfig の Dtlogin.servers: 行を次のように変更または追加する必要があります。

   ```
   Dtlogin*servers: /etc/dt/config/Xservers
   ```

3. X サーバーを始動する /etc/dt/config/Xservers の中の行をコメント化します。

   ```
   # * Local local@console /path/X :0
   ```

   これにより、**ログイン・オプション・メニュー**が使用不可になります。
4. ログイン・マネージャー構成ファイルを読み取ります。

ビットマップ・ディスプレイが 存在する場合にキャラクター・ディスプレイ ・コンソールを追加するには、次のステップを実行します。

1. /etc/dt/config/Xservers ファイルがない場合、/usr/dt/config/Xservers ファイルを /etc/dt/config ディレクトリーにコピーします。
2. Xservers ファイルを /etc/dt/config にコピーする必要がある場合、/etc/dt/config/Xconfig の Dtlogin.servers: 行を次のように変更または追加する必要があります。

   ```
   Dtlogin*servers: /etc/dt/config/Xservers
   ```

3. X サーバーを始動する /etc/dt/config/Xservers の中の行を、次のようになるように編集します。

   ```
   * Local local@none /path/X :0
   ```

4. ログイン・マネージャー構成ファイルを読み取ります。

Common Desktop Environment 用のディスプレイ・デバイスのカスタマイズ

Common Desktop Environment ログイン・マネージャーを、2 つ以上のディスプレイ ・デバイスをもっているシステムで実行できるように構成できます。

システムが複数のディスプレイを 組み込んでいる場合、以下の構成要件を満たす必要があります。

- サーバーは、各ディスプレイで 始動する必要があります。
- 各ディスプレイに 対して、非ウィンドウ表示モードを構成する必要があります。

各ディスプレイに 対して個別の dtlogin リソースを使用することが必要または望ましい場合があります。

また、各ディスプレイ ・デバイスに対して、個別のシステム 全体環境変数を使用することが必要または望ましい場合があります。

各ディスプレイ・デバイスでサーバーを始動する方法

各ディスプレイ ・デバイス上のサーバーは、この手順を使用して始動します。

1. /etc/dt/config/Xservers ファイルがない場合、/usr/dt/config/Xservers ファイルを /etc/dt/config ディレクトリーにコピーします。

2. Xservers ファイルを /etc/dt/config にコピーする必要がある場合、/etc/dt/config/Xconfig の Dtlogin.servers: 行を次のように変更する必要があります。

```
Dtlogin*servers: /etc/dt/config/Xservers
```

3. /etc/dt/config/Xservers を編集して、各ディスプレイ ・デバイスで X サーバーを始動します。

サーバーを始動する一般的な構文は、次のとおりです。

```
DisplayName DisplayClass DisplayType [ @ite ] Command
```

関連した内部端末装置エミュレーター (ITE) を装備しているディスプレイのみが、非ウィンドウ表示モードで操作できます。 非ウィンドウ表示モードは、ディスプレイのデスクトップを 一時的に使用不可にし、*getty* プロセスが始動済みでなければ、そのプロセスを実行します。 getty プロセスは、端末装置のタイプを設定し、ログイン・プロセスで使用される UNIX プログラムです。

これによりログインし、Common Desktop Environment では実行できないタスクを実行できます。 ログアウトすると、ディスプレイ ・デバイス用にデスクトップが再始動されます。 非ウィンドウ表示モードが開始されるときに、ディスプレイ ・デバイスで getty プロセスが実行中でなければ、ログイン・マネージャーはそれを始動します。

デフォルト構成では、ite が省略されると、display:0 は ITE (/dev/console) に関連付けられます。

別のディスプレイの ITE としてのセットアップ

この手順を使用して、別のディスプレイを ITE としてセットアップします。

別のディスプレイを ITE として指定するには、次のようにします。

- ITE ディスプレイで、ITE をキャラクター・デバイスに設定します。
- その他のすべてのディスプレイで 、ITE を none に設定します。

以下の例で、sysaaa:0 上の 3 つのローカル・ディスプレイでサーバーを 始動する、Xserver ファイル内のエントリーを示します。 ディスプレイ :0 はコンソール (ITE) です。

```
sysaaa:0 Local local /usr/bin/X11/X :0
sysaaa:1 Local local /usr/bin/X11/X :1
sysaaa:2 Local local /usr/bin/X11/X :2
```

ホスト sysbbb で、ビットマップ・ディスプレイ :0 は ITE ではありません。ITE は、デバイス /dev/ttyi1 に関連付けられています。 Xserver ファイルにおける以下の項目は、「非ウィンドウ表示」モードが :1 で有効になっている、2 つのビットマップ・ディスプレイでサーバーを 始動します。

```
sysaaa:0 Local local@none /usr/bin/X11/X :0
sysaaa:1 Local local@ttyi1 /usr/bin/X11/X :1
```

Xconfig でのディスプレイ名のセットアップ

/etc/dt/config/Xconfig のディスプレイ 名用に、正規の hostname:0 構文は使用できません。

Xconfig にディスプレイ 名を指定するには、次のようにします。

- コロンの代わりに下線を使用します。
- 完全修飾ホスト名では、ピリオドの代わりに下線を使用します。

以下の例で、Xconfig でのディスプレイ 名のセットアップを示します。

```
Dtlogin.claaa_0.resource: value
Dtlogin.sysaaa_prsm_ld_edu_0.resource: value
```

各ディスプレイに個別のログイン・マネージャー・リソースを使用する方法
ディスプレイごとに異なるログイン・マネージャー・リソースを使用するには、次のステップを実行します。

1. /etc/dt/config/Xconfig ファイルがない場合、/usr/dt/config/Xconfig ファイルを /etc/dt/config ディレクトリーにコピーします。
2. /etc/dt/config/Xconfig ファイルを編集して、ディスプレイごとに異なるリソース・ファイルを指定します。
   次に例を示します。

   ```
   Dtlogin.DisplayName.resources: path/file
   ```

   ここで、*path* は、使用する Xresource ファイルのパス名で、*file* は、使用する Xresource のファイル名です。
3. Xconfig ファイルの中で指定されているリソース・ファイルそれぞれを作成します。
   言語特有の Xresources ファイルは、/usr/dt/config/<LANG> にインストールされます。
4. 各ファイルに、そのディスプレイの dtlogin リソースを入れます。

以下の例で、3 つのディスプレイに異なるリソース・ファイルを指定する Xconfig ファイルの行を示します。

```
Dtlogin.sysaaa_0.resources: /etc/dt/config/Xresources0
Dtlogin.sysaaa_1.resources: /etc/dt/config/Xresources1
Dtlogin.sysaaa_2.resources: /etc/dt/config/Xresources2
```

各ディスプレイに個別のスクリプトを実行する方法
この手順は、特定のディスプレイの特定のスクリプトを実行する場合に使用します。

1. /etc/dt/config/Xconfig ファイルがない場合、/usr/dt/config/Xconfig ファイルを /etc/dt/config ディレクトリーにコピーします。
2. /etc/dt/config/Xconfig 内の始動、リセット、およびセットアップのリソースを使用して、各ディスプレイに異なるスクリプトを指定します (これらのファイルは、Xstartup、Xreset、および Xsetup ファイルに代わって実行されます)。

   ```
   Dtlogin*DisplayName*startup: /path/file
   Dtlogin*DisplayName*reset: /path/file
   Dtlogin*DisplayName*setup: /path/file
   ```

ここで、*path* は、使用するファイルのパス名で、*file* は、使用するファイルのファイル名です。 始動スクリプトは、ユーザーがログインした後、Common Desktop Environment セッションが始動する前に、root として実行されます。

スクリプト /usr/dt/config/Xreset を使用して、Xstartup ファイルの中に行った設定を逆にできます。 Xreset ファイルは、ユーザーがログアウトしたときに実行されます。

以下の例で、2 つのディスプレイに異なるスクリプトを指定する Xconfig ファイルの行を示します。

```
Dtlogin.sysaaa_0*startup:   /etc/dt/config/Xstartup0
Dtlogin.sysaaa_1*startup:    /etc/dt/config/Xstartup1
Dtlogin.sysaaa_0*setup:      /etc/dt/config/Xsetup0
Dtlogin.sysaaa_1*setup:      /etc/dt/config/Xsetup1
Dtlogin.sysaaa_0*reset:      /etc/dt/config/Xreset0
Dtlogin.sysaaa_1*reset:      /etc/dt/config/Xreset1
```

各ディスプレイに個別のシステム全体環境変数を設定する方法
この手順は、システム全体環境変数を各ディスプレイにカスタマイズする場合に使用します。

各ディスプレイに個別のシステム全体環境変数を設定するには、次のようにします。

1. /etc/dt/config/Xconfig ファイルがない場合、/usr/dt/config/Xconfig ファイルを /etc/dt/config ディレクトリーにコピーします。

2. ディスプレイごとに 個別に、環境リソースを /etc/dt/config/Xconfig に設定します。

```
Dtlogin*DisplayName*environment: value
```

次の規則が各ディスプレイの 環境変数に適用されます。

- スペースまたはタブで変数の割り当てを分離する。
- 環境リソースを使用して TZ および LANG を設定しない。
- Xconfig ファイル内ではシェル処理をしない。

以下の例で、2 つのディスプレイに 変数を設定する Xconfig ファイルの行を示します。

```
Dtlogin*syshere_0*environment:EDITOR=vi SB_DISPLAY_ADDR=0xB00000
Dtlogin*syshere_1*environment: EDITOR=emacs ¥
         SB_DISPLAY_ADDR=0xB00000
```

ホスト・イーサネット・アダプターを持つライブ区画モビリティー

IBM PowerVM® ソフトウェアのホスト・イーサネット・アダプター (HEA) フィーチャーを持つライブ区画モビリティー (LPM) を使用 すると、HEA が移行区画に割り当てられている間に AIX LPAR およびホストされているアプリケーションを ある物理区画から別の物理区画に移行できます。

移行中に、HEA は移行区画から削除され、移行が完了したとき HEA は区画に復元されません。 ただし、ご使用のネットワーク接続は影響を受けません。

HEA を持つライブ区画モビリティーの要件

HEA を持つ LPM の使用を開始する前に、ご使用のシステム環境が構成要件およびアクセス要件を満たしていることを確認する必要があります。

区画要件

- CEC ソースおよび CEC ターゲットは区画の移行に対応している必要があります。
- ソース AIX LPAR は、そのプロファイルに要求される設定を持つ物理リソースを持ってはなりません。
- ソース AIX LPAR は、HEA 以外の物理リソースを持ってはなりません。

アクセス要件

- ユーザーは、移行を望む区画上にルート権限を持つ必要があります。
- ユーザーは、ソース HMC および宛先 HMC 上への区画の移行に必要な、hscroot 権限または同等の権限を持つ必要があります。

構成要件

- HEA は区画プロファイル内で必須設定値を持ってはなりませんが、 プロファイル内で望ましい設定値を持つことができます。
- すべての HEA は、イーサチャネルの下では基本アダプターとして 構成する必要があります。
- イーサチャネル内のすべての基本アダプターは、HEA でなければなりません。
- イーサチャネルのバックアップ・アダプターは、仮想イーサネット・アダプターでなければなりません。
- 最小 1 つのイーサチャネルは HEA 付きの基本アダプターとして 構成し、仮想イーサネット・アダプターはバックアップ・アダプターとして 構成する必要があります。
- 最大 4 つのイーサチャネルがサポートされます。
- イーサチャネル・ フェイルオーバーは機能でなければなりません。
- ソース・システムおよびターゲット ・システムの双方が区画の移行用にセットアップされていることを検査する必要があります。

- 2 つの HMC 間で移行中の場合、ソース HMC およびリモート HMC 間で SSH 認証をセットアップする必要があります。 移行を開始する前に ソース HMC 上で **mkauthkeys** コマンドを実行する必要があります。

HEA を持つライブ区画モビリティーの実行

SMIT インターフェースを 使用して HEA を持つ LPM を実行できます。

HEA を持つ LPM の使用を試みる前に、237 ページの『HEA を持つライブ区画モビリティーの要件 』 トピックを参照してください。

HEA を持つ LPM の区画の移行を実行する前に、以下のステップを実行します。

1. コマンド・プロンプトから、次の SMIT 高速パス **smitty migration** を入力して、ホスト・イーサネット・アダプター (HEA) を持つライブ区画モビリティーのメニューを表示します。
2. ソース HMC ホスト名または IP アドレスを指定します。
3. ソース HMC ユーザー名を指定します。
4. 同一の HMC によりソース・システムおよび宛先システムが管理されている場合、 **no** を入力し、ステップ 5 に移動します。 異なる HMC によりソース・システムおよび宛先システムが管理されている場合、 **yes** を入力し、以下のステップを実行します。

   注 : **yes** を入力する前に ソース HMC 上で **mkauthkeys** コマンドを実行する必要があります。

   a) リモート HMC ホスト名または IP アドレスを指定します。

   b) ソース HMC ユーザー名を指定します。
5. ソース・システムの名前を指定します。
6. 宛先システムの名前を指定します。
7. 移行を望む区画の名前を指定します。
8. 妥当性検査なしで移行を望む場合、**no** を入力します。 移行の妥当性検査のみの実行を望む場合、**yes** を入力します。 **yes** を指定すると移行は実行されず、 妥当性検査のみが実行されます。

   注 : 区画の移行を実行する前に、区画の移行の妥当性検査を実行する必要があります。
9. すべてのフィールドが正しい情報を持っていることを検査し、**Enter** を押して移行を実行します。

   注 : パスワードを入力するためのプロンプトが 2 度出されます。 ステップ 4b で直前に指定した、ソース HMC ユーザー名のパスワードを入力します。

この例では、区画 X は、HMC A により管理されている CEC C から HMC B により管理されている CEC D に移行中です。 HMC A と HMC B 間の認証用に、HMC A 上で **mkauthkeys** コマンドを実行します。

この移行プロセスの場合、以下の値が SMIT で指定されます。

```
Source HMC Hostname or IP address: A (ソース HMC ホスト名または IP アドレス: A)
Source HMC username: hscroot (ソース HMC ユーザー名: hscroot)
Migration between two HMCs: yes (2 つの HMC 間の移行: はい)
    Remote HMC hostname or IP address: B (リモート HMC ホスト名または IP アドレス： B)
    Remote HMC username: hscroot (リモート HMC ユーザー名: hscroot)
Source system: C (ソース・システム： C)
Destination system: D (宛先システム： D)
Migrating Partition name: X (移行中の区画名： X)
Migration validation only: no  (移行の妥当性検査のみ： いいえ)
```

もう 1 つの例は、区画 X が、HMC A により管理 されている CEC C から、同様に HMC A により管理されている CEC D に移行する場合です。

この移行プロセスの場合、 以下の値が SMIT で指定されます。

```
Source HMC Hostname or IP address: A (ソース HMC ホスト名または IP アドレス: A)
Source HMC username: hscroot (ソース HMC ユーザー名: hscroot)
Migration between two HMCs: no (2 つの HMC 間の移行: いいえ)
    Remote HMC hostname or IP address: (リモート HMC ホスト名または IP アドレス：)
    Remote HMC username: (リモート HMC ユーザー名: )
Source system: C (ソース・システム： C)
Destination system: D (宛先システム： D)
```

```
Migrating Partition name: X（移行中の区画名： X）
Migration validation only: no （移行の妥当性検査のみ： いいえ）
```

移行に障害が発生した場合、ソース HMC からライブ区画モビリティーを実行する手順に従います。

LPM を使用した NIM クライアントの移行

マシンをある物理サーバーから別の物理サーバーに移動するためにライブ区画モビリティー (LPM) が使用され、そのマシンがネットワーク・インストール管理 (NIM) クライアントとして定義されている場合、NIM 管理者は、LPM の移行の完了後、新規のハードウェアの値を反映させるために、NIM クライアントの *cpuid* 属性を更新する必要があります。

cpuid 属性を更新するには、以下のステップを実行します。

1. NIM クライアントで、次のコマンドを実行して新規の cpuid ID を入手します。

   ```
   uname -a
   ```

2. NIM マスターで、次のコマンドを実行します。

   ```
   nim -o change -a cpuid=cpuid client
   ```

   注 : AIX オペレーティング・システムで Cluster Systems Management (CSM) のサポートが廃止されたため、現在 `OS_install` ネットワーク・インストーラーでは、Linux オペレーティング・システムのインストールはサポートされていません。

DLPAR 用アダプターの再配置

動的ロジカル・パーティショニング (DLPAR) 操作用にアダプターを 再配置する前に、グラフィックス・アダプターを構成する必要があります。

FC 5748 などのグラフィックス・アダプターを 動的に再配置するには、以下の手順に従います。

1. グラフィックス・アダプター (例えば、/dev/lft0) を使用している現行プロセス (デスクトップ、および Xserver) がないことを確認します。
2. コンソールが lft0 に設定されていないことを検査します。 定義済み、または 使用可能な依存インターフェース (lft または rcm) を識別するには、次のコマンドを入力します。

   ```
   lsdev -C | grep lft
   lsdev -C | grep rcm
   ```

   グラフィックス・アダプターを 定義した後で親 PCI アダプターを 取得するには、次のコマンドを入力します。

   ```
   odmget -q name=<cortina adapter name. for instance cor0> CuDv
   ```

   このコマンドは、親と Cortina アダプターに関する情報を提供します。
3. このアダプターを DLPAR 用に準備するために、すべての依存インターフェースを 削除するには、 次のコマンドを入力します。

   ```
   # pdisable lft0

   # rmdev -l rcm0
   rcm0 Defined

   # rmdev -l lft0
   lft0 Defined

   # rmdev -Rdl pci23
   cor0 deleted
   pci23 deleted
   ```

管理コンソールのインターフェースは、グラフィックス・アダプター 上の DLPAR 操作に使用できます。

ループバック・デバイス

ループバック・デバイスは、ファイルにアクセスするため、ブロック・デバイスとして使用できるデバイスです。

ループバック・ファイルには、ISO イメージ、ディスク・イメージ、ファイルシステム、または 論理ボリューム・イメージを収容できます。 例えば、ループバック・デバイスに CD-ROM ISO イメージを接続し、マウントすることにより、CD-ROM デバイスにアクセスするのと 同じ方法で、CD-ROM ISO イメージにアクセスできます。

ループバック・デバイスを作成し、指定されたファイルをそのループバック・デバイスにバインドし、ループバック・デバイスを マウントするには、**loopmount** コマンドを使用します。 ループバック・デバイスにすでにマウントされていた、イメージ・ファイルを アンマウントし、ループバック・デバイスを取り外すには、**loopumount** コマンドを使用します。 AIX 内のループバック・デバイスの数には制限がありません。 ループバック・デバイスはデフォルト設定では決して作成されません。ループバック・デバイスは、明示的に作成する必要があります。 ループバック・デバイスのブロック・サイズは、 常に 512 バイトです。

新規の新規デバイスは **mkdev** コマンドでも作成 でき、**chdev** コマンドで変更でき、そして **rmdev** コマンドで除去できます。 デバイスは 作成後、基調を成すイメージにアクセスするためマウントするか、またはロー I/O 用のブロック・デバイスとして使用することができます。 基調を成すイメージについての 情報は、**ioctl (IOCINFO)** コマンドで取り出すことができます。

以下の制約事項が AIX のループバック ・デバイスに適用されます。

- ディスク・イメージ上で **varyonvg** コマンドはサポートされません。
- CD ISO、および DVD UDF+ISO、およびその他の CD/DVD イメージは読み取り専用フォーマットでのみサポートされます。
- イメージ・ファイルは、1 つのループバック・デバイスにのみ関連付け可能です。
- ループバック・デバイスは、ワークロード区画ではサポートされません。

特記事項

本書は米国が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。 日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。 本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。 これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。 ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒 *103-8510*
東京都中央区日本橋箱崎町 *19* 番 *21* 号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。 本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、 自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

記載されている性能データとお客様事例は、例として示す目的でのみ提供されています。 実際の結果は特定の構成や稼働条件によって異なります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関す

る実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述は、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。 記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。 より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、類似する個人や企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。 従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、 次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (年).

このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。

© Copyright IBM Corp. _年を入れる_.

プライバシー・ポリシーに関する考慮事項

サービス・ソリューションとしてのソフトウェアも 含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。 IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。 ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項をご確認ください。

この「ソフトウェア・オファリング」は、Cookie もしくはその他のテクノロジーを使用して個人情報を収集することはありません。

この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。

このような目的での Cookie を含む様々なテクノロジーの使用の詳細については、IBM の『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

Windows は、Microsoft Corporation の米国およびその他の国における商標です。