Hewlett Packard Enterprise

UNIX の教科書

～はじめよう! Windows と Linux からのステップアップ～

この連載では、HP-UX の初心者を対象に、UNIX の世界で伝統的な CUI(キャラクタ・ユーザ・インタフェース)であるコマンドラインの操作を学んでいきます。現在パソコンなどで主流の GUI(グラフィカル・ユーザ・インタフェース)によるマウスでメニューを選択したりアイコンをクリックしたりという直感的な操作と比較すると、コマンドラインではすべてのコマンドをキーボードからタイプする必要があります。そのため、最初は取っつきにくい印象があるかもしれません。でも、大丈夫、段階的に学んでいけば決しておそれることはありません。

《連載期間：2007 年 11 月～2010 年 3 月》

登場人物紹介

マリー先生

HP-UX のエキスパート。
HP-UX のことなら何でもお任せ！
HP-UX 普及のため世界を飛び回るスーパーウーマン。

四色君

これまで使ったことのあるパソコンは Windows のみ。ただし、コマンドプロンプトは未経験

タックス君

Windows、Linux、Mac の経験あり。ただし、操作は GUI でやることが多く、コマンドラインはほとんど使ったことがない。

UNIX の教科書「基礎編」

第 1 日目 : ログインしてコマンドを実行してみる
第 2 日目 : ディレクトリやファイルを操作してみる
第 3 日目 : シェルの基本を知る
第 4 日目 : ファイルの基本操作
第 5 日目 : vi エディタの操作（基本編）
第 6 日目 : リダイレクションとパイプを活用する
期末試験

UNIX の教科書「応用編」

第 1 日目 : grep コマンドと正規表現
第 2 日目 : ファイルの検索
第 3 日目 : vi エディタの操作（活用編）
第 4 日目 : ファイルの圧縮とアーカイブ
第 5 日目 : ジョブとプロセスの操作
第 6 日目 : シェルの環境設定
第 7 日目 : シェルスクリプトでより便利に
期末試験

UNIX の教科書「運用編」

第 1 日目 : ユーザーの管理
第 2 日目 : ファイルの安全管理（その 1）
第 3 日目 : ファイルの安全管理（その 2）
第 4 日目 : ネットワークの基本を理解する
第 5 日目 : ネットワーク関連のコマンド
第 6 日目 : システム情報の表示
期末試験

UNIX の教科書「基礎編」

―目次―

期末試験

Windows や Linux の経験がある方を対象に、初歩の初歩から HP-UX について解説してきた「UNIX の教科書　基礎編」全 6 回、いかがでしたか？内容はご理解いただけたでしょうか。今回は、「期末試験：基礎編」として全 10 問の試験を用意しました。いずれもこれまでの連載からの出題ですので、ぜひ挑戦してみてください。間違えてしまった場合は正解を参考に復習し、応用編を始める前に完璧にしておきましょう。 (2008 年 5 月)

UNIX の教科書「基礎編」

第 1 日目：ログインしてコマンドを実行してみる

2007 年 11 月　大津 真

登場人物紹介

マリー先生
HP-UX のエキスパート。
HP-UX のことなら何でもお任せ！
HP-UX 普及のため世界を飛び回るスーパーウーマン。

四色君
これまで使ったことのあるパソコンは Windows のみ。ただし、コマンドプロンプトは未経験

タックス君
Windows、Linux、Mac の経験あり。ただし、操作は GUI でやることが多く、コマンドラインはほとんど使ったことがない。

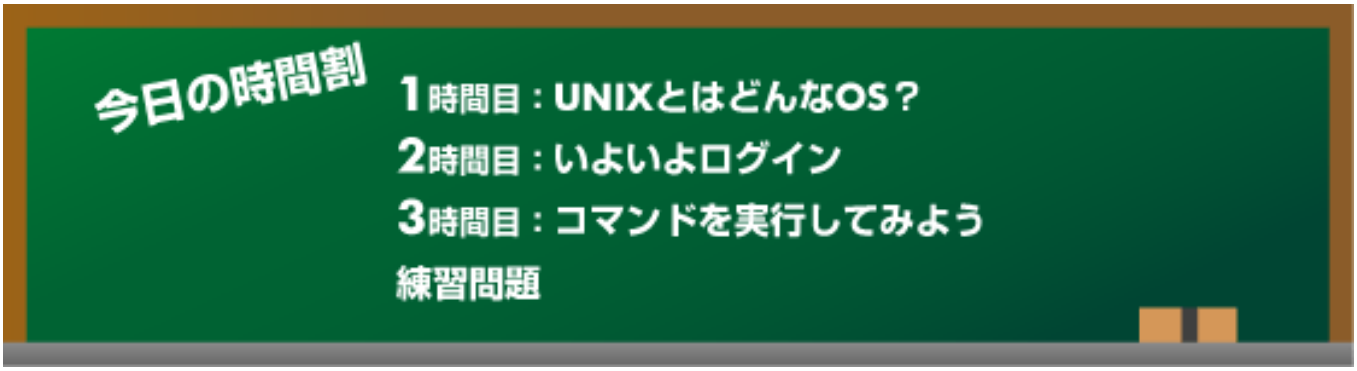
マリー先生：
さあこれからがんばって HP-UX のコマンドを学んでいきましょう。二人ともよろしくね。

四色君、タックス君：
よろしくお願いします。

タックス君：
さっそく質問なのですが、いまや UNIX でも GUI で操作できる管理ツールがありますよね。それでも CUI コマンドを覚える必要があるのですか？

マリー先生：
そう、たとえば HP-UX なら SMH（System Management Homepage）といった Web ベースの管理ツールがあるわよね。だけど、そのような GUI ツールでシステム管理のすべてをカバーしているわけではないの。より細かな操作はまだまだコマンドライン上で行う必要があるわけ。それと、セキュリティ上の配慮などから、GUI の管理ツールを利用できないように設定しているケースもあるの。

四色君：
なるほど、やはり、コマンドラインはシステム管理者になるためには避けて通ることのできない存在なのですね。



1 時間目：UNIX とはどんな OS？

HP-UX は、Hewlett Packard 社の商用 UNIX です。実際のコマンドライン操作の前に、まず、UNIX の概要について簡単に説明しましょう。

UNIX は、1970 年頃、当時 AT&T グループに属していたベル研究所で開発された、マルチユーザ、マルチタスク機能を特徴とする OS（オペレーティングシステム）です。ここでは、マルチユーザとは同時に複数のユーザが利用できることを、マルチタスクとは同時に複数の処理が行えることを言います。いまでこそパソコン用 OS には当たり前のような機能ですが、Windows や Mac OS などが登場するずっと以前から、UNIX にはそれらの機能が搭載されていたわけです。

その後、UNIX は System V 系と、BSD (Berkeley Software Distribution) 系の 2 つの系統に分かれて進化しました。その間、TCP/IP などネットワークに関するさまざまな技術も UNIX 上で開発されました。ただし、その過程で、メーカごとの機能拡張なども繰り返され、同じ UNIX といえども、互換性が乏しくなっていきました。その反省をふまえ、現在では、さまざまな企業が参加する中立な組織である The Open Group を中心に、UNIX の仕様である POSIX 規格に基づいた標準化が進められています。

四色君：
UNIX の得意分野ってあるんですか？

マリー先生：
UNIX サーバが登場した頃は、ネットワーク・サーバやファイルサーバとしての用途も多かったけれど、メーカーによって機能が拡張されるにしたがって、高速で沢山のトランザクションを処理できる様になって、現在では基幹システムで積極的に利用されているわ。

四色君：
Mac OS X も UNIX だと聞いたことがあるのですが？

マリー先生：
そうね。Mac OS X は、基本部分に Darwin という名前のオープンソースの UNIX を採用しているの。それ以前の Mac OS である OS 9 は、アプリケーションがクラッシュするとシステムを道連れにするということがよくあったの。それが UNIX ベースになってより安定した OS に生まれ変わったわ。

Linux の登場

当初 UNIX は、ワークステーションやミニコンといった個人で購入するには高価なコンピュータ上で動作していました。時代は変わり、CPU や周辺機器の急速な進化につれて、1990 年代にはパソコン上で UNIX を動作させることが現実味を帯びてきました。

そんな中注目されてきたのが、パソコン上で動作するオープンソースの UNIX ライクな OS です。その代表といえるのが Linux です。Linux は、1991 年、当時フィンランドのヘルシンキ大学の学生だったリーナス・トーバルズ（Linus Benedict Torvalds）氏が UNIX の機能をフリーで実現することを目指して、UNIX のソースを使用することなくゼロから作り上げた OS です。なお、リーナス氏が作成したのはシステムの中心部分である「カーネル」であり、開発ツールやユーティリティソフトなどは、オープンソースの普及を推進する GNU プロジェクト のプロダクトを利用しています。

Linux は、当初、マニアの間の OS というとらえ方をされていました。ところが、インターネットの普及とともに、開発に参加するボランティアも急速に増え機能や安定性が向上していくにつれ、いまではフリーの UNIX 系 OS としてすっかり認知された存在となりました。

なお、UNIX は現在 The Open Group の商標です。そのため、Linux は狭義の"UNIX"ではないとする人も少なくありません。UNIX ライクな OS、あるいは PC-UNIX という呼ばれ方をする場合があります。

タックス君：
一言で Linux と言ってもいろんな種類がありますよね。

マリー先生：
そう、狭義の Linux はシステムの中心部分である"カーネル"だけなの。そのため、普通はインストーラや、アプリケーションソフト、開発ツールなどをパッケージした"ディストリビューション"として配布されているの。ディストリビューションは完全にフリーのものからサポートの付いた商用のものまでいろんな種類があるの。たとえば、HP では、サーバ OS として、Red Hat Enterprise Linux や SUSE Linux Enterprise Server とったエンタープライズ向け Linux をサポートしているわ。

HP-UX の優位性は

HP-UX は System V の流れを汲む商用 UNIX として 1986 年に登場しました。当初は、モトローラ 68K ファミリ CPU で動作する、ワークステーション用の OS として普及していました。その後、RISC プロセッサ である PA-RISC 上に移植され、現在では、HP とインテル社とで共同開発した CPU であるインテル®Itanium®プロセッサをサポートしています。今では HP-UX はさまざまな分野で使用されていますが、特に企業の基幹業務システムや金融機関のオンラインシステムなど、高速性、信頼性が求められるいわゆる"ミッションクリティカル"な分野では圧倒的なシェアを誇っています。たとえば、日本国内の UNIX サーバ市場でも連続 6 年にわたり No.1 シェアを維持。四半期では、最新の 2007 年第 3 四半期で 18 四半期連続 No.1 シェアを獲得しています。

なお、本連載では、最新版の HP-UX 11i v3 を基にして説明します。

＜表 1：UNIX と HP-UX の歩み＞

2007 年	HP-UX 11i v3 信頼性・可用性・セキュリティが向上し、仮想化環境がより充実
2003 年	HP-UX 11i v2 Integrity サーバでの本格的なエンタープライズ向け環境を整える
2002 年	HP-UX 11i v1.6 Itanium 対応
2000 年	HP-UX 11i v1 ミッションクリティカル分野への本格的な対応
1997 年	HP-UX 11.0 64bit 化
1995 年	HP-UX 10.x　MC/ServiceGuard
1992 年	HP-UX 9.0
1991 年	HP-UX8.0　PA-RISC 版ワークステーション
1989 年	HP-UX 7.0
1989 年	UNIX SystemV R4(SVR4)
1986 年	HP-UX 1.0 PA-RISC 対応

1983 年	HP-UX が開発される
1983 年	UNIX SystemV
1979 年	UNIX/V 32bitCPU 版
1975 年	UNIX V6
1973 年	UNIX 4th Edition C 言語で書き直される
1971 年	UNIX 1st Edition
1968 年	UNIX の原型になるシステムが作られる

四色君：
ミッションクリティカル分野では、どうしてオープンソースの Linux ではなく、商用 UNIX である HP-UX が求められるのでしょう？

マリー先生：
HP ではそのキーワードとして「信頼性」「柔軟性」「管理性」の 3 つをあげているの。簡単に言えば、安定したシステムを、将来にわたって使い続けることができて、しかも管理がやりやすいということだけど、くわしくはこちらのリンクを見てね。
HP-UX

タックス君：
HP-UX は商用の UNIX ということですが、オープンソースのソフトウェアは使えないのですか？

マリー先生：
そんなことはないわ。例えば、Web サーバ「Apache」やプログラミング言語「Perl」といたオープンソースの世界では有名なソフトウェアが標準で用意されていますよ。

2 時間目：いよいよログイン

前置きはこれくらいにして、HP-UX の実際の操作方法の説明に移りましょう。

UNIX というと、昔はキャラクタ端末（文字しか表示することのできない端末）を接続して操作するという使い方が基本でした。1989 年頃からはワークステーションを中心とする UNIX では、X Window System と呼ばれるウィンドウ・システムが使用され始めました。なお、X Window System 自体は基本的な描画処理のみを担当し、ウィンドウの管理などのいわゆるルック&フ

ィールの部分はウインドウマネージャと呼ばれる独立したプログラムが行います。それらのプログラムに、ファイルマネージャなどのユーティリティを組み合わせた統一されたユーザインタフェースを持つ GUI のことをデスクトップ環境と呼びます。

現在 UNIX 系 OS で使用されるデスクトップ環境には GNOME や KDE などさまざまなものがありますが、HP-UX ではその安定性から CDE（Common Desktop Environment）と呼ばれるデスクトップ環境を採用しています。次に、CDE のログイン画面を示します。

図 1：CDE ログイン画面
ユーザ名とパスワードを入力して「了解」ボタンをクリックするとデスクトップ画面が表示されます。

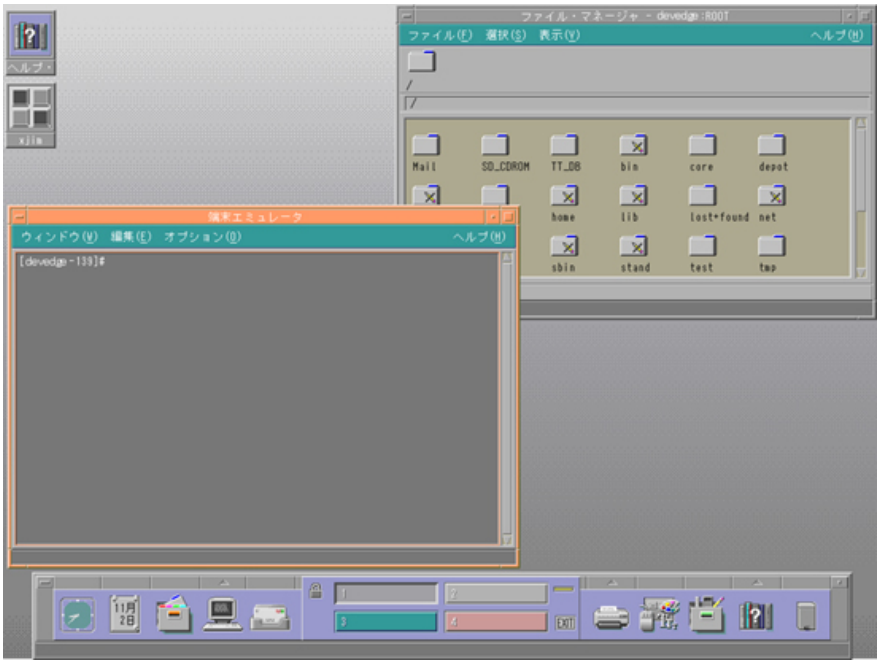


図 2：CDE デスクトップ画面
ログアウトするにはフロントパネルの「EXIT」ボタンをクリックします。

四色君：
UNIX には Windows の Administrator のような管理ユーザはあるのですか？

マリー先生：
UNIX ではあらゆる権限が許可された特別なユーザがいて、それを「スーパーユーザ」と呼ぶの。スーパーユーザはそれぞれのシステムに一人だけで、ユーザ名は「root」に決まっているわ。root でログインすると簡単にシステムを破壊したり大事な情報を盗むことができるので、スーパーユーザのパスワードは絶対外部に漏れないように注意してね。

ターミナルエミュレータを使ったログイン

さて、みなさんが実際にキーボードから UNIX のコマンドを入力するには「ターミナルエミュレータ」（以下、ターミナル）と呼ばれるソフトウェアを使用します。ターミナルとは前述のキャラクタ端末をデスクトップ上のウィンドウとして仮想的に再現したものです。CDE ではフロントパネルのアプリケーションのメニューの「端末エミュレータ」ボタンをクリックすると dtterm というターミナルのウィンドウが表示されます。ターミナルは、コンピュータシステムの中心部分であるカーネルへの入り口のようなものです。ターミナルの操作に慣れるにつれて、コンピュータへの理解がより深まっていきます。

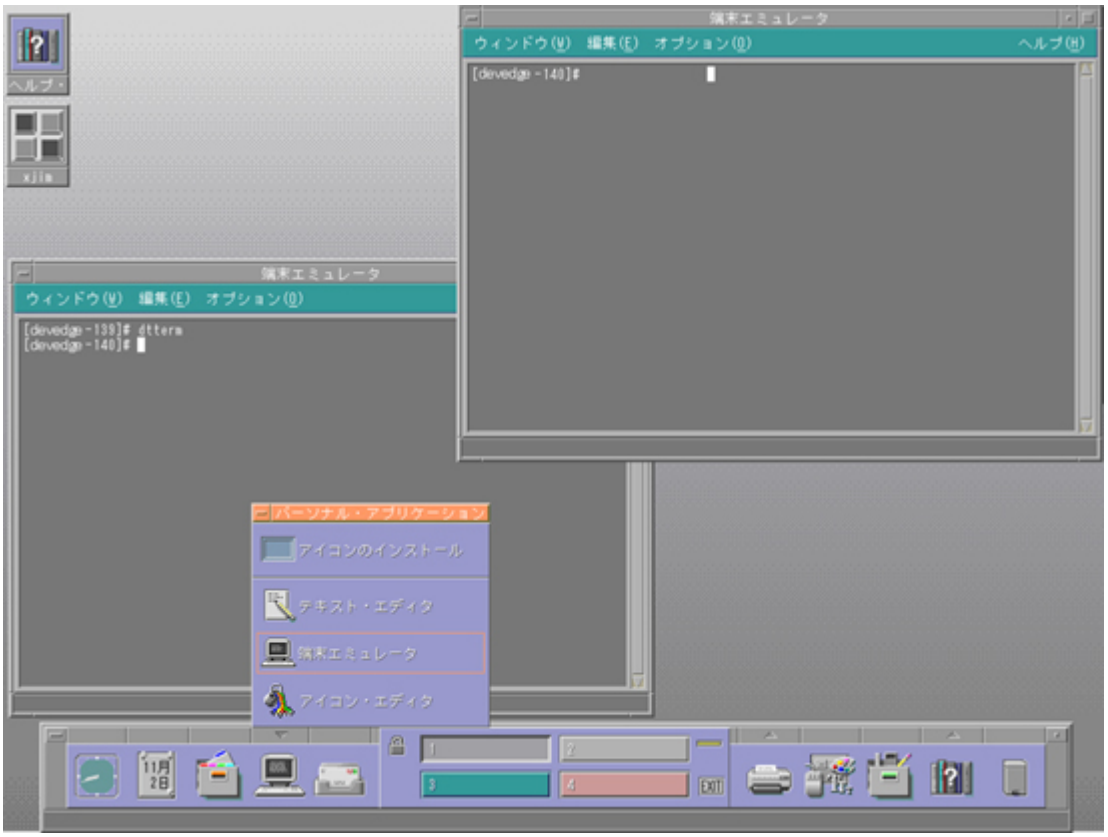


図 3：ターミナルのウィンドウ

四色君：
ちょっと今、自由に使える HP-UX がないんですけど……。

マリー先生：
そんな場合は、HP がインターネットで提供するテスト環境である「テストドライブ」を使うといいわね。ユーザ登録するとテスト用のアカウントがメールで送られてくるので、Telnet でリモートログインしていろいろなコマンドを試すことができるの。もちろん登録は無料よ。

マシンが離れた場所にあるときはリモートログインで

さて、たとえばマシンがサーバルームにあって直接手元で操作できない場合などは、ネットワークを介してコンピュータにログインする、いわゆる「リモートログイン」を使用することでコマンドを実行できます。リモートログインには、Telnet と SSH という 2 種類の方法があります。Windows ユーザの方は、UTF-8 TeraTerm Pro with TTSSH2 などを使用すると Telnet や SSH によるリモートログインが行えます。

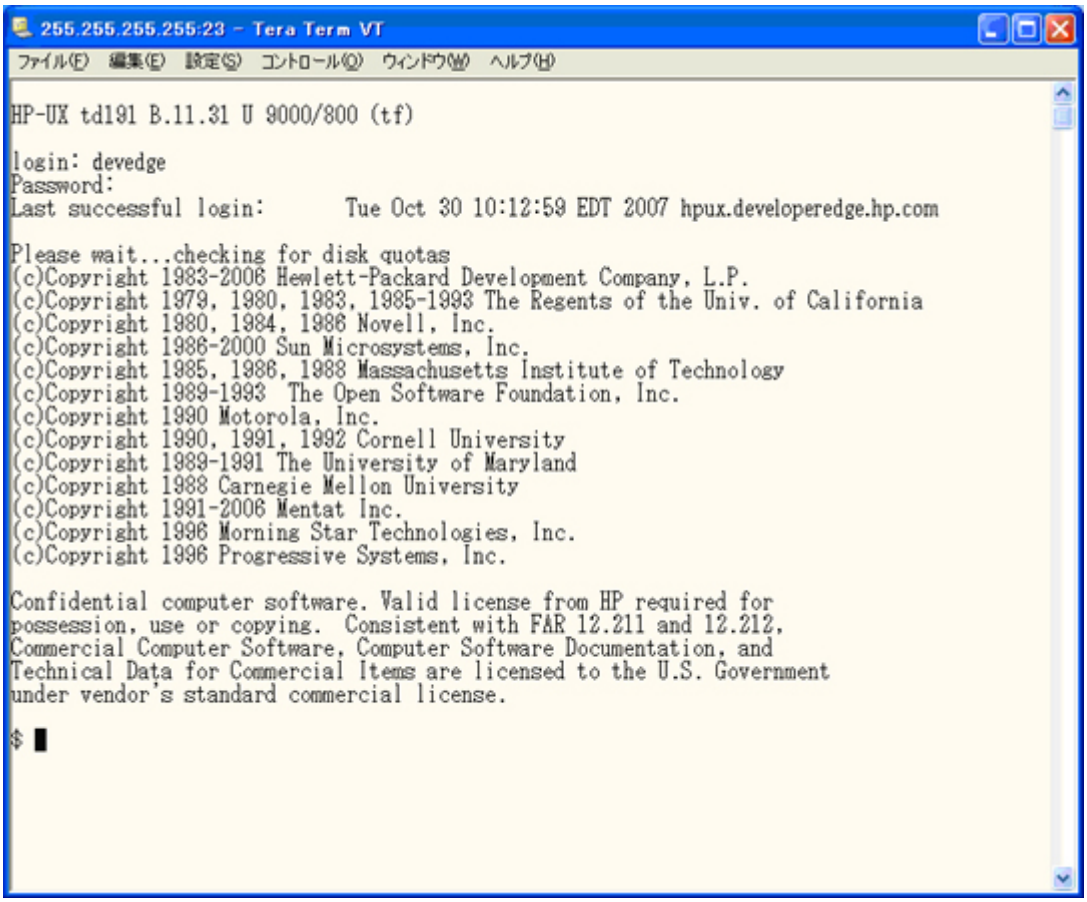


図 4：TeraTerm Pro で HP-UX にリモートログイン

Linux あるいは Mac OS X のユーザの方は、標準でターミナルが用意されています。また Telnet でログインする telnet コマンド、SSH によるリモートログインを行う ssh コマンドもインストールされているのでそれらを使えばよいでしょう。

タックス君：
Telnet と SSH ってどう違うの？

マリー先生：
Telnet は古くからあるプロトコルで、ネットワーク上をデータが暗号化されずに流れてしまうの。つまり、悪意のある第三者がネットワークを盗聴していると、ユーザ名やパスワード、そして通信の内容が漏れてしまう可能性があるわけ。そのため最近のシステムでは Telnet によるリモートログインが禁止されていることが多いわ。

それに対して SSH は、アカウント情報を含めて通信の内容はすべて暗号化されるの。さらに公開鍵暗号方式という、サーバに登録されているユーザ名とパスワードを使わないより安全なログインも可能なの。

休み時間：UTF-8 TeraTerm Pro with TTSSH2 について

UTF-8 TeraTerm Pro with TTSSH2（以下、TeraTerm Pro)は Windows で動作するフリーのターミナル・エミュレータソフトです。こちらのサイトからダウンロードできます。

UTF-8 TeraTerm Pro with TTSSH2

TeraTerm Pro でリモートログインを行うには、「接続」ダイアログボックスを開き「サービス」から「Telnet」もしくは「SSH」を選択します。「ホスト」で接続先のホスト名もしくは IP アドレスを指定し「OK」ボタンをクリックします。

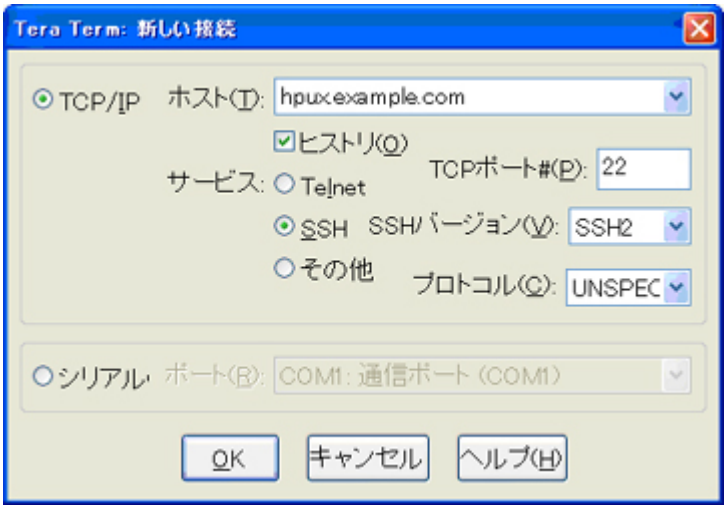


図 5：UTF-8 TeraTerm Pro with TTSSH2 の接続ダイアログボックス

3 時間目：コマンドを実行してみよう

図 4 のように、ターミナルにはカーソルの前に「$」が表示されていますが、これは現在システムがコマンドを受付可能な状態であることを示すもので、「コマンドプロンプト」（以下、プロンプト）と呼ばれています。コマンドはこのプロンプトに続いてタイプし、Enter キーを押すと実行されます。なお、初期設定では一般ユーザのプロンプトは「$」ですが、システムの管理者である root のプロンプトは「#」となります。

たとえば、現在の日付時刻を表示するコマンドに「date」があります。プロンプトに続いて「date<Enter>」とタイプして、日付時刻が表示されることを確認してください。

```
$ date 【Enter】
Thu Nov 22 13:00:00 EDT 2007
```

なお、現在のセッションを終了するには exit コマンドを実行します。

```
$ exit 【Enter】
終了します
```

四色君：
コマンドは大文字でも大丈夫ですか？

マリー先生：
Windows のコマンドプロンプトのコマンドと違って、UNIX のコマンドは大文字/小文字を区別するから注意してね。つまり「date」を「DATE」と入力することはできないの。

引数とオプション

コマンドに渡すなんらかの値のことを「引数」と呼びます。コマンドが、どのような引数を取るか、そして引数の数はコマンドによって異なります。コマンド名と引数はスペースで区切って指定します。

コマンド 引数 1 引数 2 ....

たとえば、与えられた引数をそのまま画面に表示するコマンドに「echo」があります。echo コマンドに「hello」と「world」の 2 つの引数を渡すには次のように実行します。

```
$ echo hello world 【Enter】
hello world
```

なお、スペースはそのままでは引数の区切りですが、全体をダブルクォーテーション「"」で囲むことによってスペースを引数の中に含めることができます。次の例を見てみましょう。

```
$ echo "hello world" 【Enter】
hello world
```

結果は前と同じですが、この例では「hello world」を全体をダブルクォーテーション「"」で囲っているため、echo コマンドの引数は 1 つだけです。

また、引数の順番や数も重要です。例えばカレンダーを表示するコマンドに cal がありますが、cal は引数を指定しないと今月のカレンダーを表示し、引数を 1 つだけ指定した場合には年を指定したものと見なされてその年のカレンダーが表示されます。また、引数を 2 つ指定した場合には最初の引数が月、2 番目が年とみなされます（逆ではないことに注意してください）。

```
$ cal 【Enter】 ←今月のカレンダーを表示
   November 2007
 S  M Tu  W Th  F  S
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

```
$ cal 2008 【Enter】 ←2008 年のカレンダーを表示
          Jan                    Feb                    Mar
 S  M Tu  W Th  F  S    S  M Tu  W Th  F  S    S  M Tu  W Th  F  S
       1  2  3  4  5                    1  2                    1
 6  7  8  9 10 11 12    3  4  5  6  7  8  9    2  3  4  5  6  7  8
13 14 15 16 17 18 19   10 11 12 13 14 15 16    9 10 11 12 13 14 15
20 21 22 23 24 25 26   17 18 19 20 21 22 23   16 17 18 19 20 21 22
27 28 29 30 31         24 25 26 27 28 29      23 24 25 26 27 28 29
                                              30 31
          Apr                    May                    Jun
 S  M Tu  W Th  F  S    S  M Tu  W Th  F  S    S  M Tu  W Th  F  S
       1  2  3  4  5                 1  2  3    1  2  3  4  5  6  7
 6  7  8  9 10 11 12    4  5  6  7  8  9 10    8  9 10 11 12 13 14
13 14 15 16 17 18 19   11 12 13 14 15 16 17   15 16 17 18 19 20 21
20 21 22 23 24 25 26   18 19 20 21 22 23 24   22 23 24 25 26 27 28
27 28 29 30            25 26 27 28 29 30 31   29 30
          Jul                    Aug                    Sep
 S  M Tu  W Th  F  S    S  M Tu  W Th  F  S    S  M Tu  W Th  F  S
       1  2  3  4  5                    1  2       1  2  3  4  5  6
 6  7  8  9 10 11 12    3  4  5  6  7  8  9    7  8  9 10 11 12 13
13 14 15 16 17 18 19   10 11 12 13 14 15 16   14 15 16 17 18 19 20
20 21 22 23 24 25 26   17 18 19 20 21 22 23   21 22 23 24 25 26 27
27 28 29 30 31         24 25 26 27 28 29 30   28 29 30
                       31
          Oct                    Nov                    Dec
 S  M Tu  W Th  F  S    S  M Tu  W Th  F  S    S  M Tu  W Th  F  S
          1  2  3  4                    1       1  2  3  4  5  6
 5  6  7  8  9 10 11    2  3  4  5  6  7  8    7  8  9 10 11 12 13
12 13 14 15 16 17 18    9 10 11 12 13 14 15   14 15 16 17 18 19 20
19 20 21 22 23 24 25   16 17 18 19 20 21 22   21 22 23 24 25 26 27
26 27 28 29 30 31      23 24 25 26 27 28 29   28 29 30 31
```

```
$ cal 1 2008 【Enter】 ←2008 年 1 月のカレンダーを表示
   January 2008
```

```
 S  M Tu  W Th  F  S
        1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

なお、引数の中で、特にコマンドに対する指令のようなものを「オプション」と呼びます。オプションの指定方法はコマンドによっていくつかのバリエーションがありますが、UNIX に伝統的なオプションの基本形は、ハイフン「-」に続いてアルファベット 1 文字を指定するというものです。

たとえば、前述の date コマンドには、日付時刻を UTC（世界協定時）で表示する「-u」オプションが用意されています。

```
$ date -u 【Enter】
Thu Nov 22 17:00:00 UTC 2007
```

コマンドによってはオプションを複数指定できます。その場合、個別に「-オプション」の形式で指定するか、あるいは h ハイフンの後にオプションを並べて記述します。たとえば、コンピュータの情報を表示するコマンドに uname があります。uname コマンドでは、「-s」オプションはオペレーティングシステムの名前を、「-r」オプションはリリース番号を表示します。これらの 2 つのオプションを指定して uname コマンドを実行するには次の 2 通りの方法があります。

```
$ uname -s -r 【Enter】
HP-UX B.11.31
```

```
$ uname -sr 【Enter】
HP-UX B.11.31
```

四色君：
コマンドと引数、あるいは引数の間の区切り文字にはカンマ「,」は使えるのですか？

マリー先生：
使えません。カンマを記述してもそれは引数中の 1 文字と見なされるの。

四色君：
コマンドを 2 行に分けて入力することはできるのですか？

マリー先生：
行の最後に「\」をタイプすればできるわ。次の例を見てね。2 行目以降のプロンプトは「>」になることに注目してね。

```
$ echo hello \　【Enter】←「\」はコマンドが次の行に続くことを表す
> wolrd　【Enter】　←2 行目以降はプロンプトは「>」になる
hello wolrd
```

タックス君：
HP-UX のコマンドは Linux のコマンドと同じなのですか？

マリー先生：
基本的なコマンドに関しては同じものが多いわ。今回紹介したコマンドはどれも Linux にも用意されているものばかりよ。だけど、同じコマンド名でもオプションや引数の使い方が異なるものも少なくないので注意してね。

練習問題

第 1 問：HP-UX で標準で使用されるデスクトップ環境は？

a)　GNOME
b)　KDE
c)　CDE

c)　CDE
CDE は「Common Desktop Environment」の略で、その名が示すように UNIX における共通のインタフェースを提供することを目的に開発されたデスクトップ環境。HP-UX や Solaris などで採用されている。GNOME および KDE はオープンソースのデスクトップ環境。

第 2 問：データが暗号化されたリモートログインが可能なプロトコルは？

a)　SSH
b)　Telnet
c)　FTP
d)　HTTP

a)　SSH
SSH は「Secure Shell」の略で、データが暗号化されたリモートログインやファイル転送を行うプロトコル。Telnet はリモートログインを行うプロトコル。FTP はファイル転送を行うプロトコル。HTTP は Web サーバと Web ブラウザの通信に使用されるプロトコル。

第 3 問：現在の日付時刻を表示するコマンドは？

a)　DATE
b)　date
c)　Today
d)　today

b)　date
UNIX ではコマンドの大文字/小文字を区別するので「date」を「DATE」として実行することはできない。

第 4 問：2008 年 7 月のカレンダーを表示するコマンドは？

a)　cal 2008 7
b)　cal 2008/7
c)　cal 7 2008
d)　cal "7 2008"

c)　cal 7 2008
cal コマンドは最初の引数に「月」を、2 番目の引数に「年」を指定する。なお、引数の区切り文字にはスペースを使用する。

UNIX の教科書「基礎編」

第 2 日目：ディレクトリやファイルを操作してみる

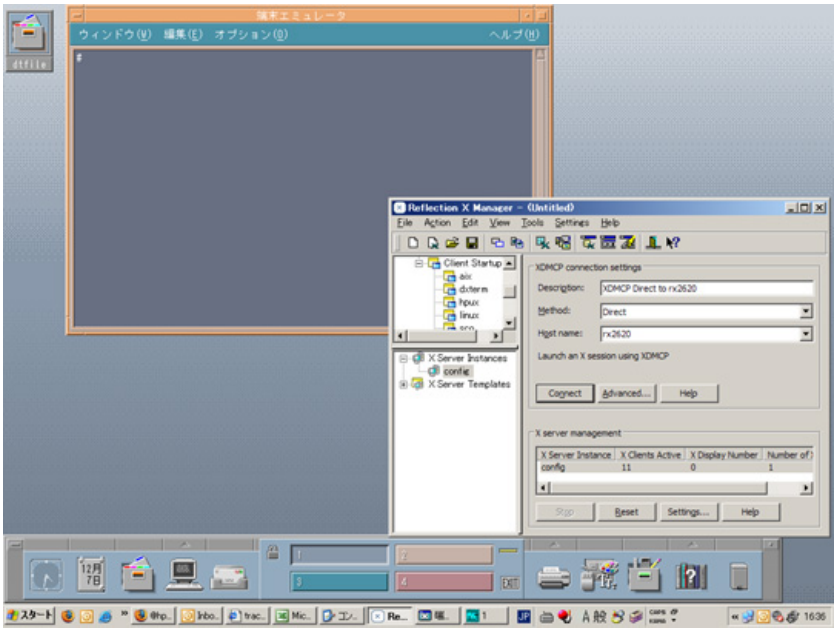
2007 年 12 月　大津 真

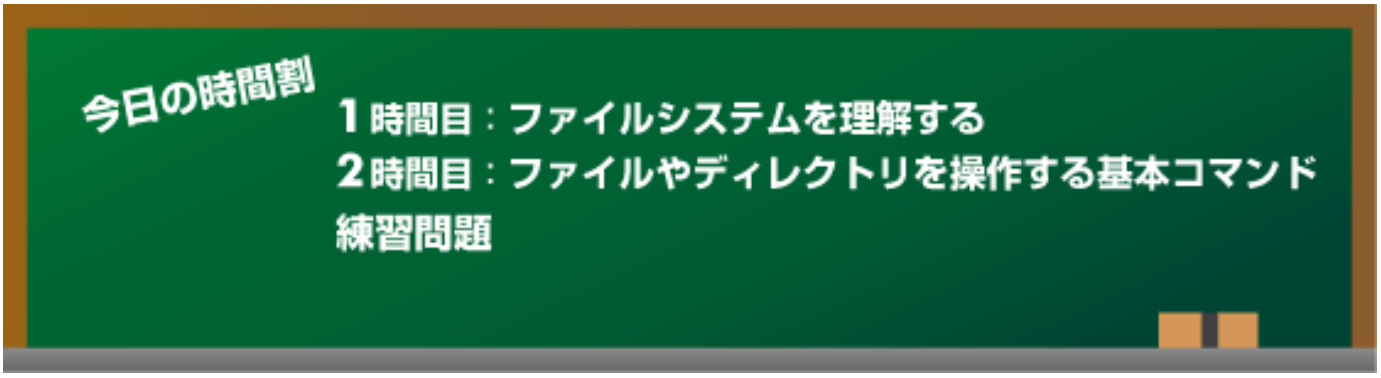
前回は HP-UX の概要とターミナルの基本操作について説明しましたが、コマンドラインの雰囲気がなんとなくつかめてきたでしょうか？第 2 回目では、まず HP-UX のファイルシステムの概要について説明します。その後で、ディレクトリやファイルを操作する基本的なコマンドをいくつか紹介していくことにしましょう。

マリー先生：
さあみなさん HP-UX の操作に少しは慣れてきたかしら？さて今回も、コマンドラインの基本操作について学んでいきましょう。ところで、前回の説明に関してなにか質問ありますか？

四色君：
前回教えてもらった Windows の TeraTerm から HP-UX にリモートログインして、いろいろ操作しています。Windows から CDE にログインして、デスクトップ環境をリモートコントロールできたらより便利だなと思うのですが……。

マリー先生：
それには「PC X サーバ」などと呼ばれる種類のソフトを使うと可能よ。代表的なのは「ASTEC-X」と、「Reflection」かしら。





1 時間目：ファイルシステムを理解する

UNIX では、すべてのファイルを「ツリー構造」などと呼ばれる階層構造で管理しています。階層構造型ファイルステムは、今では Windows や Mac といった一般的な OS でもすっかりおなじみの仕組みですが、UNIX はそのはしりであるといえます。次に、HP-UX のファイルシステムの概略図を示します。

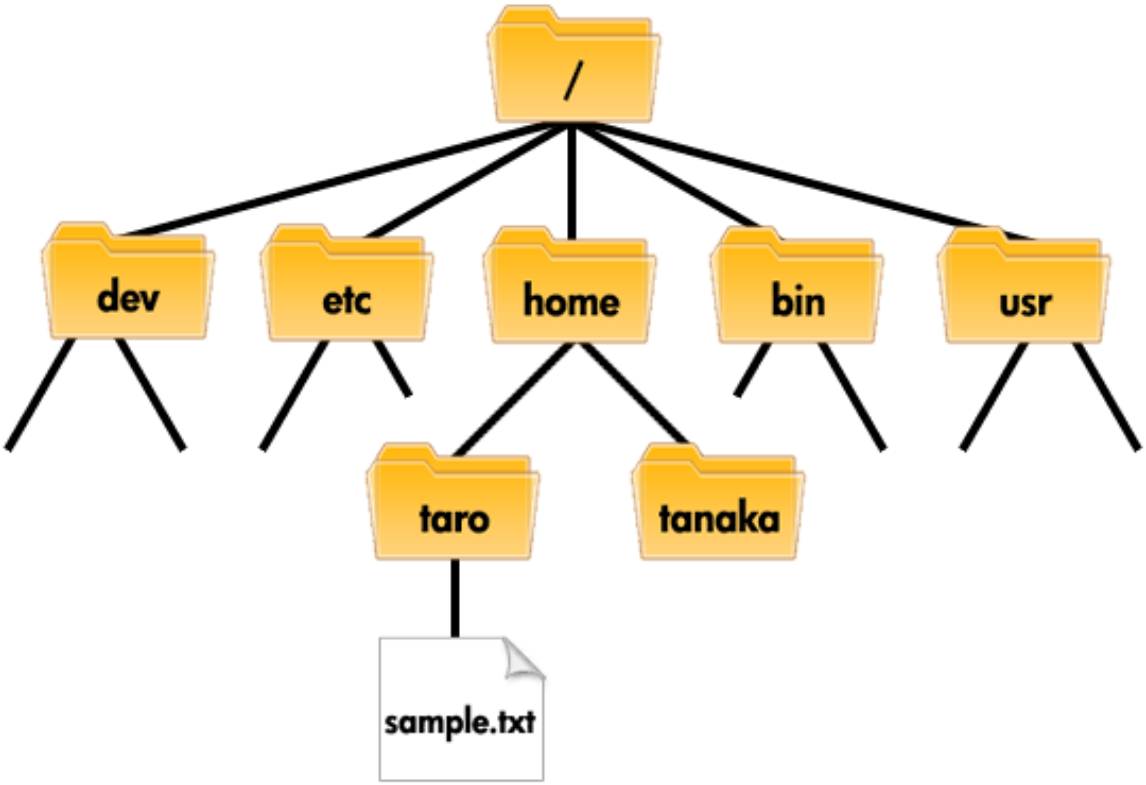


図 1：HP-UX のファイルシステム概略図

ツリー構造の“ツリー”とは日本語では木の意味ですが、ご覧のように、見た目が木を逆さまにしたようなイメージになるため、そのように呼ばれるわけです。なお、ツリー構造の頂点を表す記号は「/（スラッシュ）」です。ちょうど木の根っこの部分に相当するため、「ルート（“根っこ”という意味）」と呼ばれます。また、枝の分岐点に当たる部分を「ディレクトリ」と呼び、それぞれの葉の部分が個々の「ファイル」となるわけです。

四色君：
ディレクトリって、Windows でいうところのフォルダと同じものですか？

マリー先生：
同じよ。GUI 環境ではフォルダ、CUI 環境ではディレクトリという用語が使われることが多いの。

四色君：
そういえばスーパーユーザのユーザ名も「root」でしたね。

マリー先生：
そう、UNIX では、root（ルート）は、すべての権限が与えられた特別なユーザであるスーパーユーザのユーザ名であると同時に、ファイルシステムの頂点も表すの。

タックス君：
ディレクトリ構造は他の UNIX や Linux でも同じですか？

マリー先生：
基本的な部分に関してはね。たとえば、主な設定ファイルは/etc ディレクトリの下にまとめるとか、デバイスファイルは/dev ディレクトリに置くといった点は一緒よ。ただし、こまかな部分はシステムによってまちまちなの。

絶対パスと相対パス

コマンドを使用して、ファイルやディレクトリに対して何らかの処理を行う場合、引数では対象となるファイルやディレクトリまでの道筋である「パス」を指定する必要があります。このとき、パスの指定方法は「絶対パス」と「相対パス」に大別されます。絶対パスはファイルシステムの起点であるルート「/」から指定する方法です。たとえばユーザごとに用意されている自由に使用可能な専用のディレクトリのことを「ホームディレクトリ」と呼びますが、このホームディレクトリはルート「/」の下の「home」というディレクトリの下の「ユーザ名」ディレクトリに設定されています。たとえば、ユーザ「o2」のホームディレクトリの下の「sample.txt」というファイルを絶対パスで指定するには、次のようになります。

```
/home/o2/sample.txt
```

Web サイトのアドレスと同じく、ディレクトリ間の区切り文字には「/」を使用します。つまり「/」はルートを表すだけでなく、区切りにも使用されるわけです。このように絶対パスではルート「/」から順にディレクトリをたどっていくため、目的のファイルやディレクトリを一意に指定できます。

それに対して相対パスのほうは、その名が示すとおり相対的なパスの指定です。現在自分がいるディレクトリのことを「カレントディレクトリ（あるいはワークディレクトリ）」と呼びますが、カレントディレクトリを起点にして相対的にファイルやディレクトリを指定する方法です。

たとえば、ログインした時点ではホームディレクトリがカレントディレクトリになりますが、その下の sample.txt というファイルは次のように指定します。

```
sample.txt
```

また、カレントディレクトリの下の「Documents」ディレクトリの下の「ReadMe」というファイルは、次のように指定できます。

```
Document/ReadMe
```

なお、現在のカレントディレクトリは pwd コマンドで確認できます。

```
$ pwd 【Enter】
/home/o2
```

四色君：
Windows ではディレクトリの区切りは「\」ですけど、UNIX でも使えますか？

マリー先生：
使えません。前回の 3 時間目でも紹介したけど、「\」は行末で使うとコマンドラインが次の行に継続する意味になるの。また、「エスケープシーケンス」と呼ばれる特殊文字を表すのにも使われるわ。たとえば「\n」はキャリッジリターン（改行）、「\t」はタブね。それと、UNIX ではディレクトリもある種のファイルとして扱うの。

四色君：
ディレクトリがファイルってどういうことですか？

マリー先生：
ディレクトリは、そのディレクトリの下の階層のファイルやディレクトリに関する情報が格納されているファイルのようなイメージで考えるといいわ。これ以降の説明で、単にファイルといった場合にはディレクトリも含んでいることがほとんどなので注意してね。それだけでなく、ディスクやプリンタなどの周辺機器も「デバイスファイル」という特別なファイルとして扱えるの。つまり UNIX ではなんでもファイルというわけ。

タックス君：
ファイルシステムのフォーマットとしては Windows では「NTFS」、Mac は「HFS+」、Linux では「ext3」などが一般的に使われますけど、HP-UX の標準フォーマットはなんですか？

マリー先生：
現在 HP-UX の標準ファイルシステムはジャーナリング機能をサポートした「JFS（Journal File System）」が採用されているの。

四色君：
ジャーナリング機能って？

マリー先生：
簡単に言えば、ファイルシステムの更新情報を逐次記録しておいて、システムがクラッシュした場合の復旧を安全にそして迅速にできる仕組みよ。最近では NTFS や HFS+といったファイルシステムでもサポートされているの。なお、JFS は米 VERITAS Software 社（2004 年に米 Symantec 社と合併、現在の社名は Symantec）が開発したファイルシステムであるところから、システム情報などでは「vsfs (VERITAS File System)」と表示されるわ。

テキストファイルのファイルの中身を表示する

UNIX にはテキストファイルの中身を表示するコマンドがいくつか用意されていますが、その中でもっとも基本的なのが「cat」コマンドです。引数には目的のファイルのパスを指定します。たとえば「/etc/hosts」というファイルには IP アドレスとホスト名の対応が記述されていますが、その中身を表示するには次のようにします。

```
$ cat /etc/hosts 【Enter】
## Configured using SAM by root on Thu Nov 22 17:19:53 2007
## Configured using SAM by root on Thu Nov 22 17:19:53 2007
# @(#)B.11.31_LRhosts $Revision: 1.9.214.1 $ $Date: 96/10/08 13:20:01 $
#
# The form for each entry is:
# <internet address>    <official hostname> <aliases>
#
# For example:
# 192.1.2.34    hpfcrm  loghost
#
# See the hosts(4) manual page for more information.
# Note: The entries cannot be preceded by a space.
```

```
#      The format described in this file is the correct format.
#      The original Berkeley manual page contains an error in
#      the format description.
#
127.0.0.1       localhost       loopback
192.168.1.13    devedge
```

なお、cat コマンドに「-n」オプションを指定して実行すると、各行の左に行番号を表示します。

```
$ cat -n /etc/hosts 【Enter】
    1  ## Configured using SAM by root on Thu Nov 22 17:19:53 2007
    2  ## Configured using SAM by root on Sat Dec  1 21:50:24 2007
    3  # @(#)B.11.31_LRhosts $Revision: 1.9.214.1 $ $Date: 96/10/08 13:20:01 $
    4  #
    5  # The form for each entry is:
    6  # <internet address>    <official hostname> <aliases>
    7  #
    8  # For example:
    9  # 192.1.2.34    hpfcrm  loghost
   10  #
   11  # See the hosts(4) manual page for more information.
   12  # Note: The entries cannot be preceded by a space.
   13  #      The format described in this file is the correct format.
   14  #      The original Berkeley manual page contains an error in
   15  #      the format description.
   16  #
   17
   18  127.0.0.1       localhost       loopback
   19  192.168.1.13    devedge
   20  192.168.1.102  imac
   21  192.168.1.240 susev
```

長いファイルを表示するのに便利なページャ

cat コマンドでは単にファイルの中身が画面にはき出されているだけなので、長いテキストファイルは画面がスクロールしてしまって最初のほうが見られません。長いファイルを閲覧するのに便利なのが、ファイルを 1 画面ずつ表示してくれる「ページャ」と呼ばれる種類のプログラムです。HP-UX ではページャとして「more」コマンドが用意されています。次に、more コマンドの実行画面を示します。スペースキーで次の画面に、「b」をタイプすると 1 つ前の画面に戻ります。終了するには「q」をタイプします。

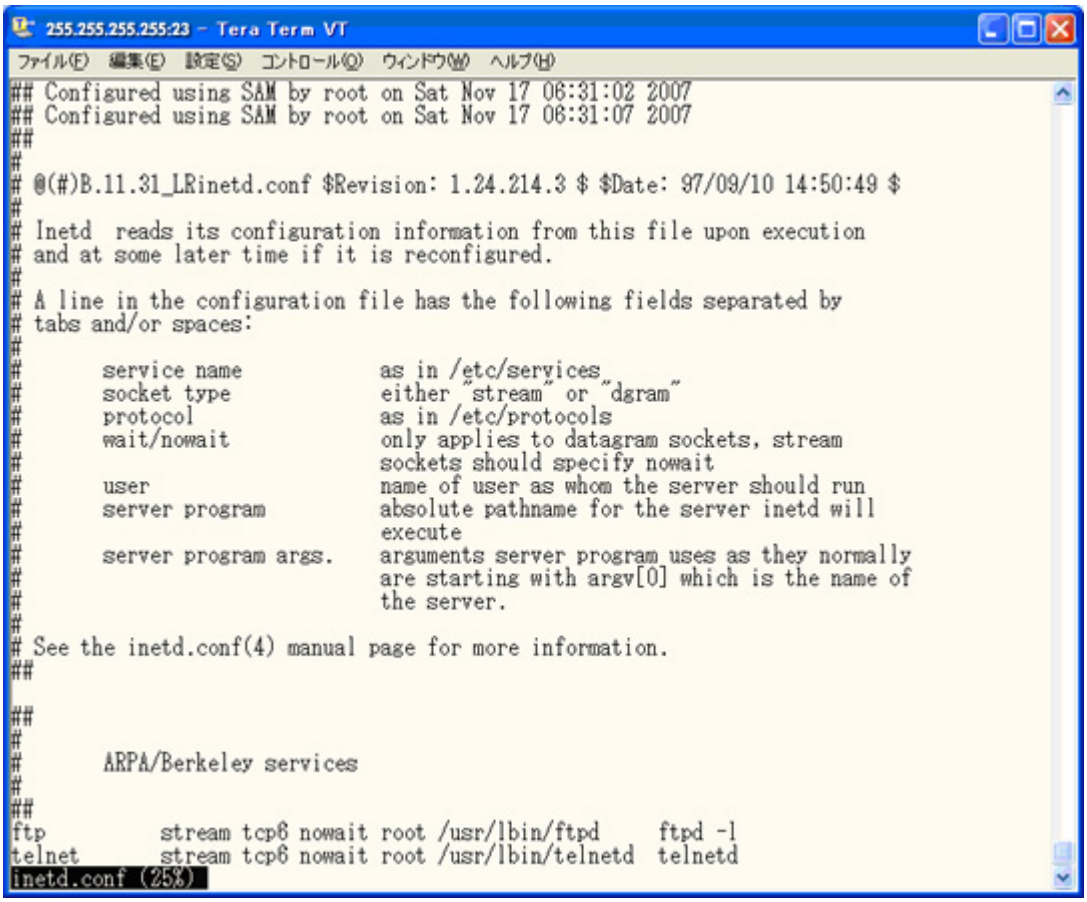


図 2：more で「/etc/inetd.conf」を表示

タックス君：
文字列を検索することはできるのですか？

マリー先生：
まず「/」をタイプして、続けてもキーワードを入力して Enter を押せば、ファイルの後ろに向かって検索できるわ。「?」の後にキーワードを指定すれば、ファイルの前の方向に向かって検索できるの。同じ文字列を同じ方向に向かって検索するには「n」をタイプしてね。

休み時間：UNIX のオンラインマニュアル

UNIX のほぼすべてのコマンドには、man 形式のオンラインマニュアルが用意されています。オンラインマニュアルは「man コマンド名」を実行すると表示されます。なお、オンラインマニュアルの表示は more コマンドが担当します。したがって、スペースキーで次のページに、前のページに戻るには「b」をタイプします。終了は「q」です。

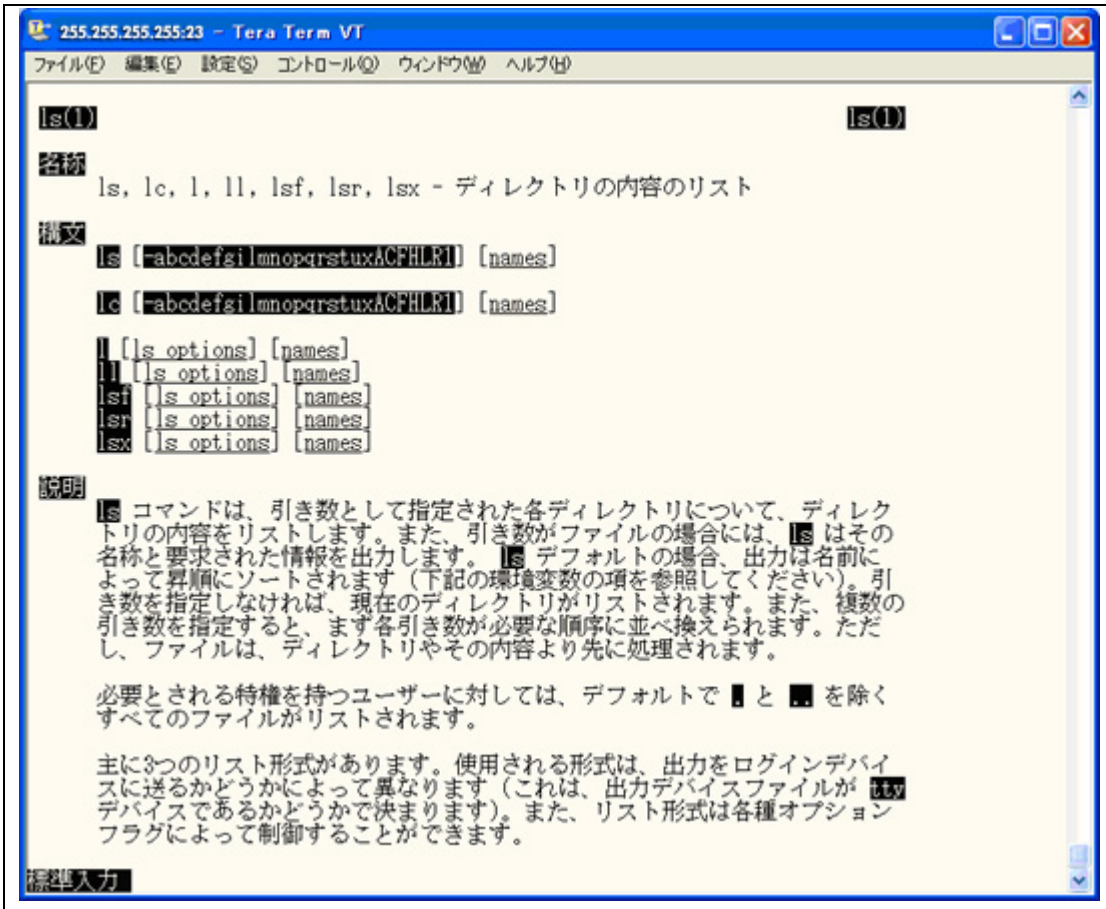


図 3：man コマンドでオンラインマニュアルを参照

HP-UX の日本語オンラインマニュアルは、シフト JIS と日本語 EUC の 2 種類の文字コードで用意されています。なお、マニュアルが英語で表示される場合は、LANG 環境変数という言語環境を設定する変数を設定したうえで man コマンドを実行する必要があります。環境変数について学ぶのはまだまだ先のことになりますので、ひとまず理解は措いて次のようにタイプしてください。

```
$ export LANG=ja_JP.SJIS        ←シフト JIS に設定する場合
$ export LANG=ja_JP.eucJP       ←日本語 EUC に設定する場合
```

2 時間目：ファイルやディレクトリを操作する基本コマンド

HP-UPX にけるツリー構造の概要とパスの指定方法が理解できたところで、早速、ディレクトリを操作するコマンドをいくつか紹介していきましょう。まず、指定したディレクトリの下のファイルを一覧表示する基本コマンドに、「ls」コマンドがあります。ls コマンドの引数にはディレクトリのパスを指定します。たとえば、「/usr」ディレクトリのファイル一覧を表示するには次のようにします。

```
$ ls /usr 【Enter】
TT_DB       contrib     keysh       man         preserve    spool
adm         dt          lbin        newconfig   pub         tmp
bin         etc         lib         news        sam         tsm
ccs         examples    local       obam        sbin        vue
conf        include     lost+found  old         share
```

もちろん、パスは相対パスで指定してもかまいません。カレントディレクトリの下の「Documents」ディレクトリのファイル一覧を表示するには次のようにします。

```
$ ls Documents 【Enter】
2007      Readme.txt
```

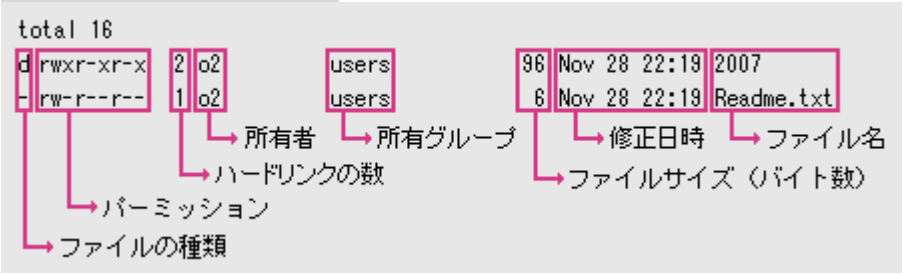
ls コマンドを引数なしで実行すると、カレントディレクトリのファイル一覧が表示されます。

```
$ ls 【Enter】
Documents   Readme.txt  Samples
```

ファイルの詳細情報を表示する

修正日時やアクセス権限といった情報を含めて一覧を表示するには、「-l」オプションを指定して実行します。たとえばカレントディレクトリの下の Documents ディレクトリの下のファイルに関する詳細情報を表示するには次のようにします。

```
$ ls -l Documents 【Enter】
```



最初の 1 文字はファイルの種類を表します。「d」はディレクトリを、「-」は通常のファイルを表します。その後ろの「drwxr-xr-x 」といった記号は「パーミッション」と呼ばれるアクセス権限です（パーミッションについては後ほど説明します）。その後の数字はハードリンクの数です。続いて所有者、所有グループ、サイズ、修正日時、名前となります。なお、引数にディレクトリではなくファイルのパスを指定した場合には、そのファイルに関する詳細情報が表示されます。

```
$ ls -l Documents/Readme.txt 【Enter】
-rw-r--r--  1 o2      users         6 Nov 28 22:19 Documents/Readme.txt
```

もしくは

```
$ ll Documents/Readme.txt 【Enter】
-rw-r--r--  1 o2      users         6 Nov 28 22:19 Documents/Readme.txt
```

タックス君：
ハードリンクってなんですか？

マリー先生：
異なるパスで同じファイルをアクセスできるようにした仕組みを"リンク"というのは知っているわよね。UNIX のリンクはハードリンクとシンボリックリンクの 2 種類があるの。それについてはまた別の機会に説明するわ。

四色君：
「ls -l ディレクトリのパス」を実行すると、そのディレクトリの下のファイルに関する情報が表示されますよね。それでは、そのディレクトリ自体の詳細情報を見るにはどうすればよいのですか？

マリー先生：
「-l」に加えて「-d」を指定すればいいわ。この場合「-l -d」としても「-ld」としてもどちらでも OK よ。

```
$ ls -l -d Documents 【Enter】
drwxr-xr-x  3 o2      users       96 Nov 28 22:19 Documents
```

もしくは

```
$ ls -ld Documents 【Enter】
drwxr-xr-x  3 o2      users       96 Nov 28 22:19 Documents
```

隠しファイルを表示する

UNIX ではファイル名の先頭がピリオド「.」で始まるファイルは「隠しファイル」あるいは「ドットファイル」と呼ばれ、デフォルトでは表示されないようになっています。それらを含めてすべてのファイルを表示さするには「-a」オプションを含めて実行します。

```
$ ls -a 【Enter】
.                .bash_history     .firefox-license  .xjim_defaults    Samples
..               .cshrc            .fonts.cache-1    .xjim_keybind     samples
.ICEauthority    .dt               .login            .xjim_learn
.TTauthority     .dtprofile        .mozilla          .xjim_onlud
.Xauthority      .dtprofile.org    .profile          Documents
.atokx           .exrc             .sh_history       Readme.txt
```

ディレクトリを作成する

新たにディレクトリを作成するには「mkdir」コマンドを使用します。たとえばカレントディレクトリの下に「samples」ディレクトリを作成するには、次のようにします。

```
$ mkdir samples
```

なお、mkdir で深いディレクトリを作成するには、途中のディレクトリが存在している必要があります。たとえば、カレントディレクトリの下に「pictures/2007/nov」ディレクトリを作成するためには、その時点で「pictures/2007」までが存在している必要があります。ここで「-p」オプションを指定して実行すると、途中のディレクトリが存在していない場合には作成してくれます。

```
$ mkdir -p pictures/2007/nov
```

四色君：
ディレクトリはどんな場所にも作成できるのですか？

マリー先生：
ls コマンドのところでちょっと触れたけど、UNIX にはパーミッションと呼ばれるアクセス制御の仕組みが備わっているの。パーミッションでは、ユーザを「ファイルの所有者」「ファイルの所有グループ」「その他」という単位に分類して、ファイルひとつひとつに対してユーザ単位ごとに「読み込み」「書き変え」「実行」という権限を設定できるわ。たとえば example.txt というファイルに対して、ファイルの所有者には「読み込み」「書き換え」を許可、ファイルの所有グループには「読み込み」だけを許可、その他にはすべて不許可、といったふうに設定できるということね。

さて、ディレクトリはその下のファイルの一覧が格納されたファイルのようなイメージと 1 時間目に説明したわよね。つまり、ディレクトリを作成するためには、その上のディレクトリの「書き換え権限」が許可されている必要があるわけ。初期状態では、一般ユーザが自由にファイルやディレクトリを作成したり削除したりできるのは、ホームディレクトリの下だけと考えたほうがいいわね。

四色君：
アクセス権限の「実行」とはなんですか？

マリー先生：
UNIX のコマンドのほとんどは、/usr/bin などのディレクトリに保存されているファイルなの。ファイルの「実行権限」とは、そのファイルをコマンドとして実行できるかどうかを表しているわけ。

ディレクトリを移動する

カレントディレクトリを移動するには「cd」コマンドを実行します。たとえば、カレントディレクトの下の samples ディレクトリに移動するには次のように実行します。

```
$ cd samples
$ pwd 【Enter】
/home/o2/samples
```

なお、cd コマンドを引数なしで実行するとホームディレクトリに移動します。

練習問題

第 1 問：ユーザ「taro」のホームディレクトリを絶対パスで表記したものはどれか？

a)　home/taro
b)　/home/taro
c)　/taro
d)　/users/taro

b)　/home/taro
絶対パスはパスを「/」（ルート）から指定する方法。HP-UX では、ホームディレクトは初期状態で「/home/ユーザ名」に設定されている。

第 2 問：隠しファイルを含めてカレントディレクトリの一覧を表示するコマンドはどれか？

a)　ls -a
b)　ls current
c)　la
d)　ls -t

a)　ls -a
「-a」オプションをつけて ls コマンドを実行するとすべてのファイルが表示される。

第 3 問：カレントディレクトリの下に「samples/2007」を作成するコマンドはどれか？　このとき samples ディレクトリは存在していないものとする。

a)　mkdir samples/2007
b)　mkdir 2007
c)　mkdir -p samples
d)　mkdir -p samples/2007

d)　mkdir -p samples/2007

途中のディレクトリを含めて深いディレクトリを作成するには、「-p」オプションを指定して mkdir コマンドを実行する。

第 4 問：次の中でページャとして使用されるコマンドはどれか？

a)　cat
b)　page
c)　more
d)　ls

c)　more
ページャとはテキストファイルを一画面ずつ表示するコマンド。

UNIX の教科書「基礎編」

第 3 日目：シェルの基本を知る

2008 年 1 月　大津 真

UNIX では、ユーザが入力したコマンドの解釈/実行や実行中のプログラムの制御は、ユーザとシステムの仲介役である「シェル」というプログラムが担当します。UNIX のシェルは、単なるユーザインターフェースとして存在するだけではなく、プログラミング言語としての機能もあります。今回はシェル入門ということで、シェルの概要と基本的な操作を説明しましょう。

マリー先生：
さあみなさん、今回はコマンドラインの心臓部分といえる「シェル」の基本操作について学んでいきましょう。その前に、前回の説明に関してなにか質問ありますか？

四色君：
ls コマンドを実行するとファイルもディレクトリも同じように表示されて、ファイルの種類がわかりにくいんですけど……。

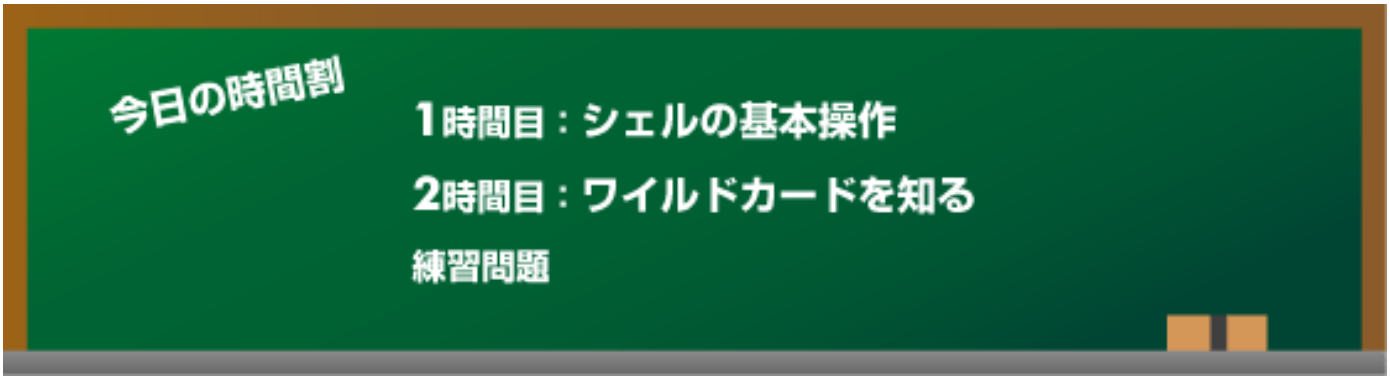
マリー先生：
ls コマンドに「-F」オプションを指定すると、ディレクトリの場合には最後に「/ (スラッシュ)」、シンボリックリンクの場合には「@」といった記号を表示してくれるので、ファイルの種類が一目でわかるわよ。

```
$ ls -F 【Enter】
Documents/  Readme.txt  ok.txt@    profile.txt  sample.png
Pictures/   bkup/       original/  public_html/ samples/
```

ls にはこれ以外にもいろいろと便利なオプションがあるから、「man ls」を実行してマニュアルを見ておいてね。たとえば、デフォルトでは一覧はファイル名の順に表示されるけど、「-t」オプションを指定すると、修正日時の順に並び替えて表示してくれるわ。

また、ls には、それぞれ以下の省略形式のコマンドが用意されています。

```
lsf  =  ls -F
ll   =  ls -l
lsx  =  ls -x
lsr  =  ls -R
```



1 時間目：シェルの基本操作

ユーザがターミナルで打ち込んだコマンドは、「シェル」というプログラムが解釈して、システムの中心部分であるカーネルに伝えます。“シェル（shell）”とは日本語では“貝殻”といった意味になります。カーネルを貝殻のように包み込んでユーザとの橋渡しをする存在といったイメージで考えてください。

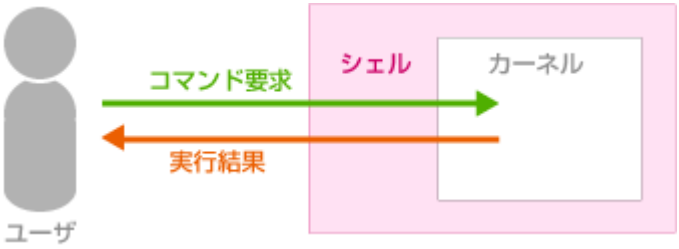


図 1：シェルはカーネルとユーザとの橋渡しをする

「シェル」は UNIX における名称で、一般的には「コマンドインタプリタ」などと呼ばれる種類のプログラムです。現在ではさまざまな種類のシェルが存在しますが、HP-UX では POSIX SHELL（sh）が標準シェルとして使用されています。

現在使用されているシェルを確認するには次のように実行します。

```
$ echo $SHELL 【Enter】
/sbin/sh
```

四色君：
シェルは Windows のコマンドプロンプトみたいなものですか？

マリー先生：
広い意味ではそうね。だけど UNIX のシェルのほうが遙かに高機能で使い勝手も上よ。

タックス君：
シェルもプログラムファイルとしてどこかに保存されているのですか？

マリー先生：
システム的にはシェルもひとつのプログラムファイルにすぎないの。たとえば POSIX SHELL のパスは「/sbin/sh」よ。

シェルの履歴機能

POSIX SHELL には、コマンドラインの編集機能が用意されています。これを使用すると、たとえば以前に実行したコマンドを呼び出すことが可能です。なお、UNIX には vi と emacs という二大エディタがありますが、シェルの編集機能では、キーアサインをそのいずれかのエディタのものに設定可能です。本稿では、vi エディタのキーアサインを基本に説明します。

コマンドラインで esc キーを押すと編集モード（vi のコマンドモード）に入ります。編集モードとは、ユーザがタイプした文字がカーソル位置に挿入されるのではなく、コマンドとして認識されるモードです。

たとえば、編集モードで「k」をタイプすると過去に実行したコマンドをひとつずつ遡って表示します。行き過ぎてしまった場合には「j」で戻ります。目的のコマンドが見つかったら Enter キーを押すと実行されます。

なお、編集モードを抜け、カーソル位置にテキストを入力していく入力モードに戻るには「i」をタイプします。

タックス君：
あれ？　esc キーを押してもだめなんですけど……。

マリー先生：
デフォルトのエディタを示す EDITOR 環境変数という変数が設定されていないのかもね。次のように実行して vi に設定すればいいわ。

```
$ export EDITOR=vi
```

シェルの環境設定ファイルはホームディレクトリの下の「.profile」なので、このファイルに「export EDITOR=vi」を追加しておけば、次回のログインからは自動的に設定されるわよ。

編集モードのキー操作

編集モードでは vi エディタのさまざまなキー操作が使用できます。vi エディタの使い方については次の機会に説明するとして、ここではもっとも基本的なキー操作を説明しましょう。たとえば、カーソルの前後の移動は「h」と「l」に割り当てられています。

```
echo Hallo UNIX
      ←h   l→
```

また、カーソル位置の文字を削除するには「x」をタイプします。表 1 に、基本的なキーアサインをまとめておきます。

＜表1：viの基本的なキーアサイン＞

h	1 文字左に移動
j	1 つ後に実行したコマンドを呼び出す
k	1 つ前に実行したコマンドを呼び出す
l	1 文字右に移動
x	カーソル位置の文字を削除
D	カーソル位置から最後までを削除
$	最後の文字に移動
^	最初の文字に移動

四色君：
UNIX のコマンドはどこに保存されているのですか？

マリー先生：
基本的なコマンドは/bin、/usr/bin/、/sbin、/usr/sbin といったディレクトリに保存されているわ。主に/bin と/usr/bin ディレクトリはユーザコマンド、/sbin と/usr/sbin ディレクトリは管理コマンドというように分類されているの。

たとえば、/bin ディレクトリを ls コマンドで表示してみると、数多くのコマンドが用意されていることがわかるわ。

```
$ ls /bin 【Enter】
Uutry           expr            mm              showaudio
X11             factor          model           showexternal
acl_edit        false           more            shownonascii
adb             fastmail        mpsched         showpicture
～中略～
evmwatch        mkstr           sh              zcat
ex              mktemp          shar            zcmp
expand          mkuupath        shl             zdiff
```

シェルが、ユーザが入力したコマンドを探しにいくディレクトリのことを「コマンド検索パス」と呼ぶのだけど、コマンド検索パスに保存されていて、かつ実行権限が与えられたファイルは、ファイル名つまりコマンド名で実行できるわけ。

四色君：
UNIX のコマンドはすべてファイルとして存在するのですか？

マリー先生：
それだけじゃなくて、あらかじめシェルに内蔵されているコマンドもあるの。それらのコマンドのことをシェルの「組み込みコマンド」と呼ぶの。たとえばディレクトリを移動する cd コマンドや pwd、echo といったコマンドは組み込みコマンドね。

よく使うディレクトリを表す特別な記号

シェルには、ホームディレクトリや 1 つ上のディレクトリといった、頻繁にアクセスするディレクトリに関して、次のような記号が割り付けられ、簡単に指定できるようになっています。

＜表2：ディレクトリを表す記号＞

~	ホームディレクトリ
..	1 つ上のディレクトリ
.	カレントディレクトリ

これらの 3 つは、シェルを活用していくうえで欠かせない表記なので、確実に覚えるようにしてください。

たとえば、「.. （ピリオド 2 つ）」は、カレントディレクトリの 1 つ上のディレクトリを表します。現在ホームディレクトリにいるときに、1 つ上のディレクトリ、つまり/home ディレクトリの一覧を表示するには次のようにします。

```
$ ls .. 【Enter】
iwww      lost+found  naoko     o2        owww      sfmdb     taro
```

これらの記号は通常のパスの指定と同じく、ディレクトリの区切りを示す「/」と組み合わせて使えます。たとえば、現在ホームディレクトリにいるときに、2 つ上のディレクトリの下にある lib ディレクトリの一覧を表示するには次のようにします。

```
$ ls ../../lib 【Enter】
Motif1.1                  libcext.1                 libomp.sl
Motif1.2                  libcext.sl                libpam.1
Motif1.2_R6               libcl.0                   libpam.sl
Motif2.1                  libcl.1                   libpsm.1
X11                       libcl.2                   libpsm.sl
X11R4                     libcl.sl                  libpthread.1
～以下略～
```

マリー先生：
「..」と「.」の 2 つは実際にはファイルで、ls コマンドの一覧で見ることもできるの。

タックス君：
あれ？「ls 【Enter】」を実行しても見えないんですけど……。

マリー先生：
「..」と「.」はファイル名の先頭が「.」で始まるから隠しファイルなの。「-a」オプションをつけて実行すれば見えるわよ。

```
$ ls -a 【Enter】
.                .cshrc           .login           .xjim_onlud
..               .dt              .mozilla         Documents
.ICEauthority    .dtprofile       .profile         Pictures
.TTauthority     .dtprofile.org   .sh_history      Readme.txt
.Xauthority      .exrc            .xjim_defaults   original
.atokx           .firefox-license .xjim_keybind    profile.txt
.bash_history    .fonts.cache-1   .xjim_learn      samples
```

四色君：
あっ、ほんとだ。でも「..」と「.」は普通のファイルとしては使わないから、ちょっとじゃまですね。

マリー先生：
それだったら「-a」の代わりに「-A」オプションを指定すれば、「..」と「.」は表示されなくなるわよ。

```
$ ls -A 【Enter】
.ICEauthority    .dtprofile       .profile         Pictures
.TTauthority     .dtprofile.org   .sh_history      Readme.txt
.Xauthority      .exrc            .xjim_defaults   original
.atokx           .firefox-license .xjim_keybind    profile.txt
.bash_history    .fonts.cache-1   .xjim_learn      samples
.cshrc           .login           .xjim_onlud
.dt              .mozilla         Documents
```

「~（チルダ）」は自分のホームディレクトリを表します。たとえば、任意のディレクトリにいるときに、ホームディレクトリの下にある Documents ディレクトリに移動するには次のようにします。

```
$ cd ~/Documents
```

四色君：
「ls」とか「cd」といったように、UNIX のコマンドはアルファベット 2 文字とかか 3 文字のものが多いのですか？

マリー先生：
そうね、たとえば「cd」は「Change Directory」の頭文字といったように、英語の単語を略したものが多いの。コマンドを作った人たちは、少しでもタイプする量を少なくしたかったのかもね。

休み時間：いろいろなシェル

現在では UNIX 系 OS で動作する多くのシェルがあり、使い勝手や機能はさまざまです。使い慣れた日本語入力システム文書がすらすら作成できるようにに、自分に合ったシェルを使うとコマンドラインの作業効率も向上します。

HP-UX の標準シェルである POSIX SHELL は、ksh をもとに POSIX に準拠させたものです。シェルは Bourne シェルを起源とする B シェル系と、csh を起源とする C シェル系に分類されることがあります。以下に、主なシェルについてまとめておきましょう。

- **sh - Bourne シェル**

UNIX の黎明期に開発された標準的なシェルです。対話機能がとぼしいことから現在ではログインシェルとして使用されることはほとんどありませんが、シェルスクリプト（シェルで記述したプログラム）のリファレンス的シェルとして使用されています。なお、システムによっては、他のシェルが「sh」という名前でインストールされています。たとえば HP-UX では POSIX SHELL が/sbn/sh として、Linux や Mac OS X では bash が/bin/sh としてインストールされています。

- **ksh - Korn シェル**

Bourne シェルを機能強化し、コマンドラインの編集機能やヒストリ機能といった対話機能を備えたシェルです。

- **bash - Bourne Again Shell**

GNU プロジェクトによる Bourne シェルの機能強化版です。「生まれ変わった Bourne シェル」という意味で、bash（Bourne Again SHell）と名付けられています。Bourne シェルに比べて特に対話機能が強化されており、Linux では標準シェルとして採用されています。

- **csh - C シェル**

BSD（カリフォルニア大学バークレー校）で開発された C シェル系の元祖で、if 文や while 文といった C 言語の文体に似た制御構造が用意されています。

- **tcsh - TENEX C シェル**

csh の機能拡張版です。ファイル名補完機能やコマンドライン編集の機能が強化され、C シェル系としてはもっとも広く使用されています。フリーの UNIX として Linux と並ぶ人気の FreeBSD での標準シェルです。

- **zsh - Z シェル**

B シェル系と C シェル系のメリットを併せ持つ、現在もっとも多機能なシェルです。

POSIX SHELL 以外のシェルを使いたいのであれば、bash、tcsh、zsh のいずれかがお勧めです。いずれも Porting And Archive Centre for HP-UX より HP-UX 11i v3 用のパッケージがダウンロードできます。インストールにはスーパーユーザの権限が必要ですので、システムの管理者にインストールしてもらってください。

2 時間目：ワイルドカードを知る

ここでちょっと話題を変えて、ファイルのコピー方法について簡単に説明しておきましょう。ファイルのコピーには cp コマンドを使います。cp コマンドの詳しい使い方の説明は次回に譲ることにして、ここでは、基本的な使い方を紹介しましょう。

次に示すように、引数の順番は、最初にコピーするファイルのパス、2 番目にコピー先のパスを指定します。逆ではないことに注意してください。

```
cp コピーもとのパス コピー先のパス
```

たとえば、カレントディレクトリの下の「Readme.txt」を、カレントディレクトリの下の Documents ディレクトリの下に「backup.txt」という名前でコピーするには次のようにします。

```
$ cp Readme.txt Documents/backup.txt
```

このとき、同じ名前で別のディレクトリにコピーする場合には、コピー先にファイル名は省略可能です。前述の Readme.txt を Documents ディレクトリの下に同じ名前でコピーするには次のようにします。

```
$ cp Readme.txt Documents/
```

このとき、コピー先の最後の「/」は付けても付けなくてもかまいません。

四色君：
「~」や「..」といった記号が便利なのはわかりましたが、カレントディレクトリを表す「.」は使い道があるのですか？　たとえば、カレントディレクトリの一覧を表示するのに、わざわざ「ls . <Enter>」としなくても、単に「ls <Enter>」としても同じですよね。

マリー先生：
そうね、ls コマンドなどでは引数を指定しないとカレントディレクトリが対象になるから、ピリオド「.」は不要ね。ピリオド「.」がよく使われるは、たとえば、カレントディレクトリにファイルをコピーするといったときかな。次の例では、cp コマンドのコピー先に「.」を指定して、「samples/sample.txt」をカレントディレクトリにコピーしているの。

```
$ cp samples/sample.txt .
```

ワイルドカード

たとえば、たくさんのファイルの中から拡張子が「.html」のファイルだけを、別のディレクトリにコピーしたいといった場合はどうすればよいでしょうか？　1 つずつコピーしてもかまいませんが面倒ですし、コピーし忘れもあるでしょう。そのようなケースで活躍するのが、「ワイルドカード」と呼ばれる特別な記号です。ワイルドカードはトランプにおけるジョーカーのような万能カード的なイメージで使える記号です。

＜表3：ワイルドカードの記号と意味＞

*	0 個以上の任意の文字列に一致する
?	1 文字と一致する

まず、「*（アスタリスク）」はファイル名の中の任意の文字列と一致します。たとえば、拡張子が「.html」のファイルは「*.html」と表記できるわけです。したがって、カレントディレクトリの Documents ディレクトリから拡張子が「.html」のファイルを、public_html ディレクトリにコピーするには次のようにします。

```
$ cp Documents/*.html public_html/
```

タックス君：
「*」は 0 個以上の任意の文字列に一致するということは、たとえば、「.html」という名前の隠しファイルがあった場合、「cp Documents/*.html public_html/」でそれもコピーされるのですか？

マリー先生：
隠しファイルはコピーされないわ。ワイルドカードはファイル名の先頭の「.」には一致しないようになっているの。

「?」は任意の 1 文字に一致します。たとえば Documents ディレクトリから、ファイル名が 7 文字のファイルの一覧を ls コマンドで表示するには次のようにします。

```
$ ls Documents/???????  【Enter】
Documents/2007doc  Documents/ABCnews
```

「?」と「*」を組み合わせることでより柔軟な指定も可能です。 たとえば、カレントディレクトリの下で、拡張子が 3 文字のファイルの一覧を表示するには次のようにします。

```
$  ls *.???  【Enter】
Readme.txt   profile.txt  sample.png
```

四色君：
ワイルドカードはそれぞれのコマンドがファイル名に展開しているのですか？

マリー先生：
いいえ、ワイルドカードはシェルによって展開された後に、カーネルに伝えられるの。たとえば、ユーザが「ls *.html」のように実行すると、シェルがカレントディレクトリから拡張子が「.html」のファイルを探して「ls sample.html info.html 」のようなコマンドがカーネルに送られるというわけ。

練習問題

第 1 問：2 つ上のディレクトリの下にある usr ディレクトリの一覧を表示するコマンドはどれか？

a)　ls ../usr
b)　ls ~/./usr
c)　ls ./usr
d)　ls ../../usr

d)　ls ../../usr
1 つ上のディレクトリは「..」で表記できる。したがって 2 つ上のディレクトリは「../..」となる。

第 2 問：現在/home ディレクトリにいるときに、ユーザ「taro」のホームディレクトリに移動するコマンドとして正しくないのはどれか？

a)　cd ./taro
b)　cd taro
c)　cd /home/taro
d)　cd home/taro

d)　cd home/taro
先頭が「/」で始まらないパスは相対パスと見なされる。したがって「home/taro」はカレントディレクトリの下の「home/taro」、つまり「/home/home/taro」を表すことになる。

第 3 問：Documents ディレクトリの下の profile.txt をカレントディレクトリにコピーするコマンドはどれか？

a)　cp Documents/profile.txt .
b)　cp Documents/profile.txt /home
c)　cp Documents/profile.txt ~
d)　cp Documents/profile.txt ..

a)　cp Documents/profile.txt .
カレントディレクトリにファイルをコピーする場合、cp コマンドの 2 番目の引数に「.」を指定すればよい。

第 4 問：~/Documents ディレクトリの下で、ファイル名の先頭が A で始まりかつ拡張子が 3 文字のファイルを、カレントディレクトリの下の backup ディレクトリにコピーするコマンドはどれか？　このとき拡張子を除くファイル名は 2 文字以上とする。

a)　cp ~/Documents/A?*.* backup
b)　cp ~/Documents/A?*.??? backup
c)　cp ~/Documents/A*.??? backup
d)　cp ~/Documents/A??.??? backup

b)　cp ~/Documents/A?*.??? backup
(c)のように cp ~/Documents/A*.??? backup とすると、「A.txt」のような拡張子を除くファイル名が 1 文字のファイルもコピーされてしまう。

UNIX の教科書「基礎編」

第 4 日目：ファイルの基本操作

2008 年 2 月　大津 真

今回は、ファイルのコピーや移動、削除といったファイルの基本操作のためのコマンドを掘り下げて説明していくことにしましょう。その後で、別名でファイルにアクセスするリンクについても説明します。今回紹介するコマンドはどれも、前回紹介した「*」や「?」といったワイルドカード、および「..」や「.」といった特別なディレクトリを示す記号と組み合わせることで、より柔軟な処理が可能になります。

マリー先生：
今回はファイル操作に関する基本コマンドの取り扱いについて説明しましょう。ここまでの操作を理解すれば、ターミナル上での基本的な操作はだいたい OK なのでがんばりましょう。
その前に、前回の説明に関してなにか質問ありますか？

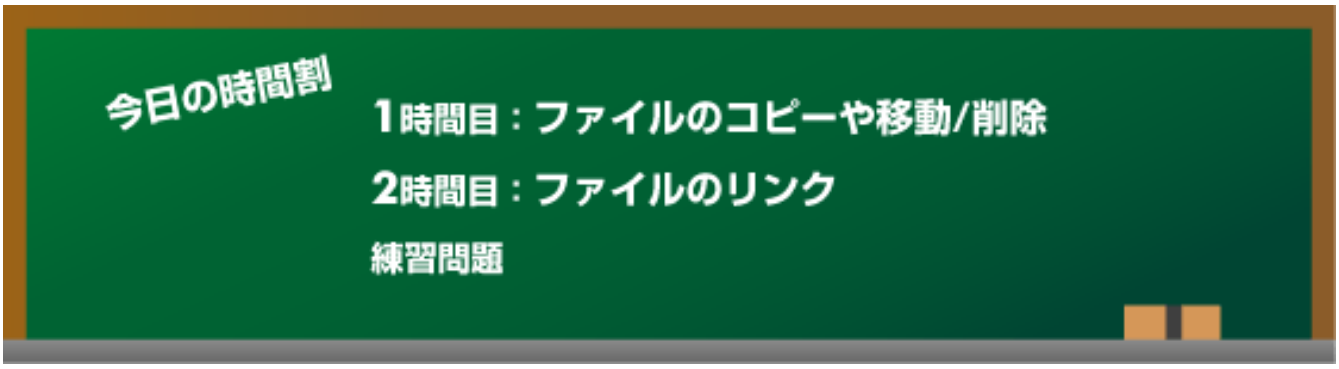
タックス君：
先日、先輩に bash のパッケージをインストールしてもらったのですが、bash を起動するにはどうすればいいのですか？

マリー先生：
bash のインストール先にもよるけど、/usr/local/bin/bash にインストールされている場合、単に試すだけだったら、コマンドラインで「/usr/local/bin/bash 【Enter】」（設定によっては「bash 【Enter】」）とすればいいわ。

```
$ /usr/local/bin/bash 【Enter】
```

タックス君：
ログインしたときに POSIX SHELL の代わりに bash を使うように設定することはできますか？

マリー先生：
ログインしたときに起動するシェルのことを「ログインシェル」というの。ログインシェルは chsh コマンドで変更できるわ。ただし、初期状態ではシステムファイルを設定しないと、システムに標準で用意されているシェルしかログインシェルにできないの。あるいは、HP-UX 管理ツールである SMH（System Management Homepage)の ugweb を使ってもログインシェルを変更できるわ。いずれにしてもスーパーユーザの権限が必要なので、もし bash をログインシェルにしたかったら管理者に相談してね。



1 時間目：ファイルのコピーや移動／削除

ファイルをコピーする cp コマンドの基本的な使い方について前回簡単に説明しましたが、この時間ではもう少し掘り下げて説明しましょう。

前回は、あるディレクトリの下の複数のファイルをまとめてコピーするのに使用するのに、「*」「?」といったワイルドカードを使用する方法について紹介しました。覚えていますでしょうか？

たとえば、カレントディレクトリの下にある拡張子が「.txt」のファイルを、ホームディレクトリ「~」の下の backup ディレクトリにコピーするには、次のようにします。

```
$ cp *.txt ~/backup
```

こうするとワイルドカード「*」がシェルによって展開されます。カレントディレクトリに sample.txt、info.txt、readme.txt の 3 つのファイルがあったとすると、次のような命令がカーネルに送られるわけです。

```
cp sample.txt info.txt readme.txt ~/backup
```

マリー先生：
cp コマンドでは、コピー先に同じ名前のファイルがあっても警告なしに上書きしてしまうので注意してね。

四色君：
それはちょっとキケンですね。

マリー先生：
心配なら「-i」オプションを指定すれば、上書きするかどうかをファイルごとに確認しくれるわ。前の例で「-i」オプションを指定すれば、backup ディレクトリに同じファイルがあった場合には上書きしていいかその都度訊いてくるの。「y 【Enter】」で上書き、それ以外の文字を入力すればコピーされないわ。

```
$ cp -i  *.txt ~/backup
overwrite /home/taro/backup/A.txt? (y/n) y    ←上書きする
overwrite /home/taro/backup/AB.txt? (y/n) n ←上書きしない
```

ディレクトリを丸ごとコピーする-r オプション

それでは、あるディレクトリを丸ごと階層構造を保持したまま、別のディレクトリにコピーすることはできるでしょうか？　答えは、もちろん可能です。それには「-r」オプション（もしくは「-R」オプション）を指定して cp コマンドを実行します。このことを、"ディレクトリを再帰的にコピーする"などといいます。

たとえば、Documents ディレクトリを丸ごと、backup ディレクトリの下にコピーするには次のようにします。

```
$ cp -r Documents backup/
```

四色君：
この例だと、backup ディレクトリの下に Documents ディレクトリが作成されますけど、違うディレクトリ名でコピーすることもできますか？

マリー先生：
できるわよ。たとえば、DocBackup というディレクトリ名にするには次のようにすればいいわ。

```
$ cp -r Documents backup/DocBackup
```

このとき、同じ名前のディレクトリが存在していた場合には、その下にコピーされてしまうので注意してね。前の例で、cp コマンドを実行するときに、backup/DocBackup ディレクトリが存在していた場合には backup/DocBackup/Documents ディレクトリが作成されてしまうの。

ファイルの情報を保持したままコピーする-p オプション

cp コマンドでファイルをコピーすると、ファイルの更新日時はコピーした日時になります。またファイルのパーミッション（アクセス権限）などの情報はシェルの設定によって変更される場合があります。

```
$ cp index.html index2.html
$ ls -l
total 32
-rw-rw-rw-   1 o2     users       5 Jan  1 00:00 index.html
-rw-r--r--   1 o2     users       5 Jan 29 22:48 index2.html  ←パーミッションと更新日時が変更された
```

もとのファイルの情報をそのまま保持したい場合には、「-p」オプションを指定して実行します。

```
$ cp -p index.html index3.html
$ ls -l
total 48
-rw-rw-rw-   1 o2     users       5 Jan  1 00:00 index.html
```

```
-rw-r--r--   1 o2     users        5 Jan 29 22:48 index2.html
-rw-rw-rw-   1 o2     users        5 Jan  1 00:00 index3.html          ←情報が保持される
```

ファイルを移動する mv コマンド

コピーではなくてファイルを移動したい場合には mv コマンド使います。cp コマンドの引数の順番と同じく、最初の引数にもとのファイルのパスを、2 番目の引数にコピー先のパスを指定します。

たとえば、カレントディレクトリの下の ReadMe.txt を、ホームディレクトリ「~」の下に移動するには次のようにします。

```
$ mv Readme.txt ~/
```

なお、mv コマンドはディレクトリを丸ごと移動することもできます。このとき、cp コマンドと異なり「-r」オプションは必要ありません。たとえばカレントディレクトリの下にある samples ディレクトリを、ホームディレクトリの下にある Pictures ディレクトリに移動するには、次のようにします。

```
$ mv samples ~/Pictures
```

タックス君：
移動ではなく名前を変更するにはどうすればいいのですか？

マリー先生：
それも mv コマンドで OK よ。つまり mv には名前の変更といった意味合いもあるの。たとえば、カレントディレクトリの下にある「file.txt」の名前を「myFile.txt」にするには、次のようにすればいいわ。

```
$ mv file.txt myFile.txt
```

ファイルを削除する rm コマンド

ファイルの削除には rm コマンドを使います。たとえば、カレントディレクトリの下にある old.txt を削除するには次のようにします。

```
$ rm old.txt
```

四色君：
rm コマンドの引数にワイルドカードは使えますか？

マリー先生：
もちろん使えますよ。たとえば、UNIX 系 OS ではバックアップファイルのファイル名の最後にチルダ「~」が使用されることがしばしばあるの。カレントディレクトリからそれらのファイルをまとめて削除するには、次のようにすればいいわ。

```
$ rm *~
```

タックス君：
ワイルドカードを使ってファイルをまとめて削除できるのは便利だけど、ちょっとコワイですね。

マリー先生：
そんな場合には、cp コマンドと同じく「-i」オプションをつけるといいわ。そうすれば、ひとつずつ確認してからファイルを削除してくれるの。

```
$ rm -i *.txt
Readme.txt: ? (y/n) n
ok.txt: ? (y/n) y
profile.txt: ? (y/n) m
```

ディレクトリを丸ごと削除する-r オプション

「-r」オプションを指定すると、ディレクトリを丸ごと削除することができます。このオプションを使用するとホームディレクトリを丸ごと削除するといったことも簡単にできてしまいますので、実行には細心の注意をはらってください。

たとえば、old ディレクトリ以下を丸ごと削除するには次のようにします。

```
$ rm -r old
```

ディレクトリを削除する rmdir コマンド

ディレクトリの削除に特化したのが rmdir コマンドです。 たとえば、カレントディレクトリの下にある work ディレクトリを削除するには次のようにします。

```
$ rmdir work
```

ただし、rmdir が削除できるのは空のディレクトリだけです。その下にファイルやディレクトリがあるとエラーとなり削除できません。

タックス君：
「rm -r」コマンドでディレクトリを丸ごと削除できるのですから、rmdir コマンドは必要ない気がしますけど。

マリー先生：
「rm -r」はその下にファイルがあっても、指定したディレクトリ以下をすべて削除してしまうので、ある意味危険なコマンドなわけ。だから実行には細心の注意を払う必要があるの。それに対して、rmdir はその下にファイルがあると削除できないから安心なのよ。

休み時間：CDE のファイルマネージャ

HP-UX のデスクトップ環境には、The Open Group が管理する商用のデスクトップ環境である CDE（Common Desktop Environment）が採用されています。ファイルのコピー／移動、削除、2 時間目で説明するシンボリックリンクの作成といったファイルの基本操作は、CDE のファイルマネージャを使っても行えます。

そのままファイルをドラッグ&ドロップすると移動になりますが、Ctrl キーを押しながらドラッグ&ドロップすると移動になります。

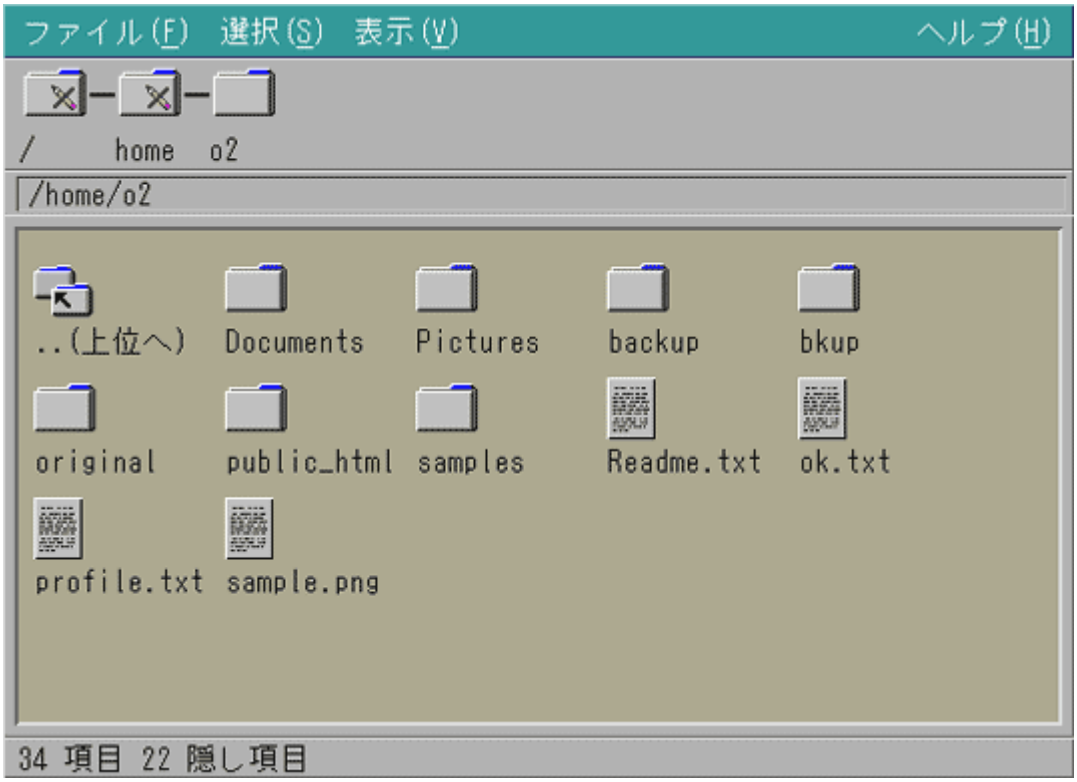


図 1：CDE のファイルマネージャ

なお、コピー／移動先のパスをダイアログボックスで指定することもできます。たとえば、コピーする場合にはファイルを選択してから、「選択」メニューから「コピー先を選択」を選択します。表示されるダイアログボックスで、コピー先のディレクトリのパスと、ファイル名を指定して Enter キーを押すとコピーが実行されます。

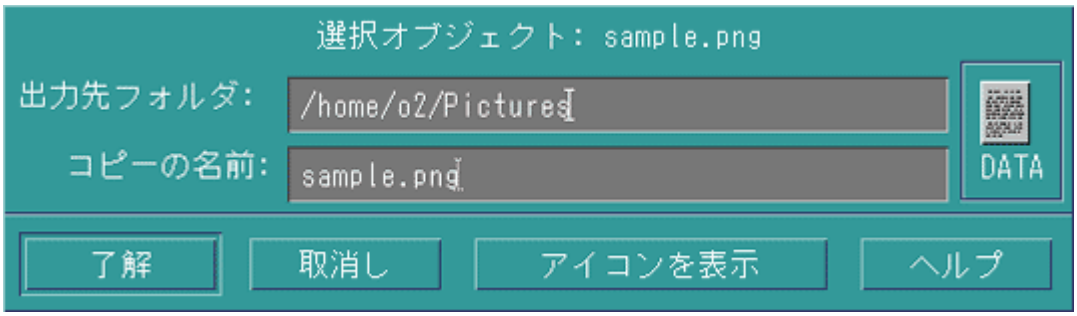


図 2：ダイアログボックスを使ったファイルのコピー

2 時間目：ファイルのリンク

UNIX には、「リンク」と呼ばれるファイルやディレクトリに別名でアクセスする仕組みがあります。Windows におけるショートカットや Mac OS X におけるエイリアスのようなイメージでとらえるといいでしょう。リンクは「シンボリックリンク」と「ハードリンク」の 2 種類に大別されます。いずれも ln コマンドを使用して作成します。なお、リンクを作成することを"リンクを張る"と言うことがあります。

四色君：
シンボリックリンクとハードリンクってどっちがよく使用されるのですか？

マリー先生：
普通にユーザが使用するのはシンボリックリンクがほとんどね。というのは、ハードリンクにはその仕組みから次のような制約があるの。

- 異なるパーティションにリンクを作成できない
- ディレクトリのリンクは作成できない

シンボリックリンクを作成する

リンクを張るには ln コマンドを使用します。cp コマンドの引数の順番と同じく、最初の引数にもとのファイルのパスを指定します。

```
ln [-s] もとのファイルのパス リンクのパス
```

このとき、ハードリンクの場合にはオプションは不要ですが、シンボリックリンクを作成する場合には「-s」オプションを指定します。

たとえば、カレントディレクトリの下にある original.txt のシンボリックを symlink.txt として作成するには、次のようにします。

```
$ ln -s original.txt symlink.txt
```

「ls -F」コマンド（もしくは lsf コマンド）で結果を確認すると、シンボリックリンクには最後に「@」が表示されます。

```
$ ls -F 【Enter】
original.txt   symlink.txt@
```

また「ls -l」コマンド（もしくは ll コマンドでは）では、先頭のファイルタイプのハイフンが「l」となり、最後に「-> リンク先のパス」が表示されます。

```
$ ls -l 【Enter】
total 256
-rw-r--r--   1 o2         users       128847 Jan 28 17:45 original.txt
lrwxr-xr-x   1 o2         users           11 Jan 28 17:45 symlink.txt -> original.txt
```

```
↑                                ↑                ↑
シンボリックリンクは「l」          サイズは小さい     リンク先のパス
```

シンボリックリンクの実体は、リンク先のファイルまでのパスが格納されたファイルです。そのため、ファイル容量は非常に小さくなっています。

なお、シンボリックリンクを削除する際には rm コマンドを実行します。rm コマンドはリンク先のファイルではなく，シンボリックリンク自体を削除します。

同じ名前でシンボリックリンクを作成する

ln コマンドの 2 番目の引数にディレクトリを指定すると、そのディレクトリの下に、同じ名前でシンボリックリンクが作成されます。たとえば、「link1/original.txt」のシンボリックリンクを、link2 ディレクトリの下に同じ名前で作成するには次のようにします。

```
$ ln -s link1/original.txt link2
$ ls -l link2
total 0
lrwxr-xr-x   1 o2     users        18 Jan 29 17:39 original.txt -> link1/original.txt
```

タックス君：
使われるのはシンボリックリンクのほうが一般的なのに、ハードリンク作成はオプションなしの ln コマンドで、シンボリックリンク作成に「-s」オプションが必要なのはなぜですか？
逆でもいいのに……。

マリー先生：
シンボリックリンクのほうが歴史的に新しい機能だからよ。初期の ln コマンドはハードリンクだけを作成するコマンドだったわけ。

タックス君：
リンク先のファイルがなくなってしまったらどうなるのですか？

マリー先生：
シンボリックリンクはそのまま残るわ。ただし、参照先がないわけだから、たとえば cat コマンドで中身を表示しようとするとファイルが見つからないといったエラーになるの。

```
$ rm original.txt
$ cat symlink.txt
cat: Cannot open symlink.txt: No such file or directory
```

シンボリックリンクの注意点

シンボリックリンクはリンク先のパスを格納しているだけなので、ディレクトリの階層構造を理解していないと場合によってはおかしなことになるので注意してください。たとえば現在ホームディレクトリにいるときに、その下の original.txt のシンボリックリンクを、work ディレクトリの下に symlink.txt という名前で作成したいとしましょう。

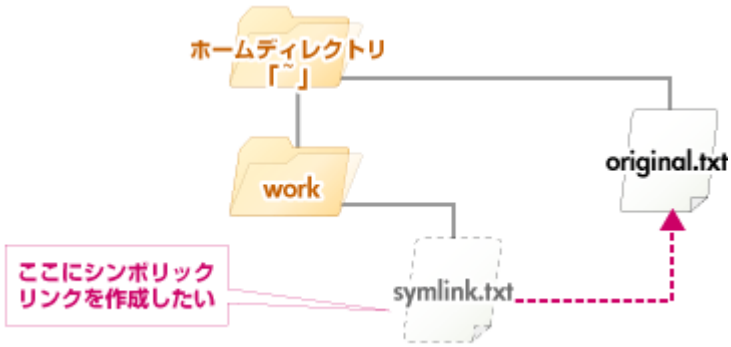


図 3：work ディレクトリに original.txt のシンボリックリンクを作成したい

この場合、次のように、ln コマンドの 2 番目の引数として「work/symlink.txt」を指定してシンボリックリンクを作成してもうまくいきません。

```
$ ln -s original.txt work/symlink.txt       ←シンボリックリンクは作成できるが
$ cat work/symlink.txt
cat: Cannot open work/symlink.txt: No such file or directory     ←表示しようとするとエラーとなる
```

というのは、「ls -l」コマンドで結果を確認するとわかりますが、work/symlink.txt は、同じディレクトリ内に original.txt を参照しているからです。

```
$ ls -l work/symlink.txt
lrwxr-xr-x   1 o2     users        12 Jan 28 17:58 work/symlink.txt -> original.txt
```

この場合、最初の引数を「../original.txt」のように、シンボリックリンクからの相対パスで指定する必要があるわけです。

```
$ ln -s ../original.txt work/symlink.txt
$ cat work/symlink.txt
Hello    ←今度はうまくいく
```

タックス君：
ln コマンドの最初の引数のリンク先を、常に「/home/taro/original.txt」のように絶対パスで指定すれば、問題はありませんよね。

マリー先生：
それでもいいけど、ディレクトリを丸ごとコピーしたり、バックアップしたファイルを復元する時に、リンクがおかしくなる場合があるので注意してね。

ハードリンクの作成

続いてハードリンクについて説明しましょう。ハードリンクを作成するには ln コマンドを「-s」オプションなしで実行します。ハードリンクは、ファイルの"実体"を別名でアクセスする機能です。たとえば、カレントディレクトリの下に original.txt というファイルがあるとしましょう。「ls -l」で確認すると初期状態ではハードリンクの数は「1」です。

```
$ ls -l original.txt 【Enter】
-rw-r--r--  1 o2      users     128847 Jan 29 16:57 original.txt
            ↑ハードリンクの数は「1」
```

original.txt のハードリンクを hardlink.txt として作成するには次のようにします。

```
$ ln original.txt hardlink.txt 【Enter】
```

「ls -l」で確認すると、どちらもハードリンクの数は「2」となります。

```
$ ls -l 【Enter】
total 512
-rw-r--r--  2 o2      users     128847 Jan 29 16:57 hardlink.txt
-rw-r--r--  2 o2      users     128847 Jan 29 16:57 original.txt
            ↑ハードリンクの数は「2」
```

ハードリンクを作成すると、リンク先、ハードリンクのどちらも同じファイルの実体（ハードディスク上のファイルが保存されている領域）を参照します。この例では、orginal.txt と hadrdlink.txt は、同じファイルの実体を指し示すようになります。つまり、システム的にはどちらも等価で、シンボリックリンクのようにリンク先、リンク元といった区別はありません。

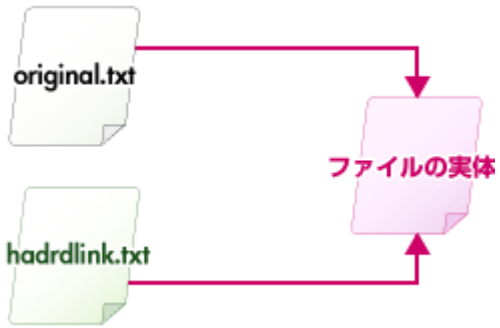


図 4：リンク先とハードリンクは同じファイルの実体を指す

UNIX システムでは、ハードディスク上の個々のファイルの実体を「i ノード番号」という番号で管理しています。i ノード番号は、ls コマンドに「-i」オプションを指定すれば確認できます。次に実行結果を示します。orginal.txt と hadrdlink.txt のどちらも同じ i ノード番号である点に注目してください。

```
$ ls -i 【Enter】
236 hardlink.txt    236 original.txt        ←i ノード番号が同じ
```

タックス君：
ハードリンクを削除するとどうなるのですか？

マリー先生：
ハードリンクにはリンク先とリンク元という区別はないからどちらを削除しても同じよ。
同じ i ノード番号のファイルを削除していって、最終的にハードリンクの数が 0 になった時点でファイルの実体が削除されるわけ。

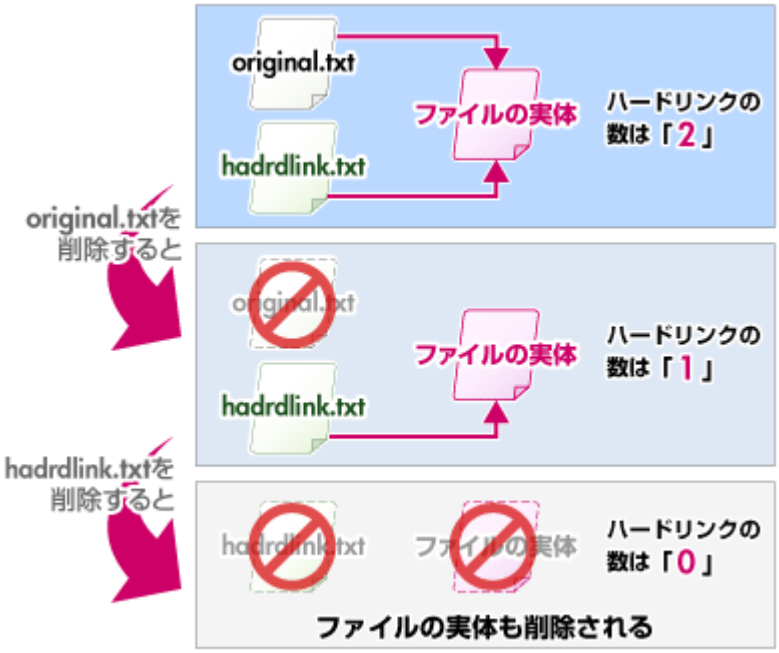


図 5：ハードリンクがすべて削除されるとファイルの実体も削除される

タックス君：
なぜディレクトリのハードリンクは作成できないのですか？

マリー先生：
ディレクトリのハードリンクの作成を許可してしまうと、意図せずディレクトリの無限ループが作成される可能性があるからよ。たとえば、「ln /dirA /dirA/linkDir」のように、/dirA ディレクトリのハードリンクを/dirA/linkDir に作成できたとしましょう。すると、「/dirA/linkdir/linkdir/linkDir……」というように無限のディレクトリができてしまっておかしなことになりるわけ。
ただし、システム的にはディレクトリのハードリンクは活躍しているの。

四色君：
というと？

マリー先生：
前回説明したカレントディレクトリを表す「.」や、ひとつ上のディレクトリを表す「..」は、実はディレクトリのハードリンクなの。たとえば、ディレクトリを作成すると、そのハードリンクの数は「1」ではなく「2」になるわよね。不思議だと思わない？

```
$ mkdir sample←sample ディレクトリを作成
$ ls -ld sample/
drwxr-xr-x   2 o2       users         96 Jan 29 17:12 sample/
                ↑ハードリンクの数は「2」
```

この「.」はそのディレクトリへのハードリンクなの。ls コマンドで、i ノード番号を確認するとわかるわよ。

```
$ ls -lid sample/
  310 drwxr-xr-x   2 o2     users       96 Jan 29 17:12 sample/
  ↑i ノード番号は「310」
$ ls -lia sample/
total 0
  310 drwxr-xr-x   2 o2     users       96 Jan 29 17:12 .
  235 drwxr-xr-x   3 o2     users       96 Jan 29 17:12 ..
  ↑「.」の i ノード番号は「310」
```

同じように、「..」はひとつ上のディレクトリへのハードリンクなの。

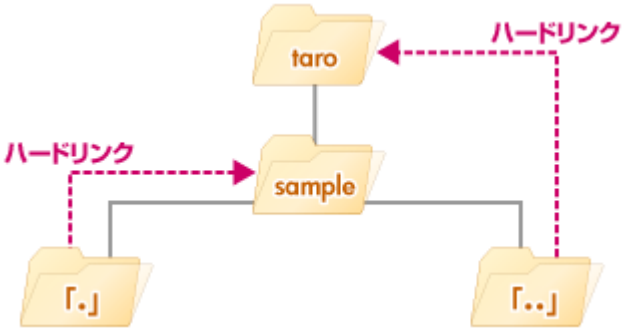


図 6：「.」や「..」はハードリンク

ディレクトリの下にディレクトリを作成していくと、ディレクトリのハードリンクの数がどのように増えていくかを実際に実行して確認してみてね。

練習問題

第 1 問：ホームディレクトリの下の work ディレクトリを、カレントディレクトリの下の 2008 ディレクトリの下に、もとの情報を保ったまま、まるごとコピーするコマンドはどれか？

a)　cp -p ~/work 2008
b)　cp -pr ~/work 2008
c)　cp -pi ~/work 2008
d)　mv -r ~/work 2008

b)　cp -pr ~/work 2008
「-r」はディレクトリを丸ごとコピーするオプション。「-p」はもとのファイルの情報を保持するオプション。

第 2 問：カレントディレクトリの下の news.html を、ひとつ上のディレクトリへ移動するコマンドはどれか？

a)　mv news.html ..
b)　mv news.html .
c)　rm news.html ..
d)　mv new.html ../..

a)　mv news.html ..
1 つ上のディレクトリに移動するには mv コマンドの、2 番目の引数に「..」を指定すればよい。

第 3 問：backup ディレクトリの下の拡張子が 3 文字のファイルを、確認しながら削除するコマンドはどれか？

a)　rm bakup/*
b)　rm -i backup/.???
c)　rm -p bakup/*.???
d)　rm -i backup/*.???

d)　rm -i backup/*.???
「-i」は確認しながら削除するオプション。拡張子が 3 文字のファイルは「*.???」で表記できる。

第 4 問：1 つ上のディレクトリの下の info.html のシンボリックリンクを、カレントディレクトリに同じ名前で作成するコマンドはどれか？

a)　ln .. /info.html
b)　ln -s ../info.html .
c)　ln ../info.html ..
d)　ln .-s info.html ..

b) ln -s ../info.html .
シンボリックリンクを作成するには、ln コマンドを「-s」オプションを指定して実行する。

UNIX の教科書「基礎編」

第 5 日目：vi エディタの操作（基本編）

2008 年 3 月　大津 真

今回は UNIX 系 OS における定番テキストエディタのひとつである vi エディタについて説明します。非常に高機能なテキストエディタですが、慣れるのに多少時間がかかるかもしれません。今回は基礎編ということで、基本的な操作にしぼって解説していくことにしましょう。

マリー先生：
今回は、UNIX におけるテキストエディタの西の横綱ともいえる vi エディタ（以下、vi）について説明しましょう。vi は、最初はちょっととっつきにくい印象があるかもしれないけれど、慣れてくるに従って手放せないツールとなること請け合いよ！　なお、東の横綱といえるのが emacs エディタです。vi と emacs はどちらも熱狂的なユーザを抱えていて、どちらが優れているかという話になると白熱した議論が朝まで……ということにもなりかねないので注意してね。

その前に、前回の説明に関してなにか質問ありますか？

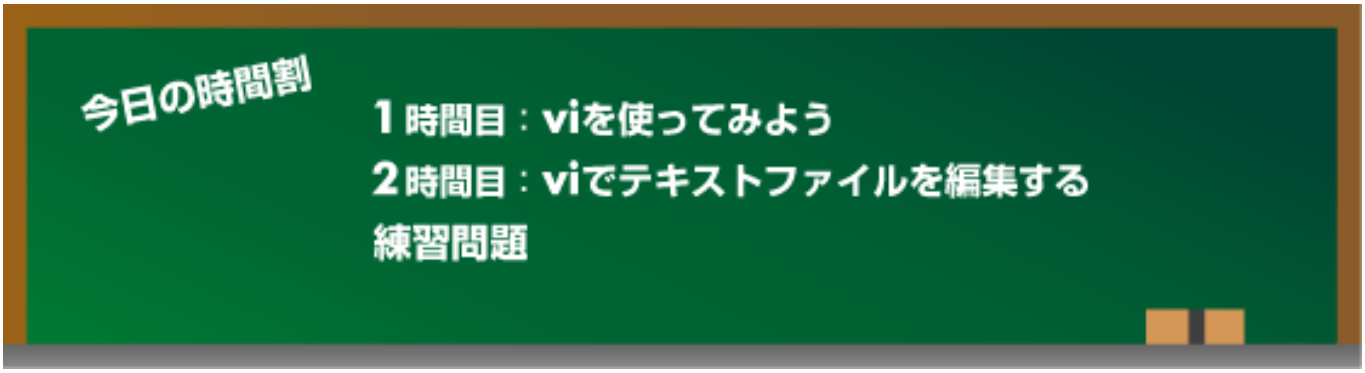
タックス君：
前回学習した mv コマンドと「?」や「*」といったワイルドカードを使用して、ファイルの拡張子をまとめて変更するということはできるのでしょうか？　たとえば、カレントディレクトリの下に「.text」という拡張子のファイルが複数あったとして、それらをまとめて「.txt」という拡張子へ変更するのに、次のようにすることはできますか？

```
$ mv *.text *.txt
```

マリー先生：
それは無理ね。ワイルドカード「*」は mv コマンドではなくシェルによって展開されるから。たとえばカレントディレクトリに「a.text」「b.text」、「c.txt」というファイルがあると、次のように展開されてしまうの。

```
$ mv a.text b.text c.txt
```

mv コマンドは引数にファイルを 2 つしか指定できないので、これを実行するとエラーになってしまうわ。拡張子をまとめて変更するといった処理は、普通はシェルスクリプトと呼ばれるシェルを使ったプログラムを作成して、for 文という繰り返しを制御するコマンドで行う必要があるの。



1 時間目：vi を使ってみよう

vi が、Windows の「メモ帳」のような GUI のテキストエディタに比べて習得に時間が多少かかる理由のひとつに、「インサートモード」と「コマンドモード」という 2 つのモードの存在があります。両者の相違を理解し、現在どちらのモードにいるかを把握しておくことが vi 使いになるための最初の関門といえます。まずはその相違について説明し、その後で vi の起動と終了方法について説明しましょう。

インサートモードとコマンドモード

インサートモードとコマンドモードの相違は、ユーザがキーボードからタイプした文字が vi でどのように扱われるかです。まずインサートモード（テキスト入力モード）ですが、これは一般的な GUI のテキストエディタと同じく、タイプした文字がカーソル位置に挿入されていくモードです。それに対してコマンドモードはタイプした文字が vi に対するコマンド、つまり指令として扱われるモードです。たとえば、インサートモードでユーザが「x」をタイプすると「x」という文字がカーソル位置に挿入されます。それに対して、コマンドモードでは「x」コマンドとして扱われます。この「x」コマンドはカーソル位置の文字を削除するコマンドです。

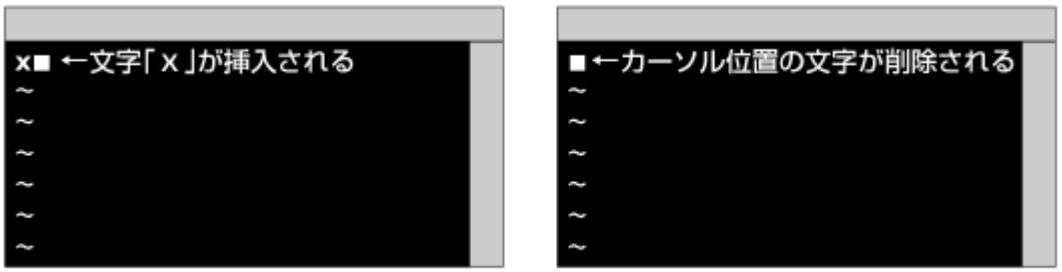


図 1：インサートモードとコマンドモードとでは同じ文字をタイプしても扱われ方が違う

四色君：
なぜ、コマンドモードが必要なのですか？

マリー先生：
ひとつには、GUI のエディタと違って、ターミナル上で動作する vi にはメニューがないし、マウスも使えないからよ。なんらかの方法でコマンドを入力しなければいけないわけ。

タックス君：
でも、普通のキーボードには文字の削除用に Delete キーがあるし、カーソルの移動は矢印キーでできるような気がするのですが……。

マリー先生：
設定によっては、文字の削除やカーソルの移動にはそれらのキーが使えるわ。でも、vi を終了したりファイルを開いたりといった操作や、文字列の検索や置換にはコマンドを実行する必要があるでしょう？

タックス君：
コマンドの入力を、Ctrl キーや Alt キーとのキーコンビネーションで行うタイプのテキストエディタもありますよね。

マリー先生：
そうね、vi のライバルといえる emacs なんかはそのタイプね。

四色君：
あと、HP-UX にも CDE 上で動作する GUI のテキストエディタもありますよね？

マリー先生：
GUI 環境が利用できるケースでは CDE のテキストエディタを使ってかまわないわ。ただしシステム管理者になると、別のホストからリモートログインしてシステムをメインテナンスするという状況も少なくないので、vi の基本的な使い方くらいはマスターしておいたほうがいいわね。

vi を起動する

前置きはこれくらいにして、実際に vi エディタを使ってみましょう。vi を起動してファイルを編集するには、次の書式で vi コマンドを実行します。

```
vi ファイルのパス
```

引数に存在しないファイルを指定した場合には、新規のファイルが作成されます。たとえば、存在しないファイル「test.txt」を引数に、「vi test.txt 【Enter】」を実行すると次のような画面になります。

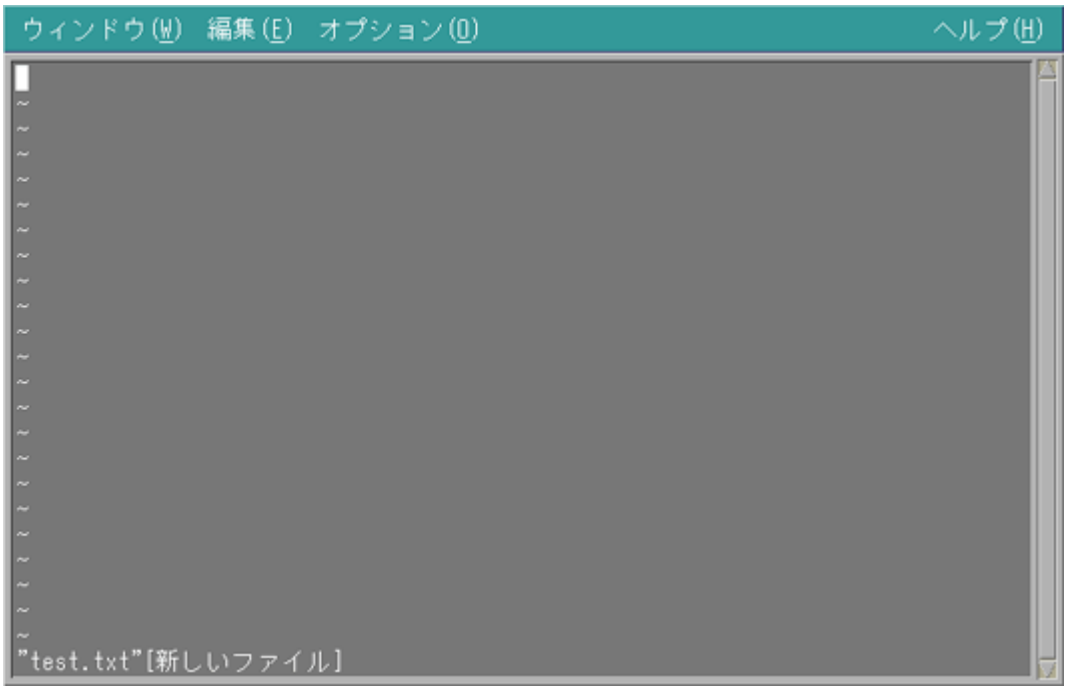


図 2：存在しないファイルを指定すると新規のファイルを作成して起動する

カーソルが左上隅で点滅し、その下の各行の先頭には「~」が表示されています。これは実際にその行に「~」という文字が格納されているわけではありませんので注意してください。vi の編集画面では、「~」は、まだなにも入力されていない行を表します。なお、編集中のファイルは「バッファ」と呼ばれるメモリ上の一時的な領域に格納され、保存コマンドを実行するまでファイルに書き込まれません。

インサートモードに移行する

起動した直後の vi はコマンドモードになっていますので、文字を入力するためにはインサートモードに移行する必要があります。ここで、「i」をタイプしてみてください。「i」はインサートモードに移行するためのコマンドのひとつです。画面には何の変化もありませんが、モードはインサートモードになるはずです。試しに、適当な文字列をタイプしてみましょう。今度はタイプした文字がカーソル位置に挿入されます。

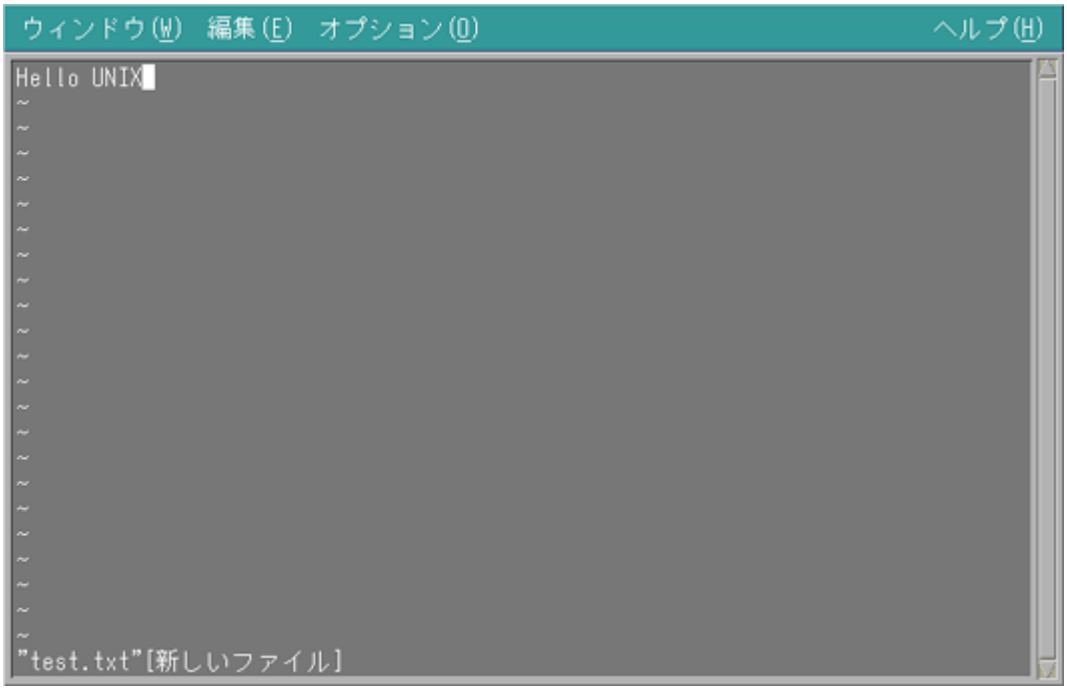


図 3：インサートモードでは文字がカーソル位置に挿入される

続いて、再びコマンドモードに戻ってみましょう。インサートモードからコマンドモードに移行するには esc キーを押します。するとカーソルが 1 文字文戻りますが、他に見た目上の変化はありません。ただし、モードは確実にコマンドモードです。試しに「h」をタイプしてみましょう。「h」はカーソルを 1 文字左に移動するコマンドです。「h」をタイプするごとにカーソルが左に移動していきます。

```
Hello UNI[X]
        ↓       「h」をタイプ【Enter】
Hello UN[I]X
        ↓       「h」をタイプ
Hello U[N]IX
```
※[]はカーソルです

ここまでの説明で、インサートモードとコマンドモードの相違がなんとなく理解できたでしょうか？　インサートモードとコマンドモードの行き来をなんども繰り返して確実に覚えるようにしてください。

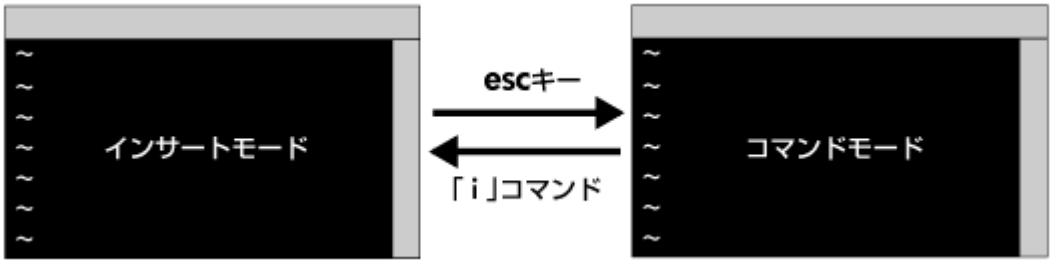


図 4：「i」コマンドと esc キーとでモードを行き来する

タックス君：
どちらのモードにいるのかわからない場合は？

マリー先生：
モードがわからなくなったら、とりあえず esc キーを押してみるといいわ。コマンドモードにいるときに esc キーを押してもモードは変わらないので、どちらのモードにいる場合でも esc キーを押すとコマンドモードになるというわけね。

バッファの内容をファイルに書き込んで vi を終了する

次に、vi を終了してみましょう。まず、コマンドモードで「:」をタイプします。 すると最下行の左に「:」が表示されます。

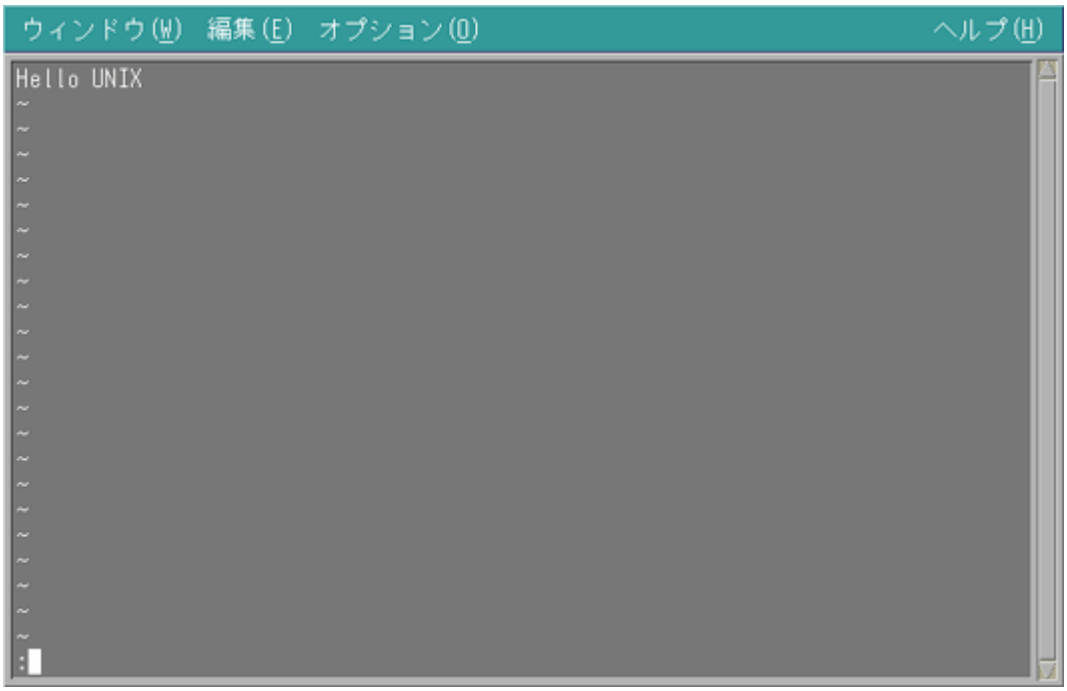


図 5：コマンドモードで「:」をタイプする

続いて「wq」とタイプして【Enter】キーを押します。「w」はファイルの書き込み（Write）、「q」は終了（Quit）の略です。

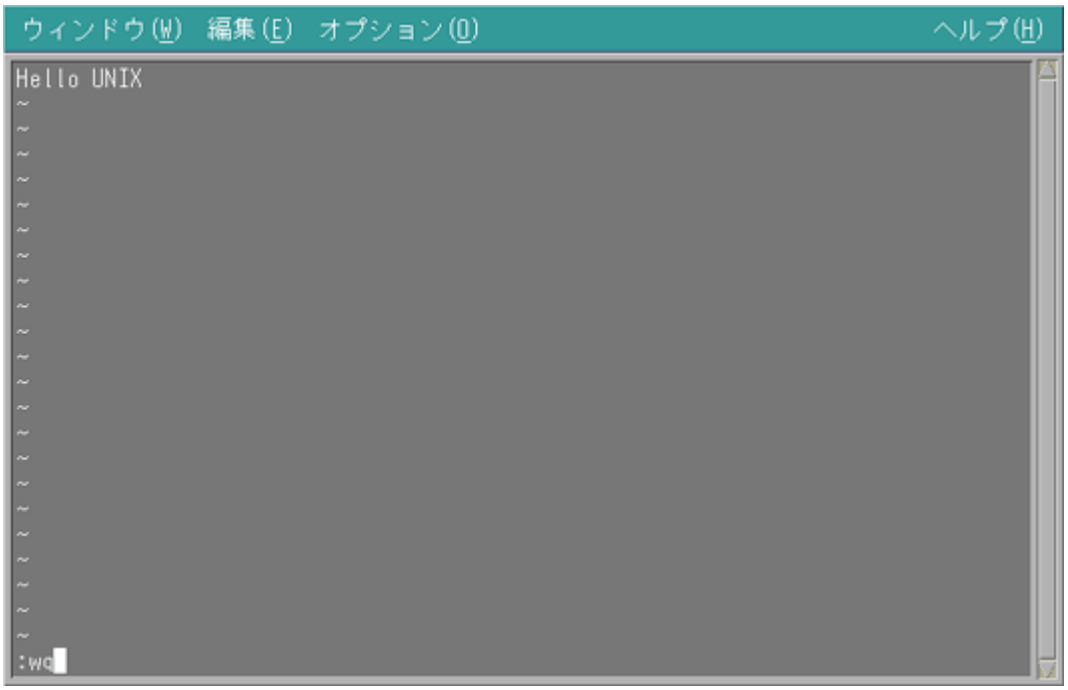


図 6：「:」に続けて「wq」とタイプするとファイルに書き込み・終了

変更内容をファイルに書き込まずに終了する

なお、バッファの内容を破棄して vi を終了するには「:q」コマンドを実行します。このときファイルが編集されていた場合には、誤って変更内容を破棄するのを防ぐため終了できないようになっています。強制的に終了するには最後に「!」を付けて、「:q!」とします。

マリー先生：
vi の前身は ex エディタという「ラインエディタ」なの。vi のコマンドモードで「:」をタイプすると、ex エディタのコマンドを実行する、いわゆる「ex モード」に移行するのね。ex モードでは「w」はファイルの書き込み、「q」は終了するためのコマンドなわけ。

四色君：
ラインエディタって何ですか？

マリー先生：
一般的なテキストエディタのように、画面全体にファイルの内容が表示されて、カーソルを動かしながら自由に編集できるテキストエディタを「スクリーンエディタ」と呼ぶの。ラインエディタは、スクリーンエディタが一般的になる前に使用されていたタイプのテキストエディタで、コマンドと数値で行を指定して、内容を表示させたり、置換や挿入をしたりといった編集を行うものよ。現在ではラインエディタそのものが使用されることはあまりないけど、vi は、ファイルの書き出しや、vi の終了、文字列の検索や置換といった処理に ex エディタのコマンドを引き継いでいるの。それから、「:wq」の代わりに「ZZ」コマンドを実行しても vi を終了できるわよ。

インサートモードに移行するためのコマンドたち

前述の例ではインサートモードに移行するのに「i」コマンドを使用しました。「i」コマンドは、現在のカーソル位置に文字を入力します。それ以外にも多くのインサートモードに移行するコマンドが用意されていますので、適宜使い分けるようにしましょう。

コマンド	インサートモードに移行したときの状態
i	カーソル位置に文字を入力する
a	カーソルの右側に文字を入力する
I	現在行の先頭に文字を入力する
A	現在行の末尾に文字を入力する
o	現在行と次の行の間に行を挿入する
O	現在行と前の行の間に行を挿入する

表 1：インサートモードに移行するためのコマンド

たとえば「o」コマンドは、現在行の下に新たな行を挿入した状態で、インサートモードに移行します。

```
line1[ ]
line2
↓　　「o」コマンド【Enter】
line1
[ ]
line2
※[]はカーソルです
```

四色君：
vi のコマンドは大文字/小文字を区別するのですか？

マリー先生：
するわよ。たとえば「o」コマンドと「O」コマンド（Shift キーを押しながら O キーを押す）は、どちらもインサートモードに移行するコマンドだけど、前者はカーソル位置の次の行に、後者は前の行に新しい行を挿入するの。

休み時間：CDE のテキストエディタ

CDE 環境には、Windows のメモ帳に似たシンプルなテキストエディタである「テキスト・エディタ」（dtpad）が用意されています。

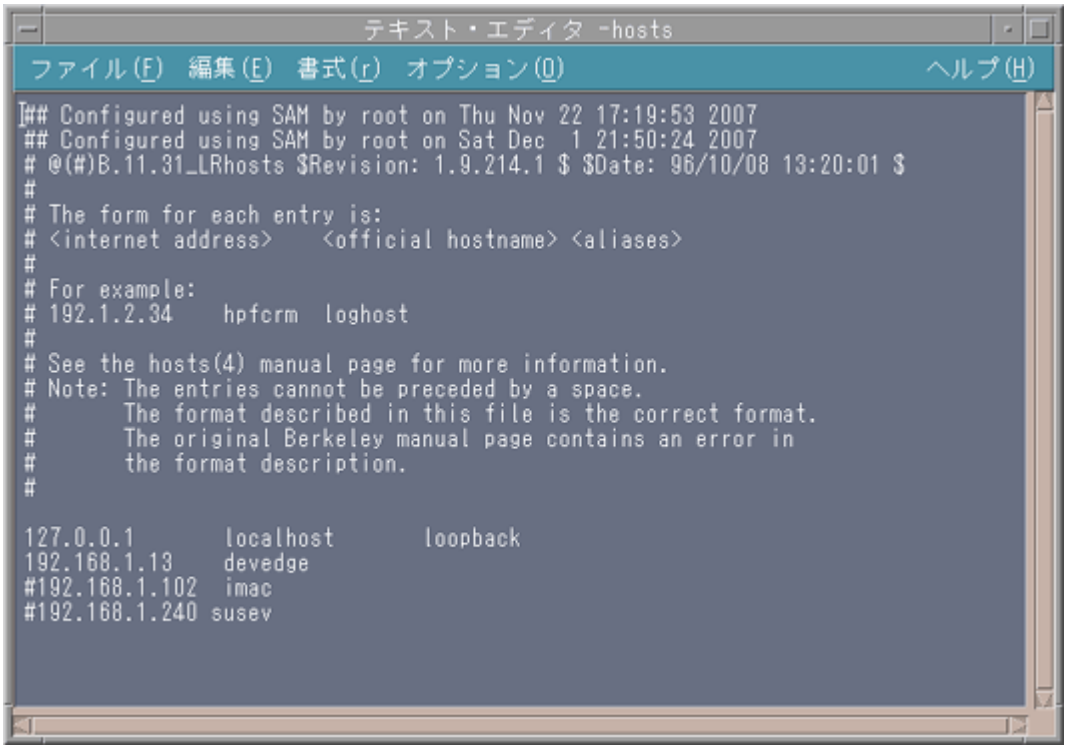


図 7：CDE で利用できる「テキスト・エディタ」

「テキスト・エディタ」は、パネルのアイコンをクリックして起動します。あるいは、ターミナルのコマンドラインで、次のように実行すると指定したパスのファイルを開くことができます。

```
$ dtpad ファイルのパス
```

2 時間目：vi でテキストファイルを編集する

1 時間目の説明で、vi のインサートモードとコマンドモードの使い分けが理解できたと思います。この時間では、vi を使用して実際にテキストファイルを編集する方法について説明していきましょう。

カーソルの移動

ファイルの編集に欠かせない基本操作が、カーソルの 1 文字ずつの移動でしょう。コマンドモードでのカーソルの上下左右の移動は、「h」「j」「k」「l」キーに割り当てられています。

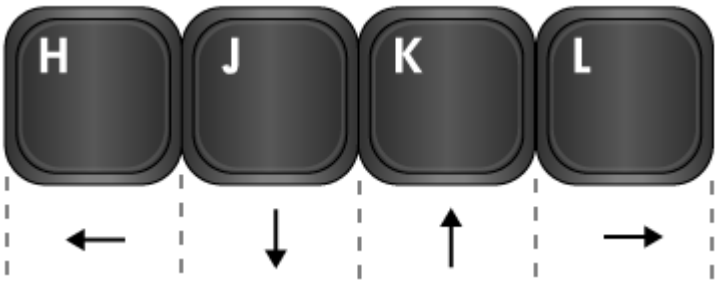


図 8：カーソル移動は「h」「j」「k」「l」キーに割り当てられている

四色君：
カーソルの移動に矢印キーは使えないのですか？

マリー先生：
たいていのシステムでは矢印キーでもカーソルが行えるように設定されているわ。ただし、「h」「j」「k」「l」の 4 つのキーは右手のホームポジション付近に並んで配置されているので、カーソルを素早く移動することが可能なの。かならず覚えるようにしましょう。

カーソル移動コマンドは、もっとも頻繁に使用するコマンドです。そのため、「h」「j」「k」「l」キーといった 1 文字ずつの移動以外にも、目的の場所に簡単に移動できるようにいろいろなキーが割り当てられています。たとえば、「$」コマンドはカーソルを現在行の末尾に移動します。

```
Hel[l]o UNIX
        ↓       「$」コマンド【Enter】
Hello UNI[X]
```

※[]はカーソルです
また、「^」コマンドはカーソルを行頭に移動します。

```
Hello UNI[X]
        ↓       「^」コマンド【Enter】
[H]ello UNIX
```

※[]はカーソルです

次の表によく使うカーソル移動コマンドをまとめておきます。

コマンド	カーソルの動作
h	カーソルを 1 文字左に移動
L、スペース	カーソルを 1 文字右に移動
j	カーソルを 1 文字下に移動
k	カーソルを 1 文字上に移動
^	カーソルを行頭に移動
$	カーソルを行末に移動
w	カーソルを次の単語の先頭に移動
e	現在カーソルがある単語の末尾に移動。カーソルが単語の末尾にある場合は、次の単語の末尾に移動
b	現在カーソルがある単語の先頭に移動。カーソルが単語の先頭にある場合は、前の単語の先頭に移動
H	カーソルを画面の一番上の行の先頭に移動
M	カーソルを画面の中央の行の先頭に移動
L	カーソルを画面の最下行の先頭に移動
G	バッファの最終行に移動

表 2：さまざまなカーソル移動コマンド

回数を指定したカーソルの移動

編集コマンドの中には、前に数値を指定すると処理を実行する回数を指定できるものがあります。表 2 に挙げたカーソル移動コマンドの多くも、数値を指定できます。たとえば「w」は次の単語の先頭に移動するコマンドですが、先頭に「3」を付けて「3w」とすると、3 つ後の単語の先頭に移動します。

```
U[N]IX is a computer operating system
        ↓       「w」コマンド【Enter】
UNIX [i]s a computer operating system          ← 次の単語の先頭に移動
        ↓       「3w」コマンド
UNIX is a computer [o]perating system          ←3 つ後の単語の先頭に移動
```

※[]はカーソルです

四色君：
たとえば次のような、「HP-UX」の先頭「H」にカーソルがあるときに「w」コマンドを実行すると、「-」にカーソルが移動してしまうのですが……。

[H]P-UX Developer Edge
↓　　「w」コマンド【Enter】
HP[-]UX Developer Edge
※[]はカーソルです

マリー先生：
そうね。「w」コマンドは句読点やハイフンなどを別の単語とみなすの。句読点などを単語の一部として移動するには、「W」コマンド（Shift キーを押しながら W キーを押す）が使えるわ。

[H]P-UX Developer Edge
↓　　「W」コマンド【Enter】
HP-UX [D]eveloper Edge
※[]はカーソルです

タックス君：
指定した行番号の行に移動することはできますか？

マリー先生：
「G」コマンドの前に行番号を数値で指定すれば可能よ。たとえば、「1G」は 1 行目に、「15G」は 15 行目に移動するの。

文字の削除

カーソルの移動と並んで頻繁に使用するコマンドが、文字の削除コマンドです。まず、カーソル位置の 1 文字を削除するには「x」コマンドを使用します。

HP-UX [D]eveloper Edge
　　↓　　「x」コマンド【Enter】
HP-UX [e]veloper Edge
※[]はカーソルです

「x」コマンドの前に数字を指定すると、カーソル位置からその数の文字を削除できます。たとえば「4x」では、カーソル位置から行の終わりに向かって 4 文字が削除されます。

HP-UX [e]veloper Edge

```
        ↓        「4x」コマンド【Enter】
HP-UX [o]per Edge
```
※[]はカーソルです

行単位の削除

現在行全体を削除するには、「dd」コマンド（D キーを 2 回押す）を使用します。

```
line1
li[n]e2
line3
        ↓        「dd」コマンド【Enter】
[l]ine1
line3
```
※[]はカーソルです

「dd」コマンドは、前に削除する行数を指定できます。たとえば「3dd」では、現在行からバッファの後方に向かって 3 行が削除されます。

```
line1
li[n]e2
line3
line4
line5
        ↓        「3dd」コマンド【Enter】
line1
[l]ine5
```
※[]はカーソルです

行末までを削除する

「D」コマンドを実行すると、カーソル位置から行末までが削除されます。

```
HP-UX [D]eveloper Edge
        ↓        「D」コマンド【Enter】
HP-UX[ ]
```
※[]はカーソルです

カーソル移動コマンドを組み合わせた削除

前述のカーソル移動コマンドと組み合わせて使用できる削除コマンドに、「d」コマンドがあります。「d」コマンドは次の形式で使用します。

```
d<カーソル移動コマンド>
```

「d」コマンドを実行すると、カーソル位置（もしくはカーソルのある行）から、カーソル移動コマンドの異動先までの範囲がすべて削除されます。たとえば、カーソルを 1 文字右に移動する「l」コマンドと組み合わせて「d3l」とすると、カーソル位置から 3 文字が削除されます。これは「3x」を実行した結果と同じです。

```
AB[C]DEFG
        ↓       「d3l」コマンド【Enter】
AB[F]G
※[]はカーソルです
```

また「d2w」では、カーソル位置から 2 つ先の単語の先頭までが削除されます。

```
W[o]rd1 Word2 Word3 Word4
        ↓       「d2w」コマンド【Enter】
W[W]ord3 Word4
※[]はカーソルです
```

なお、「d」コマンドと行を移動するカーソル移動コマンドとを組み合わせると、行単位の削除になります。たとえば「d2j」では、カーソルのある行から 3 行が削除されます。

```
line1
lin[e]2
line3
line4
line5
        ↓       「d2j」コマンド【Enter】
[l]ine1
line5
※[]はカーソルです
```

また「dG」では、カーソルのある行からバッファの最後までが削除されます。

```
line1
line2
li[n]e3
line4
line5
↓       「dG」コマンド【Enter】
line1
[l]ine2
※[]はカーソルです
```

タックス君：
間違って削除してしまった場合は？

マリー先生：
そんな時は「u」コマンドを実行すれば元に戻るわ。想像が付くと思うけど「u」は Undo の略ね。

別のファイルを読み込む

「:r ファイルのパス」を実行すると、カーソル位置に別のファイルの内容を読み込むことができます。たとえばカレントディレクトリの下の「old.txt」を読み込むには「:r old.txt」コマンドを実行します。

四色君：
ターミナルで vi エディタを起動しているときに、ls コマンドでディレクトリの一覧を確認したいといった場合には、いったん、vi を終了しないとだめですか？

マリー先生：
実は、「:!UNIX コマンド」を実行すると、vi の中で UNIX のコマンドが実行できるの、たとえば、「:!ls /etc」とすると「/etc」ディレクトリの一覧が表示されるわ。

図 9：vi の中で UNIX のコマンドを実行

結果を確認したら【Enter】キーを押すと、もとの編集画面にもどるの。 別のテクニックとして、「:r!UNIX コマンド」を実行すると、現在行の次にその UNIX コマンドの実行結果を挿入できるの。たとえば、date は現在の日付時刻を表示するコマンドだけど、「:r!date【Enter】」とすると、次の行にその時点での日付時刻が挿入されるわけ。

タックス君：
大きなファイルを編集していると、今バッファのどのあたりにいるのかわからなくなることがあるのですが……。

マリー先生：
そんなときには、「Ctrl+g」コマンド（Ctrl キー押しながら G キーを押す）を実行すると、行番号、全体の行数、全体の何%の位置にいるかという情報が最下行に表示されるわよ。

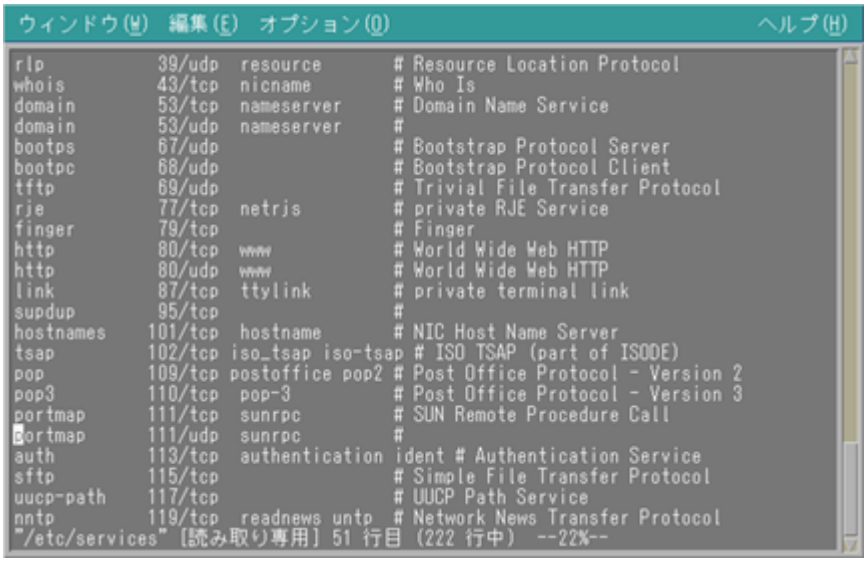


図 10：行番号、全体の行数、何%の位置にいるかを表示

練習問題

第 1 問：バッファの内容をファイルに書き込んで vi を終了するコマンドはどれか？

a)　:w
b)　:q
c)　:wq
d)　:q!

c)　:wq
コマンドモードで「:」をタイプすると ex モードになる。「:wq」の「w」はファイルに書き込むコマンド、「q」は vi を終了するコマンド。

第 2 問：インサートモードからコマンドモードに移行する方法として正しいのはどれか？

a)　「c」をタイプする
b)　esc キーを押す
c)　Ctrl キーを押す
d)　Alt キーを押す

b)　esc キーを押す
esc キーを押すとコマンドモードに移行する。コマンドモードにいるときに esc キーを押してもモードは変わらないため、モードがわからなくなってしまった場合には、esc キーを押してみればよい。

第 3 問：カーソル位置の文字を削除するコマンドはどれか？

a)　x
b)　d
c)　スペース
d)　a

a)　x
「x」をタイプするとカーソル位置の文字が削除される。「3x」のように「x」の前に数値を指定するとその数だけ文字が削除される。

第 4 問：カーソル位置の次の行に、カレントディレクトリのファイル「sample.txt」の中身を読み込むコマンドはどれか？

a)　:q sample.txt
b)　:r sample.txt
c)　:i sample.txt
d)　:r ../sample.txt

b)　:r sample.txt
ex コマンドである「:r ファイルのパス」を実行すると、現在行の次の行に指定したファイルの中身が読み込まれる。

UNIX の教科書「基礎編」

第 6 日目：リダイレクションとパイプを活用する

2008 年 4 月　大津 真

6 回に渡って連載してきた Linux の教科書も、今回で基礎編のひと区切りとし、次回からは新たに応用編が始まります。基礎編最終回となる今回は、UNIX のシェルを特徴づけている、リダイレクションとパイプという 2 つの基本機能を中心に解説しましょう。どちらも、シェルにおける柔軟なテキスト処理には欠かすことのできない機能です。

マリー先生：
今回は、コマンドラインを活用する上での必須の機能である「リダイレクション」と「パイプ」について説明しましょう。これらはちょっと高度な機能なのだけれど、うまく使うとコマンドラインが格段に便利になるのでがんばって理解してね。
その前に、前回の vi エディタの説明に関してなにか質問ありますか？

四色君：
vi エディタでテキストファイルを開いたときに、各行に行番号を表示することはできますか？

マリー先生：
できるわよ。ex モードのコマンドである「:set number」（または、「:set nu」でも可）を実行すれば、各行の左側に行番号が表示されるの。

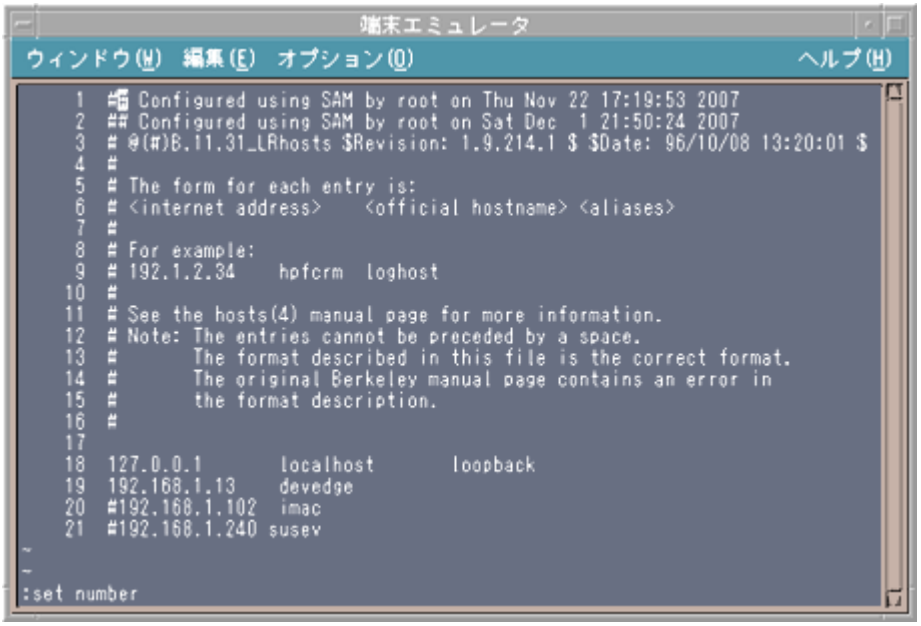
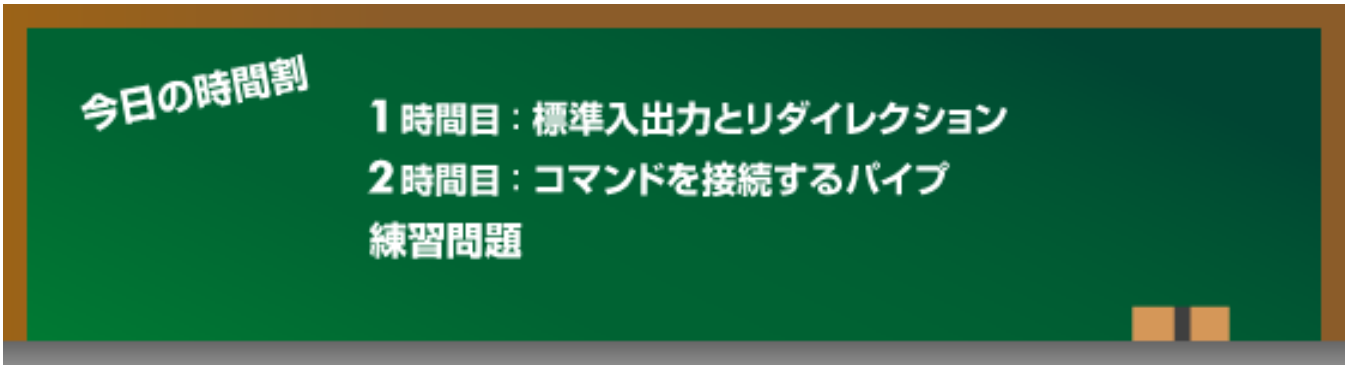


図 1：vi エディタで行番号を表示する

タックス君：
もとの状態に戻すには？

マリー先生：
行番号を消すには「:set nonumber」（または、「:set nonu」でも可）を実行してね。



1 時間目：標準入出力とリダイレクション

この時間では、まずシステムに用意されているコマンドのデフォルトの入出力先である標準出力と標準入力について説明します。その後で、標準入出力をファイルに割り当てるリダイレクションの使い方を紹介しましょう。

標準出力とは

シェル上で実行されるコマンドには、結果をターミナルの画面に表示するものが少なくありません。たとえば、date コマンドは現在の日付時刻を画面に表示します。

```
$ date
Wed Mar 26 21:53:09 JST 2008
```

このとき、date コマンドの内部では、画面という物理的なデバイスではなく、「標準出力」と呼ばれる UNIX システムにあらかじめ用意されている仮想的な出力デバイスに日付時刻を出力しています。初期状態では「標準出力」は画面に割り当てられているため、結果が画面に表示されるわけです。

図 2：date コマンドの結果は標準出力に出力されている

タックス君：
「標準出力」の「標準」とは？　意味がよくわからないのですが…。

マリー先生：
そうね。これは英語の「standard output」を直訳したものなの。デフォルトの出力先のようなイメージでとらえてね。参考までにちょっと詳しく説明しましょう。
UNIX のコマンドの多くは C 言語というプログラミング言語で作られているの。C 言語では結果を表示するのに printf()という命令（関数）を使うのだけど、この printf()というのは標準出力に文字列を表示する命令なの。言い換えると、プログラムの中で printf()を使って出力された文字列は標準出力に送られるのね。標準出力は初期状態で画面に設定されているため、結果として画面に実行結果が表示されるというわけ。

標準入力とは

デフォルトの出力先である標準出力と同じように、UNIX にはデフォルトの入力先として「標準入力」が用意されています。想像がつくと思いますが、標準入力は初期状態でキーボードに割り当てられています。
コマンドによっては、なんらかの入力を標準入力から受け取るものがあります。たとえば cat は、引数で指定したファイルの内容をそのまま表示するコマンドです。ただし、引数にファイルのパスを指定しない場合には、標準入力、つまりキーボードから入力を受け取りとるようになります。
実際に試してみましょう。キーボードから文字列をタイプすると、Enter キーを押すごとにそれをそのまま標準出力、つまり画面に表示します。終了するには「Ctrl + D」キーを押します。

```
$ cat
Hello【Enter】   ←文字列をタイプして Enter キーを押す
Hello            ←入力した文字列がそのまま表示される
HP-UX【Enter】   ←文字列をタイプして Enter キーを押す
HP-UX            ←入力した文字列がそのまま表示される
                 ←「Ctrl + D」キーを押すと終了
$
```

四色君：
「標準入力」は英語では「standard output」ということは、標準入力は「standard input」の略ですね。

マリー先生：
そうね。あと、エラーのデフォルトの出力先として「標準エラー出力（standard error）」というのも用意されているの。略して、標準出力は「stdout」、標準入力は「stdin」、標準エラー出力は「stderr」と表記されることもあるわ。

タックス君：
cat コマンドを終了するのに使った「Ctrl + D」はどういう意味があるのですか？

マリー先生：
標準的なターミナルの設定では「Ctrl + D」キーを押すと、ファイルの終わりを示す記号である「EOF（End Of File）」という特殊な文字コードが送られるの。ターミナルの設定内容を表示する「stty -a」コマンドを実行すると確認できるわよ。

```
$ stty -a
speed 9600 baud; line = 0;
rows = 44; columns = 163
min = 4; time = 0;
intr = ^C; quit = ^\; erase = ^H; kill = ^U
eof = ^D; eol ; eol2 ; swtch
stop = ^S; start = ^Q; susp = ^Z; dsusp = ^Y
werase = ^W; lnext = ^V
～以下略
```

実行結果の「eof = ^D」という部分が、EOF が「Ctrl + D」キーにアサインされていることを示しているの。

標準出力のリダイレクション

標準入出力の概要が理解できたところで、リダイレクションの説明に移りましょう。その名前から想像がつくかもしれませんが、リダイレクションとは標準入力もしくは標準出力を指定したファイルにリダイレクト、つまり切り替える機能です。
まずは、イメージしやすい標準出力のリダイレクションから説明しましょう。次に書式を示します。

```
コマンド > ファイルのパス
```

記号には「>」を使用する点に注目してください。出力のリダイレクションを使うことにより、コマンドの実行結果を格納するファイルを作成できます。たとえば、date コマンドの出力をカレントディレクトリの下にファイル「date.txt」を作成して書き込むには、次のようにします。

```
$ date > date.txt
$ cat date.txt    ←cat コマンドで確認
Wed Mar 26 22:18:34 JST 2008
```

イメージとしては図 3 のようになります。

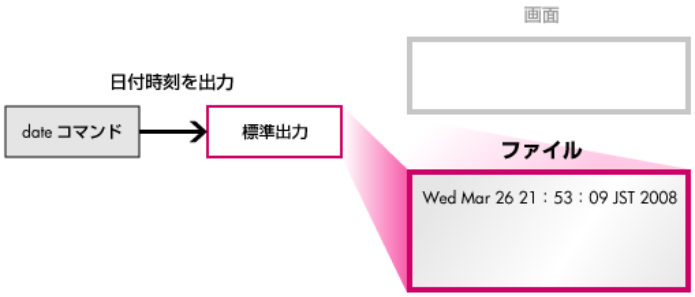


図 3：date コマンドの標準出力を画面からファイルに切り替える

別の例を示しましょう。cat コマンドの出力をファイルにリダイレクトすることによって、キーボードから入力した文字をファイルに書き込めます。短いテキストファイルを簡単に作成できるので、覚えておくと便利です。

```
$ cat > test.txt
Hello【Enter】　←文字列を入力
HP-UX【Enter】　←文字列を入力
                ←「Ctrl + D」キーを押すと終了
$ cat test.txt　←cat コマンドで確認
Hello
HP-UX
```

タックス君：
出力のリダイレクションを行うときに、リダイレクト先のファイルが存在していた場合はどうなりますか？

マリー先生：
その場合、何の警告もなく上書きされてしまうので注意してね。

追加のリダイレクション

前述のように出力リダイレクションの記号として「>」使用すると、ファイルが存在していた場合にはファイルが上書きされてしまいます。それに対して「>>」を使うと、ファイルの最後に追加されます。

```
コマンド >> ファイルのパス
```

たとえば、前項で作成した「date.txt」に今月のカレンダーを追加するには、cal コマンドの出力をリダイレクトして次のようにします。

```
$ cal >> date.txt
$ cat date.txt　←cat コマンドで確認
Wed Mar 26 22:18:34 JST 2008
   March 2008
 S  M Tu  W Th  F  S
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

四色君：
「>>」を使用する場合に、リダイレクト先のファイルがない場合にはどうなりますか？

マリー先生：
その場合は「>」と同じ動作になるの。つまり、ファイルが作成されてそこに書き込まれていくわけ。

標準入力のリダイレクション

標準出力と同じく、標準入力をファイルにリダイレクトすることができます。この場合、データは指定したファイルから読み込まれることになります。

```
コマンド < ファイルのパス
```

たとえば、次のようにすることで、ファイル「date.txt」の内容を行番号付きで画面に表示できます（「-n」は行番号を表示するオプション）。

```
$ cat -n < date.txt
     1  Wed Mar 26 22:18:34 JST 2008
     2     March 2008
     3    S  M Tu  W Th  F  S
     4                    1
     5    2  3  4  5  6  7  8
     6    9 10 11 12 13 14 15
     7   16 17 18 19 20 21 22
     8   23 24 25 26 27 28 29
     9   30 31
```

標準入力と標準出力のリダイレクションを組み合わせて使用することもできます。たとえば、次のようにすることで「date.txt」を「newfile.txt」にコピーすることができます。

```
$ cat < date.txt > newfile.txt
```

四色君：
cat コマンドは表示するファイルを引数に取るので、標準入力をリダイレクトしなくても同じことができますよね。

マリー先生：
そうね、「cat -n < date.txt」は「cat -n date.txt」、また、「cat < date.txt > newfile.txt」は「cat date.txt > newfile.txt」としても同じね。つまり、実際には cat コマンドの場合、標準入力のリダイレクションは使わなくてもいいわけ。
それと、標準出力と標準入力を同じファイルにリダイレクトしないように注意してね。

タックス君：
そうすると結果はどうなるのですか？

マリー先生：
たとえば「cat < date.txt > date.txt」のようにすると、最初にファイルの内容がクリアされて、「date.txt」が空になってしまうの。

文字を置換する tr コマンド

cat コマンドなどテキスト処理に関するコマンドの多くは引数にファイルのパスを指定できるため、標準入力をリダイレクトしなければならないケースはそれほど多くありません。ただし、コマンドによっては入力を標準入力からしか受け取らないものがあります。たとえば、文字を置換する tr コマンドがそのひとつです。

```
tr 置換前の文字 置換後の文字
```

この tr コマンドは、標準入力からデータを受け取り、結果を標準出力に送ります。処理したいファイルのパスを引数に取ることはできません。したがって、テキストファイル内の文字を置換するには標準入力をリダイレクトするか、あるいは後述するパイプを使う必要があります。
たとえば、ファイル「sample.txt」の中の「:」を「,」に置換して、画面に表示するには次のようにします。

```
$ cat name.txt              ←元のファイルを表示
makoto otsu:41
taro yamada:50
tomiko nekota:30
$ tr ":" "," < name.txt     ←「:」を「,」に置換
makoto otsu,41
taro yamada,50
tomiko nekota,30
```

休み時間：改行コードの OS による相違

テキストファイルの各行には、最後に 1 行の終わりを示す「改行コード」が埋め込まれています。この改行コードは OS によって標準で使用されるコードが異なります。

OS	改行コード
Windows	CRLF（キャリッジリターン＋ラインフィード）
Mac OS9 以前	CR（キャリッジリターン）
UNIX/Mac OSX 以降	LF（ラインフィード）

表 1：各 OS で標準とされる改行コード

CR（キャリッジリターン）と LF（ラインフィード）は、どちらも英文タイプライターに用意されているキーに由来します。CR でヘッドが初期位置に戻り、LF は 1 行分だけ用紙が送られます。
Windows や Mac OS の改行コードを UNIX 標準の LF に変換するには、tr コマンドが使用できます。たとえば、Windows の「CRLF」から UNIX の「LF」に変換するには、CR を削除します。
文字の削除は、次の書式で tr コマンドを実行します。

```
tr -d "文字"
```

また、CR は「\r」、LF は「\n」で表記できます。したがって、改行コードが CRLF の「win.txt」の、改行コード「CR」を削除し、「unix1.txt」に保存するには次のようにします。

```
$ tr -d "\r" < win.txt > unix1.txt
```

また、改行コードが「CR」の「mac.txt」の、改行コードを「LF」に置換し、「unix2.txt」に保存するには次のようにします。

```
$ tr "\r" "\n" < mac.txt > unix2.txt
```

2 時間目：コマンドを接続するパイプ

ここまでの説明で、標準入力と標準出力、およびそれらをファイルに切り替えるリダイレクションの概要がつかめたと思います。続いて、標準入出力を組み合わせてコマンドを接続する「パイプ」について説明しましょう。この連載を読み進めてきたみなさんは、UNIX のコマンドは、比較的シンプルな機能のものが多いということに気がついたことでしょう。それらのコマンドをパイプで組み合わせて使うことによって柔軟な処理が可能になります。

パイプの概要

「パイプ」とは、文字通り土管のようなイメージで、あるコマンドの標準出力と別のコマンドの標準入力に接続する機能です。

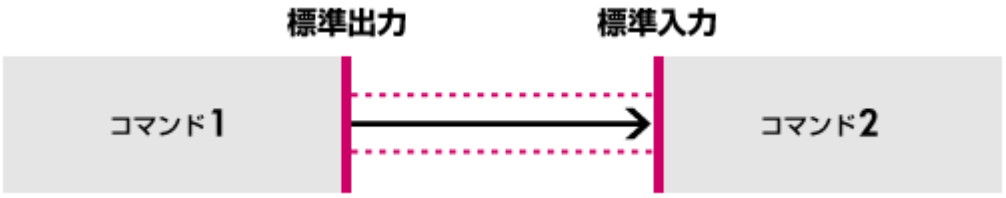


図 4：パイプはあるコマンドの標準出力を別のコマンドの標準入力に接続する

つまりパイプを使うと、あるコマンドの実行結果を別のコマンドで処理できるわけです。たとえば、「/etc/services」の先頭の 5 行を行番号付きで表示したいとしましょう。ここで、テキストファイルの先頭部分（デフォルトで 10 行）を表示するコマンドに head があります。

```
head -n 行数 ファイルのパス
```

ただし、head だけでは行番号を表示できません。そこで、行番号付きでファイルの内容を表示する「cat -n」コマンドから結果を受け取り、パイプを使用して「head -n 5」で先頭の 5 行を表示すればよいわけです。

```
$ cat -n /etc/services | head -n 5
 1  # @(#)B.11.31_LRservices $Revision: 1.32.214.7 $ $Date: 97/09/10 14:50:42 $
 2  #
 3  # This file associates official service names and aliases with
 4  # the port number and protocol the services use.
 5  #
```

四色君：
1 時間目に習った出力のリダイレクションを使えば、結果をファイルに書き出せることがわかりました。それでは、結果をファイルに書き出しつつ、画面に表示することはできないのですか？

マリー先生：
標準入力から受け取ったデータを、標準出力と引数で指定したファイルの両方に出力する tee というコマンドがあるの。したがって、コマンドの出力をパイプで tee コマンドに渡せば、ファイルと画面の両方に出力できるわ。たとえば、cal コマンド出力を、ファイル「cal.txt」に書き出しつつ、標準出力にも表示するには次のようにすればいいわけ。

```
$ cal | tee cal.txt
   March 2008
 S  M Tu  W Th  F  S
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

3 つ以上のコマンドをパイプで接続する

パイプで接続できるコマンドは 2 つだけではありません。3 つ以上のコマンドを組み合わせたより高度な処理も可能です。たとえば、「/etc/services」というファイルのの 96 行目から 100 行目までを行番号付きで表示したいとしましょう。まず、行番号を表示するのは「cat -n」で可能です。それでは、96 行目から 100 行目を表示するには、どうすればよいでしょう？それは、前述の head コマンドと、ファイルの最後の部分を表示する tail コマンドを組み合わせることによって実現できます。
次に示すように、tail コマンドの書式は head コマンドと同じです。

```
tail -n 行数 ファイルのパス
```

つまり、「head -n 100」で最初の 100 行を取り出し、パイプで「tail -n 5」に渡して最後の 5 行を表示します。全体としては、次のように実行します。

```
$ cat -n /etc/services | head -n 100 | tail -n 5
    96  uucp            540/tcp  uucpd           # uucp daemon
    97  dhcpv6-client    546/tcp                  # DHCPv6 Client
    98  dhcpv6-client    546/udp                  # DHCPv6 Client
    99  dhcpv6-server    547/tcp                  # DHCPv6 Server
   100  dhcpv6-server    547/udp                  # DHCPv6 Server
```

タックス君：
リダイレクションとパイプをいっしょに使うことはできるのですか？

マリー先生：
もちろん。たとえば、前の例の実行結果を「out.txt」に書き出すには次のようにすればいいわ。

```
$ cat -n /etc/services | head -n 100 | tail -n 5 > out.txt
```

フィルタコマンド

標準入力からデータを受け取り、その結果を標準出力に出力するコマンドのことを「フィルタコマンド」などと呼びます。つまり、フィルタコマンドは、パイプやリダイレクションでの使用を前提にしたコマンド群であるわけです。表 2 に基本的なフィルタコマンドの例を示します。

コマンド	説明
tr	文字を置換する
uniq	隣接する重複行を取り除く
wc	文字数や行数を数える
head	ファイルの先頭部分を表示する
tail	ファイルの最後の部分を表示する
cut	各行のフィールドを抜き出して表示する
sort	ファイルの各行をソートする

表 2：フィルタコマンドの例

フィルタコマンドをうまく活用すること、柔軟な処理が可能になります。いくつか例を示しましょう。
たとえば wc は、テキストファイルの行数、単語数、文字数などをカウントするコマンドです。「-w」オプションを指定すると単語数を表示します。おなじみの ls コマンドの出力をパイプ「|」を介して「wc -w」コマンドに渡すことによって、指定したディレクトリ以下のファイル数（ディレクトリを含む）をカウントできます。たとえばルート「/」ディレクトリ以下のファイル数をカウントするには次のようにします。

```
$ ls / | wc -w
16
```

また、カレントディレクトリの下にある、拡張子が「.txt」のファイルの数を表示するには、ワイルドカード「*」を使用して次のようにします。

```
$ ls *.txt | wc -w
11
```

マリー先生：
「wc -w」は単語数だけど、「wc -l」は行数をカウントするの。前の例の「ls *.txt | wc -w」は「ls *.txt | wc -l」としても同じよ。

```
$ ls *.txt | wc -l
11
```

タックス君：
あれ、でも「ls *.txt」を実行すると結果は横に並びますよね。

```
$ ls *.txt
A.txt     AKK.txt    backup.txt   mac.txt    profile.txt  unix.txt
AB.txt    C01.txt    cal.txt      out.txt    sample.txt   win.txt
```

「wc -l」で行数をカウントすると 2 行になるんじゃないですか？

マリー先生：
実は ls コマンドは、結果を画面に表示する場合にはわかりやすくするために横に並べて表示するのだけど、パイプで別のコマンドに渡す場合には 1 行にひとつずつ渡されるの。cat コマンドにパイプで渡してみるとわかるわよ。

```
$ ls *.txt | cat
AB.txt
AKK.txt
C01.txt
backup.txt
cal.txt
mac.txt
out.txt
profile.txt
sample.txt
unix.txt
win.txt
```

別の例を示しましょう。ファイル内のユニークな行を見つける、つまり重複する行を取り除くコマンドに uniq があります。
uniq コマンドで重複する行を取り除くためには、それらの行が隣接している必要があります。中身が同じ行でも、離れた場所にある行は取り除けません。
たとえば、次のようなファイル「sample.txt」があるとします。

```
AAA
BBB
BBB
CCC
AAA
CCC
```

このファイルを引数に、uniq コマンドを実行すると次のような結果になります。

```
$ uniq sample.txt
AAA
BBB
CCC
AAA
CCC
```

隣接した重複行である「BBB」のみが取り除かれ、離れた位置の重複行である「AAA」と「CCC」は取り除くことはできません。この場合ファイルの行を並び替える sort コマンドを実行し、その結果をパイプ「|」で uniq コマンドに渡すと、離れた場所にある重複行も取り除くことができます。

```
$ sort sample.txt | uniq
AAA
BBB
CCC
```

四色君：
どのくらい重複しているかを調べることはできるのですか？

マリー先生：
uniq コマンドを「-c」オプションを指定して実行すれば、重複回数が各行の左に表示されるわ。

```
$ sort sample.txt | uniq -c
  2 AAA
  2 BBB
  2 CCC
```

練習問題

第 1 問：新たにテキストファイル「hp.txt」を作成し、文字列「hp-ux」を書き込むコマンドはどれか？

a)　echo "hp-ux" > hp.txt
b)　echo "hp-ux" hp.txt
c)　cat "hp-ux" > hp.txt
d)　echo "hp-ux" < hp.txt

a)　echo "hp-ux" > hp.txt
echo は引数をそのまま標準出力に出力するコマンド。標準出力のリダイレクション「>」を使用してファイルにリダイレクトすることにより、ファイルに文字列を書き込める。

第 2 問：カレントディレクトリの下の「orig.txt」を「new.txt」にコピーするコマンドとして正しくないのはどれか？

a)　cp orig.txt new.txt
b)　cat orig.txt > new.txt
c)　cat < orig.txt > new.txt
d)　cat < orig.txt new.txt

d) cat < orig.txt new.txt
cat コマンドでファイルをコピーするには、引数に入力ファイルのパスを指定し、標準出力のリダイレクション「>」を使って出力ファイルを指定する。あるいは、入力ファイルを標準入力のリダイレクション「<」を使用して指定してもよい。

第 3 問：Pictures ディレクトリの下にある拡張子が「.png」のファイルの数を表示するコマンドはどれか？

a)　ls Pictures/*.png | number
b)　ls Pictures/*.png | wc -w
c)　ls Pictures/*.png > wc.txt
d)　ls Pictures/*.png < wc -w

b)　ls Pictures/*.png | wc -w
「wc -l」は単語をカウントするコマンド。したがって「ls Pictures/*.png」の出力をパイプで「wc -l」に渡せば、拡張子が「.png」のファイル数が求められる。

第 4 問：ファイル「sample.txt」の重複行を削除し、行番号付きで画面に表示するコマンドはどれか？

a)　sort sample.txt | uniq
b)　uniq | cat -n

c)　sort sample.txt | cat
d)　sort sample.txt | uniq | cat -n

d)　sort sample.txt | uniq | cat -n
uniq コマンドで重複行を削除するためには、あらかじめソートしておく必要がある。

UNIX の教科書「基礎編」

期末試験

2008 年 5 月　大津 真

Windows や Linux の経験がある方を対象に、初歩の初歩から HP-UX について解説してきた「UNIX の教科書　基礎編」全 6 回、いかがでしたか？内容はご理解いただけたでしょうか。今回は、「期末試験：基礎編」として全 10 問の試験を用意しました。いずれもこれまでの連載からの出題ですので、ぜひ挑戦してみてください。間違えてしまった場合は正解を参考に復習し、応用編を始める前に完璧にしておきましょう。

期末試験：基礎編

マリー先生：
これまでの授業を理解できたかのおさらいを兼ね、今日は期末試験を行います。10 問全て今まで学習してきたことなので、満点を目指してくださいね。

第 1 問：/etc ディレクトリの下のファイルの一覧を、ファイルサイズや更新日時などの情報とともの表示するコマンドはどれか?

a)　cd /ect
b)　ls –l /etc
c)　ls –a /etc
d)　ls /etc

b)　ls –l /etc
ls は「List Directory」の略で、指定したディレクトリの一覧を表示するコマンド。ls コマンドに「-l」オプションを指定すると詳細情報を表示する。

第 2 問：/etc/services ファイルの中身を 1 画面ずつ閲覧するにはどのコマンドを実行すればよいか?

a)　display /etc/services
b)　page /etc/services
c)　cat –n/etc/services
d)　more /etc/services

d)　more /etc/services
テキストファイルを 1 画面づつ表示するようなプログラムを「ページャ」と呼ぶ。More は代表的なページャ。スペースキーで次の画面に、「b」をタイプすると 1 つ前の画面に戻る。終了するには「q」をタイプする。

第 3 問：任意のディレクトリにいるときにホームディレクトリの下の Documents ディレクトリに移動するコマンドはどれか?

a)　cd ~
b)　cd ../..
c)　cd ~/Documents
d)　cd Documents

c)　cd ~/Documents
「~」はユーザのホームディレクトリを表す。したがってホームディレクトリの下の Documents ディレクトリは「~/Documents」と表記できる。

第 4 問：現在ユーザ「taro」のホームディレクトリ「/home/taro」にいるとき、 /etc ディレクトリの一覧を表示するコマンドとして正しくないのはどれか?

a)　ls /etc
b)　ls etc
c)　ls ../../etc
d)　ls ./../../etc

b)　ls etc
「..」はひとつ上のディレクトリを、「.」はカレントディレクトリを表す。したがって現在ホームディレクトリにいるとき「../..」および「./../..」はルート「/」ディレクトリを表すので、(c)と(d)はどちらも(a)と同じ結果になる。(b)は相対パスで指定しているから、ホームディレクトリの下の etc ディレクトリの一覧を表示しようとしているから間違い。

第 5 問：カレントディレクトリの下の Sites ディレクトリ下にある拡張子が「.html」のすべてのファイルを、カレントディレクトリの下の backup ディレクトリにコピーするコマンドはどれか?

a)　cp Sites backup
b)　mv Sites backup
c)　cp Sites/*.html backup
d)　cp sites/???.html backup

c)　cp Sites/*.html backup
ファイルのコピーには cp コマンドを使用する。「*」は 0 個以上の任意の文字列を表すワイルドカード。そのため、Sites ディレクトリ下の拡張子が「.html」のファイルを表すには、「Sites/*.html」と指定する。なお、ワイルドカードはシェルによって展開されコマンドに渡される。

第 6 問：カレントディレクトリの下の Readme.txt ファイルのシンボリックリンクを、カレントディレクトリの下の test ディレクトリに link.txt という名前で作成するコマンドはどれか?

a)　ln -s ../Readme.txt test/link.txt
b)　ln -s Readme.txt test/link.txt
c)　ln ../Readme.txt test/link.txt
d)　ln Readme.txt test/link.txt

a)　ln -s ../Readme.txt test/link.txt
シンボリックリンクを作成するには ln コマンドに「-s」オプションを指定する。このとき、リンク元のパスを相対パスで指定する場合、リンク先を起点にした相対パスで指定する必要がある点に注意。

第 7 問：vi エディタでファイルを編集中に、現在行からバッファの最後までを削除するコマンドはどれか?

a)　dd
b)　5D
c)　G
d)　dG

d)　dG
「d」コマンドの後ろにカーソル移動コマンドを指定すると、カーソル位置からそのコマンドによる移動先までの範囲が削除される。「G」コマンドはバッファの最後に移動するコマンドなので「dG」で現在行かバッファの最後までが削除される。

第 8 問：vi エディタでファイルを編集中に、カレントディレクトリの一覧を表示するコマンドはどれか?

a)　!ls
b)　:!ls
c)　ls
d)　:ls

b)　:!ls
コマンドモードで「:」でタイプすると ex モードに移行する。続けて「!シェルコマンド【Enter】」をタイプすると、シェルのコマンドを実行できる。

第 9 問：カレントディレクトリの下の Documents ディレクトリの下にある、拡張子が「.txt」と「.html」のファイルの合計を求めるコマンドはどれか?

a)　ls Documents/*.txt *.html | wc -l
b)　ls Documents/*.txt Documents/*.html > wc -l
c)　ls Documents/*.txt Documents/*.html | wc -l
d)　ls Documents/*.txt Documents/*.html < wc -l

c)　ls Documents/*.txt Documents/*.html | wc -l
パイプ「|」を使用するとコマンド実行結果を別のコマンドで処理することができる。「wc -l」は行数を表示するコマンドである。したがって、ls コマンドの結果をパイプで「wc -l」に渡せばファイル数を求めることができる。

第 10 問：テキストファイル「sample.txt」内の文字「,」をすべて「:」に置換し、新たに new.txt ファイルに保存するコマンドとして、正しくないのはどれか?

a)　tr "," ":" < sample.txt > new.txt
b)　tr "," ":" sample.txt > new.txt
c)　cat sample.txt | tr "," ":" > new.txt
d)　tr "," ":" < sample.txt | cat > new.txt

b)　tr "," ":" sample.txt > new.txt
文字を置換する tr コマンドは、引数にファイルのパスを指定できないから、標準入力からデータを受け取る必要がある。
そのため、ファイルからデータを読み込むためには、cat コマンドでファイルを読み込み結果をパイプ「|」で渡すか、入力リダイレクション「<」を使用する。また結果をファイルに保存するには、出力リダイレクション「>」を使用する。

≪次回までひと休み≫シェルのコマンド置換機能

標準出力と標準入力を接続することによって、あるコマンドの結果を別のコマンドで処理するパイプ「|」については、6 日目「リダイレクションとパイプを活用する」で説明しました。

```
コマンド 1 | コマンド 2
```

パイプと同じように、複数のコマンドを組み合わせることを可能にするシェルの機能に「コマンド置換」があります。コマンド置換とは、コマンド全体をバッククォーテーション「`」で囲むことによって、その実行結果に置き換える機能です。

```
`コマンド`
```

これを使用することで、コマンドの実行結果を別のコマンドの引数の一部として使うことができます。たとえば、次の例は「Now:」の後に現在の日付時刻を表示します。

```
$ echo "Now:" `date`【Enter】
Now: Tue Apr 29 13:44:32 JST 2008
```

echo は引数で指定した文字列を画面に表示するコマンドです。このように引数に「`date`」を指定することによって、その部分が date コマンドの実行結果に置き換わり、結果として「Now: 現在の日付時刻」という文字列が表示されるわけです。

```
echo "Now:" `date`【Enter】
↓`date`が date コマンドの実行結果に置き換わる
echo "Now:" "Tue Apr 29 13:44:32 JST 2008"
```

別の例を示しましょう。引数で指定したコマンドの絶対パスを表示するコマンドに which があります。たとえば cp コマンドの絶対パスを表示するには次のようにします。

```
$ which cp【Enter】
/usr/bin/cp
```

which コマンドにコマンド置換を使用して、cp コマンドの絶対パスの詳細情報を「ls -l」コマンドで表示するには、次のようにします。

```
$ ls -l `which cp`【Enter】
-r-xr-xr-x   1 bin       bin         79644 Feb 16  2007 /usr/bin/cp
```

なお、バッククォーテーション「`」は、シングルクォーテーション「'」と間違えやすいので注意してください。

UNIX の教科書「応用編」

―目次―

第 7 日目：シェルスクリプトでより便利に

7 回にわたってお送りしてきた応用編も、最終回となりました。今月はシェルを利用したプログラムであるシェルスクリプトについて学びます。プログラムといっても難しく考える必要はありません。まずは、よく使う処理をファイルに保存しておいてコマンドとして呼び出せるようにしてみましょう。シンプルなシェルスクリプトの書き方から始めて、今日の授業の終わりにはファイルの拡張子をまとめて変換するシェルスクリプトを作成できるようになります。(2009 年 4 月)

期末試験

シェルのより高度な利用方法を解説してきた「UNIX の教科書　応用編」(全 7 回)、いかがでしたか？内容はご理解いただけたでしょうか。今回は、「期末試験」として全 10 問の試験を用意しました。いずれもこれまでの連載からの出題ですので、ぜひ挑戦してみてください。 (2009 年 5 月)

UNIX の教科書「応用編」

第 1 日目：grep コマンドと正規表現

2008 年 8 月　大津 真

いよいよ「UNIX の教科書」応用編が始まります。シェルのより高度な利用方法に入っていきますが、基礎編で学んだことを踏まえて着実に一歩ずつ進んでいきましょう。初回となる今月は grep コマンドと、grep コマンドを使いこなすために必須となる正規表現を取り上げます。grep コマンドはテキストファイルから指定した文字列を含む行を取り出すのが基本的な使い方です。正規表現と組み合わせればさまざまなテキスト処理が可能になります。

マリー先生：
さあ、UNIX の教科書も今回から応用編ということで、ちょっと難しく感じるかもしれないけどがんばりましょう。みんな、基礎編の内容をもう一度復習しておいてね。それはそうと、期末試験の結果はどうだったかな?

タックス君：
僕は 90 点で合格でした!　第 6 問がどうしてもわからないのですが…。

マリー先生：
この問題ね。

第 6 問：カレントディレクトリの下の Readme.txt ファイルのシンボリックリンクを、カレントディレクトリの下の test ディレクトリに link.txt という名前で作成するコマンドはどれか?
a)　ln -s ../Readme.txt test/link.txt
b)　ln -s Readme.txt test/link.txt
c)　ln ../Readme.txt test/link.txt
d)　ln Readme.txt test/link.txt

タックス君：
カレントディレクトリの下のファイル「Readme.txt」のシンボリックリンクを作成するわけだから、答えは b だと思うのですが、不正解だったんです。

マリー先生：
勘違いしやすい点だけど、シンボリックリンクは単にリンク先のパスを格納しているだけなの。従って、もとのファイルを相対パスで指定する場合、カレントディレクトリではなくシンボリックリンクを起点にした相対パスで指定する必要があるわけ。つまり、「test/link.txt」を起点にして見ると「Readme.txt」のパスは「../Readme.txt」となるから、答えは a になるの。図 1 を見て理解してね。

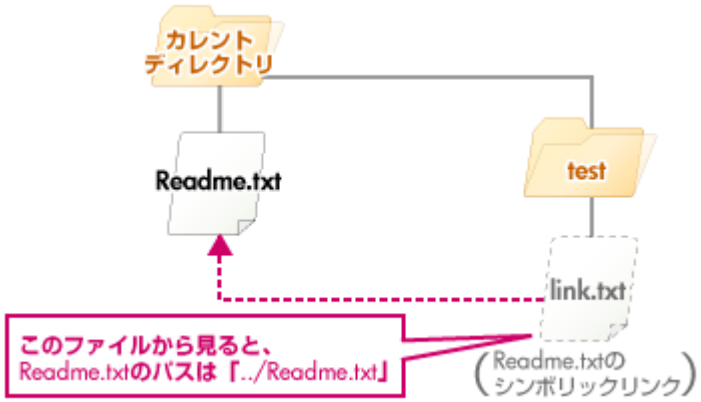


図 1：シンボリックファイルから見たリンク先のパスが格納される

四色君：
僕は、1 回目は 60 点でした。気を取りなしてもう 1 回チャレンジしたら、次はなんとか 80 点! 第 6 問の他に、第 10 問がわかりませんでした。

マリー先生：
第 10 問は、文字の置換に使用される tr コマンドの問題ね。

第 10 問：テキストファイル「sample.txt」内の文字「,」をすべて「:」に置換し、新たに new.txt ファイルに保存するコマンドとして、正しくないのはどれか?
a)　tr "," ":" < sample.txt > new.txt
b)　tr "," ":" sample.txt > new.txt
c)　cat sample.txt | tr "," ":" > new.txt
d)　tr "," ":" < sample.txt | cat > new.txt

tr はちょっと特殊なコマンドで、引数に処理するファイルのパスを指定できないの。だから標準入力のリダイレクションでファイルのデータを取りこむ（a と d のコマンド）か、パイプでデータを渡す（c のコマンド）必要があるわけ。したがって、「sample.txt」を引数にしている b が正しくないコマンドということになるのね。

1 時間目：grep コマンドの基本的な使い方

UNIX にはテキストを処理するコマンドが多数用意されています。基礎編でも、テキストファイルの中身を表示する cat や、文字を置換する tr コマンドなど、さまざまなテキストコマンドを使ってきました。今回はちょっと高度なテキスト処理コマンドとして、テキストファイルから指定した文字列を含む行を取り出す「grep」コマンドを紹介しましょう。なお、grep コマンドは「正規表現」と呼ばれる特別な表記法を使って目的の文字列を指定できます。正規表現については 2 時間目に説明することにして、まずこの時間では、grep コマンドの基本的な使い方について説明しましょう。

ファイルから指定した文字列を含む行を取り出す

grep コマンドを使用してファイルから指定した文字列を含む行を取り出す場合の、もっとも基本的な書式を示します。

```
grep 文字列 ファイルのパス
```

最初の引数に目的の文字列を、2 番目の引数に対象となるテキストファイルのパスを指定するわけです。

例として、次のような内容のテキストファイル「mail.txt」があるとしましょう。このファイルには、各行に 1 組ずつ、名前、メールアドレス、年齢がカンマ「,」で区切られて格納されています。

```
makoto otsu,mako@example.com,41
yamada taro,taroy@example.com,33
oyamada hana,oyama33@example.jp,44
kataoka ichiro,k-ichi@example.com,43
oka hanako,hana41@example.com,51
```

このファイルから文字列「makoto」を含む行をとりだすには、次のようにします。

```
$ grep makoto mail.txt 【Enter】
makoto otsu,makoto@example.com,41
```

また、文字列「33」を含む行を取り出すには次のようにします。

```
$ grep 33 mail.txt 【Enter】
yamada taro,taroy@example.com,33
oyamada hana,oyama33@example.jp,44
```

四色君：
「yamada taro」のような途中にスペースを含む文字列を指定するにはどうすればよいのですか?

マリー先生：
引数全体をダブルクォーテーション「"」（もしくはシングルクォーテーション「'」）で囲うの。具体的には、次のようになるわ。

```
$ grep "yamada taro" mail.txt 【Enter】
yamada taro,taroy@example.com,33
```

それから、引数の中に「*」や「?」といったワイルドカードを文字そのものとして含めたときにも、それがシェルによって展開されないようにダブルクォーテーション「"」で囲む必要があるの。いつも引数をダブルクォーテーション「"」で囲むように習慣づけておいてもいいわね。
ダブルクォーテーション「"」で囲むかわりに、スペースの前に「\」を付けてもいいわよ。

```
$ grep yamada\ taro mail.txt 【Enter】
yamada taro,taroy@example.com,33
```

タックス君：
あれ?「\」って、コマンドラインが複数行にわたるときにも使いましたよね?

マリー先生：
そうね。コマンドラインの行末で「\ 【Enter】」とすると、コマンドラインが次の行に継続することを表すの。

```
$ grep taro \
> mail.txt 【Enter】
yamada taro,taroy@example.com,33
```

それに対して、引数の途中で「\」を使った場合にはその後ろの特殊文字を、文字そのものとして扱うの。シェルにとってスペースは引数の区切り文字なのだけど、前に「\」を付けることによって「スペース」という文字として扱うわけ。このように「\」で後ろの特殊文字の機能を失わせることを「エスケープする」というの。
別の例として、ダブルクォーテーション「"」で囲まれた文字列の中で、ダブルクォーテーション「"」を文字そのものとして扱いたければ、その前に「\」を付けるの。

```
$ echo "hallo \"HP-UX\"" 【Enter】
hallo "HP-UX"
```

あるいは、「\」を使わずに全体をシングルクォーテーション「'」で囲ってもいいわよ。

```
$ echo 'hallo "HP-UX"' 【Enter】
hallo "HP-UX"
```

文字列を含まない行を取り出す

逆に grep コマンドを使用して、ファイルから指定した文字列を含まない行を取り出すには、次の形式で実行します。

```
grep -v 文字列 ファイルのパス
```

たとえば、mail.txt から「44」という文字列を含まない行を取り出すには次のようにします。

```
$ grep -v 44 mail.txt 【Enter】
makoto otsu,mako@example.com,41
yamada taro,taroy@example.com,33
kataoka ichiro,k-ichi@example.com,43
oka hanako,hana41@example.com,51
```

いずれかの文字列に一致する行を取り出す

引数に複数の文字列を指定して、そのいずれかに一致する行を取り出すこともできます。その場合、次の書式で実行します。

```
grep -e 文字列 1 -e 文字列 2 ... ファイルのパス
```

つまり、「-e 文字列」オプションを必要な数だけ指定すればよいわけです。たとえば、mail.txt から「41」と「44」のどちらかの文字列を含む行を取り出すには、次のようにします。

```
$ grep -e 41 -e 44 mail.txt 【Enter】
makoto otsu,makoto@example.com,41
oyamada hana,oyama33@example.jp,44
oka hanako,hana41@example.com
```

四色君：
最初の例の「grep makoto mail.txt」を、「grep -e makoto mail.txt」とすることもできるのですか?

マリー先生：
それでもかまわないわ。実際には、文字列が 1 つのときは「-e」を省略できるわけ。

タックス君：
行番号を表示するには、「cat -n」コマンドとパイプで組み合わせて次のようにすればよいのですね。

```
$ cat -n mail.txt | grep 41 【Enter】
1  makoto otsu,mako@example.com,41
5  oka hanako,hana41@example.com,51
```

マリー先生：
もちろんそれでもいいけど、grep コマンド自体にも行番号を表示する「-n」オプションがあるので、「cat -n」コマンドと組み合わせなくても行番号を表示できるわよ。

```
$ grep -n 41 mail.txt 【Enter】
1:makoto otsu,makoto@example.com,41
5:oka hanako,hana41@example.com
```

指定した複数の文字列を含む行を取り出す

複数の文字列を指定して、それらをすべて含む行を取り出したい場合にはどうすればよいでしょうか?　残念ながら、grep には指定したすべての文字を含む行を取り出すといったオプションは用意されていません。ただし、grep は引数にファイルを指定しないと入力を標準入力から読み込むように作られています。したがって、コマンドの標準出力と標準入力を結びつけるパイプ機能を使って、2 つの grep コマンドを結びつけてやればいいわけです。

たとえば、mail.txt から文字列「44」と「33」の両方を含む行を取り出すには次のようにします。

```
$ grep 44 mail.txt | grep 33 【Enter】
oyamada hana,oyama33@example.jp,44
```

タックス君：
複数のファイルから指定した文字列を含む行を取り出すことはできますか?

マリー先生：
grep コマンドの最後の引数にファイルのパスを並べて記述すれば OK よ。このとき結果をわかりやすくするために、先頭にファイル名を表示してくれるの。たとえば、mail.txt と mail2.txt から文字列「44」を含む行を取り出すには、次のようにするの。

```
$ grep 44 mail.txt mail2.txt 【Enter】
mail.txt:oyamada hana,oyama33@example.jp,44
mail2.txt:sakurai taro,ss10@example.com,44
mail2.txt:nakata saburo,nsabu1@example.com,44
```

休み時間：CDE の仮想デスクトップ

HP の標準デスクトップ環境である CDE には、初期状態で 4 つの仮想的なデスクトップ画面が用意されています。個々の仮想デスクトップのことを「ワークスペース」と呼びます。これらは、パネル中央の「1」「2」「3」「4」ボタンで切り替えることができます。

図 2：ワークスペースの切り替えボタン

なお、ワークススペースの数を増やすには、ボタンの上を右クリックし、表示されるメニューから「ワークスペースの追加」を選択します。
また、「1」「2」「3」「4」のワークスペース名を任意のワークスペース名に変更することもできます。

2 時間目：正規表現の基礎を知る

さて、1 時間目では grep コマンドを使用して、テキストファイルから指定した文字列を含む行を取り出す方法に説明しました。実は、grep コマンドの醍醐味は「正規表現」と呼ばれるパターンを使って、柔軟な文字列の指定ができることにあります。正規表現は複雑なパズルの世界に似て実に奥深いものですが、この時間では、その入門編として基本的な使い方について説明します。

正規表現とは

まずは、正規表現を使った簡単な例を示しましょう。1 時間目の例に使った、テキストファイル「mail.txt」を思い出してみましょう。

```
makoto otsu,mako@example.com,41
yamada taro,taroy@example.com,33
oyamada hana,oyama33@example.jp,44
kataoka ichiro,k-ichi@example.com,43
oka hanako,hana41@example.com,51
```

このファイルの各行の最初のフィールドには名前が格納されています。この中から「oka」という名前の人を取り出そうと、次のように実行したとします。

```
$ grep oka mail.txt 【Enter】
kataoka ichiro,kichi@example.com,43
oka hanako,hana41@example.com
```

結果として 2 行が表示されました。途中に「oka」を含む「kataoka」という名前も取り出されてしまったからです。
この場合、次のようにすると、行頭が「oka」で始まる行だけが表示されます。

```
$ grep "^oka" mail.txt 【Enter】
oka hanako,hana41@example.com
```

上記の「^oka」というのが正規表現のパターンです。想像が付くかもしれませんが、「^」は行の先頭を表す正規表現の「メタキャラクタ」（特殊文字）です。これで先頭が「oka」から始まる行だけが取り出されます。

タックス君：
「^」が行の先頭なら、行の終わりは?

マリー先生：
行の終わりは「$」で表すの。次の例を見てね。

```
$ grep "33" mail.txt 【Enter】 ←「33」を含む行を表示
yamada taro,taroy@example.com,33
oyamada hana,oyama33@example.jp,44
$ grep "33$" mail.txt 【Enter】 ←行の終わりが「33」の行を表示
yamada taro,taroy@example.com,33
```

任意の 1 文字を表すピリオド「.」

シェルのワイルドカードを思い出してみましょう。シェルのワイルドカードでは「?」が任意の 1 文字を表しました。それに対して、正規表現ではピリオド「.」が改行を除く任意の 1 文字を表します。次のようなファイル「sample1.txt」があるとします。

```
HP-UX v11
11i v3
1.3 V1
v11.0
11i v2
00va
```

このファイルから、たとえば行末が「v<任意の文字>」という文字列を含む行を取り出すには、パターンに「v.$」を指定して次のようにします。

```
$ grep "v.$" sample1.txt 【Enter】
11i v3
11i v2
00va
```

タックス君：
ピリオド「.」そのものを含む文字列を指定するにはどうすればよいのですか?

マリー先生：
その場合、ピリオド「.」の前に「\」を記述すると、ピリオド「.」を通常の文字として扱うの。たとえば、「.0」という文字列を含む行を取り出すには、次のようにすればいいわ。

```
$ grep "\.0" sample1.txt 【Enter】
v11.0
```

前にも説明したように、このように「\」によってピリオドのような特殊文字の働きを打ち消すことを「エスケープする」というの。

直前の文字の繰り返し

アスタリスク「*」は、直前の文字（もしくは直前の正規表現）の 0 回以上の繰り返しにマッチします。このとき、「1 回以上」ではなく「0 回以上」という点に注意してください。たとえば、次のようなテキストファイル「sample2.txt」があるとします。

```
HP-UX
HPP-UX
H-UX
```

パターンに「HP*-UX」を指定して grep コマンドを実行すると、すべての行が表示されます。最後の「H-UX」のように「P」を含まない行まで表示される点に注目してください。「P*」は空の文字列にもマッチするからです。

```
$ grep "HP*-UX" sample2.txt 【Enter】
HP-UX
HPP-UX
H-UX
```

四色君：
あれ？　シェルのワイルドカードでは、「*」は 0 文字以上の任意の文字列を表しましたよね。

マリー先生：
そうね。正規表現では、「*」の意味がシェルのワイルドカードと異なるので注意してね。正規表現では 0 文字以上の任意の文字列は「.」と「*」をつなげて「.*」で表すの。あと、「*」がシェルによって展開されないように引数をかならずダブルクォーテーション「"」で囲む点にも注意してね。

いずれかの文字を表す

「[ ]」内に文字を並べて記述すると、その中のいずれかの 1 文字とマッチします。このような指定方法を「文字クラス」などと呼びます。たとえば[ab]は「a」か「b」にマッチします。なお、[0123456789]のような連蔵した文字は、最初と最後の文字をハイフン「-」で繋いで[0-9]のように指定することもできます。
たとえば、sample1.txt から「行頭が 2 桁の数字で始まる行」は、次のようにして取り出せます。

```
$ grep "^[0-9][0-9]" sample1.txt 【Enter】
11i v3
11i v2
00va
```

このとき、行頭を表すの使用した「^」ですが、文字のリストの先頭に付けると、否定を表します。したがって、次の例は「行頭が数字以外で、2 文字目が数字の行」を表示します。

```
$ grep "^[^0-9][0-9]" sample1.txt 【Enter】
v11.0
```

名前付き文字クラス

文字クラスには、名前で文字を指定できる「名前付き文字クラス」と呼ばれる表記も用意されています。

＜表 1：名前付き文字クラスの例＞

[:alpha:]	文字
[:upper:]	大文字
[:lower:]	小文字
[:digit:]	10 進数の数字
[:alnum:]	文字あるいは 10 進数の数字
[:blank:]	空白文字

たとえば、数字は「0-9」の代わりに[:digit:]と表記できます。つまり、sample1.txt から、「行頭が 2 桁の数字で始まる行」を取り出す例は次のように記述できます。

```
$ grep "^[[:digit:]][[:digit:]]" sample1.txt 【Enter】
11i v3
11i v2
00va
```

grep ファミリー

grep には、「egrep」と「fgrep」という仲間のコマンドが用意されています。これらをまとめて grep ファミリーと呼びます。egrep は「Extend grep」の略で、拡張されたパターンが指定可能です。fgrep は「Fixed grep」の略で正規表現の使えない grep です。なお、これらはそれぞれ「grep -E」、「grep -F」を実行しても同じです。egrep および fgrep は将来サポートされなくなる可能性があるため、「grep -E」、「grep -F」の使用が推奨されています。

たとえば、「grep -E」で使用可能な拡張正規表現では、新たに「+」が特殊記号として使用できます。通常の正規上限では「*」は前の文字の 1 回以上の繰り返しでしたが、「+」は前の文字の 1 回以上の繰り返しを表します。
次に、「*」と「+」の相違を示す例を示します。

```
$ grep -E "HP*-UX" sample2.txt 【Enter】 ←「*」を使用
HP-UX
```

```
HPP-UX
H-UX ←「H」と「-」の間になにもなくてもマッチしてしまう。
$ grep -E "HP+-UX" sample2.txt 【Enter】 ←「+」を使用
HP-UX
HPP-UX
```

タックス君：
「grep -E」、つまり egrep が一番高機能なら、ほかの grep の仲間のいらないような気がするですが

マリー先生：
grep は機能が豊富で柔軟なパターンが記述できるのは、指定できるメタキャラクタが増えているからなの。それらの、メタキャラクタを覚えておかないと、間違ってメタキャラクタを普通の文字として使ってしまってしまう可能性もあるわけ。また、メタキャラクタを文字として扱うときにその都度エスケープするのも面倒でしょう。
たとえば、次のような「sample3.txt」から、「...*」とい文字列を含む行を取り出したいとしましょう。

```
abcde
a...*a
sample
```

この場合、grep および「grep -E」では、次のようにしてピリオド「.」をエスケープしなければならないの。

```
$ grep "\.\.\.\*" sample3.txt 【Enter】
a...*a
```

それに対して、「grep -F」、つまり frep では正規表現の特殊文字を気にしないで次のようにできるわけ。

```
$ grep -F "...*" sample3.txt 【Enter】
a...*a
```

練習問題

第 1 問：/etc/services ファイルから文字列「 23/」（*注 最初がスペース）を含む行を取り出すコマンドはどれか?

a)　grep 23/ /etc/services
b)　grep -i 23 /etc/serices
c)　get 23/ /etc/services
d)　grep " 23/" /etc/services

d)　grep " 23/" /etc/services
スペースを含む文字列を検索するには全体をダブルクォーテーション「"」で囲む。
また、次のコマンドでも同じ結果が得られる。
grep \ 23/ /etc/services

第 2 問：/etc/services ファイルから、文字列「echo」を含み、かつ、文字列「udp」を含まない行を取り出すコマンドはどれか

a)　grep -e echo -e udp /etc/services
b)　grep echo /etc/services | grep udp
c)　grep echo udp /etc/services
d)　grep echo /etc/services | grep -v udp

d)　grep echo /etc/services | grep -v udp
複数の条件で絞り込むにはパイプ「|」を使用して grep コマンドを組み合わせる。また、指定した文字列を含まない行を取り出すには「-v 文字列」オプションを指定する。

第 3 問：次のようなテキストファイル「sample4.txt」から、数字だけの行を表示するコマンドはどれか?

```
111 11
aa22
11bb
101aa2
2222
555
bb77
```

a)　grep "[[:digit:]][[:digit:]]*" sample4.txt
b)　grep "^1-9$" sample4.txt
c)　grep "^[[:digit:]][[:digit:]]*$" sample4.txt
d)　grep "^[[:digit:]]*$" sample4.txt

c)　grep "^[[:digit:]][[:digit:]]*$" sample4.txt

名前付き文字クラスでは数字は「[:digit:]」と記述する。また、文頭は「^」、文末は「$」、前の文字の 0 回以上の繰り返しは「*」と記述できる。したがって(c)が正解。(d)だと空行も出力されてしまう点に注意してほしい。
また、次の拡張機能を持った egrep コマンドでも同じ結果が得られる。
egrep "^[0-9]+$" sample4.txt

第 4 問：sample4.txt ファイルから、先頭が「1 から 4 の間の数字」で始まらない行を表示するコマンドはどれか?

a)　grep "^[^1-4]" sample4.txt
b)　grep "^[1-4]" sample4.txt
c)　grep "^[^[:digit:]]" sample4.txt
d)　grep -F "^[^1-4]" sample4.txt

a)　grep "^[^1-4]" sample4.txt
[文字の並び]の先頭に「^」を記述すると、否定を表す。つまり「[^1-4]」は「1 から 4 の間の数字」以外の文字にマッチする。

UNIX の教科書「応用編」

第 2 日目：ファイルの検索

2008 年 9 月　大津 真

応用編第 2 回となる今月は、find コマンドについて学びます。find はファイルやディレクトリを検索するコマンドで、さまざまな検索条件のオプションを付けることによって、複雑な検索をこなすことができます。単にファイルを検索するだけではなく、検索結果を他のコマンドに引数として渡す方法も覚えましょう。find と連携させることで、前回の grep をはじめ、これまで学んできたコマンドをより便利に使えるようになります。

マリー先生：
ディレクトリの階層が深くなってくると、ファイルをどこに保存したかわからなくなってしまった……なんてことはよくあるわよね。今回は、そんな時に便利なファイルの検索コマンドである「find」について説明しましょう。その前に、前回の grep コマンドと正規表現の説明に関してなにか質問ありますか？

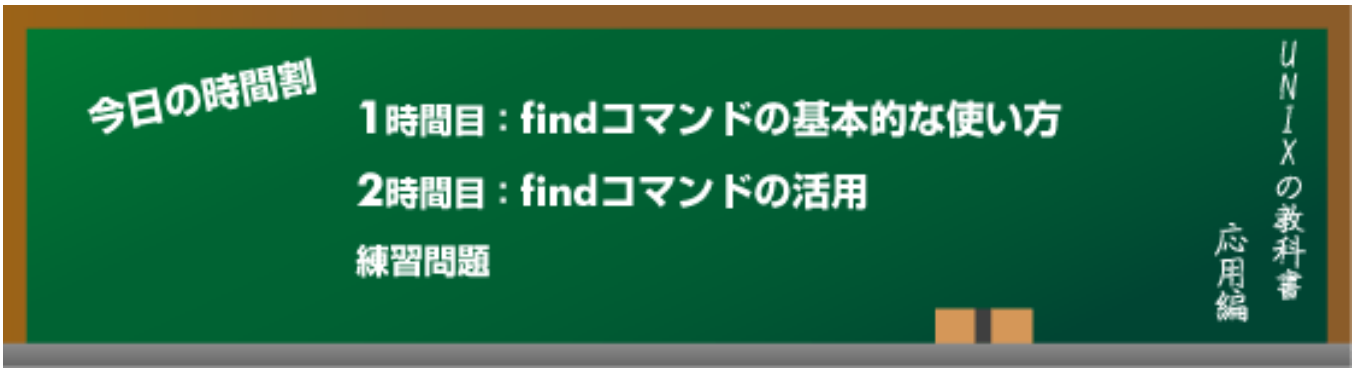
四色君：
正規表現が便利だということは理解できたのですが、パターンの作り方がいまいちピンとこなくて……。

マリー先生：
そうね、正規表現はすごく奥深い世界だから慣れるまでが大変ね。でも正規表現は grep だけでなく、Perl や Ruby といった Web で活躍する言語にも必須だし、高機能なテキストエディタにも搭載されているから、基本的なパターンは覚えておくとなにかと便利よ。

タックス君：
僕は正規表現をいろいろ実験している最中です。質問なのですが、テキストファイルから、文字のある行だけを表示することはできますか？

マリー先生：
それは、ちょっと難しいわね。「文字のある行だけを表示する」ということを「空行、スペースとタブだけの行以外の行を表示する」と言い換えて、コマンドを考えてみましょう。実は、スペースやタブといった表示するときに空白になる文字は「[:space:]」という名前付きクラスで表せるの。マッチしなかった行を表示するには「-v」オプションを指定すればいいから、全体としては次のようになるわね。

```
$ grep -v  "^[[:space:]]*$" sample.txt 【Enter】
HP-UX
find
grep
```



1 時間目：find コマンドの基本的な使い方

今回紹介する「find」コマンドは、UNIX の世界では定番の、ファイルやディレクトリの検索コマンドです。find コマンドには、単に検索条件に一致するファイルの一覧を表示するという使い方だけでなく、見つかったファイルに対して別のコマンドを実行するといった活用法があります。この時間では、まず基本的な使い方について説明しましょう。

名前で検索する

次に find コマンドの基本的な書式を示します。

```
find [オプション] 検索を開始するディレクトリ 検索条件
```

find コマンドを実行すると、引数で指定したディレクトリを辿りながら、検索条件に一致するファイルを表示していきます。もっともよく使用する検索条件は、指定した名前に一致するファイルやディレクトリを検索するというものでしょう。この場合、次のような書式で検索条件を指定します。

```
-name 文字列
```

たとえば、Documents ディレクトリ以下で、名前が「sample」のファイルもしくはディレクトリを検索するのは次のようにします。find コマンドは、検索条件に一致するファイルがひとつ見つかってもそこで終了しません。指定したディレクトリ以下の階層のディレクトリすべてが検索対象となり、見つかったファイルがすべて表示されます。また、なにも指定しないと、通常のファイルだけでなくディレクトリも検索対象となります。

```
$ find Documents -name sample 【Enter】
Documents/2008/sample
Documents/sample
```

四色君：
試しにルート「/」を起点に検索するとエラーがいっぱい表示されるのですが……。

マリー先生：
コマンドを実行するユーザに対して検索が許可されていないディレクトリを検索しようとするとエラーが表示されるの。そのようなディレクトリを検索するにはスーパーユーザの権限が必要よ。

ワイルドカードを使用した検索

初級編で解説した、「*」と「?」といったシェルのワイルドカードを覚えているでしょうか。

*	0 個以上の任意の文字列に一致する
?	1 文字と一致する

これらのワイルドカードは、find コマンドの名前による検索にも使えます。たとえば、カレントディレクトリを起点に、拡張子が「.txt」のファイルを検索するにはどうすればよいでしょう？まず、カレントディレクトリはピリオド「.」で、拡張子が「.txt」のファイルは「*.txt」というパターンで表せます。このとき、パターンがシェルによって展開されないように、ダブルクォーテーション「"」で囲って「"*.txt"」とします。
したがって、次のように実行すればよいわけです。

```
$ find . -name "*.txt"　【Enter】
./docs/mail.txt
./Documents/2007/ok.txt
./Documents/2007/ng.txt
./Documents/2007/readme.txt
./Documents/2008/sep.txt
./Documents/2008/magic.txt
./Documents/sample.txt
./Pictures/sample.txt
./sample.txt
```

タックス君：
あれ、こちらでは「*.txt」をダブルクォーテーション「"」で囲まなくても OK ですけど？

マリー先生：
ワイルドカードにマッチするファイルがないときはアスタリスク「*」がシェルによって展開されずに、find コマンドにそのまま渡るの。でも、カレントディレクトリにどんなファイルがあるかを気にしなくてすむから、ダブルクォーテーション「"」で囲む習慣を付けておいたほうがいいわよ。

四色君：
先月の grep コマンドで習った、[abc]や「0-9」といった文字クラスは使えるのですか？

マリー先生：
使えるわ。たとえば、名前が数字で始まって拡張子が「.txt」のファイルを、カレントディレクトリを起点に検索するには次のようにすればいいの。

```
$ find . -name "[0-9]*.txt" 【Enter】
./Documents/2008/9th.txt
./2008-3-1.txt
```

いろいろなオプション

find コマンドには、検索条件を設定するためのさまざまなオプションが用意されています。表 1 に代表的なオプションをまとめておきます。

＜表 1：find の代表的な検索オプション＞

検索オプション	説明
-atime n	n 日前にアクセスされたファイル。 n ではちょうど n 日、-n では n 日より後、+n では n 日より前になる。
-ctime n	n 日前にステータス情報が変更されたファイル。 n ではちょうど n 日、-n では n 日より後、+n では n 日より前になる。
-mtime n	n 日前に修正されたファイル。 n ではちょうど n 日、-n では n 日より後、+n では n 日より前になる。
-group <グループ名>	指定したグループに属するファイル。
-user <ユーザ名>	指定したオーナのファイル。
-newer <ファイルのパス>	指定したファイルより後に修正されたファイル。
-perm <パーミッション>	指定したパーミッションが設定されているファイル。
-size <サイズ>	指定したサイズのファイル。 サイズはブロック数（1 ブロックは 512 バイト）。「+」を指定した場合にはそれ以上、「-」の場合にはそれ未満。
-type <タイプ>	指定したタイプのファイル。 「b」はブロック型スペシャルファイル、「c」はキャラクタ型スペシャルファイル、「d」はディレクトリ、「f」は通常のファイル、「l」はシンボリックリンク。

検索条件を複数指定した場合には、それらのすべてに一致するファイルが検索されます。たとえば「-type d」はディレクトリを指定します。したがって、カレントディレクトリ以下で名前が 5 文字のディレクトリを検索するには、「-name "?????"」オプションと「-type d」オプションの 2 つを指定します。

```
$ find . -name "?????" -type d 【Enter】
./.dt/types
./.dt/icons
./.dt/Trash
```

修正日時による検索

「-mtime n」は修正日時による検索です。数字の前に「+」を指定した場合にはそれより前、「-」を指定した場合にはそれより後になります。

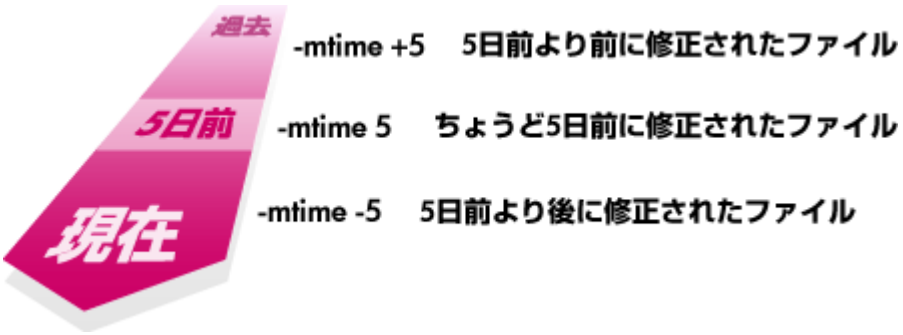


図 1：-mtime での検索結果

たとえば、Documents ディレクトリ以下で、5 日前より後に修正されたファイルを検索するには次のようにします。

```
$ find Documents -mtime -5 -type f 【Enter】
Documents/2007/ok.txt
Documents/2007/ng.txt
Documents/2007/readme.txt
Documents/2008/sep.txt
Documents/2008/magic.txt
Documents/2008/sample.txt
Documents/2008/sample
Documents/2008/9th.txt
Documents/sample.txt
Documents/sample.bak
```

四色君：
逆に、検索条件に一致しないファイルを見つけることはできるのですか？

マリー先生：
検索条件の前に否定を表す「!」を指定すれば可能よ。たとえば、Documents ディレクトリ以下の拡張子が「.bak」以外のファイルを見つけるには、次のようにすればいいの。このとき「!」の前後にはスペースが必要なので気をつけてね。

```
$ find Documents ! -name "*.bak" -type f 【Enter】
Documents/2007/ok.txt
～以下略～
```

休憩時間：Telnet/SSH クライアント PuTTY

読者の方の中には、Windows から HP-UX にリモートログインして操作を行っている方も少なくないでしょう。Windows で利用できるフリーの Telnet/SSH クライアントとしては UTF-8 TeraTerm Pro with TTSSH2 が代表的ですが、それ以外にも、「PuTTY このリンクをクリックすると、HP 社外へリンクします。」というターミナルソフトも人気です。PuTTY は、 Simon Tatham 氏によって開発されているオープンソースの Telnet/SSH クライアントです。SSH のプロトコルはバージョン 1 とバージョン 2 の両方に対応しています。PuTTY 日本語版は以下のサイトからダウンロードできます。

http://hp.vector.co.jp/authors/VA024651/PuTTYkj.html

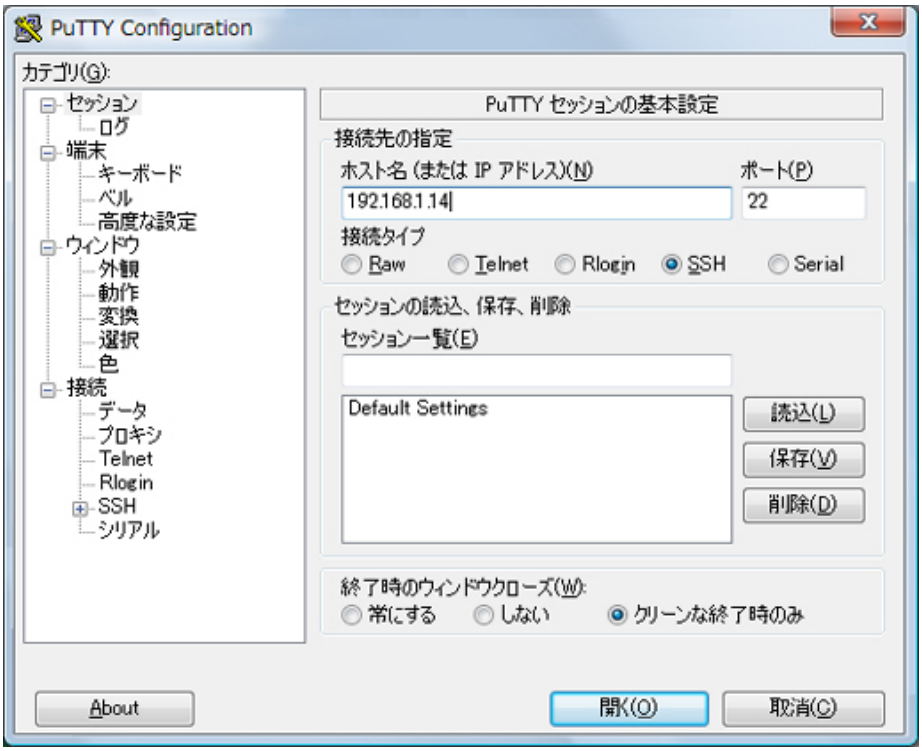


図 2：接続先の指定

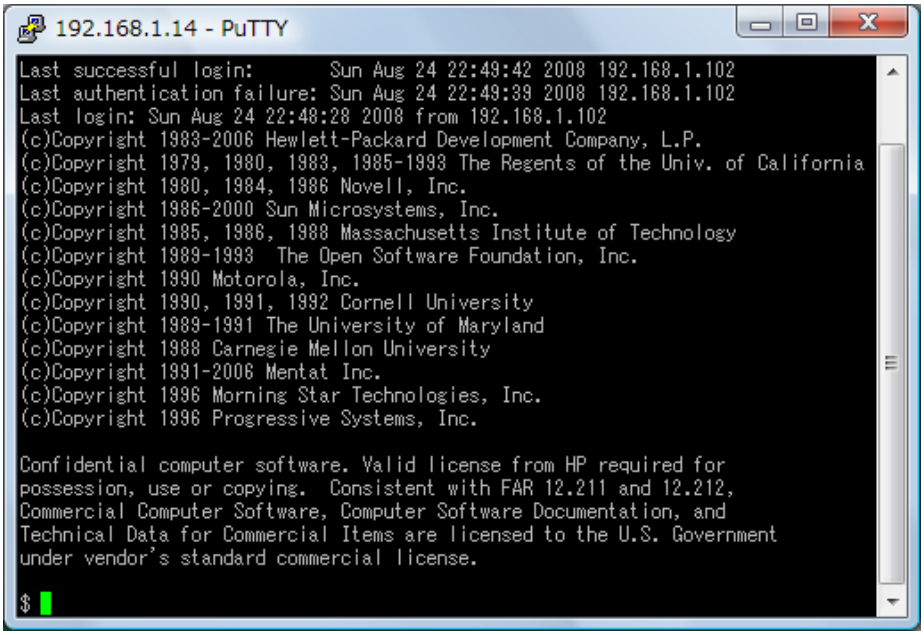


図 3：HP-UX にリモートログイン

2 時間目：find コマンドの活用

1 時間目の説明で find コマンドの基本的な使い方が理解できたと思います。2 時間目では、find コマンドを使いこなすためのテクニックをいくつか紹介していきましょう。

いずれかの条件に一致するファイルを検索する

前述のように、find コマンドに検索条件を複数記述するとすべての条件に一致するファイルが検索されます。そのような検索を AND 検索などと呼びます。そうではなく、いずれかの条件に一致するファイルを検索するという、いわゆる OR 検索を行う場合には、検索条件の間に「-o」オプションを記述します。たとえば、カレントディレクトリ以下で拡張子が「.txt」、もしくは「.html」のファイルを検索するには次のようにします。

```
$ find . -name "*.txt" -o -name "*.html" 【Enter】
./docs/mail.txt
./docs/abc.html
./Documents/2007/ok.txt
./Documents/2007/ng.txt
./Documents/2007/readme.txt
～以下略～
```

四色君：
それじゃ、もっと検索条件を複雑にして、拡張子が「.txt」もしくは「.html」で、かつ 3 日以内に更新されたファイルを見つけるといったことはできるのですか？

マリー先生：
算数の演算と同じように、( )で優先する条件を囲むことで可能よ。ただし、「()と「」」はシェルにとって特殊文字なので、その前には「\」を付けて通常の文字として扱う必要があるの。このように特殊文字の働きを失わせることを、“エスケープする”というので覚えておいてね。

```
$ find . \( -name "*.txt" -o -name "*.html" \) -mtime -3 【Enter】
./docs/abc.html
～以下略～
```

検索結果を他のコマンドで処理する

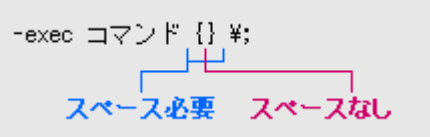
これまでの例では、find コマンドを実行すると検索されたファイルの一覧が画面に表示されましたが、これは検索結果を標準出力に表示する「-print」というオプションが省略されていたためです。
つまり、次の 2 つのコマンドは同じです。

```
$ find . -name "*.html"
$ find . -name "*.html" -print
```

コマンドを画面に表示する代わりに、別のコマンドに引数として渡すこともできます。それには、「-print」オプションの代わりに「-exec」オプションを次の書式で指定します。

```
-exec コマンド {} \;
```

想像が付くかと思いますが、「{}」が検索されたファイルのリストに置き換わるわけです。最後の「;」はコマンドの終わりを示しますが、シェルの特殊文字のため、その前に「\」を記述してエスケープしています。このとき、コマンドの直後、および「\」の前にはスペースが必要です。また「{}」の内部にはスペースを入れてはいけません。



たとえば、カレントディレクトリ以下の拡張子が「.html」のファイルを、「ls -l」コマンドで表示するには次のようにします。

```
$ find . -name "*.html" -exec ls -l {} \; 【Enter】
-rw-r-----  1 o2      users       6677 Aug 24 23:20 ./docs/abc.html
-rw-r-----  1 o2      users       6677 Aug 24 23:19 ./abc.html
-rw-r-----  1 o2      users       6677 Aug 24 22:54 ./Site/index.html
```

タックス君：
「-exec」オプションを使わないで、単順に、find コマンドの結果をパイプ「|」で「ls -l」コマンドに渡せばいいんじゃないですか？

マリー先生：
それはだめなの。なぜなら ls コマンドは標準入力からデータを受け取るタイプのコマンドではないからよ。実際に実行してみるとわかるけど、find コマンドの結果は破棄されて、単にカレントディレクトリの一覧が表示されるわ。

```
$ find . -name "*.html" | ls -l 【Enter】
total 80
-rw-r-----   1 o2     users       25 Aug 24 23:11 2008-3-1.txt
drwxr-x---   2 o2     users       96 Aug 24 23:01 3sum
drwxr-x---   5 o2     users     8192 Aug 24 23:14 Documents
drwxr-x---   5 o2     users     8192 Aug 24 22:54 Pictures
drwxr-x---   2 o2     users       96 Aug 24 22:51 Sample
```

マリー先生：
find コマンドと前回紹介した文字列を検索する grep コマンドを組み合わせたちょっとしたテクニックを紹介しましょう。カレントディレクトリ以下の拡張子が「.html」ファイルを検索して、文字列「HP-UX」含む行を表示するにはどうしたらよいかわかる？

タックス君：
「-exec grep "HP-UX" {} \;"」を使って、次のようにすればできますね。

```
$ find . -name "*.html" -exec grep "HP-UX" {} \; 【Enter】
<h1>HP-UX</h1>
<h2>HP-UX Chapter1</h2>
<h1>HP-UX SUPPORT</h1>
<h2>HP-UX 11i Version3:Feb</h2>
```

マリー先生：
そうね。ただ、それだとファイル名はわからないわよね。先月も説明したけど、grep コマンドは引数にファイルを複数指定すると、最初にファイルのパスを表示してくれるの。

```
$ grep 44 mail.txt mail2.txt 【Enter】
mail.txt:oyamada hana,oyama33@example.jp,44
mail2.txt:sakurai taro,ss10@example.com,44
mail2.txt:nakata saburo,nsabu1@example.com,44
```

だから、なんらかのダミーのファイルを引数に加えればファイル名が表示されるわけ。指定するファイルは検索条件にひっかからないものなら何でもいいけど、よく使われるのは中身が空の/dev/null という特別なファイルよ。

```
$ find . -name "*.html" -exec grep "HP-UX" {} /dev/null \; 【Enter】
./abc.html:<h1>HP-UX</h1>
./Site/index.html:<h2>HP-UX  Chapter1</h2>
./Site/samples/info.html:<h1>HP-UX  SUPPORT</h1>
./Site/samples/info.html:<h2>HP-UX 11i  Version3:Feb</h2>
```

ちなみに、/dev/null は読み出したときは空のファイルだけど、データを書き込んでも空のままなの。

```
↓/dev/null にファイルをコピーしても
$ cp /etc/hosts /dev/null 【Enter】
$ cat /dev/null 【Enter】
          ←なにも表示されない
```

なんでも飲み込んでしまう、いわばブラックホールのような存在なわけ。不要なデータを画面に表示したくない場合にも便利だから覚えておいてね。

確認しながらコマンドを実行する

「-exec」オプションの代わりに、「-ok」オプションを使うと、検索されたそれぞれのファイルに対して、確認しながら指定コマンドを実行します。書式は「-exec」オプションと同じく次のような形式です。

```
-ok コマンド {} \;
```

たとえば、カレントディレクトリ以下でファイル名の最後がチルダ「~」のファイルを、ひとつずつ確認しながら削除するには次ようにします。

```
$ find . -name "*~" -type f -ok rm {} \; 【Enter】
< rm …… ./Documents/2008/info.txt~ >?  y  ←削除する
```

四色君：
「some.txt~」や「readme~」のようにファイル名の最後がチルダ「~」のファイルを時々見かけるのですが、なにか意味があるのですか？

マリー先生：
UNIX 系の OS では、最後がチルダ「~」のファイル名は、バックアップファイルの名前としてしばしば使われるの。

練習問題

第 1 問：Documents ディレクトリ以下で、拡張子が「.txt」のファイルを検索するコマンドはどれか？

a)　find -name "*.txt" -type f
b)　find Documents -name "*.txt" -type f
c)　find Documents -name *.txt
d)　find -name "*.txt" -type f Documents

b)　find Documents -name "*.txt" -type f
拡張子が「.txt」のファイルは「-name "*.txt"」で指定できる。このときシェルによってワイルドカードが展開されないようにダブルクォーテーション「"」で囲む必要がある点に注意してほしい。

第 2 問：カレントディレクトリ以下で、名前の先頭が数字で始まり、4 文字のディレクトリを検索するコマンドはどれか？

a)　find . -name "[0-9]*"
b)　find . -name ":number:" -type d
c)　find . -name "[0-9]???" -type f
d)　find . -name "[0-9]???" -type d

d)　find . -name "[0-9]???" -type d
数字は「[0-9]」で、任意の 1 文字は「?」で表記できる。したがって、名前の先頭が数字で始まり、4 文字のディレクトリを検索するオプションは「-name "[0-9]???" -type d」となる。

第 3 問：Pictures ディレクトリ以下で拡張子が「.png」もしくは「.jpg」ファイルの数を表示するコマンドはどれか？

a)　find Pictures/ -name "*.png" -o -name "*.jpg" | wc -l
b)　find Pictures/ -name "*.png" -o -name "*.jpg" -exec wc -l {} \;
c)　find Pictures/ -name "*.png" -exec wc -l {} \;

d)　find Pictures/ -name "*.png" -name "*.jpg" -exec wc -l {} \;

a)　find Pictures/ -name "*.png" -o -name "*.jpg" | wc -l
検索結果を「wc -l」コマンドでカウントするには、find の結果をパイプ「|」で「wc -l」に渡す。

第 4 問：Documents ディレクトリ以下の拡張子が「.txt」のファイルを、確認しながら Backup ディレクトリの下にコピーするコマンドはどれか？

a)　find Documents/ -name "*.txt" -type f -exec cp {} Backup \;
b)　find Documents/ -name "*.txt" -type d -ok cp {} Backup \;
c)　find Documents/ -name "*.txt" -type f -ng cp {} Backup \;
d)　find Documents/ -name "*.txt" -type f -ok cp {} Backup \;

d)　find Documents/ -name "*.txt" -type f -ok cp {} Backup \;
検索結果に対して、その都度確認しながら指定したコマンドを実行するには「-ok コマンド {} \:」オプションを指定する。

UNIX の教科書「応用編」

第 3 日目：vi エディタの操作（活用編）

2008 年 10 月　大津 真

今月のテーマは vi エディタの操作です。基礎編で取り上げた基本的な操作はもう身に付いているでしょうか？応用編では、vi エディタを日常的に使う上でぜひ覚えておきたい便利な機能について学んでいきます。1 時間目はカーソルの移動方法とコピー&ペーストを、2 時間目は置換と削除を取り上げました。どれもテキスト編集の際には欠かせない機能ですから、しっかりマスターして vi エディタを使いこなせるようになりましょう。

マリー先生：
ターミナルで動作するテキストエディタの代表といえる vi エディタについては、UNIX の教科書 基礎編の 5 日目「vi エディタの操作（基本編）」で解説したけど、基本的な操作はできるようになったかな？今回はその応用編として、より実践的な使い方について説明しましょう 。
その前に、前回の find コマンドについてなにか質問ありますか？

四色君：
find コマンドの検索結果を別のコマンドに渡す場合の書式が覚えにくいのですけど……。

マリー先生：
「-exec コマンド {} \:」オプションを使う方法ね。たとえば、カレントディレクトリ以下の拡張子が「.html」のファイルを検索して、文字列「HP-UX」含む行を表示するには次のように実行したわね。

```
$ find . -name "*.html" -exec grep "HP-UX" {} /dev/null \; 【Enter】
./abc.html:<h1>HP-UX</h1>
./Site/index.html:<h2>HP-UX Chapter1</h2>
./Site/samples/info.html:<h1>HP-UX SUPPORT</h1>
./Site/samples/info.html:<h2>HP-UX 11i Version3:Feb</h2>
```

確かに、これは長くて慣れないと難しいでしょう。別の方法として、find と xargs というコマンドをパイプで組み合わせても同じことができるの。xargs は、標準入力から受け取ったデータを指定したコマンドの引数として展開してくれるコマンドで、次の形式で使うの。

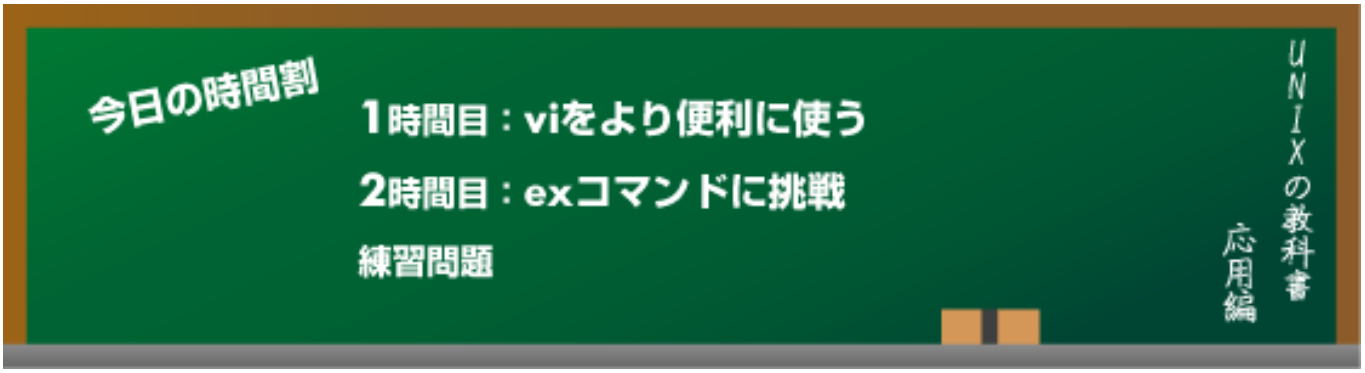
```
コマンド 1 | xargs コマンド 2
```

「コマンド 1」の実行結果が、「コマンド 2」の引数になるわけね。xargs コマンドで、前の例と同じ処理を行うには次のようにすればいいわ。

```
$ find . -name "*.html" | xargs grep  "HP-UX" 【Enter】
./abc.html:<h1>HP-UX</h1>
./Site/index.html:<h2>HP-UX  Chapter1</h2>
./Site/samples/info.html:<h1>HP-UX  SUPPORT</h1>
./Site/samples/info.html:<h2>HP-UX 11i  Version3:Feb</h2>
```

四色君：
あっ、こっちのほうが覚えやすいかも。

マリー先生：
それに、「-exec コマンド {} \:」オプションの場合にはその都度コマンドが実行されるけど、xargs の場合には引数として一度に展開するので、たいていの場合実行時間が短くてすむというメリットもあるのよ。



1 時間目：vi をより便利に使う

vi による基本的な編集方法については基礎編 5 日目の 2 時間目で説明しましたので、この時間では vi を日常的に便利に使う上で覚えておきたい編集機能をまとめておきましょう。

カーソルを行内の指定した文字に移動する

行内カーソルの移動は「h」（1 文字左）、「l」（1 文字右）、「w」（次の単語）などで行えますが、移動したい位置の文字がわかっている場合に「f 文字」「F 文字」コマンドが便利です。

表 1：f 文字・F 文字によるカーソル移動

コマンド	カーソルの動作
f 文字	行末に向かって指定した文字の位置に移動
F 文字	行頭に向かって指定した文字の位置に移動

たとえばコマンドモードで「fh」とタイプすると、現在位置から行末に向かって文字「h」が検索され最初に見つかった場所に移動します。

```
hello hp-ux
        ↓　「fh」コマンドを実行【Enter】
hello hp-ux
```

「Fe」とタイプすると、逆に、行頭に向かって文字「e」が検索されます。

```
hello hp-ux
        ↓　「Fe」コマンドを実行【Enter】
hello hp-ux
```

タックス君：
同じ文字を続けて検索したい場合には「f<文字>」を繰り返し実行しなければならないのですか？

マリー先生：
同じ文字を同じ方向に向かって検索したい場合にはセミコロン「;」をタイプすれば OK よ。
あと、逆方向に検索するにはカンマ「,」を使ってね。

```
hp-ux hp-ux hp-ux hp-ux
          ↓　「fh」コマンドを実行【Enter】
hp-ux hp-ux hp-ux hp-ux
        ↓　「fh」コマンドを実行【Enter】
hp-ux hp-ux hp-ux hp-ux
      ↓　「;」コマンドを実行【Enter】
hp-ux hp-ux hp-ux hp-ux
    ↓　「,」コマンドを実行【Enter】
hp-ux hp-ux hp-ux hp-ux
```

1 文字を置き換える

それでは、カーソル位置の 1 文字だけを別の文字にしたいといった場合はどうすればよいでしょうか？　普通に考えれば、「x」コマンドで 1 文字削除して、「i」コマンドでインサートモードに移行し、目的の文字をタイプしてから esc キーでコマンドモードに戻すという手順になるでしょう。実は「r」コマンドを使用することで簡単にカーソル位置の 1 文字を置き換えられます。esc キーでコマンドモードに戻す必要もありません。次に「hp-ux」の「h」を「H」に置き換える例を示します。

```
hp-ux
    ↓　「r」コマンドを実行【Enter】
  ↓　「H」をタイプ【Enter】
Hp-ux
```

タックス君：
あっ、これは便利ですね。

四色君：
行全体を置き換えることもできますか？

マリー先生：
「r」コマンドの代わりに「S」コマンドを使えばできるわよ。「S」コマンドを実行すると現在カーソルのある行全体がクリアされるので、文字列をタイプしてね。

文字列の検索

文字列を検索するには、まず「/」をタイプします。続けて検索したい文字列を入力し Enter キーを押すと、ファイルの後方に向かって検索が行われ、最初に見つかった文字列にカーソルが移動します。同じ文字列を同じ方向に向かって検索するには「n」コマンドを実行します。

```
unix
hp
unix
hp
unix
    ↓　「/hp」コマンドを実行【Enter】
unix
hp
unix
hp
unix
  ↓　「n」コマンドを実行【Enter】
unix
hp
unix
hp
unix
```

タックス君：
ファイルの先頭に向かって検索することはできますか？

マリー先生：
「/」の代わりに「?」を使えばできるわよ。その場合、「n」コマンドと「N」コマンドの向きが逆になるから注意してね。あと、検索文字列には正規表現も使えるわよ。たとえば、文頭に向かって行末の「hp-ux」を検索するには「?hp-ux$」と実行してね。

範囲のコピー&ペースト

GUI のエディタの場合にはマウスのドラッグによるコピー&ペーストで文字列のコピーができましたが、伝統的な vi ではマウスが使えません。vi でコピー&ペーストを行うには、いったん範囲をバッファと呼ばれる一時的な領域に格納し、その内容を目的の位置にペーストします。

まず、行のコピーから説明しましょう。現在行をバッファに格納するには「yy」コマンド（もしくは「Y」コマンド）を使用します。また、バッファの内容を現在行の直前に挿入するには「P」コマンドを使います。したがって、現在行を別の行にコピーする手順は次のようになります。

1. コピーしたい行に移動し、「yy」コマンドでバッファに移動する
2. コピー先の行に移動し、「P」コマンドでペーストする

次に、3 行目を 1 行目の前にコピーする例を示します。

```
1 HP-UX
2 HP-UX
3 HP-UX
4 HP-UX
    ↓　「yy」コマンドを実行【Enter】
    ↓　1 行目にカーソルを移動
1 HP-UX
2 HP-UX
3 HP-UX
4 HP-UX
  ↓　「P」コマンドを実行【Enter】
3 HP-UX
1 HP-UX
2 HP-UX
3 HP-UX
4 HP-UX
```

なお、現在行の次の行にペーストしたい場合には「P」ではなく「p」コマンドを使います。

タックス君：
複数行をコピーすることはできますか？

マリー先生：
もちろんできるわよ。その場合は「yy」の前に行番号を指定するの。たとえば現在行から 4 行をバッファに格納するには「4yy」となるわけね。

なお、「y」コマンドを次の形式で使用すると、現在位置からカーソルの異動先までの範囲をバッファに格納できます。

```
y<カーソル移動コマンド>
```

たとえば、カーソル位置から行末までをバッファに格納するには「y$」を実行します。

範囲の移動

コピーではなく、指定した範囲を移動するにはどうすればよいでしょうか？　実は「dd」コマンドや「d<カーソル移動>」コマンドを実行すると、削除した範囲がバッファに格納されます。したがって、目的の位置で「p」コマンドもしくは「P」コマンドで目的の位置にペーストすれば範囲を移動できます。
次に、2 行目と 3 行目をファイルの先頭に移動する例を示します。

```
1 HP-UX
2 HP-UX
3 HP-UX
4 HP-UX
      ↓　「2dd」コマンドを実行【Enter】
1 HP-UX
4 HP-UX
   ↓　1 行目にカーソルを移動
1 HP-UX
4 HP-UX
   ↓　「P」コマンドを実行【Enter】
2 HP-UX
3 HP-UX
1 HP-UX
4 HP-UX
```

休憩時間：Windows でも vi

vi の操作に慣れてくると、Windows のエディタとしても使いたいと思う方も少なくないでしょう。
伝統的な vi を拡張したエディタを vi クローンなどと呼びます。vi クローンの代表といえるのが vim です。vim は現在 Windows 版も用意されています。Windows 版 vim の日本語対応を強化したものが以下のサイトからダウンロード可能です。

http://www.kaoriya.net/

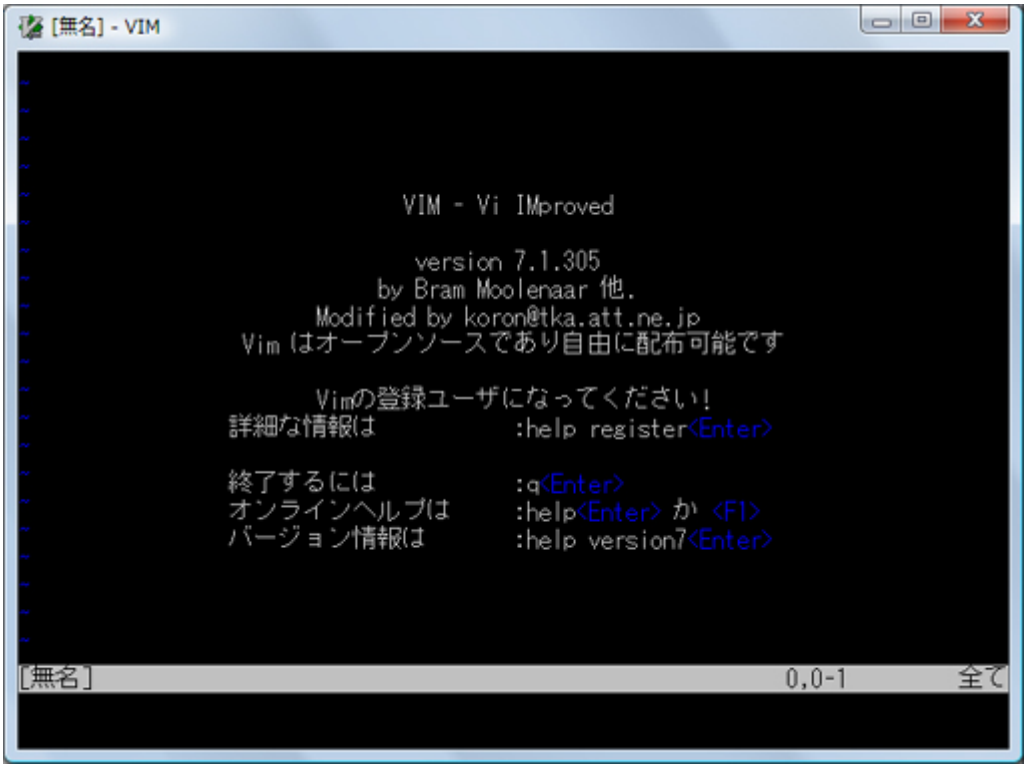


図 1：Windows 版 vim の起動画面

2 時間目：ex コマンドに挑戦

基礎編でも説明したように、vi エディタの前身はラインエディタ「ex」です。たとえば vi を終了するのに使用した「:q」や「:wq」のようにコロン「:」で始まるコマンドは ex コマンドでした。ex コマンドを使いこなせるようになると、よりきめ細やかな編集が可能になります。ex コマンドは多岐にわたりますが、ここでは vi でファイルの編集をしていく上で特に便利な、置換を行う「s」コマンドと、削除を行う「d」コマンドを中心に解説しましょう。

現在行の文字列を置き換える

ex コマンドの中でもっとも頻繁に使用するコマンドが、文字列を置換する「s」コマンドです。これを使って、現在行の文字列を置き換えるには次の書式で実行します。

```
:s/置換対象の文字列/置換後の文字列/
```

たとえば、現在行の文字列「hp」を「HP」に置き換えるにはつぎのようにします。

```
hp-ux hp1 hp unix hp2
      ↓　「:s/hp/HP/」コマンドを実行【Enter】
HP-ux hp1 hp unix hp2
```

四色君：
これだと、カーソルの位置にかかわらず最初に見つかった文字列のみが置換されますね。

マリー先生：
そうね、デフォルトでは行の頭から検索していって最初に見つかった文字列だけが置換されるの。最後に「g」を指定して「:s/hp/HP/g」とすると、見つかったすべての文字列が置換されるわ。

```
HP-ux hp1 hp unix hp2
      ↓　「:s/hp/HP/g」コマンドを実行【Enter】
HP-ux HP1 HP unix HP2
```

タックス君：
スラッシュ「/」を置換することはできますか？

マリー先生：
スラッシュ「/」はそのままでは置換対象の文字列と置換後の文字列との区切りの記号として使われるの。「/」を文字として扱いたければ、前に逆スラッシュ「\」を付けてエスケープすればいいわ。たとえば、現在行のスラッシュ「/」をすべてコロン「:」に置換するには次のようにするの。この場合も最後に「g」を付けることを忘れないでね。

```
Sample1/Work/15
      ↓　「:s/\//:/g」コマンドを実行【Enter】
Sample1:Work:15
```

ファイル内にあるすべての文字列を置き換える

「s」コマンドはデフォルトでは現在行を対象としますが、その前に行の範囲を指定することで、その範囲を置換対象とすることが可能です。もっとも頻繁に使用する範囲を表す記号が「%」です。「%」はファイル全体を表します。したがって、ファイル全体にわたって「hp」を「HP」に置換するには次のようにします。

```
hp-ux hp-ux hp unix
hp-ux unix hp
hp-ux hp unix
      ↓　「:%s/hp/HP/g」コマンドを実行【Enter】
HP-ux HP-ux HP unix
HP-ux unix HP
HP-ux HP unix
```

タックス君：
この場合も最後に「g」が必要ですか？

マリー先生：
そうね。「g」を付けないと、各行の最初に見つかった文字列しか置換されないわ。

```
hp-ux hp-ux hp unix
hp-ux unix hp
hp-ux hp unix
      ↓　「:%s/hp/HP/」コマンドを実行【Enter】
HP-ux hp-ux hp unix
HP-ux unix hp
HP-ux hp unix
```

四色君：
置換するかどうかを、ひとつずつ確認しながら決めることはできますか？

マリー先生：
最後に「c」を追加して「:%s/hp/HP/gc」とすればできるわよ。そうすると、見つかった文字列ごとに画面の下に次のように表示されるの。

```
2 hp-UX
 ^^
```

続けて「y」をタイプすれば置換、それ以外の文字では置換されなくなるわ。

行番号を指定して範囲を指定する

範囲の行番号を、次のように開始行と終了行をカンマ「,」で区切って指定することもできます。

```
開始行,終了行
```

終了行を省略した場合には開始行のみが対象になります。たとえば、「2s/hp/HP/g」とした場合 2 行目が対象となり、「2,4s/hp/HP/g」とすれば 2 行目から 4 行目が対象となって、文字列「hp」が「HP」に置換されます。

タックス君：
たとえば、現在行からファイルの最後までを対象にしたい場合には、あらかじめ行番号を調べる必要があるのですか？

マリー先生：
もっと簡単な方法があるわ。ex では、現在行をピリオド「.」、ファイルの最後を「$」で表記できるの。

したがって、現在行からファイルの最後までを対象に「hp」を「HP」に置換するには、「:.,$s/hp/HP/g」を実行すればいいの。

```
hp-ux hp-ux hp unix
hp-ux unix hp
hp-ux hp-ux hp unix     ←現在行
hp-ux unix hp
hp-ux hp unix
hp-ux hp unix
      ↓   「:.,$s/hp/HP/g」コマンドを実行【Enter】
hp-ux hp-ux hp unix
hp-ux unix hp
HP-ux HP-ux HP unix
HP-ux unix HP
HP-ux HP unix
HP-ux HP unix
```

また、ファイルの最初から現在行までを対象にしたければ、範囲を次のように指定すればいいわけ。

正規表現を使用した置換対象行の指定

置換対象となる範囲を次のような形式で指定すると、正規表現のパターンに一致する行を対象に置換が行われます。

```
g/正規表現/
```

たとえば、行頭が「NG」で始まる行を対象に文字列「AA」を「xx」に変換するには、範囲を「g/^NG/」のように指定して「:g/^NG/s/AA/xx/g」を実行します。

```
OK:AA:12
NG:AA:01
OK:AA:01
NG:AA:01
NG:AA:01
      ↓　「:g/^NG/s/AA/xx/g」コマンドを実行【Enter】
OK:AA:12
NG:xx:01
OK:AA:01
NG:xx:01
NG:xx:01
```

なお、行番号の代わりに「/正規表現/」で行を指定することもできます。たとえば、1 行目から行頭が「30」で始まる行までを対象として文字列「hp」を「HP」に置換するには、範囲を「1,/^30/」のように指定して「:1,/^30/s/hp/HP/g」のようにします。

```
10      hp
20      hp
30      hp
40      hp
50      hp
      ↓　「:1,/^30/s/hp/HP/g」コマンドを実行【Enter】
10    HP
20    HP
30    HP
40    hp
50    hp
```

行の削除

続いてもうひとつ、便利な ex コマンドとして行を削除する「d」コマンドを紹介しましょう。「s」コマンドと同じように前に範囲を指定できます。

```
範囲 d
```

たとえば、3 行目から 4 行目までを削除するには「:3,4d」、現在行から行末までを削除するには「:.,$d」を実行します。
もちろん、「s」コマンドと同じく「g/正規表現/」を使って正規表現のパターンに一致する行を削除することもできます。たとえば、「:g/^[0-9]/d」とすると行頭が数字で始まる行が削除されます。

```
1 hp-ux
unix
3 hp-ux
4 hp
unix
    ↓　「:g/^[0-9]/d」コマンドを実行【Enter】
unix
unix
```

タックス君：
行の削除は vi の「dd」や「d」コマンドでもできますよね。

マリー先生：
そう、ex コマンドの「:.,$d」は、vi のコマンドでは「dG」ね。ただ、範囲の開始行と終了行を指定したり、正規表現のパターンにマッチする行のみを削除したりといった場合は、ex コマンドを使う必要があるわね。

それから、範囲の指定はファイルの書き込みを行う「w」コマンドでも使えるのよ。

タックス君：
指定した範囲だけをファイルに書き込めるということですか？

マリー先生：
そう。たとえば「:3,$w test.txt」では 3 行目からファイルの最後までをファイル「test.txt」に書き込むの。

練習問題

第 1 問：現在行の行末に向かって文字「f」を検索し、最初に見つかった文字「f」にカーソルを移動するコマンドはどれか？

a)　sf
b)　f
c)　ff
d)　xf

c)　ff
現在行の後方に向かって文字を検索するには「f<文字>」コマンドを実行する。

第 2 問：現在行から 3 行をバッファに格納するコマンドはどれか？

a)　3yy
b)　yy3
c)　3y
d)　y3s

a)　3yy
「<数値>yy」で指定した行数をバッファに格納する。

第 3 問：2 行目から現在行までの範囲で、文字列「hp」をすべて「HP」に置換するコマンドはどれか？

a)　:2,$s/hp/HP/g
b)　:2,.s/hp/HP/g
c)　:2,.s/hp/HP/
d)　:1,.s/hp/HP/

b)　:2,.s/hp/HP/g
現在行はピリオド「.」で表せる。また、すべての文字列を置換するには最後に「g」が必要となる。

第 4 問：空行をすべて削除するコマンドはどれか？

a)　:g/^[[:space:]]*$/d
b)　:/^[[:space:]]*$/d
c)　:g%d
d)　:g/^ *$/d

a)　:g/^[[:space:]]*$/d
「/正規表現/」で行の指定ができる。したがって、正規表現のパターンに一致する行を削除するには「:g/正規表現/d」コマンドを実行すればよい。空行を表すパターンは「/^[[:space:]]*$」である。

UNIX の教科書「応用編」

第 4 日目：ファイルの圧縮とアーカイブ

2008 年 11 月　大津 真

今月のテーマはファイルの圧縮・解凍と、アーカイブの管理です。1 時間目にはファイルサイズを小さく圧縮する compress と gzip の使い方を、2 時間目には複数のファイルを 1 つにまとめて管理する tar の使い方を学びます。メールでファイルをやり取りする場合などに便利なだけではなく、インターネットで公開されているソフトウェアを利用する際にも必須の知識なので、よく理解しておきましょう。

マリー先生：
今回はファイルの圧縮・解凍と、複数のファイルを 1 つにまとめて管理するアーカイブの作成について説明しましょう。
その前に、前回の vi エディタの操作についてなにか質問ありますか？

四色君：
vi コマンドの使い方にも少しずつ慣れてきて、わりと大きなファイルでも編集できるようになってきました。 ところで、離れた場所を行き来したりするのに便利な機能があるとラクなのですが……。

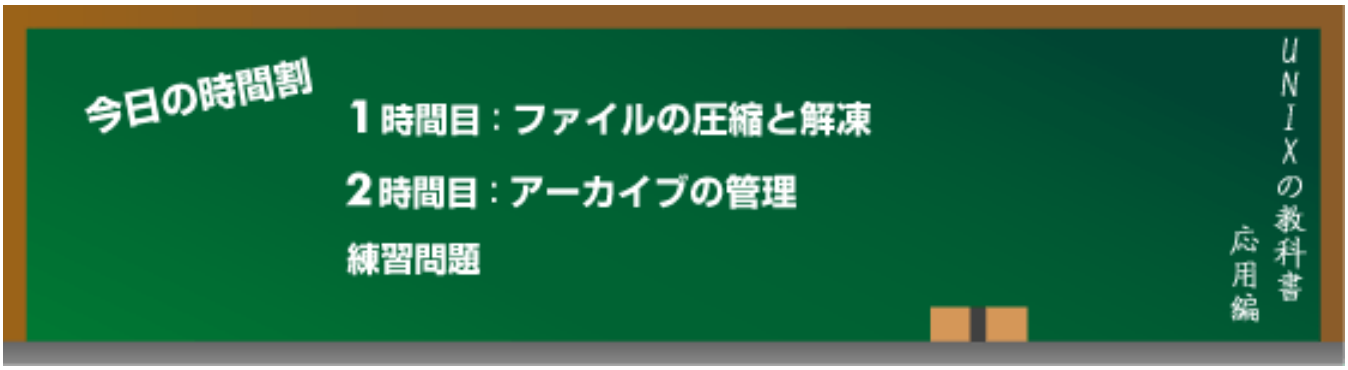
マリー先生：
そんなときは、「m<アルファベット小文字>」コマンドでカーソル位置にマークを付けておけばいいわ。つまり、「ma」とするとカーソル位置に「a」というマークが付くの。マークした位置にジャンプするには次のいずれかを使うの。

```
'<マーク>
`<マーク>
```

ちょっと間違えやすいけど、最初の記号はシングルクォーテーション「'」で、2 番目はバッククォーテーション「`」ね。前者はマークした行の行頭に、後者はカーソル位置にジャンプできるわ。

タックス君：
あっ、それは便利ですね。たとえば、現在位置とファイルの最後を行ったり来たりするには、現在位置に「ma」で「a」というマークを付けておいて、「G」で最後に移動し「`a」で戻るというのを繰り返せばよいのですね。

マリー先生：
そうね。それから、マークは範囲の消去や移動にも使えるのよ。「d`<マーク>」とすれば現在位置とマークした位置の間が消去されるの。消去された範囲はバッファに格納されているから、「p」などでペーストできるわ。



1 時間目：ファイルの圧縮と解凍

ファイルの圧縮と言うと、Windows における ZIP のように複数のファイルをまとめて圧縮するといったイメージがある方もいると思います。UNIX 系でももちろんそれは可能ですが、まずはファイルを個別に圧縮する方法について説明しましょう。

compress コマンド

現在 UNIX にはさまざはな圧縮コマンドがありますが、compress はもっとも伝統的な圧縮コマンドです。次のような書式で使用します。

```
compress ファイルのパス
```

「-v」オプションを指定して実行すると、圧縮率などの情報が表示されます。 たとえば、カレントディレクトリの下にある約 23MB のファイル「house.bmp」を compress 形式で圧縮するには次のようにします。圧縮が完了すると元のファイルは削除され、拡張子「.Z」が付けられた圧縮ファイルが生成されます。また、元のファイルのアクセス権限や所有者、修正日時などの情報は保持されます。

```
$ ls -l【Enter】
total 46832                                          約23MBのbmpファイル↓
-rw-------   1 o2         users      23970870 Oct 21 15:57 house.bmp
$ compress -v house.bmp【Enter】
house.bmp: Compression: 17.96% -- replaced with house.bmp.Z ←約18%圧縮された
$ ls -l【Enter】
total 38416                                          約19MBの圧縮ファイル↓
-rw-------   1 o2         users      19664562 Oct 21 15:57 house.bmp.Z
```

マリー先生：
コンピュータのファイルの圧縮方式は、可逆圧縮と非可逆圧縮の 2 種類があることは知っているわよね。

タックス君：
名前からすると、可逆圧縮は、圧縮ファイルを解凍して元の状態に戻せる形式で、非可逆圧縮は戻せない形式ですよね。

マリー先生：
そうね。通常のデータファイルは元に戻せないと困るから可逆圧縮で圧縮されるの。それに対して、不可逆圧縮はマルチメディア系のデータで活躍しているわ。たとえば、画像フォーマットの JPEG や、音声ファイルの MP3 は、人間にとって不要なデータを間引くことで高い圧縮率を実現しているかわりに、元には戻せないわけね。

圧縮ファイルの解凍

compress 形式の圧縮ファイルの解凍（伸張）には uncompress コマンドを使用します。解凍が完了すると圧縮ファイルは削除されます。

```
$ ls -l【Enter】
total 38416
-rw-------   1 o2       users     19664562 Oct 21 15:57 house.bmp.Z
$ uncompress house.bmp.Z
$ ls -l【Enter】
total 46832
-rw-------   1 o2       users     23970870 Oct 21 15:57 house.bmp
```

マリー先生：
compress コマンドに「-d」オプションをつけて実行しても、ファイルの解凍ができるのよ。

四色君：
それでは uncompress コマンドの存在はディスクの無駄ではないのですか？

マリー先生：
そんなことはないわ。実は compress コマンドと uncompress コマンドは同じ実体、つまりハードリンクなの。次のように「ls -il」コマンドを実行すると i ノード番号が同じであることがわかるわ。

```
$ ls -il /usr/bin/compress /usr/bin/uncompress【Enter】
1049 -r-xr-xr-x  3 bin   bin    76548 Feb 16  2007 /usr/bin/compress
1049 -r-xr-xr-x  3 bin   bin    76548 Feb 16  2007 /usr/bin/uncompress
 iノード番号が同じ
```

タックス君：
同じコマンドを、呼び出したコマンド名によって処理を切り分けているわけですね。

マリー先生：
そうね。UNIX にはそのようなコマンドがいくつかあるわ。たとえば、grep ファミリーの grep、egrep、fgrep なんかもそうね。

```
$ ls -il /usr/bin/grep /usr/bin/egrep /usr/bin/fgrep【Enter】
1416 -r-xr-xr-x  3 bin   bin    74728 Feb 16  2007 /usr/bin/egrep
1416 -r-xr-xr-x  3 bin   bin    74728 Feb 16  2007 /usr/bin/fgrep
1416 -r-xr-xr-x  3 bin   bin    74728 Feb 16  2007 /usr/bin/grep
 iノード番号が同じ
```

ワイルドカードによるファイル指定

compress コマンドの引数に複数のファイルを指定して、一度に圧縮することも可能です。もちろん、「*」（0 個以上の任意の文字列）や「?」（任意の 1 文字）といったシェルのワイルドカードも使用できます。たとえば、Documents ディレクトリの下の拡張子が「.txt」のファイルをまとめて圧縮するには次のようにします。

```
$ ls -l Documents/【Enter】
total 96
drwxr-x---  2 o2      users       96 Aug 24 23:00 2007
drwxr-x---  2 o2      users     8192 Aug 24 23:35 2008
-rw-r-----  1 o2      users    12526 Oct 21 16:15 info.txt
-rw-r-----  1 o2      users     5023 Oct 21 16:16 mail.txt
-rw-r-----  1 o2      users      553 Oct 21 16:18 main.html
drwxr-x---  2 o2      users       96 Aug 24 23:03 sample
-rw-r-----  1 o2      users       25 Aug 24 22:59 sample.txt
$ compress Documents/*.txt
$ ls -l Documents/【Enter】
total 80
drwxr-x---  2 o2      users       96 Aug 24 23:00 2007
```

```
drwxr-x---  2 o2      users      8192 Aug 24 23:35 2008
-rw-r-----  1 o2      users      5517 Oct 21 16:15 info.txt.Z
-rw-r-----  1 o2      users      2482 Oct 21 16:16 mail.txt.Z
-rw-r-----  1 o2      users       553 Oct 21 16:18 main.html
drwxr-x---  2 o2      users        96 Aug 24 23:03 sample
-rw-r-----  1 o2      users        25 Aug 24 22:59 sample.txt
```

四色君：
あれ？　sample.txt が圧縮されずにそのまま残っていますけど……。

マリー先生：
圧縮してもファイルサイズが小さくならない場合、compress はそのファイルを無視してそのまま残すの。「-f」オプションを付けるとそういうファイルも強制的に圧縮できるけれど、このオプションを付けると圧縮後のファイル名と同じファイルがあった場合にも確認せずに上書きされてしまうから注意してね。

同じように、uncompress コマンドで解凍する場合もワイルドカードが使えます。Documents ディレクトリの下にある compress 形式の圧縮ファイルをすべて解凍するには、引数に「 Documents/*.Z」を指定します。

```
$ uncompress Documents/*.Z
```

ディレクトリをたどって圧縮する

指定したディレクトリを再帰的にたどって、すべてのファイルを圧縮するにはどうしたらよいでしょうか？

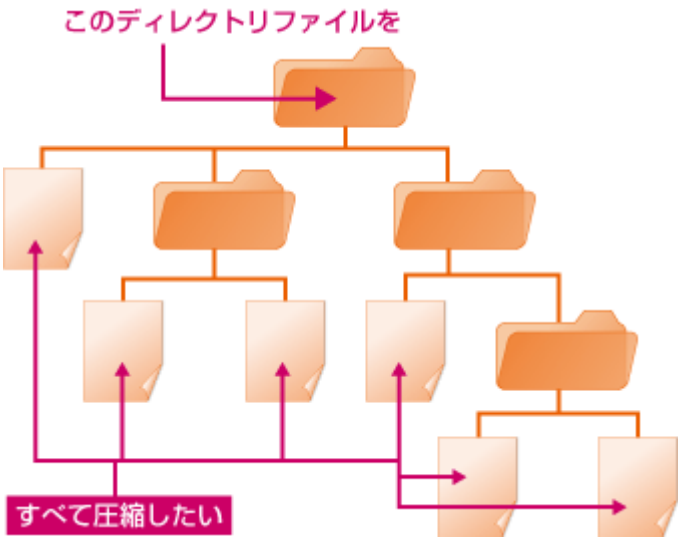


図 1：ディレクトリの下にあるファイルをすべて圧縮するには？

これは、find コマンドと compress コマンドを組み合わせることによって実現できます。たとえば Documents ディレクトリ以下のサブディレクトリをたどってすべてのファイルを圧縮するには、次のようにします。

```
$ find Documents/ -type f | xargs compress
```

もしくは

```
$ find Documents/ -type f -exec compress {} \;
```

実は、いつもこのような長いコマンドを実行しなくてもすむように、HP-UX には compressdir と uncompressdir という便利なコマンドも用意されています。

```
$ compressdir Documents/←Documentsディレクトリ以下を再帰的にたどってファイルを圧縮
$ uncompressdir Documents/←Documentsディレクトリ以下を再帰的にたどってファイルを解凍
```

マリー先生：
compressdir と uncompressdir は、普通のコマンドのようなバイナリファイルではなくてテキストファイルなの。

四色君：
テキストファイルがコマンドなのですか？

マリー先生：
そう。実際に compressdir（/usr/bin/compressdir）の中をのぞいてみるとわかるけど、シェルのコマンドが羅列されたテキストファイルなの。シェルというのはユーザのコマンドを解釈するコマンドインタプリタとしての役割だけではなく、プログラミング言語でもあるのよ。そのようなシェルによって記述したプログラムのことを「シェルスクリプト」と呼ぶので覚えておいてね。

gzip コマンドによる圧縮

compress は古くからあるコマンドで、ほとんどの UNIX 系 OS に搭載されていますが、あまり圧縮率が高くありません。そのため、gzip というオープンソースの圧縮プログラムも普及していきます。gzip は最近の HP-UX には標準搭載されています。
前述の compress コマンドの例で使用したカレントディレクトリの下にある約 23MB のファイル「house.bmp」を compress 形式で圧縮するには、次のようにします。

```
$ ls -l【Enter】
total 46832
-rw-------   1 o2       users     23970870 Oct 21 15:57 house.bmp
$ gzip -v house.bmp【Enter】
```

```
house.bmp:       24.5% -- replaced with house.bmp.gz   ←約24.5%圧縮
$ ls -l【Enter】
total 35360
-rw-------   1 o2       users     18103138 Oct 21 15:57 house.bmp.gz
```

gzip 形式の圧縮ファイルを解凍するには、gunzip コマンド（もしくは「gzip -d 」）を使用します。

```
$ gunzip house.bmp.gz
$ ls -l【Enter】
total 46832
-rw-------   1 o2       users     23970870 Oct 21 15:57 house.bmp
```

なお、gunzip（もしくは「gzip -d 」）では compress 形式の圧縮ファイルを解凍することもできます。

マリー先生：
gzip は、GNU プロジェクトというフリーの UNIX 互換 OS を目指すプロジェクトのプロダクトなの。GNU のプロダクトはオープンソースのはしりのようなもので、GPL というライセンスに従ってソースコードが公開されていて、だれでも自由に利用できるの。

タックス君：
GNU というと Linux を思い出しますが……。

マリー先生：
そうね、狭義の Linux はカーネル部分だけで、コンパイラやコマンド群などの多くは GNU のプロダクトを利用しているの。

休憩時間：GNU プロジェクト

近年のオープンソース・ソフトウエア・ブームを語る上で欠かすことのできない存在が GNU プロジェクトです。GNU プロジェクトは UNIX 互換 OS をすべてフリーソフトウエアで実装することを目指して、リチャード・ストールマン氏を中心に 1984 年に設立されました。
たとえば、現在さまざまな環境で広く使われている、GCC（コンパイラ）、bash（高機能シェル）、emacs（エディタ）などは GNU のプロダクトです。
また、オープンソースソフトウエアのライセンスとして広く普及している GPL（GNU General Public License）も、元々は GNU プロジェクトで作成したソフトウェアの配布のために制定されたものです。

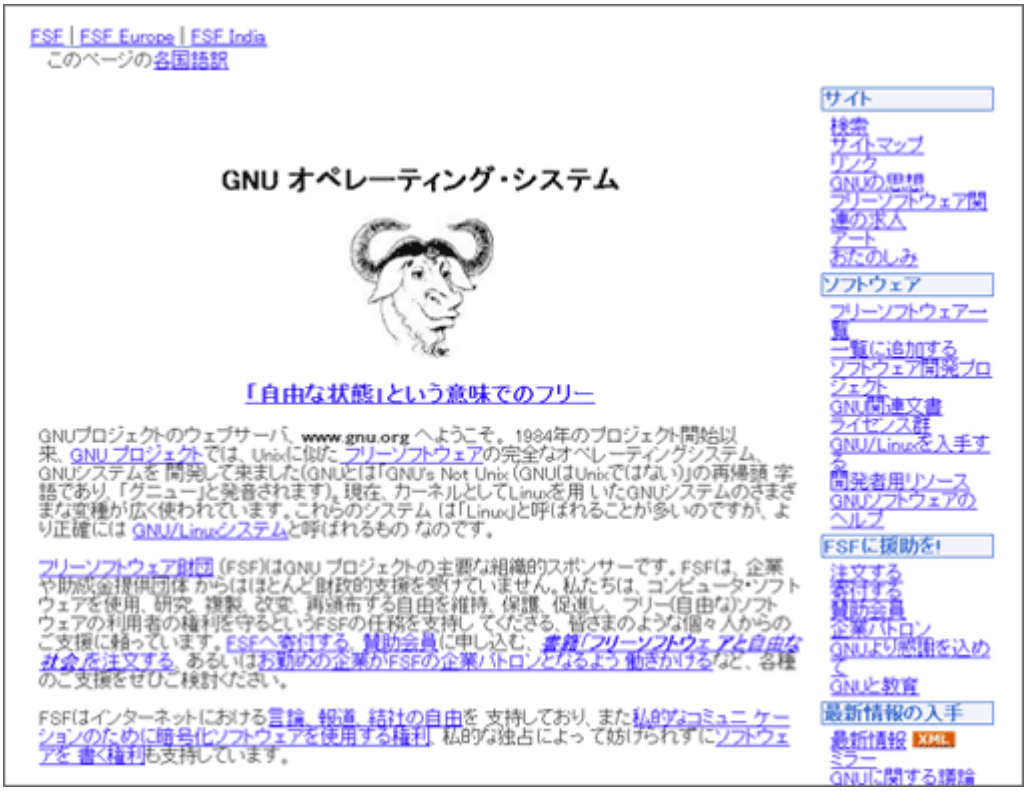


図 2：GNU プロジェクト

2 時間目：アーカイブの管理

2 時間目では、複数のファイルをまとめて管理する「アーカイブ」（書庫）について説明しましょう。UNIX の世界でアーカイブ管理ツールの代表といえるのが tar コマンドです。tar は「Tape Archive」の略で、もともとはテープドライブにバックアップ用のアーカイブを作成するためのツールでしたが、最近ではディスク上のファイルとしてのアーカイブの作成に使用されることも増えています。

アーカイブの作成

tar コマンド使用にはいくつかのオプション指定のパターンがあります。基本的な状況に応じて説明しますので、できればそのまま覚えてしまいましょう。
まず、指定したディレクトリからアーカイブを作成するには次の書式を使用します。

```
tar -cvf アーカイブのパス ディレクトリのパス
```

tar 形式のアーカイブの拡張子は一般的に「.tar」になります。たとえば Documents ディレクトリのアーカイブを「Documents.tar」という名前で作成するには次のようにします。

```
$ tar -cvf  Documents.tar Documents/【Enter】
a Documents/2007/ok.txt 1 blocks
a Documents/2007/ng.txt 1 blocks
a Documents/2007/readme.txt 1 blocks
a Documents/2008/sep.txt 1 blocks
a Documents/2008/magic.txt 1 blocks
a Documents/2008/sample.txt 1 blocks
a Documents/2008/sample 1 blocks
a Documents/2008/9th.txt 1 blocks
a Documents/sample.txt 1 blocks
a Documents/mail.txt 10 blocks
a Documents/info.txt 25 blocks
a Documents/main.html 2 blocks
```

四色君：
「-cvf」オプションにはどのような意味があるのですか？

マリー先生：
「c」は作成（Create）、「v」は詳細メッセージを表示（Verbos）で、最後の「f」でアーカイブ・ファイルのパスを指定しているの。

タックス君：
compress コマンドや gzip コマンドのように、作成されたアーカイブ・ファイルに拡張子「.tar」は自動的に付かないのですか？

マリー先生：
そうね、拡張子は自分で付けないとだめなの。拡張子のないアーカイブも作成できるけど、後で見てわかりにくいので、かならず付けるようにしてね。

アーカイブの中身を表示する

アーカイブの中にあるファイルの一覧を表示するには次の書式で実行します。

```
tar -tvf アーカイブのパス
```

たとえば、Documents.tar の中身を表示するには次のようにします。

```
$ tar -tvf  Documents.tar Documents/【Enter】
rwxr-x--- 111/20     0 Oct 21 16:56 2008 Documents/
rwxr-x--- 111/20     0 Oct 21 16:56 2008 Documents/2007/
rw-r----- 111/20    25 Aug 24 23:00 2008 Documents/2007/ok.txt
rw-r----- 111/20    25 Aug 24 23:00 2008 Documents/2007/ng.txt
rw-r----- 111/20    25 Aug 24 23:00 2008 Documents/2007/readme.txt
rwxr-x--- 111/20     0 Oct 21 16:56 2008 Documents/2008/
rw-r----- 111/20    25 Aug 24 23:00 2008 Documents/2008/sep.txt
rw-r----- 111/20    25 Aug 24 23:00 2008 Documents/2008/magic.txt
rw-r----- 111/20    25 Aug 24 23:02 2008 Documents/2008/sample.txt
rw-r----- 111/20    25 Aug 24 23:03 2008 Documents/2008/sample
rw-r----- 111/20    25 Aug 24 23:11 2008 Documents/2008/9th.txt
rw-r----- 111/20    25 Aug 24 22:59 2008 Documents/sample.txt
rwxr-x--- 111/20     0 Aug 24 23:03 2008 Documents/sample/
rw-r----- 111/20   5023 Oct 21 16:16 2008 Documents/mail.txt
rw-r----- 111/20  12526 Oct 21 16:15 2008 Documents/info.txt
rw-r----- 111/20    553 Oct 21 16:18 2008 Documents/main.html
```

アーカイブを展開する

カレントディレクトリにアーカイブの中身をすべて展開するには、次の書式で実行します。

```
tar -xvf  アーカイブのパス
```

たとえば、Documents.tar をカレントディレクトリに展開するには次の書式で実行します。

```
$ ls -l【Enter】
total 80
-rw-r-----   1 o2   users   40960 Oct 21 17:31 Documents.tar
$ tar -xvf Documents.tar【Enter】
x Documents/2007/ok.txt, 25 bytes, 1 tape blocks
x Documents/2007/ng.txt, 25 bytes, 1 tape blocks
x Documents/2007/readme.txt, 25 bytes, 1 tape blocks
x Documents/2008/sep.txt, 25 bytes, 1 tape blocks
x Documents/2008/magic.txt, 25 bytes, 1 tape blocks
x Documents/2008/sample.txt, 25 bytes, 1 tape blocks
x Documents/2008/sample, 25 bytes, 1 tape blocks
x Documents/2008/9th.txt, 25 bytes, 1 tape blocks
x Documents/sample.txt, 25 bytes, 1 tape blocks
x Documents/mail.txt, 5023 bytes, 10 tape blocks
x Documents/info.txt, 12526 bytes, 25 tape blocks
x Documents/main.html, 553 bytes, 2 tape blocks
```

```
$ ls -l【Enter】
total 96                                                    ↓アーカイブの中身が展開された
drwxr-x---   5 o2   users    8192 Oct 21 16:56 Documents
-rw-r-----   1 o2   users   40960 Oct 21 17:31 Documents.tar
```

タックス君：
アーカイブから指定したファイルだけを取り出すことはできますか？

マリー先生：
「tar -xvf アーカイブのパス」の後の引数に、取り出すファイルパスを指定すればできるわよ。このとき取り出すファイルのパスはアーカイブ内のパスで指定することに注意してね。たとえば「Documents/sample.txt」を取り出すには次のようにするの。

```
$ tar -xvf Documents.tar Documents/sample.txt
x Documents/sample.txt, 25 bytes, 1 tape blocks【Enter】
$ ls Documents
sample.txt  ←sample.txtのみが取り出された
```

圧縮アーカイブの作成

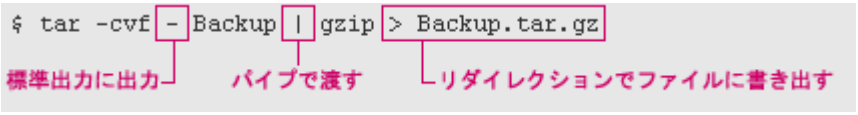
アーカイブはそれ自体では圧縮されていません。メールなどで受け渡す場合には、通常アーカイブをさらに圧縮コマンドを使用して圧縮したほうがよいでしょう。圧縮されたアーカイブのことを「圧縮アーカイブ」と言います。
たとえば、Documents.tar の gzip 形式の圧縮アーカイブを作成するには次のようにします。gzip 形式の圧縮アーカイブの拡張子は「.tar.gz」となることを覚えておきましょう。

```
$ gzip Documents.tar
$ ls -l【Enter】
total 16
-rw-r-----   1 o2   users    6129 Oct 21 17:31 Documents.tar.gz ←圧縮アーカイブ
```

なお、アーカイブの作成と圧縮を一度に行うには、パイプを使用して tar コマンドと gzip コマンドを組み合わせます。たとえば Backup ディレクトリの圧縮アーカイブ「Backup.tar.gz」を作成するには、次のようにします。

```
$ tar -cvf - Backup | gzip > Backup.tar.gz【Enter】
a Backup/ok.txt 1 blocks
a Backup/ng.txt 1 blocks
a Backup/2008/sep.txt 1 blocks
a Backup/2008/magic.txt 1 blocks
a Backup/2008/sample.txt 1 blocks
a Backup/2008/sample 1 blocks
a Backup/2008/9th.txt 1 blocks
```

このとき、tar コマンドの引数のファイル部分には「-」を指定していますが、これは結果を標準出力に書き出す指定です。結果をパイプで gzip コマンドに接続し、それを標準出力のリダイレクション「>」を使用して「Backup.tar.gz」に書き出しています。



なお、圧縮アーカイブを解凍する場合には gunzip コマンドに「-c」オプションを指定します。これは結果を強制的に標準出力に書き出すオプションです。

```
$ gunzip -c Backup.tar.gz | tar -xvf -【Enter】
x Backup/ok.txt, 25 bytes, 1 tape blocks
x Backup/ng.txt, 25 bytes, 1 tape blocks
x Backup/2008/sep.txt, 25 bytes, 1 tape blocks
x Backup/2008/magic.txt, 25 bytes, 1 tape blocks
x Backup/2008/sample.txt, 25 bytes, 1 tape blocks
x Backup/2008/sample, 25 bytes, 1 tape blocks
x Backup/2008/9th.txt, 25 bytes, 1 tape blocks
```

マリー先生：
tar + gzip 形式の圧縮アーカイブ形式はインターネットでソフトウェアを公開する場合によく使われる形式で、「tar ボール」などと呼ばれるの。拡張子は「.tar.gz」のほかに、その短縮形として「.tgz」も使われるのでそれも覚えておいてね。

四色君：
compress 形式の圧縮アーカイブもパイプを使ってできますか？

マリー先生：
compress コマンドの場合も次のようにすれば OK よ。
拡張子を「.tar.Z」とすることを忘れないようにね。

```
$ tar -cvf - Backup | compress > Backup.tar.Z
```

練習問題

第 1 問：compress 形式の圧縮ファイル「info.txt.Z」を解凍するコマンドとして正しくないのはどれか？

a)　compress -d info.txt
b)　uncompress info.txt
c)　compress -k info.txt
d)　gunzip info.txt.Z

c)　compress -k info.txt
gunzip では compress 形式の圧縮ファイルを解凍することもできる。

第 2 問：カレントディレクトリの拡張子が「.txt」のすべてのファイルを gzip 形式で圧縮するコマンドはどれか？

a)　gzip *.txt
b)　gzip txt
c)　gzip -d txt
d)　gzipd

a)　gzip *.txt
圧縮コマンドの引数にはワイルドカードを使用してファイルを指定できる。

第 3 問：Pictures ディレクトリのアーカイブを「Pictures.tar」というファイル名で作成するコマンドはどれか？

a)　tar -xvf Pictures.tar Pictures/
b)　tar -tvf Pictures.tar Pictures/
c)　tar -cvf Pictures.tar Pictures/
d)　tar Pictures.tar Pictures/

c)　tar -cvf Pictures.tar Pictures/
アーカイブの作成は「-cvf アーカイブのパス」オプションを指定して tar コマンドを実行する。

第 4 問：Samples ディレクトリをまとめて tar + gzip 形式の圧縮アーカイブ「Samples.tar.gz」を作成するコマンドはどれか？

a)　tar -cvf - Samples | gzip > Samples.tar.gz
b)　tar -cvf Samples.tar.gz Samples/
c)　tar -cvf Samples | gzip -c > Samples.tar.gz
d)　tar -cvf - Samples | compress

a)　tar -cvf - Samples | gzip > Samples.tar.gz
圧縮アーカイブをひとつのコマンドラインで作成するには、tar コマンドと圧縮コマンドをパイプで接続する。

UNIX の教科書「応用編」

第 5 日目：ジョブとプロセスの操作

2008 年 12 月　大津 真

応用編も後半に入りました。今月はジョブとプロセスについて学びましょう。これまでは、あるコマンドを実行して完了するのを待ってから、次のコマンドを実行していました。ジョブとプロセスを理解すれば、処理に時間のかかるコマンドを実行している間に、他のコマンドを実行するといったことができるようになります。1 時間目はジョブの基礎について、2 時間目はジョブの管理とプロセスについて取り上げました。

マリー先生：
コマンドラインで実行中のプログラムはシェルからは「ジョブ」として管理されているの。今回はそのジョブと、システムから見た実行中のプログラムの単位であるプロセスの取り扱いについて説明していきましょう。その前に、前回学んだファイルの圧縮とアーカイブの操作についてなにか質問ありますか？

四色君：
「シェルアーカイブ」というアーカイブ形式があるというのを聞いたのですが……？

マリー先生：
そうね、シェルアーカイブは最近ではあまり使われないけど覚えておいて損はないでしょう。シェルのコマンドで記述されたプログラムのことをシェルスクリプトと呼ぶのだけど、シェルアーカイブは、シェルスクリプトを利用して作成されたアーカイブなの。

シェルアーカイブは、shar コマンドの引数にアーカイブに格納するファイルまたはディレクトリを指定して作成するの。結果は標準出力に書き出されるので、出力リダイレクション「>」を使用してファイルに書き出してね。たとえば、Documents ディレクトリのシェルアーカイブを「Documents.shar」として作成するには次のようにするの。

```
$ shar -cmos Documents > Documents.shar
```

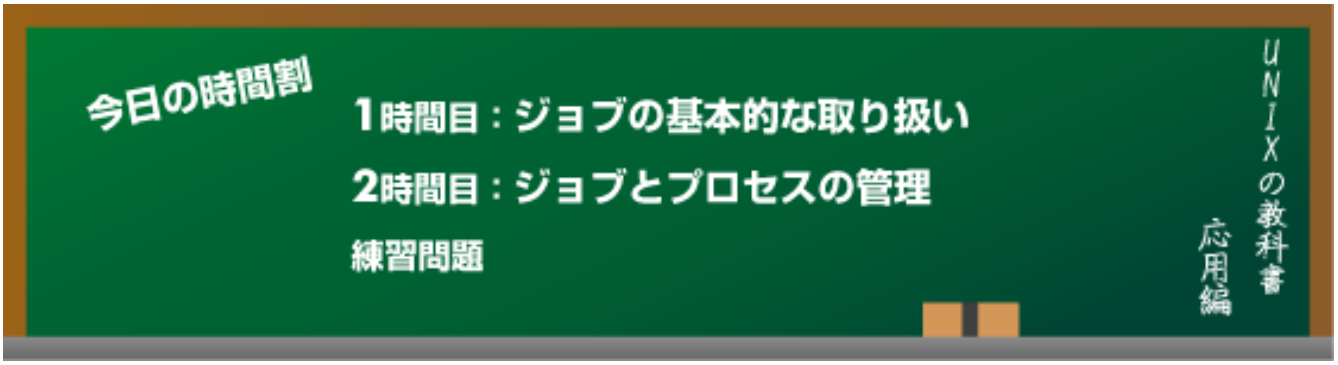
実際に見てみるとわかるけど、作成されたファイルはシェルのコマンドが羅列されたテキストファイルになるの。

＜Documement.shar の内容（抜粋）＞

```
LANG=""; export LANG
PATH=/bin:/usr/bin:/usr/sbin:/usr/ccs/bin:$PATH; export PATH
EXIT_STATUS=0

if sum -r </dev/null >/dev/null 2>&1
then
    sumopt='-r'
else
    sumopt=''
fi

echo mkdir - Documents
mkdir Documents

echo mkdir - Documents/2007
mkdir Documents/2007
```

マリー先生：
シェルアーカイブを解凍するには、次のように sh（シェル）の引数にアーカイブを指定してね。これでシェルアーカイブ内のシェルスクリプトが実行されてファイルが展開されるわ。

```
$ sh Documents.shar
mkdir - Documents
mkdir - Documents/2007
x - Documents/2007/ok.txt
x - Documents/2007/ng.txt
x - Documents/2007/readme.txt
mkdir - Documents/2008
～以下略～
```



1 時間目：ジョブの基本的な取り扱い

さて、これまで説明してきたいろいろなコマンドの実行例では、あるコマンドを実行してそれが完了するのを待ってから、次のコマンドを実行していました。それでは、たとえば find コマンドのように処理時間が長いコマンドの場合、実行完了までユーザは次の処理が行えなくなります。それに対して、この時間で説明するバックグラウンドジョブと呼ばれるシェルの機能を使用すると、同じターミナル内で複数のコマンドを同時に実行できるようになります。もちろん、X Window System の環境では、ターミナルのウィンドウを複数開いてそれぞれ別のコマンドを実行するといったこともできますが、バックグラウンドジョブは、別のコンピュータからリモートログインした状態でも使えますので、ぜひ覚えておきましょう。

フォアグラウンドジョブとバックグラウンドジョブ

シェルが管理するジョブは「フォアグラウンドジョブ」と「バックグラウンドジョブ」に大別されます。ここでは、X Window System で動作する xclock というアナログ時計を例に説明しましょう。通常、ユーザが実行したコマンドはフォアグラウンドジョブとなり、コマンドの完了までプロンプトは戻りません。

たとえば、

```
$ xclock
```

と実行すると時計のウィンドウが表示され、そのウィンドウを閉じるまでプロンプトは表示されず、次のコマンドは実行できないわけです。

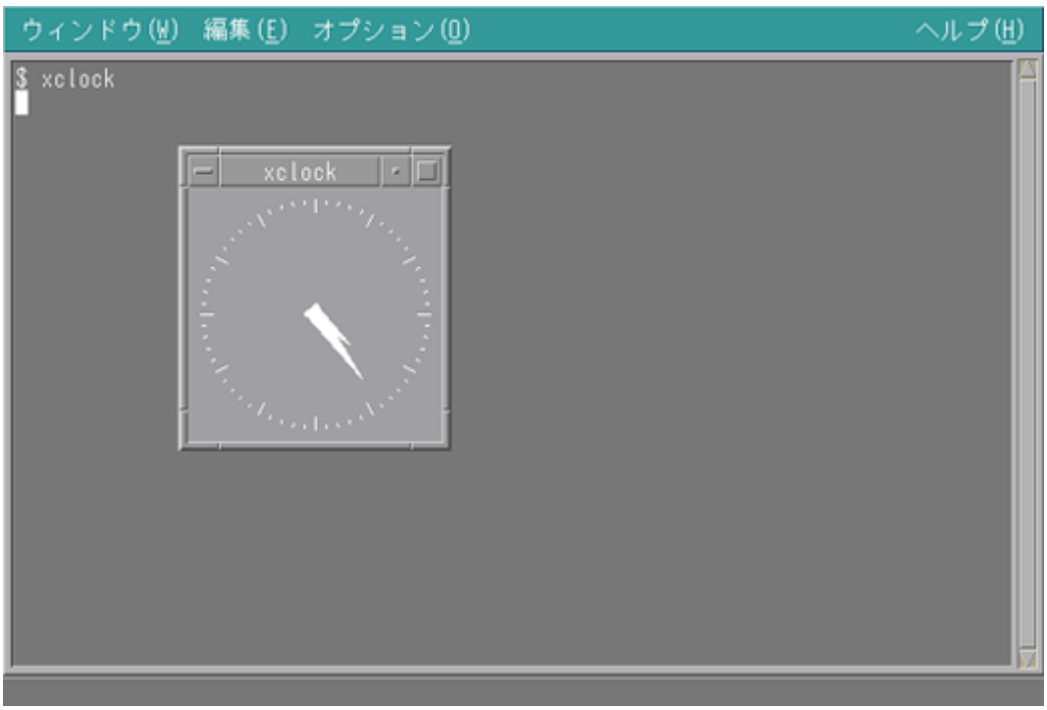


図 1：フォアグラウンドで xclock を実行するとプロンプトは表示されない

タックス君：
フォアグラウンドジョブをコマンドラインから終了させることはできますか？

マリー先生：
「Ctrl + C」キーを押すと強制終了できるわよ。

コマンドをバックグラウンドジョブとして実行する

それに対して、コマンドの後に「&」を付けて実行するとバックグラウンドジョブとなり、プロンプトはすぐに戻ります。[ ]内にジョブを識別するための「ジョブ番号」が表示されます。ジョブ番号は、実行した順に 1 から始まる整数となります。

```
$ xclock & 【Enter】
[1] ←ジョブ番号
$  ←プロンプトがすぐに戻る
```

四色君：
バックグラウンドジョブは、フォアグラウンドジョブと同じようにちゃんと実行を続けているのですか？

マリー先生：
もちろん。xclock の場合、背面で実行を続けているかどうかは時計の針が動くことで動作を確認できるでしょう。

```
$ xclock & 【Enter】
[2] ←ジョブ番号 2
$ xclock & 【Enter】
[3] ←ジョブ番号 3
```

実行中のジョブを確認する

現在実行中のジョブの一覧は、jobs コマンドで確認できます。

```
$ jobs 【Enter】
[3] +  Running                 xclock&
[2] -  Running                 xclock&
[1]    Running                 xclock&
```

この例では、バックグラウンドジョブとして 3 つの xclock が動作中です。ジョブ番号の後に「+」が表示されているのが「カレントジョブ」と呼ばれるジョブで、通常は最後に実行したジョブです。「-」が表示されているのが「プリビアスジョブ」（直前のジョブ）と呼ばれるジョブで、カレントジョブの 1 つ前に実行したジョブです。

マリー先生：
jobs はシェルの組み込みコマンドなのよ。

四色君：
組み込みコマンドって、何でしたっけ？

マリー先生：
UNIX のコマンドは、個別のコマンドファイルとして存在するものと、シェルに内蔵されている組み込みコマンドに大別されるの。あるコマンドが組み込みコマンドかどうかは、そのコマンド名を引数に type コマンドを実行すると調べられるわ。コマンドファイルの場合にはその絶対パスが、組み込みコマンドの場合には「shell builtin」と表示されるの。

```
$ type ln 【Enter】
ln is /usr/bin/ln ←コマンドファイル
$ type jobs 【Enter】
jobs is a shell builtin. ←組み込みコマンド
```

休憩時間：システム状況を表示する top コマンド

現在の、CPU の負荷状況、メモリの使用状況、およびプロセスの一覧を表示するコマンドに top があります。top コマンドを実行すると、CPU とメモリの状況、その下に CPU の使用率の高い順にプロセスが表示されます。デフォルトでは、表示は 5 秒おきに自動的に更新されていきます。 top コマンドを終了するには Q キーを押します。

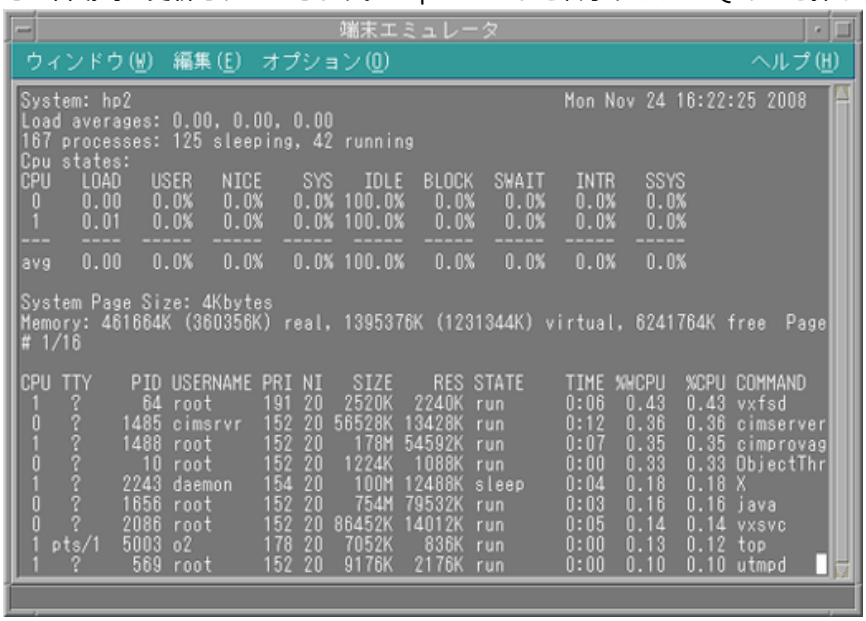


図 2：top コマンドでシステム状況を表示する

2 時間目：ジョブとプロセスの管理

フォアグラウンドジョブとバックグラウンドジョブが理解できたところで、ジョブの停止やフォアグラウンドジョブとバックグラウンドジョブの切り替えといった管理について説明しましょう。また、システムにおける実行中のプログラムの単位であるプロセスについても説明します。

フォアグラウンドジョブを一時停止する

フォアグラウンドジョブを実行中に別のコマンドを実行したくなった場合には、いったんフォアグラウンドジョブを一時停止（サスペンド）させた後に、バックグラウンドジョブとして再開します。その後で、別のコマンドを実行すれば両方のコマンドを同時に実行できます。
フォアグラウンドジョブを一時停止するには「Ctrl + Z」キーを押します。すると「Stopped」と表示されます。

```
$ xclock
 ←Ctrl + Z を入力 【Enter】
[1]+  Stopped                 xclock
```

この状態で jobs コマンドを実行すると、一時停止されたジョブがカレントジョブであることがわかります。

```
$ jobs 【Enter】
[1]+  Stopped                 xclock ←カレントジョブとなる
```

四色君：
一時停止されたジョブは完全に止まっているのですか？

マリー先生：
そうよ、「Ctrl + Z」キーが押された時点の状態で停止しているの。xclock の場合、時計の針が動かなくなることで確認できるわ。

一時停止中のジョブを再開する

一時停止中のジョブをバックグラウンドジョブとして再開するには、bg コマンドを使います。

```
bg %ジョブ番号
```

引数にはジョブ番号を「%ジョブ番号」の形式で指定します。ジョブ番号「1」のジョブをバックグラウンドジョブとして再開するには、次のようにします。

```
$ bg %1 【Enter】  ←バックグラウンドジョブとして再開
[1]    xclock
$ jobs 【Enter】   ←jobs コマンドで確認
```

```
[1] +  Running                 xclock ←ステータスが「Running」になった
```

バックグラウンドジョブをフォアグラウンドジョブにする

逆に、指定したバックグラウンドジョブをもしくは一時停止中のジョブを、フォアグラウンドジョブにするには fg コマンドを使います。

```
fg %ジョブ
```

たとえば、ジョブ番号「2」のバックグラウンドジョブを、フォアグラウンドジョブにするには次のようにします。

```
$ fg %2 【Enter】
xclock
```

タックス君：
bg コマンドと fg コマンドを、引数を指定しないで実行するとどうなるのですか？

マリー先生：
引数を指定しないか「%+」を指定しても、カレントジョブが対象となるの。プリビアスジョブを対象とするには「%-」を指定してもいいわ。

ジョブとプロセス

個々のターミナルで実行中のコマンドは、それぞれのシェルで個別にジョブとして扱われます。つまり、複数のターミナルでコマンドを実行した場合、それぞれのターミナルでジョブ番号が 1 から順に割り当てられます。それに対して、システムから見たプログラムの実行単位のことを「プロセス」と呼びます。個々のプロセスには「プロセス ID」と呼ばれる番号が割り当てられますが、ジョブ番号と異なりシステム内で重複はありません。たとえば、2 つのターミナルを開いてそれぞれで xclock を起動した場合、どちらもジョブ番号は「1」となりますが、プロセス ID は異なる番号になります。

jobs コマンドに、「-l」オプションをつけて実行すると、ジョブ番号とプロセス ID の両方を確認できます。

```
$ jobs -l 【Enter】
[3] + 5574      Running                 xclock &
[2] - 5565      Running                 xclock &
[1]   5561      Running                 xclock &
ジョブ番号  プロセスID
```

実行中のプロセス一覧を表示する

現在実行中のプロセスの一覧を表示するには ps コマンドを使います。引数なしで実行すると、ターミナル内で実行されているプロセスの一覧が表示されます。次に、xclock をバックグラウンドジョブとして実行している状態での ps コマンドの結果を示します。PID のフィールドにプロセス ID が表示されています。

```
$ ps 【Enter】
  PID TTY          TIME COMMAND
 7345 pts/0    0:00 ps
 7344 pts/0    0:00 xclock
 5547 pts/0    0:00 sh
```
プロセスIDが表示されている

タックス君：
ps コマンド自体もプロセスとして表示されるのですね。

マリー先生：
そうよ。それと「sh」と表示されているのがシェルね。シェルもシステム的には単なるコマンドファイルにすぎず、ひとつのプロセスなの。「TTY」のフィールドにはプロセスの実行されているターミナルの名前が表示されるわ。

なお、システム内のすべてのプロセス一覧を詳しく表示するには「-ef」オプションを指定して実行します。

```
$ ps -ef 【Enter】
   UID   PID  PPID  C    STIME TTY      TIME COMMAND
  root     0     0  0 15:11:09 ?       0:14 swapper
  root     1     0  0 15:10:36 ?       0:00 init
  root    13     0  0 15:10:51 ?       0:00 net_str_cached
  root    12     0  0 15:10:51 ?       0:00 net_str_cached
  root    11     0  0 15:10:51 ?       0:00 escsid
  root    10     0  0 15:10:36 ?       0:00 ObjectThreadPool
～中略～
    o2  4182  4175  0 15:38:10 pts/2    0:00 /sbin/sh
    o2  3993  3991  0 15:30:40 ?       0:00 /usr/dt/bin/dtsession
    o2  4175  4174  0 15:37:54 ?       0:00 /usr/dt/bin/dtterm
～以下略～
```

四色君：
TTY のフィールドが「?」になっているプロセスがたくさんありますね。

マリー先生：
「?」が表示されているプロセスは、特定のターミナルと結びついていないプロセスなの。たとえば、通常システムの起動時に起動してバックグラウンドで動作し続ける「デーモン」と呼ばれるプロセスがそうね。あとは、デスクトップのメニューから起動したアプリケーションも「?」と表示されるわ。

タックス君：
PPID のフィールドに表示されている番号はなんですか？

マリー先生：
プロセスには親子関係があるのだけど、PPID は「Parent Process ID」の略で、そのプロセスを起動したプロセス、つまり親プロセスのプロセス ID なの。

マリー先生：
ところで、2 つのコマンドをパイプで接続して実行したような場合には、ジョブとプロセスではどのように扱われると思う？

タックス君：
2 つのコマンドを実行しているわけだから、2 つのジョブ、2 つのプロセスとなるのではないでしょうか？

マリー先生：
実はジョブとしてはひとつだけど、プロセスとしては 2 つになるの。例として、xclock と grep コマンドをパイプで接続して実行してみましょう。

```
$ xclock | grep "test" & 【Enter】
[2]    7241
```

マリー先生：
xclock は標準出力にメッセージを出さないからこのコマンドライン全体に意味はないけど、ジョブとプロセスの確認にはなるでしょう。次のように jobs コマンドと ps コマンドで確認してね。

```
$ jobs 【Enter】
[2] +  Running                 xclock | grep "test" &
↑ジョブとしてはひとつ
$ ps 【Enter】
  PID TTY      TIME COMMAND
 7241 pts/0    0:00 grep   ←コマンドごとにプロセスになる
 7242 pts/0    0:00 xclock ←コマンドごとにプロセスになる
```

```
5547 pts/0    0:00 sh
7255 pts/0    0:00 ps
```

プロセスにシグナルを送信する

実行中のジョブまたはプロセスは、「シグナル」と呼ばれるメッセージを受け取ります。たとえば、フォアグラウンドジョブを実行中に「Ctrl + C」キーを押すとジョブが強制終了されますが、これは「Ctrl + C」キーが、「SIGINT」というコマンドを終了させるシグナルの送信にアサインされているからです。また、フォアグラウンドジョブを一時停止させる「Ctrl + Z」キーでは、「SIGTSTP」というコマンドを一時停止させるシグナルが送られます。

コマンドラインでシグナルを送るには、kill コマンドを使用します。

```
kill -s シグナル プロセスもしくはジョブ
```

ジョブを送り先にするには「%ジョブ番号」を指定します。プロセスを送り先にするには「プロセス ID」を指定します。シグナルはシグナル名もしくはシグナル番号で指定します。
なお、「-s シグナル」を省略した場合には、プロセスを終了させる「SIGTERM」というシグナルが送られます。たとえば、ジョブ番号「1」のジョブに SIGTERM シグナルを送信して終了するには次のようにします。

```
$ kill %1 【Enter】
$ 【Enter】 ←もう一度 Enter キーを押す
[1] - Terminated              xclock & ←終了したメッセージが表示される
```

なお、「SIGINT」や「SIGTERM」を送っても終了できない、いわゆる暴走状態のプログラムを終了させるには「SIGKILL」という強制終了のシグナルを送ります。
たとえば、プロセス ID が「7990」のプロセスに SIGKILL シグナルを送るには次のようにします。

```
$ kill -s SIGKILL 7990
```

四色君：
「ps -ef」で表示される任意のプロセスにシグナルを送れるのですか？

マリー先生：
システムのプロセスや、他人のプロセスが勝手に終了されては困るから、一般ユーザがシグナルを送れるのは自分のプロセスだけなの。

タックス君：
シグナルは、いろいろな種類があるのですか？

マリー先生：
そうね、ここで紹介した以外だと、たとえば「SIGCONT」は一時停止中のプロセスを再開させるシグナルなの。「kill -l」を実行するとシグナルの一覧が表示できるわ。シグナル名の最初の「SIG」は省略されて表示されるの。あと、左に表示される番号はシグナル番号ね。

```
$ kill -l 【Enter】
 1) HUP
 2) INT
 3) QUIT
 4) ILL
 5) TRAP
 6) ABRT
 7) EMT
～以下略～
```

練習問題

第 1 問：xclock コマンドをバックグラウンドジョブとして実行するコマンドはどれか？

a)　xclock
b)　xclcok &
c)　&xclcok
d)　xclock $

b)　xclcok &
バックグラウンドジョブとして実行するには、コマンドの後ろに「&」を記述する。

第 2 問：ジョブ番号「2」のジョブが一時停止している時に、そのジョブをバックグラウンドジョブとして再開させるコマンドはどれか？

a)　fg %2
b)　bg 2
c)　fg 2
d)　bg %2

d)　bg %2
bg コマンドは一時停止中のジョブをバックグラウンドジョブとして再開するコマンド。ジョブ番号は「%ジョブ番号」で指定する。

第 3 問：現在実行中のジョブを一覧表示するコマンドはどれか？

a)　ps
b)　ls -j
c)　jobs
d)　showj

c)　jobs
jobs は実行中のジョブを一覧表示する組み込みコマンド。なお、a)は実行中のプロセスを一覧表示するコマンドである。

第 4 問：ジョブ番号が 2 のバックグラウンドジョブを一時停止させるコマンドはどれか？

a)　kill %2
b)　kill -s SIGTERM %2
c)　kill -s SIGKILL 2
d)　kill -s SIGTSTP %2

d)　kill -s SIGTSTP %2
ジョブを一時停止させるには kill コマンドで SIGTSTP シグナルを送ればよい。

UNIX の教科書「応用編」

第 6 日目：シェルの環境設定

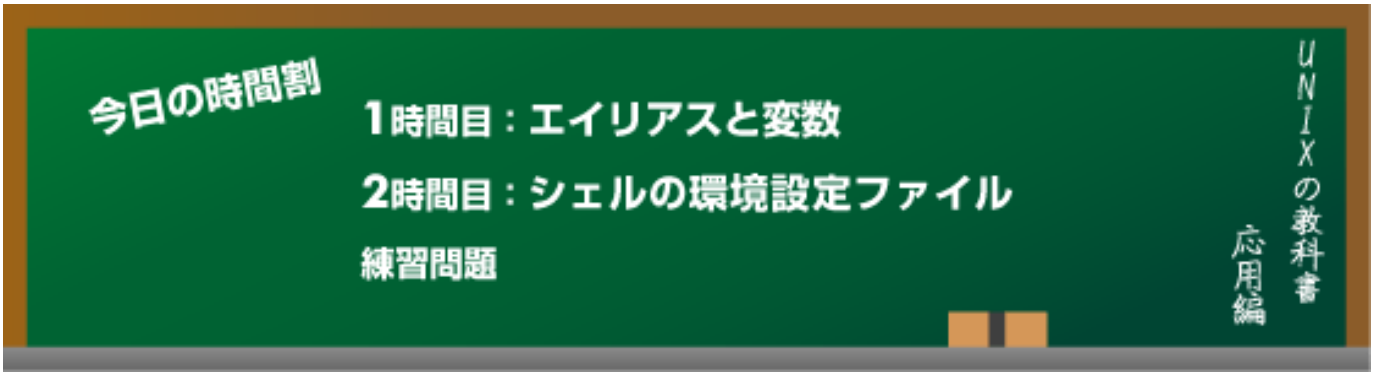
2009 年 2 月　大津 真

たいぶシェルを使いこなせるようになってきたでしょうか。今回は、使いやすいようにシェルの環境設定をカスタマイズする方法がテーマです。まず 1 時間目では、シェルの環境設定において重要な役割を果たすエイリアスと変数の取り扱いについて説明します。2 時間目では、ログインするたびに同じようにカスタマイズした環境を使えるようにする、シェルの環境設定ファイルの取り扱いについて学んでいきましょう。

マリー先生：
シェルは、使いやすいように必要に応じて環境をカスタマイズすることができるの。今回はシェルの環境設定に関する機能について説明しましょう。その前に、前回のジョブとプロセスの操作についてなにか質問ありますか？

タックス君：
デスクトップで暴走状態にある GUI アプリケーションを強制終了するためには、その都度 ps コマンドでプロセス番号を調べて、kill コマンドで SIGKILL シグナルを送る必要があるのですか？

マリー先生：
それでもいいけれど、X Window System には「xkill」という、X アプリケーションを強制終了させる専用のプログラムが用意されているの。「xkill 【Enter】」を実行すると、端末に「Select the window whose client you wish to kill with button 1....」と表示されるので、目的のアプリケーションのウィンドウをマウスでクリックしてね。これで、そのアプリケーションが強制終了されるわ。



1 時間目：エイリアスと変数

1 時間目では、シェルに用意されている機能であるエイリアスと、シェル変数の取り扱いについて説明しましょう。なお、現在ではさまざまなシェルがあり、機能や使い方が少しずつ異なりますが、ここでは HP-UX のデフォルトのシェルである POSIX シェルを基本に解説します。

エイリアスとは

エイリアスとは、コマンドを別の名前で定義する機能です。alias コマンドを使用して、次のような書式で実行します。

```
alias エイリアス名=コマンド
```

たとえば、ファイルを削除する rm コマンドのエイリアスとして「del」を設定するには、次のようにします。

```
$ alias del=rm
```

これで、rm コマンドの代わりにエイリアス「del」が使用できるようになります。

```
$ del tmp.txt ←rm の代わりに del でファイルを削除できる
```

四色君：
あれ、次のように実行したらエラーになるのですが……。

```
$ alias del = rm
del=rm
sh: =: Invalid alias name.
```

マリー先生：
それは「=」の前後にスペースを入れているからね。alias コマンドは「エイリアス名=コマンド」をひとつの引数として扱うから「=」の前と後にはスペースやタブをいれてはダメなの。

コマンドの内部にスペースや特殊文字を含む場合には、コマンド部分をシングルクォーテーション「'」（もしくはダブルクォーテーション「"」）で囲ってクォーティングします。たとえば、「ls /home/o2/Documents」というコマンドのエイリアスを「lsdocs」として定義するには次のようにします。

```
$ alias lsdocs='ls /home/o2/Documents' ←エイリアス「lsdocs」を定義
$ lsdocs 【Enter】 ←lsdocs を実行
2007    info.txt   mail.txt   sample
2008    ln         main.html  sample.txt
```

コマンドを再定義する

エイリアスを使用すると、既存のコマンドをオプションや引数を含めて同じ名前で再定義することもできます。たとえば、cp コマンドを、コピー先のファイルが存在する場合は常に確認のメッセージを表示するようにするには、次のようにして cp コマンドを「-i」オプション付きで再定義します。

```
$ alias cp="cp -i" ←cp コマンドを「-i」オプション付きで再定義
$ cp sample.txt new.txt 【Enter】 ←再定義した cp コマンドを実行
overwrite new.txt? (y/n) y
↑コピー先のファイルが存在したら確認のメッセージが表示される（「Y」キーで削除）
```

四色君：
コマンドがエイリアスかどうかを確認するにはどうしたらよいのですか？

マリー先生：
type コマンドを実行すると、エイリアスの場合には「alias」と表示されるの。通常のコマンドはその絶対パスが、シェルの組み込みコマンドの場合には「shell builtin」と表示されるわ。

```
$ type cp 【Enter】
cp is an alias for cp -i ←cp は「cp -i」のエイリアス
$ type ls 【Enter】
ls is a tracked alias for /usr/bin/ls ←コマンドファイル
$ type cd 【Enter】
cd is a shell builtin. ←組み込みコマンド
```

マリー先生：
それから、alias コマンドを引数なしで実行すれば、現在設定されているエイリアスの一覧が表示されるわ。

```
$ alias 【Enter】
autoload='typeset -fu'
command='command '
cp='cp -i'
del=rm
functions='typeset -f'
history='fc -l'
integer='typeset -i'
local=typeset
ls=/usr/bin/ls
lsdocs='ls /home/o2/Documents'
nohup='nohup '
r='fc -e -'
rm=/usr/bin/rm
stop='kill -STOP'
suspend='kill -STOP $$'
type='whence -v'
```

エイリアスを削除する

設定済みエイリアスを削除するには、unalias コマンドを使用します。

```
unalias エイリアス名
```

たとえば、前の例の「cp -i」を「cp」として再定義したエイリアスを削除するには次のようにします。

```
$ unalias cp ←cp のエイリアスを削除
$ type cp 【Enter】 ←type コマンドで確認
cp is /usr/bin/cp ←コマンドファイルが使用されるようになる
```

変数とは

一般的なプログラミング言語において、変数とはなんらかの値を格納しておく箱のようなものです。その値には変数名と呼ばれる名前でアクセスできます。シェルはユーザーインターフェースであると同時にプログラミング言語でもあるため、変数が使用できます。シェルで使用する変数のことを「シェル変数」と呼びます。
シェル変数に値を代入するには次の書式で実行します。

```
変数名=値
```

たとえば「val1」という名前の変数に「Hello」という文字列を格納するには、次のようにします。

```
$ val1=Hello
```

タックス君：
エイリアスの定義と同じく、「=」の前後にはスペースを入れてはダメなのですね。

マリー先生：
そうね。それから、これもエイリアスの定義と同じく、値にスペースや特殊文字を含む場合にはクォーティングしてね。

```
$ val2="Hello World"
```

もしくは、次のようにスペースの前に「\」を記述してエスケープしてもいいわね。

```
$ val2=Hello\ World
```

変数の値を取り出す場合には、変数名の前に「$」を記述します。

```
$変数名
```

たとえば、変数「val2」の値を echo コマンドで表示するには次のようにします。

```
$ echo $val2 【Enter】
Hello World
```

変数を削除するには unset コマンドを使います。

```
unset 変数名
```

このとき変数名の前には「$」は記述しません。次に変数「val2」を削除する例を示します。

```
$ unset val2
```

シェル変数の活用例

たとえば、よく使うディレクトリのパスを変数に格納しておくと、そのディレクトリに簡単に移動したり、一覧を表示したりできます。
次に例を示します。

```
$ myDir="/home/o2/Documents/2009/info" ←変数 myDir にパスを代入
$ ls $myDir ←ls コマンドで一覧を表示 【Enter】
new.txt  samples
$ cd $myDir ←cd コマンドで移動
```

四色君：
現在設定されている変数の一覧を表示することはできますか？

マリー先生：
set コマンドを引数なしで実行すれば、すべての変数とその値が表示されるわ。

```
$ set 【Enter】
COLUMNS=80
DISPLAY=192.168.1.14:10.0
EDITOR=vi
～以下略～
```

休憩時間：HP-UX から他のシステムにリモートログインするには

読者のみなさんの中には、Windows 用の端末エミュレーターである「TeraTerm Pro」などを利用して Windows マシンから HP-UX にリモートログインして学習をしている方も多いでしょう。
HP-UX には標準で Telnet クライアント/SSH クライアントが用意されていますので、Telnet サーバーあるいは SSH サーバーにログインすることも可能です。

Telnet サーバーにリモートログインを行うには telnet コマンドを次の書式で実行します。

```
telnet ホスト名もしくは IP アドレス
```

SSH サーバーにログインする場合には ssh コマンドを次の書式で実行します。

```
ssh -l ユーザー名 ホスト名もしくは IP アドレス
```

2 時間目：シェルの環境設定ファイル

エイリアスや変数の設定はログアウトするとクリアされます。毎回同じ設定を行いたい場合には、ログイン時に自動的に読み込まれるシェルの環境設定ファイルに記述しておくとよいでしょう。この時間では、まずシェル環境を設定する環境変数と呼ばれる変数について説明し、次に環境設定ファイルの概要について説明します。

環境変数とは

シェル変数はそのシェルの内部でのみ使うことができる変数で、シェルの設定を変更したり、シェルスクリプトで変数に使ったりします。

一方、環境変数はシェルを含むすべての UNIX プロセスで持つことができる変数で、シェルやシェルから起動したコマンドの動作を変えるのに使います。あるプロセスから別のプロセスを起動した場合、起動した側のプロセスを親プロセス、起動された側のプロセスを子プロセスと呼びます。シェルからコマンドを起動した場合には、シェルが親プロセス、コマンドが子プロセスとなります。通常のシェル変数と環境変数の相違はその値が子プロセスに引き継がれるかどうかであり、環境変数のみが引き継がれます。たとえば、シェルからコマンドを実行した場合、シェルで設定された環境変数は引き継がれますので、コマンドはその環境変数に応じた動作を行えるわけです。

現在設定されている環境変数の一覧は printenv コマンドで確認できます。

```
$ printenv 【Enter】
_=/usr/bin/printenv
MANPATH=/usr/share/man/%L:/usr/share/man:/usr/contrib/man/%L:/usr/contrib/man:/usr/local/man/%L:/usr
/local/man:/opt/ldapux/share/man/%L:/opt/ldapux/share/man:/opt/ipf/man:/opt/ldapux/ypldapd/man:/opt/
samba/man:/opt/samba/WTEC_Support_Tools/man:/opt/samba/cfsm_man:/opt/cifsclient/share/man:/opt/ope
nssl/man:/opt/openssl/prngd/man:/opt/wbem/share/man:/opt/graphics/common/man:/opt/amgr/man:/opt
/amgr/man/%L:/opt/sec_mgmt/share/man:/usr/dt/share/man:/opt/drd/share/man/%L:/opt/drd/share/man:/o
pt/dsau/man:/opt/resmon/share/man/%L:/opt/resmon/share/man:/opt/gnome/man:/usr/contrib/kwdb/share
/man:/opt/perl_32/man:/opt/perl_64/man:/opt/sfmdb/pgsql/man:/opt/sfm/share/man:/opt/swm/share/man
/%L:/opt/swm/share/man:/opt/sec_mgmt/share/man/%L:/opt/ssh/share/man:/opt/swa/share/man/%L:/opt/s
wa/share/man:/opt/VRTS/man:/opt/gwlm/man/%L:/opt/gwlm/man:/opt/ignite/share/man/%L:/opt/ignite/sh
are/man:/opt/hpsmdb/pgsql/man:/opt/mx/share/man/%L:/opt/mx/share/man:/opt/perf/man/%L:/opt/perf/
man:/opt/prm/man/%L:/opt/prm/man:/opt/wlm/share/man/%L:/opt/wlm/share/man:/opt/mpi/share/man:/o
pt/vse/man/%L:/opt/caliper/man/%L:/opt/caliper/man:/opt/mlib/share/man:/opt/sentinel/man/%L:/opt/senti
nel/man:/opt/spb/share/man:/opt/langtools/share/man/%L:/opt/langtools/share/man
SSH_TTY=/dev/pts/0
PATH=/usr/bin:/usr/ccs/bin:/usr/contrib/bin:/usr/contrib/Q4/bin:/opt/perl/bin:/opt/ipf/bin:/opt/nettladm/bin
:/opt/fcms/bin:/opt/wbem/bin:/opt/wbem/sbin:/opt/sas/bin:/opt/graphics/common/bin:/opt/atok/bin:/usr/
bin/X11:/usr/contrib/bin/X11:/opt/sec_mgmt/bastille/bin:/opt/drd/bin:/opt/dsau/bin:/opt/dsau/sbin:/opt/res
mon/bin:/opt/gnome/bin:/usr/contrib/kwdb/bin:/opt/firefox:/opt/mozilla:/opt/perl_32/bin:/opt/perl_64/bin:
/opt/sfm/bin:/opt/swm/bin:/opt/sec_mgmt/spc/bin:/opt/ssh/bin:/opt/swa/bin:/opt/hpsmh/bin:/opt/gwlm/bi
n:/opt/mx/bin:/opt/perf/bin:/opt/prm/bin:/opt/wlm/bin:/opt/mpi/bin:/opt/vse/bin:/opt/caliper/bin:/opt/igni
te/bin:/var/opt/netscape/server7/shared/bin:/var/opt/netscape/server7/bin:/opt/sentinel/bin:/opt/java1.4/jre
/bin:/opt/spb/bin:/opt/thunderbird:/opt/langtools/bin:.
```

```
COLUMNS=80
EDITOR=vi
LOGNAME=o2
MAIL=/var/mail/o2
SFTP_UMASK=
ERASE=^H
SFTP_PERMIT_CHOWN=1
USER=o2
DISPLAY=192.168.1.14:10.0
SHELL=/sbin/sh
HOME=/home/o2
SSH_CONNECTION=192.168.1.102 52508 192.168.1.14 22
SSH_CLIENT=192.168.1.102 52508 22
TERM=xterm
PWD=/home/o2/Documents/2009/info
TZ=JST-9
SFTP_PERMIT_CHMOD=1
LINES=24
```

たとえば、環境変数「LOGNAME」と「USER」にはユーザー名が、「TZ」にはタイムゾーンが格納されています。また、PATH にはコマンドが保存されているディレクトリの絶対パスがコロン「:」で区切られて格納されています。

四色君：
特定の環境変数の値だけ表示することもできますか？

マリー先生：
それには引数に環境変数名を指定して printenv コマンドを実行すればいいわ。たとえば、「EDITOR」はデフォルトのエディターを示す環境変数だけど、その値を表示するには次のようにするの。

```
$ printenv EDITOR 【Enter】
vi
```

なお、通常のシェル変数と同じく echo コマンドでも値を表示できるけど、その場合変数名の前に「$」が必要な点に注意してね。

```
$ echo $EDITOR 【Enter】
vi
```

シェル変数を環境変数にする

通常のシェル変数を環境変数にするには export コマンドを使用します。

```
export 変数名
```

このことを「変数をエクスポートする」と言います。あるいは、次のようにすることで値の設定とエクスポートとを同時に行えます。

```
export 変数名=値
```

環境変数の名前は通常のシェル変数と区別するために、すべて大文字にするとよいでしょう。たとえば、変数「ENV1」を環境変数としてエクスポートするには次のようにします。

```
$ export ENV1="hello hp-ux"  ←変数「ENV1」をエクスポート
$ echo $ENV1  ←echo コマンドで確認
```

タックス君：
環境変数が子プロセスに引き継がれるっていう意味がよくわからないのですが……。

マリー先生：
環境変数をエクスポートした後にシェルを起動して確認してみると、イメージがつかめると思うわよ。

```
$ export ENV1="hello hp-ux"  ←(1)
$ sh  ←(2)
$ printenv ENV1  【Enter】  ←(3)
hello hp-ux
$ exit  ←シェルを終了
```

この例では、(1)で環境変数「ENV1」設定して、(2)で別のシェルを子プロセスとして起動しているの。(3)で子プロセスのほうのシェルで printenv コマンドを実行すると値が表示されることから、環境変数が引き継がれているとわかるわけ。
同じことを通常のシェル変数でやってみると、子プロセスに引き継がれないことを確認できるわよ。

```
$ val1="hello unix"  ←通常の変数に値を設定
$ sh  ←シェルを起動
$ echo $val1  【Enter】
  ←なにも表示されない
$ exit  ←シェルを終了
```

環境設定ファイルの種類

環境変数などをシェルの環境設定ファイルに記述しておくと、ログイン時に自動的に読み込まれて毎回同じ設定でシェルを使用できます。POSIX シェルの主な環境設定ファイルは次の 2 つです。

```
~/.profile
/etc/profile
```

/etc/profile はすべてのユーザーに共通の環境設定を、ホームディレクトリに保存された「~/.profile」ではユーザーごとの環境設定を行います。
たとえば、初期状態の/etc/profile には次のような変数を設定する行があります。

```
PATH=/usr/bin:/usr/ccs/bin:/usr/contrib/bin:/usr/contrib/Q4/bin:/opt/perl/bin
MANPATH=/usr/share/man:/usr/contrib/man:/usr/local/man
```

どちらも環境変数ですが、最初の行はコマンド検索パスを、2 番目の行はマニュアルの保存先を設定します。

「~/.profile」に設定を記述する

通常、ユーザーごとのエイリアスや環境設定は「~/.profile」に記述します。ただし、初期状態では「-r--r--r」（444）、つまり、すべてのユーザーに対してリードオンリーに設定されているため編集できません。

```
$ ls -l ~/.profile
-r--r--r--   1 o2       users        700 May 21  2008 .profile
```

そのため次のように chmod コマンドを実行し、所有者に対して書き換えを許可した上で、vi エディターなどで編集を行います。

```
$ chmod u+w ~/.profile
```

「~/.profile」の設定をテストする

「~/.profile」の設定をテストするには、いったんログアウトしてログインし直してもかまいませんが、「~/.profile」の設定を誤ると、ログインできないといった問題が起こる危険性があります。
そこで、通常はログインし直す代わりに、引数で指定したファイルからコマンドを読み込んで、現在のシェル環境で実行する「.（ピリオド）」コマンドを使用するとよいでしょう。

```
. ファイルのパス
```

「.」コマンドの引数に「~/.profile」を指定して実行すると、ファイル内に記述した環境変数などの動作を確かめることができます。たとえば、次のような環境変数「HP」の設定を「~/.profile」に加えたとしましょう。

```
export HP="hp-ux"
```

次のように「.」コマンドを実行することで、環境変数「HP」が正しく設定されたことが確認できます。

```
$ . ~/.profile
$ printenv HP 【Enter】
hp-ux ←追加した環境変数の値が表示される
```

四色君：
「.」コマンドを使わないで、次のように sh コマンドの引数に環境設定ファイルのパスを指定してもいいのですか？

```
$ sh ~/.profile
```

マリー先生：
そうすると、子プロセスとしてシェルが起動するので、現在のシェルには環境変数などの設定が反映されないのでダメなの。あと、環境設定ファイルはシェルの種類によって異なるから注意してね。

タックス君：
シェルの種類によってファイル名が違うということですか？

マリー先生：
そう。たとえば csh というシェルの場合には /etc/csh.login と「~/.cshrc」、「~/.login」といったファイルが環境設定ファイルとして使用されるの。さらに、POSIX シェルと csh ではエイリアスや変数の設定方法も異なるため、ファイルの中身の互換性もないので環境設定ファイルを共有することもできないわ。
もう 1 点、X Window System の CDE 環境を使用している場合には、「~/.dtprofile」という環境設定ファイルが使用されるので、これも注意してね。

四色君：
「~/.profile」は読み込まれないのですか？

マリー先生：
そう。もし「~/.profile」を自分でカスタマイズしたりしていて、その設定を CDE 環境の端末などにも反映させたければ「~/.dtprofile」の最後を次のように変更する必要があるの。

```
# DTSOURCEPROFILE=true
↓書き換える
 DTSOURCEPROFILE=true
```

「#」はその行がコメントであることを表しているの。「#」を削除することにより、DTSOURCEPROFILE という変数が「true」に設定され、「~/.profile」が読み込まれるようになるわけね。

練習問題

第 1 問：「cp -r」コマンドのエイリアスとして「cpDir」を定義するコマンドはどれか？

a)　alias cpDir=cp -r
b)　alias cpDir='cp -r'
c)　cpDir="cp -r"
d)　alias cpDir "cp -r"

b)　alias cpDir='cp -r'
スペースを含むコマンドのエイリアスを定義するにはクォーティングする必要がある。

第 2 問：エイリアス「cpDir」を削除するコマンドはどれか？

a)　unalias cpDir
b)　alias -d cpDir
c)　delete cpDir
d)　cpDir=""

a)　unalias cpDir
エイリアスを削除するには unalias コマンドを使用する。

第 3 問：変数「MYENV」に値として「SAMPLE」を代入し環境変数にするコマンドはどれか？

a)　MYENV="SAMPLE"
b)　export MYENV="SAMPLE"
c)　export $MYENV="SAMPLE"
d)　env MYENV="SAMPLE"

b)　export MYENV="SAMPLE"
変数を環境変数にするには export コマンドでエクスポートする。

第 4 問：環境設定ファイル「~/.profile」の動作を確かめるコマンドはどれか？

a)　sh ~/.profile
b)　~/.profile
c)　exec ~/.profile
d)　. ~/.profile

d)　. ~/.profile
環境設定ファイルのようなコマンドファイルを現在のシェル環境で実行するにはピリオド「.」コマンドを使用する。

UNIX の教科書「応用編」

第 7 日目：シェルスクリプトでより便利に

2009 年 4 月　大津 真

7 回にわたってお送りしてきた応用編も、最終回となりました。今月はシェルを利用したプログラムであるシェルスクリプトについて学びます。プログラムといっても難しく考える必要はありません。まずは、よく使う処理をファイルに保存しておいてコマンドとして呼び出せるようにしてみましょう。シンプルなシェルスクリプトの書き方から始めて、今日の授業の終わりにはファイルの拡張子をまとめて変換するシェルスクリプトを作成できるようになります。

マリー先生：
「UNIX の教科書」応用編も今日が最後。みんなよくがんばったわね。今回は、シェルを使用したプログラムである「シェルスクリプト」の基本について説明しましょう。その前に、前回のシェルの環境設定に関してなにか質問ありますか？

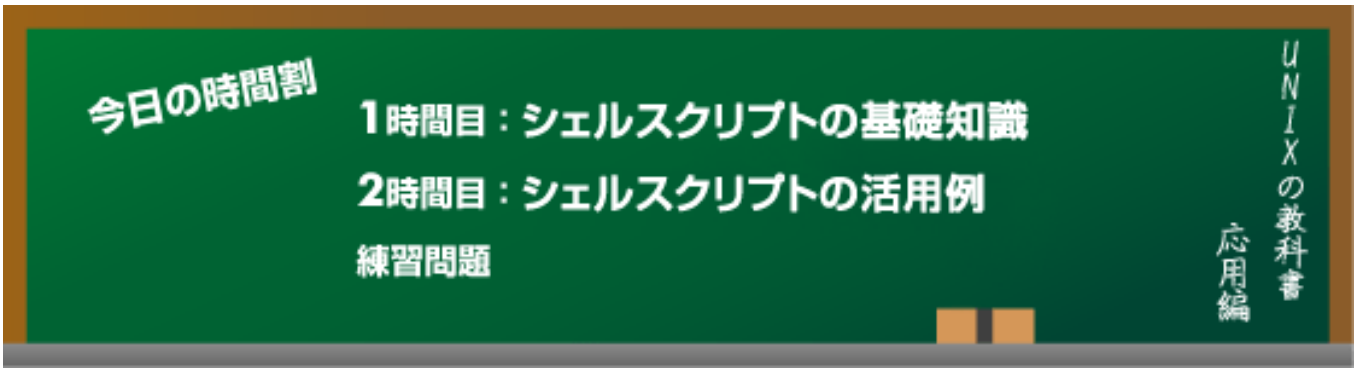
タックス君：
えーと、スペースや特殊文字の働きを打ち消すのにクォーティングをしましたが、シングルクォーテーション「'」と、ダブルクォーテーション「"」って違いはあるのですか？

マリー先生：
まず、シングルクォーテーション「'」とダブルクォーテーション「"」のどちらかを内部に含む場合には、逆のクォーテーションでクォーティングするの。たとえば、シングルクォーテーション「'」を内部に含む「I'm happy」を変数「myPhrase」に代入するには、ダブルクォーテーション「"」でクォーティングして次のようにすればいいわけ。

```
$ myPhrase="I'm happy"
```

あと、クォーティングの強さも違うの。シングルクォーテーション「'」のほうが強力で、すべての特殊文字が文字そのものとして扱われる。それに対して、ダブルクォーテーション「"」は変数などの一部の特殊文字が展開されるの。
次の例を見て理解してね。

```
$ val="guitar"　←変数「val」に「guitar」を代入
$ echo "This is a $val"　←ダブルクォーテーション「"」でクォーティング
This is a guitar　←変数が展開される
$ echo 'This is a $val'　←シングルクォーテーション「'」でクォーティング
This is a $val　←「$」が文字そのものとして扱われる
```



1 時間目：シェルスクリプトの基礎知識

シェルは、ユーザーインターフェースであると同時に、シェルスクリプトを作成するためのプログラム言語でもあります。プログラムというと尻込みする方もいるかもしれませんが、最初から複雑なものを作る必要はありません。単に、よく使う処理をファイルに保存しておいてコマンドとして呼び出せるようにするだけで、日常の操作がより便利に行えるようになります。まず、1 時間目では、テキストファイルにシンプルなシェルのコマンドを記述して、それをコマンドファイルにする手順について説明します。

コマンドをファイルに保存する。

例として、「~/Documents ディレクトリ以下で、ここ 1 週間のうちに更新されたファイルの一覧を、カレントディレクトリの下の「newFile.txt」というテキストファイルに保存する」という処理をシェルスクリプトにしてみましょう。「newFile.txt」の先頭には、その時点の日付時刻を記述するものとします。
処理の流れとしては、date コマンドと find コマンドを順に実行すればよいでしょう。まずは、次のようにコマンドを実行して確認してみましょう。

```
$ date > newFile.txt　　←日付時刻を保存
$ find ~/Documents -mtime -7 -type f >> newFile.txt　　←検索結果を保存
$ cat newFile.txt ←作成されたファイルを確認
Wed Feb 18 14:58:57 JST 2009
/home/o2/Documents/sample/ok.info
/home/o2/Documents/mail.txt
/home/o2/Documents/info.txt
```

四色君：
「>」と「>>」の違いってなんでしたっけ？

マリー先生：
どちらも出力のリダイレクションだけど、「>」はファイルが存在すると上書きしてしまうのに対して、「>>」は結果をファイルの最後に追加するの。

```
date > newFile.txt
find ~/Documents -mtime -7 -type f  >> newFile.txt
```

シェルスクリプトを実行する

シェルスクリプトを実行するには、シェルの引数にファイルのパスを指定します。HP-UX では POSIX シェルの場合コマンド名は「sh」になるため、次のような書式となります。

```
sh ファイルのパス
```

したがって、前述の「listFile」がカレントディレクトリにある場合、それを実行するには次のようにします。

```
$ sh listFile
```

タックス君：
POSIX シェルの sh コマンドってどこに保存されているのでしたっけ？

マリー先生：
次のように which コマンドを実行すれば、絶対パスがわかるわよ。

```
$ which sh
/usr/bin/sh
```

コマンドとして実行できるようにする

この状態では、作成したシェルスクリプトを実行するためには、sh コマンドの引数にファイルのパスを指定しなければなりません。続いて、「ファイルのパス 【Enter】」だけで実行できるようにしてみましょう。

まず、シェルスクリプト・ファイルの先頭に次の形式でコマンドのパスを指定します。

```
#!実行するプログラムのパス
```

POSIX シェルの場合、絶対パスは「/usr/bin/sh」になるため、次の行を「listFile」に加えます。

```
#!/usr/bin/sh
```

次に、chmod コマンドを実行して「listFile」に実行権を加えます。

```
$ chmod u+x listFile
```

タックス君：
chmod コマンドの最初の引数「u+x」ってなんですか？

マリー先生：
「x」は「eXecute」（実行）の意味で、「u+x」でユーザーに実行権を与えるの、「ls -l」コマンドで表示すると「x」が表示されているはずよ。

```
$ ls -l listFile
-rwxr-----   1 o2       users         84 Feb 18 15:04 listFile
```

これで、シェルスクリプトのパスを指定して実行できるようになります。たとえば、カレントディレクトリにある「listFile」を実行するには次のようにします。

```
$ listFile
```

四色君：
僕のところでは、これを実行すると「not found.」というエラーになるんですけど……。

マリー先生：
システムによっては、安全のために環境変数 PATH にカレントディレクトリ「.」が入っていない場合があるの。その場合、次のようにしてカレントディレクトリ「.」からの相対パスで「listFile」を指定してね。

```
$ ./listFile
```

スクリプト言語について

「シェルスクリプト」の「スクリプト」には映画や演劇の台本といった意味がありますが、プログラムの世界では比較的に小規模なプログラムといった意味でも使用されます。スクリプトを作成するのに適した言語のことを「スクリプト言語」と呼びます。多くのスクリプト言語は、ソースプログラムを実行時に機械語に翻訳していく「インタープリター方式」で実行されます。
以下に、代表的なスクリプト言語をいくつか紹介します。

言語	説明
Perl	テキスト処理に優れたスクリプト言語の代表的な存在
Ruby	まつもとゆきひろ氏により開発されたオブジェクト指向のスクリプト言語
PHP	HTML に埋め込んで使用する Web アプリケーション開発用スクリプト言語
JavaScript	Web ブラウザー上で実行される言語

2 時間目：シェルスクリプトの活用例

1 時間目の説明でシェルスクリプトの基本が理解できたと思います。2 時間目では、まず作成したシェルスクリプト・ファイルがカレントディレクトリになくても、コマンドとしてファイル名だけで実行できるようにする方法について説明します。続いて、ちょっと実践的なシェルスクリプトとして、拡張子をまとめて変換する例を示します。

環境変数 PATH の設定

1 時間目で作成したスクリプトがカレントディレクトリ以外にある場合、実行するのにそのパスを入力する必要があります。たとえば前述の「listFile」を「~/bin」ディレクトリに保存した場合には、「~/bin/listFile」と指定しなければなりません。それに対して、環境変数 PATH で設定されているディレクトリに保存されているコマンドはコマンド名だけで実行できます。デフォルトでは次のようなディレクトリがコロン「:」で区切られて格納されています。

```
$ echo $PATH 【Enter】
/usr/bin:/usr/ccs/bin:/usr/contrib/bin:/usr/contrib/Q4/bin:/opt/perl/bin:/opt/ipf/bin:/opt/nettladm/bin:/opt/fcms/bin:/opt/wbem/bin:/opt/wbem/sbin:/opt/sas/bin:/opt/graphics/common/bin:/opt/atok/bin:/usr/bin/X11:/usr/contrib/bin/X11:/opt/sec_mgmt/bastille/bin:/opt/drd/bin:/opt/dsau/bin:/opt/dsau/sbin:/opt/resmon/b
```

```
in:/opt/gnome/bin:/usr/contrib/kwdb/bin:/opt/firefox:/opt/mozilla:/opt/perl_32/bin:/opt/perl_64/bin:/opt/sfm/bin:/opt/swm/bin:/opt/sec_mgmt/spc/bin:/opt/ssh/bin:/opt/swa/bin:/opt/hpsmh/bin:/opt/gwlm/bin:/opt/mx/bin:/opt/perf/bin:/opt/prm/bin:/opt/wlm/bin:/opt/mpi/bin:/opt/vse/bin:/opt/caliper/bin:/opt/ignite/bin:/var/opt/netscape/server7/shared/bin:/var/opt/netscape/server7/bin:/opt/sentinel/bin:/opt/java1.4/jre/bin:/opt/spb/bin:/opt/thunderbird:/opt/langtools/bin:.
```

ここでは、自分用のコマンドの保存場所として「~/bin」ディレクトリを用意し、そこにシェルスクリプトを保存しましょう。まず、環境変数 PATH に「~/bin」を追加するには次のようにします。

```
$ export PATH=$PATH:~/bin
```

マリー先生：
これを「export PATH=~/bin」としないように注意してね。

四色君：
そうするとどうなるのですか？

マリー先生：
/bin や/usr/ccs/bin といった最初から設定されているパスが削除されて「~/bin」のみが設定されてしまうの。つまり、ls や cp といった普通のコマンドがファイル名だけで実行できなくなるわけね。

タックス君：
ログインする度に毎回 export コマンドを実行する必要があるのですか？

マリー先生：
前回で説明したシェルの環境設定ファイルで設定しておけば、その必要はないわ。

for 文を利用する

続いて、複数のファイルを順に処理するのに便利な for 文について説明しましょう。次のような書式で使用します

```
for 変数 in リスト
do
        コマンド 1
        コマンド 2
```

```
        ....
done
```

for 文では、「リスト」から値を順に取り出し「変数」に格納し、「do」と「done」との間のコマンドを実行します。このとき、コマンドライン引数のリストは、シェルスクリプト内では「$*」で参照できます。たとえば、次の例はすべての引数を順に表示するシェルスクリプト「forTest」です。

```
#!/usr/bin/sh
for val in $*
do
  echo $val
done
```

このシェルスクリプト「forTest」を使用して、カレントディレクトリの下にある拡張子が「.txt」のファイルの一覧を表示するには、引数に「*.txt」を指定して次のようにします。

```
$ forTest *.txt
2008-3-1.txt
linus.txt
new.txt
newFile.txt
sample.txt
```

タックス君：
コマンドライン引数を個別に参照することはできますか？

マリー先生：
もちろん。引数は順に「$1」、「$2」、「$3」…といった変数に格納されるから、それらを参照すればいいの。

変数のパターンマッチ

シェルスクリプトの便利なテクニックをもうひとつ紹介しましょう。変数から「パターン」に一致する部分を取り除くというものです。これは「パターン照合演算子」などと呼ばれ、次のような形式で使用します。

```
${変数名%パターン}
```

変数の内容の最後の部分とパターンがマッチしたら、もっとも短く一致する部分を取り除いた残りの部分を返します。たとえば、パターンに「.*」を指定することで、ファイルのパスから拡張子を取り除くといった処理が行えます。これを利用して、変数「file」に保存されているファイルの、拡張子を「txt」に変換するには次のようにします。

```
$ mv $file ${file%.*}.txt
```

タックス君：
パターン照合演算子はこれ例外にもあるのですか？

マリー先生：
次の 4 種類があるわ。各自コマンドラインで実行して結果を確認してね。

書式	説明
${変数名%パターン}	最後の部分とパターンがマッチしたら、もっとも短く一致する部分を取り除いた残りの部分を返す
${変数名%%パターン}	最後の部分とパターンがマッチしたら、もっとも長く一致する部分を取り除いた残りの部分を返す
${変数名#パターン}	先頭部分とパターンがマッチしたら、もっとも短く一致する部分を取り除いた残りの部分を返す
${変数名##パターン}	先頭部分とパターンがマッチしたら、もっとも長く一致する部分を取り除いた残りの部分を返す

引数で指定したファイルの拡張子をすべて「.txt」にする

ここまで学んだことをふまえ、引数で指定したファイルの拡張子をすべて「.txt」にそろえるシェルスクリプト例「cvext」を見てみましょう。次にリストを示します。

```
#!/usr/bin/sh
for val in $*
do
    new=${val%.*}.txt   ←(1)
    echo "Now moving... $val to $new"   ←(2)
    mv $val $new        ←(3)
done
```

(1)で変数「val」に格納されたファイル名から拡張子を取り除き、「.txt」を加えて変数「new」に格納しています。(2)の echo コマンドでは実行中であることを画面に表示し、(3)でファイルの名前を変更しています。
次に、この cvext を使用してカレントディレクトリの下にある拡張子が「.html」と「.mail」のファイルを、拡張子「.txt」に変換する例を示します。

```
$ ls        ←ls コマンドで確認

index.html  info.html   new.mail   sample.txt  you.html   ←元のファイル
$ cvext *.html *.mail      ←シェルスクリプトを実行して拡張子を変換
Now moving... index.html to index.txt
```

```
Now moving... info.html to info.txt
Now moving... you.html to you.txt
Now moving... new.mail to new.txt
$ ls       ←ls コマンドで確認
index.txt   info.txt    new.txt     sample.txt  you.txt          ←変換後のファイル
```

四色君：
ファイル名を変更するときに、その名前のファイルがすでにある場合は警告なしに上書きされてしまいますよね。

マリー先生：
そうね。たとえば、「sample.text」を「sampl.txt」に変更しようとしたときに、「sample.txt」が存在していた場合にはそのまま上書きされてしまうわ。 上書きするかどうかをその都度確認したい場合には、mv コマンドに「-i」オプションを指定するといいわよ。

```
mv -i  $val $new   ←「-i」オプションを指定
```

こうすれば、同じ名前のファイルがあると確認のメッセージが表示されるの。

```
$ cvext *.html
Now moving... new.html to new.txt
remove new.txt? (y/n) y            ←「y」で上書き
```

練習問題

第 1 問：POSIX シェルを使って、シェルスクリプト・ファイル「mycmd」を実行するコマンドはどれか？

a)　sh mycmd
b)　posix mycmd
c)　execute mycmd
d)　exec mycmd

a)　sh mycmd
HP-UX では POSIX シェルのコマンド名は「sh」である。

第 2 問：シェルスクリプト「mycmd」をコマンドとして実行するためはファイルの先頭にどんな行を加えればよいか？

a)　/usr/bin/sh
b)　!usr/bin/sh
c)　#!/usr/bin/sh
d)　#/usr/bin/sh

c)　#!/usr/bin/sh
ファイルを指定したコマンドで実行するには、先頭行に「#!実行するプログラムのパス」を指定する。

第 3 問：変数「myfile」にあるファイルのパスが格納されているとき、拡張子を取り除いて表示するコマンドはどれか？

a)　${myfile%.*}
b)　echo ${myfile%.*}
c)　echo ${myfile#.*}
d)　echo ${myfile#.*}

b)　echo ${myfile%.*}
拡張子を取り除くパターン照合演算子は「${変数名%パターン}」である。

第 4 問：シェルスクリプトの中で、すべてのコマンドライン引数を参照する変数はどれか？

a)　$*
b)　$1
c)　$$
d)　#*

a)　$*
コマンドライン引数はまとめて「$*」に格納される。個々のコマンドライン引数は順に「$1」、「$2」、「$3」…に格納される。

UNIX の教科書「応用編」

期末試験

2009 年 5 月　日本ヒューレット・パッカード株式会社

シェルのより高度な利用方法を解説してきた「UNIX の教科書 応用編」(全 7 回)、いかがでしたか？内容はご理解いただけたでしょうか。今回は、「期末試験」として全 10 問の試験を用意しました。いずれもこれまでの連載からの出題ですので、ぜひ挑戦してみてください。

マリー先生：
これまでの授業を理解できたかのおさらいを兼ね、今日は期末試験を行います。10 問全て今まで学習してきたことなので、満点を目指してくださいね。

第 1 問：/etc/inetd.conf ファイルから、文字列「rpc」を含み、かつ、先頭の文字が「#」ではない行を取り出すコマンドはどれか？

a)　grep -e rpc -v ^# /etc/inetd.conf
b)　grep rpc /etc/inetd.conf | grep #
c)　grep rpc ^# /etc/inetd.conf
d)　grep rpc /etc/inetd.conf | grep -v ^#

d)　grep rpc /etc/inetd.conf|grep -v ^#
複数の条件で絞り込むにはパイプ「|」を使用して grep コマンドを組み合わせる。また、指定した文字列を含まない行を取り出すには「-v 文字列」オプションを指定する。

第 2 問：docs ディレクトリ以下の拡張子が「.txt」のファイルを、確認しながら backup ディレクトリの下に移動するコマンドはどれか？

a)　find docs -name "*.txt" -type f -exec mv {} backup \;
b)　find docs -name "*.txt" -type f ?exec mv -i {} backup \;
c)　find docs -name "*.txt" -type f -i mv {} backup \;
d)　find docs -name "*.txt" -type f -ok mv {} backup \;

d)　find docs -name "*.txt" -type f -ok mv {} backup \;
検索結果に対して、その都度確認しながら指定したコマンドを実行するには「-ok コマンド {} \:」オプションを指定する。

第 3 問：vi エディタでテキストの 2 行目から現在行までの範囲で、文字列「hp」をすべて「HP」に置換するコマンドはどれか？

a)　:2,$s/hp/HP/g
b)　:2,.s/hp/HP/g
c)　:2,.s/hp/HP/
d)　:1,.s/hp/HP/

b)　:2,.s/hp/HP/g
現在行はピリオド「.」で表せる。また、すべての文字列を置換するには最後に「g」が必要となる。

第 4 問：カレントディレクトリの拡張子が「.txt」のすべてのファイルを gzip 形式で圧縮するコマンドはどれか？

a)　gzip *.txt
b)　gzip txt
c)　gzip -d txt
d)　gzipd

a)　gzip *.txt
圧縮コマンドの引数にはワイルドカードを使用してファイルを指定できる。

第 5 問：Documents ディレクトリのアーカイブを「Documents.tar」というファイル名で作成するコマンドはどれか？

a)　tar -xvf Documents.tar Documents
b)　tar -tvf Documents.tar Documents
c)　tar -cvf Documents.tar Documents
d)　tar ?cvf Documents Documents.tar

c)　tar -cvf Documents.tar Documents
アーカイブの作成は「-cvf アーカイブのパス」オプションを指定して tar コマンドを実行する。

第 6 問：現在実行中のジョブを一覧表示するコマンドはどれか？

a)　ps
b)　ls -j
c)　jobs
d)　showj

c)　jobs
jobs は実行中のジョブを一覧表示する組み込みコマンド。なお、a)は実行中のプロセスを一覧表示するコマンドである。

第 7 問：ジョブ番号「2」のジョブが一時停止している時に、そのジョブをバックグラウンドジョブとして再開させるコマンドはどれか？

a)　fg #2fg #2
b)　bg #2
c)　fg 2
d)　bg %2

d)　bg %2
bg コマンドは一時停止中のジョブをバックグラウンドジョブとして再開するコマンド。ジョブ番号は「%ジョブ番号」で指定する。

第 8 問：POSIX シェルにて、「cp -r」コマンドのエイリアスとして「cpDir」を定義するコマンドはどれか？

a)　alias cpDir=cp -r
b)　alias cpDir='cp -r'
c)　export cpDir="cp -r"
d)　alias cpDir "cp -r"

b)　alias cpDir='cp -r'
スペースを含むコマンドのエイリアスを定義するにはクォーティングする必要がある。

第 9 問：POSIX シェルにて、変数「MYENV」に値として「SAMPLE」を代入し環境変数にするコマンドはどれか？

a)　MYENV="SAMPLE"
b)　export MYENV="SAMPLE"
c)　export $MYENV="SAMPLE"
d)　env MYENV="SAMPLE"

b)　export MYENV="SAMPLE"
変数を環境変数にするには export コマンドでエクスポートする。

第 10 問：POSIX シェルのシェルスクリプト・ファイルをコマンドとして実行するためはファイルの先頭にどんな行を加えればよいか？

a)　/usr/bin/sh
b)　exec /usr/bin/sh
c)　#!/usr/bin/sh
d)　#/usr/bin/sh

c)　#!/usr/bin/sh
ファイルを指定したコマンドで実行するには、先頭行に「#!実行するプログラムのパス」を指定する。

UNIX の教科書「運用編」

―目次―

コマンドです。いずれもシステムの状況を把握しなければならな いシステム管理者にとっては必須のコマンド群ですので、しっかり覚えて使いなせるようになってください。(2010 年 2 月)

期末試験

ユーザーからネットワークやシステムなど、管理者への第一歩を解説してきました「UNIX の教科書 運用編」(全 6 回)、いかがでしたでしょうか？内容はご理解いただけたでしょうか。今回は、「期末試験」として全 10 問の試験を用意しました。いずれもこれまでの連載からの出題ですので、ぜひ挑戦してみてください。(2010 年 3 月)

UNIX の教科書「運用編」

第 1 日目：ユーザーの管理

2009 年 9 月　大津 真

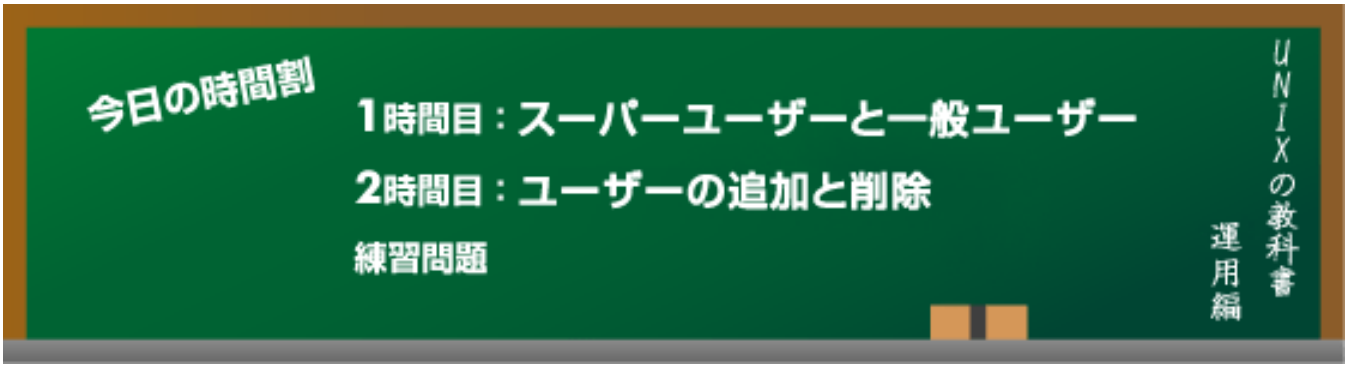
基礎編、応用編とステップアップしてきた「UNIX の教科書」も、運用編に入ります。これまでは一般ユーザーとしての使用方法を学んできましたが、管理者への勉強の始まりです。学んできた内容をしっかり踏まえてついてきてください。今月は、システムに対するすべての権限を持つ「スーパーユーザー」について説明します。2 時間目にはスーパーユーザーとしての仕事のひとつ、システムに新規ユーザーを追加する作業をしてみましょう。

マリー先生：
さあ、UNIX の教科書も今回から運用編ということで、より実践的な HP-UX の使い方について説明していきましょう。基礎編と応用編の内容を理解していることが不可欠なので、もういちど見直しておいてね。それはそうと、応用編の期末試験の結果はどうだったかな？

タックス君：
僕は全問正解！

四色君：
僕は第 3 問がだめでした。vi エディタの置換コマンドの書式が思い出せなくて……。

マリー先生：
まあ、エディタは慣れだからね。毎日使っていれば、自然とコマンドが身に付いていくわよ。



1 時間目：スーパーユーザーと一般ユーザー

HP-UX など一般的な UNIX 系 OS では、ユーザーは伝統的に「一般ユーザー」と「スーパーユーザー」の 2 種類に大別することができます。一般ユーザーは日常的なログインに使用するアカウントで、ディレクトリやファイルを作成できるのは通常そのユーザーのホームディレクトリ以下に限られます。また、ユーザーの作成／削除やネットワークの設定といった管理コマンドは実行できません。一方、スーパーユーザーはそのシステムに関するすべての権限が与えられたユーザーで、システムに対してあらゆる操作が可能になっています。たとえば、システムの設定ファイルを書き換えたり、他のユーザーが作成したファイルを削除したりといったことが行えます。

四色君：
スーパーユーザーのユーザー名は決まっているのですか？

マリー先生：
UNIX 系 OS では、伝統的にスーパーユーザーの名前は「root」（ルート）に決まっているの。

タックス君：
そういえばファイルシステムの頂点も「root」（/）ですよね。

マリー先生：
そうね。UNIX では「root」が、重要なキーワードのひとつっていうことね。あと、スーパーユーザーのパスワードが悪意のある第三者に漏れると、システムに壊滅的なダメージを加えられる可能性があるので、root のパスワード管理は一般ユーザーのパスワード管理よりもさらに慎重に行う必要があるの。

一時的にスーパーユーザーになる

一般ユーザーでログインした状態で、一時的にスーパーユーザーになるには su コマンドを使います。su コマンドを実行し、「Password:」に続いてスーパーユーザーのパスワードを入力すると、プロンプトが「$」から「#」に代わり、スーパーユーザーに移行したことを表します。なお、現在のユーザー名は whoami コマンドで確認できます。

```
$ whoami 【Enter】
marry ←現在のユーザー名は「marry」
$ su 【Enter】
Password: ←スーパーユーザーのパスワードを入力
# whoami 【Enter】 ←スーパーユーザーに移行
root ←現在のユーザー名は「root」
```

元のユーザーに戻るには exit コマンドを実行します。

```
# exit 【Enter】
$ ←一般ユーザーに戻った
```

四色君：
su コマンドを使わないで、直接 root としてログインしてもいいのですか？

マリー先生：
それも可能だけど、ログアウトしたまま席を離れたりするとキケンでしょう。だから、通常は管理作業が必要なときだけ su コマンドでスーパーユーザーに移行したほうがいいわね。

環境も含めてスーパーユーザーに移行する

su コマンドを引数なしで実行した場合、カレントディレクトリなどの環境は変更されません。

```
$ su 【Enter】
Password: ←スーパーユーザーのパスワードを入力
# pwd 【Enter】
/home/marry ←カレントディレクトリは「marry」のまま
```

なお、コマンドの保存先を管理する環境変数「PATH」は次のようにセットされます。

```
# echo $PATH 【Enter】
/usr/bin:/usr/sbin:/sbin
```

それに対して、su コマンドを「-」オプションを指定して実行すると、環境もスーパーユーザーのものに移行します。たとえばカレントディレクトリはスーパーユーザーのホームディレクトリ（初期状態で「/」）になります。

```
$ su -　【Enter】
Password:　←スーパーユーザーのパスワードを入力
# pwd　【Enter】
/　←カレントディレクトリは「/」
```

また、環境変数「PATH」などがスーパーユーザーの環境設定ファイルに応じて再設定されます。

```
$ echo $PATH　【Enter】
/usr/sbin:/usr/bin:/usr/ccs/bin:/usr/contrib/bin:/usr/contrib/Q4/bin:/opt/perl/bin:/opt/ipf/bin:/opt/nettladm/
bin:/opt/fcms/bin:/opt/wbem/bin:/opt/wbem/sbin:/opt/sas/bin:/opt/graphics/common/bin:/opt/atok/bin:/u
sr/bin/X11:/usr/contrib/bin/X11:/opt/sec_mgmt/bastille/bin:/opt/drd/bin:/opt/dsau/bin:/opt/dsau/sbin:/opt/
resmon/bin:/opt/gnome/bin:/usr/contrib/kwdb/bin:/opt/firefox:/opt/mozilla:/opt/perl_32/bin:/opt/perl_64/b
in:/opt/sfm/bin:/opt/swm/bin:/opt/sec_mgmt/spc/bin:/opt/ssh/bin:/opt/swa/bin:/opt/hpsmh/bin:/opt/gwlm
/bin:/opt/mx/bin:/opt/perf/bin:/opt/prm/bin:/opt/wlm/bin:/opt/mpi/bin:/opt/vse/bin:/opt/caliper/bin:/opt/i
gnite/bin:/var/opt/netscape/server7/shared/bin:/var/opt/netscape/server7/bin:/opt/sentinel/bin:/opt/java1.4
/jre/bin:/opt/spb/bin:/opt/thunderbird:/opt/langtools/bin:/sbin:/home/root
```

タックス君：
su コマンドでスーパーユーザー以外のユーザーになることもできるのですか？

マリー先生：
できるわ。その場合引数にユーザー名を指定するの。

```
$ su yamada　【Enter】　←ユーザー「yamada」に移行
Password:　←パスワードを入力
```

このとき、「-」オプションを指定すると環境もそのユーザーに移行する点は、スーパーユーザーの場合と同じね。

shutdown コマンドによるシステムの終了

スーパーユーザーでは、一般ユーザーには許可されていないさまざまな管理コマンドが実行できます。たとえば、システムを終了する shutdown コマンドなどがそうです。システムをすぐに終了（シャットダウン）するには、次のように実行します。

```
# shutdown -h now　【Enter】
SHUTDOWN PROGRAM
08/06/09 17:22:28 JST

Broadcast Message from o2 (pts/0) Thu Aug  6 17:22:28...
SYSTEM BEING BROUGHT DOWN NOW ! ! !
```

～以下略～

四色君：
システムを停止するのではなく再起動することもできますか？

マリー先生：
「-h」の代わりに「-r」を指定すればリブートになるわよ。あと、「now」の代わりに秒数を指定すると、その秒数経過後にシャットダウン／リブートするの。たとえば 10 秒後にリブートするには次のように実行すればいいわ。

```
# shutdown -r 10
```

ユーザーをまとめるグループ

UNIX では、複数のユーザーをまとめた「グループ」という管理単位があります。たとえば、ファイルの読み書きの許可はグループ単位で設定することが可能です。現在自分の属するグループは groups コマンドで確認できます。システムにもよりますが、一般ユーザーは少なくとも「users」といった名前の共通のグループに属しています。

```
$ groups 【Enter】 ←自分の属するグループを確認
users
```

それに対して、スーパーユーザーは「adm」や「sys」といった複数の管理グループに属しています。

```
# groups 【Enter】
adm bin daemon lp mail other root sys users
```

なお、システムの内部では、ユーザーとグループは、それぞれ「ユーザーID」、「グループ ID」と呼ばれる ID 番号で管理されています。自分の ID 番号は id コマンドで確認できます。

```
$ id 【Enter】
uid=111(o2) gid=20(users)
    ユーザーID    グループID
```

タックス君：
利用可能な ID 番号というのは決まっているのですか？

マリー先生：
まず、スーパーユーザーのユーザ ID は「0」ね。一般ユーザーの場合は通常「111」番以降の番号が使用されるわ。

四色君：
複数のグループに属している場合、グループの優先順位みたいなものはあるのですか？

マリー先生：
メインとなるグループのことを「プライマリグループ」と呼ぶの。もちろん属しているグループがひとつだけの場合はそれがプライマリグループになるわ。
複数のグループに属している場合に、どれがプライマリグループかは id コマンドで確認できるの。
次は、スーパーユーザーの実行例だけど、ちょっと見てみて。

```
# id 【Enter】
uid=0(root) gid=3(sys) groups=0(root),1(other),2(bin),
4(adm),5(daemon),6(mail),7(lp),20(users)
```

プライマリグループ

このように、まずプライマリグループは「gid=」の後に表示され、「groups=」の後に「グループ ID（グループ名）」がカンマで区切られて表示されるの。つまり、HP-UX ではスーパーユーザーのプライマリグループはグループ ID が「3」の「sys」という名前のグループなわけね。

休憩時間：HP SMH（System Management Homepage）

HP-UX 11i v3 には、Web ベースのシステム管理ツールとして HP SMH（System Management Homepage）が標準搭載されています。SMH を使用すると、設定ファイルを編集したり CUI コマンドを実行したりする必要なく、ユーザー管理やネットワーク設定などの管理作業が行えます Web ブラウザで「http://ホスト名もしくは IP アドレス:2301」にアクセスし証明書を受け入れると、ログイン画面が表示されます。ユーザー名（初期状態で「root」）およびパスワードを入力し「Sign In」ボタンをクリックすると、SMH のメニュー画面が表示されます。

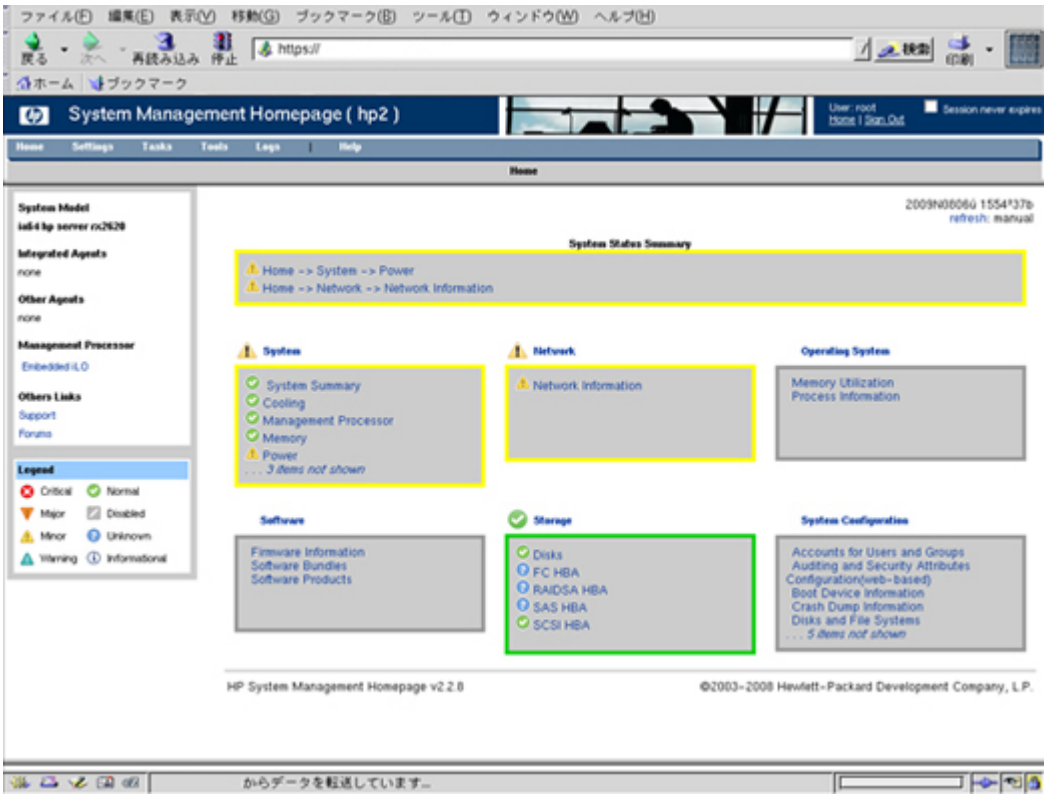


図 1：SMH のメニュー画面

なお、スーパーユーザーのターミナル上で smh コマンドを実行すると、テキストインターフェース・ベースで SMH を実行できます。
参考：SMH でらくらく HP-UX システム管理

2 時間目：ユーザーの追加と削除

続いて、システムに新規ユーザーを登録する方法を説明します。この作業には 1 時間目で説明したスーパーユーザーの権限が必要です。コマンドラインで新規ユーザーを追加するには useradd コマンドを使います。useradd コマンドには多くのオプションが指定可能ですが、デフォルト値を使用する場合にはオプションは省略可能です。「-D」オプションのみを指定して実行すると現在のデフォルト値が表示されます。

```
# useradd -D 【Enter】
GROUPID  20 ←(1)
BASEDIR  /home ←(2)
SKEL    /etc/skel ←(3)
```

```
SHELL   /sbin/sh ←(4)
INACTIVE -1
EXPIRE
COMMENT
CHOWN_HOMEDIR no
CREAT_HOMEDIR no
ALLOW_DUP_UIDS no
```

(1)は、一般ユーザーのプライマリグループの設定です。この例では「20」（グループ名は「users」）に設定されています。(2)はホームディレクトリの起点のパスです。この例のように「/home」の場合には、「/home/ユーザー名」がホームディレクトリとなります。(3)はホームディレクトリにコピーする環境設定ファイルの保存先です。(4)はデフォルトのシェルの設定です。なお、ユーザーID は、「現在のユーザーID の最大値+1」がデフォルト値となります。
デフォルトの設定値を使用してユーザー「taro」を追加するには、次のようにします。

```
# useradd taro
```

タックス君：
次のように useradd コマンドを実行するとエラーになるんですけど……。

```
# useradd yamadataro 【Enter】
Login 'yamadataro' is invalid
```

マリー先生：
ユーザー名は初期状態では最大８文字までよ。あと、慣習的に小文字の半角アルファベットで指定するの。

ホームディレクトリを作成する

次に、mkdir コマンドでユーザーのホームディレクトリを作成します。デフォルトではホームディレクトリの起点は「/home」であるため、たとえばユーザー「taro」の場合は次のようにします。

```
# mkdir /home/taro
```

ただし、この状態では作成したホームディレクトリの所有者は「root」、所有グループは「sys」です。

```
# ls -dl /home/taro/ 【Enter】
drwxr-x---   2 root      sys            96 Aug  6 14:23 /home/taro/
                所有者     所有グループ
```

そのため、chown コマンドを使用して、ホームディレクトリの所有者を作成したユーザーに、所有グループをそのユーザーのプライマリグループ（デフォルトは「users」）に設定します。chown コマンドの最初の引数には「所有者:所有グループ」を、2番目の引数には対象となるファイルやディレクトリを指定します。

```
# chown taro:users /home/taro/ 【Enter】
# ls -dl /home/taro/ 【Enter】
drwxr-x---    2 taro      users          96 Aug  6 14:23 /home/taro/
                 所有者     所有グループ
```

パスワードを設定する

ユーザーを追加した状態では、まだパスワードが設定されていないためログインできません。ユーザー名を引数に passwd コマンドを実行して、パスワードを設定します。

```
# passwd taro 【Enter】
Changing password for taro
New password: ←パスワードを入力
Re-enter new password: ←もう一度パスワードを入力
Passwd successfully changed
```

マリー先生：
この passwd コマンドはパスワードの変更にも使用できるわよ。

タックス君：
自分のパスワードを変更するときもユーザー名を指定するのですか？

マリー先生：
その場合は、passwd コマンドを引数なしで実行すれ OK よ。あと、他人のパスワードを変更できるのはスーパーユーザーだけなの。

ログインできるかを確認する

最後に、ログインし直して、作成したユーザーでログインできるかを確認します。あるいは、「su - ユーザー名」を実行してユーザーを移行することでも確かめられます。

```
$ su - taro 【Enter】 ←ユーザー「taro」に移行する
Password: ←パスワードを入力
(c)Copyright 1983-2006 Hewlett-Packard Development Company, L.P.
(c)Copyright 1979, 1980, 1983, 1985-1993 The Regents of the Univ. of California
```

(c)Copyright 1980, 1984, 1986 Novell, Inc.
(c)Copyright 1986-2000 Sun Microsystems, Inc.
(c)Copyright 1985, 1986, 1988 Massachusetts Institute of Technology
(c)Copyright 1989-1993 The Open Software Foundation, Inc.
(c)Copyright 1990 Motorola, Inc.
(c)Copyright 1990, 1991, 1992 Cornell University
(c)Copyright 1989-1991 The University of Maryland
(c)Copyright 1988 Carnegie Mellon University
(c)Copyright 1991-2006 Mentat Inc.
(c)Copyright 1996 Morning Star Technologies, Inc.
(c)Copyright 1996 Progressive Systems, Inc.

Confidential computer software. Valid license from HP required for
possession, use or copying. Consistent with FAR 12.211 and 12.212,
Commercial Computer Software, Computer Software Documentation, and
Technical Data for Commercial Items are licensed to the U.S. Government
under vendor's standard commercial license.

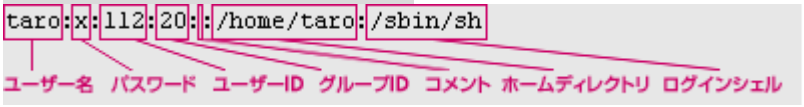
$ pwd 【Enter】 ←カレントディレクトリを確認
/home/taro ←ユーザーのホームディレクトリが表示される
$ whoami 【Enter】 ←ユーザー名を確認
taro ←ユーザー名が表示される

ユーザー情報が格納されているファイル

ユーザー情報は「/etc/passwd」ファイルに、1 行に 1 ユーザーずつ格納されています。

/etc/passwd の例
root:x:0:3::/:/sbin/sh
daemon:x:1:5::/:/sbin/sh
bin:x:2:2::/usr/bin:/sbin/sh
sys:x:3:3::/:
adm:x:4:4::/var/adm:/sbin/sh
uucp:x:5:3::/var/spool/uucppublic:/usr/lbin/uucp/uucico
lp:x:9:7::/var/spool/lp:/sbin/sh
nuucp:x:11:11::/var/spool/uucppublic:/usr/lbin/uucp/uucico
hpdb:x:27:1:ALLBASE:/:/sbin/sh
nobody:x:-2:-2::/:
www:x:30:1::/home/www:
smbnull:x:101:101:DO NOT USE OR DELETE - needed by
Samba:/var/opt/samba/nologin:/bin/false
sfmdb:x:102:20::/home/sfmdb:/sbin/sh
sshd:x:103:103:sshd privsep:/var/empty:/bin/false
iwww:x:104:1::/home/iwww:/sbin/sh

owww:x:105:1::/home/owww:/sbin/sh
hpsmh:x:106:104:System Management Homepage:/var/opt/hpsmh:/sbin/sh
ids:x:107:105:HP-UX Host IDS Administrator:/opt/ids/home:/sbin/sh
cimsrvr:x:108:106:WBEM Services:/var/opt/wbem:/sbin/sh
hpsmdb:x:109:20::/home/hpsmdb:/sbin/sh
tftp:x:110:107:Trivial FTP user:/home/tftp:/usr/bin/false
o2:x:111:20::/home/o2:/sbin/sh



四色君：
/etc/passwd には一般ユーザーとスーパーユーザー以外に、いろいろなユーザーが登録されていますね。

マリー先生：
そうね。ただし、ユーザーID が 110 以下のアカウントは、一般ユーザーのようにログインするためのものではないの。いろいろなサービスを実行するためのユーザーとして使用されるものがほとんどね。

タックス君：
ユーザーの管理は SMH でもできますよね？

マリー先生：
たしかに SMH の「ugweb」を使うと簡単だけど、仕組みを理解するためにもコマンドラインでのユーザーの追加方法と、ユーザー情報の保存されているファイルを覚えておいたほうがいいわね。

「passwd」という名前が示すように、このファイルにはかつて暗号化されたパスワードが 2 番目のフィールドに格納されていました。しかし、この/etc/passwd は一般ユーザーに読み書きできるファイルです。暗号化されているとはいえ、そのまま格納されているのはセキュリティ的に好ましくありません。そのため、最近では「シャドウパスワード」というパスワード管理方式が主流です。シャドウパスワードを採用しているシステムでは「/etc/passwd」の 2 番目のフィールドが「x」となっています。パスワードなどの情報は「/etc/shadow」といいうスーパーユーザーのみが読み書き可能なファイルに保存しています。

/etc/shadow の例
root:cmlKlnASqfNUE:14012::::::
daemon:*:14012::::::
bin:*:14012::::::
～中略～

```
tftp:*:14012::::::
o2:CgqiIQrG.ubuI:14020::::::
taro:4/2MisjtuhEww:14462:::::
```

暗号化されたパスワード

ユーザーの削除

既存のユーザーを削除するには、ユーザー名を引数に userdel コマンドを実行します。

```
# userdel makoto ←ユーザー「makoto」を削除
```

タックス君：
ユーザーを削除するとホームディレクトリはどうなるのですか？

マリー先生：
ホームディレクトリはそのまま残るわ。必要に応じてバックアップを取った後に、rm コマンドで削除してね。

```
# rm -r /home/makoto
```

四色君：
userdel コマンドでホームディレクトリを削除することはできないんですか？

マリー先生：
それもできるわよ。「-r」オプションを指定して userdel コマンドを実行すると、ユーザーとそのホームディレクトリが削除されるわ。

```
# userdel -r makoto
```

練習問題

第 1 問：環境も含めてスーパーユーザーに移行するコマンドはどれか？

a) su
b) super
c) sudo
d) su -

d)　su -
su コマンドを「-」オプション付きで実行すると環境を含めてスーパーユーザーに移行する。

第 2 問：20 秒後にシステムを終了するコマンドはどれか？

a)　shutdown -h 20
b)　shutdown -r 20
c)　shut 20
d)　stop 20

a)　shutdown -h 20
システムを終了するには「shutdown -h 秒数」コマンドを実行する。

第 3 問：ユーザー「hanako」を作成するコマンドはどれか？

a)　passwd hanako
b)　useradd hanako
c)　new hanako
d)　create user hanako

b)　useradd hanako
新規ユーザーを登録するには useradd コマンドを実行する。

第 4 問：パスワード情報の格納されているファイルはどれか？

a)　/etc/passwd
b)　/etc/shadow
c)　/etc/secret
d)　/etc/password

b)　/etc/shadow
シャドウパスワードを採用したシステムでは、暗号化されたパスワードは/etc/shadow に格納されている。

UNIX の教科書「運用編」

第 2 日目：ファイルの安全管理（その 1）

2009 年 10 月　大津 真

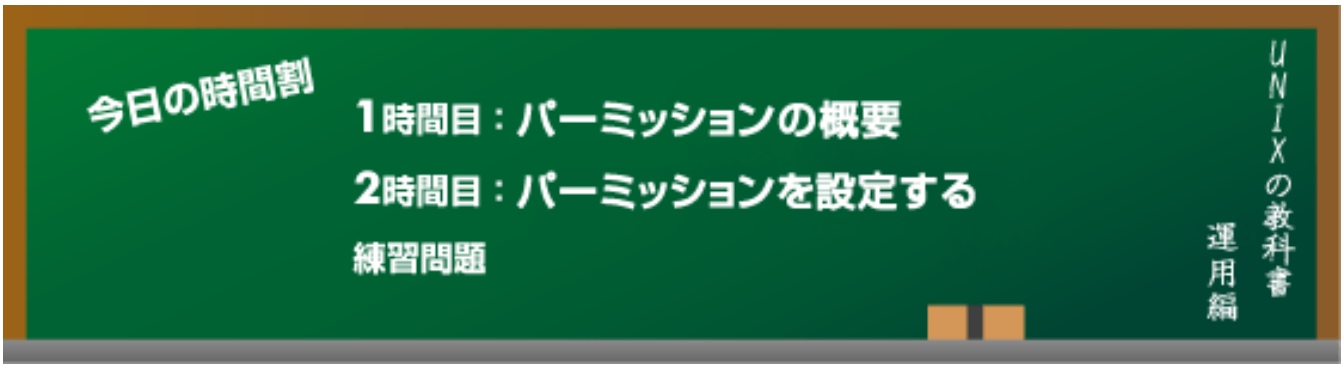
コンピューターを運用していく上で、今後いっそう重要になっていくのがセキュリティ対策です。HP-UX にはセキュリティ対策のためにさまざまな機能が搭載されていますが、まずははじめの一歩、基礎の基礎であるファイルのアクセス権限を理解しましょう。伝統的な UNIX システムでは、個々のファイルやディレクトリに対して「パーミッション」というアクセス権限が設定されています。1 時間目ではパーミッションの概要を、2 時間目では実際にパーミッションを設定する方法を学びます。

マリー先生：
みんなも知っていると思うけど、コンピューターを安全に運用していくためには、さまざまなセキュリティ対策が必須です。今日は、セキュリティ管理の基本中の基本ともいえる、ファイルのアクセス権限の設定方法について説明しましょう。その前に、前回のユーザー管理の説明でなにか質問ありますか？

タックス君：
useradd コマンドで新規ユーザーを登録するときに、ホームディレクトリを自動で作成することはできますか？

マリー先生：
「-m」オプションを指定すれば OK よ。この場合、デフォルトの設定ファイルの保存場所であるスケルトンディレクトリ（初期状態で「/etc/skel」）から「.login」や「.profile」などの設定ファイルもコピーされるの。

```
# useradd -m user3　【Enter】
 ←「-m」オプションを指定して「user3」を作成
# ls -a /home/user3/　【Enter】
 ←ホームディレクトリ「/home/user3」が作成される
.　　..　　.cshrc　.exrc　.login　.profile
←設定ファイルがコピーされる
```



1 時間目：パーミッションの概要

伝統的な UNIX システムでは、個々のファイルやディレクトリに対して「パーミッション」と呼ばれるアクセス権限が設定されています。まずは、パーミッションの概要について説明しましょう。

パーミッションを確認する

個々のファイルには「所有者」および「所有グループ」が設定されています。パーミッションは、ファイルの「所有者」「所有グループ」「その他のユーザー」それぞれについて、「読み出し可」「書き換え可」「実行可」を設定できます。
現在のパーミッションは、ls コマンドを「-l」オプションを指定して実行すると確認できます。

```
$ ls -l 【Enter】
total 39712
-rw-r-----   1 o2         users     10157964 Sep  6 15:34 index.htm
-rw-r-----   1 o2         users       12526 Sep  6 15:34 info.txt
-rw-rw-r--   1 o2         users     10157964 Feb 18  2009 sample.txt
drwxr-x---   2 o2         users           96 Sep  6 15:34 samples
```


左端の 1 桁はファイルの種類を表しており、「-」が通常のファイル、「d」がディレクトリです。その右の 9 桁がパーミッションです。3 桁ずつ、所有者、所有グループ、その他のユーザーに対する設定で、「r」は読み出し、「w」は書き換え、「x」は実行が許可されていることを表し、「-」は当該の権限がないことを示しています。

図 1：パーミッションの表示例

たとえば「rw-rw-r--」は、所有者と所有グループには読み書き可、その他のユーザーは読み出しのみ可に設定されていることを表します。

四色君：
読み込み可の「r」は「Readable」、書き込み可の「w」は「Writable」の頭文字ですよね。実行可の「x」はなんの略ですか？

マリー先生：
「x」は「eXecutable」（実行可能）の 2 文字目なの。

ファイルのパーミッション

パーミッションは、対象が通常のファイルなのかそれともディレクトリなのかによって多少意味が異なります。まず、ファイルの場合について説明しましょう。「r」（読み出し可）は、そのファイルの内容を表示できるかどうかです。次に「w」（書き換え可）は、そのファイルを編集したり上書きしたりできるかどうかです。

タックス君：
書き換えが許可されていないファイルは削除もできないんですか？

マリー先生：
いい質問ね。書き換えが許可されていなくても、削除はできるの。ただし rm コマンドでは削除するときに確認のメッセージが表示されるわ。次の例を見てね。

```
$ ls -l tmp.txt 【Enter】
-r--r--r--  1 o2      users      12527 Sep  6 16:06 tmp.txt
 ←読み出しのみ許可されている
$ rm tmp.txt 【Enter】 ←削除しようとすると……
tmp.txt: 444  mode ? (y/n) y    ←「y」で削除
```

この確認メッセージは、誤って削除しないように rm コマンドが表示しているの。後で説明するけど、システム的には削除できるかどうかは、そのファイルが保存されているディレクトリのパーミッションに依存するの。

四色君：
たとえば、所有者に読み出し許可がなくて、所有グループに読み出し許可があるとしましょう。所有者がそのグループに属している場合はファイルを表示することはできるのですか？

マリー先生：
できないわ。所有者のパーミッションが優先されるわけ。ただし、その所有グループに属する他のユーザーは表示できるわよ。

```
$ ls -l sample.txt 【Enter】
----r--r--  1 o2       users        12 Sep  6 16:30 sample.txt
 ←所有者には読み出し許可がない
$ cat sample.txt 【Enter】
 ←cat コマンドで表示しようとすると……
cat: Cannot open sample.txt: Permission denied ←表示できない
$ su taro 【Enter】 ←users グループに属する他のユーザーに移行
Password:
$ cat sample.txt 【Enter】
hallo  ←表示できる
hp-ux
…
```

なお、「x」（実行可）はそのファイルがコマンドとして実行できるかどうかの設定です。環境変数 PATH に登録されているディレクトリに保存されている実行が許可されたファイルは、コマンド名で実行できます。
たとえば、ls コマンドは「/usr/bin/ls」というパスに保存されている、すべてのユーザーに実行が許可されているファイルです。

```
$ ls -l /usr/bin/ls 【Enter】
-r-xr-xr-x   7 bin        bin          77852 Feb 16  2007 /usr/bin/ls
```

所有者に実行許可　所有グループに実行許可　その他のユーザーに実行許可

四色君：
環境変数 PATH の値を表示するにはどうすればいいのでしたっけ？

マリー先生：
変数にアクセスするには変数名の前に「$」が必要よ。したがって、文字列を表示する echo コマンドを使って次のようにすればいいの。

```
$ echo $PATH　【Enter】
/usr/bin:/usr/ccs/bin:/usr/contrib/bin:/usr/contrib/Q4/bin:/opt/perl/bin:/opt/
ipf/bin:/opt/nettladm/bin:/opt/fcms/bin:/opt/wbem/bin:/opt/wbem/sbin:/o
pt/sas/bin:/opt/graphics/common/bin:/opt/atok/bin:/usr/bin/X11:/usr/contri
b/bin/X11:/opt/sec_mgmt/bastille/bin:/opt/drd/bin:/opt/dsau/bin:/opt/dsau
/sbin:/opt/resmon/bin:/opt/gnome/bin:/usr/contrib/kwdb/bin:/opt/firefox:/
opt/mozilla:/opt/perl_32/bin:/opt/perl_64/bin:/opt/sfm/bin:/opt/swm/bin:/o
pt/sec_mgmt/spc/bin:/opt/ssh/bin:/opt/swa/bin:/opt/hpsmh/bin:/opt/gwlm/
bin:/opt/mx/bin:/opt/perf/bin:/opt/prm/bin:/opt/wlm/bin:/opt/mpi/bin:/opt
/vse/bin:/opt/caliper/bin:/opt/ignite/bin:/var/opt/netscape/server7/shared/b
in:/var/opt/netscape/server7/bin:/opt/sentinel/bin:/opt/java1.4/jre/bin:/opt/
spb/bin:/opt/thunderbird:/opt/langtools/bin:.
```

ディレクトリのパーミッション

ディレクトリのパーミッションは、UNIX 系 OS ではディレクトリはその下のファイルの一覧が格納されたある種のファイルであることをイメージするとわかりやすいでしょう。つまり、「r」（読み出し可）が許可されていないと、ls コマンドで一覧を表示することができません。

```
$ ls -ld samples　【Enter】
d-wx--x---  2 o2       users         96 Sep  6 15:34 samples ←読み出し許可がない
$ ls samples　【Enter】　一覧を表示しようとすると……
samples unreadable ←エラーになる
```

また、「w」（書き換え可）はファイルの一覧表を書き換えできるかどうかを表します。許可されていない場合にはその下にファイルを作成したり、ファイルを削除したりすることができません。

```
$ ls -ld test　【Enter】

dr-xr-xr-x  3 o2  wheel  102  9  6 22:50 test ←書き換え許可がない
$ rm test/uso　【Enter】　その下のファイルを削除しようとすると……
rm: test/uso: Permission denied ←エラーになる
```

なお、ディレクトリの場合の「x」（実行可）はファイルの場合と意味が異なり、そのディレクトリより下の階層にアクセスできるかどうかを表します。禁止マークのような意味合いでとらえてもよいでしょう。「x」が許可されていないと、cd コマンドでそのディレクトリに移動したり、find コマンドでそのディレクトリ以降を検索したりできなくなります。

```
$ ls -ld test2 【Enter】
drw-r-----  3 o2       users        96 Sep  7 14:27 test2 ←実行許可がない
$ ls test2 【Enter】 ←ls コマンドで一覧を表示しようとすると……
aDir    infotxt ←一覧は表示できる
$ cd test2 【Enter】 ←cd コマンドで移動しようとすると……
bash: cd: test2: Permission denied ←エラーになる
```

休憩時間：セキュリティ管理はますます重要に

最近、クレジットカード情報が漏洩して悪用されたなどといったニュースを日常的に耳にすることが多くなってきたと感じている方は少なくないでしょう。インターネットの普及に伴い、天文学的な規模のデータがコンピューターに蓄積されているようになってくると、そのセキュリティ対策はますます重要な課題となってきています。ただ、一言でシステムセキュリティ対策といっても、伝統的なパスワード管理やパーミッションによるファイルのアクセス制御といった基礎的なものから、ネットワークではファイアウォールや侵入検知システム、最近では RBAC（Role-based Access Control）による root 権限の分散化など多岐にわたり、そのすべてを完璧に把握することは容易ではありません。

HP-UX はセキュアで信頼性の高い OS としても定評があります。多少内容は高度になりますが、以下に HP-UX に備わっているさまざまなセキュリティ機能の中で注目すべき機能を全 7 回に渡って解説しています。ぜひ、一度目を通しておくとよいでしょう。

知っておくべきセキュリティ対策 -HP-UX のセキュリティを極める-

2 時間目：パーミッションを設定する

1 時間目の説明で UNIX におけるパーミッションの概要がつかめたと思います。2 時間目では既存のファイルのパーミッションを変更する方法を中心に説明します。chmod コマンドを使いますが、パーミッションの指定方法は記号による方法と、数値による方法の 2 種類があります。

記号によるパーミッションの設定

記号によってパーミッションを変更する場合の、chmod コマンドの書式は次のようになります。

```
chmod <ユーザー><オペレーター><アクセス権限> ファイル
```

ユーザー	
u	所有者（「User」の略）
g	所有グループ（「Group」の略）
o	その他のユーザー（「Other」の略）
a	すべてのユーザー（「All」の略、ugo を指定したのと同じ）

オペレーター	
+	許可を加える
-	許可を取り消す
=	設定する（元の設定をクリアする）

アクセス権限	
r	読み出し可
w	書き換え可
x	実行可

次にいくつかの例を示します。

1) 所有者と所有グループに対して書き換えを許可する
```
$ chmod ug+w sample.txt
```

2) すべてのユーザーに読み出しを許可する
```
$ chmod a+r sample.txt
```

3) 所有グループとその他のユーザーの書き換えを許可しない
```
$ chmod go-w sample.txt
```

四色君：
オペレーターの「+」と「=」の違いがよくわからないんですけど……。

マリー先生：
「+」はそれまでの設定はそのままで新たにアクセス権を追加し、「=」は、指定したユーザーに対するそれまでの設定をクリアして再設定するの。たとえば、「a=r」とした場合には、「すべてのユーザー」（a）に対して、「読み出し」（r）、「書き換え」（w）、「実行」（x）のそれまでの設定がいったんクリアされ、新たに「読み出し」（r）のみが許可されるようになるの。次の例を見て理解してね。

```
$ ls -l sample.txt 【Enter】
-rw-r-----  1 o2      users         12 Sep  6 16:30 sample.txt
←初期状態のパーミッションは「rw-r------」
$ chmod g+w sample.txt 【Enter】 ←所有グループに書き換えを許可する
```

```
$ ls -l sample.txt 【Enter】
-rw-rw----  1 o2      users        12 Sep  6 16:30 sample.txt
←パーミッションは「rw-rw-----」
$ chmod a=r sample.txt 【Enter】 ←すべてのユーザーに読み出しのみを許可する
$ ls -l sample.txt 【Enter】
-r--r--r--  1 o2      users        12 Sep  6 16:30 sample.txt
←パーミッションは「r--r------」
```

タックス君：
他人の作ったファイルのパーミッションを変更することはできるのですか？

マリー先生：
一般ユーザーの場合、パーミッションを変更できるのは所有者が自分自身、つまり通常は自分の作ったファイルだけなの。ただし、スーパーユーザーはすべてのファイルのパーミッションを変更できるわ。

数値によるパーミッションの設定

パーミッションを 3 桁の数値で指定するには、chmod を次の書式で実行します。

```
chmod <3 桁の 8 進数> ファイルのパス
```

各桁は順に「所有者」「所有グループ」「その他のユーザー」のアクセス権に対応しています。
8 進数の数値の作り方を説明しましょう。まず、「読み出し」（r）、「書き換え」（w）、「実行」（x）の順に、許可されていれば「1」、許可されていなければ「0」と表して並べます。これを 2 進数と見立てて 8 進数の数値に変換します。たとえば、「読み出し」（r）と「書き換え」（w）のみが許可されている場合、8 進数の数値は「6」になります。

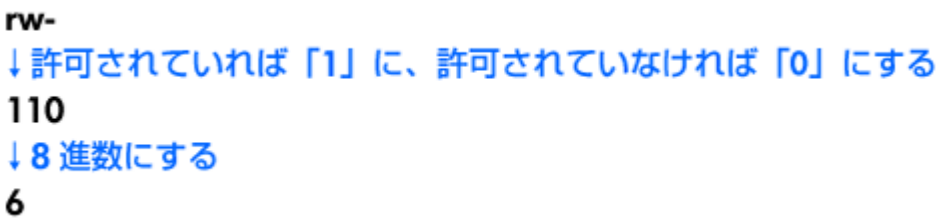


図 2：パーミッションを 8 進数で表現する

これを「所有者」「所有グループ」「その他のユーザー」ごとに行って数値を順につなげれば、アクセス権を表す 3 桁の 8 進数ができるというわけです。たとえば、所有者が読み出しと書き換え可（rw-）、所有グループは読み出しのみ可（r--）、その他のユーザーは許可なし（---）の場合には「640」となります

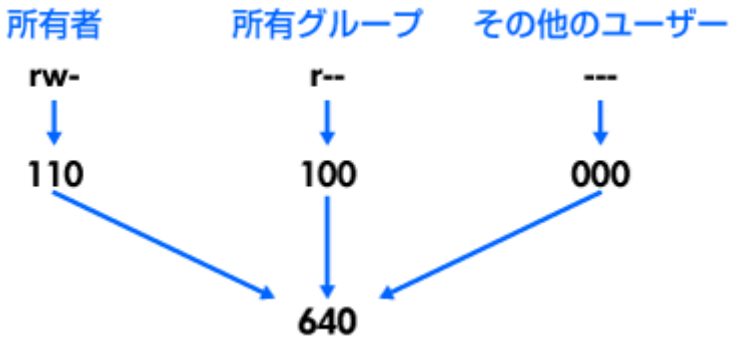


図 3：パーミッションを 3 桁の 8 進数で表現する

タックス君：
2 進数から 8 進数にするのはどうやるのでしたっけ。そういえば、中学生の時に習ったような……。

マリー先生：
次の表に、3 桁の 2 進数から 8 進数への変換をまとめておくわね。

2 進数	8 進数
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

四色君：
複数のファイルのパーミッションを、まとめて設定することはできますか？

マリー先生：
chmod コマンドの引数に「*」や「?」といったシェルのワイルドカードを使うこともできるわ。たとえば、カレントディレクトリの下の拡張子が「.txt」のすべてのファイルのパーミッションを「600」にするには、次のようにするの。

```
$ chmod 600 *.txt
```

タックス君：
指定したディレクトリ以下のファイルやディレクトリをまとめて変更することもできますか？

マリー先生：
その場合は「-R」オプションを指定すれば OK よ。「-R」は「Recursive」（再帰的）の頭文字を表し、指定したディレクトリ以下を順にたどって、まるごとパーミッションを設定するの。
たとえば、samples ディレクトリ以下を丸ごと、「a+r」（すべてのユーザーに対して読み出し権限を追加）に設定するには、次のようにすればいいわ。

```
$ chmod -R a+r samples/
```

練習問題

第 1 問：パーミッションの説明に関して、正しくないのはどれか？

a)　読み出し権限「r」がないファイルは表示できない
b)　書き換え権限「w」がないファイルは削除できない
c)　読み出し権限「r」がないディレクトリの下の一覧は表示できない
d)　書き換え権限「w」がないディレクトリの下のファイルは削除できない

b)　書き換え権限「w」がないファイルは削除できない
ファイルが削除できるかどうかは、それが保存されているディレクトリの書き換え権限に依存する。

第 2 問：ファイル「sample.txt」に対して、所有グループへの書き換えを許可する権限を追加するコマンドはどれか？ ただし、その他のパーミッションは変更しないものとする。

a)　chmod g+w sample.txt
b)　mod gw sample.txt
c)　chmod g=w sample.txt
d)　chmod g-w sample.txt

a)　chmod g+w sample.txt
オペレーターに「+」を使用することに注意。「=」を使うと、対象となるユーザーに設定されているその他のパーミッションがクリアされてしまう。

第 3 問：カレントディレクトリの下の、拡張子が「.txt」のファイルのパーミッションを、所有者のみ読み書き可に設定するコマンドはどれか？

a)　chmod 400 *.txt
b)　chmod 666 *.txt
c)　chmod 133 *.txt
d)　chmod 600 *.txt

d)　chmod 600 *.txt
所有者のパーミッションを「rw-」、つまり読み出し可「r」書き換え可「w」に設定する場合の 8 進数の値は「6」になる。

第 4 問：samples ディレクトリ以下の、すべてのファイルとディレクトリに対して、所有者および所有グループへの書き換え権限を追加するコマンドはどれか？

a)　chmod ug+w samples
b)　chmod -R ug+w samples
c)　chmod -a ug+w sample
d)　chmod -r ug+w samples

b)　chmod -R ug+w samples
指定したディレクトリ以下のパーミッションを再帰的に変更するには、chmod コマンドを「-R」オプションを指定して実行する。

UNIX の教科書「運用編」

第 3 日目：ファイルの安全管理（その 2）

2009 年 11 月　大津 真

今回も、引き続きファイルの安全管理を取り上げます。前回は、所有者・所有グループ・その他のユーザーに対して、ファイルの読み出し・書き換え・実行に関する許可を与えるパーミッションについて学びましたが、UNIX ではさらに「setuid ビット」

「setgid ビット」「sticky ビット」という 3 つの特殊なパーミッションを用意しています。これらがどんなものなのか、どういうふうに使えるのかを学んでいきましょう。

マリー先生：
前回は、システムの安全管理において基本中の基本と言えるファイルのアクセス権限、つまりパーミッションについて説明したわけだけど、今回はその続きとしてファイルのモードという概念について説明していきましょう。その前に、前回のパーミッションについて何か質問ありますか？

四色君：
「ls -l」でファイルの詳細情報を表示したときにシンボリックリンクにもパーミッションが表示されますが、これは何を表しているんですか？

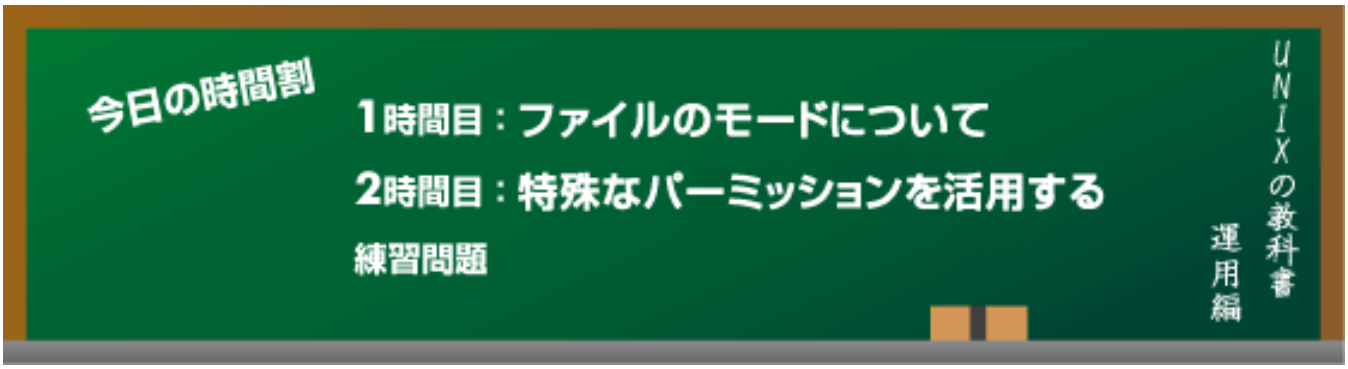
マリー先生：
「ls -l」で表示されるシンボリックのパーミッションは意味がまったくないの。元のファイルとのパーミッションとも関係ないから、無視してね。

```
$ ls -l　【Enter】
total 16
-rw-r-----   1 o2       users        591 Oct 10 14:47 orig.txt
 ←元のファイル
lrwxr-x---   1 o2       users          8 Oct 10 14:49 slink.txt
 -> orig.txt
 ←シンボリックリンク。こちらのパーミッションには意味がない
```

タックス君：
それじゃあ chmod コマンドでシンボリックリンクのパーミッションを変更すると、どうなるのでしょう？

マリー先生：
その場合、元のファイルのパーミッションが変更されるの。

```
$ chmod 666 slink.txt　【Enter】
 ←シンボリックリンクのパーミッションを変更しようとすると……
$ ls -l　【Enter】
total 16
-rw-rw-rw-   1 o2       users        591 Oct 10 14:47 orig.txt
 ←元のファイルのパーミッションが変更される
lrwxr-x---   1 o2       users          8 Oct 10 14:49 slink.txt
 -> orig.txt
 ←シンボリックリンクのパーミッションはそのまま
```



1 時間目：ファイルのモードについて

前回説明したファイルの許可属性であるパーミッションは、ユーザーをファイルの「所有者」「所有グループ」「その他のユーザー」という 3 種類に分け、それぞれに「読み出し可」「書き換え可」「実行可」という 3 つの属性値を表すものです。「読み出し可」や「書き換え可」などの設定値は、「1」（オン）か「0」（オフ）か、つまり 1 ビットで表せられるので、ユーザーの分類ごとに 3 ビットずつ、全体として 9 ビットの値となります。これらは各ファイルやディレクトリごとの属性として inode と呼ばれるディスク内の領域に保存されています。

図 1：パーミッションは 9 ビットの値からなる

さて、ファイルの許可属性は、これらの基本パーミッション部分の上位に特殊なパーミッションとしてさらに 3 ビットが用意されています。3 ビットは上位の桁から順に「setuid ビット」「setgid ビット」「sticky ビット」です。なお、基本パーミッションと特殊なパーミッションを合わせてファイルの「モード」と呼ぶことがあります。

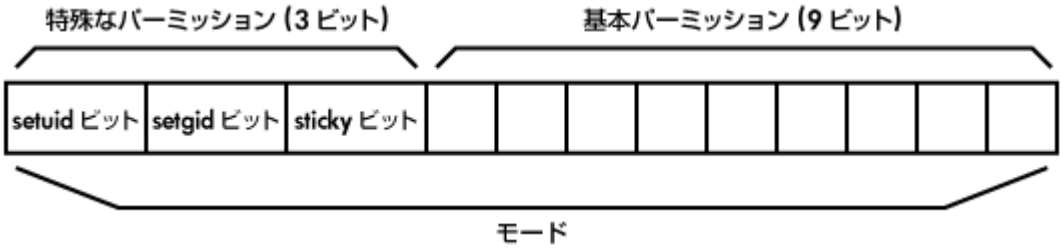


図 2：基本パーミッションと特殊なパーミッションを合わせてモードと呼ぶ

setuid ビット

UNIX では実行中の個々のプログラムの単位を「プロセス」と呼びます。このときプロセスの実行権限を決めるユーザーID を「実効ユーザーID」（Effective User ID）と呼びます。通常、プロセスはそれを実行したユーザーの権限で実行されます。たとえばユーザー「taro」が ls コマンドを実行した場合、ユーザー「taro」のユーザーID が実効ユーザーID となります。それに対して setuid を設定したプログラムの場合、実効ユーザーID はそのファイルの所有者となります。

タックス君：
setuid ビットの働きはなんとなくわかってのですが、なぜ、実効ユーザーID をファイルの所有者にする必要があるのでしょう？

マリー先生：
setuid ビットは、実行時に root 権限が必要なコマンドを一般ユーザーが実行する必要がある場合に使用されることが多いわ。setuid ビットが必要になる典型的な例が、一般ユーザーが自分のパスワードを変更するときね。パスワードを変更する passwd コマンドを「ls -l」で表示してみて。コマンドの保存先のパスは which コマンドでわかるわよ。

タックス君：
まず、passwd コマンドコマンドの保存先を which コマンドで調べてみますね。

```
$ which passwd　【Enter】
/usr/bin/passwd
```

次に、これを「ls -l」で表示すると……。

```
$ ls -l /usr/bin/passwd　【Enter】
-r-sr-xr-x   4 root      bin        150932 May  3  2007 /usr/bin/passwd
 ↑
 所有者の実行許可が「s」となっている
```

所有者は「root」ですが、所有者の実行許可を表す部分が「s」になっていますね。

マリー先生：
そうね。setuid ビットがセットされたファイルは、所有者の実行部分の表示が「s」となるの（実行許可がないファイルの場合には大文字で「S」）。passwd コマンドの変更内容は /etc/passwd や/etc/shadow といったファイルに書き込まれるのよ。

四色君：
それでは、/etc/passwd の情報を「ls -l」コマンドで表示してみましょう。

```
$ ls -l /etc/passwd　【Enter】
-r--r--r--   1 root      root        1012 Sep  7 14:08 /etc/passwd
```

すべてのユーザーに対して書き換えが禁止されていますね。

マリー先生：
書き換え許可がないファイルでも root ユーザーは書き換えが可能よ。さっき見たとおり passwd コマンドには setuid ビットがセットされているから、一般ユーザーが passwd コマンドを実行すると実効ユーザーID が「root」になって――つまり root の権限で実行されて、一般ユーザーには書き換えができない/etc/passwd ファイルが変更できるようになるというわけね。

setgid ビット

setgid ビットは、setuid ビットのグループ版です。setgid ビットのセットされたコマンドを実行すると、実効グループ ID がそのファイルの所有グループとなります。
setgid ビットの設定されたファイルは、所有グループの実行許可部分の表示が「s」（実行許可がない場合には大文字で「S」）となります。

```
$ ls -l /bin/stmkfont 【Enter】
-r-xr-sr-x   1 bin       bin        245760 Jan 13  2007 /bin/stmkfont
    ↑
 所有グループの実行許可が「s」となっている
```

setgid ビットをディレクトリにセットすると

setgid ビットをディレクトリに対してセットすると、複数のユーザーから構成されるグループでディレクトリ以下を共有するときに便利になります。setgid ビットを設定していない場合、ディレクトリの下にファイルを作成すると、作成者のプライマリグループが所有グループになります。一方、setgid ビットをセットしたディレクトリの場合には、ディレクトリの所有グループがその下に作成したファイルに引き継がれるのです。
たとえば、setgid ビットがセットされた、所有グループが「newprj」のディレクトリ「newproject」があるとします。

```
$ ls -ld newproject/ 【Enter】
drwxrws---   2 root      newprj        96 Oct 10 22:08 newproject/
                              ↑所有グループが「newprj」
```

プライマリグループが「users」で、そのほかに「newprj」グループに属するユーザー「o2」がいるとします。ユーザー「o2」が、newproject ディレクトリの下にファイルを作成すると所有グループが「newprj」になります（setgid ビットがセットされていない場合には所有グループが「users」になります）。

```
$ id 【Enter】 ←id コマンドで属するグループを確認
uid=111(o2) gid=20(users) groups=108(newprj) ←プライマリグループは「users」
$ mkdir sampledir 【Enter】 ←ディレクトリを作成
$ ls -l newproject/ 【Enter】
total 0
drwxr-s---   2 o2        newprj        96 Oct 10 22:13 sampledir
                              ↑所有グループが引き継がれ「newprj」になる
```

四色君：
ディレクトリの所有者は引き継がれないのですか？

マリー先生：
引き継がれないわ。所有者は作成したユーザーになるの。

休憩時間：SMH によるユーザーの管理

第 1 回の休憩時間で紹介した SMH（HP System Management Homepage）でも、ユーザーやグループの管理が行えます。
たとえばユーザーを追加するには SMH のメイン画面から「Tools」を選択します。

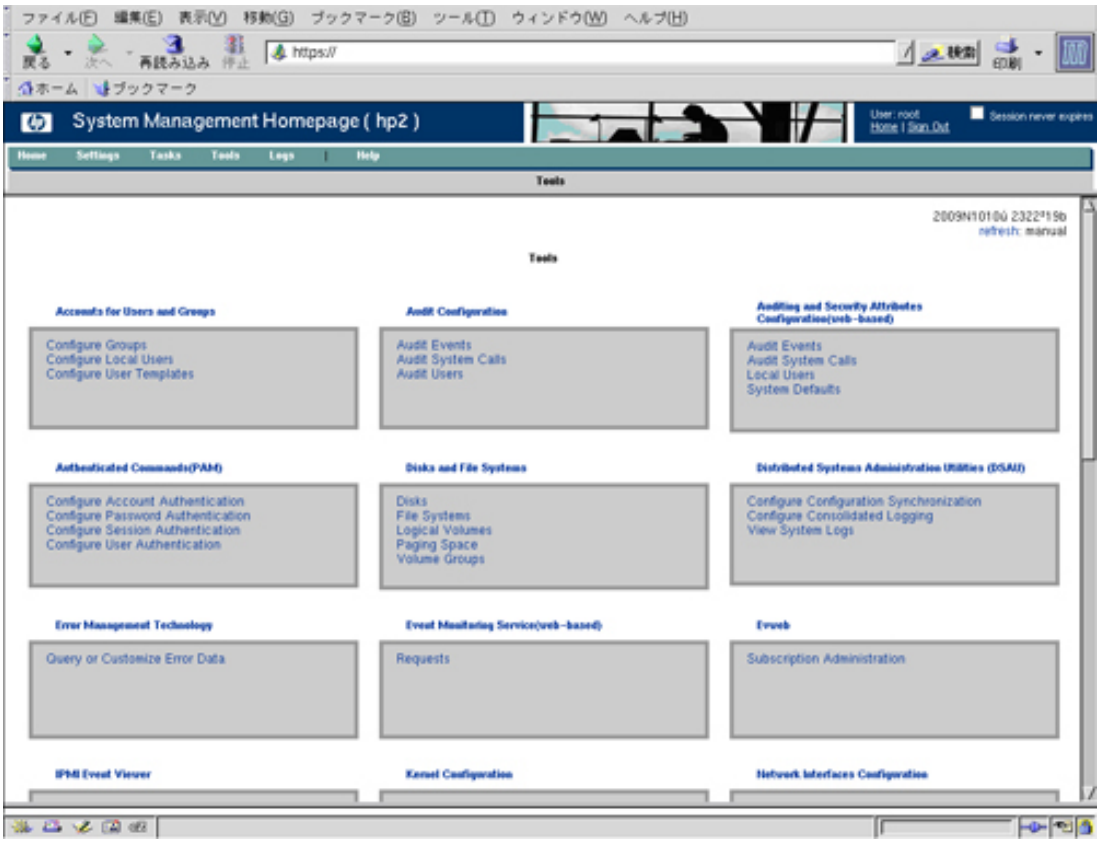


図 3：Tools 画面

次に、「Configure Local Users」を選択するとユーザーの管理画面が表示され、「ローカルユーザー」に現在登録されているユーザーの一覧が表示されます。

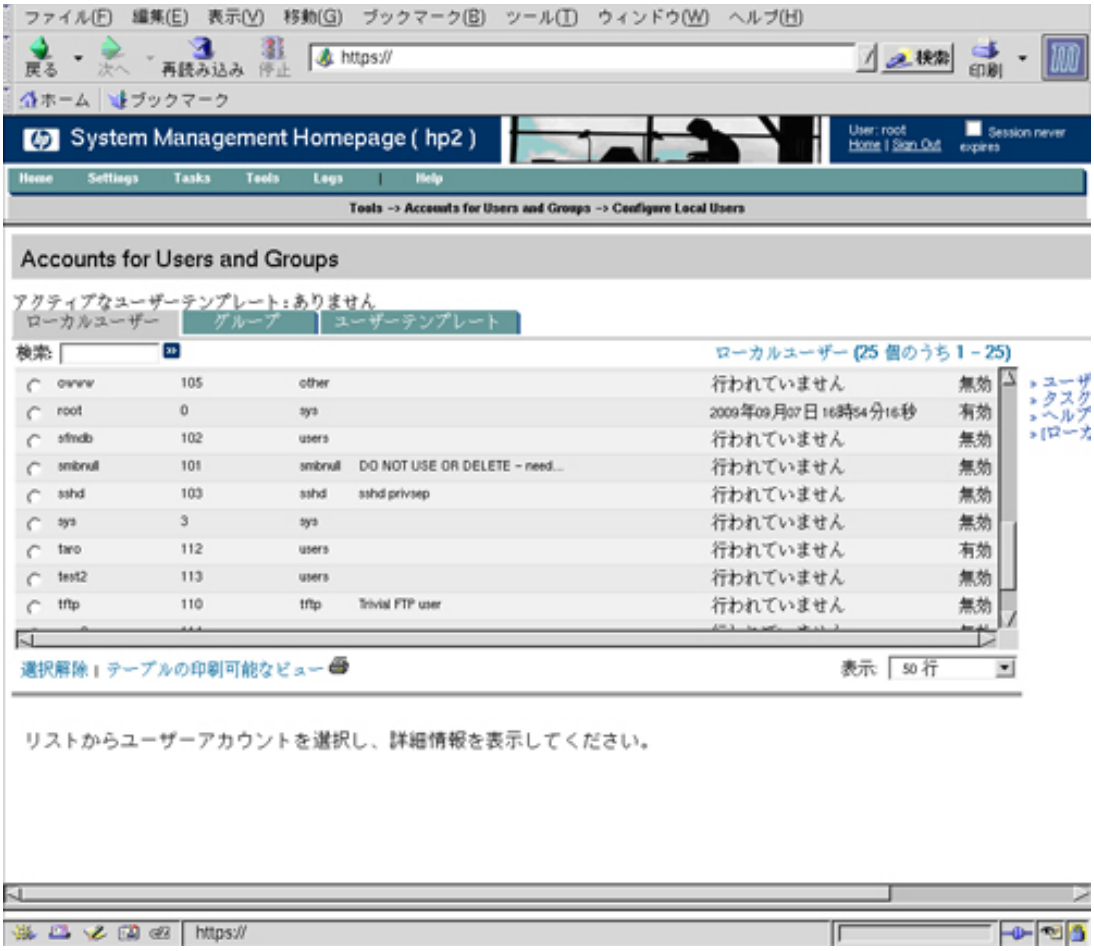


図 4：ユーザー一覧画面

ユーザーを追加するには「ユーザーカウントの追加」をクリックします。すると、ユーザー情報を入力する画面が表示されるので、ユーザー名やパスワードなどの情報を入力して「追加」をクリックすると、ユーザーが登録されます。

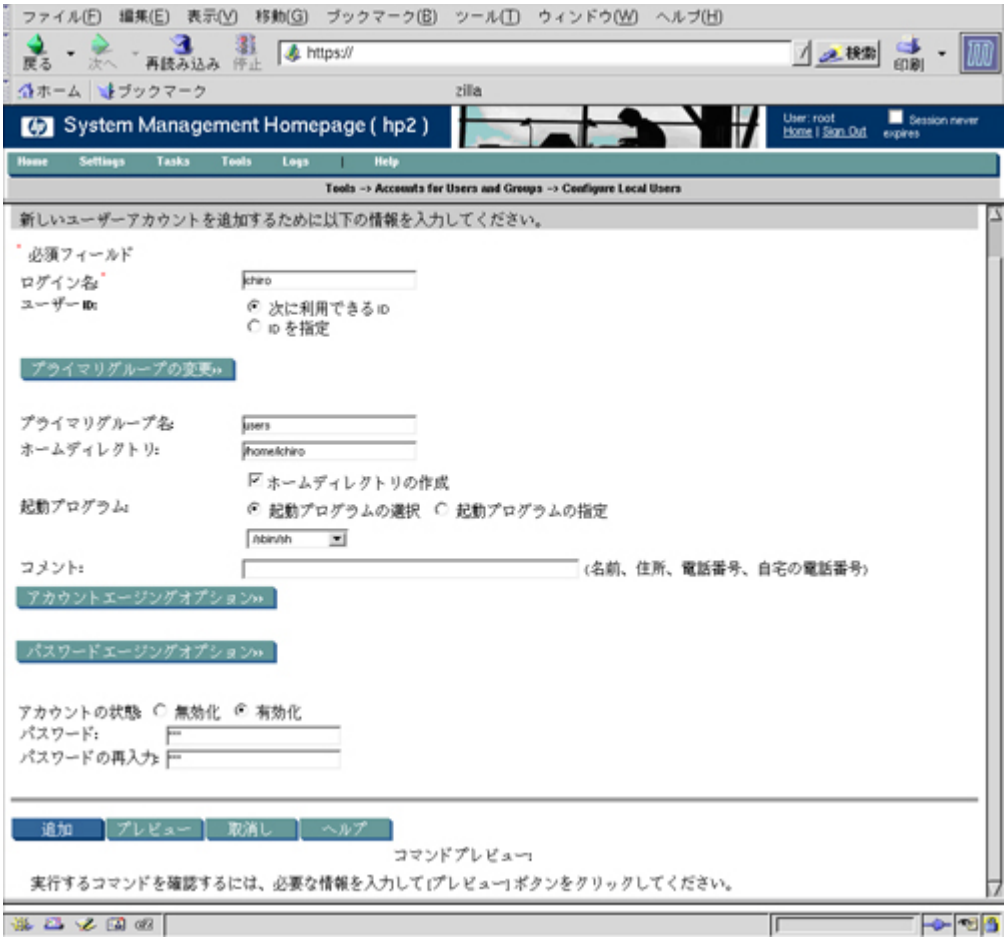


図 5：ユーザー情報を入力して「追加」をクリックすれば完了

2 時間目：特殊なパーミッションを活用する

2 時間目でも引き続き、特殊なパーミッションについて説明しましょう。まずは sticky ビットからです。

sticky ビット

setgid ビットと同じく sticky ビットも、あるディレクトリ以下をグループで管理している場合に有効です。sticky ビットがセットされていない場合、所有グループに「書き換え」が許可されているディレクトリは、グループに属するユーザーなら誰でもその下のファイルを削除できてしまいます。一方、ディレクトリに sticky ビットをセットしておくと、所有者以外はファイルを削除できなくなります。sticky ビットがセットされたファイルは「その他のユーザー」の実行許可部分の表示が「t」となります（実行許可がない場合には「T」）。

```
$ ls -ld project/　【Enter】
drwxrwx--t   2 root      users         96 Oct 10 21:46 project/
        ↑
 その他のユーザーの実行許可が「t」
```

四色君：
ほかのユーザーが削除するときにエラーが出るのですか？

マリー先生：
そうね、たとえばこの sticky ビットが設定された project ディレクトリに「sample」というファイルがあります。このファイルの所有者は「taro」、所有グループは「users」ね。

```
$ ls -l project/　【Enter】
total 16
-rw-rw----   1 taro      users         5 Oct 10 21:55 sample
```

これを、グループ「users」に属する別のユーザーが削除しようとすると次のようなエラーになるの。

```
$ rm project/sample　【Enter】
 ←別のユーザーが削除しようとすると……
rm: project/sample not removed.  Not owner ←エラーになる
```

特殊なパーミッションを設定する

基本パーミッションの設定には chmod コマンドを使用しましたが、特殊なパーミッションを設定する場合にも chmod コマンドを使います。

```
chmod 設定値 ファイル
```

基本パーミッションと同じく、特殊なパーミションの設定値は記号と数値、どちらかの形式で指定します。まず、記号による場合には次のように指定します。

	設定	削除
setuid ビット	u+s	u-s
setgid ビット	g+s	g-s
sticky ビット	+t	-t

たとえば、newproject ディレクトリに setgid ビットをセットするには次のようにします。

```
# chmod g+s newproject/　【Enter】　←setgid ビットをセット
# ls -ld newproject/　【Enter】　←「ls -ld」で確認
```

```
drwxrws---   3 root      newprj        96 Oct 10 22:13 newproject/
```

さらに sticky ビットをセットするには次のようにします。

```
# chmod +t newproject/  【Enter】  ←sticky ビットをセット
# ls -ld newproject/  【Enter】  ←「ls -ld」で確認
drwxrws--T   3 root      newprj        96 Oct 10 22:13 newproject/
```

タックス君：
setuid ビットを使用すると root 権限でコマンドか実行できるようになって便利なのですが、自由に設定してかまわないのですか？

マリー先生：
setuid ビットや setgid ビットをコマンドに設定すると、一般ユーザー以上の権限でコマンドを実行できる場合が多いので、セキュリティ的にはあまり好ましいものではないの。したがって、必要のないファイルにそれらのビットを設定すべきではないわ。

数値による特殊なパーミッションの設定

数値によって特殊なパーミッションを設定することもできます。前回説明したように基本パーミッションの場合には 3 桁の 8 進数で指定しました。特殊なパーミッションは 3 ビットあり、モード全体で 12 ビットとなるため、4 桁の 8 進数で指定します。たとえば、setgid ビットと sticky ビットを立てて、所有者と所有グループに読み書き実行を許可するための数値は「3770」となります。

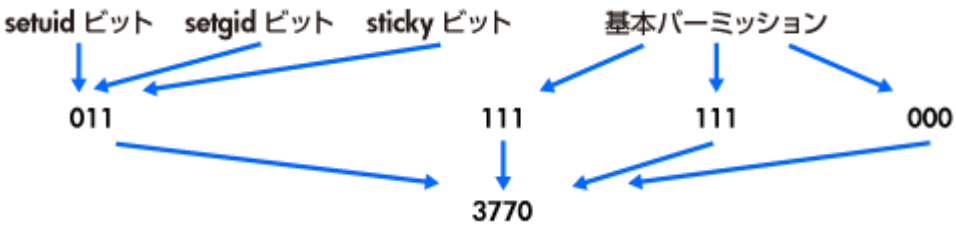


図 6：モードは 4 桁の 8 進数で表現する

次に、project ディレクトリのモードを「3770」に設定する例を示します。

```
# chmod 3770 project  【Enter】  ←モードを「3770」に設定
# ls -ld project  【Enter】  ←「ls -ld」で確認
drwxrws--T   3 root      newprj        96 Oct 10 22:04 project/
```

四色君：
3 桁の 8 進数でパーミッションを設定した場合、特殊なパーミッション部分はどうなるのですか？

マリー先生：
3 桁で指定した場合、先頭に「0」が指定されているものとみなされるの。そのため特殊なパーミッションはクリアされてしまうので注意してね。

ファイルを作成したときのパーミッション

ユーザーがファイルを作成したときのパーミッションは「umask 値」（ファイル生成マスク値）と呼ばれる値で決定されます。現在の umask 値は umask コマンドで表示できます。

```
$ umask 【Enter】
027 ←現在の umask 値
```

ファイルの場合「666」を、ディレクトリの場合「777」を、それぞれ umask 値でマスクした値が作成時のパーミッションとなります。なお、マスクとは 2 つの 2 進数同士を比較して、対応するマスク値の「0」の部分をそのまま、「1」の部分を「0」にすることを言います。
たとえば、umask 値が「027」の場合、新規ファイルのパーミッションは「640」（「rw-r-----」）となります。

図 7：umask 値が「27」のときの新規ファイルパーミッション

同様に、新規ディレクトリのパーミッションは「750」（drwxr-x---）となります。

図 8：umask 値が「27」のときの新規ディレクトリパーミッション

タックス君：
umask 値を変更することはできるのですか？

マリー先生：
umask コマンドの引数にマスク値を指定すればできるわ。たとえば、所有者以外には読み書き実行を許可しないような umask 値「077」を設定するには、次のようにするの。

```
$ umask 077 【Enter】     ←umask 値を「077」に
$ touch testfile 【Enter】 ←ファイルを作成
$ mkdir testDir 【Enter】 ←ディレクトリを作成
$ ls -l 【Enter】         ←「ls -l」コマンドで確認
total 0
drwx------  2 o2      users        96 Oct 10 22:57 testDir
-rw-------  1 o2      users         0 Oct 10 22:57 testfile
```

四色君：
umask 値はどこで設定されているのですか？

マリー先生：
通常、ログイン時に実行されるシェルの環境設定ファイルで設定するの。POSIX シェルの場合には/etc/profile で設定されているわ。設定をユーザーごとに変更したい場合には「~/.profile」で設定するといいわよ。

ファイルをコピーした場合のパーミッション

umask 値はファイルやディレクトリをコピーした場合のパーミッションにも影響を与えます。コピーしたファイルのパーミッションは、コピー元のファイルのパーミッションを umask 値でマスクした値となります。たとえば、現在の umask 値が「077」の場合には、パーミッションが「666」のファイルをコピーするとコピー後のパーミッションは「600」（rw-------)になります。

```
$ umask 【Enter】
077 ←umask 値は「077」
$ cp testfile newfile 【Enter】 ←ファイルをコピーする
$ ls -l 【Enter】
total 0
-rw-------  1 o2      users         0 Oct 10 23:01 newfile ←コピーされたファイル
-rw-rw-rw-  1 o2      users         0 Oct 10 22:57 testfile ←元のファイル
```

四色君：
コピー後のパーミッションを変更したくない場合にはどうすればいいのですか？

マリー先生：
その場合には「-p」オプションをつけて cp コマンドを実行すればいいわ

```
$ cp -p testfile newfile2　【Enter】
 ←「-p」オプションをつけて cp コマンドを実行
$ ls -l　【Enter】
total 0
-rw-------   1 o2       users          0 Oct 10 23:01 newfile
 ←「-p」オプションなしでコピーされたファイル
-rw-rw-rw-   1 o2       users          0 Oct 10 22:57 newfile2
 ←「-p」オプションをつけてコピーされたファイル
  （パーミッションは変わらない）
-rw-rw-rw-   1 o2       users          0 Oct 10 22:57 testfile
 ←元のファイル
```

練習問題

第 1 問：setgid ビットについて正しい説明はどれか？

a)　ファイルに対して setgid ビットをセットするとファイルの所有者が実効ユーザーとなる
b)　ディレクトリに setgid ビットをセットすると、その下に作成したファイルに親ディレクトリの所有グループが引き継がれる
c)　ディレクトリに setgid ビットをセットすると、ファイルの作成者以外は削除できなくなる
d)　setgid ビットを設定したディレクトリは「ls -l」の表示でその他のユーザーの実行部分が「t」となる

b)　ディレクトリに setgid ビットをセットすると、その下に作成したファイルに親ディレクトリの所有グループが引き継がれる
a)は setuid ビットの説明、 (c)(d)は sticky ビットの説明である。

第 2 問：testdir ディレクトリに setgid ビットをセットするコマンドはどれか？

a)　chmod g+s testdir
b)　chmod u+s testdir
c)　chmod +s testdir
d)　chmod u+t testdir

a)　chmod g+s testdir
setgid ビットを設定するには「chmod g+s ディレクトリ」コマンドを実行する。

第 3 問：umask 値を「022」に設定するコマンドはどれか？

a)　mask 022
b)　umask 022
c)　set mask 022
d)　chmod 022

b)　umask 022
umask 値の設定・確認には umask コマンドを使用する。

第 4 問：現在の umask 値が「022」の場合に新規ファイルを作成したときのパーミッションはどれか？

a)　666（rw-rw-rw-）
b)　622（rw--w--w-）
c)　644（rw-r--r--）
d)　755（rwxr-xr-x）

c)　644（rw-r--r--）
新規ファイルのパーミッションは「666」を umask 値でマスクした値となる。

UNIX の教科書「運用編」

第 4 日目：ネットワークの基本を理解する

2009 年 12 月　大津 真

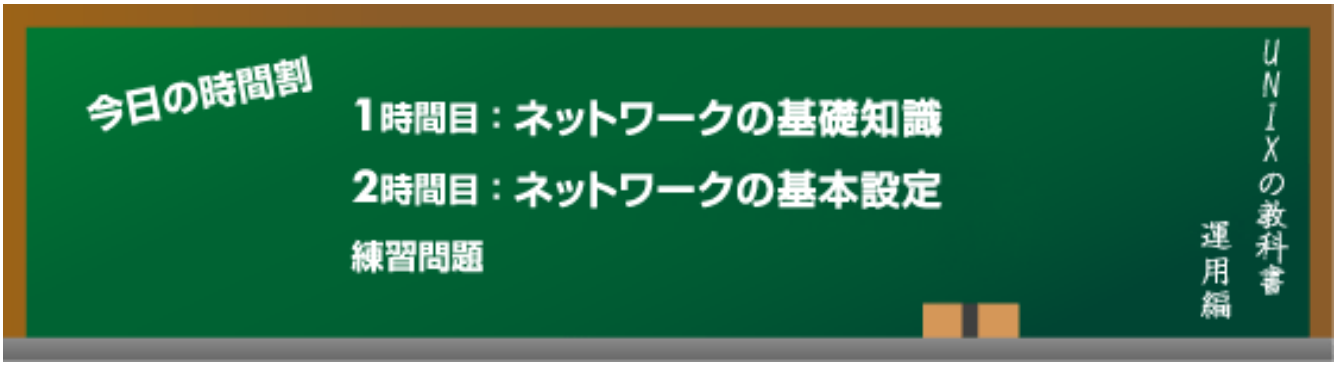
今回は、ネットワークの基本について学びましょう。1 時間目は TCP/IP ネットワークの基礎知識について説明します。ネットワークを介して通信を行う上での取り決めをプロトコルと呼びますが、ここで勉強する TCP/IP プロトコルは、よく知られているようにインターネットでも使われているプロトコルです。2 時間目は、HP-UX におけるネットワークの基本設定を解説していきます。

マリー先生：
今回は、TCP/IP ネットワークの基礎知識と基本設定について説明しましょう。その前に、前回のファイルの安全管理について何か質問ありますか？

タックス君：
前回の説明では、必要のないコマンドに setuid ビットや setgid ビットを設定すべきではないということでしたが、それらの特殊なパーミッションが設定されているファイルを見つけることはできるのですか？

マリー先生：
find コマンドの「-perm」オプションで検索することができるわ。たとえば、/usr/bin ディレクトリ以下で setuid ビットが設定されているファイルを検索するには、次のようにするの。

```
# find /usr/bin/ -perm -u=s 【Enter】
/usr/bin/mediainit
/usr/bin/bdf
/usr/bin/rcp
/usr/bin/remsh
/usr/bin/at
/usr/bin/crontab
/usr/bin/mail
/usr/bin/rmail
/usr/bin/df
～略～
```



1 時間目：ネットワークの基礎知識

ネットワークを介して通信を行う上での取り決めのことをプロトコルと呼びます。これまでさまざまなネットワーク・プロトコルが開発されてきましたが、インターネットの爆発的な普及によって現在では TCP/IP プロトコルが主流になっています。

ここで、ちょっと余談になります。TCP/IP は、今ではインターネットやイントラネットで標準的に使われるプロトコルですが、そのきっかけは UNIX だったとご存知でしょうか？誕生は 1970 年代の初頭のアメリカ防総省の研究開発でしたが、その技術を標準実装して普及させたのが UNIX だったのです。普段使っているインターネットの URL が「\」ではなく「/」で構成されてい

るのは、そうした背景があります。ですので、これまで UNIX を勉強してきたことが、実はネットワークの理解に非常に役に立つのです。1 時間目では、TCP/IP に関する基礎知識について説明しましょう。

IP アドレスは 32 ビットの数値

TCP/IP では、「IP アドレス」と呼ばれるアドレスによってそれぞれのコンピュータを識別します。なお、IP アドレスには伝統的な IPv4（IP バージョン 4）と、次世代 IP である IPv6 のアドレスがありますが、ここでは、伝統的な IPv4 について説明します。

IP アドレスは 32 ビットの長さを持つ数値ですが、人間にとってわかりやすいように、1 オクテット、つまり 8 ビットずつに分けてピリオド「.」で区切った 10 進数として記述します。たとえば、「192.168.1.0.2」といったように表記されます。

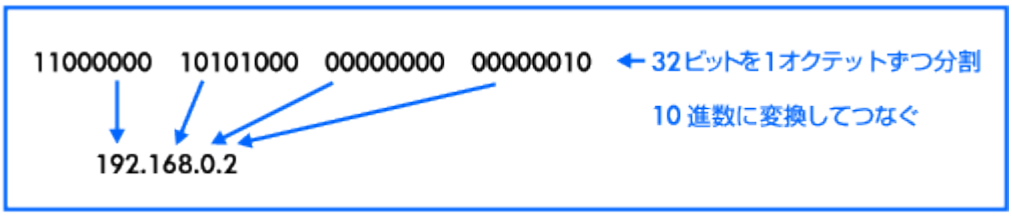


図 1：IP アドレスの表記方法

四色君：
「オクテット」って、いわゆる「バイト」のことですか？

マリー先生：
そう、一般的には同じね。本来 1 バイトは 1 文字を表す単位で、処理系によってビット数が違っていたの。それで、ネットワークの世界では 8 ビットであることを明示する意味で「バイト」より「オクテット」という用語がつかわれることが多いの。

タックス君：
IP アドレスは 1 台のマシンに 1 つずつ割り当てられるのですか？

マリー先生：
実際には IP アドレスはネットワーク・インターフェイスに割り当てられるアドレスなの。したがって 1 台のマシンでも複数のネットワーク・インターフェイスがあれば、複数の IP アドレスを割り当てることもできるのよ。

タックス君：
システムにインストールされているネットワーク・インターフェイスを確認するにはどうしたらいいでしょう。

マリー先生：
方法はいくつかあるけど、簡単なのは nwmgr コマンドを引数なしで実行することかな。

```
# nwmgr 【Enter】
```

```
Name/          Interface Station          Sub-    Interface      Related
ClassInstance  State     Address         system  Type           Interface
============== ========= ============== ======== ============== =========
lan0           UP        0x000E7F7E108C iether   1000Base-T
lan1           DOWN      0x000E7F7E108D iether   1000Base-T
lan900         DOWN      0x000000000000 hp_apa   hp_apa
lan901         DOWN      0x000000000000 hp_apa   hp_apa
lan902         DOWN      0x000000000000 hp_apa   hp_apa
lan903         DOWN      0x000000000000 hp_apa   hp_apa
lan904         DOWN      0x000000000000 hp_apa   hp_apa
```

この例では、「lan0」と「lan1」という名前の「1000Base-T」、いわゆるギガビットインターフェイスが 2 つあることがわかるわ。「Interface State」が「UP」の「lan0」だけが利用可能な状態ね

サブネットマスク

IP アドレスの前半部分をネットワーク部、後半部分をホスト部と呼びます。前者はネットワークを識別する部分で、後者はそのネットワーク内のホストを示します。ネットワーク部が異なれば、別のネットワークに属するものと見なされます。ネットワーク同士を接続するには「ルーター」が必要です。
IP アドレスの中でどこまでがネットワーク部であるかを示す値が「サブネットマスク」です。サブネットマスクは、ネットワーク部を「1」、ホスト部を「0」とする 32 ビットの 2 進数ですが、通常は IP アドレスを同じように 8 ビットずつピリオド「.」で区切った 10 進数で表されます。
たとえば、IP アドレスが「192.168.0.1」、サブネットマスクが「255.255.255.0」であれば、ネットワーク部は「192.168.0」、ホスト部は「1」になります。なお、この場合、ホストアドレス部は 8 ビットとなり、ホストに割り当てることができる IP アドレスは「192.168.0.0」～「192.168.0.255」の最大 256 個（正確には特殊な用途に予約されているアドレスがあるので 254 個）となります。

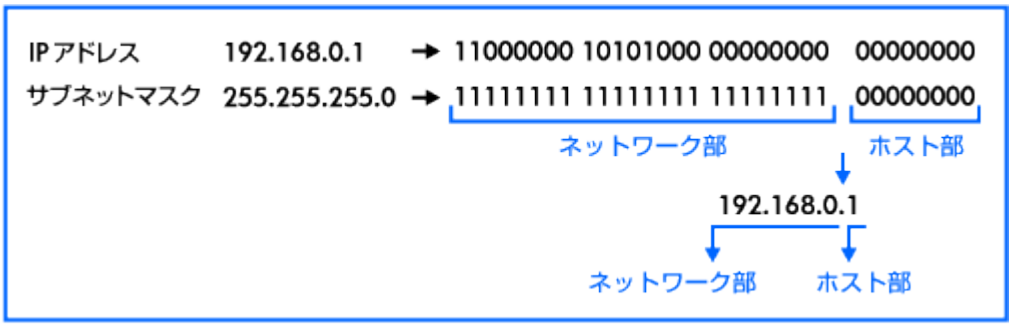


図 2：IP アドレスとサブネットマスク

なお、IP アドレスが「192.168.0.x」でサブネットマスクが「255.255.255.0」のネットワークは、ネットワーク部が 24 ビット（8 オクテット×3）なので「192.168.0.0/24」と表記されます。

四色君：
現在設定されている IP アドレスやサブネットマスクを確認するにはどうすればいいのですか？

マリー先生：
「ifconfig インターフェイス」コマンドを実行すると簡単よ。たとえば、「lan0」インターフェイスを確認するには次のようにするの。

```
# ifconfig lan0 【Enter】
lan0: flags=1843
      inet 192.168.1.14 netmask ffffff00 broadcast 192.168.1.255
        ↑                    ↑                  ↑
      IP アドレス    サブネットマスク    ブロードキャスト
```

グローバルアドレスとプライベートアドレス

インターネットに直接接続されているマシンは正式な機関から割り当てられた IP アドレスが設定されている必要があります。これを「グローバルアドレス」と呼びます。それに対して、LAN 内に限って自由に割り振ってよいことになっている IP アドレスに「プライベートアドレス」があります。たとえば、「192.168.0.0 ～ 192.168.255.255」の範囲の IP アドレスはプライベートアドレスです。なお、プライベートアドレスとグローバルアドレスとの変換にはルーターが必要です。

経路情報とデフォルトゲートウェイについて

TCP/IP ネットワークでは「パケット」（小包）という単位でデータを転送します。ネットワーク同士はルーター（ゲートウェイ）を介して接続されますが、ひとつまたは複数のルーターがある場合に、どのネットワークに宛てたパケットはどのルーターを通ればよいかといった情報を「経路情報」と呼び、経路情報が登録されている表のことを「ルーティング・テーブル」といいます。
このとき IP パケットのデフォルトの経由先のことを「デフォルトルート」あるいは「デフォルトゲートウェイ」と呼びます。たとえばルーターを介してインターネットに接続されているマシンの場合、そのルーターの IP アドレスがデフォルトゲートウェイとなります。

マリー先生：
現在設定されている経路情報が登録されているルーティング・テーブルは「netstat -nr」コマンドで確認できるわ。

```
# netstat -nr 【Enter】
Routing tables
Destination          Gateway           Flags Refs Interface  Pmtu
127.0.0.1            127.0.0.1         UH    0    lo0        32808
192.168.1.14         192.168.1.14      UH    0    lan0       32808
192.168.1.0          192.168.1.14      U     2    lan0        1500
127.0.0.0            127.0.0.1         U     0    lo0        32808
default              192.168.1.1       UG    0    lan0        1500
```

四色君：
「Destination」欄の「default」がデフォルトルートですね。

マリー先生：
そう。この場合は IP アドレスが「192.168.1.1」のルーターが、パケットのデフォルトの送り先ということになるの。

タックス君：
「Gateway」の「127.0.0.1」といったアドレスはなんでしょう？

マリー先生：
これは「ローカルループバック」と呼ばれるもので、主にネットワークのテスト用に使用される、自分自身を示すアドレスなの。通常「127.0.0.1」というアドレスが使用されるのよ。

休憩時間：SMH によるネットワークの設定

SMH（HP System Management Homepage）でも、ネットワーク・インターフェイスの設定が行えます。その場合は、「HP-UX Network Interfaces Configuration ツール」(「Tools」→「Network Interfaces Configuration」)を使用します。

図 3：SMH でネットワーク・インターフェイスの設定をする

また、「Network Information」（「Network」→「Network Information」）では、ネットワーク・インターフェイスやルーティング・テーブルの状態を確認できます。

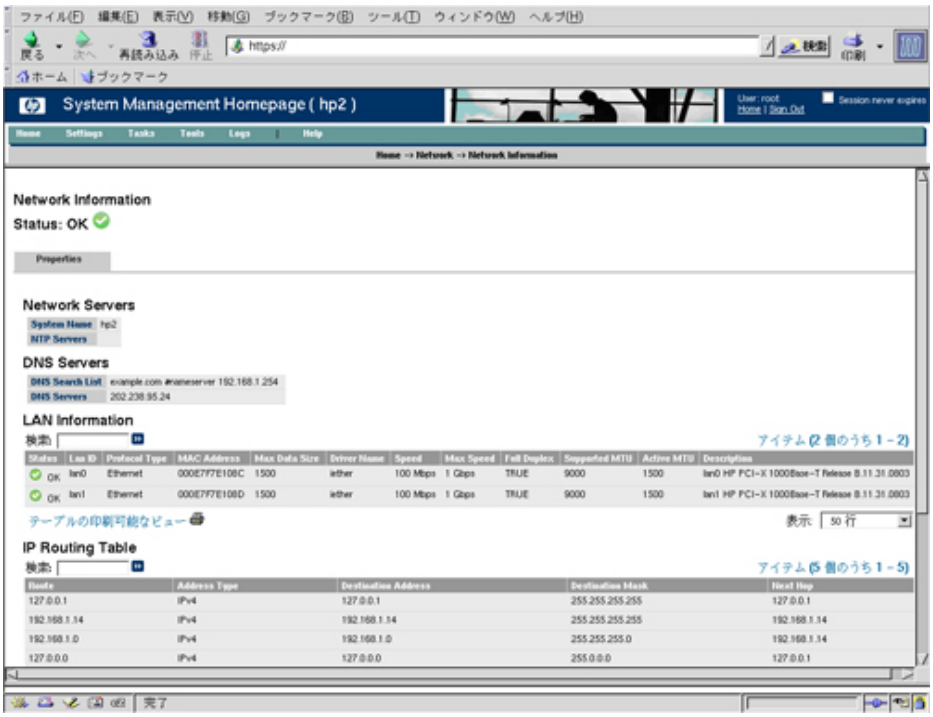


図 4：SMH で状態を確認する

2 時間目：ネットワークの基本設定

2 時間目では、まず、ホスト名と IP アドレスを変換する仕組みについて説明します。その後で、HP-UX におけるネットワークの基本設定について説明します。

IP アドレスとホスト名の変換

IP アドレスによる指定は人間にとってあまりわかりやすいものではありません。そのため、通常は「host1.example.com」や「hpux1」といったホスト名で接続先のシステムを指定します。このとき、ホスト名と IP アドレスとの変換にはさまざまな方法があります。たとえば、インターネット上のホストの場合は「DNS」(Domain Name System)と呼ばれるデータベースシステムによって集中管理されています。DNS データベースを管理するサーバーを「DNS サーバー」（ネームサーバー）と呼びます。

マリー先生：
DNS サーバーに問い合わせて、ホスト名から IP アドレスを調べるには host コマンドを使うの。こういう風にね。

```
# host www.hp.com 【Enter】
www.hp.com is an alias for www.hpgtm.nsatc.net.
www.hpgtm.nsatc.net has address 15.201.49.22
www.hpgtm.nsatc.net has address 15.200.30.22
www.hp.com is an alias for www.hpgtm.nsatc.net.
www.hp.com is an alias for www.hpgtm.nsatc.net.
```

タックス君：
逆に、IP アドレスからホスト名を調べることもできますか？

マリー先生：
できるわ。host コマンドの引数に、ホスト名の代わりに IP アドレスを指定すればいいの。

```
# host 15.201.49.22 【Enter】
22.49.201.15.in-addr.arpa domain name pointer g4u0180.houston.hp.com.
```

ホスト名から IP アドレスを調べることを「正引き」、IP アドレスからホスト名を調べることを「逆引き」というので覚えておいてね。

参照する DNS サーバーの設定

どの DNS サーバーを参照するかは/etc/resolv.conf ファイルで設定されています。次に/etc/resolv.conf の設定例を示します。

```
search example.com
nameserver 192.168.1.55 ←ネームサーバー
```

四色君：
この例では「192.168.1.55」という IP アドレスのネームサーバーを参照しろということですね。

マリー先生：
そうね。nameserver 行を複数指定して複数のネームサーバーを指定することもできるわ。その場合、上から順に問い合わせが行われるの。あと、「resolv.conf」というファイル名は「resolve.conf」と間違えやすいので注意してね。

タックス君：
「search example.com」の行は何ですか？

マリー先生：
search 行では、短いホスト名を指定したときに自動補完されるドメイン名を指定できるの。たとえば「search example.com」と設定されている場合に、Web ブラウザで「http://hp」と指定すると「http://hp.example.com」と補完されるというわけね。

/etc/hosts ファイルの利用

LAN 内のマシンの場合、DNS サーバーでホスト名と IP アドレスの対応を管理する代わりに、各マシンの「/etc/hosts」ファイルにホスト名と IP アドレスの対応を記述するという方法も一般的です。次に/etc/hosts の設定例を示します。

```
192.168.1.14    hp2.example.com    hp2
192.168.1.102   host1.example.com host1
127.0.0.1       localhost       loopback
```

マリー先生：
「/etc/hosts」には名前のほかに、別名を指定できるの。次の行を見てね。

```
192.168.1.14    hp2.example.com    hp2
      ↑                ↑               ↑
IP アドレス         ホスト名          別名
```

この例では「hp2.example.com」のホストの別名として「hp2」を指定しているの。このように名前としては正式なドメイン名を指定して、別名には短いホスト名を指定するこ

四色君：
最後の「localhost」の行は何ですか？

マリー先生：
これは 1 時間目に学んだ「netstat -nr」コマンドの説明でも出てきた、自分自身を示すループバックアドレスの設定よ。ループバックのホスト名としては、このように「localhost」が使われるの。
なお、ホスト名から IP アドレスを求めることを「名前解決」と呼ぶの。名前解決にはここまで説明してきた DNS による方法と/etc/hosts ファイルによる方法以外にも、NIS（もしくは NIS+）というサービスを使用する方法があるの。どれを優先して利用するかは/etc/nsswitch.conf の「hosts」行で設定されているわ。
次の例は、まず「files」で/etc/hosts を調べて、ホストが見つからなければ「dns」で DNS サーバーを検索するようにするための設定よ。

```
hosts:      files [NOTFOUND=continue UNAVAIL=continue] dns
```

netconf ファイルによるネットワークの設定

HP-UX におけるネットワークの基本設定ファイルは、「/etc/rc.config.d/netconf」です。このファイルでは、1 行にひとつずつ「変数」と「値」のペアで設定を記述します。

```
変数=値
```

たとえば、ホスト名は「HOSTNAME」変数で設定します。

```
HOSTNAME="hp2"
```

以下に、netconf ファイルの設定例を見ていきましょう。

ネットワーク・インターフェイスの設定

ネットワーク・インターフェイスを設定する変数は[0]、[1]…といった添字を指定します。ネットワーク・インターフェイスは複数搭載されている場合があるからです。たとえばデバイス名が「lan0」のネットワーク・インターフェイスの設定例は次のようになります。

```
INTERFACE_NAME[0]=lan0     ←設定するネットワーク・インターフェイス名
IP_ADDRESS[0]=192.168.1.14 ←IP アドレス
SUBNET_MASK[0]=255.255.255.0   ←サブネットマスク
BROADCAST_ADDRESS[0]=""    ←(1)ブロードキャストアドレス
INTERFACE_STATE[0]=""      ←(2)デフォルトのステート
DHCP_ENABLE[0]=0           ←(3)DHCP サーバーから取得するかどうか
```

マリー先生：
(2)「INTERFACE_STATE」はシステム・ブート時のインターフェイスの状態を指定するの。「UP」（動作）、「DOWN」（停止）のどちらかだけど、無指定では「UP」が指定されているものとみなされるのよ。(3)「DHCP_ENABLE」は、IP アドレスなどの情報を DCHP サーバーから取得するかどうかの指定よ。このように「0」を指定すると手動設定となるの。

四色君：
(1)の「BROADCAST_ADDRESS」とは何でしょう？

マリー先生：
ネットワーク内のすべてのホストにパケットを送るために用意されている特別の IP アドレスを「ブロードキャストアドレス」というの。たとえば「192.168.1.0/24」ネットワークのブロードキャストアドレスは通常「192.168.1.255」になるのだけど、「BROADCAST_ADDRESS」を設定することで別のアドレスを使用できるようになるのよ。

デフォルトゲートウェイの設定

1 時間目にも説明しましたが、どのネットワークに宛てたパケットがどのルーターを通るかという情報を「経路情報」と呼びます。また、ネットワーク・インターフェイスと同様に経路情報も複数あるため、変数名に[0][1][2]…といった添字を付けて設定します。通常、デフォルトゲートウェイの添字は[0]を使用します。次に設定例を示します。

```
ROUTE_DESTINATION[0]="default"  ←デフォルトゲートウェイは「default」を指定
ROUTE_GATEWAY[0]=192.168.1.1    ←IP アドレスを指定
```

タックス君：
root で vi を起動し「/etc/rc.config.d/netconf」を変更した後に「:w」コマンドを実行しても書き込みできないんですが……。

マリー先生：
「/etc/rc.config.d/netconf」は初期状態で、パーミッションで書き込みが禁止されているの。ただし、vi で「:w!」コマンドを実行すれば強制的に書き換えできるわよ。

タックス君：
「/etc/rc.config.d/netconf」で記述した IP アドレスなどの設定を反映するには、システムを再起動すればいいのですか？

マリー先生：
それでもいいけど、別の方法もあるわ。実はシステムの起動時には「/sbin/init.d」ディレクトリ以下に用意されている初期化スクリプトが順に実行されていくんだけど、「/sbin/init.d/net」がネットワークワークの設定用のスクリプトなの。このスクリプトは「/etc/rc.config.d/netconf」を読み込んでネットワークの初期化を行うのよ。次のように「stop」と「start」を引数に 2 回実行すればネットワークが再設定されるわ。

```
# /sbin/init.d/net stop  【Enter】
# /sbin/init.d/net start
```

練習問題

第 1 問：ネットワーク・インターフェイス「lan0」の IP アドレスなどの情報を表示するコマンドはどれか？

a)　ifconfig
b)　ifconfig lan0
c)　lan lan0
d)　show lan0

b)　ifconfig lan0
ifconfig はネットワーク・インターフェイスの状態を設定/表示するコマンド。

第 2 問：現在のルーティング・テーブルの状態を表示するコマンドはどれか？

a)　show route
b)　netstat
c)　status lan0
d)　netstat -nr

d)　netstat -nr
ルーティング・テーブルの状態は「netstat -nr」コマンドで表示できる。

第 3 問：ホスト「hp3.example.com」の IP アドレスを DNS サーバーに問い合わせるコマンドはどれか？

a)　host hp3.example.com
b)　netstat hp3.example.com
c)　ip hp3.example.com
d)　network hp3.example.com

a)　host hp3.example.com
host は DNS サーバーに対して正引き、逆引きを行うコマンド。

第 4 問：IP アドレスとホスト名の対応を記述するファイルはどれか？

a)　/etc/host.conf
b)　/etc/rc.config.d/netconf
c)　/sbin/init.d/net
d)　/etc/hosts

d)　/etc/hosts
/etc/hosts は IP アドレスとホスト名の対応をマシンごとに管理するファイル。

UNIX の教科書「運用編」

第 5 日目：ネットワーク関連のコマンドを学ぶ

2010 年 1 月　大津 真

ネットワークの基本は理解できたでしょうか？　今回は、実際にネットワークを利用してみましょう。1 時間目はネットワークをテストするためのコマンドについて学びます。2 時間目は telnet、ssh、ftp といったリモートログインやファイル転送のためのネットワーク・クライアントの基本的な使い方です。Windows では GUI ベースのクライアントを使っていた方も多いでしょうが、HP-UX でのコマンドラインからの使い方もしっかり押さえておきましょう。

マリー先生：
前回に引き続き、今日のお題はネットワーク。ネットワークのテスト用のコマンドや、クライアント・コマンドを紹介していきましょう。その前に、前回のネットワークの基本設定について何か質問ありますか？

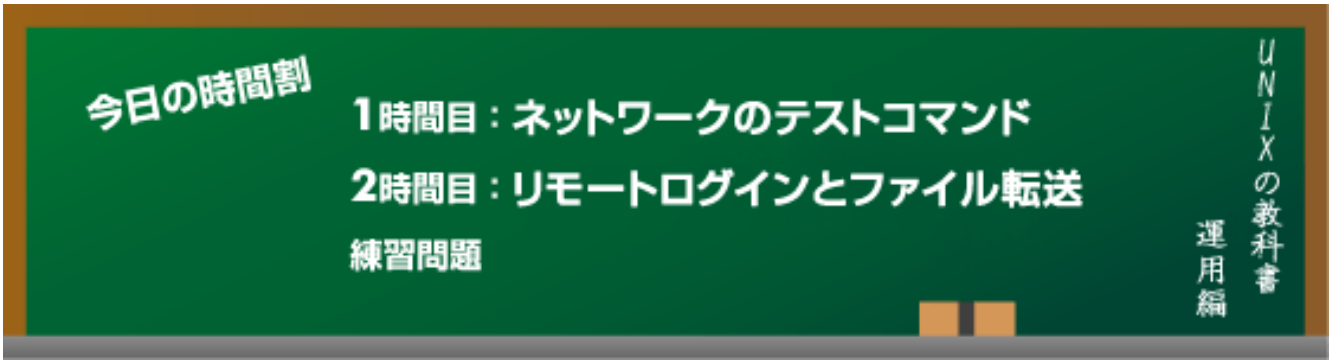
タックス君：
TCP/IP は、TCP と UDP の 2 種類に大別されるって聞いたのですが、どのように違うのでしょう？

マリー先生：
簡単に言えば、TCP はデータが正しく送られたことをチェックできるプロトコルで、UDP はデータを送りつけるだけでそれがきちんと送られたかどうかは気にしないプロトコルということかな。Web やメールなどたいていのインターネット通信には TCP が使用されるわ。ただし、TCP の場合には、エラーチェックとかが必要になるから速度的には不利になるわね。

四色君：
データが正しいことが保証できない UDP にも使い道はあるのですか？

マリー先生：
もちろん。UDP は信頼性よりも速度が優先されるマルチメディア系の通信、たとえば音声やビデオのリアルタイム配信などで活躍するの。あと TCP/IP は、実際にはさまざまなプロトコルの集合体で、次のような 5 つのレイヤーに分割されるの。TCP と UDP は、トランスポート層のプロトコルね。

5 層　アプリケーション層
4 層　トランスポート層
3 層　ネットワーク層
2 層　データリンク層
1 層　物理層
図 1：TCP/IP 階層モデル



1 時間目：ネットワークのテストコマンド

コマンドの説明の前に、イーサネットによる LAN のデータリンク層でコンピュータを識別する MAC アドレス（物理アドレス、ハードウェア ステーションアドレス）について説明しましょう。
MAC アドレスは、6 バイト（48 ビット）の長さの、LAN カードなどのハードウェアに固有のアドレスです。最初の 3 バイトは「ベンダーコード」と呼ばれ、IEEE（米国電気電子技術者協会）によって機器メーカーごとに割り当てられています。つまりベンダーコードを見ればメーカーがわかるわけです。残りの 3 バイトはメーカーごとのシリアルナンバーのようなもので、同じ製品であっても重複しないようになっています。たとえば、HP のベンダーコードは「000E7F」です。

図 2：MAC アドレス

四色君：
それぞれのコンピュータは IP アドレスで区別されますから、それだけで十分ではないのでしょうか？

マリー先生：
IP アドレスは、TCP/IP のプロトコル階層でいうところのネットワーク層の IP で管理されるアドレスです。そのため、IP アドレスではその下のデータリンク層での相手の特定ができないの。
MAC アドレスは、わかりやすくするために 1 バイトずつコロン「:」でつないで表記することも多いわ。

00:0E:7F:7E:10:8C

lanscan コマンド

各 LAN カードの MAC アドレスは lanscan コマンドを実行すると確認できます。lanscan コマンドは LAN デバイスに関する情報を表示する、HP-UX のオリジナルコマンドです。一般ユーザーが実行するには、次のように絶対パスで指定します。2 番目のフィールドに MAC アドレスが表示されます。

```
$ /usr/sbin/lanscan 【Enter】
Hardware Station        Crd Hdw   Net-Interface  NM  MAC       HP-DLPI DLPI
Path     Address        In# State NamePPA        ID  Type      Support Mjr#
Hardware Station        Crd Hdw   Net-Interface  NM  MAC       HP-DLPI DLPI
Path     Address        In# State NamePPA        ID  Type      Support Mjr#
0/1/2/0  0x000E7F7E108C 0   UP    lan0 snap0     1   ETHER     Yes     119
0/1/2/1  0x000E7F7E108D 1   UP    lan1 snap1     2   ETHER     Yes     119
～略～
```

四色君：
MAC アドレスの先頭に「0x」が付いていますが……。

マリー先生：
「0x」は続く数値が 16 進数であることを表しているの。

タックス君：
lanscan は、なんでコマンド名だけで実行できないんですか？

マリー先生：
lanscan コマンドが保存されている/usr/sbin ディレクトリは、初期状態で一般ユーザーのコマンド検索パスに含まれていないのね。スーパーユーザーのコマンド検索パスには含まれているので、「su -」コマンドでスーパーユーザーに移行すればコマンド名だけで実行できるわよ。

linkloop コマンド

linkloop コマンドは、IEEE 802.2 で規定されたリンク層のテストフレームを使用して、LAN の接続をテストする HP-UX オリジナルのコマンドです。引数には MAC アドレスを指定して、次のように実行します。

```
$ /usr/sbin/linkloop 0x000E7F7E108D 【Enter】
Link connectivity to LAN station: 0x000E7F7E108D
 -- OK
```

ping コマンド

linkloop はデータリンク層での接続をテストするコマンドですが、次に紹介する ping（ピング）コマンドはネットワーク層での接続をテストするコマンドです。ネットワークの診断コマンドとしてはもっとも一般的です。引数にはホスト名もしくは IP アドレスを指定します。デフォルトで 1 秒ごとにテスト用のパケットを送り、その応答時間を表示します（間隔は「-I 秒数」オプションで指定可能）。終了は Ctrl + C キーです。次に実行例を示します。

```
$ /usr/sbin/ping www.example.com 【Enter】
PING www.example.com: 64 byte packets
64 bytes from 192.0.32.10: icmp_seq=0. time=121. ms
64 bytes from 192.0.32.10: icmp_seq=1. time=120. ms
64 bytes from 192.0.32.10: icmp_seq=2. time=114. ms
64 bytes from 192.0.32.10: icmp_seq=3. time=117. ms
64 bytes from 192.0.32.10: icmp_seq=4. time=117. ms
64 bytes from 192.0.32.10: icmp_seq=5. time=114. ms
64 bytes from 192.0.32.10: icmp_seq=6. time=122. ms←Ctrl + C キーで終了
----www.example.com PING Statistics----
7 packets transmitted, 7 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 114/118/122
```

タックス君：
たまに ping に応答しないサイトがありますよね。

マリー先生：
残念なことに、ping は悪意のある人間が攻撃対象のサイトを調べる目的にも使用されるの。ping は「ICMP」というパケットを送るんだけど、セキュリティ上の配慮から最近ではその ICMP に応答しないように設定されているホストもあるのね。

四色君：
ping で表示される時間はなにを表しているのですか？

マリー先生：
パケットの往復時間ね。ラウンドトリップタイムと呼ぶの。これを見ると途中の経路の混み具合がわかるわけ。Ctrl + C キーを押すと、ラウンドトリップ時間の最小値（min）、平均（avg）、最大値（max）を表示して終わるわ。

サービスとポート番号

1 時間目の最後に、それぞれのネットワークサービスに割り当てられたポート番号について説明しておきましょう。TCP と UDP は、「ポート番号」と呼ばれる番号を使用して接続先のサービスを指定します。IP アドレスを住所とするなら、ポート番号はマンションの部屋番号のようなものと考えるとわかりやすいでしょう。どのサービスが基本的にどのポートを使うかを決めておくと、接続時にポート番号を気にしなくてすむため、あらかじめ特定のサービスのために予約されている番号があります。それらのポート番号をウェルノンポート(Well Known Port)と呼びます。

プロトコル	ポート番号
FTP(データ用)	20
FTP(制御用)	21
SSH	22
Telnet	23
SMTP	25
HTTP	80

表 1：ウェルノンポートの例

たとえば、Web に使用される HTTP のウェルノンポートは 80 番です。そのため Web ブラウザで URL を入力するときに、最後の「:80」を省略できるわけです。

```
http://www.example.com:80
 ↓ポート番号を省略
http://www.example.com
```

タックス君：
ウェルノンポートの一覧はどこかに保存されているのですか？

マリー先生：
基本的なサービスとポート番号の対応は/etc/services というファイルに記述されているの。

/etc/services の例（一部）

```
tcpmux      1/tcp                   # TCP port multiplexer (RFC 1078)
echo        7/tcp                   # Echo
echo        7/udp                   #
discard     9/tcp   sink null       # Discard
discard     9/udp   sink null       #
systat     11/tcp   users           # Active Users
daytime    13/tcp                   # Daytime
daytime    13/udp                   #
qotd       17/tcp   quote           # Quote of the Day
chargen    19/tcp   ttytst source   # Character Generator
chargen    19/udp   ttytst source   #
ftp-data   20/tcp                   # File Transfer Protocol (Data)
ftp        21/tcp                   # File Transfer Protocol (Control)
telnet     23/tcp                   # Virtual Terminal Protocol
```

休憩時間：より便利なシェル bash

HP-UX では、POSIX シェルが標準シェルですが、POSIX シェルには必要最小限の機能しか用意されていません。最近では、さまざまな便利な機能を加えたオープンソースの高機能シェルが普及していますので、POSIX シェルの操作性に不満を感じたら別のシェルを試してみるとよいでしょう。

現在、もっとも広く使用されているシェルのひとつが、Linux でも標準シェルとして採用されている bash です。bash は「Bourne-Again Shell」の略で、POSIX シェルのもとになった sh（Bourne Shell）を基本に拡張されていますので、乗り換えにも違和感はないでしょう。

bash には、タブキーによるパスの補完機能や、過去に実行したコマンドのインクリメンタルサーチ機能など多数の機能が用意されています。
HP-UX 用にコンパイルされた bash のバイナリパッケージは、以下のサイトからダウンロードできます。
Porting And Archive Centre for HP-UX

なお、bash 本体の他に以下のパッケージが必要になります。

- gettext
- libiconv
- termcap

2 時間目：リモートログインとファイル転送

2 時間目では、telnet、ssh、ftp といったリモートログインやファイル転送のためのネットワーク・クライアントの基本的な使い方について説明しましょう。いずれも HP-UX に標準で用意されているコマンドです。

telnet コマンドによるリモートログイン

telnet コマンドは、古くからあるリモートログインコマンドです。Telnet サーバーが動作しているホストに接続するには、次の書式で実行します。

```
telnet ホスト名もしくは IP アドレス
```

たとえば、Telnet サーバーが動作中のホスト「hp2」に接続するには次のようにします。

```
$ telnet hp2　【Enter】
Trying...
Connected to hp2.
Escape character is '^]'.
Local flow control on
Telnet TERMINAL-SPEED option ON

*******************************************************************
                        NOTICE TO USERS
～中略～
login: taro　←ユーザー名を入力
Password:　　←パスワードを入力
～中略～
$ ←ログインプロンプトが表示される
```

接続を解除するには exit コマンドを実行します。

マリー先生：
ネットワークのプロトコルはテキストベース、つまりコマンドのやりとりを ASCII 文字列で行うものが少なくないの。telnet コマンドはそれらのテキストベースのプロトコルを確認する目的でもしばしば使われるのよ。

四色君：
たとえばどんなプロトコルですか？

マリー先生：
例を挙げれば、Web やメールなどはコマンドや内容の一部はテキスト、つまり文字データとして送信されるわ。そのためには、次のようにポート番号を指定して telnet コマンドを実行するの。

```
telnet ホスト名もしくは IP アドレス ポート番号
```

タックス君：
Telnet のウェルノンポートは 23 ですから、ポート番号を省略すると 23 番ポートに接続されるわけですね。

マリー先生：
そういうこと。それじゃあ、一例として Web サーバーが動作中のホスト「host1」に接続して HTML ファイル「index.html」を取得する例を示すわね。

```
$ telnet host1 80   ←host1 の 80 番ポートに接続  【Enter】
Trying...
Connected to host1.example.com.
Escape character is '^]'.
GET /index.html HTTP/1.1   ←GET コマンドで index.html を取得
host:host1.example.com      ←ホストを指定
          ←空行
HTTP/1.1 200 OK
Date: Fri, 08 Jan 2010 13:43:01 GMT
Server:
Apache/2.2.13 (Unix) mod_ssl/2.2.13 OpenSSL/0.9.8k DAV/2 PHP/5.3.0
Content-Location: index.html.en
Vary: negotiate
TCN: choice
Last-Modified: Sat, 20 Nov 2004 20:16:24 GMT
ETag: "d67240d-2c-3e9564c23b600;4748fc723cc40"
Accept-Ranges: bytes
Content-Length: 44
Content-Type: text/html
Content-Language: en

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">   ↑メッセージボディ
<html><head>
<title>Sample</title>
<meta http-equiv="Content-type" content="text/html; charset=UTF-8">
</head>
<body bgcolor="#FFFFFF">
<h1>Hello World</h1>
</body>
</html>

Connection closed by foreign host
```

SSH による安全なリモートログイン

Telnet プロトコルは、アカウント情報や通信の内容がすべて平文（暗号化されていないテキスト）で送信されますので、容易に盗聴されてしまいます。そのため、特にインターネット上に公開されているサーバーでは Telnet によるリモートログインを禁止している場合がほとんどです。その代わりに普及しているのが、ログイン情報を含めて通信の内容をすべて暗号化してモートログインが可能な SSH（セキュアシェル）です。

タックス君：
SSH には商用のものとオープンソースのものがあると聞いたのですが。

マリー先生：
そうね。HP-UX に標準付属しているのはオープンソースの SSH である OpenSSH よ。ssh コマンドを「-V」オプションを付けて実行すると、バージョンを確認できるわ。

```
$ ssh -V　【Enter】

OpenSSH_4.7p1+sftpfilecontrol-v1.2-hpn12v17,
OpenSSL 0.9.7m 23 Feb 2007
HP-UX Secure Shell-A.04.70.023, HP-UX Secure Shell version
```

四色君：
リモートログインは SSH に限定することにした場合、Telnet サーバーを停止することはできるのですか？

マリー先生：
できるわよ。まずスーパーユーザーで/etc/inetd.conf を編集して telnet の行をコメントにするの。

```
telnet     stream tcp6 nowait root /usr/lbin/telnetd  telnetd
   ↓先頭に「#」を記述してコメントにする
#telnet     stream tcp6 nowait root /usr/lbin/telnetd  telnetd
```

次に、inetd コマンドを「-c」オプションを指定して実行するの。

```
# /usr/sbin/inetd -c
```

以上で、設定が反映され、telnet によるログインができなくなるわ。

```
$ telnet hp2　←接続しようとすると…　【Enter】
Trying...
telnet: Unable to connect to remote host: Connection refused
   ↑接続が拒否される
```

SSH サーバーが実行中のリモートホストにログインするには次のように slogin コマンドを実行します（slogin コマンドの代わりに ssh コマンドを使用してもかまいません）。

```
slogin -l ユーザー名 ホスト名もしくは IP アドレス
```

次にユーザー「taro」としてホスト「host1」に接続する例を示します。

```
$ slogin -l taro host1　【Enter】
The authenticity of host 'host1 (192.168.1.102)' can't be established.
RSA key fingerprint is 12:70:08:25:2f:3e:ac:b1:7d:4b:2a:f4:b3:fc:bd:dd.
Are you sure you want to continue connecting (yes/no)? yes　←(1)「yes」を入力
Warning: Permanently added 'host1,192.168.1.102' (RSA) to the list of known hosts.
Password:　←(2)パスワードを入力
Last login: Fri Jan  8 17:30:42 2010
$　←ログイン完了
```

SSH では、ユーザー認証の前に「ホスト認証」が行われます。初回接続時には、ホストの公開鍵と呼ばれる認証に必要な鍵が保存するかをたずねてきます（(1)の部分）。「yes」を入力すると、そのホストの公開鍵が「~/.ssh/known_hosts」に登録されます。この操作は 2 回目以降は不要です。
(2)でパスワードを入力するとリモートログインが完了します。

四色君：
ホスト認証ってなんのために行われるのですか？

マリー先生：
ホストが改ざんされていないことを確認するための認証よ。たとえば、別のホストがそのホストになりすましているような場合には、ホスト認証の段階でエラーになって、ホストが正しくないことがわかるわけ。
あと、SSH では通常のアカウントによるログインだけではなく、より安全な「公開鍵暗号方式」によるログインも可能なの。

タックス君：
公開鍵暗号方式ってなんですか？

マリー先生：
秘密鍵と公開鍵という鍵のペアを使ってログインする方式よ。ホスト認証はこの公開鍵暗号方式で行われるけど、ユーザー認証もその方式で行うことができるの。

四色君：
普通のユーザー名とパスワードによるログインと比べて、安全なのですか？

マリー先生：
そう、公開鍵暗号方式を使うとパスワードをネットワーク上に送る必要がなくなるの。また、公開鍵暗号方式ではパスワードに相当するパスフレーズというものを使うのだけど、万が一パスフレーズが外部に漏れたとしても、秘密鍵が漏洩しなければログインできないの。それについては、また別の機会に説明するわね。

FTP によるファイル転送

FTP はファイル転送用の古典的なプロトコルです。ここでは、もっともシンプルな FTP クライアントである ftp コマンドの使い方を説明しましょう。

```
ftp <FTP サーバーのホスト名または IP アドレス>
```

ftp コマンドを実行し、正しいユーザー名とパスワードを入力すると、プロンプト「ftp>」が表示されます。次に FTP サーバー「host1」にユーザー名「taro」で接続する例を示します。

```
$ ftp host1　【Enter】
Connected to host1.example.com.
220 192.168.1.102 FTP server (tnftpd 20080929) ready.
Name (host1:o2): taro　←ユーザー名を入力
331 User taro accepted, provide password.
Password:　←パスワードを入力
230 User taro logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>　←プロンプトが表示されコマンド待ちの状態になる
```

四色君：
僕は Windows 用のソフトを FTP サーバーからダウンロードすることがよくあるんですが、そのときにはユーザー名とパワードは不要ですよね。

マリー先生：
FTP サーバーには、アカウントが必要なタイプと、Anonymous FTP サーバー（匿名 FTP サーバー）と呼ばれる誰でもゲストとして接続できるものがあるの。

タックス君：
でも、ftp コマンドでは、Anonymous FTP サーバーに接続する場合でもユーザー名とパスワードを要求されますよね。

マリー先生：
そう、ftp コマンドで Anonymous FTP サーバーにログインするには、通常はユーザー名に「anonymous」、パスワードに電子メールアドレスを入力するの。ただ、パスワードにメールアドレスを入力するのはあくまで儀礼的なもので、実際にはどんな文字列を入力してもログインできるわ。

「ftp>」プロンプトに続いて、サブコマンドを入力することによりファイルのダウンロードやディレクトリの移動などの操作が行えます。たとえば、ダウンロードには get コマンドを使用します。
次に、「110.jpg」ファイルをダウンロードする例を示します。

```
ftp> get 110.jpg
200 PORT command successful.
150 Opening BINARY mode data connection for '110.jpg' (49351 bytes).
226 Transfer complete.
49351 bytes received in 0.02 seconds (2863.26 Kbytes/s)
```

ftp コマンドを終了するには quit コマンドを実行します。

```
ftp> quit
221-
   Data traffic for this session was 49351 bytes in 1 file.
   Total traffic for this session was 56541 bytes in 2 transfers.
221 Thank you for using the FTP service on 192.168.1.102
```

次の表に、ftp コマンド内で使用できる主なサブコマンドをまとめておきます。

コマンド	説明
ascii	アスキー転送モードにする（改行コードを OS によって変更する）
bin	バイナリ転送モードにする
ls <ディレクトリ>	リモート側のファイルの一覧を表示する
put <ファイル>	ファイルをアップロードする
mput <ファイル>	複数のファイルをアップロードする（ワイルドカード使用可）
get <ファイル>	ファイルをダウンロードする
mget <ファイル>	複数のファイルをダウンロードする（ワイルドカード使用可）
prompt	インタラクティブモードのオン・オフを切り替える（デフォルトでオン）
cd <ディレクトリ>	リモート側のディレクトリを変更する
lcd <ディレクトリ>	ローカル側のディレクトリを変更する
delete <ファイル>	リモート側のファイルを削除する
mkdir <ディレクトリ>	リモート側にディレクトリを作成する
help	コマンドの一覧を表示する
quit	セッションを終了する

表 2 : ftp 内で使用できるサブコマンド

マリー先生：
FTP も Telnet と同じようにアカウント情報が平文で流れるので、あまり安全なプロトコルではないの。

タックス君：
それじゃあ、より安全なファイル転送用プロトコルにはなにがあるんですか？

マリー先生：
最近では FTP の SSH 版といえる sftp が普及してきているわね。次の形式で使うの。

sftp ユーザー名@ホスト名もしくは IP アドレス

ログインしたら、利用可能なサブコマンドの一覧については help コマンドで確認してね。

練習問題

第 1 問：ホスト「www.example.com」が ICMP パケットに応答するかどうかを調べるコマンドはどれか？

a)　test www.example.com
b)　netstat www.example.com
c)　linkloop www.example.com
d)　ping www.example.com

d)　ping www.example.com
ping は ICMP パケットを送信してその応答を調べるコマンド。

第 2 問：基本的なサービスとポート番号の対応とが記述されているファイルはどれか？

a)　/service
b)　/etc/services
c)　/etc/hosts
d)　/etc/protocols

b)　/etc/services
/etc/services にはウェルノンポートの一覧が格納されている。

第 3 問：SSH を使用して、ホスト「hp2」にユーザー「taro」としてリモートログインするコマンドはどれか？

a)　ssh -l taro hp2
b)　ssh taro hp2
c)　slogin hp2 taro
d)　ssh hp2 taro

a)　ssh -l taro hp2
SSH サーバーにログインするには、ssh もしくは slogin コマンドを使用する。

第 4 問：FTP サーバーに ftp コマンドで接続している状態で、ファイル「sample.jpg」をダウンロードするコマンドはどれか？

a)　put sample.jpg
b)　download sample.jpg
c)　sample.jpg
d)　get sample.jpg

d)　get sample.jpg
ファイルのダウンロードは「get ファイル名」を使用する。

UNIX の教科書「運用編」

第 6 日目：システム情報の表示コマンドを学ぶ

2010 年 2 月　大津 真

6 回にわたってお送りしてきた運用編もいよいよ最終回となりました。今回は、システムのさまざまな情報を表示するコマンドを学びます。1 時間目は OS やハードウェア情報を表示するコマンド、2 時間目はログイン情報やソフトウェア情報を表示するコマンドです。いずれもシステムの状況を把握しなければならないシステム管理者にとっては必須のコマンド群ですので、しっかり覚えて使いなせるようになってください。

マリー先生：
今日は、システム管理者の必須コマンドとして、システムのさまざまな情報を表示するコマンドを紹介していきましょう。その前に、前回のネットワーク関連のコマンドについて何か質問ありますか？

タックス君：
ftp コマンドで FTP サーバに接続した状態でファイルをダウンロードするときに、たとえば拡張子が「.txt」のファイルをまとめてダウンロードするといったことはできますか？

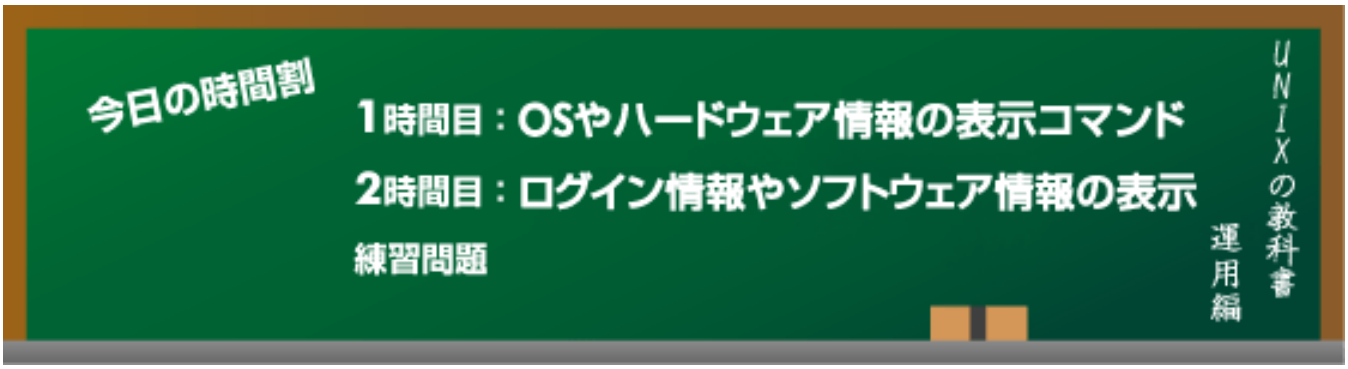
マリー先生：
get コマンドの代わりに mget コマンドを使えば、「*」や「?」といったワイルドーカードが使えるわよ。ただし、デフォルトではそれぞれのファイルをダウンロードするときに確認を求めてくるの。

```
ftp> mget *.txt　【Enter】
 ←拡張子が「.txt」のファイルをダウンロード
mget sample.txt? y ←「y」でダウンロードする
200 PORT command successful.
150 Opening BINARY mode data connection for 'sample.txt' (100 bytes).
226 Transfer complete.
100 bytes received in 0.03 seconds (3.67 Kbytes/s)
mget sample2.txt? n ←「n」でダウンロードしない
～略～
```

四色君：
そのつど確認することなしにダウンロードすることはできないんですか？

マリー先生：
確認するのはインタラクティブモードというモードなの。デフォルトではインタラクティブモードだけど、prompt コマンドを実行するごとに、インタラクティブモードのオン/オフを切り替えられるわよ。

```
ftp> prompt　【Enter】
Interactive mode off. ←インタラクティブモードがオフになる
ftp> mget *.txt　【Enter】
 ←確認なしで拡張子「.txt」のファイルをダウンロード
local: sample.txt remote: sample.txt
200 PORT command successful.
150 Opening BINARY mode data connection for 'sample.txt' (100 bytes).
226 Transfer complete.
100 bytes received in 0.00 seconds (2381.86 Kbytes/s)
local: sample2.txt remote: sample2.txt
200 PORT command successful.
150 Opening BINARY mode data connection for 'sample2.txt' (12 bytes).
226 Transfer complete.
12 bytes received in 0.02 seconds (0.53 Kbytes/s)
～略～
```



1 時間目：OS やハードウェア情報の表示コマンド

1 時間目では、OS 情報や、ハードウェアの情報、プロセスの実行状況、ディスクの使用状況などを表示するコマンドを説明しましょう。

uname コマンド

uname コマンドは、現在のシステムに関するオペレーティングシステムの名前やリリースレベル、マシン番号などを表示します。引数なしで実行した場合には OS 名である「HP-UX」を表示します。

```
$ uname 【Enter】
HP-UX
```

リリースレベルを表示するには「-r」オプションを指定して実行します。

```
$ uname -r 【Enter】
B.11.31
```

「-a」オプションを指定して実行すると、OS 名、ホスト名、リリースレベル、バージョンレベル、ハードウェア名、マシン識別番号、ライセンスレベルを順に表示します。これは「-s -n -r -v -m -i -l 」を指定したのと同じです。

```
$ uname -a 【Enter】
HP-UX hp2 B.11.31 U ia64 3862204912 unlimited-user license
```

タックス君：
ホスト名だけを表示することはできないのですか？

マリー先生：
「-n」オプションを指定すればできるわ。

```
$ uname -n　【Enter】
hp2
```

あと、hostname コマンドを引数なしで実行してもホスト名を表示できるわね。

```
$ hostname　【Enter】
hp2
```

machinfo コマンド

CPU や搭載メモリ、ファームウェアの情報といったマシン情報を表示するには machinfo コマンドを実行します。

```
$ machinfo　【Enter】
CPU info:      ←CPU 情報
 2 Intel(R) Itanium 2 processors (1.6 GHz, 6 MB)
       400 MT/s bus, CPU version A1

Memory: 8184 MB (7.99 GB)      ←メモリ情報

Firmware info:      ←ファームウェア情報
  Firmware revision:  03.10
  FP SWA driver revision: 1.18
  IPMI is supported on this system.
  BMC firmware revision: 3.47

Platform info:      ←プラットフォーム情報
  Model:              "ia64 hp server rx2620"
  Machine ID number:     e63491f0-9ed6-11d9-af6f-0adab462a823
  Machine serial number:  JPA0508027

OS info:      ←OS 情報
  Nodename:  hp2
  Release:   HP-UX B.11.31
  Version:   U (unlimited-user license)
```

```
  Machine:  ia64
  ID Number: 3862204912
  vmunix _release_version:
@(#) $Revision: vmunix:    B.11.31_LR FLAVOR=perf
```

bdf コマンド

bdf コマンドは、ファイルシステム全体のサイズ、使用領域、利用可能領域などの情報を表示します。

```
$ bdf 【Enter】
Filesystem          kbytes    used    avail %used Mounted on
/dev/vg00/lvol3    4194304  611144  3555224   15% /
/dev/vg00/lvol1    1835008  308448  1514720   17% /stand
/dev/vg00/lvol8   17416192 1494088 15797744    9% /var
/dev/vg00/lvol7    8388608 2914888  5431064   35% /usr
/dev/vg00/lvol6    4194304  321176  3842912    8% /tmp
/dev/vg00/lvol5    8388608 6624224  1750624   79% /opt
/dev/vg00/lvol4    9551872   19427  8936675    0% /export/home
DevFS                    3       3        0  100% /dev/deviceFileSystem
```

ファイルシステム名　全体のサイズ　使用領域　利用可能領域　使用率　マウント先

四色君：
ファイルシステムが「/dev/vg00/lvol3」のように表示されていますが……。

マリー先生：
この例では、複数台のディスクを 1 つのグループとして管理する LVM（Logical Volume Manager）が使用されているの。「vg00」はボリュームグループを「lvol3」は論理ボリュームを表しているのよ。

四色君：
最後の「DevFS」というファイルシステムはなんでしょう。

マリー先生：
これは「Device File System」と呼ばれる仮想的なファイルシステムで、マウスやキーボード、DVD ドライブなどの USB デバイスの管理に使用されるの。

タックス君：
bdf コマンドの代わりに df コマンドでも未使用領域などの情報が表示できますよね？

マリー先生：
そうね、df コマンドを「-b」オプションを指定して実行すると、ファイルシステムごとの未使用領域が K バイト単位で表示されるわ。

```
$ df -b　【Enter】
/dev/deviceFileSystem  (DevFS              ) :        0 Kbytes free
/export/home           (/dev/vg00/lvol4    ) :  9532445 Kbytes free
/opt                   (/dev/vg00/lvol5    ) :  1764384 Kbytes free
/tmp                   (/dev/vg00/lvol6    ) :  3873128 Kbytes free
/usr                   (/dev/vg00/lvol7    ) :  5473720 Kbytes free
/var                   (/dev/vg00/lvol8    ) : 15922080 Kbytes free
/stand                 (/dev/vg00/lvol1    ) :  1526560 Kbytes free
/                      (/dev/vg00/lvol3    ) :  3583160 Kbytes free
```

du コマンド

du コマンドは引数で指定したディレクトリを再帰的にたどって、各ディレクトリの容量、および総容量を表示します。デフォルトの単位はブロック数（512 バイト）ですが、「-k」オプションを指定することにより K バイト単位で表示します。たとえば Documents ディレクトリ以下の容量を、K バイト表示するには次のようにします。

```
$ du -k Documents　【Enter】
24      Documents/2007
48      Documents/2008
23416   Documents/sample
8       Documents/2009/info/samples
9928    Documents/2009/info
9928    Documents/2009
24      Documents/2010
33488   Documents
```

←各ディレクトリごとの容量

←総容量

タックス君：
総容量だけを表示することはできますか？

マリー先生：
それには「-s」オプションを指定すればいいわ。

```
$ du -sk Documents　【Enter】
33488   Documents
```

四色君：
各ファイルのサイズを表示することはできますか？

マリー先生：
「-a」オプションを指定すれば OK よ。ただし、この場合も各ディレクトリの容量も表示されちゃうけどね。

```
$ du -ak Documents　【Enter】
8     Documents/2007/ok.txt        ←ファイルの容量
8     Documents/2007/ng.txt        ←ファイルの容量
8     Documents/2007/readme.txt    ←ファイルの容量
24    Documents/2007               ←ディレクトリの容量
8     Documents/2008/sep.txt       ←ファイルの容量
8     Documents/2008/magic.txt     ←ファイルの容量
～中略～
33488  Documents                   ←総容量
```

top コマンド

top コマンドは、CPU 負荷や、メモリの使用状況、プロセスの一覧をリアルタイムで表示するコマンドです。プロセスリストには、CPU の使用率の高い順にプロセスの一覧が表示されます。デフォルトでは表示は 5 秒ごとに更新されます。終了するには q キーを押します。

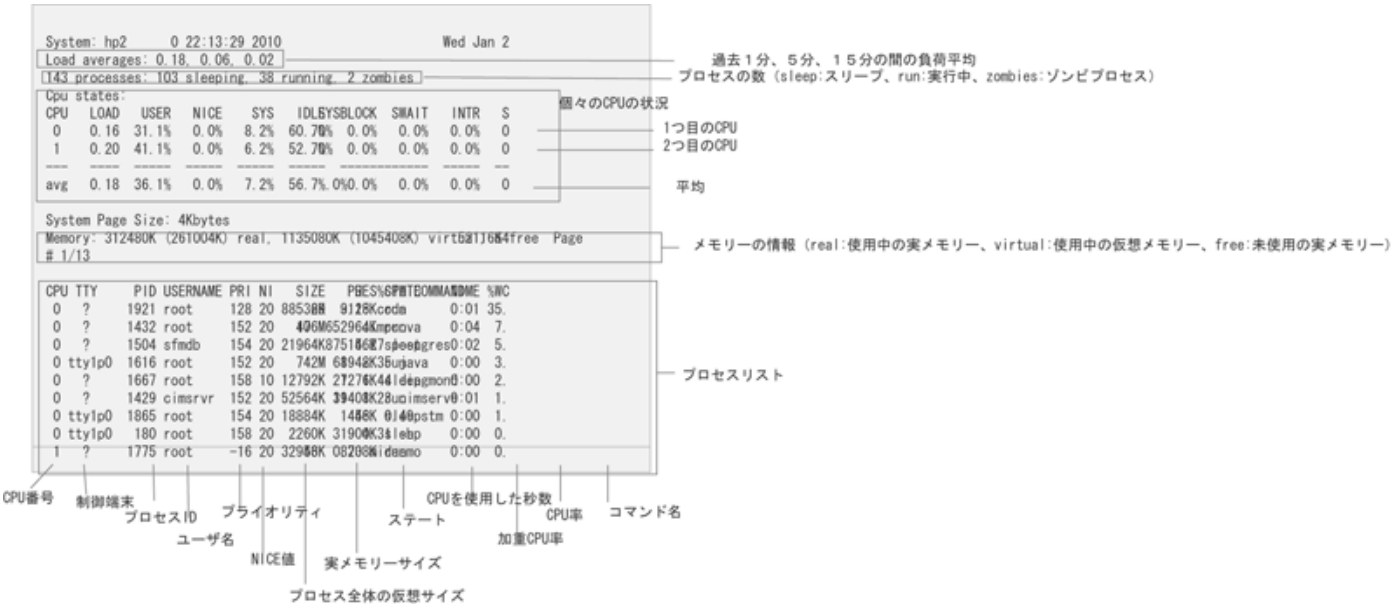


図 1：top コマンドの実行例

タックス君：
メモリ使用状況のところ、括弧で囲まれた数値はなんですか？

マリー先生：
括弧の中には、現在アクティブな状態のプロセスが使用しているメモリ容量が表示されるの。

休憩時間：HP-UX Internet Express

HP-UX では、標準で Apache（Web サーバ）や、OpenSSH といったメジャーなオープンソース・ソフトウェアが用意されていますが、それ以外にも、さまざまなオープンソース・ソフトウェアが利用可能です。オープンソース・ソフトウェアは自分でソースからコンパイルすることでインストールすることも可能ですが、HP では HP-UX 上で検証した人気の高いソフトウェアをバイナリー・パッケージ化したコレクションを「HP-UX Internet Express」として提供しています。

HP-UX Internet Express は、DVD メディアとして提供されるほか、インターネットからダウンロードすることも可能です。詳しくは以下のサイトを参照してください。

HP-UX Internet Express

2 時間目：ログイン情報やソフトウェア情報の表示

2 時間目では、ログインしているユーザーの情報や、インストールされているソフトウェアの情報などを表示するコマンドを紹介していきましょう。

who コマンド

who コマンドは、現在システムにログイン中のユーザーの一覧を表示します。

```
$ who 【Enter】
o2          console      Jan 20 14:10
yama        pts/0        Jan 20 22:13
john        pts/1        Jan 20 23:24
taro        pts/2        Jan 20 23:25
```

四色君：
2 番目の欄の「pts/0」というのはなんですか？

マリー先生：
2 番目の欄には端末の名前が表示されるの。「pts」は「pseudo-terminal slave」の略、つまり仮想端末のことね。 あと、最後の欄にはログイン時刻が表示されるわ。
それから、自分自身のユーザー名は whoami コマンドで表示できるわよ。

```
$ whoami 【Enter】
taro
```

last コマンド

last コマンドは、過去にログインしたユーザーの情報を表示します。

```
$ last 【Enter】
taro     pts/2        Wed Jan 20 23:25   still logged in
o2       pts/1        Wed Jan 20 23:24   still logged in
o2       pts/0        Wed Jan 20 22:13   still logged in
reboot   system boot  Wed Jan 20 22:11   still logged in
～中略～
root     pts/ta       Thu May  8 10:33 - 10:35  (00:01)
reboot   system boot  Thu May  8 10:10 - 10:38  (00:27)
root     console      Wed May  7 17:45 - 17:47  (00:01)
reboot   system boot  Wed May  7 17:38 - 10:10  (16:31)

WTMPS_FILE  begins at Wed May  7 17:38:45
```

四色君：
指定したユーザーだけのログイン情報を表示することはできないのですか？

マリー先生：
それには、ユーザー名を引数に実行すれば OK よ。

```
$ last o2 【Enter】
o2      pts/1       Wed Jan 20 23:24   still logged in
～中略～
o2      pts/0       Wed May 21 23:37 - 23:40  (00:03)

WTMPS_FILE  begins at Wed May  7 17:38:45
```

タックス君：
一覧に reboot っていう名前のユーザーがありますが……。

マリー先生：
reboot っていうのは擬似的なユーザーで、システムをリブートするごとに reboot が記録されるの。したがって、「last reboot」は、システムリブートの間隔を調べるのに有効ね。

lastb コマンド

lastb コマンドは、過去にログインしようとして失敗したユーザーの一覧を表示します。

```
$ lastb 【Enter】
taro    ssh:notty    Wed Jan 20 23:46
taro    ssh:notty    Wed Jan 20 23:45
o2      ssh:notty    Fri Jan  8 22:46
～略～

BTMPS_FILE  begins at Mon May 12 14:56:36
```

四色君：
lastb コマンドの情報は、なにか役立つのですか？

マリー先生：
悪意のある第三者が不正なログインを試みていないかといったことを調べるのに使うことが多いわ。なお、last コマンドと lastb コマンドは、それぞれ「/var/adm/wtmps」と「/var/adm/btmps」というログファイルの情報をわかりやすく表示するコマンドなの。

uptime コマンド

uptime コマンドはシステムの稼働時間などの情報を表示します。

```
$ uptime 【Enter】
  3:33pm  up  1:30,  5 users,  load average: 0.01, 0.00, 0.00
     |        |       |                        |
現在時刻  稼働時間  ログイン中のユーザー数  負荷平均
```

四色君：
「load average」に 3 つの数値が表示されますが……。

マリー先生：
順に、最新の 1、5、および 15 分間の負荷平均ね。あと、「-w」オプションでは、ログイン中のユーザーの状況が一覧表示されるの。

```
$ uptime -w 【Enter】
  3:34pm  up  1:30,  5 users,  load average: 0.00, 0.00, 0.00
User     tty           login@  idle   JCPU   PCPU  what
o2       console       2:10pm  1:28                /usr/sbin/getty
                                                    console console
o2       pts/0         2:10pm  1:23                /sbin/sh
taro     pts/1         2:10pm  1:23                /sbin/sh
john     pts/2         2:10pm  1:23                /sbin/sh
o2       pts/3         2:35pm                      uptime -w
```

「uptime -w」の代わりに w コマンドを引数なしで実行しても同じよ。

swlist コマンド

swlist コマンドは、現在システムにインストールされているソフトウェアの一覧を表示するコマンドです。

```
$ swlist 【Enter】
# Initializing...
# Contacting target "hp2"...
#
# Target:  hp2:/
#

#
# Bundle(s):
#

 AccessControl                 B.11.31.04     HP-UX Role-Based Access C
ontrol Infrastructure
 AppDiscCMS                    3.1.00.01      Application Discovery Cen
tral Management Server
 AtomicLib                    B.11.31.0803.01 Library for Atomic APIs.

 B2491BA                      B.11.31       MirrorDisk/UX (Server)
 B3701AA                      C.04.70.000    HP GlancePlus/UX Pak for
11.31
 B3835DA                      C.03.03.01     HP Process Resource Manag
er
 B3929EA                      B.11.31       HP OnLineJFS (Server)
 B5140BA                      A.11.31.02     Serviceguard NFS Toolkit
～略～
```

練習問題

第 1 問：Pictures ディレクトリの総容量のみを K バイト単位で表示するコマンドはどれか？

a)　bdf -sk Pictures
b)　dsize -s Pictures
c)　du Pictures
d)　du -sk Pictures

d)　du -sk Pictures

「-s」は総容量のみを表示するオプション、「-k」は K バイト単位で表示するオプション。

第 2 問：CPU 負荷やメモリ状況、CPU 使用率の高いプロセスの一覧を表示するコマンドはどれか？

a)　cpuinfo
b)　top
c)　head
d)　plist

b)　top
top コマンドはデフォルトでは 5 秒ごとに表示内容を更新する。

第 3 問：システムの稼働時間、負荷平均、およびログイン中のユーザーの一覧といった情報を表示するコマンドはどれか？

a)　uptime
b)　uptime -w
c)　top
d)　who

b)　uptime -w
「uptime -w」（もしくは w）はシステムの稼働時間やログインユーザーの一覧などを表示する。

第 4 問：システムにインストールされているソフトウェアの一覧を表示するコマンドはどれか？

a)　swlist
b)　swinfo
c)　sw list
d)　showsw

a)　swlist
swlist コマンドには CUI モード、TUI モード、GUI モードの 3 種類があります。DISPLAY 環境変数が未設定の場合、CUI モードで起動します。また、「-i」オプションを指定して実行すると、TUI または GUI モードで起動します。TUI または GUI のどちらになるかは、適切な DISPLAY 環境変数の設定次第です。不適切な値が設定されていた場合、または X サーバーの設定に不備があった場合、swlist は適切な対処方法を指示するとともに、TUI モードで起動するかどうかの確認を求めてきます。なお、GUI モードの使用には別途 X サーバーの起動が必要です。

UNIX の教科書「運用編」

期末試験

2010 年 3 月　日本ヒューレット・パッカード株式会社

ユーザーからネットワークやシステムなど、管理者への第一歩を解説してきました「UNIX の教科書 運用編」(全 6 回)、いかがでしたでしょうか？内容はご理解いただけたでしょうか。今回は、「期末試験」として全 10 問の試験を用意しました。いずれもこれまでの連載からの出題ですので、ぜひ挑戦してみてください。

マリー先生：
これまでの授業を理解できたかのおさらいを兼ね、今日は期末試験を行います。10 問全て今まで学習してきたことなので、満点を目指してくださいね。

第 1 問：環境も含めてスーパーユーザーに移行するコマンドはどれか？

a)　super
b)　root
c)　sudo
d)　su -

d)　su -
su コマンドを「-」オプション付きで実行すると環境を含めてスーパーユーザーに移行する。

第 2 問：ユーザー「hanako」を作成するコマンドはどれか？

a)　passwd hanako
b)　useradd hanako
c)　adduser hanako
d)　create user hanako

b)　useradd hanako
新規ユーザーを登録するには useradd コマンドを実行する。

第 3 問：ファイル「sample.txt」に対して、所有グループへの書き換えを許可する権限を追加するコマンドはどれか？ ただし、その他のパーミッションは変更しないものとする。

a)　chmod g+w sample.txt
b)　chperm g+w sample.txt
c)　chmod g=w sample.txt
d)　chperm g=w sample.txt

a)　chmod g+w sample.txt
オペレーターに「+」を使用することに注意。「=」を使うと、対象となるユーザーに設定されているその他のパーミッションがクリアされてしまう。

第 4 問：samples ディレクトリ以下の、すべてのファイルとディレクトリに対して、所有者および所有グループへの書き換え権限を追加するコマンドはどれか？

a)　chmod ug+w samples
b)　chmod -R ug+w samples
c)　chmod -r ug+w samples
d)　chmod +w ug samples

b)　chmod -R ug+w samples
指定したディレクトリ以下のパーミッションを再帰的に変更するには、chmod コマンドを「-R」オプションを指定して実行する。

第 5 問：testdir ディレクトリに setgid ビットをセットするコマンドはどれか？

a)　chmod g+s testdir
b)　chmod -s testdir
c)　chmod +s testdir
d)　chmod u+t testdir

a)　chmod g+s testdir
setgid ビットを設定するには「chmod g+s ディレクトリ」コマンドを実行する。

第 6 問：現在の umask 値が「022」の場合に新規ファイルを作成したときのパーミッションはどれか？

a)　655（rw-r-xr-x）
b)　622（rw--w--w-）
c)　644（rw-r--r--）
d)　755（rwxr-xr-x）

c)　644（rw-r--r--）
新規ファイルのパーミッションは「666」を umask 値でマスクした値となる。

第 7 問：ネットワーク・インターフェイス「lan0」の IP アドレスなどの情報を表示するコマンドはどれか？

a)　netconfig lan0
b)　ifconfig lan0
c)　netstat lan0

d)　nwmgr lan0

b)　ifconfig lan0
ifconfig はネットワーク・インターフェイスの状態を設定/表示するコマンド。

第 8 問：現在のルーティング・テーブルの状態を表示するコマンドはどれか？

a)　show route
b)　netstat
c)　route lan0
d)　netstat -nr

d)　netstat -nr
ルーティング・テーブルの状態は「netstat -nr」コマンドで表示できる。

第 9 問：FTP サーバーに ftp コマンドで接続している状態で、ファイル「sample.jpg」をダウンロードするコマンドはどれか？

a)　copy sample.jpg
b)　download sample.jpg
c)　send sample.jpg
d)　get sample.jpg

d)　get sample.jpg
ファイルのダウンロードは「get ファイル名」を使用する。

第 10 問：Pictures ディレクトリの総容量のみを K バイト単位で表示するコマンドはどれか？

a)　ls -ld Pictures
b)　dsize -s Pictures
c)　du Pictures
d)　du -sk Pictures

d)　du -sk Pictures
「-s」は総容量のみを表示するオプション、「-k」は K バイト単位で表示するオプション。

HP-UX
www.hpe.com/jp/hpux

© Copyright 2018 Hewlett Packard Enterprise Development LP.

本書の内容は、将来予告なく変更されることがあります。日本ヒューレット・パッカード製品およびサービスに対する保証については、当該製品およびサービスの保証規定書に記載されています。本書のいかなる内容も、新たな保証を追加するものではありません。日本ヒューレット・パッカードは、本書中の技術的あるいは校正上の誤り、脱字に対して、責任を負いかねますのでご了承ください。

Hewlett Packard Enterprise