UNIX(Linux) 入門

農林水産研究情報総合センター

Ver. 2022/06/01

目次

1. 予備知識

1.1. UNIXとは

UNIX は、アメリカ AT&T 社のベル研究所で開発された、マルチタスク、マルチユーザ機能を有する OS (オペレーティングシステム) の一種です。 また、後にカリフォルニア大学バークレー校で実装された UNIX (BSD) などと共に、これらから派生した OS の総称です。

1.2. Linuxとは

1991年にリーナス・トーバルズ氏が Linux カーネルを開発しました。 この Linux カーネルは POSIX に準拠し、 GPL ライセンスが採用されたことで、フリーの UNIX クローン (UNIX互換OS) として利用されるようになりました。

- kernel: OSの基本機能を実装したソフトウェア。
- POSIX(Portable Operations System Interface:ポジックス): は、 UNIX など各種 OS に 共通のアプリケーションインターフェースを規定し、 移植性の高いアプリケーション開発を容易にする目的で IEEE が策定した規格。
- GPL(GNU General Public License): GPLは、概ねプログラムの実行、改変、再頒布などを許諾するライセンスで、 派生的プログラム(著作物)も GPL が適用される。 プログラム改変を許諾するためにソースコードの公開が前提となる。

1.3. 一般ユーザとスーパーユーザ

UNIX(Linux)のユーザには、システム上であらゆる操作が可能な権限(管理者権限)を持つスーパーユーザ(root)と、権限(できること)が制限された一般ユーザがあります。

通常のシステム利用には、一般ユーザで作業を行います。 スーパーユーザは、システムの設定を変更したりすることが可能であり、ミスなどによってシステムを破壊する場合もあるため、システム管理者がシステムに対する操作を行う場合のみに使用します。

1.4. UNIX(Linux)のユーザインターフェース

1.4.1. ウィンドウシステム(GUI)

UNIXやLinuxで使われているウィンドウシステム(GUI)は、X Window Systemという仕組みを使っています。

最近のLinuxディストリビューションでは、デスクトップの外観や操作性を提供する「ウィンドウマネージャ」というソフトウェアを組み合わせて、WindowsやMacと同じようにGUI操作で多くの作業ができる環境が提供されています。

- Linuxディストリビューション: Linuxカーネルと Webブラウザやメールソフトなどのアプリケーションソフトや、 それらを動かすための周辺ソフトウェアを組み合わせた配布パッケージで、 多くのディストリビューションがある。 (RedHat, Debian GNU/Linux, SUSE, Ubuntu, CentOS, Fedora, Rocky, AlmaLinux…)
- GNOME, KDEが代表的な総合デスクトップ環境

1.4.2. コマンドライン操作

キーボードから文字によるコマンドで操作します。 コマンド操作を行うには、端末(ターミナルエミュレータ、仮想端末)を使います。

パソコンのLinux環境であれば、GNOME端末などを起動すれば、コマンド操作が可能となります。 ネットワークを経由してサーバに接続する場合は、パソコン上のターミナルエミュレータソフト (PuTTY, Tera Termなど) から、 SSH(Secure Shell) で接続し、ログインした状態で、コマンド操作が可能となります。

PuTTY, Tera Term: Windows用のターミナルエミュレータソフトウェアで、SSH(SSH Ver.2対応) でのリモート接続ができる。

起動した端末、あるいはサーバにログインした端末で、カーソルが表示されている状態を「コマンドプロンプト」、「プロンプト」と呼び、現在コマンドを受け付けられる状態であることをユーザに示しています。

```
[norin@fe02 ~]$ ■
```

システムによってプロンプトは異なりますが、ユーザ名、ホスト名、現在のディレクトリ、一般ユーザ($)かroot(#)か、といった情報が表示されています。

「~」(チルダ)の部分が現在のディレクトリで、「~」は「ホームディレクトリ」(/home/login-id)を表しています。

科学技術計算システムのフロントエンドサーバのプロンプトは、 [norin@fe02 ~]$ ([ユーザ名、ホスト名、ディレクトリ]) となっています。

1.5. ログインとログアウト

1.5.1. ログイン(login)

コンソールやネットワーク(SSH)を経由して UNIX のコマンド操作を行える状態になることをログイン (login) といいます。 このときに利用者毎に登録されている ログインID (ログイン名、ユーザ名) と パスワードによって認証を行います。 (SSH公開鍵認証の場合は、登録した鍵のペアを照合して認証します。)

ログイン名、ユーザ名(login name, login-id, user name)

システムに登録されたユーザを識別する名前です。

SSH接続

SSH(Secure Shell)は、ネットワークを経由して別のコンピュータにログインしたり、コマンドを実行するためのプログラムで、ネットワーク上を流れるデータは暗号化されるため、安全に操作することができます。

SSH公開鍵認証

SSHキーペア(秘密鍵と公開鍵)を作成し、公開鍵をシステム(サーバ)に登録することで、手元の秘密鍵と公開鍵を照合し、認証を行います。 秘密鍵は厳重に管理し、パスフレーズを設定することで、秘密鍵の所有とパスフレーズの照合が必要となり、パスワード認証よりも安全性が高くなります。

Windows PCでは、PuTTYやTera Termなどで接続することができます。 MacやLinux環境からは、sshコマンドを使って接続します。

1.5.2. ログアウト(logout)

ログイン状態を解除する(シェルを終了する)ことをログアウト(logout)と言います。 SSHなどネットワークを経由している場合は、 接続も解除します。

ログアウトするには、 `exit` コマンドを使います。 `exit` コマンドの代わりに、「`Ctrl+D`」(Ctrlキーを押しながらdを押す)を使うことも可能です。

2. コマンドライン操作の基本

基本的なコマンドの実行は、以下の例の様にプロンプトからコマンドをタイプして「`Enter`」キーを押すと、結果が表示されます。

```
$ pwd
/home/norin       <- 実行結果
```

実行例などで、単に `$` と記述した場合は一般ユーザのプロンプトの意味で、$を入力する必要はない。root権限の場合は `#` とする。

`pwd` コマンドの実行結果として表示された `/home/norin` は、今いる場所「カレントディレクトリ」(current directory, working directory)を表示しています。(pwd: Print Working Directory)

```
$ ls
cow2b.sas   lfs    sastest
```

`ls` コマンドは、ディレクトリの内容を一覧表示するコマンドです。何も指定せずに実行すると、カレントディレクトリにあるファイルやディレクトリが表示されます。

2.1. シェル(shell)

ユーザがタイプしたコマンドは、**シェル(shell)**というプログラムによって解釈され、実行されます。シェルは、システムの中核プログラムのカーネルとユーザの間で、コマンドの解釈、結果の表示といったことを行います。 また、プログラミング(シェルスクリプト)言語としてシェルを利用する事もでき、自動実行などに使われています。

利用者がログインすると自動的に起動するシェルを標準シェル(ログインシェル)と言います。 多くのLinux ディストリビューションで標準シェルとなっているのが「bash」ですが、システムによって、csh(tcsh), zsh などが使われています。

- bash(/bin/bash) は、Bシェル(bourne shell, /bin/sh)を拡張したシェルです。また、Cシェル(csh, /bin/csh)を拡張した tcsh も使われています。最近のLinuxディストリビューションでは、Bシェルを bash に、Cシェルを tcsh に置き換えていることが多い。
- 商用のUNIXなどで本来(拡張版でない)のCシェル、Bシェルが使われている場合、シェルスクリプト等で拡張された機能や書式が使えない場合がありますので注意してください。

自分が使っているシェルを確認するには、「echo $SHELL」というコマンドを使います。

```
$ echo $SHELL
/bin/csh
```

表 1. 主なシェル

シェル	説明
sh	Stephen R. Bourne氏による Bourneシェル(Bシェル)は、UNIXシステムの標準となっている。シェルスクリプトの標準として使用されている。
bash	shの上位互換シェルで、Linuxの標準シェルとして使用されている。
csh	カリフォルニア大学バークレイ校で開発されたシェルで、Cシェルと呼ばれている。 文法がC言語に似ている。
tcsh	cshの機能拡張版で、Linuxなどで広く採用されている。
zsh	Bシェル系の機能拡張版で、Cシェル系の機能も一部取り込まれた高機能シェルです。 macOSの 標準シェルです。

2.2. コマンドの書式

コマンドの書式(入力形式)は、以下の様になります。

```
コマンド名 引数1 引数2 ・ ・ ・ ・ ・ ・
```

引数(ひきすう)は、コマンドに引き渡す値のことで、引数の間は1つ以上のスペースで区切られ複数の値を引き渡すことができます。

Windowsなどで作成した、ファイル名にスペースが入ったファイルを転送した場合に、 複数の引数と解釈されコマンドが正常に処理できない場合があります。

UNIX(Linux)では、コマンド、引数、ファイル名などで、大文字、小文字を区別します。 多くのコマンドは、小文字になっています。

コマンドによって引数の値や順番などが違います。

引数の指定例として、cal(calendar)コマンドで引数なし、引数1(2017)、引数1(6)と引数2(1990)を指定した場合の出力を以下に示します。

リスト 1. 引数なし(今月のカレンダー)

```
$ cal
    January 2017
Su Mo Tu We Th Fr Sa
 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
```

リスト 2. 引数1を指定(2017年のカレンダー)

```
$ cal 2017
                              2017

       January               February                March
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7               1  2  3  4            1  2  3  4
 8  9 10 11 12 13 14    5  6  7  8  9 10 11    5  6  7  8  9 10 11
15 16 17 18 19 20 21   12 13 14 15 16 17 18   12 13 14 15 16 17 18
22 23 24 25 26 27 28   19 20 21 22 23 24 25   19 20 21 22 23 24 25
29 30 31               26 27 28               26 27 28 29 30 31

        April                  May                    June
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                    1      1  2  3  4  5  6               1  2  3
 2  3  4  5  6  7  8    7  8  9 10 11 12 13    4  5  6  7  8  9 10
 9 10 11 12 13 14 15   14 15 16 17 18 19 20   11 12 13 14 15 16 17
16 17 18 19 20 21 22   21 22 23 24 25 26 27   18 19 20 21 22 23 24
23 24 25 26 27 28 29   28 29 30 31            25 26 27 28 29 30
30
```

リスト 3. 引数2を指定(1990年6月のカレンダー)

```
$ cal 6 1990
      June 1990
Su Mo Tu We Th Fr Sa
                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
```

2.3. コマンドオプション

コマンドの動作を変更する指令を「オプション」といい、コマンドの引数として指定します。

多くのコマンドオプションは、「-a」のようにハイフン「-」に続きアルファベットで指定します。 コマンドオプションのアルファベットは、大文字、小文字を区別します。 例えば、 ls コマンドで更新日時の順に表示するには、「-t」オプションをつけたり、詳細な情報を表示する「-l」オプションを付けたりします。 また、複数のオプションを続けて「ls -lt」のように指定することもできます。

```
$ ls
        cow2b.sas  lfs  SAS-BI  sasuser.v92
$ ls -t
        lfs  sasuser.v92  SAS-BI  cow2b.sas
$ ls -lt
total 12
        lrwxrwxrwx 1 norin center    7 Jun 26 10:41 lfs -> /lfs/norin
        drwxr-xr-x 2 norin center 4096 Jun 26 08:42 sasuser.v92
        drwxr-xr-x 2 norin center 4096 Jun 17 19:07 SAS-BI
```

「-アルファベット」のような指定方法の他に、「--」ハイフン2つに続けて英単語で指定するオプションもあります。

先ほどのlsコマンドで更新日時順と詳細な情報を表示するオプションは以下の様になります。

```
$ ls --sort=time --format=long
total 12
        lrwxrwxrwx 1 norin center    7 Jun 26 10:41 lfs -> /lfs/norin
        drwxr-xr-x 2 norin center 4096 Jun 26 08:42 sasuser.v92
        drwxr-xr-x 2 norin center 4096 Jun 17 19:07 SAS-BI
        -rw------- 1 norin center 3882 Feb 26 17:49 cow2b.sas
```

ローカルのLinux(PC)とリモートのサーバでは、lsコマンドの表示が違う場合があります。 これはシステムによって標準で指定されたオプションが違っているためです。 (ls -F や ls --color=auto などをお試しください)

2.4. オンラインマニュアル(manコマンド)

コマンドの利用方法やオプションを調べるために、「オンラインマニュアル」を参照することができます。

オンラインマニュアルを参照するには、コマンド man の引数にコマンド名を指定します。

```
$ man ls
LS(1)                          User Commands                          LS(1)
NAME
       ls - list directory contents
SYNOPSIS
       ls [OPTION]... [FILE]...
DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort.
       Mandatory arguments to long options are  mandatory  for  short  options
       too.
       -a, --all
              do not ignore entries starting with .
       -A, --almost-all
              do not list implied . and ..
       --author
 Manual page ls(1) line 1
```

マニュアルが長い場合は、1画面ごとに表示するページャ(less など)と呼ばれるプログラムによって表示されます。 次のページを表示するには「スペースキー」を、前のページに戻るには「b」押します。 終了するには、「q」を入力します。

2.5. コマンドラインの編集とヒストリー

2.5.1. コマンドラインの編集

コマンドオプションなど引数が多くなると、入力ミスなどが増えます。 Enterキーを押す前であればコマンドは実行されませんので、コマンドラインの修正(編集)が可能です。

修正には、矢印キー(↔)や BS(Back Space), DEL(Delete) キーなどが使えます。 また、「Ctrl+A」 (Ctrlキーを押しながらA)などのコンビネーションキーによる操作(Ctrl+A: 行の先頭に移動))も可能です。

2.5.2. ヒストリー

シェル(bash, tcshなど)には、コマンドの履歴(ヒストリー)を記憶する機能があります。

上矢印キー「↑」を押すか「Ctrl+P」を押すと、コマンドを一つずつさかのぼって表示します。 下矢印キー「↓」か「Ctrl+N」で一つずつ戻ります。

表示されたコマンドは、 Enter キーを押すと実行されます。 また、編集することができますので、間違えた箇所を修正したり、引数を少しずつ変更して実行することができます。

`history` コマンドを使うと、これまで実行したコマンドを一覧表示することができます。表示したコマンドには、連番がついていますので、「`!21`」のように、番号を指定して再実行することができます。

表 2. コマンドラインのキー操作

キー操作	説明
Ctrl + F, →	カーソルを一文字右に移動
Ctrl + B, ←	カーソルを一文字左に移動
Ctrl + D, Delete	カーソル位置の文字を削除
Ctrl + A	カーソルを行の先頭に移動
Ctrl + E	カーソルを行末に移動
Ctrl + K	カーソル位置から行末までを削除
Ctrl + U	先頭からカーソル位置までを削除
Alt + F, Esc F	カーソルを1単語右に移動
Alt + B, Esc B	カーソルを1単語左に移動
Alt + D, Esc D	カーソル位置から単語の終わりまでを削除
Ctrl + P, ↑	コマンドヒストリーを1つさかのぼって表示
Ctrl + N, ↓	さかのぼったヒストリーを1つ戻す

2.6. UNIX(Linux)でよく使うコマンド

以下に、UNIX(Linux)でよく使う主なコマンドを一覧にしました。 詳しくは、オンラインマニュアル(man コマンド名)などを参照してください。

表 3. UNIX(Linux)で使う主なコマンド

機能	コマンド	利用法	使用例
ファイル操作			
ファイルのリスト表示	ls	ls [ディレクトリ]	ls -l, ls -lv
ファイルをコピー	cp	cp [コピー元] [コピー先]	cp -r, cp -rp
ファイルを削除	rm	rm [ファイル名]	rm -r

機能	コマンド	利用法	使用例
ファイル名を変更、移動	mv	mv [元のファイル] [変更(移動)先]	mv hoge work/
ディレクトリを作成	mkdir	mkdir [ディレクトリ名]	mkdir work
カレントディレクトリを変更	cd	cd [ディレクトリ]	cd work, cd .., cd
現在のワーキングディレクトリ	pwd	pwd	
ディレクトリを削除	rmdir	rmdir [ディレクトリ名]	rmdir work
ファイル検索	find	find [ディレクトリ名] [オプション]	find work/ -name hogehge
ファイルタイプを表示	file	file [ファイル名]	file /bin/bash, file .cshrc
テキスト操作			
テキストファイルの内容を表示	cat	cat [ファイル名]	cat filename
テキストを位置画面ずつ表示	more(less)	more(less) [ファイル名]	less filename
ファイルの先頭を表示	head	head [オプション] [ファイル名]	head -30 file
ファイルの末尾を表示	tail	tail [オプション] [ファイル名]	tail -30 file, file -f file
パターンマッチ検索	grep	grep [オプション] [ファイル名]	grep [keyword] file, grep -v [keyword] file
ソート	sort	sort [オプション] [ファイル名]	sort file, sort -k 3 -n file
文字数カウント	wc	wc [オプション] [ファイル名]	wc filename
アクセス権限			
アクセス権限の変更	chmod	chmod [オプション] [ファイル名]	

機能	コマンド	利用法	使用例
ファイルアーカイブ、圧縮			
複数のファイルを一つに	tar	tar [オプション] [アーカイブ先] [アーカイブもと]	tar zcvf hoge.tar.gz dir/, tar zxvf hoge.tar.gz
ファイルの圧縮、展開	gzip, gunzip	gzip [圧縮するフィル], gunzip [解凍するファイル]	gzip hoge.txt, gunzip hoge.txt.gz
ファイルの圧縮、展開	bzip2, bunzip2	bzip2 [圧縮するファイル], bunzip2 [解凍するファイル]	bzip2 hoge.txt, bunzip2 hoge.txt.bz2
圧縮されたファイルの内容を表示	zcat	zcat [圧縮されたファイル]	zcat hoge.txt.gz
各種情報の表示			
オンラインマニュアル	man	man [コマンド名]	man ls, man man
サーバの状況を表示	w	w [オプション]	w
サーバの状況を表示	top	top [オプション]	top
プロセス表示	ps	ps [オプション]	
ユーザの環境設定表示	env	env	
ディスクの使用状況表示	df	df [オプション]	df -k
ディスクの使用量表示	du	du [オプション] [ディレクトリ名]	du -s ./*
ディスクの容量制限表示	quota	quota [オプション] quota_state (科学技術計算システムでは、quota_stateを使う)	
現在の日時を表示	date	date [オプション]	date
カレンダー表示	cal	cal [オプション] [表示指定期日]	cal -3, cal -y
シェル環境			
bashの起動	bash	bash	

機能	コマンド	利用法	使用例
aliasの設定、表示	alias	alias [エイリアス名]='[コマンド [オプション]]'	alias
aliasの解除	unalias	unalias [コマンド]	
環境変数設定 (csh)	setenv	setenv [環境変数] [設定値]	setenv LANG C
環境変数設定 (sh,bash)	export	変数名=設定値; export [環境変数]	LANG=C; export LANG
コマンド履歴	history	history	
ネットワーク			
SSH ログイン	ssh(slogin)	ssh -X norin@scion.cc.affrc.go.jp	
SCP ファイルコピー	scp	scp [コピー元] [コピー先]	scp file.name norin@scion.cc.affrc.go.jp:dirname/
SFTP ファイルコピー	sftp	sftp [ユーザ名]@[接続先ホスト名]	sftp norin@scion.cc.affrc.go.jp

`quota_state` : 科学技術計算システムでは、利用者のデータを格納したファイルサーバ上で `quota` コマンドを実行する必要があるため、別のコマンドを設定しています。

3. UNIX(Linux)のディレクトリ構造

3.1. ファイルシステム

ファイルを格納する入れ物をディレクトリと言います。 WindowsやMacでフォルダと呼ばれている物がディレクトリにあたります。

ディレクトリ自体もファイルと同様にディレクトリに格納できるので、UNIX(Linux)のファイルシステムはツリー(木)構造となっています。

ツリーの頂点(木を逆さまにしたイメージで根にあたる部分)をルート(root)と呼び「/(スラッシュ)」で表記します。

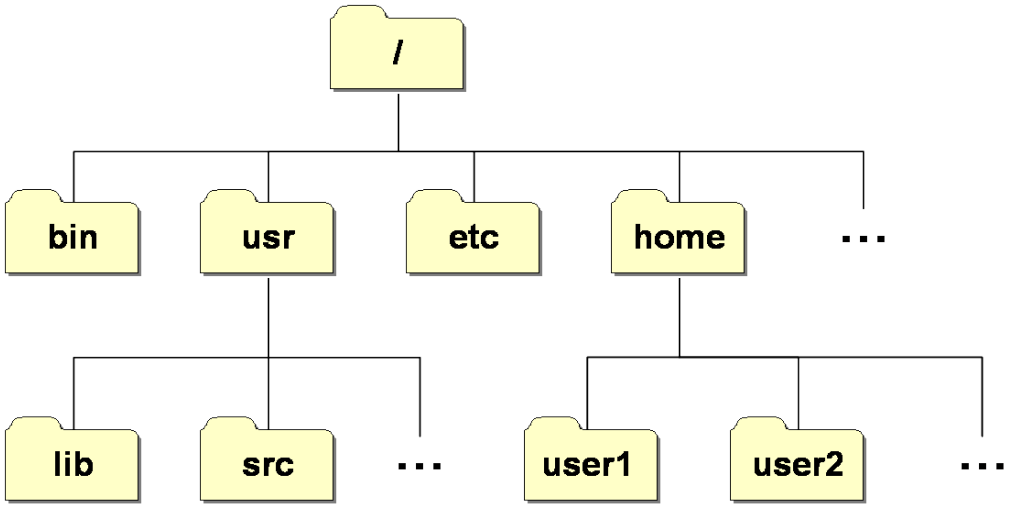

図 1. UNIXファイルシステムのイメージ

3.1.1. ユーザの位置

現在、ユーザがいるディレクトリのことをカレントディレクトリと呼びます。 pwd コマンドでカレントディレクトリを表示することができます。

ログインした時点では、ホームディレクトリと呼ばれる自分専用のディレクトリがカレントディレクトリとなります。 ディレクトリを指定せずにファイルやディレクトリの操作(ls 等コマンド操作)を行った場合には、カレントディレクトリでの作業として処理が行われます。

3.1.2. 絶対パスと相対パス

ファイルやディレクトリまでの道筋のことを「パス」と呼びます。 このパスの指定方法には、「絶対パス」と「相対パス」があります。 絶対パスは、ルート「/」からたどって指定する方法で、相対パスは、カレントディレクトリを起点として指定する方法です。

pwdコマンドを実行すると、「/home/login-id」のように絶対パスで表示されます。 ディレクトリの区切りも「/」が使われています。

「ls SAS-BI」のようにlsコマンドの引数が「/」で始まる絶対パスでない場合は、カレントディレクトリからの相対パスとして処理されます。

```
$ pwd
/home/norin
$ ls
cow2b.sas  lfs  SAS-BI  sasuser.v92
$ ls SAS-BI
milk.sas7bdat  systemcpu.sas7bdat
```

3.2. ファイルの表示

3.2.1. テキストファイルとバイナリファイル

テキストファイル(text file)は、文字や数字などの文字コードで構成され、人間が読んで理解できる形式のデータ(シェルスクリプトやCSVデータなど)のファイルです。 テキストファイル(テキストデータ)は、端末に表示したりテキストエディタで編集することができます。

これに対して、バイナリファイル(binary file)は、コンピュータが処理するために2進数で構成されたファイルで、人間がそのまま読んで理解することは困難です。

バイナリファイルには、画像、音声、圧縮されたファイルなどがあります。 また、プログラム言語 (C, FORTRANなど) で書かれたソースコード(テキスト)に対して、 コンパイル(コンピュータで実行可能な形式に変換)されたプログラム(オブジェクトファイルや実行ファイル)をバイナリと呼ぶことがあります。

テキストファイルかバイナリファイルかを調べるコマンド `file`

```
$ file hello.c
hello.c: C source, ASCII text
$ file hi
hi: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=3fa1025d3a53dcc98a69359b2486e4ed1c620171, not stripped
$ file amedas-kion.png
amedas-kion.png: PNG image data, 640 x 480, 8-bit colormap, non-interlaced
```

3.2.2. テキストの中身を表示する

単にテキストファイルの中身を表示するには、cat(concatenate) コマンドを使います。

```
cat 表示するファイルのパス
```

`cat` の逆にファイルの末尾から表示する `tac` コマンドや、行の文字を逆の並びで表示する `rev` コマンドもあります。

ファイルの最後まで表示しますので、長いファイルでは最後しか見ることができません。

```
norin@fe02 $ cat cow2b.sas
 中略
run;
proc reg data=cow;
    model bl=dayl day;
    data cow3;
set cow;
    bh=exp(3.26617)*day**(0.05858)*exp(-0.00307*day);
proc print;
    symbol1 interpol=join color=green;
    symbol2 interpol=join color=red;
```

```
proc gplot;
    plot b*day bh*day /overlay; run;
norin@fe02 $
```

長いファイルを画面毎にスクロール表示するには、lessコマンドを使います。

```
norin@fe02 $ less cow2b.sas
data cow;
    input day a b;
    dayl=log(day);
    al=log(a);
    bl=log(b);
cards;
6     20.3    23.3
7     18.1    24.6
8     17.6    24
9     16.3    25.6
10    19.9    25.6
11    19.9    26.4
12    19.6    27.3
13    19.9    28.4
14    19.2    28.2
15    20.3    27.4
16    20.5    29.7
17    20.9    30.4
18    20.8    29.4
19    21.8    31.6
20    21.2    31.3
21    21.3    31.2
22    21.9    31.4
cow2b.sas
```

lessは、逆スクロールや対話的な文字検索なども可能な、moreコマンドの上位互換のツールです。
lessを終了するには q を押します。

lessを実行中に、hを押すとlessのヘルプが表示されます。

```
                SUMMARY OF LESS COMMANDS
    Commands marked with * may be preceded by a number, N.
    Notes in parentheses indicate the behavior if N is given.
h  H                 Display this help.
q  :q  Q  :Q  ZZ     Exit.
---------------------------------------------------------------------------
                         MOVING
e  ^E  j  ^N  CR  *  Forward  one line   (or N lines).
y  ^Y  k  ^K  ^P  *  Backward one line   (or N lines).
f  ^F  ^V  SPACE  *  Forward  one window (or N lines).
b  ^B  ESC-v      *  Backward one window (or N lines).
```

3.2.3. ファイルの先頭や末尾を表示する

沢山のテキストファイルの先頭だけを確認する場合など、ファイルの先頭の部分を表示するコマンドが head です。

ファイルの末尾を見るコマンドが tail です。 tail -f オプションを使うと、ログファイルなどの末尾に追加された行を続けて表示することができます。

4. ディレクトリ操作

4.1. カレントディレクトリの移動

ログイン時の作業ディレクトリ(カレントディレクトリ)が、ホームディレクトリ(/home/login-id)です。 特定のディレクトリ配下のファイルに対して続けて作業(コマンド操作)を行う場合、ファイルが格納されているディレクトリに移動した方がパスの指定が簡単になります。

ディレクトリを移動するには cd(change directory) コマンドを使います。

```
$ cd 移動先のパス
```

SAS-BIディレクトリに移動するには、以下の様にします。

```
[norin@fe03 ~]$ cd SAS-BI
[norin@fe03 ~/SAS-BI]$ pwd
/home/norin/SAS-BI
```

ホームディレクトリに戻るには、 cd /home/login-id のようにしても良いですが、引数無しで cd コマンドを実行すると、ホームディレクトリに戻ることができます。

```
[norin@fe03 ~/SAS-BI]$ cd
[norin@fe03 ~]$ pwd
/home/norin
```

4.2. ディレクトリを表す記号

よく使うディレクトリを表す記号を使うとパスの指定が簡単になります。

4.2.1. ホームディレクトリ

コマンドプロンプトでも紹介しましたが、「~」(チルダ)はホームディレクトリを表しています。 どこかのディレクトリで作業を行っている時に、ホームディレクトリ配下のディレクトリに移動したり、ファイルにアクセスするときなどに使います。 (単にホームディレクトリに戻るだけなら、引数無しの cd コマンドで可能です。)

例えば、 SAS-BI ディレクトリから、ホームディレクトリ配下の lfs ディレクトリに移動するには、以下の様にします。

```
[norin@fe03 ~/SAS-BI]$ cd ~/lfs
```

4.2.2. 1つ上位のディレクトリ

「..」(ピリオドを2つ続けた記号)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。

例えば、カレントディレクトリを1つ上のディレクトリに移動するには、以下の様にします。

```
[norin@fe03 ~/SAS-BI]$ cd ..
[norin@fe03 ~]$ pwd
/home/norin
```

「/」(スラッシュ)を組み合わせてディレクトリを上位にたどって行くことができます。

例えば、2つ上位のディレクトリに移動するには、以下の様にします。

```
$ cd ../..
```

4.2.3. カレントディレクトリ

「.」(ピリオド1つ)は、カレントディレクトリを表します。

この「.」は、ファイルのコピーや移動先にカレントディレクトリを指定する際に使ったり、シェルスクリプトやプログラムの実行のときにプログラムファイルを指定する場合に使います。

例えば、SAS-BI/milk.sas7bdat ファイルを、カレントディレクトリにコピーする際に、絶対パスではなく「.」を使うことができます。

```
[norin@fe03 ~]$ cp SAS-BI/milk.sas7bdat .
```

4.2.4. 直前の作業ディレクトリ

cd コマンドは、引数に「-」を指定すると1つ前にいたディレクトリに戻ることができます。 これを使うと、2つのディレクトリを行ったり来たりすることができます。

```
[norin@fe03 ~]$ cd SAS-BI/
[norin@fe03 ~/SAS-BI]$ cd ../lfs
[norin@fe03 ~/lfs]$ cd -
[norin@fe03 ~/SAS-BI]$ cd -
[norin@fe03 ~/lfs]$
```

コマンドプロンプトの表示で、ディレクトリが変わっていることが確認できます。

ただし、 cd コマンド以外では「-」を引数に指定すると、別の機能と見なされますのでご注意ください。

bash(/bin/bash)の場合、 ls ~- の様にすると、直前のディレクトリの一覧を表示することができます。 ただし、csh, tcshなどでは無効です。

4.3. ディレクトリの作成と削除

4.3.1. ディレクトリの作成

ディレクトリを作成するには、 mkdir コマンドを使います。

```
$ mkdir 作成するﾃﾞｨﾚｸﾄﾘのﾊﾟｽ
```

例えば、カレントディレクトリ配下に Data と言うディレクトリを作成するには、以下の様にします。

```
$ mkdir Data
```

また、 Data ディレクトリとその配下に 2009 ディレクトリを一気に作成するには、 -p オプションを使って、以下の様に実行します。

```
[norin@fe03 ~]$ mkdir -p Data/2009
[norin@fe03 ~]$ ls Data
2009
```

4.3.2. ディレクトリの削除

ディレクトリを削除するには、rmdirコマンドを使います。

rmdir で削除するディレクトリ配下には、ファイルが無いことが前提です。(ファイルがあると削除できません)

```
[norin@fe03 ~]$ rmdir Data/
rmdir: Data/: Directory not empty
[norin@fe03 ~]$ rmdir Data/2009/
[norin@fe03 ~]$ rmdir Data/
```

この例の1行目では、 Data ディレクトリに 2009 ディレクトリがある(ディレクトリが空ではない)ために、削除できませんでした。

4.4. ファイル属性

4.4.1. 隠しファイル属性

ls コマンドで -a オプションを使うと、通常表示されない . (ドット) で始まるファイル(隠しファイル)が表示されます。 隠しファイルは、通常表示する必要のない設定ファイルなどが多く、ドットファイルとも呼ばれています。

.bashrc や .profile などの設定ファイルで、システムの利用環境などが設定されます。

4.4.2. ディレクトリの属性

ls コマンドで -F オプションを使うと、ディレクトリと通常ファイルを区別して表示します。 ディレクトリ名の後ろには / (スラッシュ)が表示され、通所のファイルとは区別されています。

-a オプションと組み合わせて ls -aF とすると、 ./ や ../ というディレクトリが表示されます。 これがカレントディレクトリを表す「.」と、1つ上位のディレクトリを表す「..」になります。

ls コマンドに -l オプションを使うと、詳細なファイルリスト情報が表示されますが、最初が「d」で始まるファイルがディレクトリで、「-」で始まるのが通常のファイルです。 その後に続く文字「rw-rw-r」などは、許可属性(パーミッション)で、ファイルの所有者、グループ、その他のユーザに対する r(read:読み込み) 、 w(write:書き込み) 、 x(execute:実行) 権限の有無を表しています。 ディレクトリに対する実行権限とは、そのディレクトリをカレントディレクトリとすることを許可する権限となります。

```
$ ls -alF unix-start/
合計 96
drwxr-x---   2 norin users  4096  5月 31 09:53 ./
drwx------ 120 norin users 20480  6月  1 13:17 ../
-rw-r-----   1 norin users  7116  5月 31 16:06 amedas-kion.png
-rwxr-----   1 norin users  1458  5月 31 09:37 amedas-kion.sh*
-rw-r-----   1 norin users    90  5月 31 09:41 hello.c
-rwxr-x---   1 norin users 17432  5月 31 09:42 hi*
```

5. ファイルのコピー、移動、削除

5.1. ファイルのコピー cp(copy)

ファイルをコピーするには、「cp」コマンドを使います。

引数には、「コピー元」「コピー先」の順に指定します。コピー先にファイル名を指定すると、その名前でファイルの複製が作成されます。 コピー先にディレクトリを指定すると、そのディレクトリ配下にコピー元と同じ名前のファイル名でコピーされます。

```
cp cow2b.sas milk.sas
cp milk.sas work/
cp milk.sas work/cow3.sas
cp work/cow3.sas .
```

カレントディレクトリを表す「.」を使うことも可能です。

5.1.1. ディレクトリを丸ごとコピー

ディレクトリ内のファイルを含めてを丸ごとコピーするには、-R オプションを使います。

```
cp -R work/ backup
```

5.1.2. ファイルの属性を保持してコピー

ファイルをコピーすると、更新日時などがコピーを実行した日時に変更されてしまいます。 ファイルのバックアップなど、ファイルの属性(変更日時やパーミッション)をそのまま保持してコピーしたい場合は、 -p オプションを使用します。

```
cp -p cow2b.sas cow2b.sas.bak
```

ディレクトリごとバックアップを行う場合などは、 cp -cdpR オプションを組み合わせて使います。 -a オプションは、 -cdpR オプションと同じ動作をします。

```
cp -a work/ backup
```

5.2. ファイルの移動 mv(move)

ファイルの移動コマンド「mv」は、コピー(cp)コマンドと同様の使い方ですが、移動元のファイルが消去されます。 移動先に別のファイル名を指定することで、ファイル名の変更ができます。

```
mv milk.sas cow4.sas
mv cow4.sas work/
mv work/milk.sas .
```

5.3. ファイルの削除 rm(remove)

ファイルを削除するコマンド「rm」の基本的な使い方は、「rm ファイル名」とします。 複数のファイルを指定することができます。

ディレクトリを再帰的に(ディレクトリ内のファイルを含めて全て)削除する場合は、 -r オプションを使います。

rm で削除したファイルを復活させることはできませんので、 rm コマンドの実行は慎重に行ってください。

```
rm milk.sas
rm -r work/
```

6. ワイルドカード

複数のファイルに対してコマンド操作を行う場合に、「ワイルドカード」と呼ばれる特殊文字を使うことができます。

6.1. 任意の文字列「*」

「 * 」(アスタリスク)は、0個以上の任意の文字列を表します。 例えば、「 c*.sas 」であれば、 c.sas, cow.sas, cow2b.sas などにマッチします。

拡張子が「.sas」のファイルを全て「work」ディレクトリにコピーするには、以下の様にします。

```
cp *.sas work/
```

コピーの状況を表示するには、 cp -v オプションを使います。

```
cp -v *.sas work/
```

6.2. 任意の1文字「?」

「?」(クエスチョンマーク)は、任意の1文字を表します。 例えば、「c??.sas」であれば、 cow.sas にマッチしますが、 c.sas や cow2b.sas にはマッチしません。

```
$ ls c????.*
cow2b.log   cow2b.sas
```

6.3. 指定した文字のいずれかとマッチ

指定した文字のどれかにマッチさせるには、「[」と「]」の間に文字を指定します。

例えば、「c」、「f」、「s」で始まり、任意の拡張子3文字のファイル一覧を表示するには、以下の様にします。

```
$ ls [cfs]*.???
cow2b.log     cow-test1.sas    sas.log         sasprt.pdf
cow2b.sas     file.out         sas1.fonts.txt  sasprt.png
```

```
cow2.gif      fork.txt         sasgraph.png
```

文字を範囲指定する場合は、「-」ハイフンを使います。 例えば、数字で終わるファイルの一覧を表示するには、以下の様にします。

```
$ ls *[0-9]
index.html.1       sas-proc.200907080900   STDIN.o1360
java.log.16134     sas-proc.20090707       STDIN.e1360
```

7. 標準入出力とリダイレクション

7.1. 標準入出力 STD I/O(Standard Input / Output)

UNIX(Linux)では、コマンドのパラメータやデータの入力は「標準入力(stdin)」から行い、実行結果やメッセージは「標準出力(stdout)」に出力されるようになっています。 コマンドに何も指定をしなければ、標準入力と標準出力は、それぞれキーボードと画面(端末)に接続されています。

エラーメッセージの出力先「標準エラー(stderr)」も、標準出力と同じ画面に割り当てられています。

7.2. リダイレクション

これらの標準入出力や標準エラーの接続先を変更(リダイレクション)することができます。

7.2.1. 標準出力リダイレクション(>,>>)

標準出力をファイルに切り替えるには、「>」という記号を使って以下の様にします。

```
$ cal > Jan.2017
$ cat Jan.2017
    January 2017
Su Mo Tu We Th Fr Sa
 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
```

`cal` コマンドの結果は画面には表示されず、 `Jan.2017` ファイルに格納されました。 `cat` コマンドで、`Jan.2017` ファイルを表示すると、カレンダーが表示されます。

「>」の記号を2つ続けて「>>」とすると、ファイルの最後に追加されます。

```
$ cal 2 2017 >> Jan.2017
$ cat Jan.2017
    January 2017
```

```
Su Mo Tu We Th Fr Sa
 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

   February 2017
Su Mo Tu We Th Fr Sa
          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
```

ファイルの日本語文字コードの変換ツール nkf の実行でも「>」を使います。

例：変換元ファイル(in-file.txt)を、文字コードを UTF-8 に変換、同時に改行コードをunix用に変換して、ファイル(out-file.txt)に保存する。(同じファイルへの入出力はできないので、一旦別のファイル名で保存します。)

```
nkf -w -Lu in-file.txt > out-file.txt
```

7.2.2. 標準入力リダイレクション(<)

標準出力と同様に、「<」を使って標準入力をキーボードからファイルに切り替えることができます。

bc と言うコマンドは、通常キーボードからの入力によって計算を行います。

```
[norin@fe03 ~]$ bc
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
5+9
14
1248*563
702624
quit
[norin@fe03 ~]$
```

次に、 keisan.txt というファイルに、計算式を保存します。 ファイル作成には、cat コマンドと標準出力リダイレクションを使います。 ファイルを指定せずに cat コマンドを実行すると、標準入力からの入力待ちの状態になり、入力された文字列をオウム返しのように画面に表示します。 この画面出力をファイルにリダイレクションします。

```
[norin@fe03 ~]$ cat > keisan.txt
5+9
1248*563
```

```
CTRL+D (Ctrl キーを押しながら d を押す)
[norin@fe03 ~]$ cat keisan.txt
5+9
1248*563
[norin@fe03 ~]$
```

標準入力リダイレクションを使って、計算を実行してみます。

```
[norin@fe03 ~]$ bc < keisan.txt
14
702624
[norin@fe03 ~]$
```

計算結果が画面(標準出力)に表示されます。

リダイレクションを使わずに `bc keisan.txt` とすると、計算は実行されますが、入力待ちの状態になります。 `quit` で `bc` を終了します。

```
[norin@fe03 ~]$ bc keisan.txt
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
14
702624
quit (入力待ち状態になる。quit で bc を終了します。)
```

7.2.3. 標準エラーのリダイレクション

エラーや警告のメッセージは通常「標準エラー」に出力され、標準出力と同様に画面に表示されます。

標準エラーをリダイレクトするには、「2>」という記号を使います。

```
コマンド 2> ファイルのパス
```

標準出力と標準エラーを1つのファイルに保存するには、以下「&>」の様にします。

```
コマンド &> ファイルのパス
```

7.2.4. ファイルディスクリプタ

標準エラーのリダイレクションで指定した番号2は、ファイル記述子(ファイルディスクリプタ)と呼ばれる番号で、標準入力は「0」、標準出力は「1」、標準エラーは「2」が割り当てられています。

7.3. パイプ (コマンドを接続する)

コマンドの実行結果を画面(標準出力)に出力しないで、直接、他のコマンドの入力に受け渡すことができます。 この機能を「パイプ」と言い、「|](縦棒記号)」を使います。

```
$ echo "4*a(1) " | bc -l
3.14159265358979323844
```

ここで、 echo "4*a(1)" は、「4*a(1)」という文字列を画面に表示するコマンドです。 これを bc コマンドの入力に渡すことで、計算結果が画面に表示されます。

bc の -l オプションは、数学ライブラリを読み込むオプションです。また、 a(x) は、ATAN (アークタンジェント) 関数です。

この様に「|」記号を挟んで複数のコマンドを連結することで、複雑な処理を行う事ができます。

例えば、文字コードがShift-JISやEUCなどのファイルを、文字コード変換コマンド(nkf)を使って変換し、 1画面ずつ表示するページャ(less)に渡すことで、 UTF-8の画面でも文字化けすることなくファイルを確認することができます。

```
[norin@fe03 ~]$ nkf -w sas-prog-test.log | less
1                                                          SAS システム
                            2009年11月17日 火曜日 午後02時41分30秒
NOTE: Copyright (c) 2002-2008 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software 9.2 (TS2M0 DBCS3055)
      Licensed to THE MINISTRY OF AGRICULTURE, FORESTRY AND FISHERIE, Site 10204140.
NOTE: このセッションは SunOS 5.10 (SUN X64) ﾌﾟﾗｯﾄﾌｫｰﾑ上で､実行されています。
You are running SAS 9. Some SAS 8 files will be automatically converted
by the V9 engine; others are incompatible. Please see
http://support.sas.com/rnd/migration/planning/platform/64bit.html
PROC MIGRATE will preserve current SAS file attributes and is
recommended for converting all your SAS libraries from any
SAS 8 release to SAS 9.  For details and examples, please see
http://support.sas.com/rnd/migration/index.html
This message is contained in the SAS news file, and is presented upon
:
```

8. テキストエディタ vi(vim)

プログラムやデータなどのテキストファイルを編集するため、テキストエディタ(単にエディタとも言う)を使うことがあります。

vi は、UNIX(Linux) で使われる標準的な(ほとんど全てのUNIXに入っている)エディタです。 vi には 「ノーマルモード」 と 「挿入モード」 があり、 多くのコマンドがあるため、 初心者にはわかりづらく、とっつきにくい印象ですが、慣れてくれば素早くファイル編集ができる ツールとして活用でき、 vi の操作を習得することは大変有用です。

Linux では、 vi の機能を拡張した vim (Vi IMproved) が標準の vi に置き換えられていることが多くなっています。 vim が導入されていない場合や、システムの環境設定によって、基本的な機能のみで動作する vi が起動するようになっている場合もあります。

vi が vim に置き換わっている。(シンボリックリンク)

```
norin@fe03 $ which vi
/usr/bin/vi
norin@fe03 $ ls -l /usr/bin/vi
lrwxrwxrwx 1 root root 8 12月  1  2014 /usr/bin/vi -> /bin/vim
```

UNIXで人気のあるエディタとして、 emacs というエディタもあります。 また、Windowsのメモ帳 (notepad) の様な GUI のエディタもありますが、ターミナル(端末)からリモートで操作を行うには、 vi(vim) や emacs など、キーボードだけで操作できるエディタが必要となります。

8.1. ファイルのオープン (起動)

既存のファイル、または新しいファイルに対してエディタを起動するには、 vi (または vim)コマンドを使います。

```
vi [filename]
例: vi hoge.txt
既存ファイルｶﾞあれば､そのファイルを開きます。
ファイルｶﾞ無ければ､新規のファイルとして編集することｶﾞ出来ます。
```

新しいファイルの場合、画面は次のようになります。起動直後は、「ノーマルモード」ですのでご注意ください。キーを入力すると、コマンドと解釈されます。

```
~
~
~
~
~
~
~
"hoge.txt" [New File]                                         0,0-1       All
```

8.2. ファイルのセーブと終了

編集したファイルを保存して終了するには、 ZZ (大文字)とします。 ただし、「挿入モード」の場合は、 Esc キーを押して「ノーマルモード」にしてから、 ZZ とタイプします。

ファイルを保存するだけの場合は、 :w とします。 別名で保存するには、 :w filename のようにファイル名を 指定します。

何も編集をしていない状態で終了する場合は、 :q とタイプします。 終了と保存を同時 にするには、

:wq とタイプします(:wq と ZZ は同じ)。

編集した内容がめちゃくちゃで、最初から編集をし直したい場合は、 :e! としてファイルを再読込します。 また、編集内容を破棄して強制終了するには、 :q! とします。

8.3. 基本的な編集操作

8.3.1. カーソルを移動する

viでカーソルを移動するには、「j」「k」「l」「h」キーを使います。

```
k キーは上方向に移動します。       ↑
h キーは左方向に移動します。       k
l キーは右方向に移動します。 ← h    l →
j キーは下方向に移動します。       j
                                   ↓
```

8.3.2. 文字を入力する

vi は「ノーマルモード」と「挿入モード」があります。 挿入モードにするには何通りかの方法がありますが、一番標準的なコマンドは、 i を押すやり方です。

i は表示されませんが、一番下の行が -- INSERT -- に変わり、これを押した後にタイプした文字が全て画面に入力されるようになります。

次の操作(カーソルを移動する場合も) をしたい場合は、 Esc キーを押してノーマル モードに戻ります。

表 4. vi 入力モードに関連するコマンド

コマンド	内容
a	カーソルの後にテキストを入力
A	カレント行の末尾にテキストを入力
i	カーソルの前にテキストを入力
I	行の先頭にテキストを入力
o	カーソル位置の下にテキストを挿入する空行をオープン
O	カーソル位置の上にテキストを挿入する空行をオープン

8.4. テキスト削除

削除コマンド x を使うと、カーソル位置のテキストを削除します。 単語単位削除は dw 、行削除は dd を

使います。

さらに、これらのコマンドの前に数字を入力すると、コマンドの対象範囲が指定できます。 例えば、`2x` は 2 文字、 `2dw` は 2 word、 `2dd` は 2 行の削除となります。

8.5. vimチュートリアル

基本的な使い方については、vimのチュートリアルで学習することができます。 `vimtutor ja` としてチュートリアルを起動します。

```
$ vimtutor ja
===============================================================================
=    V I M 教 本 (チュートリアル) へ よ う こ そ        -    Version 1.7      =
===============================================================================

     Vim は、このチュートリアルで説明するには多すぎる程のコマンドを備えた非常
     に強力なエディターです。このチュートリアルは、あなたが Vim を万能エディ
     ターとして使いこなせるようになるのに十分なコマンドについて説明をするよう
     になっています。

     チュートリアルを完了するのに必要な時間は、覚えたコマンドを試すのにどれだ
     け時間を使うのかにもよりますが、およそ25から30分です。

     ATTENTION:
     以下の練習用コマンドにはこの文章を変更するものもあります。練習を始める前
     にコピーを作成しましょう("vimtutor"したならば、既にコピーされています)。

     このチュートリアルが、使うことで覚えられる仕組みになっていることを、心し
     ておかなければなりません。正しく学習するにはコマンドを実際に試さなければ
     ならないのです。文章を読んだだけならば、きっと忘れてしまいます!

     さぁ、Capsロック(Shift-Lock)キーが押されていないことを確認した後、画面に
     レッスン1.1 が全部表示されるところまで、j キーを押してカーソルを移動しま
     しょう。
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                          レッスン 1.1:  カーソルの移動
```

8.6. vim関連情報

参考となる情報

- Vim のブログ http://vimblog.hatenablog.com/entry/index
- 名無しのvim使い http://nanasi.jp/

8.6.1. vimの日本語文字コード自動判別

最近の Linux ではデフォルトのロケールが゛ UTF-8 になっています。

vim も標準で文字コードの自動判別や変換に対応しているのですが、自動判別を有効にするには、

.vimrc を設定する必要があります。

科学技術計算システムのフロントエンドサーバは、 LANG=ja_JP.UTF-8 になっています。 EUC や SJIS のプログラムやデータを使うには UTF-8 にする必要があります。

WindowsなどSJISで作成したプログラムやデータを vim で確認したり編集したりする場合に、文字コードの自動判別や変換ができると大変便利になります。

文字コードと改行コード自動認識の設定

~/.vimrc に、自動判別の設定を追加します。 .vimrc が無い場合は作成します。

```
:set encoding=utf-8
:set fileencodings=iso-2022-jp,euc-jp,sjis,utf-8
:set fileformats=unix,dos,mac
```

文字コード, 改行コードを指定して保存

vim でファイルを開き、任意の文字コードでファイルを保存 (変換) することがで゛きます。

vim でファイルを開き、vim のコマンドモードで以下のコマンドで文字コードを指定すると、保存するときに、指定した文字コードに変換されます。

文字コードを指定

```
:setl fenc=文字コード (setl=setlocal, fenc=fileencoding)
```

改行コードを指定

```
:setl ff=改行コード (setl=setlocal, ff=fileformat)
```

上記、文字コードを指定した後に、ファイルを保存します。

```
:w
```

文字化けしたファイルを開き直す

自動判別に失敗した場合に、文字コードを指定してファイルを開き直すことができます。

vim でファイルを開き、文字化けした状態で、vim のコマンドモードで以下コマンドでファイルを開き直します。

```
:e ++enc=文字コード
```

vim でファイルを開き、^M という文字が表示される場合、改行コードを変更してファイルを開き直します。

```
:e ++ff=改行ｺｰﾄﾞ
```

9. ネットワークの利用

手元のコンピュータ(ローカルホスト)から、ネットワークに接続されている別のUNIX(Linux)コンピュータ(リモートホスト)に、ログインして操作する主なコマンドについて説明します。

9.1. リモートログイン SSH (Secure Shell)

SSH : Secure SHell (セキュアシェル)は、ネットワークを通じて別のコンピュータにログインしたり、コマンドを実行したり、ファイルを転送するためのロトコル(規約)、あるいはその実装プログラムです。

ネットワークを流れるデータは暗号化されるため、インターネット経由でも一連の操作を安全(盗聴や成りすましに対して安全)に行うことができます。

macOSは、UNIX(BSD系)を基本技術の一つとして作られており、これから説明するsshコマンド(OpenSSHのコマンドが利用可能)などの操作をターミナルから行う事ができます。

sshによるリモートログインには、「パスワード認証」と「公開鍵認証」の2つの方法があります。 ここでは、パスワード認証について説明します。

リモートログインを行うコマンドは、ssh (slogin) です。

```
$ ssh login-id@remote.host.jp または、
$ ssh -l login-id remote.host.jp
login-id@remote.host.jp's password: <== ここにパスワードを入力
```

接続すると、パスワードの入力を求めてきますので、パスワードを入力します。 入力したパスワードは表示されません。

接続元(ローカルホスト)のログイン名と、接続先(リモートホスト)のログイン名が同じであれば、login-idの指定を省略できます。

```
norin@norin-desktop:~$ ssh scion.cc.affrc.go.jp
norin@scion.cc.affrc.go.jp's password:
```

ローカルホストとリモートホストの日本語環境が異なる場合は、ログイン先環境の環境に合わせて、端末ソフトの設定を変更する必要があります。

- OpenSSH (Open Secure Shell) は、OpenBSDプロジェクトにより開発が行われ、 BSDライセンスで公開されているSSHプロトコルを利用するためのソフトウェアで、SSHサーバおよびSSHクライアントを含みます。
- リモートログインを行うコマンドとしては、sloginの方が本来のコマンドですが、sshとsloginの実態は同じプログラムです。

windows用ssh対応ターミナルソフト

Windows PC で使える ssh に対応したターミナル (端末) ソフトを紹介します。

PuTTY

Simon Tatham 氏によって作成された、フリーの Telnet/SSH クライアントです。

- https://www.chiark.greenend.org.uk/~sgtatham/putty/index.html
- https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

Tera Term

TeraTerm は、原作者の寺西高氏から、TeraTerm Project に開発が引き継がれ公開されている SSH に対応したターミナルエミュレータです。

- https://ttssh2.osdn.jp/

9.2. ファイル転送 scp, sftp

SCPは、遠隔のコンピュータとローカルのコンピュータ間でファイル転送を行うプログラムです。
SFTPは、SCP同様にファイル転送を行いますが、FTPと同様の使い方ができます。 どちらの場合も通信は暗号化されますので、ネットワーク上の盗聴などに対して安全です。

9.2.1. scp

基本的な使い方は、`cp` (コピー) コマンドと同じですが、リモートのファイルを指定する場合は、「ユーザ名@ホスト名:」が入ります。

: (コロン)の後のファイル名は、ホームディレクトリからの相対パスか、ルート(/)からの絶対パスで指定します。

```
scp [user@host1:]file1 [user@host2:]file2
scp コピー元ファイル名 コピー先ファイル名
```

パソコンから、リモートホストへ

```
$ scp program.f login-id@remote.host.jp:test/prog/
```

リモートホストからパソコンへ(最後の「 . 」は、カレントディレクトリを示す)

```
$ scp login-id@remote.host.jp:test/prog/program.f .
```

ディレクトリ配下を全てコピーする(作成日時やパーミッション変更無し)

```
$ scp -rp login-id@remote.host.jp:/hoem/login-id/test/ .
```

9.2.2. sftp

sftpは、ftpと同様に対話的な操作ができます。 一度の接続で複数のファイルや双方向のファイル転送を行う場合は、sftpが便利です。

```
$ sftp login-id@remote.host.jp
Connecting to remote.host.jp...
login-id@remote.hoet.jp's password: <== ここにパスワードを入力
sftp >
```

パスワードを入力し、認証に成功すると 「 sftp >」と表示したら、コマンドを入力して操作します。

終了するには、quit (bye, exitでもよい)と入力します。

表 5. sftpの主なコマンド

コマンド	機能
quit	sftpを終了する (bye, exitでもよい)
put	ローカルからリモートへ転送
get	リモートからローカルへ転送
cd	リモートのディレクトリを移動
ls	リモートのファイル一覧
pwd	リモートのカレントディレクトリを表示
rm	リモートのファイルを削除
rmdir	リモートのディレクトリを削除
lcd	ローカルのディレクトリを移動

コマンド	機能
lls	ローカルのファイル一覧
lpwd	ローカルのカレントディレクトリを表示
!command	ローカル上のcommandを実行

sftpのGUIツール

sftpをGUIで操作するツールがあります。

Windows系

- WinSCP (https://winscp.net/eng/docs/lang:jp)
- FFFTP (https://ja.osdn.net/projects/ffftp/)

Mac

- Cyberduck (https://cyberduck.io/index.html)

Linux

- gFTP (https://www.gftp.org)

9.3. SSH公開鍵認証

パスワード認証と比較して、よりセキュアな公開鍵認証を利用する手順を紹介します。

- 自分の認証キー (秘密鍵と公開鍵のペア) を生成します。
- 接続するホストに公開鍵を登録します。
- `ssh -i [認証キー(秘密鍵)] [login-name]@[server-name]` コマンドの `-i` オプションで、秘密鍵の場所 (`~/.ssh/id_ed25519` など) を指定して接続します。
- 接続先ホストのユーザ環境に公開鍵が登録されている場合、秘密鍵と照合されます。 秘密鍵を 参照するには、パスフレーズの入力が必要です。

科学技術計算システムへのログインは、公開鍵認証(パスワード認証不可)になりました。
科学技術計算システムでは、公開鍵登録用の専用Webサイトで登録を行います。
パスワード認証を禁止しているサーバについては、それぞれのシステムで公開鍵登録の方法が異なりますので、管理者に確認してください。 (公開鍵をメールやファイルなどで管理者あてに送り、管理者が登録するといった方法もあります)

9.3.1. SSH秘密鍵と公開鍵の生成

ssh(OpenSSH) の ssh-keygen コマンドを使って、SSH鍵を生成します。

リスト 4. SSHキーペアの生成(ssh-keygenコマンド)

```
% ssh-keygen -t ed25519                                                    ①
Generating public/private ed25519 key pair.
Enter file in which to save the key (/Users/norin/.ssh/id_ed25519):       ②
Enter passphrase (empty for no passphrase):                               ③
Enter same passphrase again:                                              ④
Your identification has been saved in id_ed25519
Your public key has been saved in id_ed25519.pub
The key fingerprint is:
SHA256:yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy norin@pc-22007          ⑤
The key's randomart image is:
+--[ED25519 256]--+
|   .+..    ..    |
|    o . o... .   |
|     o = O. +    |
|    . B & *o .   |
|     o S #  o    |
|      + @ =. .E.|
|       * = .o ..|
|       . + o  .  |
|          o      |
+----[SHA256]-----+
```

① ssh-keygen コマンドでキーを生成します。 オプション -t ed25519 で鍵の種類を ED25519 に指定します。 鍵の種類は、ED25519 をお勧めします。

② 生成する鍵ファイルの格納ディレクトリとファイル名を指定します。 (デフォルトならば、 Enter キーを押す)

③ 鍵に設定するパスフレーズを入力します。 (必ずパスフレーズを設定してください。)

④ 確認のため、パスフレーズを再入力します。

⑤ 生成した鍵のフィンガープリント (鍵のハッシュ値) が表示されます。

リスト 5. 生成したキーファイル

```
% ls -l ~/.ssh                                                            ①
total 48
-rw-------  1 norin  staff   464  4 18  2021 id_ed25519                    ②
-rw-r--r--  1 norin  staff   105  4 18  2021 id_ed25519.pub                ③
```

① 生成した鍵ファイルは、ホームディレクトリ(/Users/[ユーザ名]/)配下の .ssh ディレクトリに保存されます。 (~/ は、ホームディレクトリ)

② 生成された秘密鍵(ファイルのアクセス権が -rw------- になっていること。)

③ 公開鍵をログイン先ホスト(サーバ)に登録します。

生成する鍵の種類は、Ed25519 か RSA 2048 bit以上を作成してください。 楕円曲線暗号の一つである Ed25519 をお勧めします。 (暗号強度が高く鍵長が短い)
パスフレーズを必ず入力してください。 (秘密鍵を漏洩させないことが重要ですが、漏洩した場合に、なりすましを防止します。)

9.3.2. 公開鍵の登録

科学技術計算システムでは、公開鍵登録用の専用Webサイトで登録を行います。
保存した公開鍵(id_ed25519.pub)は、OpenSSH形式ですので、 `cat` コマンドなどで表示し、 コピー&ペーストして登録することができます。

リスト 6. catコマンドで公開鍵を表示

```
% cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy norin@pc-22007
```

パスワード認証を禁止しているサーバについては、それぞれのシステムで公開鍵登録の方法が異なりますので、管理者に確認してください。 (公開鍵をメールやファイルなどで管理者あてに送り、管理者が登録するといった方法もあります)

システムログイン端末による公開鍵の登録(パスワード認証が許可されている場合)

パスワード認証が許可されているシステムの場合は、次の方法で公開鍵を登録することが可能です。

- ターミナルなどから `ssh [login-id]@[server-name]` コマンドで接続し、パスワード認証でホストにログインします。
- テキストエディタや別のターミナルなどで表示(`% cat ~/.ssh/id_ed25519.pub`)した公開鍵をコピーして、 `.ssh/authorized_keys` に追加します。
- 端末上のコマンドは、以下のようにします。(一行で入力します)

```
[norin@fe01 ~]$ echo "ssh-ed25519
AAAAyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy norin@pc-22007" >>
~/.ssh/authorized_keys
```

  - `echo "` まで入力し、コピーした公開鍵をペーストします。
  - 続けて `" >> ~/.ssh/authorized_keys` と入力して、エンターキーを押す。
- `cat` コマンドで、登録された `authorized_keys` を確認します。

```
[norin@fe01 ~]$ cat ~/.ssh/authorized_keys
ssh-ed25519 AAAAyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy norin@pc-
22007
```

- ファイルパーミッション (アクセス権限) に注意してください。

```
[norin@fe01 ~]$ ls -l .ssh
合計 30
-rw------- 1 norin research  100  4月 13 11:04 authorized_keys
```

  - authorized_keys のアクセス権限が -rw------- の様になっていない場合は、 chmod 600 .ssh/authorized_keys としてください。

- 設定を確認したらログアウトしてください。

9.3.3. 公開鍵認証によるログインとファイル転送

コマンドによるSHH公開鍵認証によるログイン

リモートのシステムには、事前にローカル(PC)で作成した公開鍵を登録しておきます。(前項参照)

- ターミナルから ssh -i [秘密鍵のPATH] [login-id]@[server-name] コマンドで接続し、秘密鍵を開くためのパスフレーズを入力します。
(秘密鍵が標準のPATH(~/.ssh/id_ed25519 など)であれば、 -i [秘密鍵の場所] の指定を省略できます。)

```
% ssh -i ~/.ssh/id_ed25519 norin@scion.cc.affrc.go.jp
Enter passphrase for key '/Users/norin/.ssh/id_ed25519':   << パスフレーズ入力
Last login: Thu May 26 14:29:29 2022 from 150.26.201.116
[norin@fe03 ~]$
```

SSH公開鍵認証によるファイル転送(SCP)

scp コマンドによるファイル転送

- ローカル(PC)から、リモート(サーバ)に転送

```
% scp -i ~/.ssh/id_ed25519 amedas-kion.png norin@scion.cc.affrc.go.jp:unix-start/
Enter passphrase for key '/Users/norin/.ssh/id_ed25519':   << パスフレーズ入力
amedas-kion.png                                           100% 7116     1.0MB/s   00:00
```

- リモート(サーバ)から、ローカル(PC)に転送

```
% scp -i ~/.ssh/id_ed25519 norin@scion.cc.affrc.go.jp:unix-start/amedas-kion.png .
Enter passphrase for key '/Users/norin/.ssh/id_ed25519':   << パスフレーズ入力
amedas-kion.png                                           100% 7116     3.1MB/s   00:00
```

- リモート(サーバ)から、ローカル(PC)に転送(ディレクトリ配下)

```
% scp -i ~/.ssh/id_ed25519 -rp norin@scion.cc.affrc.go.jp:unix-start .
Enter passphrase for key '/Users/norin/.ssh/id_ed25519':   << パスフレーズ入力
```

```
hikisu.sh                                     100%   48    17.5KB/s   00:00
amedas-kion.sh                                100% 1458   572.5KB/s   00:00
script1.sh                                    100%   77    28.8KB/s   00:00
amedas-kion.png                               100% 7116     3.1MB/s   00:00
hello.c                                       100%   90    44.7KB/s   00:00
script2.sh                                    100%   82    34.0KB/s   00:00
hello.f90                                     100%   34    13.3KB/s   00:00
```

10. シェルスクリプト

ユーザがタイプしたコマンドライン操作は、シェルというプログラムで解釈され、実行されて結果を画面に表示します。

同じようなコマンド操作を繰り返したり、一連の操作を行いたいときに、その操作手順を登録しておくと便利です。 複数のコマンドが連続して動作するように、実行手順をあらかじめファイルに記述しておいたものが「シェルスクリプト」です。 また、シェルスクリプトは、変数や分岐処理などプログラミング言語としても利用できます。

スーパーコンピュータのような共同利用のシステムでは、計算機資源を各ユーザに割り当てるジョブスケジューラ(バッチジョブ管理システム)でプログラムを実行する際に、プログラムの実行環境や実行手順を記述したシェルスクリプトを使います。

10.1. シェルスクリプトの書き方と実行

ここでは、Bシェル系(sh, bash)のシェルスクリプトについて、説明します。

シェルスクリプトは、テキストファイルとして作成します。 例として、script1.sh という名前で、以下の内容のファイルをviなどのエディタを使って作成します。

リスト 7. script1.sh

```
#!/bin/sh
date
pwd
echo "I am $USER"
echo '4*a(1)' | bc -l
cal `date '+%Y'`
```

1行目の「 `#!/bin/sh` 」は、「 `#!` 」に続く部分のプログラムでスクリプトが実行されます。

2行目以降は、スクリプトで実行されるコマンドです。

```
date                    システムの日時を表示するコマンド
pwd                     カレントディレクトリを表示するコマンド
echo "I am $USER"       echo コマンドは、文字列を表示するコマンド
echo '4*a(1)' | bc -l   パイプやリダイレクションも使うことが出来ます
```

```
cal `date '+%Y'`        カレンダーを表示するコマンド
```

$USER は、ユーザ名が格納された環境変数です。

「 ` 」(バッククオート)を使うと、コマンドの実行結果を文字列として扱うことができます。 「 cal `date '+%Y'` 」では date コマンドの実行結果を cal コマンドの引数に指定しています。

- B シェル系(sh, bash) と C シェル系(csh, tcsh)では、書式が異なることがありますのでご注意願います。
- 環境変数は、シェルのプロセス及びその子プロセスで有効な変数です。 システムの利用環境として、 $USER や $PATH などの値がログイン時に設定されています。

10.1.1. シェルスクリプトの実行

シェルスクリプトを作成しても実行権限がないため、ファイル名を指定しても実行できません。

```
$ ./script1.sh
./script1.sh: Permission denied.
```

利用環境に依存しますが、 Permission denied. などのメッセージが表示され、スクリプトは実行できません。

実行権限を付与するには、chmodコマンドを使います。

```
$ ls -l script1.sh
-rw-r--r-- 1  norin  norin  87 2010-01-05 11:24 script1.sh
$ chmod u+x script1.sh
$ ls -l script1.sh
-rwxr--r-- 1  norin  norin  87 2010-01-05 11:24 script1.sh
```

chmodコマンドのオプション u(ユーザ)に対し +x(実行権を付加)することで、スクリプトを実行可能にします。

実行権限がないファイルでも、shコマンドの引数として指定すれば、シェルスクリプトが読み込まれて実行されます。

```
$ sh script1.sh
```

実行権限が付与されていれば、ファイル名を指定してシェルスクリプトを実行することができます。

```
$ ./script1.sh
Tue Jan 10 16:00:34 JST 2017
/home/norin
I am norin
```

```
3.14159265358979323844
                               2017

      January               February                March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7            1  2  3  4            1  2  3  4
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   5  6  7  8  9 10 11
15 16 17 18 19 20 21  12 13 14 15 16 17 18  12 13 14 15 16 17 18
22 23 24 25 26 27 28  19 20 21 22 23 24 25  19 20 21 22 23 24 25
29 30 31              26 27 28              26 27 28 29 30 31

-- 中略 --

      October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7            1  2  3  4                  1  2
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   3  4  5  6  7  8  9
15 16 17 18 19 20 21  12 13 14 15 16 17 18  10 11 12 13 14 15 16
22 23 24 25 26 27 28  19 20 21 22 23 24 25  17 18 19 20 21 22 23
29 30 31              26 27 28 29 30        24 25 26 27 28 29 30
                                            31
```

10.1.2. コメント

他のプログラミング言語と同様にシェルスクリプトでも、行頭に「#」を記述してコメントを入れることができます。

リスト 8. script2.sh

```
#!/bin/sh
# これはコメントです。
echo "これは表示されます。"
```

このスクリプトを実行すると、以下の様になります。

```
$ ./script2.sh
これは表示されます。
```

10.2. 変数

変数とは一時的に文字や数字などの値を入れて記憶します。変数を指定することで、値を記憶させたり、取り出して使います。

シェルの変数には、「シェル変数」と「環境変数」があります。

10.2.1. シェル変数

シェル変数は、実行中のプロセス(シェルやコマンド、シェルスクリプト)の内でのみ有効な変数です。

シェルスクリプトから起動するコマンドやスクリプトは、 [シェル変数] を参照することはできません。

変数に値を代入するには、以下の様にします。 値に何も指定しない場合は、変数は定義されますが、値は何もない(NULL)状態になります。

```
$ My_Name=Hattori
$ echo $My_Name
Hattori
$ My Name=
$ echo $My Name

$
```

10.2.2. 環境変数

環境変数は、シェルのプロセスおよびその子プロセスで有効な変数です。シェルから起動した他のコマンドやシェルスクリプトなどに変数を引き渡す場合は、環境変数を使う必要があります。

定義したシェル変数を環境変数に切り替えるには、exportコマンドを使います。

```
$ My_Name=Hattori
$ export My_Name
```

Cシェル系は、setenvコマンドを使います。 値に指定(書式)も異なりますので注意してください。

10.3. 引数

コマンドと同様に、シェルスクリプトの後にファイル名などを引数として指定することができます。

10.3.1. 位置パラメータ

シェルスクリプトの引数は、「位置パラメータ」と呼ばれる特殊な変数に代入されます。 この位置パラメータを参照することで、引数の値をシェルスクリプトで使うことができます。

位置パラメータは、「$0」「$1」「$2」「$3」……「$10」…… のように表され、「$0」には、実行したシェルスクリプト自身が代入され、「$1」以降に指定した引数が順に代入されます。

リスト 9. hikisu.sh

```
#!/bin/sh
echo "0=$0 1=$1 2=$2 3=$3 4=$4 5=$5"
```

このシェルスクリプトに引数を指定して実行すると、以下のようになります。

```
$ ./hikisu.sh hoge file 35 2000
0=./hikisu.sh 1=hoge 2=file 3=35 4=2000 5=
```

引数の5番目は指定が無いために、NULLとなっています。

10.3.2. 引数の数

引数の数は、「$#」という変数に代入されます。 引数を4個指定した上記の例で、hikisu.sh内で$#を参照すると、値は4となります。

10.4. サンプルスクリプト

簡単なサンプルスクリプトを例に、もう少し説明します。

この例では、気象（アメダス）データを取得し、グラフを作成する手順をスクリプトにしています。 実際には、欠測値の取り扱いなどを考慮する必要があります。

リスト 10. amedas-kion.sh

```
#!/bin/sh
CHITEN=40336    # 40336=つくば つくば市長峰　高層気象台
YEAR=2022       # 取得データの年
MONTH=04        # 取得データの月

FILE="AR$YEAR$MONTH.$CHITEN.csv"                        # リアルタイムアメダスのファイル名
DIR="/NDB/data/provide_data/amedas/real/$YEAR/$MONTH/"  # フロントエンドサーバのディレクトリ

# catコマンドで、gnuplotのコマンドを一時データファイルに出力
# nkfコマンドでUTF-8に変換(データファイルは、shift-jisなので)
# grep ^地点番号で、地点番号で始まるデータを取り出す(ヘッダ部分を除く)
# cutコマンドで、日時と気温のデータだけを取り出す
# trコマンドで、","(カンマ)をタブに変換して、一時データファイルに追加
# 一時データファイルにgnuplotのendコマンドを追加
# 上記の一連の結果を{ }でまとめて、amedas-kion.datファイルに出力
{
cat <<_EOT_
set term png
set output "amedas-kion.png"
set xdata time
set timefmt "%Y%m%d%H"
set title "AMeDAS"
set ylabel "Air temperature"
set xzeroaxis lt -1
set xlabel "Date"
set format x "%m/%d"
plot "-" using 1:2 with line ti "Chiten=$CHITEN"
_EOT_

nkf -w $DIR$FILE | grep ^$CHITEN | cut -d',' -f2,6 \
    | tr "," "\t"

echo "end"
```

```
} >>amedas-kion.dat

# gnuplotでグラフを作成する
gnuplot amedas-kion.dat

# 一時データファイルを削除
rm amedas-kion.dat
```

このサンプルを実行するには、 gnuplot などがインストールされている必要があります。

スクリプトの動作については、コメントに記述しています。 各コマンドの詳細は、マニュアルやインターネット上の情報を参照してください。

データファイル

科学技術計算システムのフロントエンドサーバでは、気象データなどの数値データが「/NDB」配下にあります。

gnuplot

gnuplot (ニュープロット)は、グラフを作成するためのフリーソフトウェアで、Linux, UNIX, Windows, macOSなど、多くのOSに対応したバージョンが開発されています。

10.4.1. ヒアドキュメント

このスクリプト内で gnuplot のコマンドを cat コマンドを使ってファイルに出力しています。 ここで << (二重の入力リダイレクション)記号を使っています。

```
cat <<_EOT_
set term png
set output "amedas-kion.png"
set xdata time
set timefmt "%Y%m%d%H"
set title "AMeDAS"
set ylabel "Air temperature"
set xzeroaxis lt -1
set xlabel "Date"
set format x "%m/%d"
plot "-" using 1:2 with line ti "Chiten=$CHITEN"
_EOT_
```

これをヒアドキュメントと言い、 <<_EOT_ から _EOT_ までの行を入力として取り込むことができます。

```
command <<終了文字
hoge hoge
foo bar
終了文字
```

10.4.2. コマンド中の改行

nkfから始まるデータ変換を行うコマンドは、パイプを使って幾つかのコマンドをつなげているため、長い1行のコマンドとなりますが、見やすくするために途中で改行しています。

次の行が続きの行であることを認識させるために、 \ (バックスラッシュ)を使います。

```
nkf -w $FILE | grep ^$CHITEN | cut -d',' -f2,6 \
    | tr "," "\t"
```

Windows系では、￥(円マーク)と表示されます。

10.4.3. 実行結果のグラフ

このスクリプトを実行して作成したグラフを以下に示します。

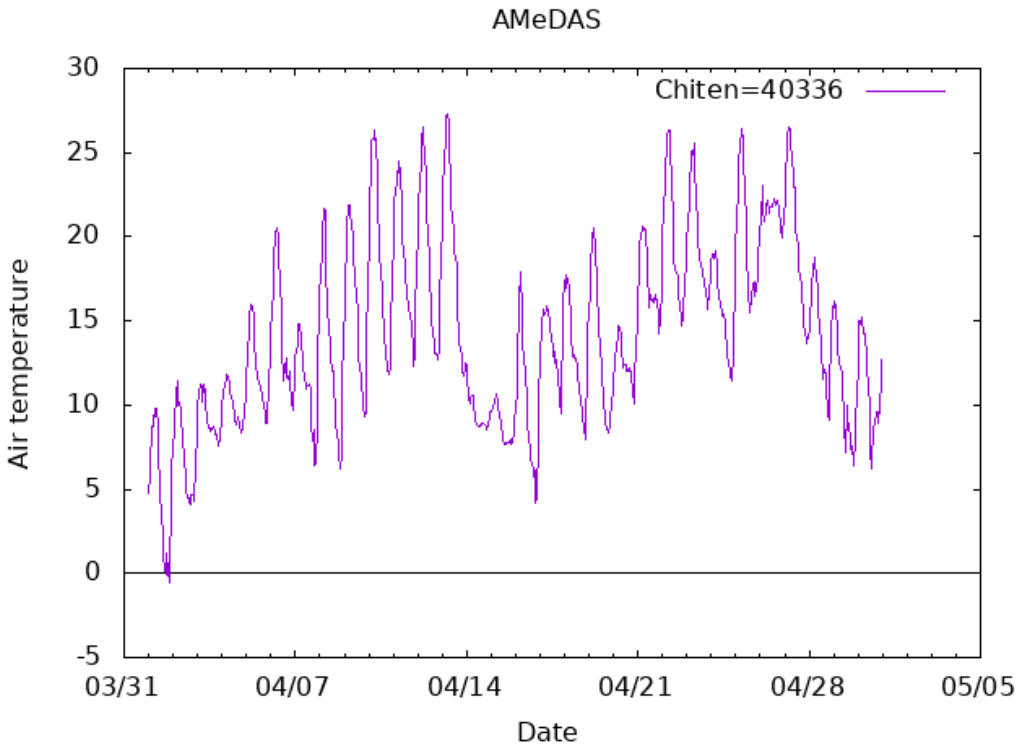

図 2. 気温データのグラフ

10.5. 参考

他のプログラミング言語と同様にシェルスクリプトには、 if, test, case などの条件判別や分岐処理、for, while などの繰り返しループ処理などがあり、高度なプログラムを作ることができます。 ここでは紹介できませんので、インターネット上の情報や書籍等を参考にしてください。

10.5.1. インターネット上の情報

- シェルスクリプト入門
  http://hocolamogg.com/unix/shellscript/index.html
- シェル・スクリプト・リファレンス INDEX
  http://itpro.nikkeibp.co.jp/article/COLUMN/20060224/230580/
- UNIX & Linux コマンド・シェルスクリプト リファレンス
  http://www.geocities.jp/geo_sunisland/

10.5.2. 書籍等

- 入門ＵＮＩＸシェルプログラミング 改訂第２版, ブルース・ブリン 著, 山下 哲典 訳, 2003-02-03, ISBN 4-7973-2194-6, ソフトバンククリエイティブ
- UNIXコマンドブック 第3版, 伊藤 真人、田谷 文彦、三澤 明 著, 2009-06-26, ISBN 978-4-7973-5278-8, ソフトバンククリエイティブ
- 仕事に使えるLinuxシェルスクリプト, 千葉 真人 著, 2004-11-29, ISBN 4-8222-8209-0, 日経BP社

11. プログラムの利用

シェルスクリプトや、Perl, Rubyなどは、ソースコードを逐次解釈しながら実行するため、エディタなどでプログラムを作成すれば、シェルスクリプトと同様に、すぐに実行することができます。 このようなプログラム(プログラミング言語)をインタープリタと言います。

これに対してC言語やFortranなどのコンパイラ方式は、ソースコードをコンピュータが実行できる形式(機械語、オブジェクトコード、バイナリーコードなどと言う)に翻訳してから、プログラムを実行します。 この翻訳を行うソフトウェアをコンパイラと呼び、翻訳することをコンパイルすると言います。

多くの場合、コンパイルされたプログラムの実行は、インタプリタよりも高速です。 しかし、プログラムを開発する際には、プログラムの変更、動作テストの度に、コンパイル作業が必要となります。

- ソースコード(Source Code)は、人が読み書きすることができるテキスト(文字列)で、一連の指示を記述したプログラムです。
- UNIX(Linux)は、C言語で書かれています。 また、UNIX系の多くのプログラム(コマンドなど)はC言語で書かれています。

11.1. コンパイラ

多くのLinuxディストリビューションには、無償で使える開発環境 `GCC(the GNU Compiler Collection)` が、パッケージとして導入できるようになっています。

- GCC環境は、標準ではインストールされていないことが多い。開発環境を追加イ

ンストールすることで利用可能となる。

- 科学技術計算システムには、GCCの他に、商用の開発環境として Intel の C, C++, Fortran コンパイラなどが導入されています。

GCC には、 C, C++, Objective-C, Fortran, Java と、それらのライブラリ群が含まれています。

ライブラリ: 汎用性の高い複数のプログラムを、再利用可能な形でまとめたもの。

CPUの種類(アーキテクチャ)によって、解釈する機械語(オブジェクトコード)が異なるため、同じプログラムを別のアーキテクチャのコンピュータで実行するためには、ソースコードからコンパイルし直す必要があります。

例えば、x86系Lnuxで作成したプログラムを、SPARC系Solarisなどのコンピュータで実行する場合は、再コンパイルが必要

11.2. プログラム作成と実行

FortranやC言語は、コンパイラ方式のプログラミング言語で、プログラム作成から実行までには、3つの手順があります。

1. ソースコード(ソースプログラム)の作成
   vi(vim)やEmacsなどのテキストエディタを用いて、FortranやC言語の書式を用いてプログラムを作成します。
   C言語のソースコードのファイル名は、「.c」の拡張子を付けるようにします。
   Fortranのソースコードのファイル名は、「.f90」「.f95」「.f」などの拡張子を付けることが一般的です。
   また、コンパイラによっては、これらの拡張子によってFortranの書式を判断する場合もあります。
2. コンパイル
   FortranやC言語で書かれたソースコードを計算機が実行できる形式(機械語, オブジェクトコード, バイナリーコードなどと言う)に変換します。
   C言語の標準的なコマンドは「cc」(c compiler)ですが、Linuxなどでは「gcc」へのシンボリックリンク(実態は「gcc」)となっていることが多く、説明では gcc を使います。
   Fortranについて、 GCC 4.0 からは、 Fortran90, Fortan95 に対応した「gfortran」が利用できます。
   コンパイル中にエラーが発生した場合は、手順1に戻りソースコードの不具合を修正してから再度コンパイルを行います。
3. コンパイル済みプログラムの実行
   コンパイルに成功すれば、実行形式のプログラム(バイナリーコード)が作成されるので、ファイル名を指定して実行します。
   プログラム実行中にエラーが発生したり、想定した動作をしない場合は、手順１に戻り、プログラムの修正を行います。

Fortranは、言語の規格が変更され Fortran77, Fortran90, Fortran95 などがあり、ソースコードファイルの拡張子やコンパイルオプションで、それぞれの書式に対応しています。 Fortran77は、構造化プログラミングができないなどの問題があります。古いプログラムをコンパイルするためのオプションと言えます。

この作業をエラーが無くなるまで繰り返します。これをデバッグ(Debug)と言います。

11.3. Cプログラムの作成と実行

C言語を使った簡単なプログラミングを説明します。

11.3.1. ソースコードの作成

viなどのエディタを使って、以下のプログラムを作成します。ファイル名は「hello.c」とします。

```
$ vi hello.c
```

リスト 11. hello.c

```
#include <stdio.h>
int main(void)
{
        printf("Hello World\n");
        return 0;
}
```

11.3.2. コンパイル

以下のコマンドでコンパイルします。

```
$ gcc -o hi hello.c -Wall
```

-o オプションは、出力する実行形式ファイルのファイル名です。 出力ファイル名を指定しない場合は、a.out というファイル名で実行形式ファイルが作成されます。

-Wall オプションは、コンパイル時のWarningを全て(all)出力します。

コンパイルが成功すれば、実行形式のファイルが作成されています。 実行権限も設定されています。

```
$ ls -l hi
-rwxr-xr-x 1   norin   norin   8298  2010-01-27  14:35 hi
```

11.3.3. 実行

実行形式ファイルのパスを指定して、プログラムを実行します。

```
$ ./hi
Hello World
```

このプログラムは、画面に「Hello World」と表示するというプログラムですので、正常に動作している ことが確認できます。

11.4. Fortranプログラムの作成と実行

C言語と同様の簡単なプログラムを作成して実行してみます。

11.4.1. ソースコードの作成

viなどのエディタを使って、以下のプログラムを作成します。ファイル名は「hello.f90」とします。

```
$ vi hello.f90
```

リスト 12. hello.f90

```
write(*,*)"Hello World"
stop
end
```

11.4.2. コンパイル

以下のコマンドでコンパイルします。

```
$ gfortran -o hi-f hello.f90 -Wall
```

-o オプションは、出力する実行形式ファイルのファイル名です。 出力ファイル名を指定しない場合は、a.out というファイル名で実行形式ファイルが作成されます。

-Wall オプションは、コンパイル時のWarningを全て(all)出力します。

コンパイルが成功すれば、実行形式のファイルが作成されています。 実行権限も設定されています。

```
$ ls -l hi-f
-rwxr-xr-x 1   norin   norin   8717  2010-01-27  15:21 hi-f
```

11.4.3. 実行

実行形式ファイルのパスを指定して、プログラムを実行します。

```
$ ./hi-f
Hello World
```

このプログラムは、画面に「Hello World」と表示するというプログラムですので、正常に動作している ことが確認できます。