第4章 アプリケーション

4.1 ログインと基本的なコマンド

4.1.1 ログインとログアウト

Linux システムを起動すると、何やらずらずらと表示されたあと最後に次のようなメッセージがコンソールに表示されます。

```
ARMA 3.0 on cosmos tty1
cosmos login:
```

最後にある login: は、ログインプロンプトと呼ばれ、ユーザがログインするのを待っている状態であることを表しています。ログインするには、まず自分のユーザ名を入力して［Enter］を押します。すると、続いてパスワードの入力を求めてくるので自分のパスワードを入力します。パスワードは覗き見されても大丈夫なように、入力した文字が画面に表示されません。正しく入力して［Enter］を押せばログインが完了し、Linux システムが使えるようになります。

```
cosmos login: junko
password:
Last login: Wed Aug 14 14:14:14 2009 on tty2

$
```

パスワードのチェックが終わると、自動的に初期設定が行われた後、シェルというプログラムが起動します。シェルは、ユーザがキーボードから打ち込んだコマンド（命令）を解釈して Linux に指示を与える「受付」のような役目のプログラムです。最後の $ はシェルがコマンドを「受付中」であることを表すしるしで、「コマンドプロンプト」と呼びます。

では、早速ですがひとつコマンドを使ってみましょう。…とは言っても、使うのは exit という、シェルを終了するためのコマンドです。

コマンドを入力した後、［Enter］を押すとコマンドが実行されます。

```
$ exit
exit

ARMA 3.0 on cosmos tty1
cosmos login:
```

ここでは、ログインしたシェルを終了してしまったので「ログアウト」となり、また次のログインを待っている状態に戻りました。

グラフィカルログイン（kdm）を使用している場合は、kdm のメニューからコンソールを選択するか、ctrl+alt+f1等でコンソールに切り替えることができます。

prompt :【動】うながして…させる ,…する気を起こさせる

4.1.2 コマンドを使う前に…ディレクトリツリーについて

さて、実際にどんどんコマンドを打ち込んで Linux を働かせる前に、前提知識として必要になる「ディレクトリツリー」について概略を説明しておきましょう。

Linux も他の多くの OS と同様に、情報を「ファイル」として管理し、ファイルを「ディレクトリ」という箱に入れて整理しています。そして、ディレクトリにはファイルだけでなく別のディレクトリを入れることもできます。こうしてできるディレクトリの階層構造によって、ARMA を構成する数万個ものファイルは見通しよく管理できるようになっています。

階層構造が多くのファイルを管理するのに相応しい理由は、多くの人や物を抱える組織が階層構造で管理されていることを考えれば明らかですね。

そして、このディレクトリの階層構造を図にしたものが「ディレクトリツリー」（以下「ツリー」）です。「ツリー」というのは、下のような図を左から見ると、ちょうど枝を生やした木（tree）のように見えることから付けられた名前です。

```
/ -+- bin
   +- home -+- harumi -+- hoge1.txt
   |        |          +- hoge2.txt
   |        |
   |        +- junko --+- hoge3.txt
   |        |          +- hoge4.txt
  （略）     （略）
```

root：【名】根。

上の図にもあるとおり、Linux ではツリーの根元は /（スラッシュ）で表された１つの「ルートディレクトリ」になっています。そして、全てのファイルは / から順にディレクトリをたどって「/ の home の harumi の中にある hoge1.txt」のように表せます。このファイルの「フルネーム」を「絶対パス」と言い、正式には「〜の」の代わりにルートと同じく / で区切りを表し、/home/harumi/hoge1.txt のように書きます。

さて、「絶対」があれば「相対」もあります。絶対パスは常に / を起点にファイルの位置を表すのに対し、相対パスは任意のディレクトリを起点にしてファイルの位置を表します。相対パスを表すに、起点になるディレクトリを .（ドット）、１階層上のディレクトリ（通称「親ディレクトリ」）を .. で表します。例えば /home/harumi を起点にすると、hoge1.txt は ./hoge1.txt 、hoge3.txt は ../junko/hoge3.txt と表せます。また、相対パスの起点になるディレクトリを「カレントディレクトリ」と言います。

もうひとつ、Linux にログインすると、まず /home/<ユーザ名> がカレントディレクトリになります。これをユーザの「ホームディレクトリ」といい、その中はユーザの個室のような扱いになり、ユーザは通常ホームディレクトリを起点に作業をします。例えば、harumi さんのホームディレクトリは /home/harumi です。

4.1.3 基本的なコマンド

最も基本的なコマンドは、「カレントディレクトリを移動する」「ファイルの一覧を見る」「ファイルをコピーする」という、ファイルやディレクトリを操作するコマンドです。そこで、ここではこれらの基本コマンドを具体的に紹介しつつ、Linux とコマンドに慣れていきましょう。

4.1.4 ファイルに関する情報を取得する（ls）

ファイル操作をするうえで最も重要な情報は、ファイルの名前やサイズなどファイルのプロフィールとも言える情報です。これらを知るには ls を使います。もう少し具体的に言えば、「ディレクトリの中に何があるの？」という質問に答えるのが ls です。

```
$ ls [< オプション >] [< ファイル ( 複数可 )>]
```

まず、何のオプションも付けずに ls を実行すると、カレントディレクトリの中身を一覧形式で表示します。

```
$ ls
hoge1.txt  hoge2.txt
```

これに、以下のようなオプションを指定すると、ls の動作が変わります。

-a	. で始まるファイル名も表示
-d	ディレクトリの中身ではなく、ディレクトリ自身を表示
-l	各ディレクトリ・ファイルの詳細な情報を表示
-R	ディレクトリの中身を下位のディレクトリも含めて全て表示
-F	ディレクトリに / 、実行可能ファイルに * を付ける
-t	新しい順にファイルを並べる
-S	サイズの大きい順にファイルを並べる
-r	逆順に並べる（-r -t は古い順、-r -S は小さい順となる）
--color	カラー表示

例えば、オプション -a を指定する場合は、次のようにします。

```
$ ls -a
.  ..  .bash_history  .bash_profile  .bashrc  hoge1.txt  hoge2.txt
```

Linux では通常「隠しファイル」として扱う、名前が . で始まるファイル名も表示されます。カレントディレクトリ、親ディレクトリを表す . や .. も . で始まる名前ですので表示されています。

-d については、付けた場合と付けない場合の実際の例を比較して見てみましょう。通常、ls のパラメータにディレクトリ名を渡すとその中身を表示しますが、-d オプションを付けるとディレクトリ自体を表示します。

```
$ ls /home/harumi
hoge1.txt  hoge2.txt

$ ls -d /home/harumi
harumi
```

-l オプションを付けると、パーミッション（後述の chmod の項を参照）・参照数・所有者・グループ・ファイルサイズ・最終更新日時・ファイル名などの詳細なファイルに関する情報を表示します。

```
$ ls -l /home/harumi
-rw-r--r--    1 harumi   users        1234 2009-03-04 05:06 hoge1.txt
-rw-r--r--    1 harumi   users        8765 2009-04-06 08:10 hoge2.txt
```

--color では、通常ファイルは白、実行可能ファイルは緑、ディレクトリは青、デバイスファイルは黄、シンボリックリンクは水色で表示されます。

ざっと Linux のコマンドはこのように使います。それでは、これからは少しスピードを上げてコマンドを紹介していきましょう。

4.1.5 カレントディレクトリの移動（cd,pwd）

cd は、カレントディレクトリを移動するコマンドです。

```
$ cd [<ディレクトリ>]
```

ディレクトリは絶対パスでも相対パスでも指定できます。また、ディレクトリを指定しないとホームディレクトリにカレントディレクトリが移ります。

さて、pwd はカレントディレクトリの名前を確認するコマンドです。

```
$ pwd
```

ただし、普通はコマンドプロンプトと一緒にカレントディレクトリ名が表示されているので、pwd の出番はないかもしれません。例えば、下の例ではコマンドプロンプトと合わせてログインしているマシン名とカレントディレクトリを表示しています。

```
cosmos:/home/junko$
```

4.1.6 ファイルの複写（コピー）・移動・削除（cp,mv,rm）

ファイルの基本操作は複写（cp）・移動（mv）・削除（rm）です。これらのコマンドには、パラメータとしてファイル名だけでなくディレクトリ名も指定でき、ディレクトリを中身ごと操作することもできます。それではまず cp から…

```
$ cp [< オプション >] < コピー元 ( 複数可 )> < コピー先 >
```

cp の主なオプションは以下の通りです。何もオプションを指定しない場合は、コピー先に同名のファイルがあっても問答無用で上書きされ、コピー先のファイルの所有者・グループはコピーを行ったユーザ・グループになり、タイムスタンプはコピーを行った時間に設定されます。

-i　コピー先に同名のファイルがある場合上書きするかどうか尋ねる
-d　シンボリックリンクを保ったままコピー
-p　コピー元と同じ所有者・グループ・タイムスタンプをコピー先にも適用
-R　ディレクトリの中身を丸ごとコピー
-a　-dpR と同じ働きをする

つづいて、mv の説明に移りましょう。

```
$ mv [< オプション > < 移動元 ( 複数可 )> < 移動先 >
```

mv の主なオプションは以下の通りです。

-i　移動先に同名のファイルがある場合上書きするかどうか尋ねる

最後に rm です。rm したファイルは二度と復活できないので、[Enter] を押す前には確認を怠らないようにしてください。

```
$ rm [< オプション >] < ファイル ( 複数可 )>
```

-i　本当に削除してよいか尋ねる
-f　問答無用でファイルを削除
-r　ディレクトリの中身ごと全て削除

4.1.7 ディレクトリの作成と削除（mkdir,rmdir）

ディレクトリを作成（mkdir），削除（rmdir）します。ただし、rmdir は空のディレクトリしか削除できないコマンドで、中にファイルやディレクトリが入っているディレクトリを削除するには rm -r しなければなりません。

```
$ mkdir [<オプション>] <ディレクトリ（複数可）>
$ rmdir [<オプション>] <ディレクトリ（複数可）>
```

mkdir, rmdir はどちらも -p というオプションが使えます。

-p　多階層のディレクトリを一気に作成または削除します。

通常 mkdir, rmdir で２階層以上のディレクトリを扱う場合は、１階層ずつコマンドを実行しなければなりませんが、-p ではこれを一気に済ませることができます。以下の例では、同じ dir1/dir2 というディレクトリを作って消す操作を -p オプションを付けた場合と付けない場合で実行しています。

```
$ mkdir dir1
$ mkdir dir1/dir2
$ rmdir dir1/dir2
$ rmdir dir1
```

```
$ mkdir -p dir1/dir2
$ rmdir -p dir1/dir2
```

4.1.8 パーミッション

Linux は１つのシステムに複数のユーザがログインできる「マルチユーザ」システムです。そのため、ファイルやディレクトリの１つ１つに「誰のものか」「誰がアクセスできるか」が決められ、勝手に他人のファイルにアクセスできないようになっています。

Linux では、この「誰のものか」を所有者と所有グループ、「誰がアクセスできるか」を「パーミッション」で設定します。パーミッションは「ファイルの所有者・グループ・その他のユーザ」の３区分それぞれに「読み・書き・実行」を認めるかどうかを設定でき、ls -l で確認できます。

```
-rw-r--r--    1 harumi   users        1234 2009-03-04 05:06 hoge1.txt
```

このファイルは、所有者 harumi、所有グループ users で、パーミッションは -rw-r--r-- の最初１文字を除いた後ろ９文字で表しています。9 文字の内

訳は3区分（所有者・グループ・その他）×3文字（読み・書き・実行）です。

ファイルを読み込める（r）・書き込める（w）・実行できる（x）権限の有無は、権限が有れば r,w,x の文字、無ければ - で表しています。すると、例の hoge1.txt は次のようなパーミッションであることが分かります。

最初の3文字（所有者）	rw-	＝読み書き可能
次の3文字（所有グループ）	r--	＝読み込みのみ可能
最後の3文字（その他のユーザ）	r--	＝読み込みのみ可能

なお、パーミッションはファイルだけでなくディレクトリにも存在し、それぞれ次のような意味を持っています。

- r　ディレクトリの中身の存在が分かる
- w　ディレクトリの中身を変更できる
- x　ディレクトリに入れる

ディレクトリに x 権限があると、そこをカレントディレクトリにして中にあるファイルの「内容」を読み書きできるようになります。これに対し w 権限では、「ディレクトリの中身を変更」でき、中にあるファイルそのものを作成・複写・移動・削除することはできますが、ファイルの内容を読めるわけではありません。

ところで、-rw-r--r-- の先頭1文字はファイルの種類を以下のような文字で表しています。

- -　通常のファイル
- d　ディレクトリ
- l　シンボリックリンク
- b　ブロック型デバイスファイル
- c　キャラクタ型デバイスファイル
- p　名前つきパイプ

この他に、誰でも所有者・所有グループと同じ権限でファイルを実行できる setuid, setgid という特殊なパーミッションがあります。例えば、root 所有のファイルに setuid を立てると一般ユーザも「そのファイルに限り」root 権限でファイルを実行できるようになります。

ls -l では、setuid は -rwsr-xr-x、setgid は -rwxr-sr-x(s の位置に注目！)のように表示されます。ただし、所有者に x 権限がないファイルに setuid を立てると、「所有者は実行できない。他のユーザは所有者に倣う。」という状態になり、もともとの「所有者と同じ権限で『実行できる』」意味がなくなってしまいます。この場合は、無意味な設定であることを強調するため s ではなく大文字の S と表示されます。これは setgid についても同様です。

「ファイルを実行できる」とは、ファイルをプログラムと見なして起動できることです。

不用意に root 所有のファイルに setuid
立てるとセキュリティホールになる可能性があります。そのような設定は十分に慎重におこなってください。

もうひとつ、sticky というパーミッションがあります。これは主に、/tmp のように誰でも書き込める共用ディレクトリで勝手に他人のファイルを消去できないようにするもので、sticky を立てるとそのディレクトリ内のファイルはディレクトリの所有者とファイル自身の所有者にしか消去できなくなります。

sticky は ls -l では -rwxrwxrwt(t の位置に注目!)のように表示されます。ただし、その他のユーザに x 権限がないと sticky が無意味になってしまいますが、この場合は setuid, setgid と同じように大文字で T と表示されます。

4.1.9 所有者・所有グループの変更(chown,chgrp)

ファイルの所有者を変更するコマンドは chown です。ただし、ファイルの所有者を変えるには変更前のファイルに w 権限が必要になるほか、一般ユーザは所有者を自分名義にしか変更できません。

```
$ chown [-R] <ユーザ>[:<グループ>] <ファイル(複数可)>
```

-R オプションをつけると、指定したディレクトリの中身も全部同じように chown します。

また、ユーザ名に続けて : とグループ名も指定すれば、所有者とグループの両方をいっぺんに変更できます。

グループだけを変更するには chgrp コマンドを使います。

```
$ chgrp [-R] <グループ> <ファイル(複数可)>
```

-R オプションの意味は chown のときと同じです。

4.1.10 パーミッションの変更(chmod)

パーミッションを変更するのは chmod コマンドです。

```
$ chmod <対象(複数可)>{+|-|=}<権限(複数可)> <ファイル(複数可)>
```

対象には以下の選択肢があります。ug のように複数を指定しても構いません。

- u 所有者
- g 所有グループ
- o その他のユーザ
- a 全員(ugo と指定するのと同じ)

引き続き、+,-,= のいずれかを指定します。

- + 権限を与える
- - 権限を剥奪する

= 権限を指定したものだけにする

権限には先ほど説明した r,w,x,s,t が使えます。rw のように複数指定しても構いません。

以上をまとめると、例えば次のように使います。

```
$ chmod u-w file1                （所有者がファイルに書き込めないようにする）
$ chmod u+s file2                （setuid を立てる）
$ chmod u=rwx go=rx file3        （ファイルのパーミッションを設定し直す）
```

chmod には、これとは別にパーミッションを数値で指定する方法もあります。

```
$ chmod <パーミッション> <ファイル（複数可）>
```

パーミッションを表す数値は、例の rwxrwxrwx を 400,200,100,40,20,10,4,2,1 として、それらの和で指定します。

setuid, setgid, sticky は 4000,2000,1000 で表します。

例えば rwsr-xr-x ならば 4000+400+200+100+40+10+4+1 = 4755、rw-r--r-- なら 400+200+40+4 = 644 となります。

4.2 シェル

シェルは一言でいうと、ユーザが入力したコマンドを解釈して Linux に指示を出すプログラムです。ただこれだけが分かっていても一通りは Linux を使えますが、この節ではもう少しステップアップして、十分に Linux を使うためのシェルの使いこなし法を紹介していきたいと思います。

シェルがシェルと呼ばれるのは、「OSのシェル」＝「殻」で、よく言われる「OSカーネル」＝「核」とペアで OS を構成するものです。つまり、シェルはカーネルの周囲にある OS の殻で、OS の外側にいるユーザと内側にいるカーネルの仲立ちをするようなイメージです。

ARMA はその名の通りカーネルに Linux を採用した OS ですので、ARMA のシェルの役目は初めに書いたように「ユーザが入力したコマンドを解釈して『Linux カーネルに』指示を出す」ことになるというふうです。

4.2.1 ARMA / OGL の標準シェル = bash

ARMA にはいくつかのシェルが収録されていますが、その中でも標準的なのは bash で、ログイン時にも bash が起動します。bash = Bourne Again SHell は GNU が開発したシェルで、その名の通り Steven Bourne 氏が開発した元祖 UNIX シェルこと sh に様々な改良を加えたものです。以降の説明は、全てこの bash の機能についての説明です。

sh 系 = B シェル系に対して、ARMA では csh 系 = C シェル系のシェルとして tcsh も配布しています。

4.2.2 コマンドライン編集機能

bash では、コマンドを打ち込んでいる途中 Enter キーを押すまでの間なら、そのコマンドライン（行）を以下のようなキーで編集することができます。以下の表では、例えば Ctrl+a は［Ctrl］を押しながら［a］を押すことを表しています。

Ctrl+a, Ctrl+e	行頭 / 行末に移動
Ctrl+b, Ctrl+f	1 文字戻る / 進む（← , → キーも同様）
Alt+b, Alt+f	1 単語戻る / 進む
Ctrl+d	カーソルがある 1 文字を削除（Del キーも同様）
Ctrl+h	カーソルの左の 1 文字を削除（Backspace キーも同様）
Ctrl+k	カーソル以降の文字列を全て削除
Ctrl+p, Ctrl+n	1 つ前 / 後のコマンドラインを呼び出す（↑ , ↓ キーも同様）
Ctrl+r	以前に入力したコマンドラインを検索する

Ctrl+r を入力するとコマンドプロンプトが一旦消えて、以前に入力したコマンドラインを検索するモードに入ります。ここに以前使ったコマンドラインの一部を入力すると、検索結果の候補が表示されます。候補が複数あるときは、さらに Ctrl+r を押すと他の候補を見られます。ここで表示されている候補をそのまま使うときは［Enter］を押し、コマンドラインを編集して使うときは先ほど紹介した編集キーを打ち込みます。

```
$
```

↓ Ctrl+r に続けて cp と入力する

```
(reverse-i-search)‘cp’: cp hoge1.txt /tmp
```

4.2.3 コマンドライン補完

突然ですが、百人一首の「決まり字」を御存知でしょうか？ 例えば、「たご」から始まる歌は「たごのうらに にうちいでてみれば しろたへの ふじのたかねに ゆきはふりつつ」しかないために「たご…」と読まれた時点で札を取るという、アレです。

bash には、この「決まり字」のような「コマンドライン補完」という機能があり、途中までコマンドやファイル名を入力して［Tab］を押すと残りを補完してくれるようになっています。例えば、dpkg-buildpackage のような長い名前のコマンドの場合、dpkg-b まで入力して［Tab］を押すと、dpkg-b から始まるコマンドは dpkg-buildpackage だけですので、シェルが uildpackage を補完してくれます。

```
$ dpkg-b
```

↓［Tab］

```
$ dpkg-buildpackge
```

百人一首では「決まり字」より前に札を取りに行くのはお手つき覚悟ですが、bash ならお手つきしても大丈夫です。先の例で dpkg までで［Tab］を押してもまだ「決まり字」ではないので何も補完されませんが、さらにもう一回［Tab］を押すと dpkg から始まるコマンドを一覧表示してくれます。

```
$ dpkg
```

↓［Tab］［Tab］

```
$ dpkg
dpkg                  dpkg-genchanges       dpkg-scansources
dpkg-architecture     dpkg-gencontrol       dpkg-shlibdeps
dpkg-buildpackage     dpkg-name             dpkg-source
dpkg-checkbuilddeps   dpkg-parsechangelog   dpkg-split
dpkg-deb              dpkg-preconfigure     dpkg-statoverride
dpkg-distaddfile      dpkg-reconfigure
dpkg-divert           dpkg-scanpackages
$ dpkg
```

コマンドライン補完を使うコツは「決まり字」を覚えようとしないことです。途中まで適当に入力したら［Tab］を押し、もしまだ「決まり字」でなかったらもう少し入力するという姿勢の方が楽にコマンドライン補完機能と付き合えます。

ちなみにコマンド名だけでなく、パラメータに与えるファイル名も同じ

原理で補完ができます。

4.2.4 ファイル名展開

bash では、ファイル名を指定するときにワイルドカードと呼ぶ特殊文字を利用して、複数のファイルをまとめて指定することができます。

- `?` 任意の１文字に置き換えられる
- `*` 任意の文字列に置き換えられる（文字列は０文字でも可）

例えば、ディレクトリの中に `1` 〜 `100` までの100個のファイルがあるとき、`1?` は10番台のファイルを、`1*` は `1` から始まるファイル全てを指定したことになります。

```
$ ls 1?
10  11  12  13  14  15  16  17  18  19

$ ls 1*
1  10  11  12  13  14  15  16  17  18  19  100
```

この他、ワイルドカードとは違いますが、`~<ユーザ名>` をユーザのホームディレクトリに置換してくれる機能もあります。例えば `~harumi` は harumi のホームディレクトリ `/home/harumi` を表します。また、自分のホームディレクトリは `~` で表せます。

4.2.5 リダイレクト

Linux では通常、入力機器はキーボード、出力機器はディスプレイです。このキーボードとディスプレイのセットをターミナルといい、「標準入出力はターミナルである」といいます。bash では `<` で標準入力を、`>` で標準出力をそれぞれ任意のファイルに変更できます。この仕組みを「リダイレクト」といいます。

以下の例では `ls` の出力を `result` というファイルにリダイレクトしています。

```
$ ls > result
$ cat result
hoge1.txt  hoge2.txt
```

4.2.6 パイプ

あるコマンドの標準出力を、別のコマンドの標準入力に渡すことができます。この機能をパイプと呼びます。

以下の例では ls の出力結果から、grep で文字列 foo を検索しています。

```
$ ls | grep foo
```

4.2.7 コマンドの連続実行

コマンドを ; で区切ると、複数のコマンドを1つのコマンドラインで連続して実行できます。

以下の例ではカレントディレクトリを /tmp に変更し、続けて ls を実行します。

```
$ cd /tmp ; ls
```

4.2.8 バッククォート

‘（バッククォート）でコマンド名を囲むと、そのコマンドの出力を別のコマンドのコマンドラインの一部として使えます。

以下の例では pwd の結果を ls のパラメータとして使っています。

```
$ ls ‘pwd‘
hoge1.txt  hoge2.txt
```

ただし、ls はパラメータを指定しない場合はカレントディレクトリのファイル名一覧を表示するので、このコマンドラインは少しナンセンスです。

4.2.9 文字のエスケープ

ここまでに出てきた、特殊文字を「単なる文字」として扱いたい場合、その文字の前に ¥（バックスラッシュ）を置きます。これを「エスケープする」と言います。

以下の例では ¥ によって > が単なる文字になってリダイレクトがなくなり、a > test という文字列が画面に表示されます。

```
$ echo a > test
$ cat test
a
$ echo a ¥> test
a > test
```

4.2.10 文字列のクォーティング

文字列全体を単なる文字列として扱いたい場合、文字列全体を '(クォート)で囲みます。このことを文字列をクォーティングすると言います。

以下の例では a > test という文字列をクォートして、中にある > という特殊文字も丸ごと普通の文字として扱っています。

```
$ echo 'a > test'
a > text
```

4.2.11 環境変数

環境変数とは、プログラムの「実行環境」を決めるために設定する情報のことです。これだけでは分かりにくいので、代表的な環境変数を紹介しましょう。

EDITOR	標準として使うテキストエディタの名前
LANG	使用している言語
PAGER	標準として使うページャ(テキストビューア)の名前
PATH	入力されたコマンドを探すディレクトリのリスト

例えば、LANG は数多くのコマンドが参照し、設定値が C なら英語、ja_JP.eucJP なら日本語でメッセージを表示するなどの判断のために使います。

この環境変数を設定・参照するコマンドは export です。

```
$ export [< 変数名 >=< 値 >]
```

パラメータを何も設定しない場合は、現在設定されている環境変数を確認できます。

4.2.12 bash の設定ファイル

bash の設定ファイルは以下の3つです。これらはシェルスクリプトという、コマンドラインをたくさん並べたものに若干の制御構造を加えた一種のプログラム形式で記録されています。シェルスクリプトの詳細は man bash などを参照して頂くとして、ここでは各設定ファイルの役割だけを示しておきます。

~/.bash_profile	ログインした時に読み込まれる
~/.bashrc	bash を起動した時に読み込まれる
~/.bash_logout	ログアウトした時に読み込まれる

4.3 テキスト処理ツール

4.3.1 テキストを扱うということ

UNIX で最も基本的なファイル形式は「テキストファイル」です。UNIX は設計段階からテキストファイルをなるべく用いるという思想で設計されており、/etc の設定ファイル、/usr/share のドキュメント類、/var/log のログなど、UNIX システムの情報の多くはテキストファイルで与えられます。さらに、スクリプト・メール・プログラミング・TeX…と、UNIX におけるテキストの重要性は強調してもし過ぎることはありません。

このように重要なテキストファイルですから、UNIX ではテキストを取り扱うツールが非常に充実しています。これらのツールひとつひとつは単純な機能しか持っていませんが、引数にファイル名を指定しなければ「標準入力を読んで処理結果を標準出力に書き出す」、いわゆる「フィルタ」としても働くため、|（パイプ）でツールを繋げば複雑なテキスト処理もできるようになっています。このパイプを使ったテキスト処理というのは、最も「UNIX 的」な作業ですので是非会得したいところです。それでは、基本的なテキスト処理ツールを順に説明していきましょう。

ツールとは、単純な機能で応用が利くという文字通りの「工具」のようなソフトウェアです。

4.3.2 テキストファイルを見る（lv, less / jless）

lv は多国語化されたテキストビューアです。多国語化と銘打つだけあって英語や日本語だけでなく、中国語や韓国語などのテキストも正しく扱えます。

その場合は gnome-terminal のような多国語に対応した端末を使って下さい。

```
$ lv <ファイル（複数可）>
```

[↑][↓] だけでもテキストを読めますが、以下のショートカットキーをいくつか覚えればさらに快適になります。

f,b	次 / 前のページ（PageDown,PageUp で代用可）
u,d	次 / 前の半ページ（→ , ← で代用可）
k,j	次 / 前の行（vi 風）
>,<	ファイルの末尾 / 先頭へ移動（指定行への移動もできる）
/,?	続けて入力した文字列をテキストの末尾 / 先頭へ向かって検索
n,N	直前に検索した文字列をテキストの末尾 / 先頭へ向かって再検索
:n,:p	次 / 前のファイルに移動
=	ファイル名・行番号・文字コードなどファイルの情報を表示
q	lv の終了

>,< コマンドで指定した行へ移動するには、コマンドの前に行数の数値を入れます。例えば、300> とすると 300 行目に移動できます。

/,? コマンドで検索するときは、コマンドに続けて検索したい文字列を入力し最後に Enter を押します。検索する文字列には正規表現も使えます。less とその日本語対応版 jless は、歴史と認知度では lv を上回りますが、国際化などの面で lv の方が優れているため、ARMA では lv を標準テキストビューアに採用しています。しかし、元々 lv は less の操作感を参考に作られたツールなので、lv と less の操作はほぼ同じになっています。

正規表現についてはここでは詳しく説明できませんが man 7 regex に詳しく書かれていますので、そちらを参照して下さい。正規表現はいわばワイルドカード(*)の超高機能版です。

4.3.3 先頭・末尾を切り出す(head,tail)

head, tail はテキストファイルの先頭・末尾を指定した行数だけ表示します。これらを使うと先頭・末尾部分を確認するためだけに、いちいち lv を立ち上げないで済みます。

```
$ head [-c <バイト数> | -n <行数>]      [<ファイル>]
$ tail [-c <バイト数> | -n <行数>] [-f] [<ファイル>]
```

オプションを指定しない場合はテキストファイルの先頭・末尾の10行を表示しますが、以下のオプションによって表示する範囲を変えられます。

-c 表示する部分をバイト単位で指定(数値にk,mを付けるとKB,MB単位)
-n 表示する部分を行数で指定

tail -f では一定の間隔でファイルの末尾を監視して、末尾に追加された部分を随時表示していくため、内容が刻々と追加されていくログなどを監視するときなどに便利です。

```
# tail -f /var/log/messages
```

4.3.4 文字列を検索する(grep, zgrep, bzgrep, lgrep)

grep はテキストから正規表現を検索して、それが見つかった行を表示するツールです。

```
$ grep [<オプション>] <正規表現> [<ファイル(複数可)>]
```

grep という名前は sed で「正規表現を検索して、それが見つかった行を表示する」という意味のコマンドの g / RE / p(RE = Regular Expression = 正規表現)に由来します。

オプションを指定しない場合は単純に正規表現を検索し、見つかった(ヒットした)行だけを表示しますが、以下のようなオプションによって動作を変えられます。

-A <n> ヒットした行の後 n 行も表示
-B <n> ヒットした行の前 n 行も表示
-C <n> ヒットした行の前後 n 行も表示
-v ヒットしなかった行を表示

-i	英語の大小文字を区別せず検索
-H, -h	出力の各行にファイル名を出力する / しない
-n	出力の各行に行番号を表示
-r	ディレクトリの中のファイルを全検索

zgrep, bzgrep は内部的に gzip, bzip2 と grep を呼び出すスクリプトで、圧縮された *.gz や *.bz2 形式のテキストファイルを検索できるようになっています。

また、lgrep は lv の検索機能を呼び出すもので、grep 互換のオプションは -v, -n 程度しかありませんが、文字コードに関係なく日本語の検索を正しく行えます。

4.3.5 辞書順・数値順に行を並べ替える（sort）

sort はテキストを行単位で文字コード順に並べて表示するツールです。米語の文字コード（ASCII）は A ～ Z, a ～ z の順に並んでいるので sort を使うと、ちょうど辞書（ABC）順に整列できます。ただし、日本語の場合は sort すると漢字コード順になりますが、漢字コードは五十音順ではありません。

```
$ sort [< オプション >] [< ファイル ( 複数可 )>]
```

オプションを指定しない場合は行単位で文字コード順、もう少しラフに言うと「ABC 順」に並べますが、以下のようなオプションによって動作を変えられます。

-b	行頭の半角空白を無視
-f	英語の大小文字を区別しない
-n,-g	数値順に整列（-g は浮動小数点なども扱えるが遅い）
-r	逆順に整列
-k <m>[,<n>]	タブや空白区切りの m ～ n 番目を比較対象にする

-n の「数値順ソート」は、0 ～ 9 を単純に文字コードで比較するのではなく、特別にひとかたまりの数値として比較します。例を挙げて見てみましょう。

```
$ cat n_sort
100
20

$ sort n_sort
100
20

$ sort -n n_sort
20
```

```
100
```

100 と 20 を普通にソートすると、文字コードでは 1 の方が 2 より前なので、100 → 20 の順になります。しかし、数値順ソートでは 100 と 20 を数値として認識し、小さい順に並べるので 20 → 100 になります。さらに、-g では -f で数値と認識できない 1.23e45 のような浮動小数点や +6.78 のようなプラス符号も扱えますが、動作は -n よりもだいぶ遅くなってしまいます。

-k では、タブや空白文字で区切られた区切りを「フィールド」として、フィールド単位でのソートを行うオプションです。まずは、ひとつ実例を見てみましょう。

```
$ cat k_sort
2 100
1 3
3 20

$ cat -k1,1 k_sort
1 3
2 100
3 20

$ cat -k2,2 k_sort
2 100
3 20
1 3

$ cat -k2n,2 k_sort
1 3
3 20
2 100
```

この例では、空白で区切られた各数値がフィールドになります。-k1,1 では各行の第１フィールドだけを見て文字コード順にソートしています。これに対し、-k2,2 では第２フィールドがソート対象になっています。さらに、-k2n,2 とフィールドに n を付けると数値順ソートをします。

また、フィールド内で、ソート対象を文字単位で指定することもできます。例えば、第１番目のフィールドの第２文字から第３文字までをソート対象にするなら -k1.2,1.3 と指定します。

4.3.6 重複行をまとめる（uniq）

uniq は辞書順にソートされたテキストから連続する同一内容の行をまとめて、2回あっても3回あっても1回だけ表示するツールです。従って、パイプで uniq を使うときは、前に必ず sort を通しておかなければなりません。

```
$ uniq [< オプション >] [< ファイル >]
```

以下のオプションによって、動作を変えられます。

-f <n>　先頭の n フィールド を飛ばして uniq
-s <m>　先頭の m 文字を飛ばして uniq
-i　英語の大小文字を区別しない
-c　出力の各行頭に表れた回数を表示する
-u　1回だけ表れる行のみ表示する
-d　2回以上表れる行のみを1回だけ表示する

-f,-s を両方指定したときは、まず n フィールド飛ばし、さらに次のフィールドの先頭から m 文字を飛ばして uniq します。

フィールドの意味は sort の項で紹介したものと同じです。

4.3.7 バイト数・単語数・行数を数える（wc）

wc はテキストファイルのバイト数・単語数・行数を表示するツールです。

ただし、単語数は「分かち書き」されたものを数えますので、日本語のように単語を分かち書きをしない言語では正しく単語を数えられません。また、日本語ではバイト数は文字数と一致しません。いわゆる「半角文字」が1文字1バイトで「全角文字」は1文字2バイトであるのはもちろん、UTF や ISO-2022-JP ではエスケープシーケンスもあるので単純にバイト数から文字数は計算できません。よって、日本語テキストファイルを wc にかけるとき、実質的な意味がある数値は行数だけと言えます。

英語では ``This is a pen.'' のように単語の間を空けて書きます。wc はこの間の空白文字を見て単語の数を数えています。

```
$ wc [< オプション >] [< ファイル >]
```

オプションを指定しない場合はバイト数・単語数・行数を全て表示しますが、以下のようなオプションによって動作を変えられます。

-c　バイト数のみ出力
-w　単語数のみ出力
-l　行数のみ出力
-L　最長の行のバイト数を出力

-L は他と少し旗色の違うオプションですが、メールなどで1行の長さを制限しなければならない場合などの確認に便利です。

4.3.8 差分行を表示する（diff）

diff は行単位で2つのファイルを比較し、差がある行を出力するツールです。

```
$ diff [< オプション >] < ファイル1> < ファイル2>
```

オプションを指定しない場合は単純に2つのファイルの差分を表示しますが、以下のようなオプションによって動作を変えられます。

-b	スペースの数を区別しない
-w	スペースの数と有無を区別しない
-B	空行を無視
-i	英語の大小文字を区別しない
-r	ディレクトリの中身を全比較する
-N	両方にファイルがあるように装う
-c	context 形式で出力
-C <n>	context 形式の「文脈行」を n 行にする
-u	unified 形式で出力する
-U <n>	unified 形式の「文脈行」を n 行にする

-b では、例えば ho ge（スペース×1）と ho　ge（スペース×2）のようなスペースの数だけが違う行を同じものとみなします。-w では、さらに hoge（スペース×0）のようなスペースを含まないものも同一視します。

-N は、主に -r でディレクトリの中身を丸ごと比較する場合に合わせて使うオプションで、片側にしかないファイルも、もう片方では空のファイルがあるかのように装います。実際の例を見て、説明に代えましょう。ここでは、dir1/x-file はあるが、dir2/x-file はないものとします。

```
$ diff -r dir1 dir2
Only in dir1/: x-file

$ diff -rN dir1 dir2
diff -rN dir1/x-file dir2/x-file
1c1
< This is dir1/x-file.
```

-N を指定していると dir2/x-file という空のファイルがあると見なして diff の結果が示されています。

さて、以下のような2つのファイルを実際に diff してみて、context，unified などの各出力形式について説明します。

```
$ cat hoge1
1行目
2行目（hoge1の2行目）
3行目

$ cat hoge2
1行目
2行目（hoge2の2行目）
3行目
4行目
```

出力形式を指定しなかった場合は、古典的な出力形式になります。

```
$ diff hoge1 hoge2
2c2
< 2行目（hoge1の2行目）
---
> 2行目（hoge2の2行目）
3a4
> 4行目
```

2c2 とは、差分が hoge1 の2行目と hoge2 の2行目であることを意味します。そして、< マーカで始まる行に hoge1 の、> マーカで始まる行に hoge2 の内容がそれぞれ出力されています。

context 形式は、差分行の前後数行を「文脈行」として出力します。文脈行は、diff の出力が人間にとって見やすくなるだけでなく、patch の動作する助けにもなります。

```
$ diff -c hoge1 hoge2
*** hoge1       Mon Jun 25 17:00:08 2001
--- hoge2       Mon Jun 25 16:59:59 2001
***************
*** 1,3 ****
  1行目
! 2行目（hoge1の2行目）
  3行目
--- 1,4 ----
  1行目
! 2行目（hoge2の2行目）
  3行目
+ 4行目
```

ここでは出てきませんが - マーカは file1 にしかない行を表します。

*** から始まる行は hoge1 に、--- から始まる行は hoge2 に関するものです。例えば *** 1,3 **** は続く出力が hoge1 の１～３行目であることを表しています。また、! マーカは両ファイルで内容が異なる行を、+ マーカは file2 にしかない行を表しています。- と + のマーカは diff を「hoge1 から hoge2 への変化」と考えれば、「消滅した（-）行」「出現した（+）行」と直観的に覚えられると思います。

さて、unified 形式も文脈行を表示するのですが、context 形式のように２つのファイルの内容を交互に書かずにまとめて出力します。

```
$ diff -u hoge1 hoge2
--- hoge1       Mon Jun 25 17:00:08 2001
+++ hoge2       Mon Jun 25 16:59:59 2001
@@ -1,3 +1,4 @@
 1行目
-2行目（hoge1の2行目）
+2行目（hoge2の2行目）
 3行目
+4行目
```

@@ -1,3 +1,4 @@ は file1 の１～３行目と file2 の１～４行目の差分であることを意味します。- と + のマーカの意味は context 形式と同様です。

4.3.9 文字コードの変換（lv, nkf）

日本語には ISO-2022-JP, EUC-JP, ShiftJIS の３種類の文字コードがあります。これらのコードは「メール用」「UNIX 標準」「Windows 標準」とそれぞれにそれなりの役割があるので、普通に ARMA を使うだけでも３種類の文字コードそれぞれで書かれた日本語テキストに接することになるでしょう。これらを相互に変換するのが文字コード変換ツールの役割です。

日本語コードの変換も lv が便利です。

```
$ lv [-k] [-I<変換前文字コード>] -O<変換後文字コード> <ファイル>
```

-I に続いて変換前の、-O に続いて変換後のテキストファイルの文字コードを指定します。変換前のテキストの文字コードはテキストファイルの内容から自動的に判断されるので、普通は -I を指定する必要はありません。文字コードは次のリストの形式で指定します。

ej	EUC-JP（日本語 EUC）
j	ISO-2022-JP（JIS コード）
s	Shift JIS（Microsoft 漢字コード）
l1 ～ l9	ISO-8859-1 ～ ISO8859-9
u7	UTF-7
u8	UTF-8

-k はJISX0201のいわゆる「半角カナ」をJISX0208のいわゆる「全角カナ」に変換する機能です。インターネット上では半角カナを使わないように推奨されています。

lv の他、日本語専用ですが現在に至るまで広く使われている文字コード変換ツールとして、nkf があります。

```
$ nkf [{-e}|{-j}|{-s}] [< ファイル >]
```

-e, -j, -s は、それぞれ変換後の文字コードを EUC-JP, ISO-2022-JP, Shift JIS にすることを意味します。

4.4 テキストエディタ

4.4.1 vi

vi は非常に軽快、かつ多機能なフルスクリーンエディタです。vi の操作性は現在主流となっているエディタとは一線を画し、少々取っつきにくいと思われる面もあるかもしれませんがリモートログイン時などにも活躍します。ここでは vi の初歩を説明します。

vi の起動は vi に続けて編集したいファイルの名前を指定するだけです。既に存在するファイルを指定した場合は、そのファイルが開かれます。

```
$ vi < ファイル >
```

ここでは、新しいファイルを指定して vi を起動してみます。vi を起動すると、次のようなシンプルな画面になります。

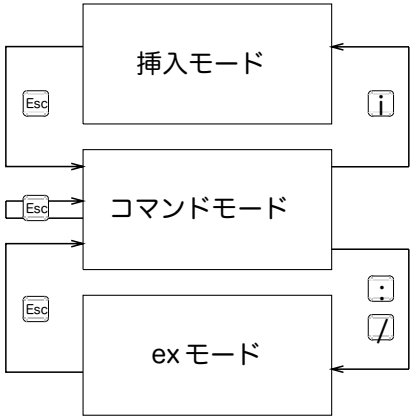
```
~
~
~
~
~
hoge.txt: new file: line 1
```

さて、vi にはさまざまなモード（状態）があり、このモードを切り替えながらテキストファイルを編集していくようになっています。特に重要なモードを以下に挙げます。

挿入モード	テキストを編集できる（入力モードとも言う）
コマンドモード	操作用のコマンドを実行できる（編集モードとも言う）
ex モード	ファイル保存、vi の終了などの命令を受け付ける

vi を起動した直後はコマンドモードになっています。

モード間の移動は、下の図のようなキー操作でできます。挿入モードと ex モードの間は直接移動できないので、［Esc］を押して一旦コマンドモードを経由してから移動してください。自分がどのモードにいるのか分からなくなった場合にも、取りあえず［Esc］でコマンドモードに移ればミスを避けられます。



挿入モードには、コマンドモードで［i］を押すと入れます。すると、画面下に -INSERT- と表示され、自由に文字を入力することができるようになります。

```
~
~
~
~
~
-- INSERT --
```

コマンドモードには、先程も紹介したように、各モードで［Esc］を押すと入れます。このモードでは、ユーザは特定の機能（キー）が割り当てられたキーをタイプすることで vi を操作できます。

まず、カーソルを移動するコマンドは以下の通りです。Ctrl+d は［Ctrl］を押しながら［d］を押すことを表しています。

h, j, k, l	カーソルを左 / 下 / 上 / 右に移動（← , ↓ , ↑ , → でも代用可）
0, $	カーソルを行頭 / 行末に移動

nG, m|	カーソルを n 行目 / m 桁目に移動（n,m には数値を指定）
Ctrl+u, Ctrl+d	半画面上 / 下にスクロール
Ctrl+f, Ctrl+b	1 画面上 / 下にスクロール

続いて、テキストの一部分を切り貼りして編集するためのコマンドは以下の通りです。vi は直前に削除・コピーした文字列を「編集バッファ」と呼ばれる一時的な記憶領域に保存していて、他の場所に貼りつけることができます。

x	カーソルがある文字を削除
X	カーソルの左の文字を削除
dd	カーソルがある行を削除
yy	カーソル行をコピー
p	バッファの内容をカーソルの次の行に貼付
P	バッファの内容をカーソルの前の行に貼付
u	直前の操作を取り消す（アンドゥ）
ZZ	編集内容をファイルに保存して vi を終了

ex モードには、コマンドモードで [/] か [:]（コロン）を押すと入れます。すると、入力に応じて画面左下に / か：というプロンプトがあらわれます。このプロンプトは、bash の $ というコマンドプロンプトと同じように、ex モードが次のコマンドを受付中であることを示しています。さて、実際に使えるコマンドを紹介しましょう。

まず、文字列を検索するには、/ のプロンプトに検索したい文字列を入力し、[Enter] を押します。すると該当する文字列にカーソルが移動します。その後、[n] を押せば同じ文字列を順方向（ファイルの末尾に進む方向）に、[N] を押せば逆方向（ファイルの先頭に戻る方向）に再検索ができます。

ファイルへ保存したり、vi を終了したりする操作は、ex モードでもできます。: のプロンプトに続けて以下のキーを入力してください。

q!	編集内容を破棄して、vi を終了
w	編集内容をファイルに保存する
q	vi を終了

4.4.2 Emacs

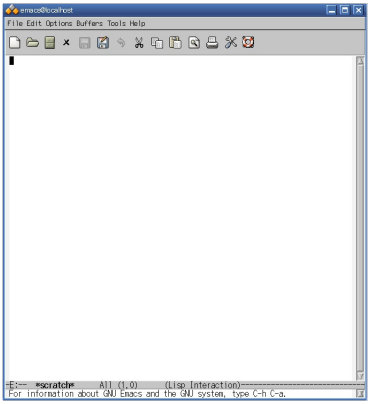
Emacs は非常に強力なテキストエディタです。同じ Emacs 系エディタとしては GNU Emacs の他に XEmacs、古くは Mule などが有名ですが、ARMA では Emacs を標準として想定しています。よって、以下の説明は全て Emacs が対象になりますが、Emacs 系エディタには互換性があるので、他の Emacs 系エディタでも大体は参考になるものと思います。

Emacs は単なるテキストエディタの範疇越える様々な機能を備えています。Emacs 上では Emacs Lisp という言語で書かれたプログラムが動くため、Emacs Lisp で書かれたメーラやニュースリーダ、果ては WWW ブラウザなどを用意すれば、およそテキストを扱う作業は全て Emacs の上で行うことができます。そのため Emacs は「環境」と見なされることもあります。

ここで Emacs について詳説することはできませんので、以下では単純にテキストファイルを読み書きするために必要な機能だけを要約したいと思います。

それでは、まず Emacs を起動しましょう。

```
$ emacs [-nw] [<ファイル(複数可)>]
```



Emacs にはウィンドウを開く GUI モードとコンソール上で動く CUI モードがあります。X Window System 上で単純に `emacs` を起動すると GUI モードになりますが、`-nw` を付けて起動すると CUI モードになります。X Window System 上ではメニューインターフェースもありますが、下記ではキーを使う方法について説明します。

前提知識として Emacs 系エディタで一般的なキーバインドの表記法について触れておきます。Emacs 系エディタでは [Ctrl] と同時にキーを押す操作を `C-` で、同様に [Alt] と同時にキーを押す操作を `M-` で表します。例え

と表します。M-x は、[Alt] と [x] の同時押しは

さて、Emacsは見かけは普通のテキストエディタで、文字を入力したり [Enter] で改行したり、カーソルを矢印キーで移動したりすることは非常に直観的にできます。各種編集機能を [Ctrl] や [Alt] と英字キーを同時に押すことで呼び出すのも普通のプログラムと同じです。この、機能とキー操作の対応関係を「キーバインド」と呼びます。では、このキーバインドを機能の系統ごとに説明していきましょう。

まず紹介するのは、キー操作をキャンセルする C-g です。キー操作の途中で分からなくなったり、コマンドの間違いに気づいたとき、C-g を押すと１手順戻ることができます。何回か押せば、画面の最下行に Quit が出て、コマンドに入る前の編集状態に戻ります。

しかし、C-g はコマンド操作の途中から１手順戻るキーなので、コマンドが確定してしまった後はもう戻れません。このときは、C-x u で直前の操作を取り消し（アンドゥ）ます。まずこの２つを覚えておけば、どんなに間違えても慌てなくて済みます。

それでは、カーソルを動かすキー操作から紹介していきましょう。

C-b	１文字戻る（←）
C-f	１文字進む（→）
M-b	１単語戻る
M-f	１単語進む
C-a	行頭に戻る
C-e	行末に進む
C-p	１行戻る（↑）
C-n	１行進む（↓）
C-v	１画面分戻る
M-v	１画面分進む
M-textless	ファイルの先頭に戻る
M-textgreater	ファイルの末尾に進む

次は、文字や文字列を削除するコマンドです。なお、よく他のソフトウェアで使われる C-h でバックスペースという操作は標準では対応していません。C-h ではヘルプシステムが起動します。

del	カーソル位置の１文字を削除（C-d も同様）
bs	カーソル位置の左の１文字を削除
C-k	カーソル位置から行末までの文字列を削除

次は、いわゆる「カット＆ペースト」「コピー＆ペースト」の操作です。Emacs では、最初に C-space でカーソル位置に「マーク」と言う、カットやコピーの原点の印をつけます。そこからカーソルをカットやコピーしたい範囲（リージョン）の終点まで動かし、C-w や M-w でリージョンをカットまたはコピーします。また、C-y を押せば、カット or コピーした内容がペースト

[Ctrl] + [Space]（設定によっては [Shift] + [Space] を押すと日本語が入力できるようになります。

カットは「切取」、コピーは「複写」、ペーストは「貼付」のことです。

されます。

C-space	マークを設定 / 解除
C-w	リージョンの内容をバッファにカット
M-w	リージョンの内容をバッファにコピー
C-y	バッファの内容をカーソル位置にペースト

続いては文字列の検索と置換です。文字列をファイルの末尾に向かって検索するときは、まず C-s を押して、検索モードに入ります。すると、画面の最下行に I-search: という文字列が表れます。

```
-E:%% *GNU Emacs*    ALL  L1   (Fundamental Isearch) ------
I-search:
```

ここに続けて検索したい文字列を入力していくと、1 文字入力するごとに候補が絞られていきます。複数の候補があるときは C-s を押すと次の候補へカーソルが動きます。目的の候補まで来たら［Enter］でそこへカーソルが移動して、検索は終了です。

文字列の置換は M-% で、まず置換モードに入ります。

```
-E:%% *GNU Emacs*    ALL  L1   (Fundamental Isearch) ------
Query replace:
```

ここに続けて置換前の文字列を入力して［Enter］を押すと、置換後の文字列を尋ねてくるので入力します。

```
-E:%% *GNU Emacs*    ALL  L1   (Fundamental Isearch) ------
Query replace: windows with: ogl
```

続けて［Enter］を押すと、置換対象が見つかるたびに置換するかどうか聞いてきます。答えは 4 通りあり、［y］は置換して次へ、［n］は置換しないで次へ、［q］は置換作業を終了、［!］ではこれ以降質問なしで全て置換になります。

それでは、検索と置換をまとめておきましょう。

C-s	文字列を順方向（ファイルの末尾に進む方向）に検索
C-r	文字列を逆方向（ファイルの先頭に戻る方向）に検索
M-%	文字列を置換

さて、Emacs では、ファイルの編集作業はバッファと呼ばれる一時的な記憶領域を使用して行われます。ファイルを開くとファイルの内容が新しいバッファに読み込まれます。Emacs では複数のバッファを切り替えたり、画面を分割して複数のバッファを同時に表示したりできます。以下はこのバッ

ファを操作するキー操作です。

C-x C-b	バッファの一覧を表示
C-x b	編集しているバッファを切り替え
C-x k	編集しているバッファを削除
C-x 2	バッファウィンドウを分割
C-x 1	バッファウィンドウを１つにする
C-x o	バッファウィンドウ間でカーソルを移動

最後はファイル操作です。Emacs を終了する前には、必ずファイルを保存しておきましょう。お疲れさまでした。

C-x C-c	Emacs を終了
C-x C-f	新しいファイルを読み込む
C-x C-s	バッファをファイルに保存
C-x C-w	バッファを別のファイルに保存

4.5 WWW ブラウザ

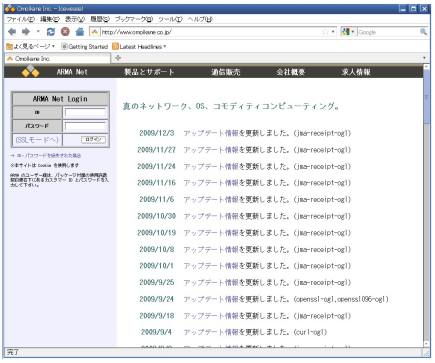
4.5.1 Iceweasel（Firefox）、Konqueror

近年では「インターネットする」と言えば「ホームページを見る」ことだと言われるようになるほど、WWW（World Wide Web）は身近な存在になりました。

ARMA はいくつか WWW ブラウザ（閲覧ソフトウェア）を収録していますが、そのなかでメインになるものは Iceweasel です。Icewasel は Mozilla Firfox をベースにした WWW ブラウザで、WWW ページを極めて正確に表現することができます。Icweasel を起動するには、X Window System 上で以下のコマンドを使うか、あるいはメニューバーの「Iceweasel」アイコンをクリックします。

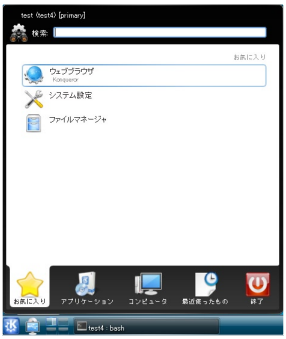

```
$ iceweasel &
```

すると、このようなウィンドウが開きます。



Konqueror を起動する場合も同様で KDE でメニューから「お気に入り」-「ウェブブラウザ」をクリックするか、以下のようにコマンドを実行します。

商標上の問題で Debian 系システムでは Firefox は Iceweaesl と呼ばれています。ソフトウェアとしては完全に同一です。




```
$ konqueror &
```

WWW ページを開くには、停止ボタンの右にあるボックスに URL（Uniform Resource Locator）という、いわゆる「アドレス」を入力します。例えば http://www.omoikane.co.jp/ と入力すればオモイカネの WWW トップページが表示されます。Konqueror についても同様に起動できますので、あとは自由に WWW サーフィンをお楽しみください。

4.6 電子メール

4.6.1 電子メール

ここでは、Linux で電子メール（以下「メール」）を利用して、メッセージの送受信を行う方法を説明します。

メールを扱うソフトウェアには、大きく分けて MUA（Mail User Agent）と MTA（Mail Transfer Agent）があります。このうち、前者が一般に「メーラ」（Mailer）と呼ばれる、ユーザがメールを受信して保存したり自分で書いたメールを送信したりするなどのメールの管理を行うソフトウェアです。ここでは、まず代表的な MUA の使い方を説明していきます。ちなみに、後者は「メールサーバ」とも呼ばれるメールの配信に使うソフトウェアで、この節では説明しません。

さて、MUA が受信したメールをハードディスク上に保存する「メールボックス」の形式には、大きく分けて次の３つがよく使われています。MUA を使うときには、その MUA がどの形式に対応しているかを確認する必要があります。

（1） Maildir 形式 : 仕組みは MH 形式と同じだが、ディレクトリやファイルの命名法が MH 形式とは違う。`~/Maildir` を使う。

（2） mbox 形式 : 全てのメールを `~/mbox` という１つのファイルに保存する。

（3） MH 形式 : メール１通を `~/Mail` 以下の１つのファイルとして管理する。

4.6.2 メールの受信とメールボックスへの振り分け（fetchmail + procmail）

一般的な MUA には自分でメールを受信する機能があります。しかし、細かい設定や自動受信などのために、MUA とは別のメール受信ツールでメールを受信し、MUA ではそれを管理するだけという使い方もできます。ここで紹介する fetchmail と procmail は、そんなメール受信専門のツールです。

では、Maildir 形式を例に fetchmail と procmail でメールを受信する方法を説明します。その前に、まずホームディレクトリにメールボックスを作成しましょう。

```
$ maildirmake.dovecot ~/Maildir
```

fetchmail は IMAP,POP3 に対応した、サーバからメールを受信するソフトウェアです。下記では IMAP サーバ imap.omoikane.co.jp から受信する例になります。アカウントは foo、パスワードは ******** としています。

`maildirmake.dovecot` は dovecot-common パッケージに含まれています。courier-imap をお使いの場合は `maildirmake.courier` コマンドで作成してください。ディレクトリとしては同じものが作成されます。

```
poll imap.omoikane.co.jp
proto IMAP
username foo
password ********
mda "/usr/bin/procmail"
```

poll 行にはメールを受信する相手のサーバを、protocol 行には POP3 か IMAP で受信に使うプロトコルを、username 行と password 行にはそれぞれメール受信のためのユーザ名とパスワードを指定します。上記以外のオプションについては、man fetchmail を参照してください。

また、次のように ~/.fetchmailrc のパーミッションを設定しておきます。

```
$ chmod 710 ~/.fetchmailrc
```

procmail は、受信したメールを自動的にメールボックスに振り分けるプログラムです。procmail の設定は ~/.procmailrc に記述します。

```
PATH=/bin:/usr/bin:/usr/local/bin
MAILDIR=$HOME/Maildir
LOGFILE=$MAILDIR/from
LOCKFILE=$HOME/.lockmail
DEFAULT=$MAILDIR/new
```

DEFAULT 行に、受信したメールを保存するディレクトリ（MH 形式か Maildir 形式の場合）やファイル（mbox 形式の場合）を指定します。

以上の設定で fetchmail コマンドを実行しますと、Maildir ディレクトリ以下にメールが配送されます。

4.6.3 mutt

mutt はキャラクタ端末上で動作する以下のような特長を持った MUA です。

- 代表的なメールボックス形式に対応しているので他の MUA からの移行が簡単。
- メールが増えても動作が遅くならない。
- キャラクタベースで軽快に動作するため、出先から細い回線で SSH で自宅のマシンにログインして mutt を起動しても快適にメールを使える。

このような特長のため、mutt は主にキャラクタ端末に慣れたユーザーを中心に好まれています。MUA の性能的に見ても、POP3 と IMAP4 両対応、複数のメールボックスの管理に対応など十分な機能があります。

mutt は単体でもメールを受信できますが、ここでは mutt 本体のメール受信機能を使わず汎用的に fetchmail と procmail を使ってメールを受信する方法を使っています。メールの受信についてはこの文章の fetchmail と procmail の項か、mutt で直接メールを受信するなら `man mutt` をご参照ください。

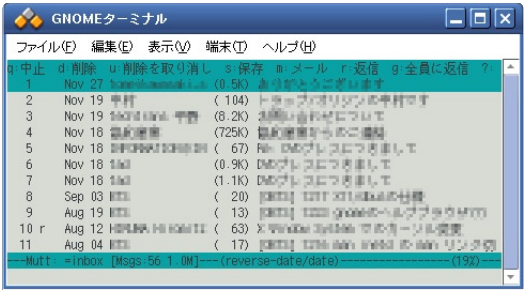
mutt の設定ファイルには `~/.muttrc` の雛形がありますので、次のようにしてホームディレクトリにコピーしておきます。

```
$ cp /usr/share/doc/mutt/examples/sample.muttrc ~/.muttrc
```

mutt を起動するには、コマンドライン上で以下のようにタイプします。

```
$ mutt
```

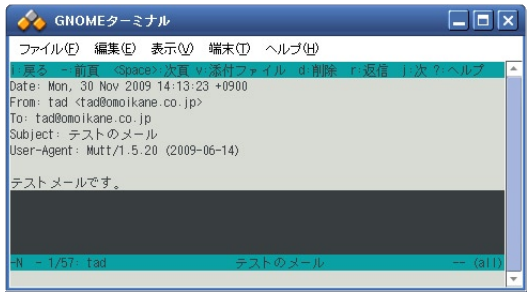
mutt を起動すると以下のようなインデックス画面になります。



インデックス画面での代表的なキー操作は以下の通りです。

d, u	メッセージを削除 / 削除を取り消す
q, x	現状を保存して / 保存せずに終了
Enter	メッセージを表示
v	添付ファイルを参照
/	検索
m	新しいメールを作成
r, g	メールの送信者宛 / 受信者全員宛に返信
f	メールを転送

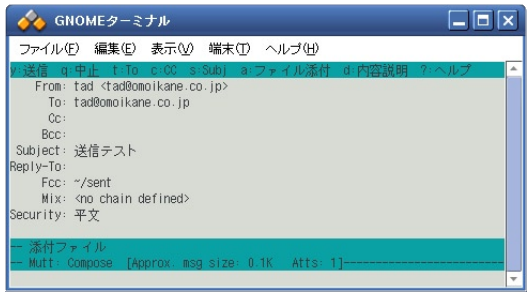
インデックス画面で [Enter] を押すとメッセージの表示画面になります。



ここでは以下のキー操作が使えます。

Enter, BackSpace	次 / 前の行に移動
Space, -	次 / 前のページに移動
n	次のメッセージを表示

インデックス画面で［m］を押すと、メール送信モードに入ります。ここではメールの宛先に続いてメールの本文を編集します。この編集には環境変数 EDITOR に設定されているテキストエディタが使われます。メール本文の編集が終了すると、メール送信画面が表示され、次のキー操作をおこなうことができます。



a	ファイルを添付
T, c, b	To / Cc / Bcc: を修正
s	Subject: を修正
y, q	メールを送信 / 送信中断

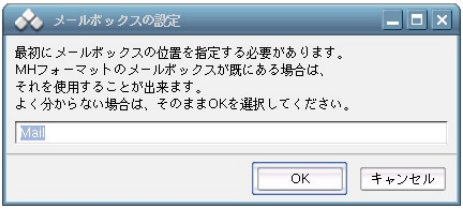
メールを送信したり送信を中断したりした後はインデックス画面に戻ります。

4.6.4 Sylpheed

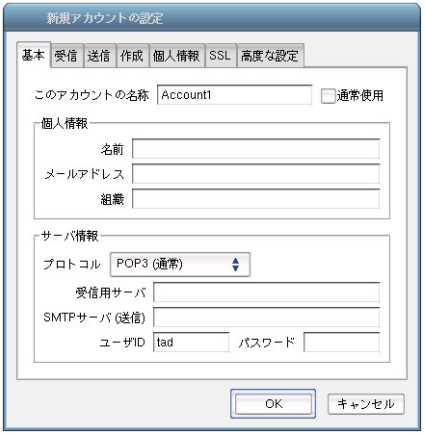
Sylpheed は X Window System 上で動作する、容易な設定と直感的な操作が可能な美しく洗練されたインタフェースの MUA です。メールボックスの形式としては、一般的な MH 形式に対応しています。それでは、早速 Sylpheed を起動してみましょう。

```
$ sylpheed
```

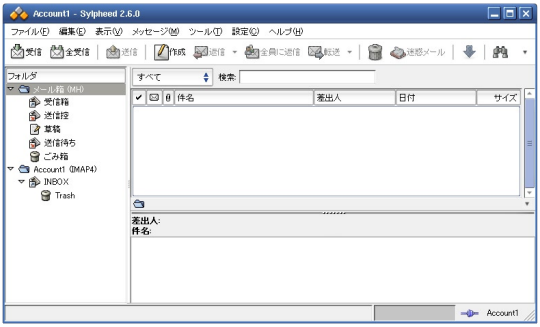
初回起動時には、自動的に設定ウィンドウが開きます。まずはメールボックスのディレクトリを作ります。MH 形式の標準は `~/Mail` です。



続いてアカウントの設定を行います。標準的なメールサーバを使う場合の設定は、全てこの「基本」画面の中で全て終わりです。特に分かりにくい設定項目はないと思いますが、所属組織を書く「組織」については書かなくても構いません。



さて、設定が終わるとメインウィンドウに戻ります。



メールの受信は左上の「受信」アイコンをクリックします。受信が終わると、左にはメッセージの入ったディレクトリツリー、右上は開いているディレクトリの中のメッセージの一覧、右下はメッセージの内容が表示されます。また、メールの送信・返信・転送なども全てウィンドウの上側に並んだアイコンをクリックするだけで、作成ダイアログが開きますので、ウィンドウに従って操作してください。

4.7 ダウンロードコマンド

ここではコマンドレベルのファイルダウンロードの方法について説明します。

4.7.1 wget

wget は HTTP や FTP サーバからファイルをダウンロードして、主に WWW, FTP サイトをミラーリングするために使うツールです。FTP サイトのミラーリングはディレクトリの丸ごとダウンロードなので NcFTP などでも同様のことができますが、WWW サイトのミラーリングは少々やっかいです。wget では、WWW サイトのミラーリングをするために、HTML のアンカータグ (<A></A>) を理解して、リンクを辿ってファイルをダウンロードできるようになっています。

wget には大量のオプションがありますが、ここではよく使われるものだけを機能別に分類して説明していくことにします。

```
$ wget [< オプション >] <URL ( 複数可 )>
```

ログ関係

```
-q              ログを表示しない
[-v|-nv]        ログを詳細 / 簡潔に表示
```

quiet={on|off}
verbose={on|off}

接続関係

```
-t <n>                          接続失敗とするまで n 回接続を試みる
-T <m>                          接続失敗とするまで m 秒間応答を待つ
-Y {on|off}                     プロキシを通すかどうか
--proxy-user=< ユーザ >         プロキシサーバのユーザ名
--proxy-password=< パスワード > プロキシサーバのパスワード
--http-user=< ユーザ >          WWW サーバのユーザ名
--http-password=< パスワード >  WWW サーバのパスワード
--referer=<URL>                 WWW サーバに伝える Referer URL
-nr                             .listing を削除しない
--passive-ftp                   パッシブモードで FTP サーバに接続
```

tries=<n>
timeout=<n>
proxy={on|off}
proxy_user=< ユーザ >
proxy_passwd=< パスワード >
http_user=< ユーザ >
http_passwd=< パスワード >
passive_ftp={on|off}

ダウンロード関係

```
-c    ダウンロード途中のファイルを続きからダウンロード
-N    サーバのファイルにある新しいファイルがあればダウンロード
-nH   ホスト名を付けたディレクトリを作成しない
```

continue={on|off}
timestamping={on|off}
add_hostdir={on|off}

-nH を指定しないと、wget はホスト名を付けたディレクトリを作り、その中にファイルをダウンロードします。例えば、http://www.omoikane.co.jp/index.html は ./www.omoikane.co.jp/index.html にダウンロードされます。しかし、-nH を指定するとこのディレクトリを作らず、./index.html にダウンロードされます。

再帰下降ダウンロード関係

-r	再帰下降ダウンロードをする
-l <n>	ダウンロードするリンク・ディレクトリを n 階層に制限
-m	ミラーリングに最適（-r -N -l inf -nr と等価）
-k	絶対リンクを相対リンクに変換
-A|-R< リスト >	ダウンロードする / しない拡張子を "," 区切りで指定
-I|-X< リスト >	ダウンロードする / しないディレクトリを "," 区切りで指定
-H	他のホストへのリンクもたどる
--follow-ftp	FTP サイトへのリンクもたどる

recursive={on|off}
reclevel=<n>
mirror={on|off}
convert_link={on|off}
accept=< リスト >
reject=< リスト >
include_directories =< リスト >
exclude_directories =< リスト >
span_hosts={on|off}
follow_ftp={on|off}

再帰下降ダウンロードは、HTTP ならリンクを、FTP ならディレクトリをたどってサイトの中身を丸ごとダウンロードするという意味です。

-l オプションで、無制限に再帰下降する場合は -l inf と指定します。なお、-l を指定しない場合は5階層までダウンロードします。

-k は、例えば、http://www.omoikane.co.jp/sayuri/hoge.html という HTML ファイルの <A_HREF="/junko/test.html"> というリンクを wget が <A_HREF="../junko/test.html"> と書き換える機能です。この変換により、ミラーリングしたファイルをブラウザで見ても正しくリンクをたどれます。

wget は、設定を ~/.wgetrc に保存できます。オプションのうち、コマンドラインでも ~/.wgetrc でも指定できるものは既に注釈で説明しましたが、以下の項目は ~/.wgetrc でのみ設定できるものです。

```
login=< ユーザ >
passwd=< パスワード >

http_proxy=< サーバ >
ftp_proxy=< サーバ >
```

login と passwd では FTP サーバのユーザ名とパスワードを指定します。

特にこの設定をしない場合は、FTP サイトにはユーザ名 anonymous 、パスワード < 自分のユーザ名 >@< 自分のドメイン名 > で接続します。http_proxy と ftp_proxy では HTTP / FTP 用のプロキシサーバのホスト名とポート番号を cosmos:8080 のように指定します。特に設定しない場合は環境変数 http_proxy, ftp_proxy の内容が採用されます。

anonymous FTP サイトでは、パスワードとして自分のメールアドレスを入力することが慣習になっているため、このような仕様になっています。

4.7.2 NcFTP

NcFTP はディレクトリを中身ごと転送できる、ブックマークを保存できるなどの機能を備えた、高機能 FTP クライアントです。

```
$ ncftp [< オプション >] {< ホスト >[:< パス >]}
```

-a	anonymous で接続
-u	接続時にユーザ名とパスワードを尋ねる
-p < ポート番号 >	接続するポート番号を指定(省略すると 21 番ポート)
-g < 回数 >	接続をあきらめるまでに接続を試みる回数を指定
-r	接続するまで、何回でも接続を試みる
-d < ポート番号 >	接続しなおすまでの秒数を指定

NcFTP を起動すると普通のコンソールアプリと同じようにプロンプトが表示されます。

```
$ ncftp -L
NcFTP 3.2.2 (Sep 04, 2008) by Mike Gleason (http://www.NcFTP.com/
contact/).
ncftp>
```

さて、どちらのモードでも NcFTP が出すプロンプトに FTP 内部コマンドを入力します。一部の FTP 内部コマンドは NcFTP 独自の拡張がなされている場合もあります。また bash 風の [↑] [↓] によるコマンド履歴の参照や、ファイル名やディレクトリ名の Tab 補完も使用することができます。

では、主な FTP 内部コマンドの解説をしていきましょう。

FTP サイトへの接続と切断 (open, close, quit)

open では、NcFTP 起動時と同じオプションを使って FTP サイトへ接続できます。close では FTP 接続を切断し、quit ではさらに NcFTP 自身も終了します。

```
ncftp> open [< オプション >] < ホスト >[:< パス >]
ncftp> close
ncftp> quit
```

リモート側のファイル参照・ 操作（less, rename, rm, cd, mkdir, rmdir）

NcFTP では、FTP サーバ（リモート）上にあるファイルやディレクトリの参照や操作を行えます。less では環境変数 PAGER に設定してあるページャ（ARMA 標準は lv）が使われます。less と rename というファイルの改名を行う FTP 内部コマンド以外は、同名のシェルコマンドの NcFTP 版と考えてください。

```
ncftp> less <ファイル（複数可）>
ncftp> rename <変更前のファイル名> <変更後のファイル名>
ncftp> rm <ファイル（複数可）>
ncftp> cd <ディレクトリ>
ncftp> mkdir <ディレクトリ（複数可）>
ncftp> rmdir <ディレクトリ（複数可）>
```

ローカル側のファイル参照・操作（lpage, lls, lcd, lpwd）

NcFTPでは、ローカル側のファイルやディレクトリの参照や操作も行えます。これらのコマンド名は「ローカル」（local）を表す l（エル）を先頭に付けています。lpage で呼び出されるページャも less と同じく環境変数 PAGER に設定してあるものです。

```
ncftp> lpage <ファイル（複数可）>
ncftp> lls <ファイル（複数可）>
ncftp> lcd <ディレクトリ>
ncftp> lpwd
```

ファイルのアップロード・ダウンロード（get, put）

FTPでは、アップロードのことをget、ダウンロードのことをputと言います。

```
ncftp> get [-f] [-R] <ファイル名（複数可）>
ncftp> put [-f] <ファイル名（複数可）>
```

-f オプションを付けるとアップ・ダウンロード先に同名のファイルがあっても構わず上書きします。また、-R オプションではディレクトリを中身ごとアップ・ダウンロードできます。

ブックマークの管理（bookmark, bookmarks）

NcFTP では、URL の別名としてブックマークを作成できます。例えば、ftp://ftp.omoikane.co.jp/arma_2.2_updates/ を oglupd という名前でブックマークすれば、次回からは ncftp oglupd と短いブックマーク名でサーバに接続できて便利というわけです。bookmark コマンドではこのブックマークを登録し、bookmarks コマンドではブックマーク管理メニューを表示します。

```
ncftp> bookmark < ブックマーク >
ncftp> bookmarks
```

コマンドの説明（help, ?）

FTP 内部コマンドの簡単な使い方を表示します。コマンドを指定しない場合は、コマンドの一覧が表示されます。help と ? は全く同じ機能です。

```
ncftp> help [< コマンド >]
ncftp> ? [< コマンド >]
```

設定の変更（set）

~/.ncftp/prefs に保存されている NcFTP の設定を変更・確認します。

```
ncftp> set {< 項目 > [< 値 >] | all | help}
```

設定項目だけを指定するとその項目の設定値を表示、設定項目と設定値を両方指定すると設定を変更、all を指定すると全設定の表示、help で全設定項目の説明が表示されます。

4.8 音楽系ツール

4.8.1 Audacious

つまり XMMS 同様、Windows の Winamp クローンということになります。

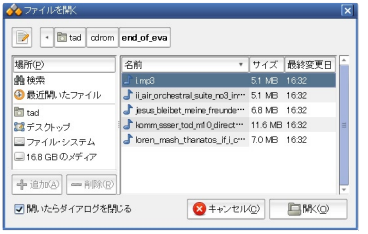
Audacious は XMMS に似たインターフェースを持つ GTK2 で書かれたオーディオプレイヤです。Audacious をコマンドラインから起動する場合は次のようにします。

```
$ audacious [< ファイル名 ( 複数可 )>]
```

GNOME メニュー中では「アプリケーション」-「サウンドとビデオ」-「Audacious2」で起動します。

さて、Audacious を起動すると、下のようなメインウィンドウが開きます。

左下に並んだアイコンは、CD プレーヤと同じ意味で左から順に「前の曲・再生・一時停止・停止・次の曲」を表します。そしてその右の CD プレーヤの「取り出し」のようなアイコンをクリックすると、ファイルダイアログが開きます。



このファイルダイアログでは、ひとつファイルをクリックした後、次のような操作で複数のファイルを選択できます。読み込める音楽ファイルは MP3 (*.mp3) や Ogg / Vorbis (*.ogg) や Wave (*.wav) などです。

- ドラッグすると、マウスカーソルが通過したファイルは全て選択される。
- もう 1 つのファイルを［Shift］+ クリックすると間のファイルが全て選択される。
- もう 1 つのファイルを［Ctrl］+ クリックするとそのファイルは追加選択される。

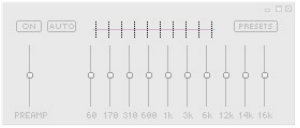
ファイルを読み込んだら再生してみましょう。一番長いスライドバーは再生位置を、その上の２本は左から音量とステレオバランスを調整するスライドバーで、ドラッグで調節できます。また、右下の Shuffle., Repeat のアイコンをクリックすると、ランダム再生や繰り返し再生ができます。

キーボードから Audacious を操作するには、以下のようなキー操作が使えます。

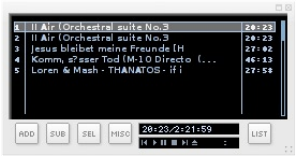
z, x, c, v, b	前曲・再生・一時停止・停止・次曲
l, Ctrl+l	ファイル / URL 読み込みの各ダイアログを表示
r, s	リピート / ランダム再生の ON / OFF
←, →	５秒戻る / 進む
↑, ↓	音量を大きく・小さく
Ctrl+p	設定ダイアログを表示
Alt+s	スキンダイアログの表示

メインウィンドウ中央右のグラフボタン、罫線ボタンをクリックすると、イコライザウィンドウとプレイリストウィンドウが表示されます。

イコライザは低音や高音を強調したりして音のバランスを取る、オーディオ機器お馴染みの機能です。周波数ごとのスライドバーを使って好みの音質に調整できます。



また、プレイリストウィンドウには、現在の Audacious のプレイリストが表示されています。表示されている曲をダブルクリックでその曲を再生できます。下に並んだアイコン類は、メインウィンドウの機能をさらに集約したものになっており、さらにプレイリストの編集も行えるようになっています。



プレイリストの各タイトルを「選択」する操作は、先ほど紹介したファイルダイアログと同じです。また、各タイトルをプレイリスト内でドラッグ

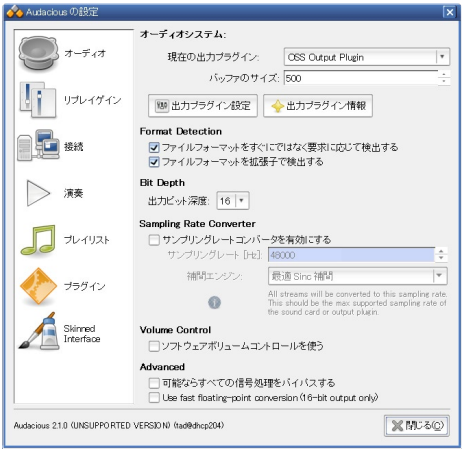
＆ドロップすると、演奏順の入れ替えができます。

プレイリストウィンドウ左下の４つ＋右下の１つのアイコンはクリックするとサブメニューが出てきます。それぞれの項目は以下のような機能を持っています。

FILE+	プレイリストにファイルを追加
DIR+	ディレクトリの中にある音楽ファイルを全てプレイリストに追加
URL+	URL を指定してストリーミング再生
FILE-	選択したファイルをプレイリストから削除
CROP-	選択したファイルだけをプレイリストに残す（FILE- の逆）
ALL-	プレイリストを空にする
MISC-	選択したファイルをディスクから削除
SEL ALL	プレイリストにある全ファイルを選択
SEL ZERO	選択を解除
SEL INV	選択を反転（選択状態と非選択状態を裏返す）
MISC OPT	
FILE INF	選択したファイルの ID3 タグを編集する
SORT LIST	プレイリストのソート（タイトル順・ファイル名順…）
LOAD LIST	プレイリストをファイルから読み込む（*.m3u）
SAVE LIST	現在のプレイリストをファイルに書き出す
NEW LIST	プレイリストを空にする（ALL- と同等）

設定ダイアログ

Audacious 本体と各種プラグインの設定は「設定」ダイアログで行います。「設定」ダイアログはメインウィンドウの一番左上のボタンから「Preferences」を選択するか、[Ctrl] + [P] で呼び出します。



各種の設定はここでおこなってください。

4.8.2 mpg123

mpg123 はコンソール上で動作する軽快な MP3 プレイヤです。ディスク上の MP3 ファイルだけでなく、URL を指定してストリーミング再生も行えます。このため、X Window System を使わない・使えない環境で威力を発揮するほか、Iceweasel などのプログラムが外部 MP3 プレイヤとして利用する場合もあります。

```
$ mpg123 [< オプション >] {< ファイル >|<URL>|-}
```

オプションには主に以下のようなものがあります。

-w < ファイル >	再生内容を Wave（*.wav）に保存
-g <n>	出力ゲイン（レベル）を n に設定
-a < デバイス >	音源のデバイスファイルを指定（初期値 /dev/dsp）

このほか、MP3 のフル再生（デコード）ができない低速な CPU のために、以下のようなオプションがあります。

-m	疑似ステレオ再生（30 ～ 40% CPU 負荷を軽減）

標準的な再生の場合、出力バッファは 1MB = -b 1024 で約 6 秒分になります。

-2, -4　　1 / 2, 1 / 4 にサンプルレートを落とす
-b <n>　　オーディオの出力バッファを n KBytes 使う

4.8.3 Grip

Grip は CD リッパ（吸いだしプログラム）です。音楽 CD のデータを読み込んで、Wave（*.wav）に保存したり、圧縮して MP3（*.mp3）や OggVorbis（*.ogg）として保存したりすることができます。

```
$ grip [-d <デバイス>]
```

-d <デバイス>　CD-ROM ドライブのデバイスファイル（省略時 /dev/cdrom）

起動すると、次のようなウィンドウが開きます。

初めて Grip を起動したときは、「設定」で各種設定を行います。設定項目はたくさんありますが、ポイントになるのは次の項目です。

「吸出し」では、音楽 CD から音声を吸い出して Wave（*.wav）に保存する場面の設定をします。このうち特に重要なのは *.wav のパスを決定する「吸出しファイルのフォーマット」で、次のようなマクロを使って指定します。

%a, %n, %t　　トラックのアーティスト名 / 曲名 / 番号
%A, %d　　ディスクのアーティスト名 / 名前

「MP3」では、Wave をエンコーダにかけて MP3（*.mp3）や OggVorbis（*.ogg）に変換する場面の設定をします。このうち特に重要なのは、「エンコーダ」と「エンコードファイルのフォーマット」です。「エンコーダ」には使用するエンコーダの名前を指定します。ARMA には OggVorbis エンコーダとして oggenc を収録しています。「エンコードファイルのフォーマット」には、出力する *.mp3 や *.ogg のファイル名を、先ほどの「吸出しファイルのフォーマット」と同じマクロを使って指定します。

さて、実際の吸い出し作業に移りましょう。まず「トラック」画面で、吸い出すトラックを右クリックすると、Rip にチェックが付きます。

そして、「吸出し」画面に移って「吸出し＋エンコード」か「吸出しのみ」を選べば作業が始まります。

4.9 DVD-RW / CD-RW のパケットライティング

DVD-RW / CD-RW を使う場合は DVD-R / CD-R 同様に扱うこともできますが、パケットライティングと呼ばれる方法で読書き可能なファイルシステム（UDF）として扱うことができます。

ただし現在の Linux 2.6.31.6 カーネルでは安定していない場合があるようですので本格運用の場合は動作を十分に確認の上おこなうようにしてください。

4.9.1 DVD-RW のフォーマット

DVD-RW メディアをフォーマットする場合は下記のコマンドを実行します。

```
# dvd+rw-format -force /dev/sr0
```

続いて下記を実行します。

```
# mkudffs /dev/pktcdvd/hoge
```

なお、ドライブにセットされているメディアを確認する場合は下記のコマンドを実行します。

```
# dvd+rw-mediainfo /dev/sr0
```

4.9.2 CD-RW のフォーマット

CD-RW メディアをフォーマットする場合は下記のコマンドを実行します。

```
# cdrwtool -d /dev/sr0 -q
```

なお、ドライブにセットされているメディアを確認する場合は下記のコマンドを実行します。

```
# cdrwtool -d /dev/sr0 -i
```

4.9.3 メディアのマウント

DVD-RW / CD-RW いずれの場合も下記の手順でメディアをマウントします。通常のマウントと異なり、pktsetup コマンドでデバイスのセットアップが必要になります。

最初に専用のデバイスをセットアップします。

```
# pktsetup 0 /dev/sr0
```

次にマウントポイントの準備をおこないます。

```
# mkdir /mnt/rw
```

以下の行を /etc/fstab に書き加えます。

```
/dev/pktcdvd/0 /mnt/rw udf noauto,user,rwnoatime  0  0
```

/dev/pktcdvd/ の後にくるパス名は、pktsetup コマンドで与えたものと共通の名前を与えます。ここでは単純に「0」としています。

マウントを実行します。

```
# mount /mnt/rw
```

あとは通常のファイルシステム同様に機能することができます。

4.10 システム管理上のヒント

システム管理では検索サイトを使うのが一般化していますが、ディストリビューション内にも管理のヒントとなるドキュメントが納められており、知っておくと役に立つ場面もあると思います。本章ではそのような文書について説明します。

4.10.1 man

UNIX には man というオンラインマニュアルが用意されています。このマニュアルは、歴史的な理由で次の 8 章からなっています。

第 1 章　実行プログラムまたはシェルのコマンド
第 2 章　システムコール（カーネルが提供する関数）
第 3 章　ライブラリコール（システムライブラリに含まれる関数）
第 4 章　スペシャルファイル（通常 /dev に置かれている）
第 5 章　ファイルのフォーマットとその約束事。例えば /etc/passwd など
第 6 章　ゲーム
第 7 章　マクロのパッケージとその約束事。man（7）, groff（7）など
第 8 章　システム管理用のコマンド

この各章に、コマンドやシステムコールや関数のひとつひとつについて、一定の書式で書かれたマニュアル = manpage が入っています。manpage は、括弧内に章番号を入れて、例えば bash の manpage は bash（1）のように表します。この manpage を見るツールは、その名の通りの man です。

```
$ man [< 章番号 >] < マニュアル名 >
```

例えば、man bash で bash の manpage が見られます。man は環境変数 PAGER で指定されたページャ（テキスト表示ツール）を使って manpage を表示するので、manpage を見ている間もページャと同じ操作でスクロールや検索ができます。

manpage は普通は英語なのですが、パッケージ自身や JM などの和訳プロジェクトによって日本語版 manpage が提供されている場合には、日本語で表示されます。ただし日本語版 manpage は、国産ソフトウェアなど一部を除けばどうしても翻訳に遅れがでるため、ソフトウェアのアップグレードに伴って英語版 manpage やソフトウェアの現状と食い違いが生じることもあります。普段は日本語版が便利ですが、正確な情報を得るには念のため英語版も確認した方がよいでしょう。この英語版 manpage を見るには LANG=C で一時的にロケールを英語に変更して man を実行します。

```
$ LANG=C man [< 章番号 >] < マニュアル名 >
```

ところで、同じ名前のマニュアルが別々の章に収められている場合もあります。例えば printf は第1章と第3章にあり、それぞれ printf コマンドと C 標準ライブラリ関数の printf について書かれています。どちらのマニュアルかを明示的に指定するには man 1 printf などと章番号を指定します。また、man -f で同名のマニュアルの一覧を表示することもできます。

```
$ man -f <マニュアル名>
$ man -f printf
printf (1)           - format and print data
printf (3)           - formatted output conversion
```

man -k はマニュアルの逆引きをします。キーワードを指定すると、そのキーワードを含む manpage が一覧表示されます。

```
$ man -k <キーワード>
$ man -k bash
bash (1)             - GNU Bourne-Again SHell
bashbug (1)          - report a bug in bash
builtins (1)         - bash built-in commands, see bash(1)
rbash (1)            - restricted bash, see bash(1)
```

4.10.2 info

いくつかのコマンドには info という man とは別系統のオンラインマニュアルが用意されています。両者の違いは主にその構造にあり、man が一本のテキストファイル的であるのに対し、info は Web ページのように各「ノード」の間にリンクが張られた形になっています。そのために、info を読むには専用のビューア（info コマンド）か、Emacs 系エディタの info モード が必要です。

Emacs 系エディタで info モードに入るには、M-x info とします。ちなみに、M-x は Alt と x を同時に押すことを意味しています。

```
$ info <インフォ名>
```

info ビューアでは次の操作キーが使えます。Web ブラウザ感覚と言ってよいでしょう。

p	「前」に表示したノードを表示。（Web ブラウザの「戻る（←）」相当）
n	「次」に表示したノードを表示。（Web ブラウザの「進む（→）」相当）
u	一階層「上」のノードに移動。
t	トップ（入口）ノードへ移動。
Enter	リンクを辿る

4.10.3 パッケージの添付文書

Debian では、各パッケージの添付文書を /usr/share/doc 以下にパッケージごとにディレクトリを作って収めています。添付文書には、ソフトウェアの原作者が提供するものと deb パッケージのメインテナが提供するものがあり、両方が同じディレクトリに入っています。原作者による典型的な添付文書には、以下のようなものがあります。

CHANGES	ソフトウェアの履歴
README	一般的な情報
TODO	今後の開発予定

逆に、メインテナが提供する添付文書には、以下のようなものがあります。

changelog.Debian	パッケージの履歴
copyright	ソフトウェアの入手元、著作権に関する情報
README.Debian	パッケージに関する一般的な情報

これ以外にも様々な添付文書がテキスト形式に限らず HTML,SGML,TeX 形式などで収められていたり、設定のサンプルなどが収録されていたりすることがあります。

また、添付文書は原作者やメインテナによって gzip 圧縮されていることがありますが、lv コマンドは gzip 圧縮されたファイルも自動的に展開して読むことができます。

```
$ lv changelog.Debian.gz
```