2010年度「すばるデータ解析春の学校」のためのLinuxのメモ

古屋 玲 (rsf あっと subaru.naoj.org)
八木 雅文 (YAGI.Masafumi あっと nao.ac.jp)

May 24, 2010

Contents

改訂履歴

Version	Date
2008 May 19	2008 年春の学校向けに作成 (初版)、学校後に改訂 by R.S.F & M.Y.
2009 May 21	2009 年春の学校向けに改訂 (May 1 事前配布版) by R.S.F & M.Y.
2010 May 24	2010 年春の学校向けに改訂及び修正 (May 24 事前配布版) by R.S.F & M.Y.

Chapter 1

ログインとログアウト

1.1 ログイン

春の学校ではデータ解析センターの利用者アカウントは subaru (subaru03 から subaru12 まで) の番号が振られます。

1. ws という名前の計算機端末のどれかの前に座って下さい。例えば ws01 としましょう。なお、subaru の番号と ws の番号は必ずしも一致する必要はありません。ここでは subaru12 で ws01 の前に座ったとします。
2. subaru12 としてログインします。今、あなたは ws01 という名前の計算機端末で、ディレクトリは subaru12 の home ディレクトリにいるはずです。
3. `startx` と打って、Xwindow を起動します。真っ赤な画面が立ち上がります。右クリックでターミナル (画面端末)[1]を開きます。これも ws01 の端末です。
4. そこから、実際に解析を行なうサーバにログインします。

   ```
   ssh ana03
   ```

   と打って下さい。この ana の後の番号は当日指示しますが、ここでは ana03 とします。パスワードを聞かれますので入力して下さい。この時点で、あなたがいるターミナル (コマンドを受け付ける窓のこと) は ana03 と呼ばれる別の計算機の端末になってます。
5. 作業用ディレクトリを ana03 に作成します。ディスクには home ディレクトリと作業用ディレクトリがあり、最初にログインしたときには、home ディレクトリにいますが、(多くの場合、ディスク占有領域が多くなる、などの理由で) 実際の作業は他の作業ディレクトリで行います。ana03 には/wa03a, /wa03b という作業領域があります。今回は/wa03a に subaru12 というディレクトリを作ることにしましょう。まず /wa03a に移動します。

   ```
   cd /wa03a
   mkdir subaru12
   ```

   これで /wa03a の下に subaru12 というディレクトリができたはずです。
6. /wa03a/subaru12 に移動しましょう。

   ```
   cd /wa03a/subaru12
   ```

[1]著者たちは安易なカタカナ英語の使用は嫌いです。しかし、日本語ではうまく表現できないものや計算機関連で人口に膾炙してしまっているカタカナ英語は我慢して使っています。このようなカタカナ英語 (的発音) のなかには、日本人にしか通用しないものがあることも心に留めておいてください。

7. 自分がどのサーバにいるか確認しましょう。

   `uname`

   自分がどのディレクトリにいるかも確認しましょう。

   `pwd`

   この/wa03a/subaru12 を「作業ディレクトリ」と呼ぶことにします。#番号はそれぞれ人によって違います。

1.2 初期パスワードの変更

Unix/Linux では、通常は `passwd` コマンドを使うが、今回の解析環境では、ws03 など、ログインしているローカルなターミナルから

`modify_userinfo -p`

で変更する。初期パスワードを入力後、変更したいパスワードを 2 回入力すれば変更される。

1.3 ログアウト

1. ana03 など、リモートでログインしているターミナルで

   `jobs`

   で実行中のジョブがないか確認します。実行中のジョブがあるとログアウトできない事があり、無理にログアウトするとジョブが殺されてややこしい事になります。実行中のジョブがある場合は指導員？に相談して下さい。

2. ターミナルで

   `exit`

   でログアウトします。

3. ws** という表示になるので、そこで再度

   `jobs`

   で実行中のジョブがないか確認し (§4.4.3 の `ps` コマンドも参照)、何もない場合は

   `exit`

   でログアウトします。これでターミナルが閉じるはずです。

4. 画面左上の System メニューから Logout subaru** を選択します。これで画面が真っ黒になります。XWindow が終了したからです。

5. そこで、再度

   `exit`

   を実行します。画面に ws** login: という表示が出ればＯＫです。

席を長時間離れる時は必ずログアウトすること。電源を切る必要はありません！

Chapter 2

困ったときは

2.1 man コマンド

困ったときは質問したり、ネットで検索するまえに man コマンドを使う習慣をつけよう。
man コマンド名とすればヘルプを得られる。例えば、man コマンドの使い方を知りたいとき、

```
man man
```

とすれば、以下のように、コマンドの書き方 (文法)、使用可能なオプションの種類など、必要な情報を得られる。

```
man(1)                                                          man(1)

NAME
       man - format and display the on-line manual pages

SYNOPSIS
       man  [-acdfFhkKtwW]  [--path]  [-m system] [-p string] [-C config_file]
       [-M pathlist] [-P pager] [-B browser] [-H htmlpager] [-S  section_list]
       [section] name ...

DESCRIPTION
       man formats and displays the on-line manual pages.  If you specify sec-
       tion, man only looks in that section of the manual.  name  is  normally
       the  name of the manual page, which is typically the name of a command,
       function, or file.  However, if name contains  a  slash  (/)  then  man
       interprets  it  as a file specification, so that you can do man ./foo.5
       or even man /cd/foo/bar.1.gz.

       See below for a description of where man  looks  for  the  manual  page
       files.
```

2.2 apropos コマンド

探したいコマンドなどの名前が正確に分からないとき、とにかく関連するかもしれないマニュアルページをすべて表示させたいときは、apropos コマンドで ”曖昧” 検索してみる。

```
apropos pass
```

画面に表示される結果は、端末に表示しきれないくらい長いかもしれない。`apropos pass | more` (§3.14.2 も参照)、あるいは `apropos pass | grep change`、などと組み合わせて欲しい情報を探そう。

2.3 ユーザーズガイド

天文台の天文データセンターのユーザーズガイドが `http://www.adc.nao.ac.jp/J/cc/misc/lm2008/html/` にある。

Chapter 3

よく使うコマンド — 使って覚えよう!

3.1 メモをとっておきたい

3.1.1 情報を選んで保存したい

もっとも単純な方法は、

```
emacs whatidid.txt &
```

とエディター (§3.2 参照) を開き、画面上の欲しい情報を copy&paste する。

3.1.2 何も考えずすべて保存したい

`script` コマンドは、あなたが打ち込んだものやターミナルにでてきたものをすべて、機械的に何も考えずに、指定したファイル (この例では `whatidid.txt`) へ書き出してくれる。

```
script whatidid.txt
```

として、記録開始したあと作業をする。作業終了後、

```
exit
```

とすれば、記録先のファイルは閉じられ、それまでの作業記録がすべて `whatidid.txt` に保存される。

3.1.3 あとで質問するために、画面全体のコピーを取っておきたい

```
ksnapshot &
```

として、まずは使ってみてください。画像として保存した、画面のコピー (ファイル名を `myscreen.png`) を講師に見せたいときは、

```
display myscreen.png &
```

ここで使う `display` コマンドは`/usr/bin/display` 等にある unix コマンドで IRAF の同名タスクとは異なることに注意。

3.2 テキストエディタ (= テキストの内容を編集する道具)

```
emacs filename.txt &
```

どのエディタを使うかは好みだが、他に

```
xemacs filename.txt &
vi filename.txt
```

などなど。"&" (アンパサンド; ampersand) については §4.4.1 参照。

3.3 ディレクトリ

```
pwd
cd
cd ..
cd ~/mydata
cd /wa03a
mkdir
```

3.4 ファイルの一覧 (list)

あるディレクトリ配下のファイル一覧をみたいとき、

```
ls
ls /wa03a/rsf/data
```

.(ドット) で始まる、各種設定ファイルなどの "隠し" ファイルの一覧が欲しいときは、

```
ls -a
```

[応用] どんな結果が得られるだろうか？ — 基本的なワイルドカード

```
ls *.FITS                                何らかの文字列
ls DATA0?.FITS                          何らかの単独文字列
ls DATA[1-3].FITS                [m-n] m から n までの数字
ls DATA[!1-3].FITS            m から n までの数字を含まない
ls DATA{1,3,4,a,c,e}.FITS              {}内に指定した文字
ls DATA[4-5].{FITS,fits}
ls DATA{1,2}[^3].FITS
```

などなど。*, ?, [ ], { }, ^ などの記号は意味を持つことに注意。開き角括弧のあとのエクスクラメーションマーク、即ち [!、を使えば指定された文字列 でない ものを選び出すことができる。

3.5 ファイルを開かずにファイルの中を見る

```
less
more
```

3.6 消去 (remove)

ファイルの場合

```
rm myfile.txt
```

ディレクトリの場合

```
rmdir mydir
```

ディレクトリの中をファイルごと一括して削除

```
rm -r mydir
```

3.7 複製 (copy)

ファイル A.txt をファイル B.txt へ複製

```
cp A.txt B.txt
```

ひとつ上のディレクトリにあるファイルをカレントディレクトリ (＝現在、あなたがいるディレクトリ) の下へ複製

```
cp ../A.txt .
```

ディレクトリの中身ごと別名のそれへコピーしたい場合

```
cp -r hisdir/ herdir/
```

複製元の時刻情報や許可属性などを維持したいときは、さらに-p オプションも加えて、

```
cp -rp original.txt copied.txt
```

3.8 mv コマンド；移動と名前の変更

3.8.1 移動

ファイル DATA1.FITS をディレクトリ scr の下へ移動

```
mv DATA1.FITS scr
```

[応用] 以下では、どんな結果を得られるだろうか?

```
mv DATA1.FITS  /wa03a/rsf/08a/
mv DATA1.FITS ~/temp/
mv DATA1.FITS .
```

2 番目の例にある、~はそのユーザーの home ディレクトリを示す。

3.8.2 名前の変更

YourDATA.FITS というファイルの名前を MyDATA.FITS へ変更

```
mv YourDATA.FITS MyDATA.FITS
```

3.9 find コマンド；指定した特性をもつファイルを探す

- 現在いるディレクトリ (カレントディレクトリ、”.”) 以下にある、.f で終わるファイルを探す。

  ```
  find . -name "*.f"
  ```

  あるいは、

  ```
  find . -name \*.f
  ```

- 最後に変更が加えられたのがちょうど 3 日前であり、かつ.pro で終わるファイルを探す。

  ```
  find . -name "*.pro" -mtime 3
  ```

  上の例は 3 日以内、+3 であれば 3 日以上前を指定することになる。詳細は `man find` で見てみよう。

3.10 grep コマンド；指定した文字列を探す

- あるファイル中のある文字列 (ここでは abc や abc def) を含む行を抜き出す。

  ```
  grep abc myfile.txt
  grep 'abc def' myfile.txt        スペースを含む文字列の場合
  ```

3.11 シンボリックリンク

3.11.1 ディレクトリの場合

皆さんの実際の作業ディレクトリは/wa03a/rsf/data/08a/であり、このディレクトリ配下のファイル一覧を知るために毎回 `ls /wa03a/rsf/data/08a` などとすべて打ち込むのは面倒。

```
ln -s /wa03a/rsf/data/08a ./08a
```

としておけば `ls /wa03a/rsf/data/08a` と打つ代わりに

```
ls 08a
```

で同じ結果を得られる。

3.11.2 ファイルの場合

ファイル DATA1.FITS は、実際はディレクトリ/wa03a/rsf/data/08a 配下に存在する。このファイルがあたかも、別のディレクトリに存在するかのようにさせたい。

```
ln -s /wa03a/rsf/data/08a/DATA1.FITS DATA1.FITS
```

としておけば、カレントディレクトリで見える、`DATA1.FITS` は`/wa03a/rsf/data/08a/`配下に存在するそれと同様に扱える。

3.12 ファイルやディレクトリの許可属性

あるプログラムを実行しようとしたり、ディレクトリの中を見ようとしたときに `Permission denied` と怒られるときがある。このようなときは、あなたに読む権利や実行権 (=まとめて許可属性と呼ぶ) がないことが理由。

- あるファイルの許可属性を知るには、

  ```
  ls -lh a.tar.gz
  ```

  として得られる結果が

  -rwxr-xr-x+ 1 rsf naoj 218K Apr 24 16:07 a.tar.gz

  であるとき、先頭の-はファイルを示す。以下、rwx で表され 3 つで一組を構成する rwx のフラグは、ファイル所有者 (user)、ファイルのグループ所属者 (group)、他のユーザー (others) の順に、読み出し (r)、書き込み (w) 及び実行権 (x) があるか否かを示す。

- 実行権を与えるときは、

  ```
  chmod +x hisprogram
  ```

- 他のユーザー (o) に実行権を与えるときは、

  ```
  chmod o+x hisprogram
  ```

- 実行権を取り消すときは、

  ```
  chmod -x hisprogram
  ```

- あるディレクトリ以下は、他人に見られたくない!?ときは、

  ```
  chmod -R 700 *
  ```

  -R や 700 の意味は、`man chmod` で調べてみよう。

3.13 ディスクの空き容量を知る

解析が進むと中間生成物のデータが増えて、知らないうちにハードディスクのかなり領域を占有し始めているものです。上の例で、pippo[1-5] という作業ディレクトリがそれぞれどれほどディスク領域を使っているでしょうか？

```
du -sh pippo[1-5]
```

h オプションを付けたので、人間 (human) にも分かりやすい単位で出力されました。カレントディレクトリの合計使用容量をみるには、

```
du -sh .
```

最後の.(ドット) は、カレントディレクトリを示す。その計算機全体での使用状況をみるには、

```
df -kh
```

3.14 標準入出力；リダイレクトとパイプ

3.14.1 リダイレクト

- 結果の出力先をターミナル画面でなく、あるファイルへ書き出す。

```
ls MyDATA*.FITS > fitslist.txt
```

- 結果の保存先ファイルである、filelist.txt へ別のファイル一覧を続けて書き出したい (= アペンドする) ときは>>を使う。

```
ls YourDATA*.FITS >> fitslist
```

3.14.2 パイプ

あるコマンドの出力結果を別のコマンドの入力として使いたいときがある。例えば、あるディレクトリの下に、pippo1, pippo2, pippo3, ..., and pippo5 という名前の 5 つの作業ディレクトリがあるとする。この作業ディレクトリそれぞれがどれほどのディスク領域を使用しているのか知りたい。さらに、その結果を使用量の大きい順番に並べたい。まずは、単純にコマンドを個別に実行した場合、

```
du -s pippo[1-5] > pippo.list
sort -nr pippo.list
```

この二つのコマンドは、|(パイプ) を使って、以下のようにまとめて実行できる。実行権を与えるときは、

```
du -s pippo[1-5] | sort -nr
```

ただし、パイプは画面にでてきた出力をそのまま次へ渡しているだけです。おかしなことをしていても、エラーを吐き出さないので注意。

3.15 印刷

基本的な使い方は、天文データセンターの web ページ、

http://www.adc.nao.ac.jp/J/cc/misc/lm2008/html/node30.html

を参照のこと。なお、今回皆さんが使用できるプリンターは、白黒の lp2 とカラーの clp2-a4 で、前者がデフォルトの印刷先の設定になっている。

Chapter 4

シェル、プロセス制御、アプリケーション

— ここでは bash の使用を前提に話を進めます。—

4.1 コマンド入力を簡単にするために

- 補完機能；すべてをタイプせずとも、tab キーを叩いてみよう。
- ヒストリ機能；↑キーで過去に打ったコマンド呼び出し
- ヒストリ機能；以下も便利、やってみよう。

  `!!`
  `!`(文字列、たいていはコマンド名)、例えば`!ls`

- エイリアス；コマンド等に短く別名を付ける。

  `alias hF='history|grep FITS'`

  と別名を付けておけば、`hF` と打つだけで、引用符内のコマンドをすべて打たなくても済む。この例では、過去に打ったコマンドのうち、キーワードとして `FITS` を含むものを抽出するコマンドを定義。ただし、このコマンドはそれを宣言したターミナル内でのみ有効なので、ログアウトするたびにエイリアス消えてしまい現実的にありがたみがない。そこで、`.bashrc` などに書き込んでおき、いつでもどのターミナルにおいても使えるようにしておくのが普通。

- `history` コマンドの出力の中の番号を一つ選ぶ (例；`history | grep awk`)。例えば `789` が繰り返したいコマンドの番号だった場合、`!789` としてみよう。
- 2 つまえのコマンドを繰り返したければ、`!-2`
- [応用] `history | grep mv` で何が表示されるだろうか? `|`については §3.14.2 参照。

4.2 アプリケーションの場所を知る

IRAF(cl) と SExtractor という、アプリケーションがどこにあるか調べる。

```
which cl
```

と打つと、
/usr/local/bin/cl と、アプリケーションの実行形式ファイルの位置が返ってくる。同様に、

```
which sex
```

/usr/local/bin/sex

先頭の/ (スラッシュ) には意味があり、絶対パスを示すことに注意。先頭に/がなれば、カレントディレクトリからの相対パスを意味する。

4.3 パス (path) を通す

実行コマンドがどこにあるかを探す時には PATH 環境変数に出てくる順に探す。今の PATH 環境変数は

```
echo $PATH
```

で表示できる。コマンドを新たに追加したときなど、そのディレクトリへ path を通す必要がある。たとえば、ds9 という画像表示プログラムが/home/rsf/opt/ds9/という名前のディレクトリにある時、

```
export PATH=$PATH:/home/rsf/opt/ds9
```

で path を追加する。この意味は、古いパスの最後に新たに/home/rsf/opt/ds9 を追加してください、の意味。逆に優先的にこちらを探させたい場合は

```
export PATH=/home/rsf/opt/ds9:$PATH
```

とする。

4.4 プロセスの制御

4.4.1 処理をバックグラウンドで実行

```
emacs myfile.text &
```

(&アンパサンド; ampersand) を付ければ、この仕事はバックグラウンドで実行され、コマンド入力待ち状態に戻ることができる。

4.4.2 処理をバックグラウンドにまわす

- &なしで実行した場合、実行させた仕事が終わるまで、次の仕事のためのコマンドを打つことはできない。このような場合、実行したターミナルで Ctrl-z (control キーを押しながら、z を押す) を打つ事で処理を停止 (サスペンド) することが出来る。停止した処理は、fg(フォアグラウンド) コマンドで再度実行が出来、bg(バックグラウンド) コマンドで、&付きで実行したようなバックグラウンド実行に変更することができる。
- 注意：ターミナルから & を付けてバックグラウンド実行した時、そのターミナルから exit が出来ないことがある。この時、無理矢理ターミナルを終了させると、実行中のジョブも強制終了されてしまうことがある。
- バックグラウンドに回したジョブを実行させたままログアウトしたい場合、nohup コマンドが便利。例えば、nohup ./command.sh &

4.4.3 現在実行中のプロセスとその番号を知る

- 現在実行中のプロセスとその番号を知る

  ```
  ps -ax | grep commandname
  ```

  commandname には、知りたいコマンド名を与える。

- どのプロセスがもっとも CPU を使っているか、知りたいとき、

  ```
  top
  ```

- 現在実行中のジョブとその番号や状態を知る

  ```
  jobs
  ```

  + が付いているのが現在見ている優先ジョブ

4.4.4 プロセスの終了と強制終了

- §4.4.3 で番号を調べたプロセスを終了させたいとき、

  ```
  kill processID
  ```

  終了させたいプロセスの番号を processID に与える。

これでも停止させられなければ、次のような方法で強制終了。

プロセスの強制終了 (例 1) `kill -KILL processID`
-KILL は大文字、processID は §4.4.3 で番号を調べたプロセスの番号。

プロセスの強制終了 (例 2) 止めたいプロセスが走っているターミナル上で、control キーを押しながら z キーを押す。その後、`kill %1` で [ ] 内の数字が割り当てられた processID を下のように強制終了させる。

```
% cat
(\texttt{control}キーを押しながら\texttt{z}キーを押す)
[1]+  Stopped                 cat
% kill %1
%
[1]+  Terminated              cat
%
```

Chapter 5

ネットワーク経由で他の計算機を利用する

5.1 他の計算機へログインする

- `ssh -X -l rsf pueo` として、パスワード入力ののち pueo という名前の計算機にユーザ名 rsf でログインできる
- 仕事が終わって出るとき、`logout`

5.2 他の計算機との複数ファイルのやりとり: sftp コマンド

```
sftp -l rsf machinename
cd /wa03a/rsf/data/08a
mget DATA*.FITS ./
bye
```

上の例は、pueo という計算機から、名前が DATA で始まり、かつ名前が.FITS で終わる全てのファイルを取ってくる (`mget`) 場合で、逆の場合 (=置きにいくとき) は、`mput`。

5.3 他の計算機へディレクトリごとデータをやりとりする

あるディレクトリ配下のデータを送る

```
scp -r targetdir rsf@pueo:/home/rsf/
```

他の計算機の、あるディレクトリ配下のデータ (以下の例では、targetdir という名前) をディレクトリごとカレントディレクトリ (.) へ持ってくる

```
scp -r rsf@pueo:/home/rsf/targetdir/ .
```

5.4 アーカイブシステムでの検索結果など、web ページに一覧表示された複数のデータを一挙に取得したい

ダウンロードしたい URL の一覧 (ここでは `whatIwant.txt`) を作ったのち `wget` コマンドで一括ダウンロードすべし。もっとも単純には

```
wget -i whatIwant.txt
```

である。ユーザー認証などに関連して文句を言われ失敗するかもしれない。エラーメッセージを手がかりに `man wget` も参考にして、然るべきオプションを加えてみよう。

Chapter 6

解析時によく使いそうなコマンド例

6.1 ディレクトリを作成し、そこへデータを持ってきて展開する

```
cd
mkdir temp
cd temp
cp /home/yagims/sample.tar.gz .
tar xvzf sample.tar.gz
```

6.2 あるディレクトリ配下にある、拡張子FITSで終わる名前のファイルの一覧を作成し、myfiles.txtという名前のファイルへ書き出す

```
ls *FITS > myfiles.tzt
```

6.3 あるテキストファイルと別のテキストファイルの内容を行をそろえて合成する

```
paste myfile.txt yourfile.txt > ourfile.txt
```

6.4 2つのテキストファイルの差を知る

```
diff myfile.txt yourfile.txt
```

6.5 あるファイルの行数を数える

```
wc -l filename.txt
```

wcコマンドに-lオプションを付ければ、lineを数える。他に何のオプションがあるかは、man wcで見てみよう。
行数を数えるだけなく、ファイルに行番号を付けたい場合は、

```
cat -n myfile.txt
```

6.6 あるファイルから、任意の行や項目を別ファイルへ書き出す

3 項目のデータのみ書き出したいとき

```
awk '{print $3}' myfile.txt > my3rdparm.txt
```

3 行目だけを書き出したいとき

```
awk '{if(NR==3)print}' myfile.txt > my3rdcolumn.txt
```

6.7 あるファイル中の、ある文字列 (パターン) を一括して別の文字列に置き換える

- 通常の文字列の場合は、

  ```
  sed -e 's/my/your/g' myfile.txt > results.txt
  ```

  ここでは my を一括して、your に置き換えた。

- もし、置換したい文字列に/などの特殊記号 (メタキャラクタ) を含む場合は、バックスラッシュ(\) を使って、

  ```
  sed -e 's/\/home\/rsf/\/scr\/jansky/g' myfile.txt > results.txt
  ```

  この例では/home/rsf を/scr/jansky に置き換えた[1]。

[1]GNU sed の s のデリミタは"/ "でなくても動作するので、例えば sed -e 's,/home/rsf,/scr/jansky,g' myfile.txt > results.txt とした方が、目に優しいかも。

Chapter 7

データの保存と持ち帰り

7.1 tar アーカイブを作成して、sftp 転送する

7.1.1 三鷹から、大学へ転送

1. tar アーカイブを作成、圧縮

   ```
   tar cvzf mydata.tar.gz mydata
   ```

   圧縮したアーカイブ (俗に tar ボールとも呼ぶ) を sftp ($5.2 参照) で、あなたの大学へ転送してしまう。

2. 上の工程をもう少し丁寧に作業したければ、

   ```
   tar cvf mydata.tar mydata
   ```

   で、先ずは tar アーカイブ作成する。次に、

   ```
   tar tvf mydata.tar | grep FITS
   ```

   で作成されたアーカイブに収納されたファイルのうち、FITS というキーワードをファイル名に持つものを抽出。結果に happy だったら、

   ```
   gzip -v mydata.tar
   ```

   で圧縮 (-v は圧縮率を表示するお遊びオプション) して、結果のファイルのサイズを

   ```
   ls -lh mydata.tar.gz
   ```

   で見てみる。「常識的」なファイルサイズであれば、sftp か scp($5.2 参照) してしまおう。

7.1.2 大学に帰ってから、データを読み出すとき

tar アーカイブの中味を見てから、解凍する。

```
tar tvzf mydata.tar.gz
gunzip -c mydata.tar.gz | tar xvf -
```

あるいは、

```
tar xzvf mydata.tar.gz
```

7.2 別の方法; rsync

今日までの作業結果を帰宅まえに毎回保存したいと思うかもしれない。しかし、すべてのデータを毎回保存先 (ネットワーク経由の計算機やローカルにつないだ記憶装置など) に転送するには無駄が多い[1]。あるいは link (§3.11) や許可属性、変更時刻情報なども含めて作業環境を保存したいと思うかもしれない。このようなときに `rsync` が非常に便利。`rsync` は変更のあったものだけコピーするので、ディレクトリやファイルツリーの同期を保つのに便利。簡単な例を示す。

```
rsync -av /home/rsf/subaru/ /media/myHD/tmp_subaru/
rsync -av -e ssh /Users/rsfuruya/public_html/ rsf@shell:/home/public_html/
```

`man rsync` で可能なオプション (たくさんある!) を見てみよう。

7.3 CD や DVD などのメディアに”焼く”

1. 持ち帰りたいデータ (この例では、mydata ディレクトリ以下とする) の総容量を確認 (§3.13 参照) する。
2. `mkisofs` (make iso file system) コマンドで ISO 9660 という標準形式でデータをひとつのファイル (ここでは、mydata.iso) にまとめる。

   ```
   mkisofs -J -r -o mydata.iso -V 2008MAY11 mydata
   ```

   `-J` オプションは Joliet の命名規則を使用することによって Windows との互換性を保つ[2]ためのオプション、`-V` オプションのあとの文字列はボリューム名 (つまり、名前) なので好きなものを与えてよい。最後の `mydata` が CD に書き出したいディレクトリの名前となる。実行すると途中経過が表示される。
3. 終了したら、CD-R を挿入したのち、

   ```
   cdrecord -v dev=1,0,0 mydata.iso
   ```

   で CD-R に焼く。CD-R/W に焼く場合は、

   ```
   cdrecord -v -tao dev=1,0,0 mydata.iso
   ```
4. 無事にすべて書き込まれたら、`eject` で CD を取り出し、再度挿入、内容を確認しよう。

- 上の例では、CD 書き込み装置の SCSI アドレスが `1,0,0` を想定したが、システムの設定によってはアドレスが異なるかも知れない。アドレスを知りたいときは、

  ```
  cdrecord -scanbus
  ```

  とすれば使用可能なデヴァイス (dev) 一覧を得られる。
- データ解析センターの FAQ、`http://www.adc.nao.ac.jp/J/cc/misc/lm2008/html/node27.html` も参照すること。

[1] §3.9 や §3.10 を参考に本日の作業結果一覧のテキストファイルを作成し、その一覧表を読み込ませて tar アーカイブを作成し、`sftp` する (§7.1.1 参照) こととほぼ同じでもある。つまり、目的と環境や手数に合わせて最適の保存法を探せばよい。

[2] `-J` オプションをつけないで作成された CD を Windows で読んだ場合、ファイル名は最初の 8 文字まで、拡張子は同 3 文字までしか認識されない。例えば、.FITS が.FIT になってやや不都合。

7.4 取り外し可能なデバイス (e.g., USB メモリ、外付け HD) に書き込む

マウントポイントの設定にもよるが、

```
ls /media
```

とすれば、差し込んだデバイスが見えるはず (以下の例では myHUGE-HD) なので、持ち帰りたいデータを作業ディレクトリごとコピー (§3.7) すればよい。

```
cd /media/myHUGE-HD/
mkdir subaru08
cd subaru08
cp -r /wa03a/subaru12/ .
```