

# Linux/Mac コマンドライン 講習会 Day 2 つづき

笠原 雅弘

東京大学 大学院新領域創成科学研究科  
メディカル情報生命専攻

Twitter: @mkasahara



Masahiro Kasahara  
@mkasahara

# 環境変数

- シェルの変数は **export** すると複数のプログラムにまたがって使用できる
  - bash で変数 I に 1 を代入
  - bash から sh を起動したときに \$I がどうなるか？
    - export しなければ \$I は空に、export すれば 1 になる。

```
bash-3.2$ I=1
bash-3.2$ sh -c 'echo $I'

bash-3.2$ I=1
bash-3.2$ export I
bash-3.2$ sh -c 'echo $I'
1
```

# 環境変数設定の省略記法

- export 変数名=値

```
bash-3.2$ export ABC=123  
bash-3.2$ sh -c 'echo $ABC'  
123
```

# システム既定の環境変数

- 特殊な意味がある環境変数
  - PWD
    - カレントディレクトリ
  - HOME
    - ホームディレクトリ
  - PATH
    - 後述のPATH
  - MANPATH
    - man コマンドがマニュアルページを探しに行くディレクトリ
  - LD\_LIBRARY\_PATH (Linux),  
DYLD\_FALLBACK\_LIBRARY\_PATH (Mac OS X) など
    - 共有ライブラリ（様々なソフトウェアから共有して利用できるモジュールのことをライブラリと言う。）をサーチする  
PATH

# 個別のアプリケーションが参照する環境変数例

- CLICOLOR\_FORCE (Mac 等のみ)
  - ls コマンドは `ls -al | cat` のようにパイプ等に結果を流した場合に色を付けない機能が付いている。
  - 何か (例えば 1) をセットすると `ls -al | cat` のように ls の結果をパイプ等に流した場合にでも色を付けるようになる。`ls -al | less -R` などする場合には便利。
- PERL5LIB
  - Perl が拡張モジュールを探すディレクトリ群
  - Python だと PYTHONPATH、Ruby だと RUBYLIB など。
- LESS
  - less はこの環境変数にセットされたオプションをデフォルトにする。(例: `export LESS=-R`)

# env

- env
  - 環境変数とその値の一覧を表示する

```
bash-3.2$ env | head
ABC=123
MANPATH=/Users/mkasa/lcl/share/man:/Users/mkasa/lcl/share
/texlive/2012/texmf/doc/man:/Users/mkasa/lcl/share/man:/U
man:/usr/local/share/man:/opt/X11/share/man:/opt/local/sh
f/doc/man
TERM_PROGRAM=iTerm.app
LPM_LOCAL=/Users/mkasa/lcl
SHELL=/bin/zsh
TERM=xterm-256color
CLICOLOR=1
TMPDIR=/var/folders/gg/13mj7x_90tdg_j1jx6gf347h0000gn/T/
I=1値の一覧を表示する
Apple_PubSub_Socket_Render=/private/tmp/com.apple.launchd
```

# env

- `env ABC=123 sh -c 'echo $ABC'`
  - 環境変数 ABC を 123 にセットして `sh -c 'echo $ABC'` を実行する

# 環境変数LANG

- 使用する言語や文字コードをセットする
  - [言語コード(2文字)]\_国コード(2文字).文字コード

```
[mkasa@zombie]~% echo $LANG
ja_JP.UTF-8
[mkasa@zombie]~% man ls | head -10
LS(1)
TERM_PROGRAM=iTerm.app
LPM_LOCAL=/Users/mkasa/lcl
SHELL=/bin/zsh
TERM=xterm-256color
CLICOLOR=1
TMPDIR=/var/folders/gg/13mj7x_90tdg_jl1x6gf347h0000gn/T/
ls, dir, vdir - ディレクトリの中身をリスト表示する
Apple_PubSub_Socket_Render=/private/tmp/com.apple.launchd
書式
ls [options] [file...]
```

LS(1)



# env で一時的に LANG を変更

- 日本語から一時的に英語に変更

```
[mkasa@zombie]~% env LANG=en_US.utf-8 man ls | head -10
LS(1)                                User Commands                                LS(1)of
NAME
  ls - list directory contents
SYNOPSIS
  ls [OPTION]... [FILE]...
```

一番「プレーン」な設定（普通は英語）は“C”で表す決まりになっている。

```
[mkasa@zombie]~% env LANG=C man ls | head -10
LS(1)                                User Commands                                LS(1)of
SYNOPSIS
  ls [OPTION]... [FILE]...
NAME
  ls - list directory contents
SYNOPSIS
  ls [OPTION]... [FILE]...
```

# LANG で出力が変わる例

- LANG=ja\_JP.utf-8 の例

```
[mkasa@zombie]~% date  
2016年  3月 29日 火曜日 11:46:03 JST  
[mkasa@zombie]~% LANG=C date  
Tue Mar 29 11:46:06 JST 2016
```

LANG 等の環境変数で言語が変化するコマンドは山ほどあるが、日本人などが開発に参加していないソフトウェアもたくさんあるし、日本語用のメッセージを格納したファイル（リソースファイル）をインストールしていない場合などもあるので、LANG=ja\_JP.utf-8 としたら必ず日本語が出てくる、というわけでもないので注意。

# 言語設定関連の環境変数

- LANGUAGE
- LC\_ALL
- LC\_CTYPE, LC\_NUMERIC, LC\_TIME, LC\_COLLATE, LC\_MONETARY, LC\_MESSAGES, ...
  - 数字の 3 桁区切りを, でやる国、. でやる国
  - 小数点を, で書く国、. で書く国、...
  - 通貨の単位
  - メッセージの言語
- LANG
  - LC\_\* で個別設定したいことは稀なので、LANG を設定しておしまい、にすることが多い。ちなみにこのスライドで上に書いてあるほど優先順位が高いので LC\_ALL と LANG があったら LC\_ALL が優先

# 言語設定の落とし穴

- ゲノム解析プログラムなど、研究の解析プログラムは英語圏の人が作っていることが多い
  - LANG=ja\_JP.utf-8 だとバグが出るプログラムも多い
  - LANG=C 等でやり直すと動くことも

# 環境変数 PATH

- 環境変数 **PATH** には : で区切られたディレクトリ名がたくさん格納されている。

```
bash-3.2$ echo $PATH
/Users/mkasa/lcl/bin:/Users/mkasa/.cabal/bin:/anaconda/b
in:/Users/mkasa/Library/Haskell/bin:/opt/local/bin:/usr/
/universal-darwin:/opt/x10/bin:/Users/mkasa/lcl/bin:/Use
r/local/sbin:/Users/mkasa/lcl/bin:/usr/local/bin:/usr/bi
usr/local/go/bin:/Users/mkasa/Library/Haskell/bin:/opt/l
universal-darwin:/opt/x10/bin
```

- **grep** など、コマンドを打った場合には **PATH** に登録されているディレクトリの直下を順に調べて最初に見つかったコマンドを実行する

# プログラムのサーチ例

- 前提条件
  - `PATH=/usr/bin:/bin:/usr/sbin:/sbin`
  - これから `grep` コマンドを実行する
- サーチ順
  - `/usr/bin/grep`, `/bin/grep`, `/usr/sbin/grep`, `/sbin/grep` を順に探し、最初に見つかった `grep` を実行する
- どれが実行されているか調べる
  - `which` コマンドで調べられる

```
bash-3.2$ which grep
/usr/bin/grep
```

# PATHの舞台裏

- 高速化のために、（厳密に言うとは違うんだけど）シェルが起動した瞬間に**PATH**に登録されている各ディレクトリの中にある全ての実行可能ファイルを読み込んでメモリー上に保持している（キャッシングしている）。
- **PATH** を変更したらメモリー上のリストも更新しないと不整合が発生することがある
  - `hash -r` コマンドでリストを更新する

# .bash\_profile 等への登録

- 例えば ~/local/bin に保存しているプログラムをコマンド名のみで実行できるようにするには ~/local/bin を PATH に登録すれば良い。
- .bash\_profile, .zshrc などのターミナルを開いたときに実行されるファイルに PATH への追加コマンドを記述する

- `export PATH=$HOME/local/bin:$PATH`

既存の PATH も残すようにするのがポイント

- チルダを使っても展開されない（チルダの前が半角スペース（区切り文字）でないと展開されない）
    - `export PATH=~/local/bin:$PATH` はよくある誤り



# 標準入力・標準出力・ 標準エラー出力

- \*NIX の全てのソフトウェアは「標準入力」と「標準出力」と「標準エラー出力」という入出力チャンネルを持つ
- パイプ、`A | B` は `A` の標準出力を `B` の標準入力に接続する、という指令になっている
- リダイレクト `A > file` は、`A` の標準出力を `file` につなぎ替える、という指令になっている

# エラーなどは 「標準エラー出力」に出す

```
[mkasa@zombie]~/test% ls -al 存在しないファイル  
ls: cannot access 存在しないファイル: そのようなファイルやディレクトリはありません  
[mkasa@zombie]~/test% ls -al 存在しないファイル > error.txt  
ls: cannot access 存在しないファイル: そのようなファイルやディレクトリはありません
```

リダイレクトしてもエラーメッセージはファイルに保存できていない

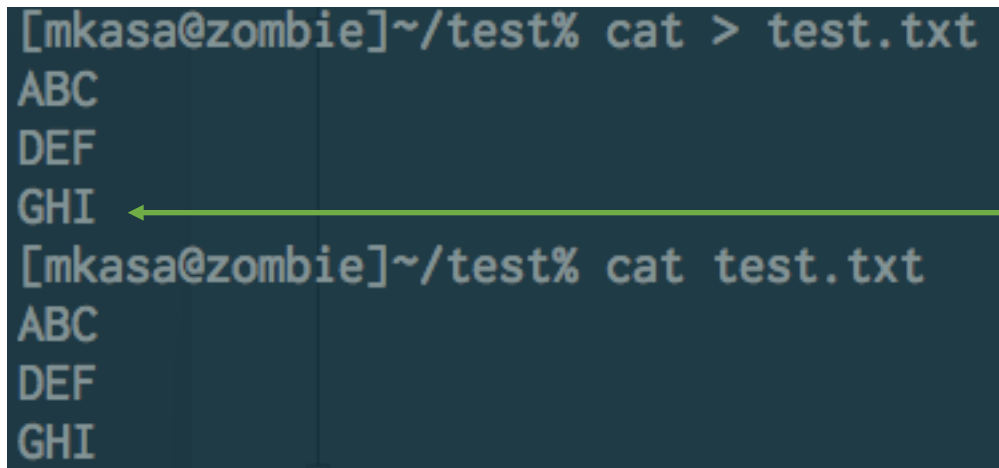
```
[mkasa@zombie]~/test% ls -al 存在しないファイル 2> error.txt  
[mkasa@zombie]~/test% cat error.txt  
ls: cannot access 存在しないファイル: そのようなファイルやディレクトリはありません
```

“**2**> ファイル”で「標準エラー出力」をリダイレクトすることが可能

# 標準入力

- 何も指定しないとキーボードからの入力が標準入力

```
[mkasa@zombie]~/test% cat > test.txt
ABC
DEF
GHI
[mkasa@zombie]~/test% cat test.txt
ABC
DEF
GHI
```



この3行をキーボードから打って最後に **CTRL+D** を押す

**CTRL+D** でキーボード入力の終了（標準入力を終端させる）を表している。

- キーボードからの入力はラインバッファリング、という方法で溜めてからプログラムに渡されている。
  - 一行入力するたびにプログラムに内容が渡されている。
  - **CTRL+D** を押したときに行頭なら標準入力終端となる。
  - 改行を送りたくないときのために、行頭以外で **CTRL+D** を押すとその時点での入力内容をプログラムに渡す決まり。

# grep で練習

```
[mkasa@zombie]~/test% grep 5  
foo  
bar  
baz  
high5  
high5  
abc
```

この行だけ **grep** が表示している。  
他の行はユーザーが入力した。

最後に行頭で **CTRL+D** を押している。

# 標準入力（続）

- **A | B** のようにパイプで繋がれば **B** の標準入力は **A** の標準出力に繋がる
- **A < file** のようにリダイレクトで標準入力を指定することもできる。
  - **A < file** は **cat file | A** と同じ。

# time

- `time /bin/sleep 10`

- `time` コマンドでコマンドの実行時間を計る
  - `Sleep` コマンドは指定した秒数停止するコマンドなので  
当たり前のように **10 秒**がカウントされている。
  - **0.001 秒**はコマンドの起動にかかる微少な時間。

```
[mkasa@zombie]~% time /bin/sleep 10
/bin/sleep 10  0.00s user 0.00s system 0% cpu 10.001 total
```

- 実行に長い時間のかかる計算などでよく使う
- ベンチマークを取るためにも便利

# sort & uniq

- `sort a.txt | uniq`
  - `a.txt` を辞書順にソートして、**連続する**同じ行を取り除いて表示する。

```
[mkasa@zombie]~/test% cat input.txt
USA
Japan
USA
Korea
China
Japan
France
USA
Japan
[mkasa@zombie]~/test% sort input.txt
China
France
Japan
Japan
Japan
Korea
USA
USA
USA
```

```
[mkasa@zombie]~/test% sort input.txt | uniq
China
France
Japan
Korea
USA
[mkasa@zombie]~/test% sort input.txt | uniq -c
  1 China
  1 France
  3 Japan
  1 Korea
  3 USA
```

# sort & uniq 試し切り

- UniPlot で肺ガンたんぱく質を検索して種別に数をカウントしてみる。

```
[mkasa@zombie]~/test% curl http://togows.org/search/uniprot/lung+cancer > lung_cancer_uniplot_list.txt
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %         0         0             0      0         0     0
102 1327    0 1327    0     0   1059      0  --:--:--  0:00:01  --:--:-- 1085
```

アンダースコアの後ろ側の文字列を集計して、例えば HUMAN は何回出ているか数えたい。

```
[mkasa@zombie]~/test% head lung_cancer_uniplot_list.txt
KKLC1_MACFA
KKLC1_HUMAN
DLEC1_HUMAN
ALDOA_HUMAN
RBM6_HUMAN
HOP_HUMAN
MED19_HUMAN
S38A9_HUMAN
NADAP_HUMAN
CQ080_HUMAN
```



sed を使ってアンダースコアとそれ以前を削除

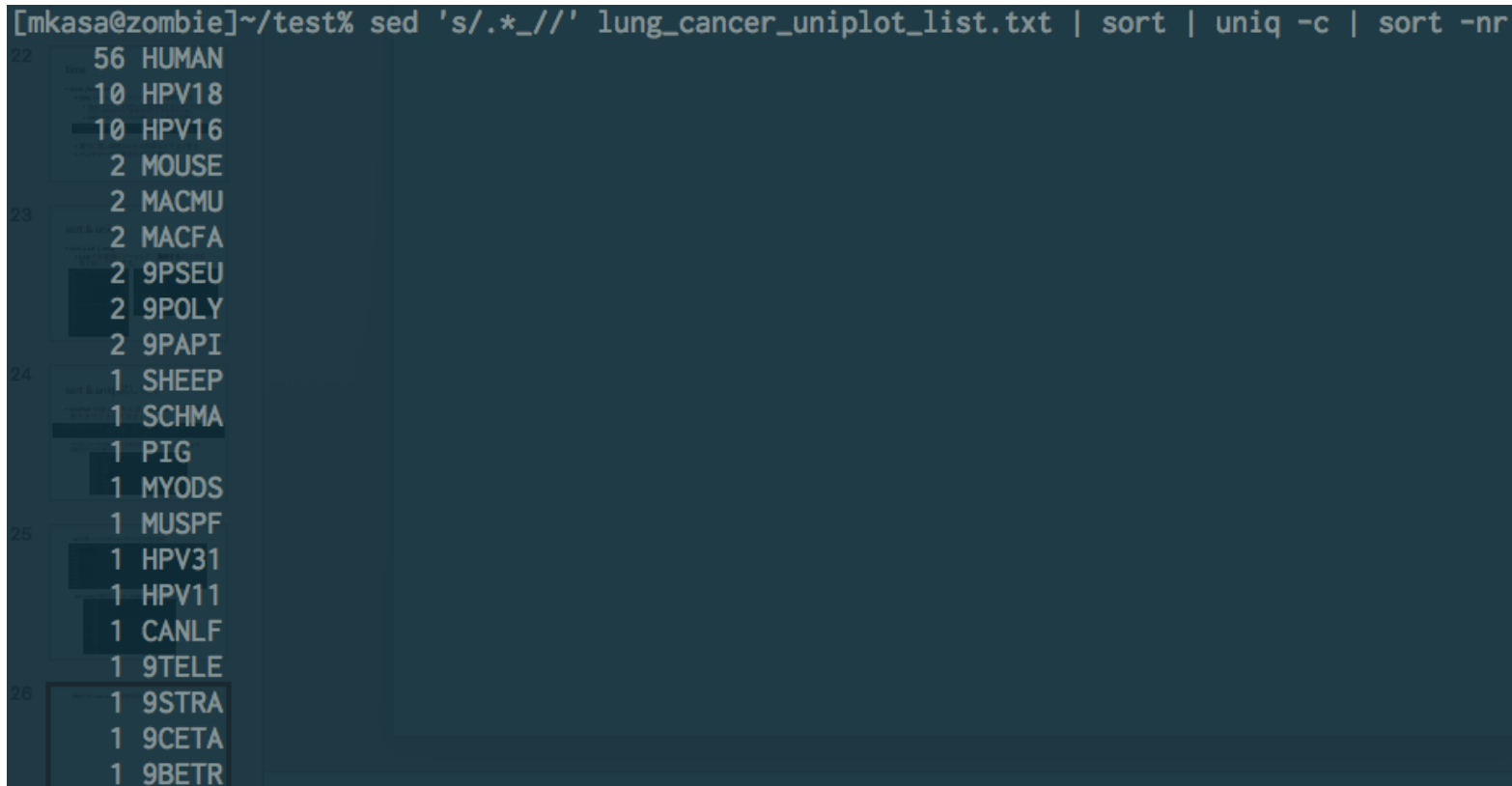
```
[mkasa@zombie]~/test% sed 's/.*_//' lung_cancer_uniplot_list.txt | head
MACFA
HUMAN
HUMAN
HUMAN
HUMAN
HUMAN
HUMAN
HUMAN
HUMAN
HUMAN
```

sort | uniq で集計してみる。HUMAN が 56 で多いですね。

```
[mkasa@zombie]~/test% sed 's/.*_//' lung_cancer_uniplot_list.txt | sort | uniq -c
  1 9BETR
  1 9CETA
  2 9PAPI
  2 9POLY
  2 9PSEU
  1 9STRA
  1 9TELE
  1 CANLF
  1 HPV11
 10 HPV16
 10 HPV18
  1 HPV31
 56 HUMAN
  2 MACFA
  2 MACMU
  2 MOUSE
  1 MUSPF
  1 MYODS
  1 PIG
  1 SCHMA
  1 SHEEP
```

結果を更に `sort -nr` で出現頻度順に並べる

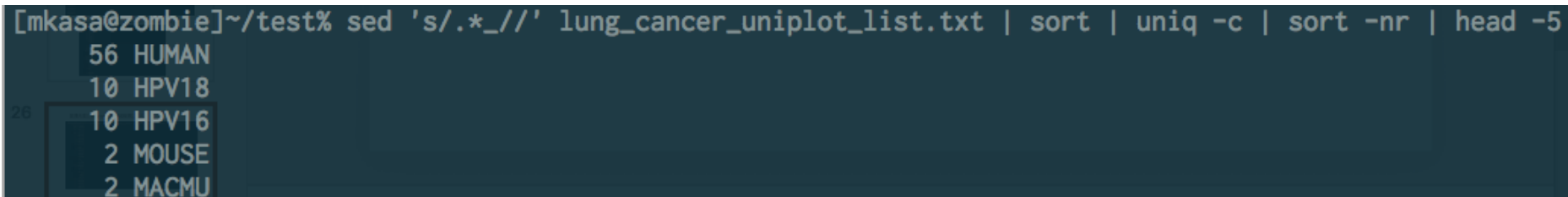
```
[mkasa@zombie]~/test% sed 's/.*_//' lung_cancer_uniplot_list.txt | sort | uniq -c | sort -nr
```



Count	Entry
56	HUMAN
10	HPV18
10	HPV16
2	MOUSE
2	MACMU
2	MACFA
2	9PSEU
2	9POLY
2	9PAPI
1	SHEEP
1	SCHMA
1	PIG
1	MYODS
1	MUSPF
1	HPV31
1	HPV11
1	CANLF
1	9TELE
1	9STRA
1	9CETA
1	9BETR

更に `| head` で Top 5 に絞ってみました。

```
[mkasa@zombie]~/test% sed 's/.*_//' lung_cancer_uniplot_list.txt | sort | uniq -c | sort -nr | head -5
```



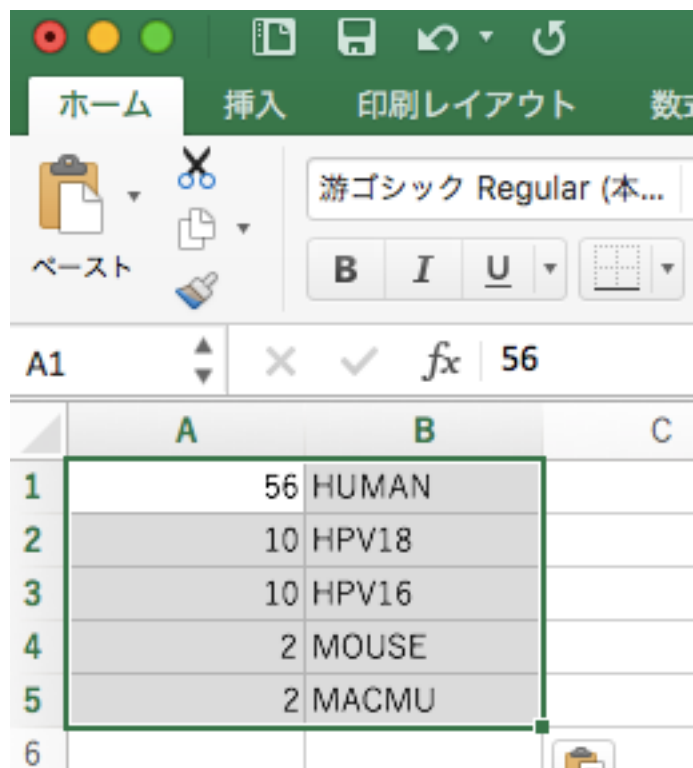
Count	Entry
56	HUMAN
10	HPV18
10	HPV16
2	MOUSE
2	MACMU

sed で行頭のスペースを除いて、区切りスペースをタブ文字に変更

```
[mkasa@zombie]~/test% sed 's/.*_//' lung_cancer_uniplot_list.txt | sort | uniq -c | sort -nr | head -5 | sed 's/^ *//; s/ /\t/g'
```

56	HUMAN
10	HPV18
10	HPV16
2	MOUSE
2	MACMU

Excelにそのまま  
コピー＆ペースト  
できるよ！



	A	B	C
1	56	HUMAN	
2	10	HPV18	
3	10	HPV16	
4	2	MOUSE	
5	2	MACMU	
6			

# せっかくなので 全部自動化してみよう！

- テキストエディタで以下のようなファイルを作成する。ファイル名は `rank_species.sh` とする。

```
#!/bin/bash
curl http://togows.org/search/uniprot/$1 \
| sed 's/^.*_//' \
| sort \
| uniq -c \
| sort -nr \
| sed 's/^ *///' \
| sed 's/ /\t/g' \
| head -5
```

\$1 はスクリプトの  
第 1 引数となる

行末にバッシュスラッシュを書くと長い行を分割できるよ！

chmod して実行可能にし、

```
[mkasa@zombie]~/test% chmod +x rank_species.sh
```

lung+cancer を引数にして実行してみよう。同じ結果が出るかな？

```
[mkasa@zombie]~/test% ./rank_species.sh lung+cancer
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload	Upload	Total	Spent	Speed
102	1327	0	1211	0	--:--:--	0:00:01	1232
56							
10							
10							
2							
2							

HUMAN  
HPV18  
HPV16  
MOUSE  
MACMU

コピー & ペーストして勿論 Excel に張れる。

“lung cancer” ではなくて “lung+cancer” としているのは URL に空白を使うことはできないため、空白の代わりに “+” を使うことになっているから。

完璧な変換ルールを知りたい人は以下のコマンドで。

```
[mkasa@zombie]~/test% echo -n "lung cancer" | php -r "echo urlencode(file_get_contents('php://stdin'))"; echo  
lung+cancer
```

```
[mkasa@zombie]~/test% echo -n "あ" | php -r "echo urlencode(file_get_contents('php://stdin'))"; echo  
%E3%81%82  
[mkasa@zombie]~/test% echo -n "+/" | php -r "echo urlencode(file_get_contents('php://stdin'))"; echo  
%2B%2F%25
```

スクリプトに組み込めばこのコマンドは “lung+cancer” ではなく “lung cancer” で検索できるね！

もちろん、単なる cancer でも検索できる。


```
[mkasa@zombie]~/test% ./rank_species.sh cancer
```

	% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current			
				Dload	Upload	Total	Spent	Left	Speed		
107	1179	0	1179	0	0	1046	0	--:--:--	0:00:01	--:--:--	1073
34											
23											
10											
5											
4											

HUMAN  
CANPG  
MOUSE  
DANRE  
XENTR

curl の標準エラー出力が鬱陶しいので消してみよう。

```
1 #!/bin/bash
2 dbcls.j
3 curl http://togows.org/search/uniprot/$1 2> /dev/null \
4 | sed 's/^.*_//' \
5 | sort \
6 | uniq -c \
7 | sort -nr \
8 | sed 's/^ *//' \
9 | sed 's/ /\t/g' \
10 | head -5
11
```



標準エラー出力を /dev/null というファイルにリダイレクトした。  
これでダウンロード統計は /dev/null というファイルに書き込まれる。

# /dev/null などの特殊デバイス

- /dev/null
  - 書き込まれた内容を捨てる特殊なファイル
  - 読み込もうとすると 0 バイトのファイルに見える
  - 何かを捨てたいときには /dev/null にリダイレクトする。
- /dev/zero
  - 1 6 進数で 00H が無限に湧いてくる特殊ファイル
  - 読み込もうとすると無限に 00H が読める
  - 書き込むと捨てられる
- /dev/urandom
  - ランダムな 1 6 進数が無限に湧いてくる特殊ファイル
  - ランダムなデータを作りたいときによく使う
  - 書き込んでみたことが無いので書き込んだらどうなるか知らない

curl の鬱陶しいダウンロード統計メッセージが出てこなくなった。  
ovarian+cancer で検索してExcel に張ってみる。

```
[mkasa@zombie]~/test% ./rank_species.sh ovarian+cancer
18      HUMAN
5        MOUSE
3        MACFA
3        9CRUS
3        9CARN
```

	A	B
1	18 HUMAN	
2	5 MOUSE	
3	3 MACFA	
4	3 9CRUS	
5	3 9CARN	

ラボセミナーのプレゼン資料も3分で作れちゃう（かも？）

極めると Excel の綺麗な表を作ってグラフを作成し、PowerPoint に張り込む、  
なんてところまでスクリプトで全自動にすることもできる。

（実際にはそこまでやるならグラフは R+ggplot2 とかで書いて下さい・・・）



# ターミナルからそれ以外に コピー & ペースト

- ターミナルのソフトウェア毎にやり方は違うので、よくある（多くのソフトウェアで採用されていて通用範囲の広い）やり方を紹介する

1) マウスカーソルをドラッグして範囲を選ぶと、選択範囲がなんとなくハイライトされる。

```
[mkasa@zombie]~/test% sed 's/.*_//' lung_cancer_uniplot_list.txt | sort | uniq -c | sort  
-nr | head -5 | sed 's/^ *//; s/ /\t/g'
```

56	HUMAN	16 HPV15
10	HPV18	2 MOUSE
10	HPV16	2 MACMU
2	MOUSE	
2	MACMU	

```
[mkasa@zombie]~/test%
```

2) Excel でペーストする（Mac なら ⌘V, Windows 等なら CTRL+V）。  
おしまい。

# ターミナルへのペースト

- ターミナルへペーストする方法も  
ターミナルのソフトに依存している。  
(ターミナルのソフトを開発した人が  
方法を決めている。)
- よくあるペースト手段としては  
「マウスを右クリック」  
「マウスの左右ボタンを同時クリック」  
「マウスの中央ボタンをクリック」  
がある。  
(中央ボタンは 3 ボタンマウスの場合のみ)  
(左右同時クリックは 2 ボタンマウスの場合のみ)

# GUI ソフトウェアとの連携

- open コマンドでファイルを開く
  - Mac のファインダーや Windows のエクスプローラーなどで「ファイルをクリックする」という動作は open コマンドで実現できる
  - open <http://www.yahoo.co.jp/>  
とすると多分ブラウザが立ち上がって Yahoo! が表示される
  - open table.xls  
とすると多分 Excel が立ち上がって table.xls が開く
  - open paper.docx  
とすると多分 Word が立ち上がって paper.docx が開く
  - open nature\_paper.pdf  
とすると多分 pdf ビューアーが立ち上がる

# GUI ソフトウェアとの連携 2

- **pbcopy** (Mac OS X のみ)
  - 標準入力を受け取って内容をクリップボードにコピーする。さっきの `rank_spicies.sh` もこれがあれば一手間減りますね。
- **pbpaste** (Mac OS X のみ)
  - クリップボードの内容を標準出力に出すコマンド。
- **Linux** の人は **xsel** (英語しか使わないなら **xclip** でも) などのソフトウェアを追加でインストールすると同様のコマンドが使える。

# パッケージマネージャ

- ソフトウェアを管理するソフトウェア
  - 普通は1コマンド（または1クリック）で簡単にソフトウェアをインストールできる
  - ソフトウェアの削除や更新にも普通は対応している
- パッケージマネージャの例
  - yum (RHEL, CentOS, Fedora, Scientific Linux, etc.)
  - zypper (SLES)
  - apt (Ubuntu, Debian)
  - Chocolatey (Windows)
  - pacman (Arch)
  - emerge (Gentoo)
  - Applistation (Windows)
  - Homebrew (MacOS X)
  - Ports (FreeBSD, NetBSD, OpenBSD)
  - MacPorts (MacOS X)

# 普通のソフトウェアインストール法

## Apache2 (Web サーバー) をインストールする例

Red Hat Enterprise Linux の場合

```
# yum install httpd
```

Ubuntu Linux の場合

```
# apt-get install apache2
```

MacOS X の場合 (homebrew を別途インストールしておく)

```
# brew tap homebrew/dupes  
# brew install homebrew/apache/httpd22
```

# yum (RHEL 等で使われている)

・rpm パッケージのダウンロードと インストールを自動化する  
ソフトウェア(管理者権限要)

多くの Linux ディストリビューションでは、  
多くのパッケージを集めた「リポジトリ」と  
呼ばれるサーバが用意されている。

インストール

```
[root@zombie tmp]# yum install mosh
```

アップデート(バージョンアップ)

```
[root@zombie tmp]# yum update mosh
```

全部アップデート

```
[root@zombie tmp]# yum update
```

パッケージのサーチ

```
[root@zombie tmp]# yum search mosh
```

1) yum install mosh

2) mosh の rpm を  
ダウンロード

例) CentOS 用の  
rpm リポジトリサーバ

yum

3) rpm -ivh で  
インストール

rpm

※rpm を使っているので管理者権限が必要な事に注意。  
必要なら sudo や su コマンドを用いて管理者権限で作業する。

# Dependency (依存関係)

- AAA というパッケージを入れるためには BBB というパッケージを入れる必要がある。
- BBB というパッケージを入れるためには CCC というパッケージを入れる必要がある。
- . . .

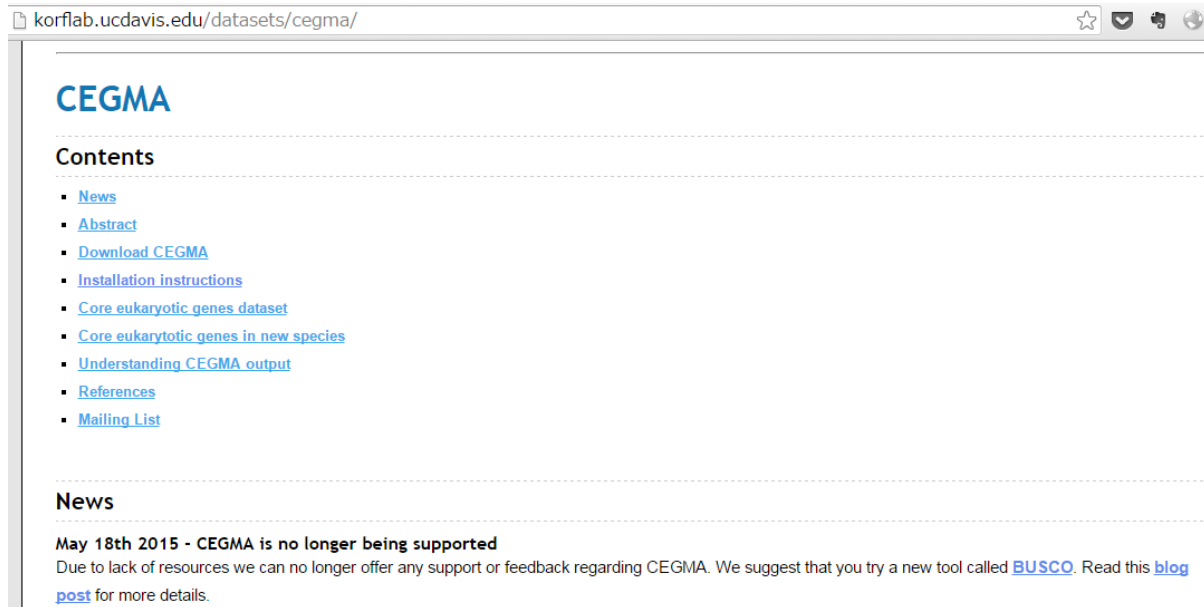
“yum install AAA” コマンドを用いる場合には、リポジトリ中から BBB や CCC も探し出して同時にインストールしてくれるので非常に便利。リポジトリ中に、AAAのインストールに必要なパッケージが足りない場合にはエラーとなる。

RedHat 系列以外の場合には yum の代わりに apt-get, zypper , emerge, pacman 等、yum と同様のパッケージマネージャが用意されていることが多い。



# 研究用の（駄目な） ソフトウェアのインストール

- ・パッケージシステムなんて研究者は使い方を知らない
- ・パッケージシステムでインストールできるように作っていない
- ・インストール手順はソフトによって異なっている
- ・ハンズオンタイム（手を動かす時間＝放っておけない時間）が長い



※著者の名誉のために言っておきますが、インストールが大変なのでもう使うなと言ってます



**CEGMA HAS BEEN DISCONTINUED**

**We suggest you consider using alternative tools**

# 研究用のソフトウェア達をインストールする際の問題点

- rpm パッケージのように便利な手段が提供されていることはほとんど無い
  - NCBI-BLAST のように、大手が開発しているソフトウェアではまれに rpm のようなバイナリパッケージが提供されている。
  - 単独の研究室で作っているソフトウェアは rpm で提供されていることはほぼあり得ない。
- **Dependency Hell**
  - rpm が無い、ということはソフトウェアの依存関係を自分で処理する必要がある。
  - ソフトウェアを一つインストールするためだけに、そのソフトウェアの動作に必要な 10 以上のソフトウェアを入れる必要があることも珍しくない。
  - 依存関係が複雑なソフトウェアはインストールに丸 1 日掛かっていても不思議ではない。一ヶ月潰した話もよく聞く。

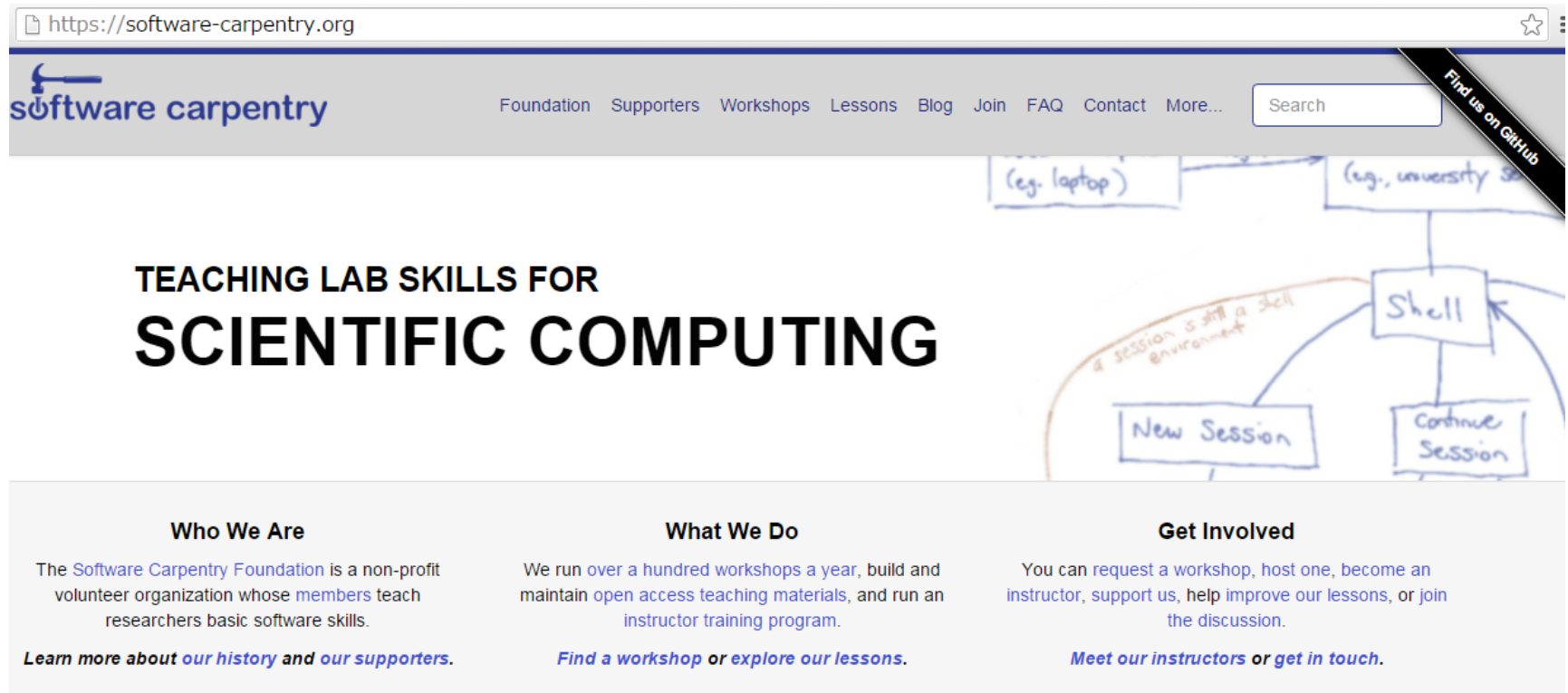
# なぜ研究用のソフトウェアは インストール・利用が難しいの か？

- 研究用のソフトウェアを作っている人達（主に大学院生・ポスドク）がソフトウェアエンジニアリングのトレーニングを受けていない。独学もしていない。
- ソフトウェアのお作法を何も知らないのでもいいの？



- 情報解析が生命科学でこれだけ重要になった世の中では、大学院生やポスドクもソフトウェアエンジニアリングのトレーニングを受けるべき！

# Software Carpentry



The screenshot shows the Software Carpentry website. At the top is a navigation bar with links: Foundation, Supporters, Workshops, Lessons, Blog, Join, FAQ, Contact, and More... There is also a search bar and a 'Find us on GitHub' button. The main heading reads 'TEACHING LAB SKILLS FOR SCIENTIFIC COMPUTING'. To the right of the heading is a hand-drawn diagram showing a flow from '(eg. laptop)' and '(eg. university server)' to a central box labeled 'Shell'. From 'Shell', arrows point to 'New Session' and 'Continue Session'. A curved arrow from 'New Session' back to 'Shell' is labeled 'a session is still a shell environment'. Below the heading are three columns: 'Who We Are' (describing the non-profit foundation), 'What We Do' (describing workshops and training), and 'Get Involved' (describing how to request workshops or become instructors). Each column has a link to learn more.


https://software-carpentry.org

software carpentry

Foundation Supporters Workshops Lessons Blog Join FAQ Contact More... Search

Find us on GitHub

## TEACHING LAB SKILLS FOR SCIENTIFIC COMPUTING



**Who We Are**

The [Software Carpentry Foundation](#) is a non-profit volunteer organization whose [members](#) teach researchers basic software skills.

[Learn more about our history and our supporters.](#)

**What We Do**

We run [over a hundred workshops a year](#), build and maintain [open access teaching materials](#), and run an [instructor training program](#).

[Find a workshop or explore our lessons.](#)

**Get Involved**

You can [request a workshop](#), [host one](#), [become an instructor](#), [support us](#), [help improve our lessons](#), or [join the discussion](#).

[Meet our instructors or get in touch.](#)

## A Practical Computing Course

By Steve Haddock / 2015-11-15

While it was not a SWC course, this summer Casey Dunn and I taught a 12-day [Practical Computing](#) summer class at Friday Harbor Labs. It was a great group of students—mostly somewhat beginner level—and we covered regular



**DATA CARPENTRY**  
MAKING DATA SCIENCE MORE EFFICIENT

# UNIX (& Linux) 界隈での標準的なソフトウェアインストール手順

- **GNU autotools** というソフトウェアを用いてパッケージングしたものが事実上の標準となっている
  - しかし、**autotools** はひどい。教えたくない。
    - 前世代の遺物
  - **autotools** でパッケージされたソフトウェアの使い方だけをここでは教えよう。
  - **autotools** でパッケージする方法は教えたくない。

# ソースコードからのインストール

- ソースコードのダウンロード
  - 例) SAMtools

[Home](#) / [Browse](#) / [Science & Engineering](#) / [Bio-Informatics](#) / [SAM tools](#) / [Files](#)



## SAM tools

Brought to you by: [bhandsaker](#), [jmarshall](#), [lh3lh3](#), [petulda](#)

[Summary](#) | [Files](#) | [Reviews](#) | [Support](#) | [Wiki](#) | [Mailing Lists](#) | [Hosted Apps](#) ▾ | [Code](#)

Looking for the latest version? [Download samtools-0.1.19.tar.bz2 \(514.5 kB\)](#)

[Home](#) / [samtools](#) / 0.1.19

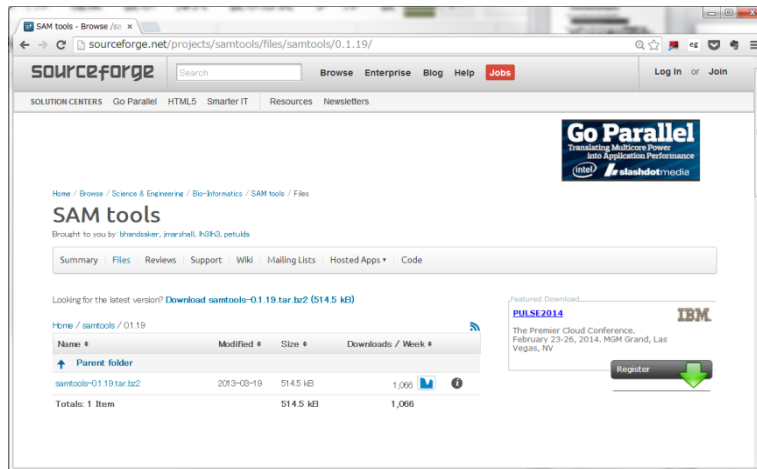
Name ▴	Modified ▴	Size ▴	Downloads / Week ▴
<a href="#">↑ Parent folder</a>			
<a href="#">samtools-0.1.19.tar.bz2</a>	2013-08-19	514.5 kB	1,066  
Totals: 1 Item			

Click to download samtools-0.1.19.tar.bz2

ソースコードは “.tar.gz”, “.tar.bz2”, “.tar.xz” など、tarを用いて複数のファイルをまとめた「tar アーカイブ」で配布するのが一般的。“tar ball” と呼ばれる。

# ソースコードをダウンロード

- (1)手元の端末(Win/Mac/Linux/...)で SAMtools の web ページを開く
- (2)右クリック等のメニューから「リンクをコピー」する。



- (3) TeraTerm (右クリック), PuTTY (右クリック), Terminal.app (CMD+V) 等の端末にアドレスをコピーしてファイルをダウンロードする。

```
[mkasa@t352]~$ wget http://sourceforge.net/projects/samtools/files/samtools/0.1.19/samtools-0.1.19.tar.bz2/download
```

wget はダウンロードするソフト  
curl -O でも同じ。

‘?’ や ‘&’ が含まれるURLは ‘’ で括らないと  
“no matches” エラーが発生するので注意。

# tarball を解凍する

- “tarball” には複数のファイルが格納されているので「解凍」と呼ばれる操作でファイルを取り出す。

tar.gz の場合

```
% tar xvfz samtools-0.1.19.tar.gz
```

tar.bz2 の場合

```
% tar xvfj samtools-0.1.19.tar.bz2
```



# 付属の説明書を読む

- たいていのソフトウェアには「説明書」が付属している。「説明書」は面倒でも必ず読むべし。

INSTALL や Install.txt、  
README、README.txt など、  
ソフトウェアをインストールし  
て使うために必要な手順が書  
かれたファイルは熟読しよう。

NEWS/AUTHORS/COPYING/Ch  
angeLog などは読まずとも問  
題は少ない。ChangeLog には  
変更履歴が書かれているの  
で、バージョンアップで問題が  
解決しているか等を知る手が  
かりになる。

```
[mkasa@t262]~% cd samtools-0.1.19
[mkasa@t262]~/samtools-0.1.19% ls
AUTHORS          bam_index.c      bedidx.c         ksort.h
COPYING          bam_lpileup.c    bgzf.c           kstring.c
ChangeLog.old    bam_mate.c       bgzf.h           kstring.h
INSTALL          bam_md.c         bgzip.c          misc
Makefile         bam_pileup.c     cut_target.c     padding.c
Makefile.mingw   bam_plcmd.c     errmod.c         phase.c
NEWS            bam_reheader.c  errmod.h         razf.c
bam.c            bam_rmdup.c     examples         razf.h
bam.h            bam_rmdupse.c   faidx.c          razip.c
bam.h~          bam_sort.c       faidx.h          sam.c
bam2bcf.c        bam_stat.c       kaln.c           sam.h
bam2bcf.h        bam_tview.c     kaln.h           sam_header.c
bam2bcf_indel.c bam_tview.h     khash.h          sam_header.h
bam2depth.c      bam_tview_curses.c klist.h          sam_view.c
bam_aux.c        bam_tview_html.c knetfile.c       sample.c
bam_cat.c        bamshuf.c       knetfile.h       sample.h
bam_color.c      bamtk.c         kprobaln.c       samtools.1
bam_endian.h     bcftools        kprobaln.h       win32
bam_import.c     bedcov.c        kseq.h
```

# Autotools を用いて作成された tarball のインストール手順

- tarball に “configure” というファイルが含まれていれば autotools を用いて作成されている（正しくは、autotools を用いて作成されたと見なして操作して構わない。）
- tarball をダウンロード (k-mer 解析用ソフト Jellyfish の例)

```
% wget 'http://www.cbcb.umd.edu/software/jellyfish/jellyfish-1.1.11.tar.gz'
```

- 解凍し、ファイルが存在するディレクトリに移動

```
% tar xvfz jellyfish-1.1.11.tar.gz  
% cd jellyfish-1.1.11
```

- 依存ライブラリ等のチェック

```
% ./configure
```

- コンパイル

```
% make
```

- インストール

```
# make install (管理者権限以外で操作しているときには sudo make install)
```

# 管理者権限が無いとき

- Autotools で正しく作成されたツールであれば  
./configure に --prefix オプションを付ければ O K

samtools を curl でダウンロード・・・、しようと思ったが  
ブラウザー経由でないとダウンロードできないように今は意地悪  
されているらしいのでブラウザーでクリックしてダウンロード。

https://sourceforge.net/projects/samtools/files/

**sourceforge** Search Browse Enterprise Blog Deals

SOLUTION CENTERS Go Parallel Resources Newsletters Cloud Storage Providers

Home / Browse / Science & Engineering / Bio-Informatics / SAM tools / Files

**SAM tools**  
Brought to you by: bhandsaker, jmarshall, lh3lh3, petulda

Summary Files Reviews Support Wiki Mailing Lists Code

Looking for the latest version? [Download samtools-1.3.tar.bz2 \(4.0 MB\)](#)

Home

Name	Modified	Size	Downloads / Week
<a href="#">samtools</a>	2015-12-15		2,198
<a href="#">tabix</a>	2012-06-18		375
<a href="#">glftools</a>	2009-05-08		1

Totals: 3 Items

ファイルを適当に解凍

```
bash-3.2$ pwd
/Users/mkasa/local
bash-3.2$ tar xvfj ~/Downloads/samtools-1.3.tar.bz2
```

configure に prefix オプションを付ける。

```
bash-3.2$ cd samtools-1.3/
bash-3.2$ ./configure --prefix=$HOME/local
```

--prefix オプションのデフォルト値は /usr である。

--prefix=/home/mkasa/local とすると、/usr/bin/samtools にインストールされる予定だったファイルは /home/mkasa/local/bin/samtools にインストールされるようになる。

```
bash-3.2$ make && make install
```

\$HOME/local/bin/samtools で samtools が動く。

絶対パス指定が面倒なら PATH に \$HOME/local/bin を入れる（「PATHを通す」と言う）などする。スタートアップスクリプトにも多分登録したい。

```
bash-3.2$ $HOME/local/bin/samtools --version
samtools 1.3
Using htslib 1.3
Copyright (C) 2015 Genome Research Ltd.
```

# ssh

- `ssh abc@host.example.com`
  - `host.example.com` のマシンに遠隔ログインする
    - ユーザー名は `abc` とする
    - 今使っているマシンのユーザー名と同じなら `abc@` の部分は省略してよい

# rsync, scp

- `rsync -av /abc/def /volume/ext HDD`
  - `/abc/def` 以下の全ディレクトリ・ファイルを `/Volume/ext HDD` 以下にコピーする。  
ただし、既にコピーされているファイルはコピーしない。
  - バックアップディスクにファイルをコピーするときなどに良く使う。
- `rsync -av -rsh=ssh /abc/def abc.example.com:/home/foo`
  - `/abc/def` 以下の全ディレクトリ・ファイルを `abc.example.com` に `ssh` でログインして `/home/foo` にコピーする。
- `scp -r /abc/def xyz@abc.example.com:/home/user`
  - `/abc/def` ディレクトリ以下全ファイル・ディレクトリを遠隔マシン `abc.example.com` の `/home/user` にコピーする。ユーザー名は `xyz` である。現在のユーザー名と同じなら `xyz@` の部分は省略可能