UNIX基礎

基礎生物学研究所

ゲノムインフォマティクス・トレーニングコース
2016秋

入門編・UNIX・R・NGSの基礎

三輪朋樹（miwa@nibb.ac.jp）

UNIXを使う理由

- UNIXでしか使えないアプリケーション
  - 最新の研究用ソフト
  - 並列化・大容量メモリ対応ソフト
- たくさんの処理を一度に行う
  - スクリプトを用いたコマンドの連続実行
- 自作プログラム
  - シェルスクリプト, Perl, Ruby, バイオ系ライブラリ
- Webサーバ、データベースサーバ
  - 高い安定性
  - apacheやmySQL, Postgresなどのフリーウェア

PCでUNIXを使うには

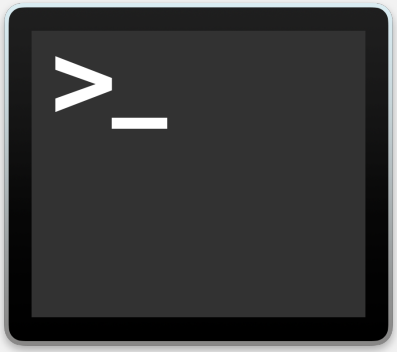
Mac	OSX自体がUNIX (#1)	アプリケーション→ターミナルを起動 UNIX端末として使用できる
	リモートログイン	UNIXサーバへリモートログイン ターミナルからsshを使用する
Windows	Cygwin	Windows上で動作するUNIXライクな環境
	VMware + Linux	仮想マシンを構築してLinuxそのものをインストールする
	リモートログイン	UNIXサーバへリモートログイン TeraTermからsshを使用する

#1) フリーウェアなどのインストールが必要な場合は「OSXでのUNIX環境構築方法」を参照

実習１

- OSXのUNIX環境を確認する

１．画面最下部にあるDockメニューから「ターミナル」を起動する。



（ターミナルの在処は、アプリケーション/ユーティリティ）

講習を始める前に

- **コマンドプロンプト**
  - 端末に表示されている”$”や”%”などの記号

    今回の環境は `dh00-216:~ nibb$`
  - コマンド入力待ちの状態を表す

    続けてコマンドを入力し、改行キーで実行する
- **ASCII文字**
  - コマンドの入力は全て半角のASCII文字を使用
  - 入力文字が全角になる日本語IMEはOFFにする

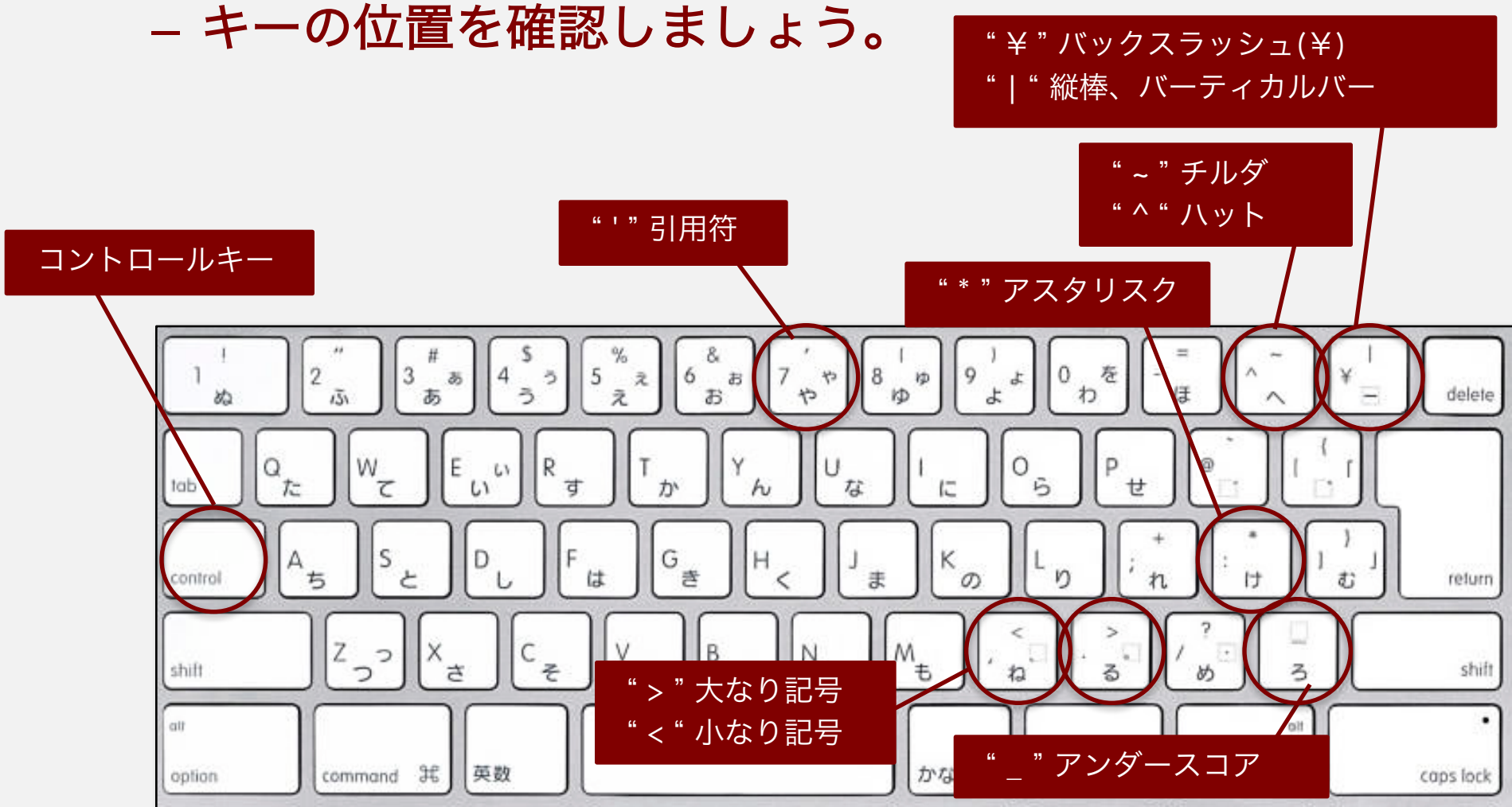
講習を始める前に

- 本講習では次のディレクトリを使用します

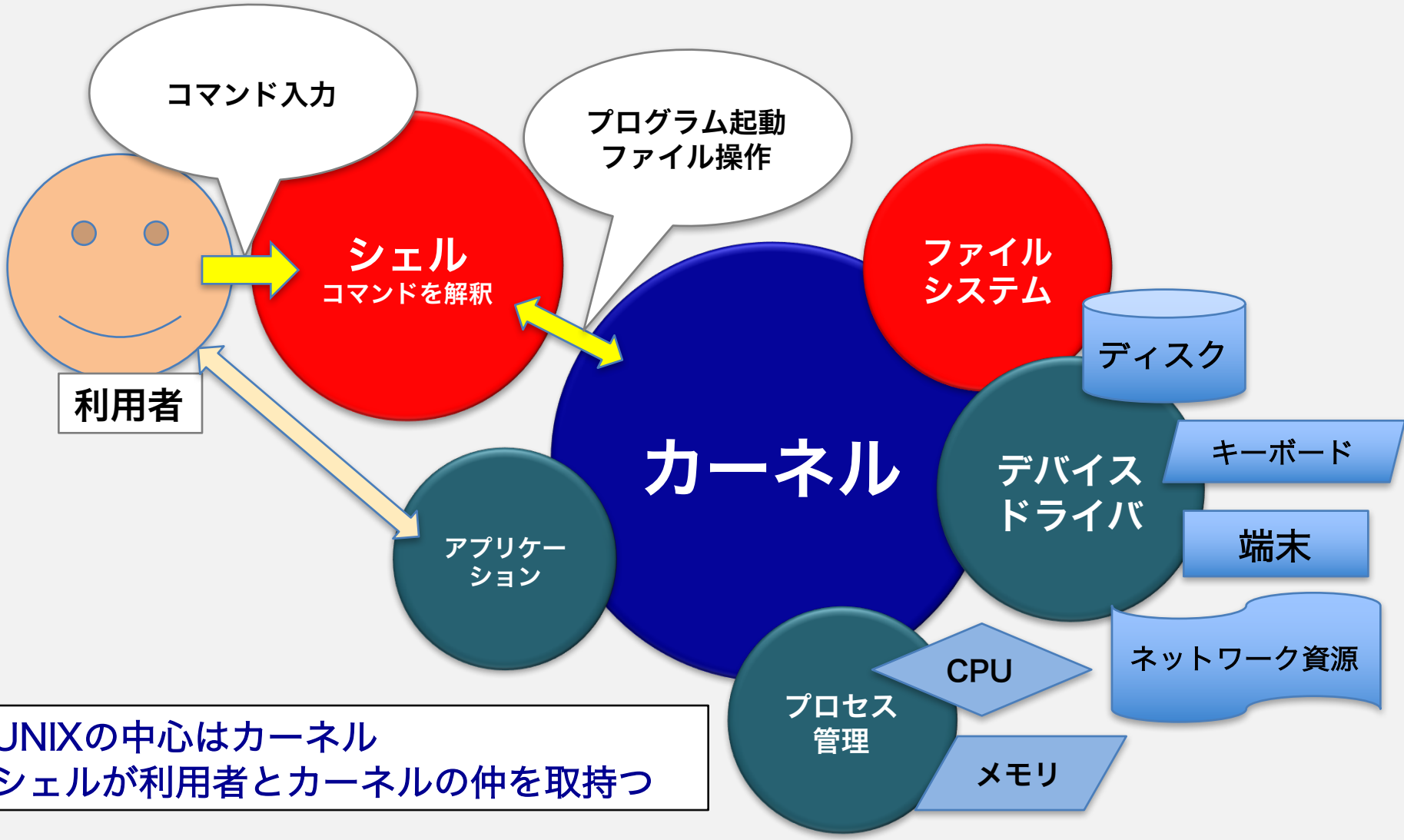
```
~/data/1_unix

~/unixtest
```

キーボード配置の確認

- 普段使用しない記号を多用します。
  - キーの位置を確認しましょう。



UNIXオペレーティングシステム



- UNIXの中心はカーネル
- シェルが利用者とカーネルの仲を取持つ

ファイルシステム

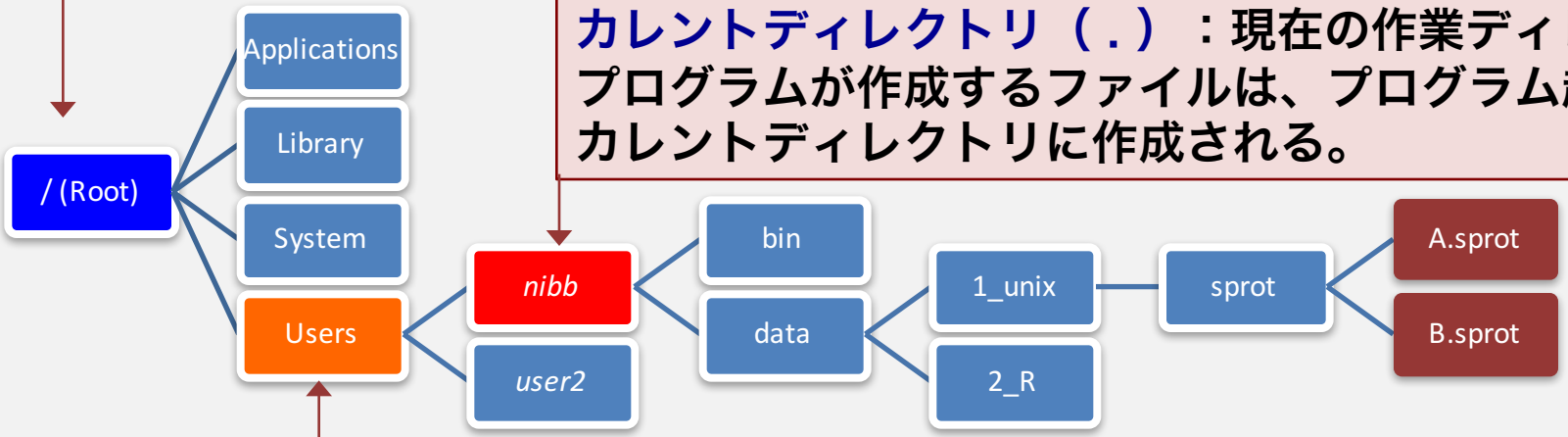
階層型ディレクトリ

トップのルートディレクトリ下に、子ディレクトリ、孫ディレクトリがあり、ファイルを配置する

ルートディレクトリ（ / ）：ファイルシステムの頂点

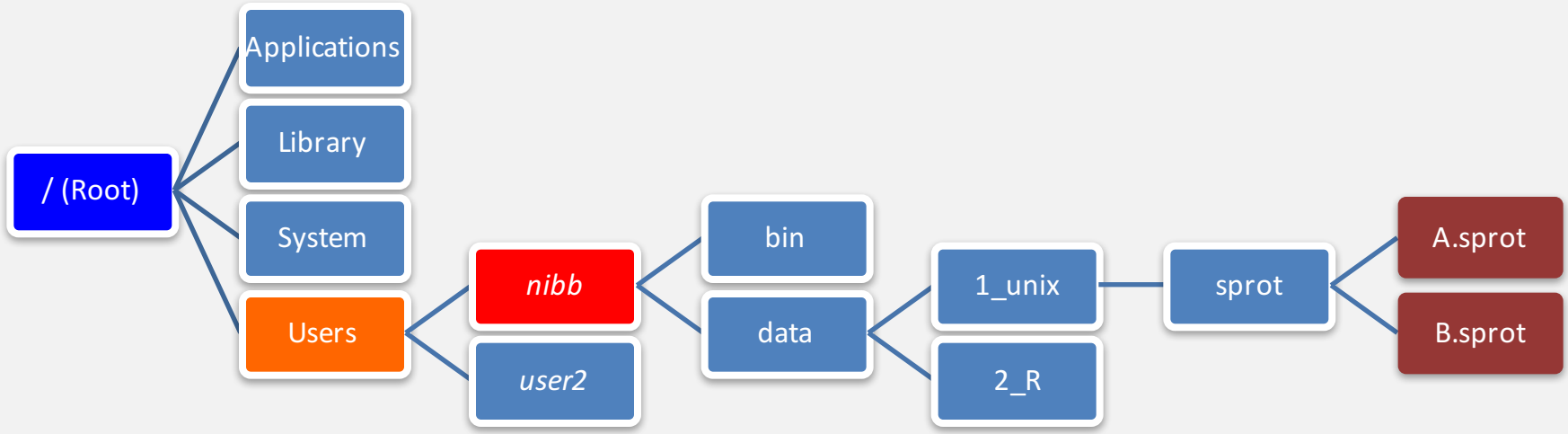
ホームディレクトリ（ ~ ）：個々のユーザ専用ディレクトリ、ログイン直後最初に位置するディレクトリ

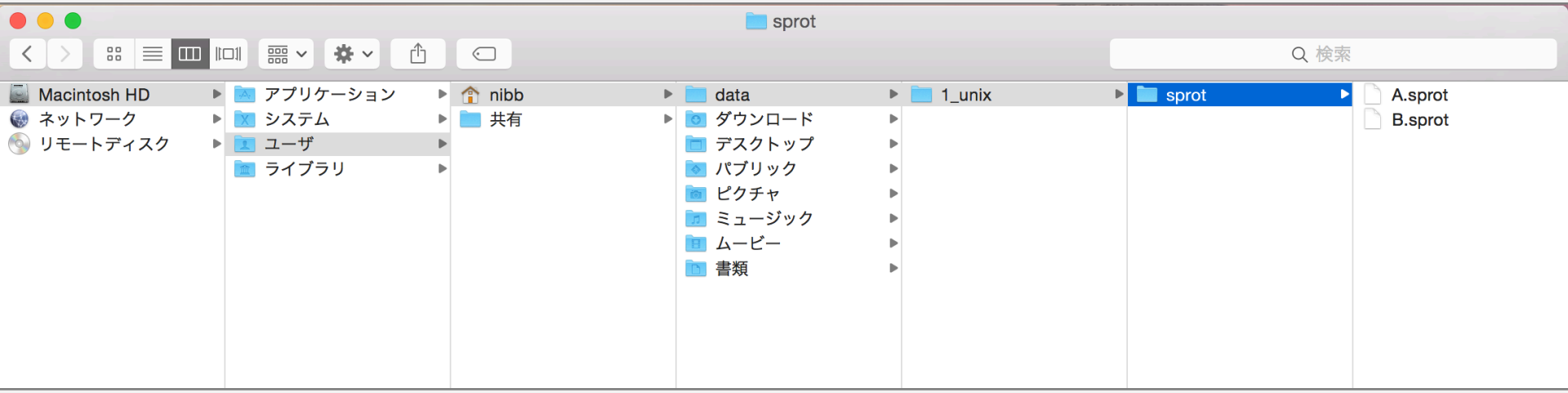
カレントディレクトリ（ . ）：現在の作業ディレクトリ、プログラムが作成するファイルは、プログラム起動時のカレントディレクトリに作成される。



親ディレクトリ（ .. ）：カレントディレクトリのひとつ上のディレクトリ。ドット２つで表記する。

階層型ディレクトリ


/ (Root)
Applications
Library
System
Users
nibb
user2
bin
data
1_unix
2_R
sprot
A.sprot
B.sprot


sprot
検索
Macintosh HD
ネットワーク
リモートディスク
アプリケーション
システム
ユーザ
ライブラリ
nibb
共有
data
ダウンロード
デスクトップ
パブリック
ピクチャ
ミュージック
ムービー
書類
1_unix
sprot
A.sprot
B.sprot

ファイル／ディレクトリの指定方法

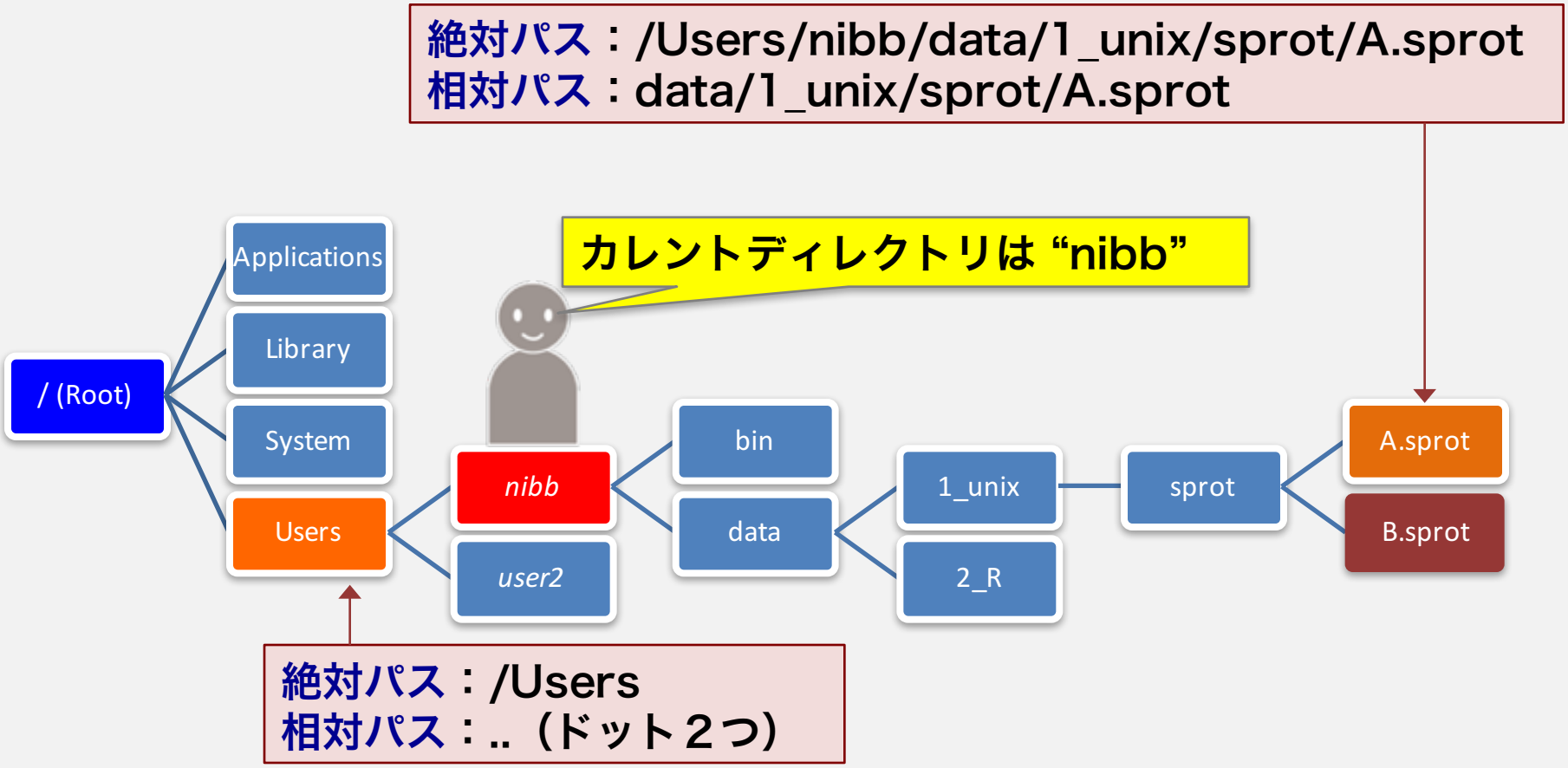
- **パス（Path）**
  - ファイル　やディレクトリを指定する記述方法
  - ディレクトリをスラッシュ”/ ”で区切る
  - 「絶対パス」と「相対パス」２通りの記述方法
- **絶対パス**
  - ルートディレクトリから目的のファイルやディレクトリへの道筋の記述
  - 行頭は必ずルートディレクトリ ”/“ となる

    例）`/Users/nibb/data/1_unix/sprot/A.fasta`
- **相対パス**
  - 起点となる現在位置から目的のファイルやディレクトリへの道筋の記述
  - 行頭はスラッシュ“ / “ 以外で始まる
  - 例）data/1_unix/sprot/A.fasta（カレントディレクトリは /Users/nibb）
  - 上位ディレクトリはドット２つ“ .. ”で記述する

    例）`../user2`（同じ階層の `user2` ディレクトリ）

ファイル／ディレクトリ名の指定方法

nibb がログイン直後 ： Users と A.sprot のパス



ディレクトリの中身を見る（ls）

- ls
  - カレントディレクトリの内容（ファイル名のリスト）を表示する
- ls ディレクトリ名
  - 指定したディレクトリの内容を表示する
    - `$ ls data` dataディレクトリの内容を表示
    - `$ ls /` ルートディレクトリの内容を表示
    - `$ ls ..` ひとつ上のディレクトリの内容を表示
    - `$ ls .` カレントディレクトリの内容を表示（lsと同じ）
- ls –F
  - ファイル名の末尾に種類に応じた記号を付けて表示する
    - `/` : ディレクトリ、`@` : シンボリックリンク、`*` : 実行権付きファイル
- ls –a
  - ファイル名の先頭がドット（ .）で始まる隠しファイルを表示する
    - `.login` : ログイン時に実行される処理を記述したファイル
    - `.bash_profile,.bashrc` : シェル起動時に実行される処理を記述したファイル

実習2

● ディレクトリの中身を見る (ls)

1. ホームディレクトリ上で**ls**と入力してリターンキーを押す（＝実行）

2. 続いて下記のコマンドもそれぞれ実行する。

   `$ ls ..`　　ホームディレクトリのひとつ上の中身を見る

   `$ ls /`　　ルートディレクトリの中身を見る

3. 隠しファイルを表示する。

   `$ ls -a`

補完機能

- コマンド名やファイル／ディレクトリ名の補完
  - ファイル名を途中まで入力して、タブキーを押す
    “`ls u`” と入力して、タブキーを押す
  - 一意に決まらない場合は、一意になるところまでを展開する
    “`ls data/1_unix/sprot/143`”と入力後タブキーを押す
  - それ以上展開できない場合は再度（つまり２回）タブキーを押すと、候補ファイルの一覧を表示する
    “`ls data/1_unix/sprot/1433`”の状態で２回続けてタブキーを押す
- 以下のメリットがあるので積極的に活用すること
  - キーボード入力を省略できる
  - 素早い入力が可能
  - 複雑なファイル名の入力ミス防止
  - うろ覚えのファイルでも入力できる
  - コマンド名も補完できる

実習3

ファイル名の補完

1. ホームディレクトリ上で `ls da` と入力後、`<tab>`（タブキー）を押してファイル名の補完機能を試してみる。
2. 続けて `ls data/1_unix/sprot/143` まで補完機能を使って動作を確認する。

```
$ ls data/1 <tab>
$ ls data/1_unix/
$ ls data/1_unix/sp <tab>
$ ls data/1_unix/sprot/
$ ls data/1_unix/sprot/143 <tab>
$ ls data/1_unix/sprot/1433
$ ls data/1_unix/sprot/1433 <tab><tab>
.... (候補一覧表示)
$ ls data/1_unix/sprot/1433B_H <tab>
$ ls data/1_unix/sprot/1433B_HUMAN.
$ ls data/1_unix/sprot/1433B_HUMAN.f <tab>
$ ls data/1_unix/sprot/1433B_HUMAN.fasta
```

コマンドヒストリと編集

●過去に実行したコマンドの呼び出しと編集

キーバインド	説明
Control + p (↑キー)	ひとつ前のコマンド (Previous)
Control + n (↓キー)	ひとつ後のコマンド (Next)
Control + b (←キー)	カーソルを左に移動 (Back)
Control + f (→キー)	カーソルを右に移動 (Forward)
Control + a	カーソルを行頭に移動 (stArt)
Control + e	カーソルを行末に移動 (End)
Control + u	行全体を削除しバッファへコピー
Control + k	カーソルから後を削除しバッファへコピー
Control + y	バッファの内容をペースト
Control + l (エル)	現カーソル行を画面上部に移動、画面消去

ファイル名のみ変更するなど長い入力行の一部を修正できるので、活用すること

実習4

● コマンドヒストリ

1. **Control + P** （コントロールキーと**P**を同時に押す）を何度か入力する。
2. **Control + N** を何度か入力する。
3. 表示されたコマンドラインのカーソル移動を行う。

   Control + B

   Control + F

   Control + A

   COntrol + E

ディレクトリを移動する（cd）

- cd ディレクトリ名
  - 指定したディレクトリに移動する
  - カレントディレクトリの変更

```
$ cd data       dataディレクトリに移動
$ cd ..         ひとつ上のディレクトリに移動
$ cd ~/data     ホーム下のdataディレクトリに移動
```

- cd
  - ディレクトリ名を省略すると、ホームディレクトリに移動する
- pwd
  - カレントディレクトリの確認

実習5

●ディレクトリを移動する

1. カレントディレクトリを確認する。

```
$ pwd
```

2. ディレクトリ**data/1_unix**に移動して**pwd**を入力する。

```
$ cd data/1_unix
$ pwd
$ ls
```

3. **sprot**ディレクトリに移動し、カレントディレクトリを確認する。

```
$ cd sprot
$ pwd
```

ワイルドカード

- ファイル名を指定するときに使う「任意の文字」
- パターンマッチにより複数のファイル名を一度に指定
  - アスタリスク（＊）：任意の文字列（０文字以上）

    ```
    $ ls *.fasta
    $ ls *_HUMAN*
    ```

  - クエスチョン（？）：任意の１文字

    ```
    $ ls 1A2?_HUMAN.fasta
    ```

  - [文字列]：[ ]に含まれる文字中の１文字

    [12345]は[1-5]と書くこともできる

    [!文字列]で含まれない１文字

    ```
    $ ls 1A2[1-5]*.fasta
    ```

  - {文字列1, 文字列2}：”文字列1”または”文字列2”

    ```
    $ ls 1A25_HUMAN.{fasta,phylip}
    ```

実習6

● ワイルドカード

（カレントディレクトリは ~/data/1_unix/sprot）

1. 下記コマンドを実行して、ワイルドカードの動作を確認する。

```
$ ls *.fasta
$ ls *_HUMAN*
$ ls 1A2[1-5]*.fasta
$ ls 1A25_HUMAN.{fasta,phylip}
$ ls *
```

ファイルの内容を一括表示する(cat)

- cat ファイル名
  - con-cat-nate

    つなぐ、連結する
  - ファイルの内容を表示

    ```
    $ cat 1A25_HUMAN.fasta
    ```
  - ファイル名を複数指定すると内容を連結して表示

    ```
    $ cat *.fasta
    ```

実習7

- **ファイルの内容を一括表示する。**

（カレントディレクトリは **~/data/1_unix/sprot**）

1. 下記コマンドを実行する。

```
$ cat 1A25_HUMAN.fasta
$ cat *.fasta
```

ファイルの部分表示(head,tail)

- head [-行数] ファイル名
  - ファイルの先頭から指定した行数を出力
  - 行数を省略すると10行出力

```
$ head 1A25_HUMAN.sprot
```

  - 巨大なファイルの内容を簡単に確認する場合に便利
- tail [-行数] ファイル名
  - ファイルの最後から指定した行数を出力

```
$ tail -20 1A25_HUMAN.sprot
```

  - “-n +行数”とすると、先頭から数えて指定された行以降を出力

```
$ tail –n +2 1A25_HUMAN.fasta
```

(FASTAファイルの配列のみ表示)

実習8

● ファイルの部分表示

（カレントディレクトリは ~/data/1_unix/sprot）

1. 下記コマンドを実行する。

```
$ head 1A25_HUMAN.sprot
$ tail -20 1A25_HUMAN.sprot
$ cat 1A25_HUMAN.fasta
$ tail –n +2 1A25_HUMAN.fasta
```

ファイルの中身を見る(less)

- less ファイル名
  - ファイルの内容を閲覧する
  - ページの移動には独自のキー操作が必要

キー操作	説明
f, SPACE	1ページ先へ進む
b	1ページ前へ戻る
j, k	1行づつ前(j)後(k)へ移動
g, G	ファイルの先頭(g)、末尾(G)へ移動
/文字列	文字列の検索、n,Nで前後のヒット行へ移動
q	終了

実習9

● ファイルの内容を見る

（カレントディレクトリは ~/data/1_unix/sprot）

1. 1433B_HUMAN.sprotの内容をlessコマンドで見る。

   $ less 1433B_HUMAN.sprot

2. ページを順に表示して、元に戻す。

   <space> , b , j , k

3. ファイルの末尾に移動してから、先頭に戻る。

   G , g

4. “binding”文字列を検索する。

   /binding

5. 次にヒットする場所に移動する。

   n

6. lessを終了する。

   q

ディレクトリの作成と削除 (mkdir, rmdir)

- mkdir ディレクトリ名
  - 新規ディレクトリの作成

```
$ mkdir unixtest
```

- rmdir ディレクトリ名
  - ディレクトリの削除

```
$ rmdir unixtest
```

  - ディレクトリ内にファイルがあると削除できない

実習10

●ディレクトリの作成と削除

（カレントディレクトリは ~/data/1_unix/sprot）

1. ホームディレクトリに移動後、実習用のディレクトリunixtestを作成する。

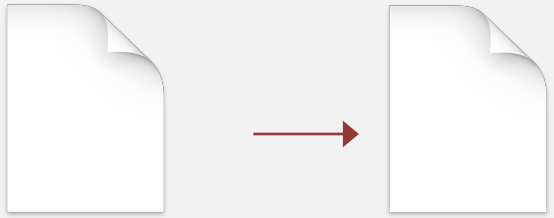
```
$ cd
$ pwd
$ mkdir unixtest
```

2. 作成したunixtestディレクトリに移動し、カレントディレクトリを確認する。

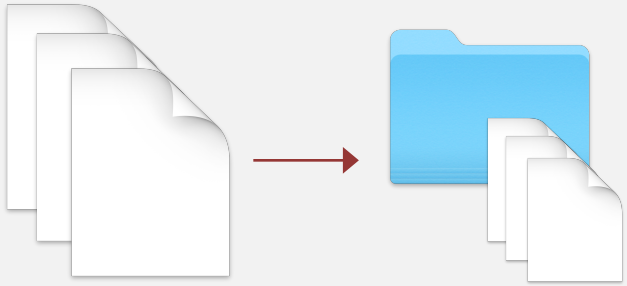
```
$ ls
$ cd unixtest
$ pwd
```

ファイル/ディレクトリのコピー (cp)

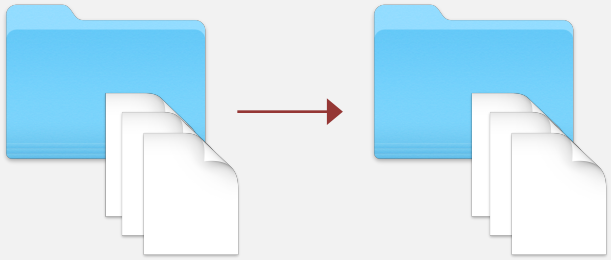
- cp file1 file2
  - file1のコピーをfile2として作成



- cp file1 [file2...] dir
  - file1 (,file2,...) のコピーをdir内に作成



- cp –r dir1 dir2
  - dir1をdir2としてディレクトリごとコピーを作成
  - コピー先のディレクトリが存在する場合は、そのディレクトリ以下にdir1として作成



同一名ファイルが存在すると上書きされるので注意

実習11

●ファイルのコピー

（カレントディレクトリは ~/unixtest）

1. ~/data/1_unix/sprot/1433B_HUMAN.sprotをカレントディレクトリにコピーする。

   ```
   $ cp ~/data/1_unix/sprot/1433B_HUMAN.sprot .
   ```

2. コピーした1433B_HUMAN.sprotを更にcopyfileという名前でカレントディレクトリにコピーする。

   ```
   $ cp 1433B_HUMAN.sprot copyfile
   ```

3. Fastaという名前のディレクトリを作成する。

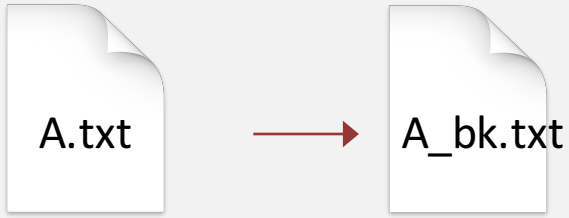
   ```
   $ mkdir Fasta
   ```

4. ~/data/1_unix/sprot/以下のfastaファイル（.fasta拡張子を持つファイル）だけを、3.で作成したFastaディレクトリにワイルドカードを使用してコピーする。

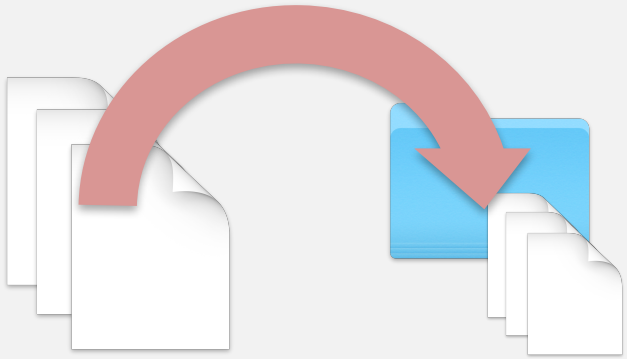
   ```
   $ cp ~/data/1_unix/sprot/*.fasta Fasta
   ```

ファイル/ディレクトリ名の変更、移動 (mv)

- mv file1 file2
  - file1の名前を file2 に変更



- mv file1 [file2...] dir
  - file1 (,file2,...) をdir内に移動



同一名ファイルが存在すると上書きされるので注意

実習12

● ファイル名の変更とファイルの移動

（カレントディレクトリは ~/unixtest）

1. copyfileの名前をnewfileに変更する。

```
$ mv copyfile newfile
```

2. 新規にHumanディレクトリを作成。

```
$ mkdir Human
```

3. Fastaディレクトリ内のHUMANが付いたファイルのみをHumanディレクトリに移動する。

```
$ ls Fasta/*HUMAN*
```

```
$ mv Fasta/*HUMAN* Human
```

実際にワイルドカードを使用して移動や削除を行う場合は、事前にlsで対象のファイルを確認すること。

シンボリックリンクの作成 (ln –s)

- ln –s ファイル／ディレクトリ名 シンボリックリンク名
  - ファイル／ディレクトリの「別名」を作成
  - OSXのエイリアスやWindowsのショートカットに相当

    $ ln –s dir/file1 .

    dir/file1のシンボリックリンクをカレントディレクトリ(**.**)に作成
  - オリジナルのファイルが移動・消去されると、シンボリックリンクを通した参照はできなくなる
- 用途
  - 他ディレクトリへの容易なアクセス

    $ ln –s ~/data/1_unix/sprot .

    よく使うディレクトリをホーム配下にシンボリックリンクする

実習13

● シンボリックリンクの作成

（カレントディレクトリは ~/unixtest）

1. Human/1433B_HUMAN.fastaのシンボリックリンクをカレントディレクトリに作成する。

   ```
   $ ln –s Human/1433B_HUMAN.fasta .
   ```

2. 作成したシンボリックリンクファイルの内容を表示する。

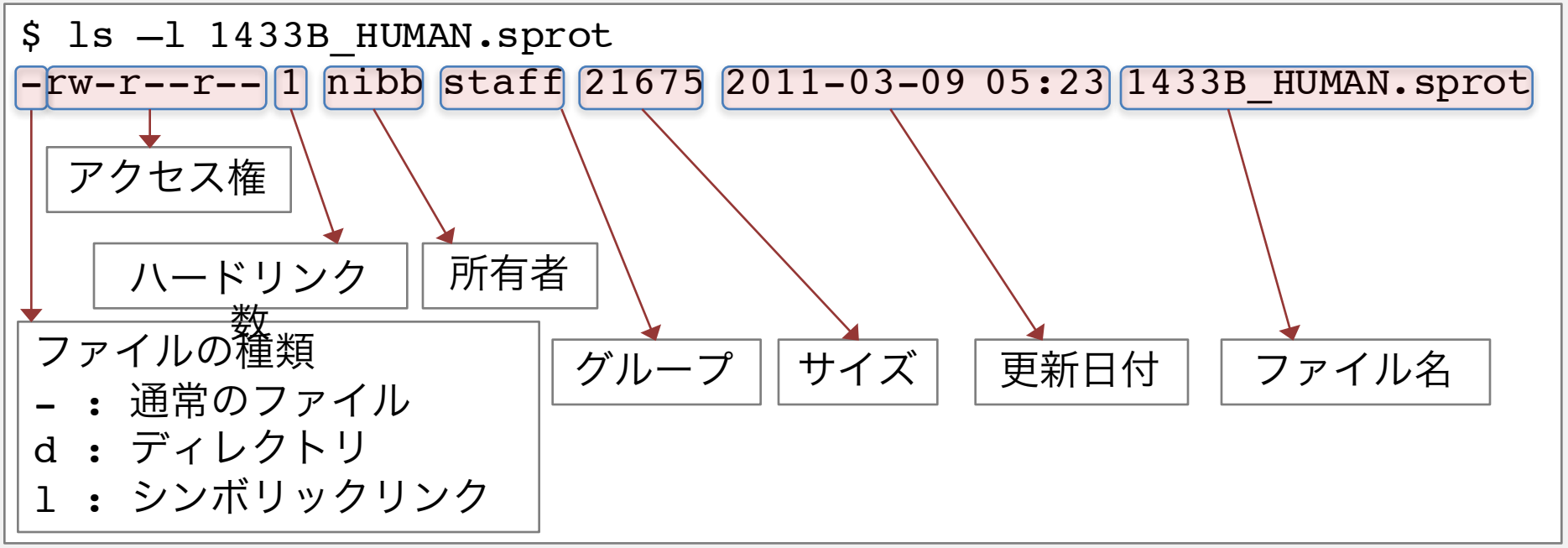
   ```
   $ cat 1433B_HUMAN.fasta
   ```

mvやcpと同様に、最後の引数がディレクトリの場合は、そのディレクトリ上にオリジナルと同じ名前のシンボリックリンクが作成される。

参考：-sオプションを付けない場合は、ハードリンクと呼ばれ、オリジナルファイルを移動・消去してもリンクを通した参照が維持される。ただし、ディレクトリのハードリンクが作れないことや、ボリュームを跨いだハードリンクが作れないなどの制約がある。

ファイル情報の表示 (ls –l)

- ls –l ファイル／ディレクトリ名
  - ファイルの大きさや更新日などの情報をリスト表示
- ls –lt ファイル／ディレクトリ名
  - 日付順に表示



実習14

● ファイル情報の表示

（カレントディレクトリは ~/unixtest）

1. カレントディレクトリの詳細情報リストを表示する。

   ```
   $ ls –l
   ```

2. 詳細情報を更新日時順に表示する。

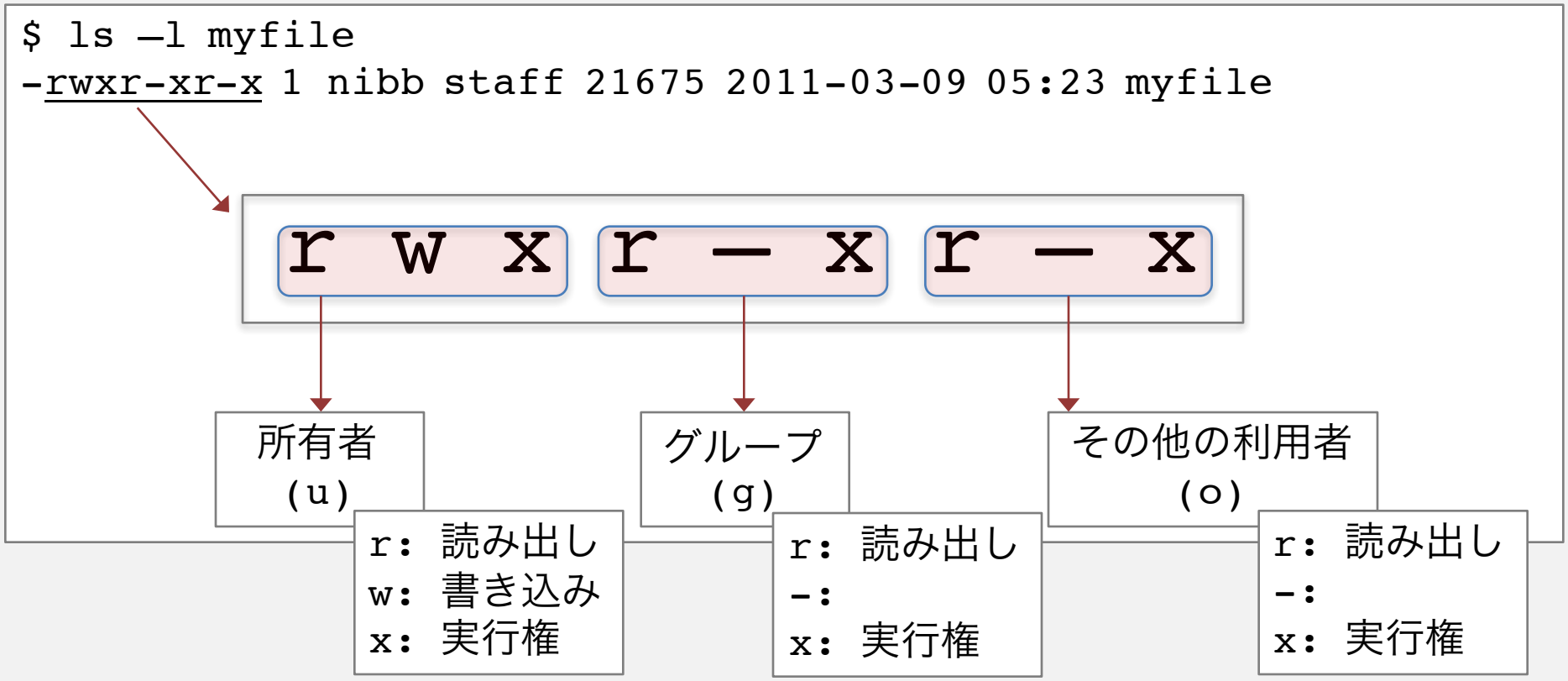
   ```
   $ ls –lt
   ```

ファイルのアクセス権

- 全てのファイル／ディレクトリに付けられている
  - OSXやWindowsでは気にしないが、UNIXでは意外と重要
- 誰に対して、何ができるか
  - 誰に対して：
    - (u) ファイルの所有者
    - (g) 同一グループの利用者
    - (o) その他
  - 何ができるか：
    「読む(r)」「書く(w)」「実行する(x)」
  - 「実行する(x)」
    ファイルをプログラムとして実行できる
    ディレクトリの場合は「移動が可能」

アクセス権モードの確認

- ls –l ファイル／ディレクトリ名



アクセス権モードの変更

- chmod [モード] ファイル名
  - ファイルのアクセス権モードの変更
  - モードの書式を記述する

```
$ chmod u+x file      所有者に実行権を与える
$ chmod go-rwx file   所有者以外のアクセスを禁止
```

  - 8進数3桁で各ユーザのレベルを列挙

```
$ chmod 600 file  所有者のみ読み書きできる
```

	所有者(u)			グループ(g)			その他(o)		
パーミッション	r	w	x	r	w	x	r	w	x
8進数	4	2	1	4	2	1	4	2	1
設定値	合計値			合計値			合計値		

実習15

●アクセス権の確認と変更

（カレントディレクトリは ~/unixtest）

1. カレントディレクトリにあるディレクトリ、ファイルのアクセス権を確認する。

   $ ls –l

2. **copyfile**のアクセス権から 自身の**Read**権限を削除する（本来はこのようなことはしませんが）。

   $ chmod u-r copyfile

3. **less**で**copyfile**を読む。

   $ less copyfile

   エラーになります。

4. 再度**copyfile**のアクセス権に自身の**Read**権限を追加する。

   $ chmod u+r copyfile

5. **less**でエラーなく**copyfile**が読めることを確認する。

   $ less copyfile

ファイルの削除 (rm)

- rm ファイル名 ...
  - 指定したファイルを削除する
- rm –rf ディレクトリ名
  - ディレクトリの中身を確認なしで全て消去した上でディレクトリを削除する
  - 確認しながら消去するには `rm –ri` とする
- 「ゴミ箱」はありません
  - 削除したファイルを復活することはできません
  - ワイルドカードを使用したファイルの削除は注意

    必ず `ls` で対象ファイルを確認しましょう

実習16

● ファイルの削除

（カレントディレクトリは ~/unixtest）

1. newfileを削除する。

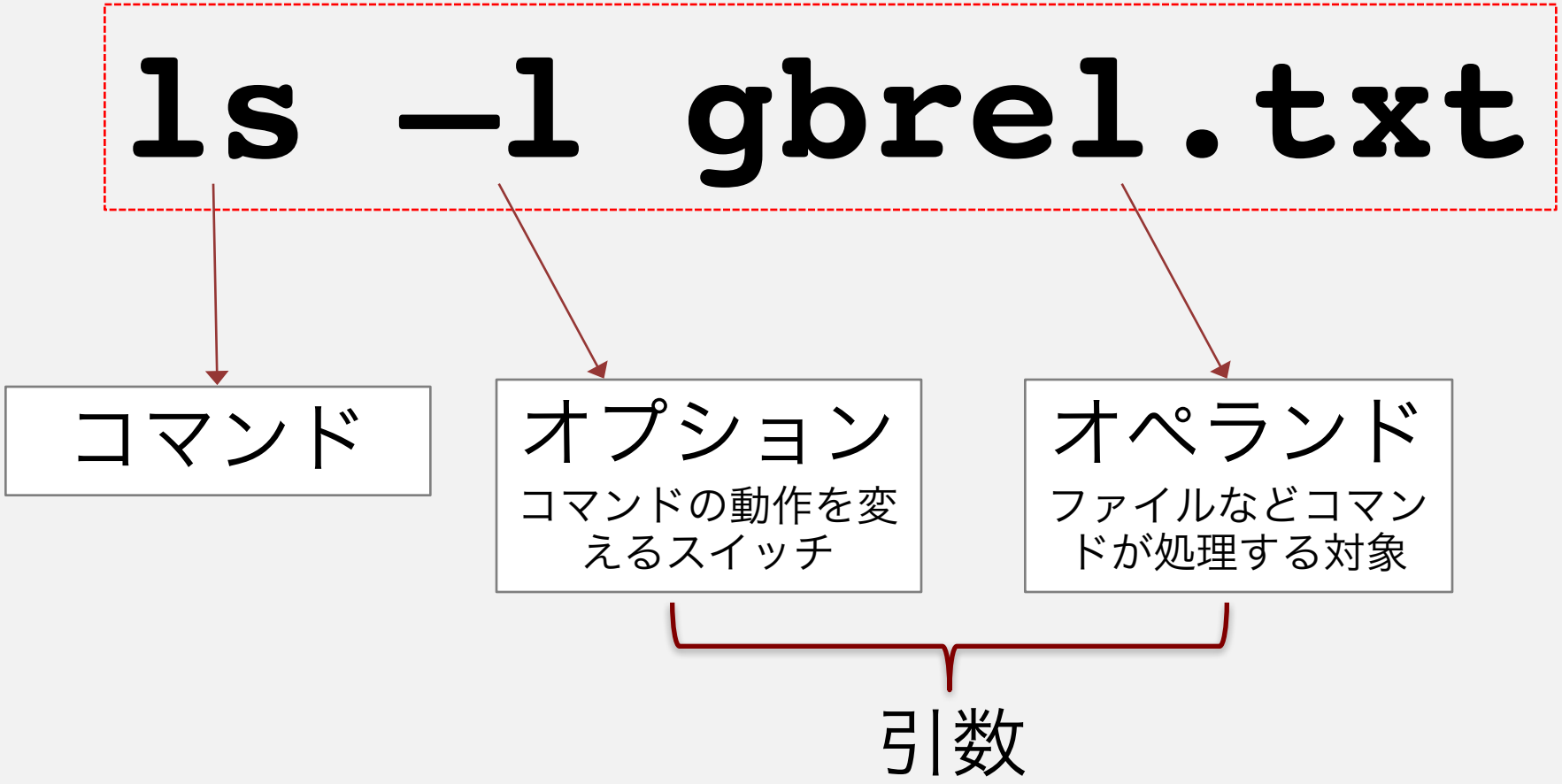
   ```
   $ rm newfile
   ```

2. Fastaディレクトリ全体を削除する。

   ```
   $ rm –rf Fasta
   ```

コマンドの実行

UNIXコマンドの基本形



コマンドマニュアルの参照 (man)

- man コマンド名

```
$ man ls
```

[ ]で囲まれている引数は省略可能

```
NAME
     ls - list contents of directory

SYNOPSIS
     ls [-RadLCxmlnogrtucpFbqisf1AMSDP] [names]

DESCRIPTION
     For each directory argument, ls lists the c                ctory; for
     each file argument, ls repeats its name and any other information
                        is sorted alphabetically by default.  When
                        , the current directory is listed.  When several
     arguments are given, the arguments are first sorted appropriately, but
     file arguments appear before directories and their contents.  ls
     processes supplementary code set characters according to the locale
     specified in the LC_CTYPE and LC_COLLATE environment variables [see LANG
     on environ(5)], except as noted under the -b and -q options below.
```

下線付きは変数

指定可能なオプション

man -k keyword でkeywordに関連するコマンド一覧を表示

実習17

●マニュアルの参照

（カレントディレクトリは ~/unixtest）

1. lsコマンドのマニュアルを表示する。

```
$ man ls
```

行数・単語数のカウント(wc)

- wc ファイル名
  - ファイルの行数、単語数、文字数を出力

```
$ wc 1433B_HUMAN.fasta
6   25 481 1433B_HUMAN.fasta
(6行、25単語、481文字)
```

  - ファイル名を省略するとキーボード入力に対して実行

```
$ wc
This is a pen.
(Control-D)
1   4    15
(1行、4単語、15文字)
```

実習18

● 行数・単語数のカウント

（カレントディレクトリは ~/unixtest）

1. 1433B_HUMAN.fastaの単語数をカウントする。

```
$ wc 1433B_HUMAN.fasta
```

2. キーボードから”This is a pen.”を入力し、その単語数をカウントする。

```
$ wc
This is a pen.
(Control-D)
```

パターン検索(grep)

- grep パターン ファイル名...
  - ファイル中でパターンを含む行を出力

```
$ grep GO 1433B_HUMAN.sprot
```

1433B_HUMAN.sprotから GO を含む行を検索

```
$ grep ^FT 1433B_HUMAN.sprot
```

1433B_HUMAN.sprotから FT で始まる行を検索

(“^” は行の先頭を意味する)。

  - grep –v とすると（オプション -v）パターンを含まない行を出力する。
  - ファイル名を省略すると、やはり端末から文字列を読み込んでパターンを検索する。

実習19

● パターン検索

（カレントディレクトリは ~/unixtest）

1. 1433B_HUMAN.sprotの内容をlessで確認し、GOとFTを検索する。

   $ less 1433B_HUMAN.sprot

2. 1433B_HUMAN.sprotから"GO"が含まれる行を表示する。

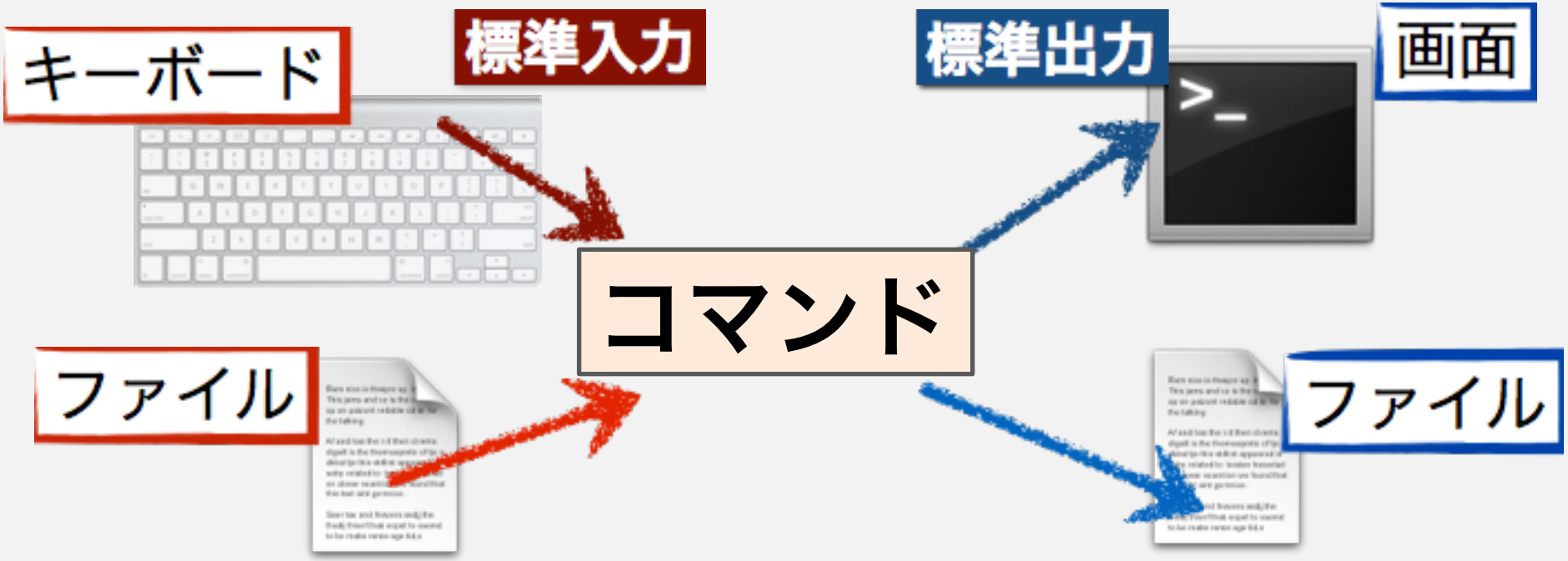
   $ grep GO 1433B_HUMAN.sprot

3. 1433B_HUMAN.sprotからFeature table data(FT)の行を表示する。

   $ grep ^FT 1433B_HUMAN.sprot

入出力のリダイレクション

- リダイレクション
  - コマンドへの入力元、出力先を切換える機能
  - 標準入力＝キーボード、標準出力＝画面



エラー内容を出力する「標準エラー出力」もある

リダイレクションの例・出力

- 端末画面に出力

```
$ grep GO 1433B_HUMAN.sprot
```

- ファイルに保存 (>)
  - コマンドの右に ” > filename “ を加える
  - 結果を GO_count というファイルに保存

```
$ grep GO 1433B_HUMAN.sprot > GO_count
```

(同名ファイルがある場合は上書きされる)

- 存在するファイルに追加書き (>>)
  - コマンドの右に ”>> filename” を加える

```
$ grep GO * 1433?_HUMAN.sprot >> GO_counts
```

(ファイルが存在しない場合は作成される)

リダイレクションの例・入力

- ファイルから入力
  - コマンドの右側に “< *filename* ” を加える
  - wc コマンドで、GO_countファイルの行数を数える

```
$ wc < GO_count
```

(wc GO_count と同じ)

実習20

●リダイレクト

（カレントディレクトリは ~/unixtest）

1. 1433B_HUMAN.sprotから"GO"が含まれる行を検索して、その結果をリダイレクト（出力）を使用してGO_countファイルに保存。

   ```
   $ grep GO 1433B_HUMAN.sprot    (ファイル保存前に確認)
   ```

   ```
   $ grep GO 1433B_HUMAN.sprot > GO_count
   ```

2. ファイルの中身をlessで確認する。

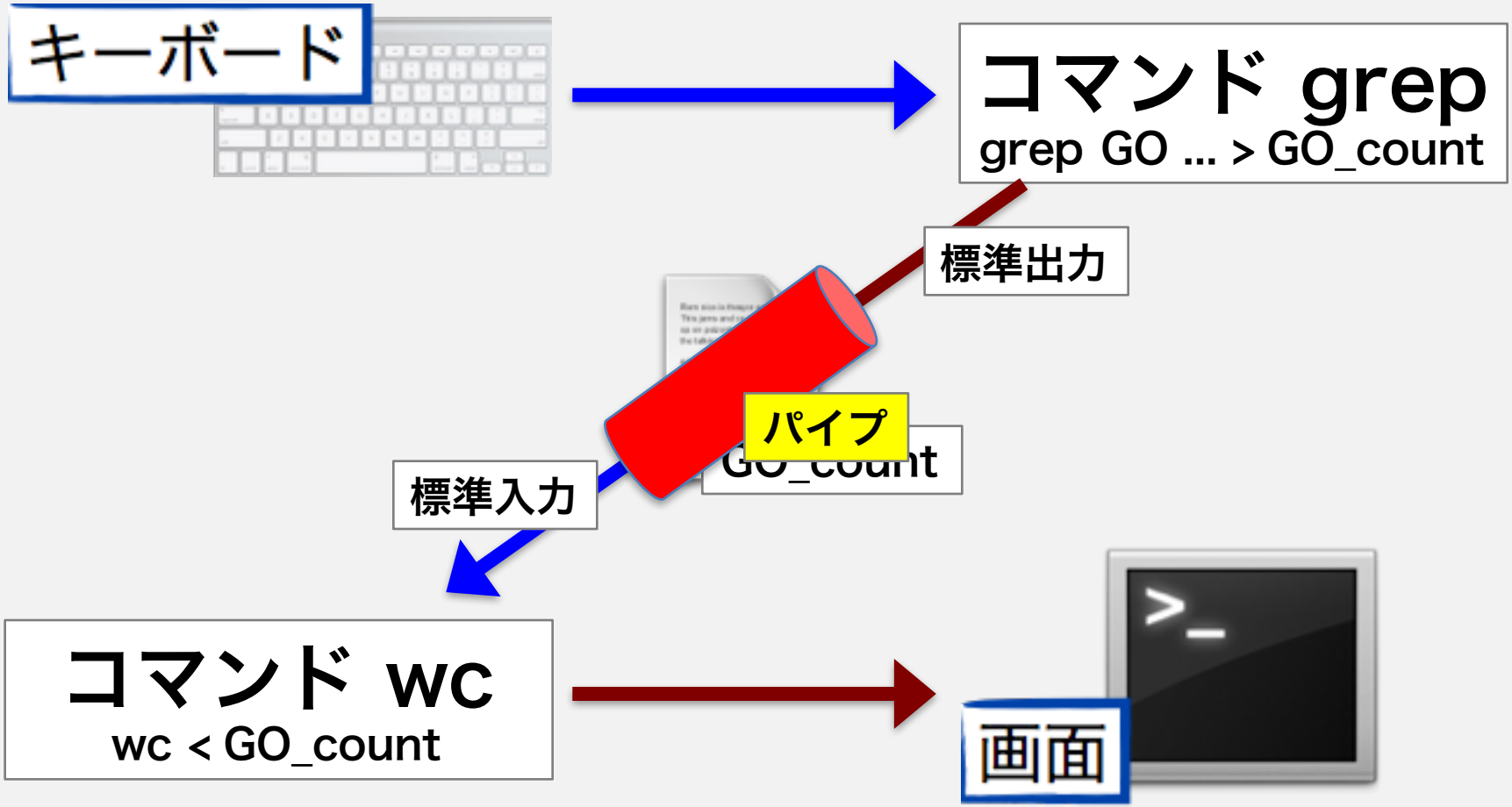
   ```
   $ less GO_count
   ```

3. リダイレクト（入力）を使用してGO_countの単語数を数える。

   ```
   $ wc < GO_count
   ```

パイプによるコマンドの結合

- コマンドの標準出力→標準入力をつなげる



パイプの例

- パイプはコマンド間を縦棒（|）でつなげる
  - grep の出力行数をwcで数える

```
$ grep GO 1433B_HUMAN.sprot | wc
```

  - grep の結果を less で見る

```
$ grep ^FT 1433B_HUMAN.sprot | less
```

  - grepの結果をさらにgrepして絞り込む。

```
$ grep ^FT 1433B_HUMAN.sprot | grep HELIX
| less
```

- コマンドはいくつでも結合できる

実習21

●パイプ

（カレントディレクトリは ~/unixtest）

（検索対象は1433B_HUMAN.sprot）

1. “GO”を検索した結果をパイプを使用してwcで単語数をカウントする。

```
$ grep GO 1433B_HUMAN.sprot | wc
```

2. “FT”から始まる行を検索した結果をパイプを使用してlessで表示する。

```
$ grep ^FT 1433B_HUMAN.sprot | less
```

3. “FT”から始まる行を検索した結果から更に“HELIX”を検索してlessで表示する。

```
$ grep ^FT 1433B_HUMAN.sprot | grep HELIX | less
```

シェルスクリプト

- 一連のコマンドを記述したファイル
  - スクリプトファイル、バッチファイル
  - 例）data/bin/testpg の内容

```
#!/bin/sh
pwd
ls –la
```

- 実行
  - ファイルを単体で実行させるためには実行権が必要

```
$ chmod +x testpg
```

  - 実行はパスを省略せずに入力

```
$ ./testpg
```

もしくは `/Users/<user>/data/1_unix/sprot/testpg`

  - コマンドサーチパスに登録されているディレクトリに保存すれば、コマンド名のみで実行できる。

実習22

● シェルスクリプト

（カレントディレクトリは ~/unixtest）

1. ~/data/1_unix/sprot/testpgをカレントディレクトリにコピーし中身を確認する。

   $ cp ~/data/1_unix/sprot/testpg .

   $ less testpg

2. testpgを実行する。（相対パス／絶対パスのいずれかを付けて実行）

   $ ./testpg（"Permission denied"のエラー）

3. 実行権がなくエラーとなるため、実行権を付与する。

   $ ls –l testpg　(実行権の確認)

   $ chmod +x testpg

   $ ls –l testpg

   $ ./testpg

4. パスを付けずに実行した場合の動作を確認する。

   $ testpg （"command not found"のエラー）

コマンドサーチパス

- **コマンドサーチパス**
  - パス無しのコマンドは、「コマンドサーチパス」リストから探し出される
    例）`ls` コマンドは `/bin/ls` の下にある
- **PATH変数**
  - コマンドサーチパスが格納されている変数
  - 通常 “.bash_profile” で設定する
  - コロン (:) 区切りで複数のパスをつなげて指定
  - 内容は “echo $PATH” コマンドで確認できる
    echoコマンドは、引数の内容を標準出力に出力
  - 同じ名前のコマンドが複数のパスに存在すると、最初のパスが優先
- **エラー “Command not found”**
  - プログラムが (正しく) インストールされていない。
  - コマンドサーチパスが正しく設定されていない。

コマンドサーチパスの設定

- 一時的に設定する場合
  - PATH変数にコマンドサーチパスを設定

    PATH=$PATH:~/bin

    既存の**PATH**変数の後に “**~/bin**” を加える

- 恒常的に設定する場合
  - ファイル ~/.bash_profile に設定を書き込む

    PATH=$PATH:~/bin

    .bash_profileはログイン時に実行されるファイル

実習23

● コマンドサーチパス

（カレントディレクトリは ~/unixtest）

1. 現在のコマンドサーチパスが格納されている**PATH**変数を確認する。

   $ echo $PATH

2. **~/bin**をコマンドサーチパスに一時的に追加する

   $ PATH=$PATH:~/bin

   ユーザは**~/bin**ディレクトリへ自身で作成したコマンドをコピーし、パスの記述なしで呼び出す事ができる。

3. 前の実習で使用した**testpg**を**~/bin**ディレクトリにコピーしてパス無しで動作するかを確認する。もし**~/bin**ディレクトリは存在しない場合はで作成すること。

   $ cd

   ($ mkdir bin)

   $ testpg　＃エラーになる

   $ cp unixtest/testpg bin

   $ testpg

文字列を標準出力へ出力 (echo)

- echo [-n] 文字列
  - 画面に “Hello World”と表示する

```
$ echo 'Hello World'
```

  - ファイル “.bash_profile” に文字列を書き込む

```
$ echo '#!/bin/sh' > .bash_profile
```

    リダイレクトで標準出力をファイルに向ける
  - 追記で文字列を書き込む

```
$ echo 'PATH=$PATH:~/bin' >> .bash_profile
```

    既存のファイルへ追記する場合は “>>” を使用

実習24

● 恒常的にコマンドサーチパスを追加

（カレントディレクトリは ~ ホームディレクトリへ移動）

1. ホームディレクトリに移動

   $ cd

2. ログイン時に実行される**.bash_profile**へ**echo**コマンドを使用して設定を書き込む。

   $ echo 'PATH=$PATH:~/bin' > .bash_profile

3. **.bash_profile**の内容を確認する。

   $ less .bash_profile

4. ターミナルで新しいウィンドウを表示して**PATH**変数を確認する。

   Command + N で新規ウィンドウ

   $ echo $PATH

エイリアス(alias)

- alias 別名='コマンド名'
- コマンドの別名を作成する
  - lsの初期オプションを”-a”(隠しファイル表示)にする

    ```
    $ alias ls='ls -a'
    ```
  - rmの初期オプションを”-i”(確認あり)にする

    ```
    $ alias rm='rm -i'
    ```
- エイリアスを取り消す
  - ls のエイリアス設定をクリアする

    ```
    $ unalias ls
    ```
  - バックスラッシュ(￥)でエイリアスを一時的に取り消す

    ```
    $ ¥ls
    ```
- 設定ずみエイリアスの一覧表示

  ```
  $ alias
  ```

実習25

● エイリアス

（カレントディレクトリは ~）

1. エイリアスを使って**ls**の動作を”隠しファイル表示 **(-a)**”且つ”ファイル種類表示 **(-F)**”に変更する。

   ```
   alias ls='ls -aF'
   ```

2. 現在設定されているエイリアスを確認する。

   ```
   alias
   ```

3. エイリアスではない本来の**ls**を実行した結果を比べてみる。

   ```
   $ ls
   ```

   ```
   $ ¥ls
   ```

4. **rm**コマンドのオプション設定を確認あり **(-i)** にする。

   ```
   $ alias rm='rm –i'
   ```

5. **unixtest/1433B_HUMAN.fasta**を削除しエイリアスされた**rm**の動作を確認する。

   ```
   $ rm unixtest/1433B_HUMAN.fasta
   ```

   確認を求められるので、良ければ”**y**”と入力、それ以外の文字を入力するとキャンセルとなる。

メタキャラクタ

- シェルにとって特別な意味を持つ記号

  例）* ? [ ] < > | ! $; & 等

- 引数に現れる時は要注意
  - スペース、メタキャラクタ
  - 引数全体を引用符 (') で囲む

    特別な意味を抑止する
  - 下記例はリダイレクトによりファイルを上書きする

    (誤) $ grep ^> 1A01_HUMAN.fasta

    (正) $ grep '^>' 1A01_HUMAN.fasta

- ファイル名の付け方
  - 英数字、ドット(.)、アンダーバー (_)、ハイフン (-)を使用
  - 上記以外の記号、2バイト文字(日本語)は使用しない

実習26

● メタキャラクタ

（カレントディレクトリは ~/）

1. **grep ^> 1A31_HUMAN.fasta**の動作を確認する。

   作業ディレクトリは**~/unixtest**とする。

   ```
   $ cd unixtest
   ```

   念のためコピーしたファイルを使用する。

   ```
   $ cp Human/1A31_HUMAN.fasta .
   ```

   ```
   $ grep ^> 1A31_HUMAN.fasta
   ```

2. 結果が戻ってこないので、適当な文字を入力し、**Control-D**で終了する。

   このコマンドラインは、キーボードから入力された文字列から行頭(^)を検索して**1A31_HUMAN.fasta**ファイルにリダイレクト出力(>)するという意味になる。

ジョブの中止と中断

- コマンド（ジョブ）の実行中
  - 端末は結果待ち状態
  - 端末からの入力を受け付けない
  - 実行が終了すると、プロンプトを表示
- 実行中のジョブを途中で中止
  - コントロールキーを押しながら ”C”

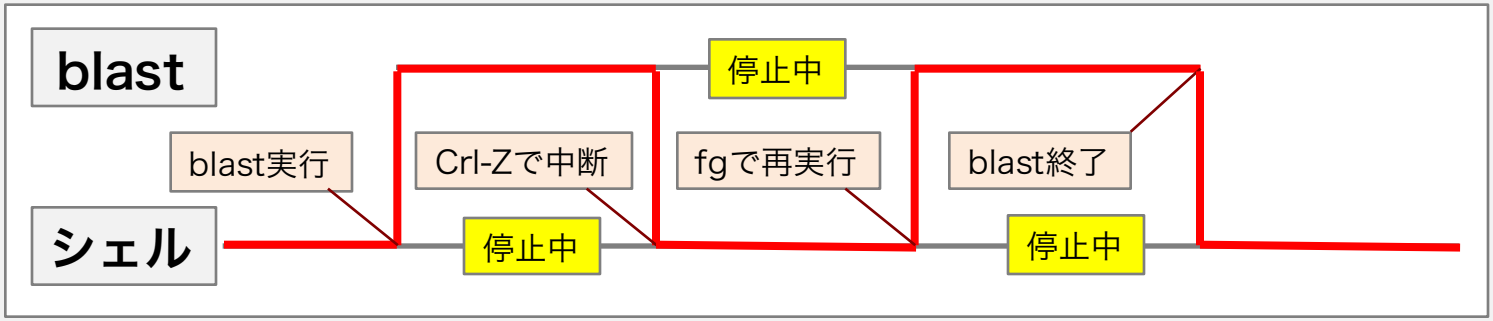
    Cntrol + C
- ジョブを一時中断
  - コントロールキーを押しながら ”Z”

    Control + Z
- 一時中断したジョブを再開
  - fg : フォアグラウンドで再開（端末は結果待ちの状態）
  - bg : バックグラウンドで再開（端末からの入力ができる）

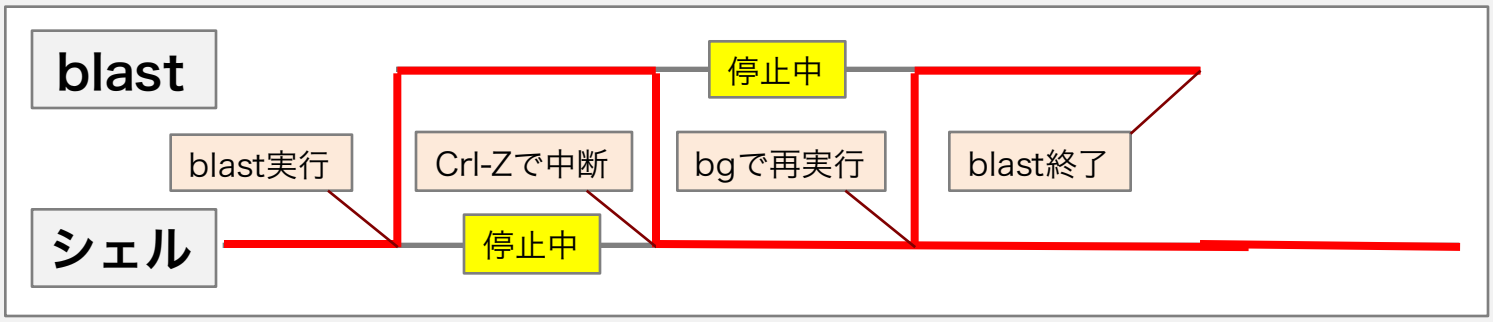
    はじめからバックグラウンドでジョブを実行するには、コマンド行末尾に “&”

フォアグラウンドとバックグラウンド

- フォアグラウンド実行 (fg)
  - 実行中はコマンド入力を受け付けない



- バックグラウンド実行 (bg)
  - 実行中でもコマンド入力が可能



プロセスの表示と停止(ps,kill)

- プロセス
  - OSが管理する単位
  - ジョブはシェルが管理する単位
- プロセスの一覧表示
  - ps コマンド

    PID:プロセス**ID,TTY:**端末

    TIME:CPU時間**,CMD:**コマンド

```
$ ps
   PID TTY          TIME CMD
218074 pts/12   00:00:00 bash
223974 pts/12   00:00:01 ls
223975 pts/12   00:00:00 ps
```

- 実行中のプロセス停止
  - kill PID...
  - PIDをpsで調べて停止

```
$ kill 223974
$ ps
   PID TTY          TIME CMD
218074 pts/12   00:00:00 bash
223980 pts/12   00:00:00 ps
[1]+  Terminated ls -lR /
```

プロセスの稼働状況を把握(top)

- top
  - UNIXマシンの稼働状況をリアルタイムに表示

```
$  top

Processes:  276 total,  3 running,  13 stuck,  260 sleeping,  1604 threads   15:41:31
Load Avg:  1.98,  2.24,  2.22   CPU usage:  3.46% user,  1.47% sys,  95.5% idle
SharedLibs:  1604K resident,  0B data,  0B linkedit.
MemRegions:  183073 total,  8815M resident,  143M private,  3238M shared.
PhysMem:  15G used (2148M wired),  6234M unused.
VM:  711G vsize,  1026M framework vsize,  18564(0) swapins,  72355(0) swapouts.
Networks:  packets:  69788526/45G in,  43784897/21G out.
Disks:  14135478/253G  read,  30377589/763G  written.

PID     COMMAND       %CPU       TIME      #TH     #WQ   #PORTS  #MREGS  MEM     RPRVT
95057   installd      0.0        00:11.75  2       0     53      161     12M     12M
95056   suhelperd     0.0        00:05.01  2       0     58      164     5840K   5448K
95054   softwareupda  0.0        01:45.06  11      0     175     372     96M     95M
87334   plugin-conta  0.0        00:30.01  6       1     230     182     3344K   2748K
87088   mdworker      0.0        00:00.07  4       0     54      80      6264K   5392Ks
```

  - 終了する時は “q” を押下

実習27

● プロセスの確認

（カレントディレクトリは ~/unixtest）

1. psコマンドを使用して、プロセス一覧を表示する。

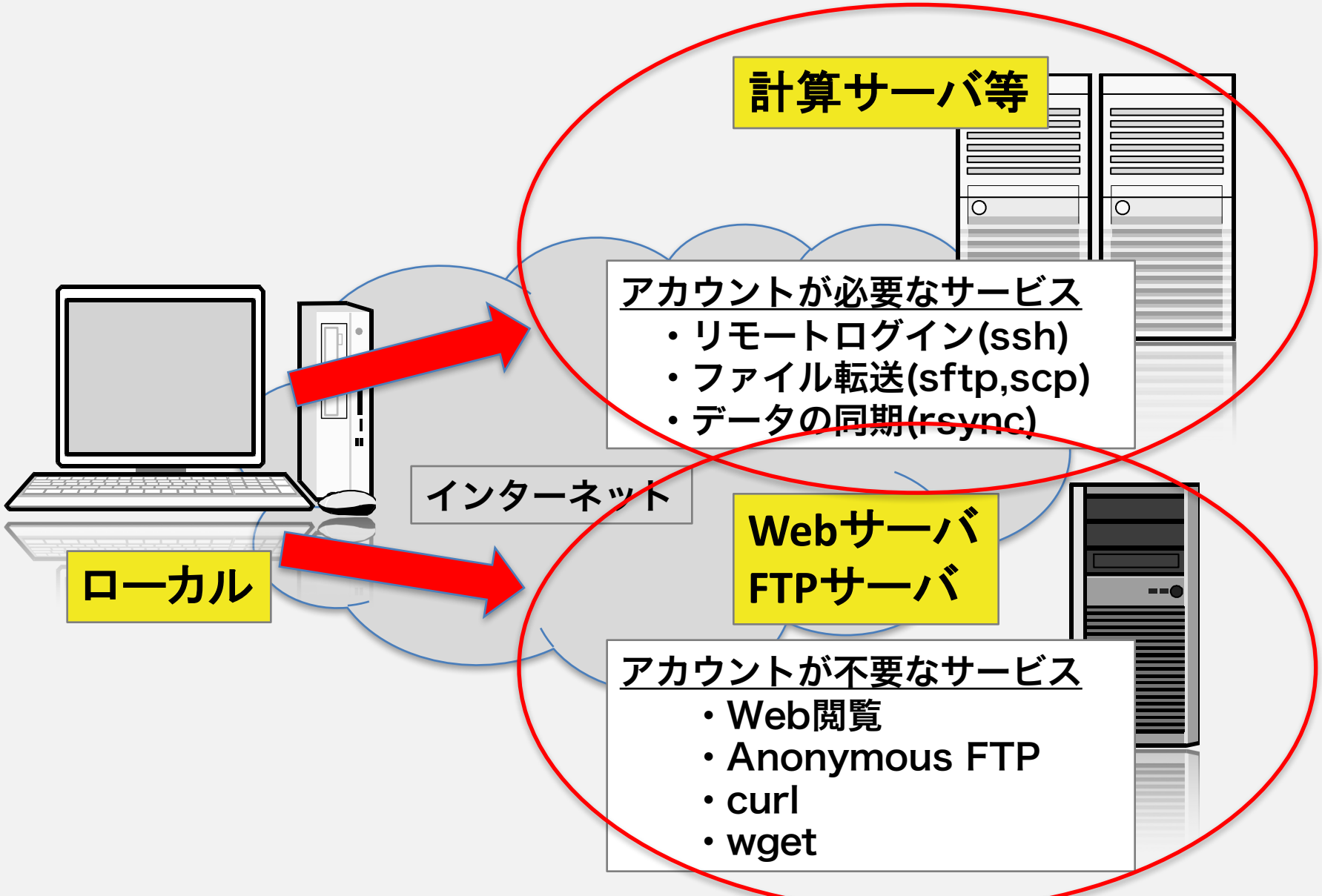
```
$ ps
```

2. topコマンドを使用して、プロセスの稼働状況を確認する。

```
$ top
```

ネットワークを経由した利用

ネットワークを介したサービス



リモートログイン(ssh)

- ssh ユーザ名@ホスト名
  - または ssh –l ユーザ名 ホスト名
  - ネットワーク経由で別の計算機にログイン

```
$ ssh guest@bias4.nibb.ac.jp
Are you sure you want to continue connecting (yes/no)? yes
guest@nibb.ac.jp's password: your_password
[guest@bias4 ~]$
[guest@bias4 ~]$ exit    （ログアウトして元のセッションに戻る）
$
```

- ssh –X ユーザ名@ホスト名
  - X11対応ソフトをローカルの画面に表示可能とする

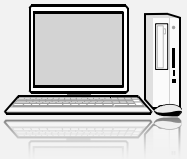
    リモート上で作成した画像ファイルの表示などに使用

    ローカルの環境にx11ソフトが必要

    OSXはXQuarts, Windowsはcygwin/X

ssh鍵認証

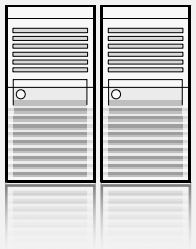
- パスワードを使用しない認証方式
  - 決められたホスト間で鍵を交換し認証を行う
  - パスワードを使用しないためセキュリティが向上
- 方法
  - ローカルで鍵を作成



```
$ ssh-keygen –t rsa
```

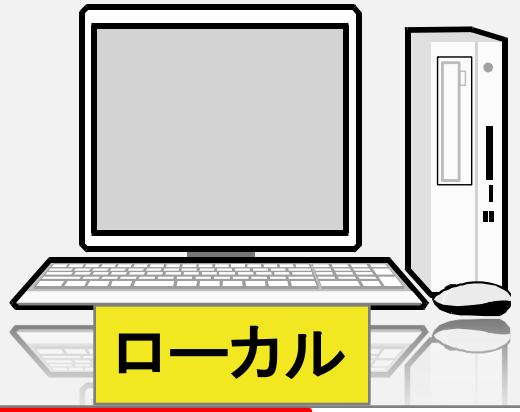
~/.ssh下に公開鍵"id_rsa.pub"
同時に作成される秘密鍵"id_rsa"は触れない。

  - リモートの設定



~/.ssh/authorized_keys
ローカルの公開鍵"id_rsa.pub"の内容を追記

ssh鍵認証

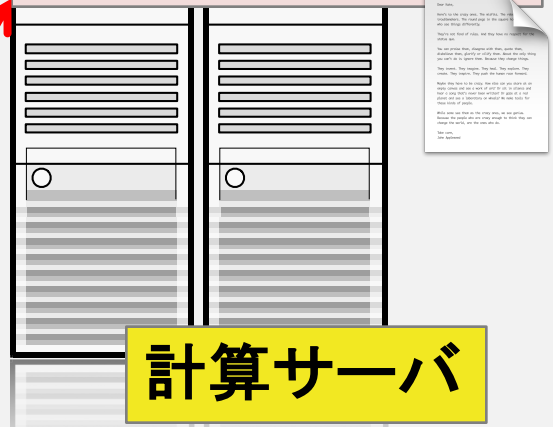


ローカル

```
$ cat ~/.ssh/authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAw6tsxk2OzSbPAwtpu8UqRnDB2W3MtrvLfA
uN9iTXCCdpjmnWh8LlPy2xqGpgkaqIxCKk9/9TUZMEovNAZNrI1N1farAcAAFT
YtRKA4p8CK4FR7vLhiYYBuN1VOXwqJ7SYSZsYxJnqzYIvvMswyOX7kBLbstk3L
UZkBwcYU1EYlTa87a2G1bPpPpHaq9QYXDkurkmIhsEAwxdEIOqmg0T4Z62IW4d
DTqiVnokpSHZc8bDbr1Gn2nK28esjfRY0VmVCt43BPt9WZY58PY4tF5IRSRId6
GixIauVgeKiVhlWJ6bhvqVrBPeWOclXegnzqwdX+EBejO==
cptest00@bias4-login
```

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/cptest00/.ssh/id_rsa)  <リターン>
Created directory '/home/cptest00/.ssh'.
Enter passphrase (empty for no passphrase):  <好きなパスフレーズ>
Enter same passphrase again:  <好きなパスフレーズ>
Your identification has been saved in /home/cptest00/.ssh/id_rsa.
Your public key has been saved in /home/cptest00/.ssh/id_rsa.pub.
The key fingerprint is:
68:8c:27:51:b6:1d:05:f3:4c:10:e3:6f:9d:f2:aa:18 cptest00@bias4-login
The key's randomart image is:
+--[ RSA 2048]----+
|       o B=o     |
|      o + B      |
|     . . o o     |
|      + . . . .  |
|     o = S + o   |
|      +   . o    |
|       E     .   |
|         o   .   |
|        . ...    |
+-----------------+
```

~/.ssh/authorized_keys



計算サーバ

~/.ssh/id_rsa.pubの内容を
末尾に貼り付ける

ssh-keygenのパスフレーズ入力を省略すると入力なしでログインが可能
セキュリティ面を考慮して、パスフレーズは設定することを推奨

リモートファイルコピー(scp)

- scp コピー元 コピー先
  - ネットワーク経由でファイルのコピーを行う

```
$ scp text.txt  user@bias4.nibb.ac.jp:data/text.txt
```

  - ネットワーク上のコピー先指定方法

    user@host:<where_to_copy_path>

    *コピー元、コピー先のいずれかに指定する

    例）**nibb@bias4.nibb.ac.jp:data/sprot/1433S_HUMAN.sprot**

    * ディレクトリ丸ごとコピーする場合 **-r** オプション

    * **<where_to_copy_path>**を指定しない場合はホームディレクトリ直下にコピー

- cpコマンドと同じ感覚で使用できる

ファイル転送(sftp/ftp)

- sftp リモートホスト名
  - sftp/ftpログイン後、専用コマンドを使用して操作

bias4へはsftpのみ使用可能。bias4からは両方使用可。

コマンド	用途
ls	リモートのファイル一覧表示
lls	ローカルのファイル一覧表示
cd	リモートのディレクトリ移動
lcd	ローカルのディレクトリ移動
get	ファイルをリモートからローカルに転送
mget	複数のファイルを転送。ワイルドカード使用可
put	ファイルをローカルからリモートに転送
mput	複数のファイルを転送。ワイルドカード使用可
help	使用できるコマンドの簡易ヘルプ

データ転送(curl)

- curl URL
  - HTTPやFTP経由のファイル取得

```
$ curl http://www.nibb.ac.jp
```

www.nibb.ac.jpのトップページを標準出力に書き出し

オプション	用途
-o *file url*	urlのデータをfileに保存
-O	url上のファイル名で保存（http://hoge.com/fig.jpgのように「ファイル名がある」urlのみ）
-O *url[start-end]*	http://www.hoge.com/[01-10].jpg等とすることで、01.jpg, 02.jpg, 03.jpg ... 10.jpgのファイル全て取得
-h	ヘルプを表示

データの同期(rsync)

- rsync [オプション] コピー元/ コピー先/
  - ディレクトリ間の同期をとる

```
$ rsync –auv user@bias4:dir/  ./dir/
```

オプション	用途
`-a`	アーカイブモード（属性などを一致させるなど）
`-u`	コピー先の同一ファイルが新しい場合コピーしない
`-v`	コピー状況の表示
`--delete`	コピー元にないファイルをコピー先から削除
`--exclude-file` *filename*	filename内のファイル名リストは対象にしない

ローカル内やリモート間のファイル群コピーの他、
手軽なバックアップの手段としてよく用いられる。

実習28

●ディレクトリの同期

（カレントディレクトリは ~/unixtest）

1. 作業用ディレクトリunixtestの同期を~/backup/unixtestディレクトリに行う。backupディレクトリは作成すること。

```
$ cd
$ mkdir backup
$ rsync –auv unixtest/ backup/unixtest/
```

2. 同期元からunixtest/1433B_HUMAN.sprotを削除

```
$ rm unixtest/1433B_HUMAN.sprot
```

3. 再度rsyncで同期を行い同期元から削除されたファイルがどうなるか確認する。オプションとして--deleteを使用する。

```
$ rsync –auv --delete unixtest/ backup/unixtest/
```

ファイルの圧縮と解凍

- データ圧縮
  - 圧縮アルゴリズムを使い、情報を失わずサイズを小さくする
  - 圧縮ファイルは使用する前に元に戻す（解凍）
- gzip（拡張子：.gz or .Z）

  圧縮：**gzip** ファイル名 （ファイル名**.gz**を作成）

  解凍：**gunzip** ファイル名**.gz** または **gzip –d** ファイル名**.gz**
- bzip2（拡張子：.gz2）

  圧縮：**bzip2** ファイル名

  解凍：**bunzip2** ファイル名**.bz2** または **bzip2 –d** ファイル名**.bz2**

アーカイブの作成と展開(tar)

- アーカイブ
  - 複数のファイルやディレクトリを１つのファイルにまとめること
  - 圧縮と組み合わせて、データ配布やバックアップ保存などに用いる
  - 通常アーカイブファイルは拡張子 .tar をつける
- 基本的な使用方法
  - dir を archive.tar としてアーカイブ

```
$ tar cvf archive.tar dir
```

    c: 新しいアーカイブを作成

    v: 処理したファイルの一覧を出力

    f file: fileというアーカイブ・ファイルを使う

  - archive.tar をカレントディレクトリに展開

```
$ tar xvf archive.tar
```

    x: アーカイブからファイルを抽出

  - 圧縮・解凍の同時実行

```
$ tar zxvf archive.tar.gz
```

    z: gzipで圧縮・解凍の処理を行う、j: bzipで圧縮・解凍の処理を行う

実習29

● 圧縮アーカイブの作成と展開

（カレントディレクトリは ~）

1. 先ほどの**~/backup/unixtest**ディレクトリの圧縮アーカイブを、**~/backup/unixtest.tar.gz**という名前で作成する。圧縮形式は**gzip**形式を使用し、作成後は元のディレクトリを削除する。

```
$ cd backup

$ tar zcvf unixtest.tar.gz ./unixtest

$ ¥rm –rf ./unixtest
```

2. **unixtest.tar.gz**の内容を確認する。

```
$ tar ztvf unixtest.tar.gz
```

3. **~/backup/unixtest.tar.gz**を**~/backup**ディレクトリ内に解凍展開する。

```
$ tar zxvf unixtest.tar.gz
```

プログラムソースからのインストール

- ソース一式の取得
  - Anonymous FTPやWebからアーカイブをダウンロード
  - ftp や curl を使用
- アーカイブを展開
  - tar や gzip を使用
- コンパイル
  - インストール方法や注意点を確認

    内包する**README,INSTALL**などのファイルを読む

    必ずしも下記のとおりと限らないので、記述されているインストール手順に従うこと
  - 一般的なコンパイル手順

    ```
    $ configure          環境にあったコンパイルの設定を行う
    $ make               プログラムをコンパイル
    $ make install       所定の場所へインストール
    ```
  - インストール先の書込権限がない場合、root権限で行う必要あり

    ```
    $ sudo make install
    ```

実習30

- SAMtools1.3.1のインストール

（カレントディレクトリは ~）

1. SAMtoolsのダウンロードと解凍

   http://www.htslib.org (Download->samtools-1.3.1)

   ~/Downloads/にbzip圧縮アーカイブとして保存される

```
$ mv Downloads/samtools-1.3.1.tar.bz2  unixtest
$ cd unixtest
$ tar jxvf samtools-1.3.1.tar.bz2
$ cd samtools-1.3.1
```

2. コンパイルとインストール

   ここでは自分のホームディレクトリ以下にインストール

```
$ make
$ make prefix=~ install
```

makeコマンドの-nオプションで実際の実行を行わずどのように処理がされるか表示できる。makeコマンドを実行する前に確認してみよう。

お疲れさまでした

この講習で使用したコマンド一覧は
資料末尾に添付しましたので
参考にしてください。