

遺伝研スパコンへの接続・Linux基本操作

ディレクトリ、ファイル操作を中心としたLinux基本コマンドの使い方を紹介します。

スパコンへの接続

スパコンへのログイン

Macのターミナル、またはWindowsのPowerShellから次のコマンドでスパコンのゲートウェイノードへ接続します。

```
ssh ユーザー名@gw.ddbj.nig.ac.jp
```

秘密鍵にパスフレーズを設定している場合にはパスフレーズを入力してください。
(アカウント登録証に記載されたパスワードとは異なります)

```
選択koshu1@gw1:~
PS C:\Users\tanizawa> ssh koshu1@gw.ddbj.nig.ac.jp
Enter passphrase for key 'C:\Users\tanizawa\.ssh/id_rsa':
Last login: Tue Oct 18 17:35:49 2022 from 219.112.170.88

-----
Thank you for using NIG supercomputer system.
This is the gateway node, do not run program here.
Please use 'qlogin' to login to a login node.
-----

[koshu1@gwB1 ~]$
```

\$で終わる文字列はプロンプトといい、これが表示されているときコマンドの入力待機状態であることを示します。初期状態ではプロンプトにはユーザー名、現在ログインしているノード名と、現在のディレクトリ名(`-` はホームディレクトリを示す記号)が表示されています。

`qlogin` コマンドを使ってログインノードに移動します。

```
qlogin
```

このときパスワード入力を要求されることがあります (最近スパコンアカウントを取得した場合にはおそらくパスワードは不要です)。このときにはアカウント登録証に記載されたパスワード (あるいは自分で変更していればそのパスワード)を入力してください。

パスワード入力を求められないようにする設定方法をこの文章の末尾に記していますのでご参照ください。

```
[koshu1@gwB1 ~]$ qlogin
Your job 16646500 ("QLOGIN") has been submitted
waiting for interactive job to be scheduled ...
Your interactive job 16646500 has been successfully scheduled.
Establishing /home/geadmin/UGER/utilbin/lx-amd64/qlogin_wrapper session to host at138 ...
Warning: Permanently added '[at138]:34341,[172.19.7.186]:34341' (ECDSA) to the list of known hosts.
Last login: Mon Oct 17 15:09:48 2022 from gw1
[koshu1@at138 ~]$
```

qloginを行うとプロンプトが変わりログインノード (上記画像ではat138というマシン)に移動したことがわかります。ログインノードは空きのあるノードが割り当てられるため、ログインごとに異なる可能性があります。

[重要] スパコン上では基本的に `qlogin` してから操作を行うようにしてください。ログイン直後のゲートウェイノード(gw)では解析処理等を行うことは想定されておらず、負荷の高い処理を行うと他のユーザにも影響がでることがあるためです。

ログアウト

```
exit
```

control (ctrl)キーを押しながらキーボードのDキーを押してもログアウトできます。ログアウトするとgwノードに戻り、もう一度行くとスパコンからログアウトされます。

TIPS

- 誤った操作などによって、入力が受け付けなくなってしまった（プロンプトが表示されなくなった）場合には、`control-c` を押すことで、処理をキャンセルし入力待機状態に戻ることができる場合があります。
`q` を押すことで処理を中断させられる場合もあります。
- タブキーを素早く2度押すと、利用可能なコマンドやファイル名が補完されます（候補が多すぎて表示しきれない場合 `q` を押すと途中でとめられます）
矢印キーの上下で過去に入力したコマンドの履歴が表示できます。
これらなるべく活用することで手入力による打ち間違いを避けるのがコツです。

```
[koshu1@at138 ~]$ cu
cuda-gdb          cups-browsed      cupsd              cups-genppd. 5.2  cupstestppd
cuda-memcheck     cups-calibrate    cupsdisable        cups-genppdupdate curl
cupsaccept        cups-config       cupsenable         cupsreject        curl-config
cupsaddsmb        cupsctl           cupsfilter         cupstestdsc       cut
[koshu1@at138 ~]$ cu
```

↑cuまで入力した状態でタブキーを2回押して入力候補を表示させた例

覚えておくとよいショートカット

- control + A** 行の先頭までカーソルを移動
- control + E** 行の最後までカーソルを移動
- control + K** カーソルより後の文字を削除
- control + L** 現在の行が一番上になるように画面をスクロール (画面のクリア)
- control + H** 直前の1文字を削除 (BackSpaceキーと同じ)
- control + D** カーソルの位置の文字を削除 (何も入力されていないときに行うとログアウト)
- alt + →** **alt + ←** 単語の区切れ目までカーソルを移動

講習用のサンプルファイルを取得

講習で用いるファイルやデータを取得するために次のコマンドを実行してください

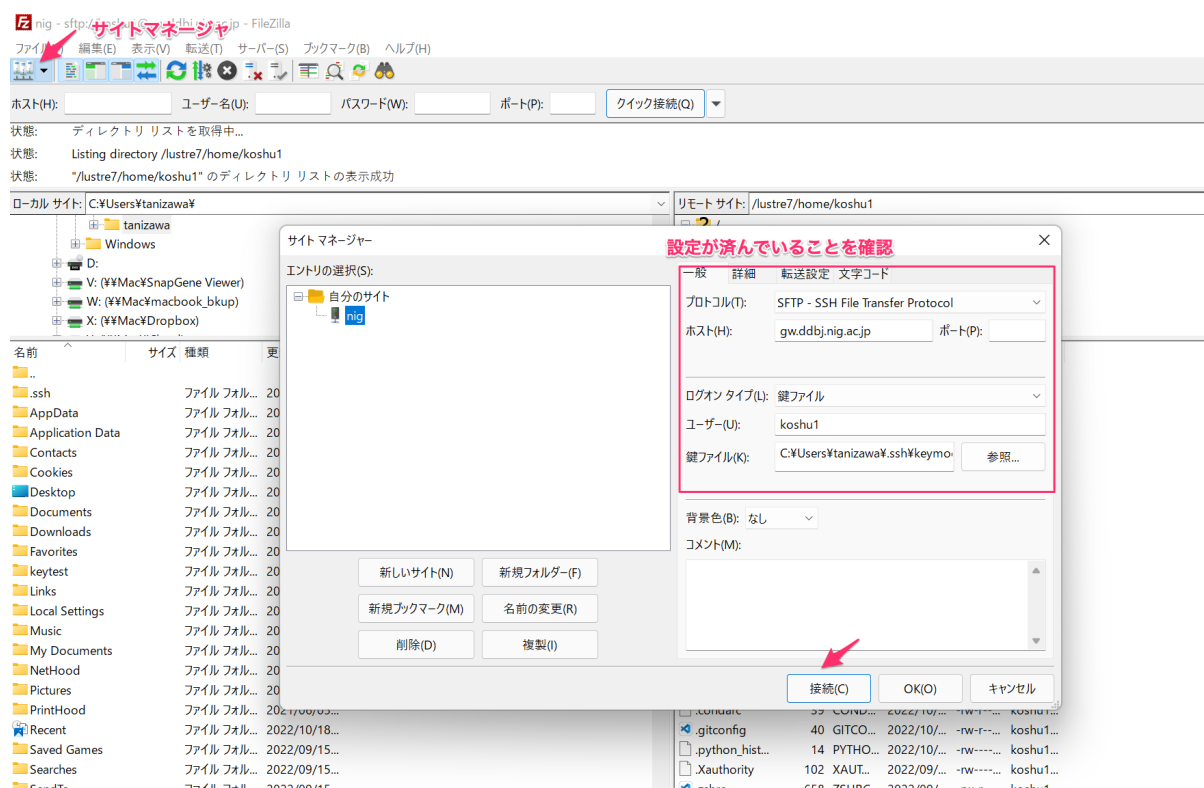
```
git clone https://github.com/genome-sci/basic_course_2023.git
```

* Windows PowerShellを使っている場合、右クリックでペーストができます。

```
(結果)
Cloning into 'basic_course_2023'...
remote: Enumerating objects: 74, done.
remote: Counting objects: 100% (74/74), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 74 (delta 33), reused 38 (delta 12), pack-reused 0
Receiving objects: 100% (74/74), 9.58 MiB | 17.84 MiB/s, done.
Resolving deltas: 100% (33/33), done.
```

Filezillaでの接続

設定が済んでいればサイトマネージャを開き、「接続」ボタンを押すことですぐに接続できる。



接続後、`basic_course_2023` というディレクトリが見えていることを確認

基本操作1. 現在の状況を把握するコマンド (pwdとls)

- `pwd` 現在自分がいるディレクトリ（フォルダ）を表示します。（print working directory）

```
pwd
```

ログイン直後には各ユーザーのホームディレクトリ `/home/userXXX/` にいます。

```
(結果)  
/home/koshu1
```

- `ls` 現在のディレクトリ内にあるファイルおよびディレクトリの一覧 (list) を表示します。

```
ls
```

ログインして初めて使う場合、先ほど取得した講習用データのディレクトリが見えているはずです。

```
(結果)  
basic_course_2023
```

`basic_course_2023` の中身を見る場合には

```
ls basic_course_2023
```

とします。

ディレクトリ名の最初の数文字を打ち込んでタブキーを押すとディレクトリ名が補完されますので、積極的に補完機能を使ってタイプミスを防ぐのがコツです。補完した場合、後に ディレクトリであることを示す `/` がつきますが意味は同じです (`ls basic_course_2023/`)

```
[koshu1@at137 ~]$ ls basic_course_2023  
1  2  3  4  README.md
```

デフォルトでは、青字はディレクトリ、白字はファイルを示しています。

不可視ファイル (. で始まるもの) を含めて表示させます。

```
ls -a
```

`.bash_profile` のように `.` で始まるファイルやディレクトリがいくつか見えるはずですが (この結果は人によって異なる可能性があります)。 `.` で始まるものは不可視項目で、その多くは普段は使用しない設定ファイルなどです。

重要 `.` は現在のディレクトリ、 `..` は一つ上の階層のディレクトリ (親ディレクトリ) を示します。

また、 `-a` は**コマンドラインオプション**のひとつです。Linuxのコマンドの多くはコマンドラインオプションを使うことで、既定の動作を変更させることができます。

オプションは複数組み合わせることもできます。例えば、以下ではファイルの更新日時やファイルサイズも含めて表示させます。

```
ls -a -l
```

なお、上記は `ls -al` としても同様の結果が得られます。

また、多くのシステムでは `ll` が `ls -l` と同じ結果が得られるショートカット(**エイリアス**)として利用できます。

ヘルプを参照することで、どのようなオプションが利用できるか確認することができます。

```
ls --help
```

多くのコマンドでは `-h` `-help` `--help` などのオプションをつけることでヘルプを表示できます。 `whatis コマンド名` で概略を調べることもできます。

基本操作2. ディレクトリの移動 (cd)

- `cd` ディレクトリの移動 (change directory)

basic_course_2023 ディレクトリに移動します。

```
cd basic_course_2023
```

プロンプトが変わり移動できたことがわかります。 `pwd` で確認しておきましょう。

一つ上のディレクトリに移動する (もとのディレクトリに戻る) には `..` を指定します。

```
cd ..
```

ホームディレクトリに戻るにはホームディレクトリを示す記号 `~` を使って

```
cd ~
```

または単に

```
cd
```

とします。

また、直前にいたディレクトリに戻るには

```
cd -
```

とするのが便利です。

Linuxのディレクトリの基本構成

WindowsやMacのディレクトリ構成と同じように階層構造になっています。最上位の階層 `/` をルートディレクトリといいます。

遺伝研スパコンの主要なディレクトリ構成

`/` (ルートディレクトリ)

`/bin` (おもにLinuxシステムの動作に最低限必要な実行ファイルを格納する)

`/usr` (各種アプリケーションと、それに付随するファイルを格納する)

`/usr/local/biotools` 各種生命科学用ツールのsingularityイメージが格納されている

`/usr/local/pkg/` Pythonなどプログラミング言語が置かれている

`/usr/local/seq/` DDBJの塩基配列データやBLASTデータベースが置かれている

`/usr/local/resources/` DRAのデータ、BioProject/BioSampleなどが置かれている

`/home` 各ユーザーごとのホームディレクトリが格納されている

`/home/userXXX/` userXXXのホームディレクトリ

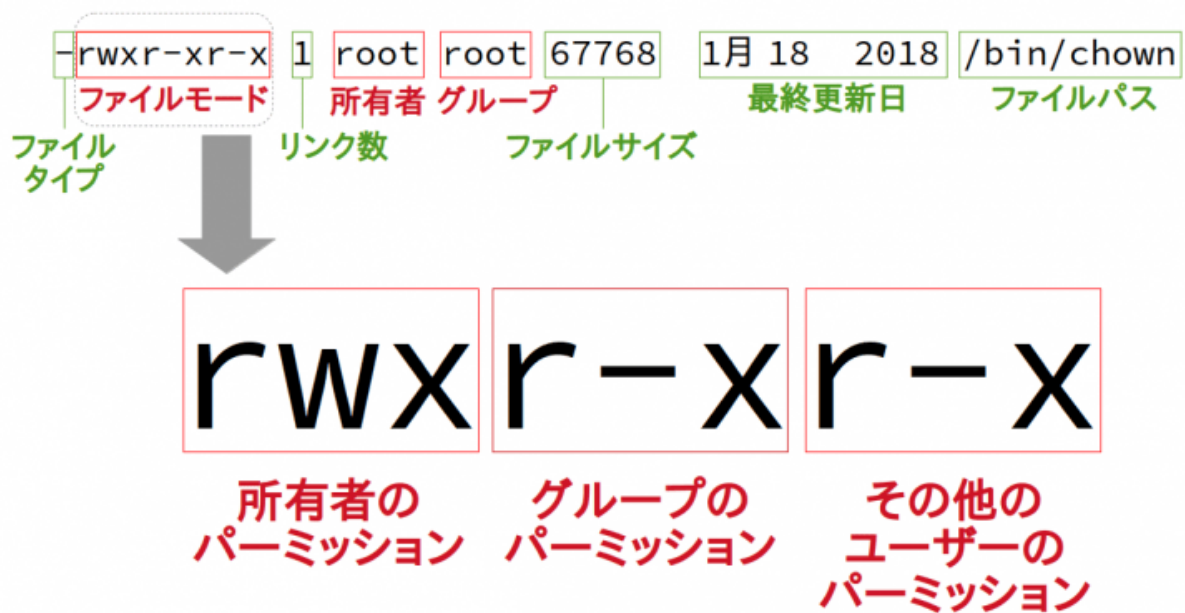
`/home/userYYY/` userYYYのホームディレクトリ

`/home/userZZZ/` userZZZのホームディレクトリ

UNIX/Linuxではファイルやディレクトリごとに所有者や書き込み・読み込み権限（パーミッション）が設定されており、通常は他のユーザーからはファイルが見られないようになっています。

遺伝研スパコンのデフォルト設定では、同じグループに属するユーザーであればファイルは互いに閲覧可能ですが、書き込みはできないようになっています。スパコンアカウント申請時に責任者ごとにグループが作られますので、同じラボのユーザーどうしであればお互いのホームディレクトリの中を見ることができます。

パーミッションは `ls -l` で見られるファイル一覧の中に示されています。



<https://linuxfan.info/filemode-and-permission> より引用

`r` は読み込み、`w` は書き込み、`x` は実行権限を示しています。

(ファイルの場合 `x` はプログラムとして実行可能であること、ディレクトリの場合にはそのディレクトリにアクセス可能であることを示す)

自分が所有するファイルであれば権限を変更することも可能です (`chmod` コマンド)

絶対パスと相対パス

`/usr/local/bin` のようにルートディレクトリ `/` を起点にしたディレクトリ場所の示し方を**絶対パス**といいます。

たとえば

```
cd /usr/local/bin
```

とするとこのディレクトリにいたとしても直接 `/usr/local/bin` に移動することができます。

`/usr/local/bin` から `/usr/local/biotools` に移動したい場合、絶対パスで移動する方法以外に、現在いる場所を起点にした**相対パス**で指定することもできます。

```
cd ../biotools
```

一つ上のディレクトリ (`..`) に移動した後で `biotools` に移動するのと同じ結果になります。

簡単な見分け方としては、`/` または `~` から始まっている場合は絶対パス、そうでなければ相対パスです。

単に `cd biotools` とした場合、`biotools` は今自分がいるディレクトリ内にある子ディレクトリのことを指しますので相対パスで指定していることになります。

基本操作3. ディレクトリの作成と削除 (`mkdir`, `rmdir` または `rm -r`)

講習用ディレクトリに移動し、ディレクトリの作成を練習します。

```
cd ~/basic_course_2023/1/  
pwd
```

(結果)
/home/ユーザー名/basic_course_2023/1

- `mkdir` ディレクトリの新規作成 (make directory)

`test` という名称のディレクトリを作ります。

```
mkdir test
```

作成できたら `ls` `cd` `pwd` などを確認やディレクトリの移動を試してみてください。

- `rmdir` ディレクトリの削除 (remove directory)

(他のディレクトリに移動していた場合は `cd ~/basic_course_2023/1/` でもとの場所に
戻ってから実行)

```
rmdir test
```

`rmdir ~/basic_course_2023/1/test` のように絶対パスで指定しても良い。

`rmdir` は空のディレクトリに対してしか使用できないので、ディレクトリとその中
身のファイルも含めて削除する場合には `rm -r` を使います (`-r` はrecursiveの意)。

```
rm -r test
```

(先の手順ですでにディレクトリを削除した後だとエラーになります)

基本操作4. ファイルの作成と削除 (touchとrm)

- `touch` 空のファイルを作成する (既存のファイルに対して使った場合、ファイル
変更日が更新される)

```
touch a.txt
```

テキストファイルであることが目で見えてわかりやすいように拡張子 `.txt` をつけて
いますが、

Windowsとは異なり、Linuxでは拡張子によってファイルの種類を認識することはありません。

したがって、拡張子は必須ではありません。(`touch a` で `a` という名称の空のファ
イルが作られます)

- `rm` ファイルの削除

```
rm a.txt
```

基本操作5. 画面への出力 (echo, cat, リダイレクト)

- `echo` 文字列を画面に出力します。

```
echo Hello
```

`>` を使うと画面へ出力するかわりにファイルに書き出すことができます (リダイレクト)。

```
echo Hello > a.txt
```

このとき `a.txt` が存在していた場合、上書きされますので気をつけてください。

`>>` を使うと追加書き込みができます。

```
echo world >> a.txt
```

他のコマンドの出力結果もファイルに書き込めます。

```
ll > b.txt
```

(`ll` でファイル一覧を表示した結果をファイルに書き込んでいます)

- `cat` ファイルの中身を画面に出力



```
cat a.txt
```

本来は concatenate の意味で、複数のファイルを連結するコマンドです。


```
cat a.txt b.txt > c.txt
```

(a.txtとb.txtを連結し、その結果をc.txtに書き込んでいます。)


基本操作6. ワイルドカード

 や  を使うとパターンに合うファイルを一括して処理できます (ワイルドカード)。

 は0文字以上の任意の文字列を示します。

 で終わるファイルの一覧を表示

```
ls *.txt
```


 は任意の1文字を示します。

```
ls ?.txt
```

(a.txt, b.txt, c.txt などが表示されます)

```
ls ???.txt
```

(ab.txtやXY.txtにはマッチしますが、a.txtにはマッチしません)

これまでに作成した  で終わるファイルを一括して削除します。

```
rm *.txt
```

単に `rm *` と打つとディレクトリ内のファイル全てが消えてしまいますので注意してください。 `rm -r *` とした場合、サブディレクトリも含めディレクトリの中身が全て削除されます。Linuxでは一度削除したファイルを復活させることはできませんので、使用する場合には注意してください。

基本操作7. ファイル・ディレクトリの移動、名称変更、コピー (mv, cp)

本講義ではFilezillaを使って移動やコピーを行いますので詳細は省きます。奥が深いので、自信がないときは練習用のファイルを作り事前にどのような挙動になるか確認することをおすすめします。

- `mv` ファイルやディレクトリの移動および名前変更

基本は `mv 移動元 移動先` の形で使います。

移動元	移動先	結果
ファイル	ディレクトリ	移動元ファイルを移動先ディレクトリに移動させる
ファイル	ファイル	移動元ファイルを移動先に移す (ファイル名を変更することもできる)。移動先ファイルが存在していた場合、上書きされる。
ディレクトリ	ディレクトリ	移動元ディレクトリを移動先ディレクトリ内にうつす。移動先ディレクトリが存在していなければ、ディレクトリの名前が移動先ディレクトリ名に変更される。
ディレクトリ	ファイル	(この操作は認められていない)

- `cp` ファイルやディレクトリのコピー

基本は `cp コピー元 コピー先` の形で使います。

コピー元	コピー先	結果
ファイル	ディレクトリ	コピー元ファイルをコピー先ディレクトリ内に複製する
ファイル	ファイル	コピー元ファイルをコピー先のファイル名で複製する。コピー先ファイルが存在していた場合、上書きされる。
ディレクトリ	ディレクトリ	コピー元ディレクトリをコピー先ディレクトリ内にうつす。移動先ディレクトリが存在していなければ、ディレクトリを別名で複製する。コピー元にディレクトリを指定する場合 <code>-r</code> オプションを指定する必要がある (<code>cp -r コピー元 コピー先</code>)
ディレクトリ	ファイル	(この操作は認められていない)

`移動元` や `コピー元` に複数のファイルやディレクトリを指定することもできます。

基本操作8. head, tail, wc

サンプルデータのあるディレクトリに移動します。

```
cd ~/basic_course_2023/1/test1
pwd
```

- `head` ファイルの先頭を表示

```
head test1-1.gbk
```

先頭10行を表示します。

オプションで行数指定もできます (100行表示の例)

```
head -100 test1-1.gbk
```


- `tail` ファイルの末端を表示

末端から100行を抽出し、結果をファイルに書き出してみます。

```
tail -100 test1-2.gff > out.txt
```

- `wc` ファイルの文字数や行数を数える (word count)

```
wc test1-1.gbk
```

表示される数値は順に、行数、単語数、文字数を表します。文字数には改行コードも含まれます。

`-l` をつけると行数のみ表示されます。

```
wc -l test1-1.gbk
```

基本操作9. 大きなファイルの中身を閲覧 (less, more)

`cat` コマンドはファイルの中身を一気に表示させるのに対し、大きなファイルを部分ごとに表示させるのが`less`や`more`コマンドです。

`less`と`more`は多少の違いはありますが、`less`は`more`の拡張版ですので、`less`を使うことの方が多いです。

```
less test1-1.gbk
```

矢印キーの上下左右でスクロール。`f` (またはスペース)と `b` で1画面ずつそれぞれ上下にスクロールします。

`h` でヘルプが表示できます。`q` で`less`を終了します。

lessコマンドは、デフォルトでは画面の右端で自動的に折り返されて表示されます。この設定を無視して画面1行にファイルの内容を1行ずつ表示させる場合には `-S` オプションを使用します。

```
less -S read.fastq
```

パイプ

`|` (読み方はパイプ、パイプライン、バーチカルバーなど)を使うとコマンドの出力を他のコマンドの入力に連結することができます。

次の例ではファイルの先頭100行を抽出した後、その末尾10行を表示させています。

```
head -100 test1-1.gbk | tail -10
```

結果的にファイルの91行目から100行目までが表示されることになります。

複数のパイプを連結することも可能です。以下は、上記の結果に対して `wc` を実行したものです。

```
head -100 test1-1.gbk | tail -10 | wc
```

ファイル内の検索 (grep)

ファイルの中から文字列“LOCUS”という文字列を含んだ行を抽出して表示させます。

```
grep LOCUS test1-1.gbkl
```

オプション `-c` をつけると抽出された行の数を表示します。

```
grep -c LOCUS test1-1.gbkl
```

パイプと組み合わせて使用することもできます

```
qstat -u "*" | grep qw
```

上記コマンドはスパコンの実行中ジョブの一覧から待機状態(qw)状態のジョブを表示させています。

ワイルドカードを使って複数のファイルを対象に検索することもできます。

```
grep contig03 test*
```

プログラムの実行 (パスを通す)

`~/basic_course_2023/1` の中にある `hello` というプログラム (画面に"Hello world!"と表示させるだけのプログラム)を実行する方法を説明します。

ディレクトリに移動

```
cd ~/basic_course_2023/1/
```

実行権限の付与

パーミッションの確認

```
$ ll hello
-rw-r--r-- 1 koshu1 koshu 18 10 19 18:55 hello
```

実行権限 **x** が付けられていないのでこのままではプログラムとして実行することができません。

chmod コマンドで実行権限を付与します。

```
chmod a+x hello
```

(すべてのユーザー **a** に対して実行権限 **x** をつけるという意味です)

確認

```
$ ll hello
-rwxr-xr-x 1 koshu1 koshu 18 10 19 18:55 hello
```

実行方法

相対パスで指定する方法

(下記で先頭の\$はプロンプトを示していますので入力する必要はありません)

```
$ ./hello
Hello world!
```

カレントディレクトリにあるプログラムを実行する場合、明示的に **./** をつける必要があります。(**.** はカレントディレクトリを表す記号)

絶対パスで指定する方法

```
$ ~/basic_course_2023/1/hello
Hello world!
```

これ以外にプログラムがある場所にパス (PATH) を通して実行する方法もあります。以下でその方法を説明します。

パスを通す

`ls` コマンドを実行するとき、`ls` コマンドのプログラム本体はどこにあるのでしょうか？ `which` コマンドを使ってプログラム本体の場所を調べることができます。

```
$ which ls
alias ls='ls --color=auto'
/usr/bin/ls
```

このようにプログラム本体は `/usr/bin/` ディレクトリの中にあることがわかります。

`ls` と打つだけでプログラムが実行できるのは、`/usr/bin/` にパス (正確にはコマンド探索パス) が通っているためです。コマンド探索パスの設定内容は**環境変数** `PATH` の中に記載されており `printenv` コマンドでその中身を確認することができます (`echo $PATH` でもOK)。

```
$ printenv PATH
/opt/pkg/gcc/9.2.0/bin:/home/geadmin/UGER/bin/lx-amd64:...省略...:/opt/pkg/curl/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/ibutils/bin:/sbin:/usr/sbin:/cm/local/apps/environment-modules/4.2.1/bin:/opt/dell/srvadmin/bin:/home/koshu1/.local/bin:/home/koshu1/bin
```

このように `/usr/bin/` が `PATH` に含まれているので、どこで `ls` コマンドを実行したとしても `/usr/bin/ls` を呼び出してプログラムを実行することができます。

* **環境変数**: さまざまなプログラムから共通して参照できる値が書き込まれた変数。`PATH` のようにシステムによって定義された設定項目が記載されていることが多い。自分で定義することもできる。

先に実行した `hello` プログラムは `~/basic_course/1` にあるのでこのディレクトリを `PATH` に追加すれば `ls` コマンドと同じようにどこからでもこのプログラムが実行できるようになります。

```
export PATH=~/basic_course_2023/1:$PATH
```

先頭に `$` をつけた `$PATH` は既存の `PATH` の設定値を表すので、上のコマンドは既存の値の先頭に `~/basic_course/1` を追加するという意味です。コロン `:` は各設定値の区切りを表しています。

正しく設定できれば `hello` と打つだけで (どこからでも) プログラムが実行できるようになります。

```
$ hello
Hello world!
```

このようにプログラムのある場所を `PATH` に追加することを**パスを通す**といいます。

なお、すでにパスが通った場所にプログラムを配置することでも同様の効果が得られます。デフォルトではホームディレクトリ配下の `~/bin` にパスが通されていますのでここにプログラムを置けば、プログラム名をコマンドとして実行することで動作させることができます。

参考 1) 次のように `$PATH` の後側に付け足すこともできます。 `PATH` は先頭に近い方から探索されていくため、同名のファイルがあった場合には先に見つかった方が優先されます。

```
export PATH=$PATH:~/basic_course_2023/1
```

参考 2) ログアウトすると `export` コマンドで設定した環境変数の内容は失われてしまいますので、ログインごとに再設定する必要があります。頻繁に使うツールについては、`export` コマンドを `.bash_profile` ファイル (または `.bashrc` ファイル) の中に記述するとログイン時に自動で設定されるようになります。

参考 3) `PATH` を設定するときに間違えると設定された値が消えてしまい、コマンドが使えなくなることがあります。そのような場合にはいったんログアウトしてから再ログインすれば復活します。

かつては配布されているプログラムの多くが自分でインストールし、パスを通して使えるようにする必要がありました。現在ではそれらの作業を自動で行ってくれるパッケージ管理ツールの利用が広がっていますので自分でパスを通す作業を行うことも少なくなりました。

補遺

以下では講習中は扱いませんが比較的良好に使うコマンドやTIPSを紹介します。

インターネット経由でファイルを取得する

`wget` URLを指定してインターネット上のファイルをダウンロードします。

例) DDBJのFTPサーバからDRAに登録されたFASTQファイルを取得

```
wget ftp://ftp.ddbj.nig.ac.jp/ddbj_database/dra/fastq/SRA117/SRA117449/SRX456287/SRR1151187_1.fastq.bz2
```

`curl` コマンドを使ってもファイルを取得することができます。

ファイルの展開および圧縮

zip 形式 → 展開 unzip、圧縮 zip
gz 形式 → 展開 gunzip、圧縮 gzip
bz2 形式 → 展開 bzip2、圧縮 bzip2
tar or tar.gz 形式 → 展開および圧縮 tar コマンド

例) tar.gzファイルの展開

```
tar xvfz archive_file.tar.gz
```

xは展開（extract）, vは展開の過程を画面に表示（verbose）を表します。zは対象がgz形式で圧縮されている場合につけますが、最近はつけなくても展開できます。fは展開する対象がファイルであることを示します。 `tar xf archive_file.tar.gz` だけでも同じ結果になります

ローカルマシンとリモートサーバ (スパコン) との間でのファイルやディレクトリの転送

`scp` コマンドを使います。以下はローカルマシン上にある `some_file` というファイルをホームディレクトリにコピーしています。

```
scp some_file USERNAME@gw.ddbj.nig.ac.jp:~/
```

スパコンから他のサーバ等に転送することも可能です。(ただしそのマシンが外部からアクセスできる必要がある)

スパコン上でテキストファイルを編集する (vi, emacs)

`vi` や `emacs` といったエディタが利用可能です。

本講習ではSFTPクライアントソフトを使ってリモートのファイルを編集しますので、

これらは使用しませんが、遺伝研スパコンを含め、一部のLinuxシステムではviが標準のエディタになっているため、最低限の操作方法は覚えておいた方が良いでしょう。

`vi` で起動。または `vi ファイル名` で既存のファイルを開いて起動。

コマンドモードと入力モードに分かれています。入力モードに切り替えるには `i`、コマンドモードに戻るときには `ESC` キーを押します。コマンドモードにおいて `:w` で保存、`:q` で終了します。

qlogin時のオプション

デフォルトではqlogin時に4GBのメモリが割り当てられます。オプションを指定することで増やすことができます。

```
qlogin -l mem_req=20G,s_vmem=20G
```

`mem_req` と `s_vmem` には同じ値を指定してください。大量のメモリを要求した場合、空きがなくログインできない場合があります。ログインノードに大量のメモリを要求するのはできるだけ開発や動作テストなどの目的だけにとどめ、実際に計算処理を行うときには `qsub` コマンドを使って計算ノードにバッチジョブとして投入するほうが良いです。

GPUを使用する場合には `-l gpu` を指定します。

```
qlogin -l gpu
```

GPU搭載ノードにログインすることができます。GPU搭載のログインノードは1台しかないので、このオプションの使用についても開発や動作テスト目的にとどめ、実際の計算処理時にはバッチジョブを投入したほうが良いです。

Javaプログラムの実行方法

Javaプログラムを実行しようとするときメモリが足りないとのエラーが出る。

```
$ java
Error occurred during initialization of VM
```

```
Could not allocate metaspace: 1073741824 bytes
```

環境変数 `MALLOC_ARENA_MAX` に小さな値を設定すると実行できるようになる。

```
$ export MALLOC_ARENA_MAX=3
$ java
Usage: java [-options] class [args...]
           (to execute a class)
    or java [-options] -jar jarfile [args...]
           (to execute a jar file) ... 以下省略
```

参考) <https://sc.ddbj.nig.ac.jp/software/java/>

スパコン接続の手間を軽減する

`~/.ssh/` ディレクトリ内に `config` という名称のファイルを作り、以下のように記載しておくと `ssh nig` と打つだけでスパコンに接続できます。

```
Host nig
  HostName      gw.ddbj.nig.ac.jp
  Port          22
  User          自分のアカウント名
  IdentityFile  ~/.ssh/id_rsa
  ServerAliveInterval 60
  TCPKeepAlive yes
```

最後の2行目は一定時間操作されなかったときに自動で接続が切断されてしまうのを防ぐために記載しています。

qlogin時にパスワード要求された場合の回避策

現在はqlogin時にパスワードが不要のようにデフォルトで設定されているはずですが、以前にアカウントを取得した場合にはqlogin時にパスワードを聞かれる場合があります。

詳細は省きますが下記のようにすることでパスワード認証のかわりに鍵認証でログインできます。

1. 公開鍵ファイルがすでに存在しているか確認

```
ls ~/.ssh/id_rsa.pub
```

上記コマンドで鍵が存在していれば次のステップはパスしてください。

2. 秘密鍵・公開鍵を生成

```
ssh-keygen
```

ファイルの保存場所やパスフレーズを聞かれますが、すべてデフォルトのまま（空欄のまま）エンターキーを押します。

3. 公開鍵を登録

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```