#RICORA

コマンドライン入門講座

RICORA Programming Team

はじめに

- 本日はRICORA Programming Teamの活動にお越し頂きありがとうございます🥳
- 今回はCUIの使い方の基本を学びます📝
- 本日使用するターミナルエミュレータ
  - Mac -> Terminal
  - Windows -> PowerShell

準備

Windows

1. PowerShellを起動する
2. 以下のコマンドを入力する(理科大Wi-Fi内で)

```
$ ssh (学籍番号)@tusedls00.ed.tus.ac.jp
```

はじめの英語文に `yes` を入力して、自分のパスワードを入力してください。パスワードは見えませんが、入力されています。

詳しい人は、wsl2, Git Bash, Cygwinなどをインストールしてみよう。

準備 つづき

MacOS

1. Terminalを起動する

ディレクトリとは

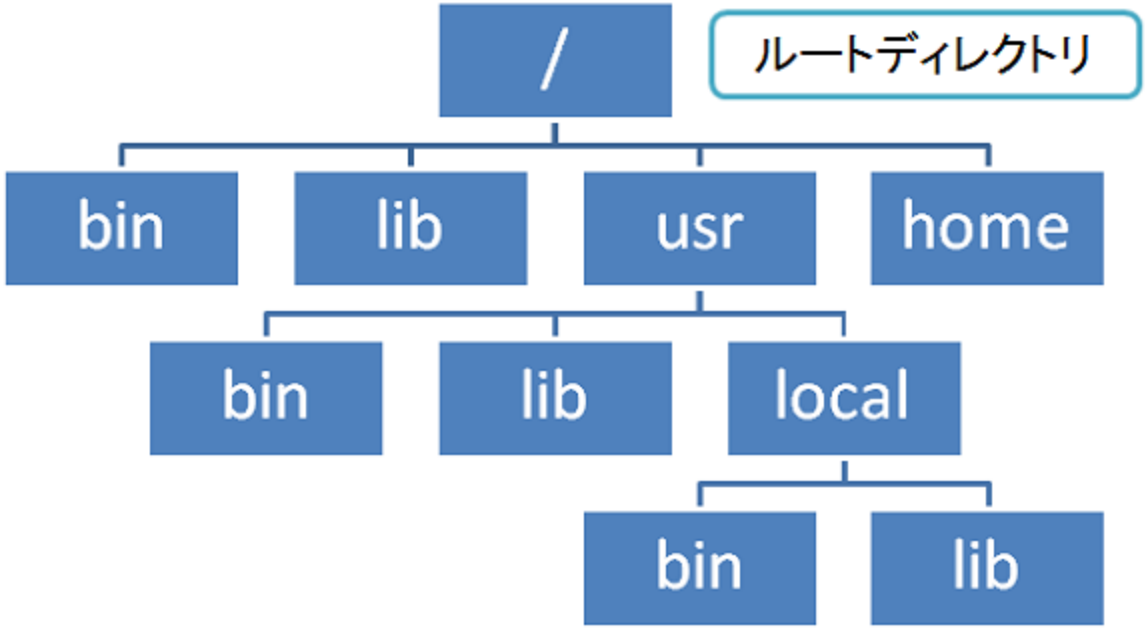
ファイルを格納する場所のことをいいます

ディレクトリ ≒ フォルダ

特に、Unix系のファイルシステムでは「ディレクトリ」, Windows系のファイルシステムでは「フォルダ」といいます

ディレクトリの構造

ディレクトリは**ルートディレクトリ**という大本のディレクトリを**根**とした**木構造**をしています



画像の転載元

ディレクトリの基本知識

#RICORA

名前	記号	説明
ルートディレクトリ	/	基準となるディレクトリ
ホームディレクトリ	~	シェルへのログイン時のディレクトリ
カレントディレクトリ	.	自分がいまいるディレクトリ
親ディレクトリ	..	カレントディレクトリの一つ上のディレクトリ

パスってなに？

目的のファイル・ディレクトリへの*道筋*のことです

「どこから道案内をするか？」で二通りの書き方があります

絶対パス

ルートディレクトリからの*道筋*を**絶対パス**といいます

例1

```
/home/shinkan/document/test.txt
```

パスってなに？　つづき

相対パス

カレントディレクトリからの*道筋*を**相対パス**といいます

例2 今、ホームディレクトリにいるとして・・・

```
./document/test.txt
```

コマンドってなに？

#RICORA

プログラムが書かれたファイルをごにょごにょすると機械語のファイルにできます（詳しくはあとで部員に聞いてね！）

そういう実行ファイルを**コマンド**といいます。

たとえば...

- *echo* echo コマンドの後に書いたことを出力してくれる
- *ls* いまいるディレクトリのファイルを全て表示してくれるよ

コマンドの使い方

#RICORA

コマンドを実行するときに、*引数*というものをとってもらうことができます。
引数はコマンドのあとにスペースをあけて入力します。

```
$（コマンド）（引数1）（引数2）（引数3） ...
```

例1. Cコンパイラ `gcc` でC言語ファイル `file.c` をコンパイルする

`gcc file.c`

コマンドの使い方 つづき

#RICORA

特に'-'(ハイフン)がついているものを**オプション**といいます。

例2. `-v`、 `--verbose` をつけるとより詳細な出力をしてくれるよ
例3. `-h`、 `--help` をつけるとヘルプ画面を表示してくれます

この後のコマンドでも引数やオプションを使う場面は多く出てきます

echo

引数に与えたものをそのまま出力する

```
$ echo hi
hi
$ echo run
run
```

pwd (print working directory)

いまいるディレクトリの場所を表示してくれます

こんな感じ：

```
$ pwd
/home/ricora/guidance/shinkan
```

cd (change directory)

ディレクトリに移動することができます

ここで引数が重要

引数なし `cd` はホームディレクトリ

```
$ cd
$ pwd
/home/ricora
```

cd (change directory) つづき

引数つき `cd` はその場所があればそこへ

- 相対パスの場合

```
$ cd ../shinkan
$ pwd
/home/shinkan
```

.(ドット)はいまいるディレクトリ、..(二重ドット)は一つ上のディレクトリという意味

cd (change directory) つづき

- 絶対パスの場合

```
$ cd /home/shinkan
$ pwd
/home/shinkan
```

ls (list)

いまいるディレクトリにあるファイル一覧を表示してくれます

```
$ ls
a.out hi.txt shinkan.pdf
```

引数に `-l` や `-a` をつけると詳しい出力結果が得られる

```
$ ls -l

drwxr-xr-x (なんとか) (なんとか) ./
drwxr-xr-x (なんとか) (なんとか) ../
-rwxr-xr-x (なんとか) (なんとか) a.out ...
```

mkdir (make directory)

#RICORA

ディレクトリを作ります

```
$ mkdir ricora-shinkan
$ ls
... ricora-shinkan ...
```

touch (change file timestamp)

ファイルを作ることができるコマンド

```
$ touch ricora-alg.hs
$ ls
... ricora-alg.hs ...
```

ファイルの作成日時は`ls -al`とかで確認できる

rm (remove)

ファイルを削除します

```
$ rm ricora-shinkan-0410.txt
```

ディレクトリを消すときは`-d`オプション、ディレクトリ内も削除するなら`-r`オプション

```
$ rm -rd ricora-shinkan-0410
```

man (manual)

コマンドのマニュアルを見るコマンド

```
$ man echo
```

マニュアルは英語だけど、ググるよりも欲しい情報が確実に載っている

/(スラッシュ)を打つと検索できる

環境変数と$PATHのはなし

#RICORA

シェルには環境変数というものがあります。`printenv`で出力してみよう。シェルの設定をする変数という感じです。

その中に`PATH`というものがあります。見つけられなかったら`printenv | grep -e '^PATH'`と打ってみよう

コマンドを打つとき、この`$PATH`にある場所を探しにいっています。

新しいコマンドをインストールするとき、この`$PATH`にその場所を追記するなどしないと動きません。

これ重要なので覚えておいてください。

やってみよう

聞きっぱなしで飽きたと思うので、実際に手を動かしてみましょう！

スライドは（多分）配布するので、それを見つつやってください。

1. シェルで'Hello, world!'を出力してみる

   出力例:

```
Hello, world!
```

   ヒント: `echo`

やってみよう

2. `gcc` のバージョンを確認しよう

   C言語のコンパイラ、 `gcc` のバージョンを確認しましょう。

   出力例:

```
$ gcc --version
gcc (...) 9.x.x
```

やってみよう

3. ファイルを作ってみよう

   ファイルを作ったら `ls` を打ってファイルを確認しましょう。

   出力例:

```
$ ls
..  (ファイル名) ..
```

   ヒント: `touch`

やってみよう

4. ファイルを削除してみよう

   ファイルを削除したら `ls` を打ってファイルがないことを確認しましょう。

   出力例:

```
$ ls
..  (ファイル名) ..
```

   ヒント: `rm`

やってみよう

5. ディレクトリをつくって、その中に移動してみよう

   ディレクトリをまず作って、別のコマンドでそのディレクトリに移動しましょう。

   出力例:

```
$ pwd
/home/(username)/.../(つくったディレクトリ名)
```

   ヒント: `mkdir` `cd`

やってみよう

6. ファイルの中身を出力してみよう

   ファイルの中身を出力するには、`cat` というコマンドを使います。

   出力例:

```
$ cat
(ファイルの中身)
```

やってみよう

7. ディレクトリの中からテキストファイルを検索してみよう

   まず、拡張子が `.txt` のファイルを作りましょう。
   その後、`find` というコマンドを使って拡張子が `.txt` のファイルを探してみましょう。

   出力例:

```
$ find ...
a.txt ricora.txt ...
```

   ヒント: `man find` ワイルドカード(*)

ご清聴ありがとうございました

#RICORA

もっと知りたい人に

リダイレクト

出力結果を標準出力以外に書き出します。

やってみよう

`echo` で一行の `python` プログラムを書き、ファイルにリダイレクトしてから `python` プログラムを実行しましょう。

パイプ

出力結果を別のコマンドに与えます。

（さっきも出てた）例:

```
$ printenv | grep -e '^PATH'
```

やってみよう

`ls` で出てきたファイルを `cat` して、そのバイト数をカウントしてみましょう。
ヒント: `xargs` <- ググって、 `man wc`

ファイルディスクリプタ

#RICORA

Linuxでファイルを開くとき、ファイルディスクリプタという数字が与えられます。以下のCプログラムを入力してみてください。

```c
#include<unistd.h>

int main() {
  const char buffer[6] = "head\n";

  // 0, 1, 2, 3番目のfdに書き込んでみましょう
  for (int fd = 0; fd <= 3; fd++) {
    write(fd, buffer, 6);
  }
  // なぜfd = 3のときに出力されないか考えてみましょう。
}
```

ほんとのほんとにおわり

#RICORA