

情報基礎 B

第 2 回

イントロダクション

長江 剛志

(nagae@m.tohoku.ac.jp)

東北大学大学院工学研究科
技術社会システム専攻

2015 年 4 月 10 日 (金)

ファイルとディレクトリ (1)

ファイルとは

OS が管理するデータ・セットの最小単位.

ファイル名と拡張子

ファイル名は, 通常,

`xxxx.yyy`

という構造をしている. 前半の `xxxx` はユーザーが自由に決めてよいが, 後半の `.yyy` の部分は **拡張子** と呼ばれ, ファイルの種類に応じて定められている.

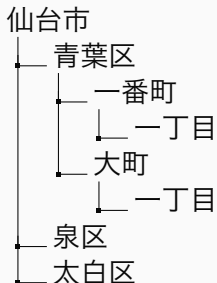
拡張子	ファイルの種類
<code>.txt</code>	テキスト・ファイル
<code>.tex</code>	T_{E} X ソースファイル
<code>.dvi</code>	DVI ファイル
<code>.c</code>	C 言語ソースファイル
<code>.docx, .xlsx</code>	Microsoft Word/Excel

ファイルとディレクトリ (2)

ディレクトリ (フォルダ)

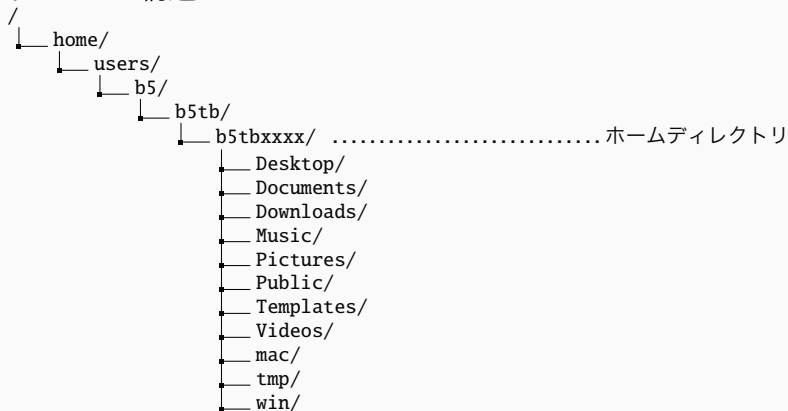
名前だけで管理するのは判りやすそうだが、ファイル数が多くなると、ファイル名の重複チェックが大変。そこで、ファイルを **階層的に管理** できるように **ディレクトリ (フォルダ)** が使われる。

住所も階層的に管理されている:



ファイルとディレクトリ (3)

情報教育 (ICL: **I**nformation and **C**omputer **L**iteracy) システムの
ディレクトリ構造:

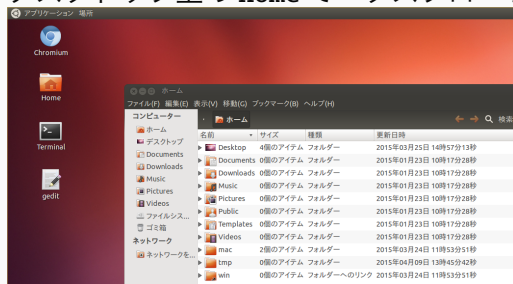


xxxx には学籍番号の下 4 桁が入る. 各学生ごとにホームディレクトリは異なる.

ファイルとディレクトリ (4)

ホームディレクトリ内のファイル確認方法

デスクトップ上の home でエクスプローラ的なウィンドウが開く



エディタ (gedit) の使い方 (1)

テキストエディタ

「情報基礎 B」ではテキストファイルを編集する **テキストエディタ** (Windows のメモ帳に相当) を利用. ICL システム にインストール済のエディタは

1. gedit (デスクトップ上に表示)
2. GNU Emacs 23 (アプリケーション→プログラミング)
3. TurtleEdit (アプリケーション→プログラミング)

の 3 種類. どれを使ってもよいが, ここでは gedit を使うことにする. ちなみに長江は 20 年以上の **Emacs 信者者**.

エディタ (gedit) の使い方 (2)

演習 1

自分の学籍番号と名前を入力したテキストファイルを作成し、
~/Documents ディレクトリに MyName.txt という名前で保存せよ

例) b5tb9999 仙台 一郎

日本語入力のキーバインド


全角/半角	日本語入力の切り替え	→	次の文節へ移動
Space	変換候補の表示	←	前の文節へ移動
Enter	変換を確定	Shift + ←	文節を短く
		Shift + →	文節を長く

現在の日本語入力モードは画面右下の言語パネルで確認できる



エディタ (gedit) の使い方 (3)

~/Documents ディレクトリに MyName.txt として保存

1. メニューバーの  保存 アイコンをクリック
2. 保存の設定ウィンドウが表示されるので、
 - ▶ Documents ディレクトリを選択
 - ▶ ファイル名 MyName.txt を入力
 - ▶ 「保存」 ボタンをクリック



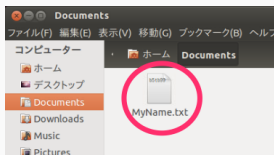
エディタ (gedit) の使い方 (4)

正しく保存されているか確認

1. 正しく保存されると, gedit のタイトルバーに MyName.txt (~/.Documents) と表示される



2. デスクトップ上から Home を開いて, Documents ディレクトリ内に MyName.txt ファイルがあることを確認する



ファイル名に関する注意

ファイル名は、原則、

- ▶ 半角英数 (大文字でも小文字でもよい)
- ▶ 数字
- ▶ _ (アンダーバー), - (ハイフン)

のみで構成すること。全角文字、スペース、括弧や?!などの記号は使ってはいけない。

下のようなファイル名は、特定の状況下でしか正常に動作しない。言い換えれば、こうした名前をつけたファイルを、別の環境下で開いたり、他人とやり取りしたり場合には、ほぼ確実にトラブルと正しい。

- ▶ My Name.txt (半角スペースを含む)
- ▶ My.Name.txt (ファイル名にピリオドを含む)
- ▶ 私の名前.txt (全角文字を含む)

演習 2 (1)

おくのほそ道 の序文を入力したテキストファイルを作成し、
MyName.txt と同じ Documents ディレクトリの中に
Oku_no_Hosomichi.txt という名前で保存せよ。

おくのほそ道 (序文)

月日は百代の過客にして、行かふ年も又旅人也。
舟の上に生涯をうかべ、馬の口とらえて老をむかふる物は、日々旅にして旅を栖とす。
古人も多く旅に死せるあり。
予もいづれの年よりか、片雲の風にさそはれて、漂泊の思ひやまず、海浜にさすらへ、去年の秋江上の破屋に蜘蛛の古巣をはらひて、やゝ年も暮、春立る霞の空に白川の関こえんと、そゞる神の物につきて心をくるはせ、道祖神のまねきにあひて、取もの手につかず。
もゝ引の破をつづり、笠の緒付かえて、三里に灸すゆるより、松島の月先心にかゝりて、住る方は人に譲り、杉風が別荘に移るに、
草の戸も住替る代ぞひなの家
面八句を庵の柱に懸置。

演習 2 (2)

(今後は使うことはない入力上の) ヒント

- ▶ 「ゝ」は「くりかえし」を変換すると出せる
- ▶ 「ゞ」は「くりかえしだくてん」を変換すると出せる
- ▶ 「草の戸も」の前の空白 (全角スペース) は日本語入力モードでスペースキーを入力すると出せる

インターフェイス (CUI と GUI) (1)

ユーザーとコンピュータとの間のやりとりを担う仕掛けを **UI (User Interface)** と呼ぶ.

GUI (*Graphical User Interface*)

ウィンドウやアイコンをマウスで操作し、キーボードで文字入力を行う

例) 一般的な *Windows/Mac OS* アプリ. *ICL* システムでは *gedit* や *Home* など.

CUI (*Character User Interface*)

キーボード入力と画面の文字表示のみで操作する

例) *Windows* の「コマンドプロンプト」, *Mac OS* の「ターミナル」, *ICL* システムでは *Terminal* など.

インターフェイス (CUI と GUI) (2)

GUI と CUI の長所/短所

	GUI	CUI
入力機器	マウス/トラックパッド とキーボード	キーボード
画面表示	文字だけでなく アイ コンやウィンドウ がグラ フィカルに表示され、操 作結果がすぐに反映さ れる	文字表示のみで、操作結果が必 ずしも表示されるとは限らない
導入し易さ	何も知らなくても 直感 的に操作できる	コマンドを覚えなければ何もで きない
操作に要する時間	長い	短い
複雑な処理	面倒	(コマンドを覚えれば) 簡単
繰返し処理	面倒	(コマンドを覚えれば) 簡単
リソース使用量	大量	少しでよい

両方習熟しておいて、必要に応じて使い分ける のが賢い

CUI による操作に慣れよう

こちらも参考に <http://seaotter.cite.tohoku.ac.jp/coda/jkiso2015/files/linux.pdf>
デスクトップ上の Terminal を起動

```
b5tb9999@zzzzz: ~$
```

と表示される (9999 や zzzz の部分は個人ごとに異なる).
この状態は **コマンドプロンプト** と呼ばれ,

- ▶ b5tb9999 という **ユーザー** が
- ▶ zzzzz という **端末** を使っていて
- ▶ ~ という **作業ディレクトリ** 上で

次のコマンドが入力されるのを待っています ということを表している.

コマンド入力 (1)

cal: カレンダーの表示

```
b5tb9999@zzzzz: ~$ cal ↵  
      4月 2015  
日 月 火 水 木 金 土  
          1  2  3  4  
 5  6  7  8  9 10 11  
12 13 14 15 16 17 18  
19 20 21 22 23 24 25  
26 27 28 29 30
```


コマンド入力 (2)

gedit: エディタの起動

gedit の起動

```
b5tb9999@zzzzz: ~$ gedit ↵
```

gedit が終了するまでコマンドプロンプトが現れない (=次のコマンドを受け付けない).

バックグラウンドでの実行

& をつけてコマンドを呼び出すと, バックグラウンド で実行される.

```
b5tb9999@zzzzz: ~$ gedit & ↵  
b5tb9999@zzzzz: ~$
```

gedit が起動した後, コマンドプロンプトが現れる.

ファイルとディレクトリの確認

pwd: 現在の作業ディレクトリの確認

```
b5tb9999@zzzzz: ~$ pwd ↵  
/home/users/zz/d54/d54a0rtt
```

実行結果は個人ごとに異なる。

ls: ファイル一覧の表示

引数を与えないと、現在のディレクトリ内の一覧が表示される。

```
b5tb9999@zzzzz: ~$ ls ↵  
Desktop      Music          Templates      tmp  
Documents    Pictures       Videos        win  
Downloads    Public         mac
```

引数にディレクトリを指定すると、その中の一覧を表示する。

```
b5tb9999@zzzzz: ~$ ls Documents ↵  
MyName.txt    Oku_no_Hosomichi.txt
```

相対パスと絶対パス (1)

CUIでファイルやディレクトリの場所は **パス** (path) で指定する。
例えば, ホームディレクトリ (~) の中の Documents ディレクトリ
のパスは

```
~/Documents/
```

であり, その下の MyName.txt というファイルのパスは,

```
~/Documents/MyName.txt
```

である.

パスの指定方法には, ホームディレクトリ (~) を基準として, それ以下のディレクトリ/file を全て記述した **絶対パス** 方式¹ と, 現在の作業ディレクトリを基準とした **相対パス** 方式がある.

相対パスと絶対パス (2)

相対パス 方式では、現在の作業ディレクトリより下のディレクトリ/ファイルのみを指定する。例えば、現在の作業ディレクトリが `~` である場合、上述の `MyName.txt` への相対パスは

`Documents/MyName.txt`

となる。現在の作業ディレクトリが `~/Documents` である場合は、

`MyName.txt`

となる。

相対パスと絶対パス (3)

現在の作業ディレクトリ以上の階層の相対パス

相対パス 方式では,

- ▶ 現在の作業ディレクトリは ./
- ▶ 現在の作業ディレクトリより 1 つ上の階層のディレクトリ (親ディレクトリ) は ../

で表される. さらに上の階層を指定するには, 階層の数だけ ../ を繰り返す.

- ▶ 2 つ上のディレクトリ (祖父ディレクトリ) は ../../
- ▶ 3 つ上のディレクトリ (曾祖父ディレクトリ) は ../../../../

相対パスと絶対パス (4)

兄弟ディレクトリ

../ をうまく使うと、兄弟ディレクトリや従兄弟ディレクトリを相対パスで指定できる。例えば、現在の作業ディレクトリが ~/Documents だった場合、同じホームディレクトリ下にある ~/Pictures ディレクトリは、

../Pictures

で指定できる。

¹一般にはルートディレクトリ / を基準としたものを指す

cd: ディレクトリの移動 (1)

引数に指定されたディレクトリに移動する. ディレクトリ名の後の / は省略可能. ホームディレクトリから子ディレクトリの Documents に移動するには

```
b5tb9999@zzzzz: ~$ cd Documents ↵  
b5tb9999@zzzzz: ~/Documents$ pwd ↵  
/home/users/zz/b5/b5tb9999/Documents
```

Documents から親ディレクトリ (~) に戻るには

```
b5tb9999@zzzzz: ~/Documents$ cd .. ↵  
b5tb9999@zzzzz: ~$ pwd ↵  
/home/users/zz/b5/b5tb9999
```


cd: ディレクトリの移動 (2)

cd で引数を省略した場合には、現在の作業ディレクトリがどこであろうと、ホームディレクトリ (~) に戻る.


```
b5tb9999@zzzzz: ~$ cd /usr/local/bin ↩ 適当なディレクトリに移動する
b5tb9999@zzzzz: /usr/local/bin$ cd ↩ 引数を与えずに cd を実行
b5tb9999@zzzzz: ~$ pwd ↩
/home/users/zz/b5/b5tb9999
```


便利な操作方法 (TAB 補完とコマンド履歴) (1)

TAB 補完

コマンドや引数 (cd の後のファイル名など) を途中まで入力して  を押すと、残りを自動的に入力 (補完入力) してくれる。

```
b5tb9999@zzzzz: ~$ ls Doc   
b5tb9999@zzzzz: ~$ ls Documents <- 残りの「uments」を自動で入力
```

候補が複数ある場合は  を 2 回押すと表示

```
b5tb9999@zzzzz: ~$ ls Do    
Documents/      Downloads/
```

コマンド履歴


コマンドプロンプトでカーソルキーの を押すと以前に入力したコマンドを次々に表示してくれる

色々なコマンドを覚えよう (1)

~/Documents/ ディレクトリに移動 (cd) してから以下を実行してみよう.

less: ファイルの中身を表示

```
b5tb9999@zzzzz: ~/Documents$ less MyName.txt
```

MyName.txt の中身が表示される. 元の画面に戻るには 

cp: ファイルをコピー

```
b5tb9999@zzzzz: ~/Documents$ cp MyName.txt MyName2.txt
```

MyName.txt の複製を MyName2.txt という名前で保存


mv: ファイルの移動/別名


```
b5tb9999@zzzzz: ~/Documents$ mv MyName2.txt MyName3.txt
```

MyName2.txt を MyName3.txt という名前に変更

色々なコマンドを覚えよう (2)

`rm -i`: ファイルを消去

```
b5tb9999@zzzzz: ~/Documents$ rm -i MyName3.txt
rm: 通常ファイル 'MyName3.txt' を削除しますか? y 
b5tb9999@zzzzz: ~/Documents$
```

MyName3.txt を消去. `-i` は, 消去する前に確認してくれるオプション. 削除しない場合は  と入力すればキャンセルできる. Terminal で消去したファイルは「ゴミ箱」などには入らないので **必ずつけること**¹ ファイルを消すか否か確認して欲しくないときは `-f` オプションをつける.

```
b5tb9999@zzzzz: ~/Documents$ rm -f MyName3.txt
b5tb9999@zzzzz: ~/Documents$
```

けっこう怖い.

色々なコマンドを覚えよう (3)

mkdir: ディレクトリを作成

```
b5tb9999@zzzzz: ~/Documents$ mkdir dir1
```

dir1 という名前のディレクトリを作成する

cp, mv で別のディレクトリにコピー/移動

```
b5tb9999@zzzzz: cp MyName.txt dir1
b5tb9999@zzzzz: mv MyName.txt dir1/MyName2.txt
b5tb9999@zzzzz: ls dir1
MyName.txt      MyName2.txt
```

dir1 の中にコピー/移動できている

色々なコマンドを覚えよう (4)

rmdir: ディレクトリの削除

```
b5tb9999@zzzzz: ~/Documents$ rmdir dir1
rmdir: 'dir1' を削除できません: ディレクトリは空ではありません
```

ディレクトリの中にファイルがあると消せない.

```
b5tb9999@zzzzz: ~/Documents$ rm dir1/MyName.txt
rm: 通常ファイル 'MyName.txt' を削除しますか? y ☐
b5tb9999@zzzzz: ~/Documents$ rm dir1/MyName2.txt
rm: 通常ファイル 'MyName2.txt' を削除しますか? y ☐
b5tb9999@zzzzz: ~/Documents$ rmdir dir1
```

空になったディレクトリは削除できる.

¹ICL システムでは `rm` の別名として `rm -i` が設定されているので, 明示的に `-i` をつけなくても消去するかどうかを確認してくれる.

コマンドを使ってみよう!!

1. ~/Documents ディレクトリに移動
2. `ls` を使って `MyName.txt` と `Oku_no_Hosomichi.txt` があることを確認
3. `intro` という名前のディレクトリを作成
4. `cp` を使って `MyName.txt` と `Oku_no_Hosomichi.txt` を `intro` の中に **コピー**
5. `cd` を使って `intro` に移動
6. `ls` を使って上記2つのテキストファイルの複製があることを確認
7. `less` を使って `Oku_no_Hosomichi.txt` の中身を確認
8. `mv` を使って `Oku_no_Hosomichi.txt` を `OnH.txt` という名前に変更
9. `cd ..` を使って親ディレクトリ `~/Documents` に戻る
10. `rm -i` を使って `~/Documents` 直下の `MyName.txt` と `Oku_no_Hosomichi.txt` を削除

tar: ファイルをひとまとめにする (1)

複数のファイルやディレクトリをまとめて1つの **書庫ファイル** として保存できる. **-z** オプションを使うと **圧縮** までしてくれる.

- ▶ 書庫ファイルを作る: `tar -czvf 書庫ファイル名 まとめたファイル/ディレクトリ`

```
b5tb9999@zzzzz: ~/Documents$ tar -czvf 0417-intro intro
intro/
intro/MyName.txt
intro/Oku_no_Hosomichi.txt
```

- ▶ 書庫ファイルの中身を確認する: `tar -tf 書庫ファイル名`

```
b5tb9999@zzzzz: ~/Documents$ tar -tf 0417-intro
intro/
intro/MyName.txt
intro/Oku_no_Hosomichi.txt
```

tar: ファイルをひとまとめにする (2)

- ▶ 書庫ファイルを展開する: `tar -xzvf` 書庫ファイル名 `intro` を別の名前に変更しておく

```
b5tb9999@zzzzz: ~/Documents$ mv intro intro2
```

`ls` して `intro` ディレクトリが無いことを確認

```
b5tb9999@zzzzz: ~/Documents$ ls
0417-intro      intro2
```

`tar -xzvf` を使って書庫ファイルを展開

```
b5tb9999@zzzzz: ~/Documents$ tar -xzvf 0417-intro
intro/
intro/MyName.txt
intro/Oku_no_Hosomichi.txt
```

`ls` して `intro` ディレクトリが復元できたことを確認

```
b5tb9999@zzzzz: ~/Documents$ ls
150417-intro.tar.gz      intro      intro2
```