Linux入門 (1)

正田 備也

masada@rikkyo.ac.jp

課題の出し方

- 2回後の土曜の23:59がしめきり
  - 「2回」というのは、休講日は含めずに数える

- Canvas LMSに提出してください

https://canvas.rikkyo.bownet.cloud/

Linux入門の意図

- MITの"The Missing Semester of Your CS Education"的な内容の授業をしたい！

https://missing.csail.mit.edu/

- ただし必要最小限の内容で。
  - LinuxのCLI
  - シェルスクリプト
  - LaTeXで文書作成
    - いずれもCS系の研究活動に必要

参考資料

Linux入門講座 （エンジニアの入り口）

https://eng-entrance.com/category/linux

Linux基本コマンドTips （atmarkIT）

https://www.atmarkit.co.jp/ait/series/3065/

In October 2018, BERT Large (Devlin et al., 2019) with 350 million parameters was the biggest Transformer model (Vaswani et al., 2017) ever trained. At the time, contemporary hardware struggled to fine-tune this model. The section "Out-of-memory issues" on BERT's GitHub[1] specifies the maximum batch size for BERT Large given 12Gb of GPU RAM and 512 tokens as **zero**. Four years in, publically available models grew up to 176 billion parameters (Scao et al., 2022; Zhang et al., 2022; Zeng et al., 2022), by a factor of 500. Published literature includes models up to 1 trillion parameters (Chowdhery et al., 2022; Shoeybi et al., 2019; Fedus et al., 2021). However, single-GPU RAM increased less than 10 times (to 80Gb) due to the high cost of HBM memory. Model size scales almost **two orders of magnitude quicker**

https://arxiv.org/abs/2303.15647

Linuxを使う環境を整える

Linuxを使うには？ (1/2)

- Macの場合
  - 最初から入っている（シェルはbashかzsh（ズィーエスエイチ））
- Windowsの場合
  - WSLを使ってLinuxをインストールする

Linuxを使うには？ (2/2)

- LinuxをPCにインストール
  - 深層学習をする人はほとんどUbuntu

- Google Colaboratory
  - 「!」の後にLinuxコマンドをタイプしてセルを実行
  - セルの最初の行に「%%shell」と書いておく

Linuxを使う

CLI (command line interface)

- LinuxはWindowsのようにGUI (graphical user interface)でも使えるが…
- CLIで使うところに醍醐味（？）がある

- コマンドと呼ばれる文字列を入力して様々な作業をする
  - `ls`, `cd`, `cat`, 等。
- テキストファイルの編集もCLIで
  - `vim`（ヴィム）や`emacs`（イーマックス）
- コマンドを実行した結果も文字列で返ってくる
  - GUIなしでも結構いろいろできてしまう

ログインとログアウト

- Linuxマシンはログインして使う
  - ユーザ名とパスワードを入力
    - パスワードの変更は`passwd`コマンドで
- ログインしたら`last`コマンドを使ってみよう
  - `last`と入力しEnterを押す
    - 今後「Enterキーを押す」と逐一言わないことにします。

```
$ last
```

- ログアウトは`exit`コマンドで

```
$ exit
```

コマンドプロンプトのショートカットキー (1/2)

- tab 長いコマンドの補完入力
- Ctrl+c　処理のキャンセル（よく使う）
- Ctrl+a　コマンドラインの行の先頭へカーソルを移動
- Ctrl+e　コマンドラインの行の末尾へカーソルを移動
- 上下の矢印キー コマンドの履歴の中を行ったり来たりする

  - https://eng-entrance.com/linux_shortcut

コマンドプロンプトのショートカットキー (2/2)

- Ctrl+r　　コマンドの履歴を検索
- Ctrl+d　　カーソルの位置にある文字を削除
- Ctrl+w　　カーソルの位置の単語を削除
- Ctrl+y　　直前に削除した文字を貼り付け（Ctrl+dとは組み合わせられない）
- Ctrl+z　　処理の一時中断
  - コマンド「`fg %`」で直前に中断した処理を再開→後で説明
- Ctrl+l　　画面クリア

  - https://eng-entrance.com/linux_shortcut

rootユーザ

- rootユーザ＝管理者（スーパーユーザ）
  - rootのパスワードはLinuxインストール時に設定する
  - rootとしてログインすることはあまりない
  - 一般ユーザとしてログインしている場合、suコマンドで、必要に応じてrootになれるが・・・
  - sudoを使ってコマンド単位で一時的に管理者権限で作業をする
- rootになるとコマンドプロンプトが「$」から「#」に変わる

rootユーザの注意事項

- rootは何でもできる
  - 注意しないと危険です！！！
  - システムの設定を下手に触ると大変なことに！
  - rootで`rm`コマンド（ファイルを削除するコマンド）などを使うときは、最大限に注意！

macのrootパスワード設定

```
$ sudo passwd -u root
```

- これでrootのパスワードを設定できるはず

suとsudo

- suコマンドは一般ユーザがrootになるとき使う
  - 「su」と入力するとrootのパスワードを聞かれる
  - rootのパスワードを知らないとsuコマンドは使えない！
    - suコマンドは他のユーザになるときにも使える
- sudoはコマンド単位で管理者権限の作業をするとき使う（こちらが推奨）
  - 「sudo コマンド」と入力すると自分のパスワードを聞かれる
  - 共有マシンの場合はsudoを使って作業するようになっていることが多い
    - rootのパスワードをユーザに知られないようにするため
    - sudoは一般ユーザが自分のパスワードで管理者権限の作業ができるようにするしくみ

    - https://eng-entrance.com/linux-root

シャットダウンと再起動

- シャットダウン

```
$ sudo shutdown -h now
```

- 再起動

```
$ sudo shutdown -r now
```

あるいは

```
$ sudo reboot
```

Linuxコマンド(1/4)

システムまわり

参考資料

Linuxコマンド集

https://eng-entrance.com/category/linux/linux-command

- この授業ではbash（バッシュ）を想定
  - シェル(shell)＝ユーザがコマンドでシステムを操作できるようにするしくみ
  - bashはいまユーザが最も多いと思われるシェル
    - 昔はtcshを使う人も多かった
    - 今はzsh（ズィーシェル、ズィーエスエイチ）を使う人が増えている？

自分の環境で使えるシェルの一覧

```
$ cat /etc/shells
```

- catコマンドは、ファイルの中身を見るコマンド（後で説明）

ls

- ファイルやディレクトリの一覧を表示するコマンド
- デフォルトでは今いるディレクトリ（カレントディレクトリ）について一覧を表示する
- 「-l」オプション：ファイル名だけでなくファイルの情報も表示する
  - 表示されているのはどのような情報か、ググって調べよう
- 「-a」オプション：名前が「.」で始まるものも表示する

```
$ ls
$ ls -al
$ ls -l /etc
$ ls -l /etc/*.conf
```

  - https://eng-entrance.com/linux_command_ls

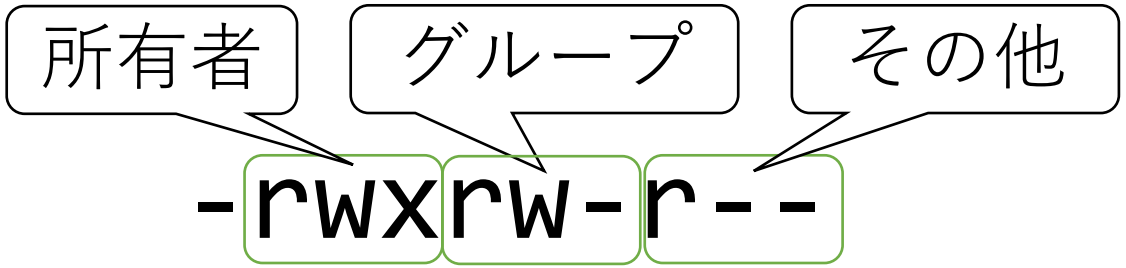
Linuxのユーザとグループ

- Linuxマシンを管理するときユーザのグループを作る
  - 「ken」というユーザを作成すると、自動的に「ken」というグループが作られる
- グループごとにファイルに対して何をしていいかを設定できる
- ファイルのパーミッション（権限）は3種類
  1. 所有者の権限
  2. グループの権限
  3. その他のユーザの権限
     - `ls`コマンドを「`ls -l`」と使って左端に表れるのがパーミッション
       - 例えば「`-rwxrw-r--`」など。この意味を調べよう。

- https://eng-entrance.com/linux-permission-basic

ファイルのパーミッション

- ファイルのパーミッション（権限）は3種類
  1. 所有者の権限
  2. グループの権限
  3. その他のユーザの権限
  - `ls`コマンドを「`ls -l`」と使って左端に表れるのがパーミッション



chmod (1/2)

- ファイルやディレクトリのパーミッションを変更するコマンド
- 「ls -l」で表示されるパーミッションの部分をイメージして使う
- 「chmod アクセス権 ファイル名」と使う
  - アクセス権はパーミッションの指定を2進数で書き、3桁ずつまとめて8進数にしたもの
    - 「rwxrwxrwx」と設定したい：「111111111」なので「chmod 777 hoge」
    - 「rwx------」と設定したい：「111000000」なので「chmod 700 hoge」
    - 「rwxr-xr-x」と設定したい：「111101101」なので「chmod 755 hoge」

chmod (2/2)

- 「chmod アクセス権 ファイル名」と使う
  - 現在のものに追加（「+」）、現在のものから削除（「-」） 、という方式でも書ける
    - その他のユーザが実行できないようにしたい：「chmod o-x hoge」
    - 自分以外の同じグループのユーザが読めるようにしたい：「chmod go+r hoge」
  - https://eng-entrance.com/linux-command-chmod

man

- マニュアルを見るコマンド

```
$ man ls
```

- manコマンド内でのショートカットキー
  - スペース：次のページ
  - b：前のページ
  - 上下カーソルキー：一行ずつ移動
  - /：文字列検索モードに入る（ctrl+cで抜ける）
    - 文字列を入力してEnterを押すとその文字列をマニュアル内で検索する
    - 記号はバックスラッシュでエスケープする
  - q：manから抜ける

  - https://eng-entrance.com/linux-command-man

date

- 日時を表示する

```
$ date
$ date "+%Y%m%d-%H%M%S"
$ date -d tomorrow
```

  - https://eng-entrance.com/linux-command-date

who

- 誰がログインしているかを表示する

```
$ who
```

  - https://eng-entrance.com/linux-command-who

top

- 実行中のプロセス（次スライドで説明）をリアルタイムで表示
- qキーを押すと終了
  - 他にも便利なショートカットキーがある（ググりましょう）

- 例えばこういうときに使います
  - マシンが重たくなったとき、原因になっているプロセスを調べる

  - topコマンドが出す情報はあまり正確でないという話もあります
    - 参照：「CPU使用率は間違っている」https://yakst.com/ja/posts/4575

Linuxのプロセス

- プロセス＝実行中のプログラム
- ゾンビプロセス
  - 役目を終えているのにメモリを開放していないプロセス
  - 共用のマシンではゾンビプロセスを走らせないように（他の人の迷惑になる）

- `top`コマンド（次のスライド）でリアルタイムにプロセスの状態を見ることができる
- `kill`コマンドで特定のプロセスを終了させることができる
  - 自分に権限があるプロセスだけ`kill`できる
  - 「`kill` プロセス番号」と入力すればプロセスを終了させることができる

  - https://eng-entrance.com/linux-process

ps

- 現在動いているプロセスを表示する

```
$ ps
$ ps aux
```

  - 「-」を付けないフォーマットでオプションを指定することが多い
  - 「-」を付ける方式のオプションとは混ぜて使えない

  - https://eng-entrance.com/linux-command-ps

演習1-1

- `top`コマンドを使って、一番CPUを使っているプロセスが何かを調べよう

sleep

- 時間を指定して、その間だけシステムを待機させる
- 例えば、システムを3秒待機させるには、以下のようにする

```
$ sleep 3
```

Linuxのジョブ

- ジョブはプロセスの集まり

- シェルのコマンドラインでコマンドを入力して「Enter」を押すと、ジョブが実行される
- そしてジョブが終わるまで、プロンプトが表示されない

- しかし、ジョブをバックグラウンドで実行させれば、そのジョブを走らせつつ、シェルで次のコマンドを打てるようになる

Linuxのバックグラウンド・ジョブ

- コマンドをバックグラウンドのジョブとして実行させるには、最後に「**&**」を付ける

```
$ sleep 60 &
```

  - 60秒間眠るというジョブは、バックグラウンドで実行される
  - そのため、すぐプロンプトが戻ってくる
    - 背後では、**sleep**コマンドが動き続けている
  - そしてまた別のジョブを実行させることができる

bg

- 例えば、システムを60秒、待機させる

```
$ sleep 60
```

- すると制御がsleepコマンドによる待機のほうへ移ってしまうが・・・
- 「Ctrl + z」と入力するとbashに制御が戻ってくる
- このとき、sleepコマンドは停止してしまう

- sleepコマンドをバックグラウンドで再開させるには、以下のようにする

```
$ bg %
```

  - bgは指定されたジョブをバックグラウンドで実行するコマンド
  - 「%」は直前に停止させたジョブの意味

jobsとbgとfg

- jobsコマンドで、ジョブの番号と状況を確認できる

- 「bg %ジョブ番号」で、特定のジョブをバックグラウンドで実行
- 「fg %ジョブ番号」で、特定のジョブをフォアグラウンドで実行

- フォアグラウンドで実行させると、そのジョブに制御が移る

- 「Ctrl + z」で、フォアグラウンドのジョブを一時停止

jobsとbgとfg

- ジョブ番号なしで「fg %」とだけ入力すると、直前に停止させたジョブをフォアグラウンドで再開できる

- 共用マシンでは、要らないジョブを走らせないようにしよう
  - 要らないジョブを止めるには・・・
  - jobsコマンドでジョブ番号を確認し・・・
  - fgコマンドでフォアグラウンドに移して、「Ctrl + c」で完全停止
  - やっぱりこのジョブ要るな、と思ったら・・・
  - 「Ctrl + z」で一時停止、「bg %」で、バックグラウンドへ送る

Linuxコマンド(2/4)

ファイル操作

pwd

- カレントディレクトリを表示する
  - **pwd** ＝ print working directory

演習1-2

- pwdコマンドを実行してみよう

`mkdir`と`rmdir`

- 新しいディレクトリを作成する
  - **`mkdir`** = make directory

- ディレクトリを削除する
  - **`rmdir`** = remove directory

演習1-3

- ホームディレクトリの下にSMLというディレクトリを作ろう

cd

- 指定したディレクトリへ移る
  - cd = change directory

- 特殊な使い方がある（次スライド）

ディレクトリの特殊な指定方法

```
$ cd
```

- ホームディレクトリへ移動する

```
$ cd .
```

- カレントディレクトリへ移動する（つまり何も起こらない）

```
$ cd ..
```

- 一つ上の階層のディレクトリへ移動する

課題1

- さっき作ったSMLディレクトリに移ろう
- SMLディレクトリの下に01というディレクトリを作ろう

- 上のことを実行した画面のスクリーンショットを提出

Linux入門 (1)の続き

正田 備也
masada@rikkyo.ac.jp

MacのFinderで隠れファイルを表示

- Cmd + Shift + . (dot) を押す

- ホームディレクトリにある「.」で始まるファイル
  - 各アプリケーションの設定が書いてあるファイル
  - あるいは、その下にそういうファイルが置いてあるディレクトリ

「No such file or directory」エラー

- 存在しないファイルやディレクトリにアクセスした時に発生
- 存在しないコマンドを実行しようとした時にも発生

- 私からの要望：エラーメッセージはきちんと読みましょう
  - 英語で表示されたとしても、きちんと読みましょう。
  - エラーメッセージをグーグル検索して、その意味を調べましょう。
  - それでも対処法がわからなければ、質問しましょう。

touch

- 中身が空の新規テキストファイルを作成する
  - 本来は、指定されたファイルのタイムスタンプを現在に変更するコマンド。

```
$ touch test.txt
$ touch test2.txt
```

  - lsコマンドを使って、ファイルが作成されたか確認してみよう

rm

- ファイルやディレクトリを削除する

  - https://eng-entrance.com/linux_command_rm

rmコマンドを安全に使う

- ファイルを誤って消さないよう、以下のようにaliasしておく
  - 「alias（エイリアス）する」＝aliasコマンドを使って別の名前をつける

- ホームの下にある「.bashrc」というファイルに、以下の一行を追加しておく（このファイルがなければ作る）

```
alias rm='rm -i'
```

  - こう書くと、「-i」オプション付きのrmコマンドに「rm」という別名を付けられる
  - すると、rmコマンドを実行するたびに、「-i」がなくても確認がなされるようになる

演習1-5

- SMLディレクトリの下の01ディレクトリに、touchコマンドで、test.txtというファイルとtest2.txtというファイルを作ろう
- rmコマンドで、今作ったtest2.txtファイルを削除しよう

vi

- Linux標準のテキストエディタを起動するコマンド
- viエディタの使い方は・・・自習してください
  - https://eng-entrance.com/linux-command-vi
  - ショートカットキーを覚えることが重要です

- 他のエディタを使ってもいいです

viの主なショートカットキー

- viエディタ起動時はコマンドモード
- コマンドモードで「i」キーを押して入力モードへ移行
- ESCキーで入力モードから抜ける
- コマンドモードで「:q」と入力するとviが終了する
- コマンドモードで「:wq」と入力すると変更内容を保存して終了

  - 個人的にはemacsのユーザのため、あまり詳しく教えられません・・・
  - 自分にとって使いやすいエディタを使ってください

演習1-6

- cdコマンドを単独で入力しよう（何が起こった？）
- viエディタで.bashrcを編集し、以下の１行を書き込もう

```
alias rm='rm -i'
```

cat

- もともとはファイルを連結して標準出力に出力するコマンド
  - 「連結 concatenate」の「cat」に由来
- テキストファイルの中身を見るために使われる

```
$ cat test.txt
```

  - https://eng-entrance.com/linux_command_cat

cp

- ファイルやディレクトリをコピーする

- これも**alias**しておくほうがよい
  - 先ほどの「**.bashrc**」のなかに以下の行を追記する

```
alias cp='cp -i'
```

  - https://eng-entrance.com/linux_command_cp

mv

- ファイルやディレクトリを移動する or 単に名前を変更する
  - 「動かす move」に由来

- これも**alias**しておくほうがよい
  - 先ほどの「**.bashrc**」のなかに以下の行を追記する

```
alias mv='mv -i'
```

  - https://eng-entrance.com/linux_command_mv

演習1-7

- ホームの下の`SML/01`ディレクトリへ移動しよう
- そこにある`test.txt`をコピーして`test2.txt`というファイルを作ろう
- `test.txt`のファイル名を`test1.txt`に変更しよう
- `ls`コマンドでファイル一覧を確認しよう

Linuxコマンド(3/4)

入出力

echo

- 標準出力に文字列や数値などを表示するコマンド

```
$ echo hello world
$ echo "hello world"
$ echo ¥"hello world¥"
```

- バックスラッシュ「¥」を使ったエスケープ
  - 通常は特殊な解釈をされる文字の直前に「¥」をくっつけると、
  - 特殊な文字としてではなく、通常の文字として解釈されるようになる

標準入力／標準出力／標準エラー出力

- Linuxではこの順に0, 1, 2と識別子が割り振られている
  - 0: 標準入力
  - 1: 標準出力
  - 2: 標準エラー出力
    - 通常、標準入力はキーボードからの入力
    - 通常、標準出力はディスプレイへの出力
    - 通常、標準エラー出力もディスプレイへの出力
    - 標準エラー出力は、システムがユーザに何か異常を伝えるために使われる

リダイレクト(1/2)

- リダイレクトとはコマンドの入力元や出力先をファイルへ変更すること
- 「>」は標準出力のリダイレクト

```
$ echo hello world > test.txt
$ cat test.txt
```

    - 「test.txt」というファイルの中身は全て消えてしまうので、注意。

- 「>>」は出力結果を追加する（上書きではなく）

```
$ echo hello world 2 >> test.txt
$ cat test.txt
```

    - https://eng-entrance.com/linux-redirect

headとtail

- headコマンドはファイルの先頭の数行を出力する
- tailコマンドはファイルの末尾の数行を出力する
- デフォルトではいずれも10行を表示する
- 「-n」オプション：表示する行数を指定する

```
$ man cat > man_of_cat.txt
$ head man_of_cat.txt
$ head -n 5 man_of_cat.txt
$ tail -n 5 man_of_cat.txt
$ tail -n +2 man_of_cat.txt
```

リダイレクト(2/2)

- 「2>」は標準エラー出力のリダイレクト

```
$ ls -lH /etc /aaaaa 2> test.txt
```

- 標準出力と標準エラー出力を破棄する（/dev/nullに出力）

```
$ ls -lH /etc /aaaaa > /dev/null 2>&1
```

- 標準入力のリダイレクト

```
$ ls -lH /etc > test.txt
$ head < test.txt
```

  - headコマンドは、test.txtの内容を、標準入力からの入力として受け取る

演習1-8

- 以下2通りのコマンドの実行結果を確認し、なぜ結果が異なるかを説明しよう

```
$ ls -lH /etc /aaaaa > /dev/null
```

```
$ ls -lH /etc /aaaaa > /dev/null 2>&1
```

less

- 長いテキストファイルを見る
  - catコマンドだとファイルの中身が画面を流れて行ってしまう
  - lessコマンドではショートカットキーでファイルの中を移動できる

- less動作中のショートカット
  - manコマンドと同じです

  - https://news.mynavi.jp/article/20100204-a032/

パイプライン

- パイプラインとはコマンドとコマンドをつないで使う手段
- 「|」という文字（「パイプ」と呼ばれる）で複数のコマンドをつなぐ
  - パイプの左側のコマンドの出力結果を、右側のコマンドが標準入力への入力として受けとる

```
$ ls -lH /etc | less
```

```
$ ls -lH /etc | head
```

  - https://eng-entrance.com/linux-pipeline

tee

- teeコマンドは、標準入力から読み込んだ内容を、ファイルと標準出力の両方へ出力する
  - パイプと組み合わせて使うことが多い

```
ls -lH /etc | tee test.txt | tail
```

演習1-9

- 以下の一連のコマンドを実行し、catコマンドで「-」という記法をパイプと組み合わせて使ったとき、何が起こっているか、説明しよう

```
$ echo hello world > test.txt
$ echo hello world 2 | cat test.txt -
$ echo hello world 2 | cat - test.txt
```

* bashのプロセス置換

- コマンドの実行結果をファイルとして扱えるようにするしくみ
- 「<(コマンド)」と書くと、この部分がコマンドの出力結果を納めたファイルのファイル名として解釈される

```
$ ls -al | head
$ head <(ls -al)
```

- 標準入力のリダイレクトと似ているので混同しないように

```
$ head < Test_File.txt # ファイルの内容を標準入力からの内容として受け取る
$ head <(ls -al) # コマンドの出力結果を納めたファイルのファイル名を指定している
$ head < <(ls -al) # そのファイル名を使ってリダイレクトしている
```

cut

- テキストファイルを横方向に分割する
- 「-d」オプション：どの文字を境に分割するかを指定する
- 「-f」オプション：何番目のフィールドを抜き出すか指定する

```
$ ls -lH /etc | cut -f 1 -d " "
```

  - 「ls -lH /etc」の出力の各行を空白文字で分割し、最初のフィールドだけを抜き出して標準出力に出力する

sort

- ファイルの内容を並べ替える
- デフォルトでは文字列として昇順に並べ替える
- 「-n」オプション：数値として並べ替える
- 「-r」オプション：降順で並べ替える
- 「-t」オプション：各行をどの文字で区切るかを指定する
- 「-k」オプション：何番目のフィールドでソートするかを指定する

```
$ ls -lH /etc | sort -k 1 -r
```

  - 「 `ls -lH /etc` 」の出力を、スペースで区切ったときの最初のフィールドで降順でソート

uniq

- ファイルから重複する行（＝同じ内容をもつ連続する行）を削除する
- デフォルトでは、重複する行を1行にまとめる
- 「-u」オプション：重複する行を表示しない
- 「-c」オプション：重複回数を表示する

```
$ ls -lH /etc | cut -f 1 -d " " | uniq
$ ls -lH /etc | cut -f 1 -d " " | sort | uniq
$ ls -lH /etc | cut -f 1 -d " " | sort | uniq -u
$ ls -lH /etc | cut -f 1 -d " " | sort | uniq -c
```

演習1-10

- 「`ls -lH /etc | tail -n +2`」の出力の第1フィールドを重複なく表示するにはどうすれば良いか

WC

- ファイルの行数、単語数、バイト数を数える
- デフォルトではこの3つの値を出力する
- 「`-l`」オプション：行数だけを表示する
- 「`-w`」オプション：単語数だけを表示する
- 「`-c`」オプション：バイト数だけを表示する

演習1-11

- いま自分のマシンでプロセスを動かしているユーザの数を表示させよう
  - 「ps axu」コマンドの出力の、USERという列の情報を使う

wgetとcurl

- URLを指定してファイルをダウンロードするコマンド
- wgetの場合
  - 「wget URL」：自動的に決められたファイル名でダウンロードした内容が保存される
  - ファイル名や保存場所を変える方法はググって調べる
- curlの場合
  - 「curl URL」：ダウンロードした内容が標準出力へ出力される
  - だからリダイレクトでファイルに保存できる
- 各コマンドのオプションについては各自調べてください

wgetがインストールされていない場合

- Macであれば以下でインストールできるはず

```
$ brew install wget
```

演習1-12

- `wget`または`curl`で、適当なWebページを一つダウンロードする
- `wc`コマンドで、ダウンロードしたHTML形式のファイルが何行あるか調べる

リモートホストに接続して使う

SSH入門

別のマシンにログインしたい…

- 別のマシンにはSSHという方式で接続する
  - 「別のマシン」とか言わずに「リモートホスト」と言うことが多い。
    - telnetはダメ。

- SSHはパスワードや通信内容を暗号化しつつリモートホストと通信できるようにするプロトコル（プロトコル＝通信の方式・約束事）
  - SSHの詳細は知らなくてよい。使い方が分かればよい。

- sshコマンドを使えばSSHプロトコルで接続できる

ssh

- SSHプロトコルでリモートホストに接続するコマンド
- リモートホストでコマンドを実行するためにも使える

```
$ ssh [user@]hostname [command]
```

  - []は省略可の意味。

- リモートホストから戻ってくるには、リモートホスト上で**exit**コマンドを使う

```
$ exit
```

- パスワード認証でもホストにログインできるが、非推奨。公開鍵認証を使う。

sshで公開鍵認証を使う(1/2)

- 自分のPC上で公開鍵と秘密鍵のペアを`ssh-keygen`コマンドで生成する。
  - 秘密鍵は誰にも知られないようにしておくこと！
  - パスフレーズは空でないほうが望ましいですが、空でもいいかもしれません。
    - ただし、空にすると、秘密鍵を盗まれた時点でアウト！

```
$ ssh-keygen -t rsa -b 4096
```

- これで`/home/[ユーザ名]/.ssh` ディレクトリの下に2つのファイルが作られる
  - 秘密鍵：`id_rsa`
  - 公開鍵：`id_rsa.pub`

sshで公開鍵認証を使う(2/2)

- scpコマンドを使って、公開鍵をリモートホストでの自分のhomeディレクトリの下の「.ssh」ディレクトリの下にコピーする

```
$ scp /home/[ユーザ名]/.ssh/id_rsa.pub [リモートユーザ名]@[リモートホスト名]:~/.ssh/my_id_rsa.pub
```

- リモートホストにログインし、ssh-copy-idコマンドで、コピーした公開鍵を登録する
  - 公開鍵が、リモートホストの~/.ssh/authorized_keysに登録される

```
$ ssh-copy-id ~/.ssh/my_id_rsa.pub
```

- これで、以下のようにsshコマンドを使うだけでリモートホストにログインできる

```
$ ssh リモートホスト名
```

演習1-12 （できる人だけ）

- 自宅で、2台以上のマシンを持っている人は、お互いにsshでログインできるようにしてみよう
  - ヒント：DHCPで固定IPが割り振られるように設定しておくとよい

課題1（再掲）

- さっきSMLディレクトリに移ろう
- SMLディレクトリの下に01というディレクトリを作ろう

- 上のことを実行した画面のスクリーンショットを提出

追加の課題（提出の必要なし＆期限なし）

- 以下のことをするにはどうすればいいか、調べよう
  - 今後こういうことをする機会がありそうなので
  - 教えていないコマンドを使う必要があるかもしれないです

1. リモートホストへファイルをコピーするには、どうすればいいか
   - ただし、リモートホストへはSSHで接続できるようになっているとする

2. リモートホストで特定のプロセスを実行し戻ってくるには、どうすればいいか
   - リモートホストからログアウトした後も、そのプロセスはリモートホストで動き続けていなければならない
   - 「`sleep 120`」などで試すと良い