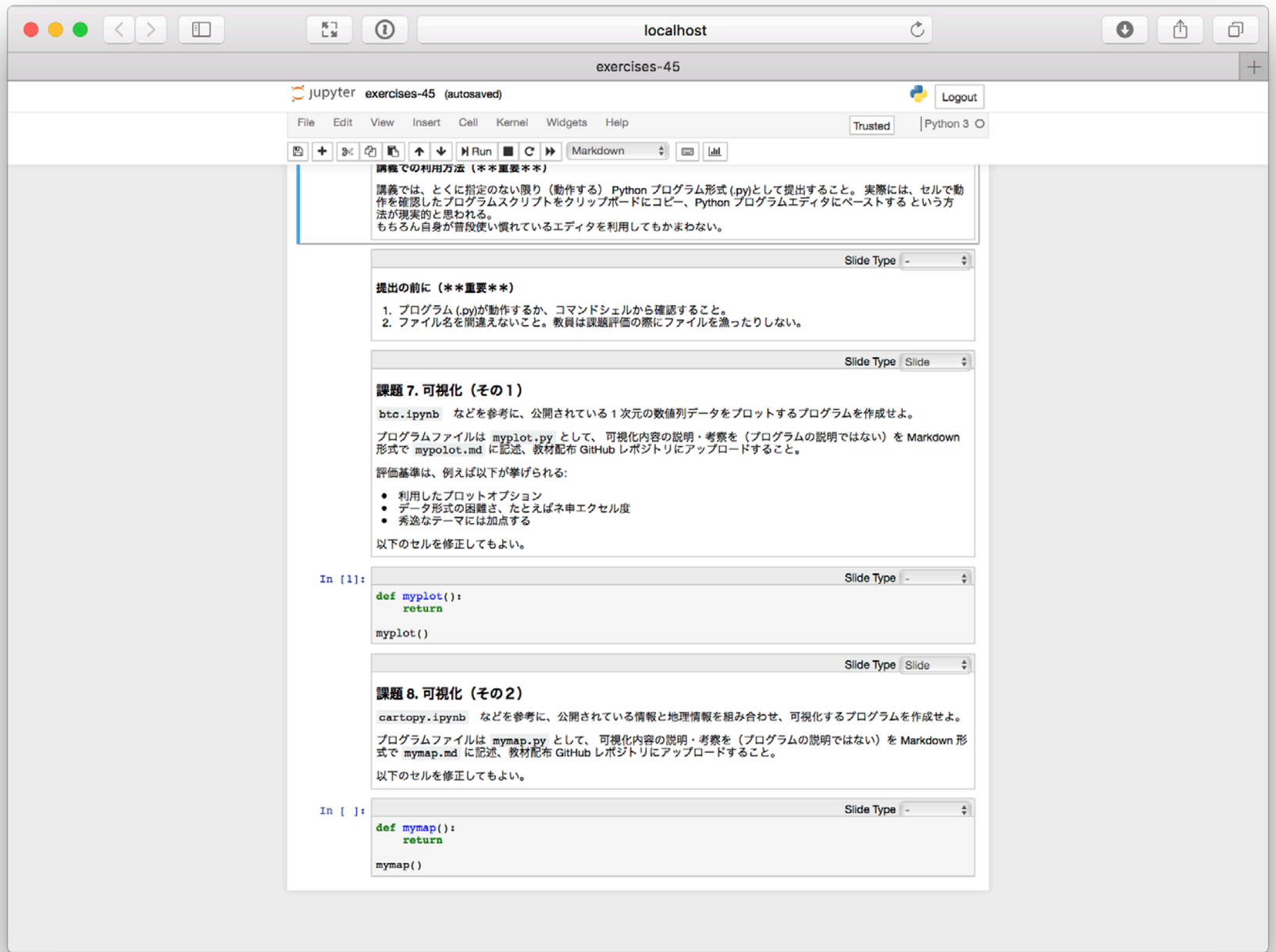
クレジット：
UTokyo Online Education Education コンピュータシステム概論 2018 小林克志

コンピュータシステム概論
第6回

小林克志

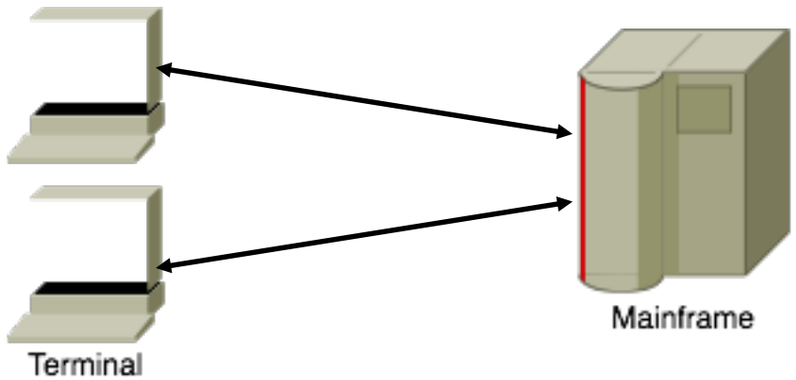
- 事務連絡
- 先週の課題、レビュー（振り返り）
- SSH (Secure SHell)
- SSH による遠隔ログイン(1)パスワード認証
- Linux におけるユーザ、ファイル
- SSH による遠隔ログイン(2)公開鍵認証

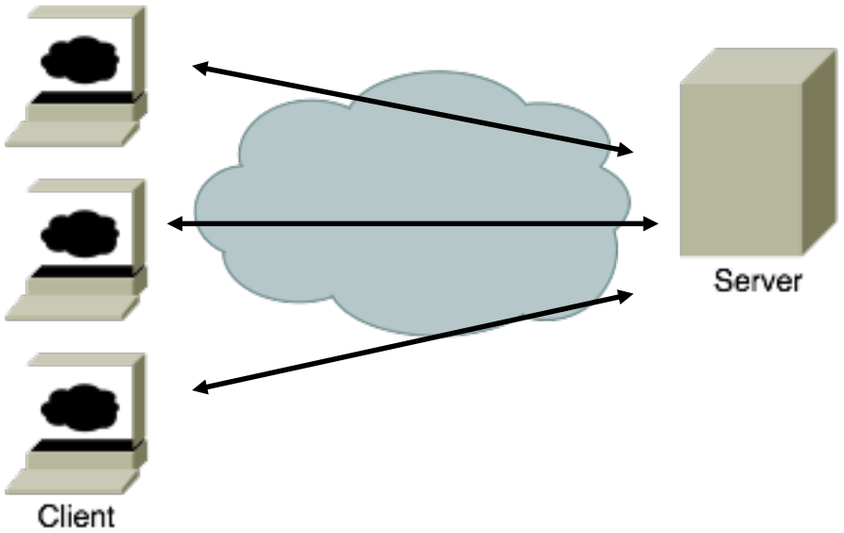
本日の課題：exercises-45.ipynb 読んで指示にしたがってください

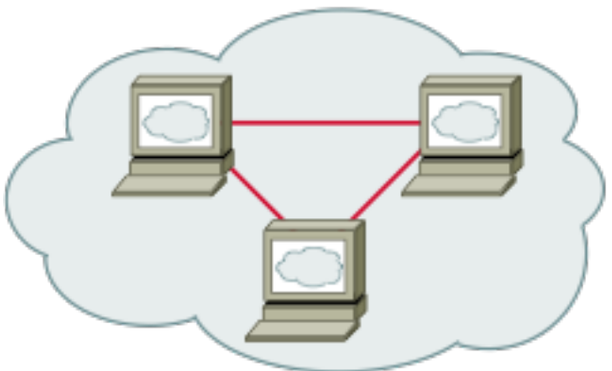


分散アプリケーション（ネットワークサービス）におけるシステム構成

- ホスト・端末：
  - 強力で高価なホストコンピュータを複数の（処理能力に乏しい）端末から同時に利用する
- クライアント・サーバ：
  - 現在のほとんどのネットワークサービス
  - 利用者側端末の処理能力向上を背景に、描画、利用者側入出力などを、クライアントに分担させる。サーバは共有資源を管理する。
  - Web サービスでは：
    - サーバ： Web コンテンツを保存、動的に生成
    - クライアント：ブラウザがコンテンツリクエスト、描画をおこなう
- P2P ：
  - 全てのノードが対等の機能をもつ
  - 制御（品質の維持・法制度への対応を含む）がむつかしい







SSH (Secure SHell) RFC4251 他 (*)

- 安全ではないネットワークを経由して、遠隔ログインや他の安全なネットワークサービスを利用するためのプロトコル（通信規約）。
  - 「安全ではないネットワーク」= 自身のコントロールがおよばない = 盗聴・改ざんされる可能性がある = インターネット
  - 通信内容を暗号化することで内容の秘匿・完全性を確保
  - クライアント・サーバモデル
    - クライアント：ユーザ側の PC
    - サーバ：遠隔コンピュータ（IoT デバイス, Web サーバ, スーパーコンピュータ）
  - OpenBSD SSH (OpenSSH) が広く利用されている（商用を含め他にもある）
    - 複数の認証方式に対応
      - パスワード / 公開鍵 / 外部モジュール…
  - 演習では、Open SSH クライアントを利用し、サーバに遠隔ログイン、CLI (Command Line Interface) による運用・管理操作に利用する

*RFC(Request For Comment) : Internet Engineering Task Force (IETF) が策定するインターネット標準

公開鍵暗号
(Public Key Cryptography)

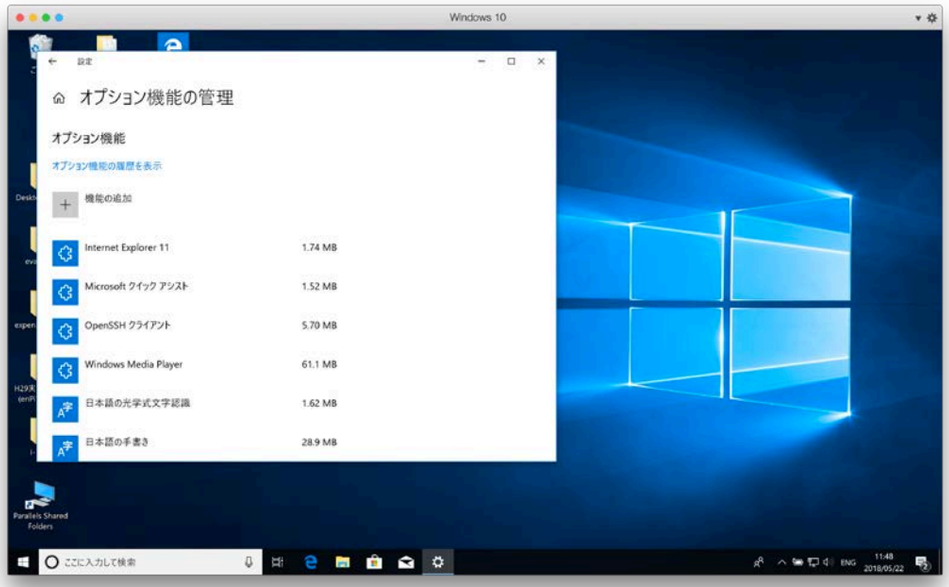
- 送信者が暗号化・受信者が復号化にあたって別の鍵、それぞれ公開鍵・秘密鍵を利用する暗号化方式。以下の性質を利用して送受信者間で安全な通信がおこなえる。
  - 公開鍵は公開情報として配布しておけば暗号化はだれにでもおこなえる、
  - 秘密鍵は公開されていないので、復号化は持ち主にしかできない。
  - 以下の課題もあるが、公開鍵暗号の応用である程度解決できる:
    - だれでも暗号化できるので送信者を特定するには、デジタル署名が必要になる。**これを利用して認証をおこなうこともできる。**
    - 配布されている公開鍵の信頼性は別に担保する必要がある。e.g. PKI(Public Key Infrastructure)
- 暗号・復号化に同じ鍵を利用する共通鍵暗号では、送受信者が安全に鍵を共有することに普及の困難さがあった。公開鍵によってこの問題を回避することができる。

準備：ssh-client を有効にする

- MacOS, Linux では不要、
  Windows 10 も ver.1803 以降では不要

- 設定 -> アプリと機能 -> オプション機能の管理で「OpenSSH クライアント（ベータ）」がインストールされていなければ、機能の追加からインストールする。

- Windows バージョンによっては ssh-keygen に不具合（パスワードを設定できない）がある。代わりに Git に付属しているものが使える：
  C:¥Program Files¥Git¥usr¥bin¥ssh-keygen を使う。



演習：ssh で遠隔ログインしてみる
パスワード認証

1.クライアント（自身の PC）のコマンドプロンプトから、以下のコマンドを実行する：

> ssh demo-pass@demo.shikob.net

- 上の ssh コマンドでは以下を指定している

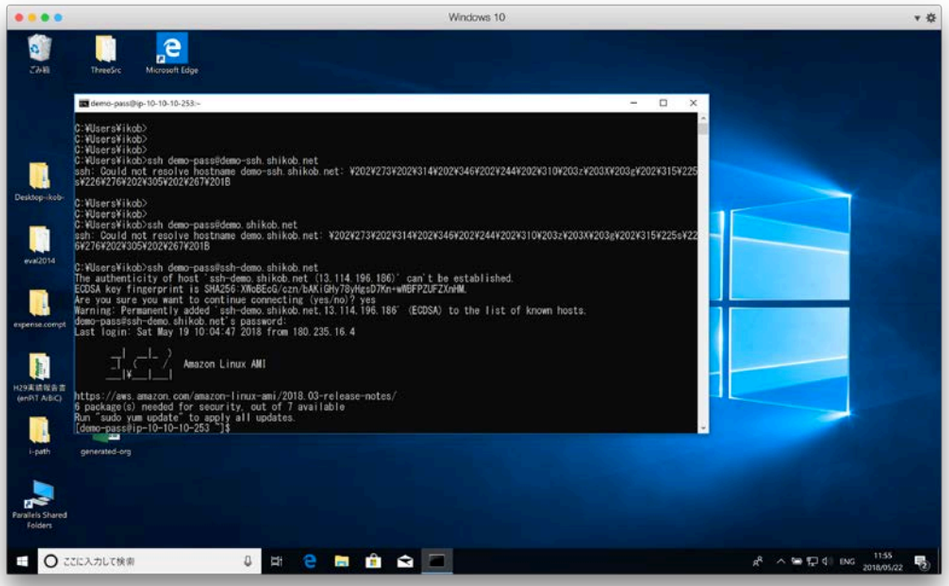
> ssh <遠隔ホストのユーザ名>@<遠隔ホスト名>

2.（初めて接続するため）相手の鍵がないという警告がだされ、sshの接続を続けるかどうか聞かれる。
今回は yes と答えて良い。

3.パスワードを聞かれるので、ユーザ demo-pass のパスワードを入力する（講義中で周知する）

4.コマンドプロンプトが表示されれば、サーバ（遠隔ログイン先のコンピュータ）で以下を実行して、Amazon Linux にログインしていることを確認する：

aws$ uname -a



演習：ssh で遠隔ログインしてみる
– パスワード認証 – (2)

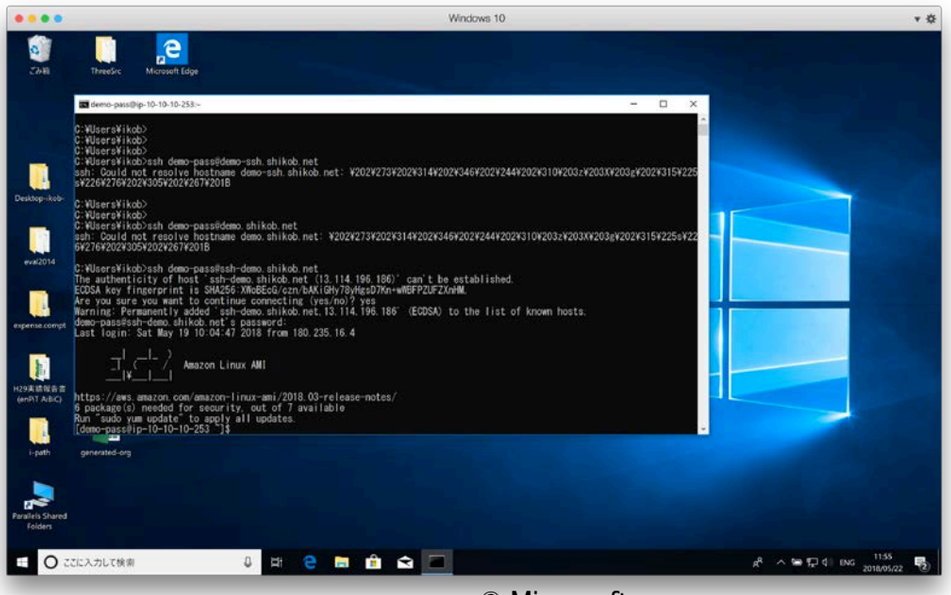
1.以下のコマンドを実行し、/home/treasure/ans.txt の内容を確認する。

$ cat /home/treasure/ans.txt

- 出力された内容を講義 Web より登録する = これを出席とする。

2.以下のコマンドを実行し、ssh セッションを終える

$ exit



チートシート

Unix/Linux Command Cheat Sheet, FOSSwire
https://files.fosswire.com/2007/08/fwunixref.pdf
CC BY-SA

- サーバ操作には、CLI (Command Line Interface)が必須となる。Linux CLIは初心者に敷居が高い。無理に覚えても良いが、はじめのうちはチートシートを参考にした方が良い。
  少し古いが、FOSSwire のものなどが知られている:
  https://fosswire.com/post/2007/08/unixlinux-command-cheat-sheet/
- サーバ上のファイル操作には、vim というツールを利用する。これもチートシートがあった方が良い。

Unix/Linux Command Reference FOSSwire.com

File Commands
- `ls` - directory listing
- `ls -al` - formatted listing with hidden files
- `cd dir` - change directory to *dir*
- `cd` - change to home
- `pwd` - show current directory
- `mkdir dir` - create a directory *dir*
- `rm file` - delete *file*
- `rm -r dir` - delete directory *dir*
- `rm -f file` - force remove *file*
- `rm -rf dir` - force remove directory *dir* *
- `cp file1 file2` - copy *file1* to *file2*
- `cp -r dir1 dir2` - copy *dir1* to *dir2*; create *dir2* if it doesn't exist
- `mv file1 file2` - rename or move *file1* to *file2* if *file2* is an existing directory, moves *file1* into directory *file2*
- `ln -s file link` - create symbolic link *link* to *file*
- `touch file` - create or update *file*
- `cat > file` - places standard input into *file*
- `more file` - output the contents of *file*
- `head file` - output the first 10 lines of *file*
- `tail file` - output the last 10 lines of *file*
- `tail -f file` - output the contents of *file* as it grows, starting with the last 10 lines

Process Management
- `ps` - display your currently active processes
- `top` - display all running processes
- `kill pid` - kill process id *pid*
- `killall proc` - kill all processes named *proc* *
- `bg` - lists stopped or background jobs; resume a stopped job in the background
- `fg` - brings the most recent job to foreground
- `fg n` - brings job *n* to the foreground

File Permissions
`chmod octal file` - change the permissions of *file* to *octal*, which can be found separately for user, group, and world by adding:
- 4 - read (r)
- 2 - write (w)
- 1 - execute (x)

Examples:
- `chmod 777` - read, write, execute for all
- `chmod 755` - rwx for owner, rx for group and world

For more options, see `man chmod`.

SSH
- `ssh user@host` - connect to *host* as *user*
- `ssh -p port user@host` - connect to *host* on port *port* as *user*
- `ssh-copy-id user@host` - add your key to *host* for *user* to enable a keyed or passwordless login

Searching
- `grep pattern files` - search for *pattern* in *files*
- `grep -r pattern dir` - search recursively for *pattern* in *dir*
- `command | grep pattern` - search for *pattern* in the output of *command*
- `locate file` - find all instances of *file*

System Info
- `date` - show the current date and time
- `cal` - show this month's calendar
- `uptime` - show current uptime
- `w` - display who is online
- `whoami` - who you are logged in as
- `finger user` - display information about *user*
- `uname -a` - show kernel information
- `cat /proc/cpuinfo` - cpu information
- `cat /proc/meminfo` - memory information
- `man command` - show the manual for *command*
- `df` - show disk usage
- `du` - show directory space usage
- `free` - show memory and swap usage
- `whereis app` - show possible locations of *app*
- `which app` - show which *app* will be run by default

Compression
- `tar cf file.tar files` - create a tar named *file.tar* containing *files*
- `tar xf file.tar` - extract the files from *file.tar*
- `tar czf file.tar.gz files` - create a tar with Gzip compression
- `tar xzf file.tar.gz` - extract a tar using Gzip
- `tar cjf file.tar.bz2` - create a tar with Bzip2 compression
- `tar xjf file.tar.bz2` - extract a tar using Bzip2
- `gzip file` - compresses *file* and renames it to *file.gz*
- `gzip -d file.gz` - decompresses *file.gz* back to *file*

Network
- `ping host` - ping *host* and output results
- `whois domain` - get whois information for *domain*
- `dig domain` - get DNS information for *domain*
- `dig -x host` - reverse lookup *host*
- `wget file` - download *file*
- `wget -c file` - continue a stopped download

Installation
Install from source:
```
./configure
make
make install
```
- `dpkg -i pkg.deb` - install a package (Debian)
- `rpm -Uvh pkg.rpm` - install a package (RPM)

Shortcuts
- `Ctrl+C` - halts the current command
- `Ctrl+Z` - stops the current command, resume with `fg` in the foreground or `bg` in the background
- `Ctrl+D` - log out of current session, similar to `exit`
- `Ctrl+W` - erases one word in the current line
- `Ctrl+U` - erases the whole line
- `Ctrl+R` - type to bring up a recent command
- `!!` - repeats the last command
- `exit` - log out of current session

* use with extreme caution.

Linux のユーザ・グループ

- ユーザの種類と識別子：username に加え uid (32bit 整数)で識別する。
  - 特権ユーザ(root)と一般ユーザ
    - uid = 0 は特権ユーザ、それ以外は一般ユーザ
      - root でログインすることは現在では推奨されていない。
        特権が必要な操作をおこなったユーザをたどることができなくなるため
        代わりに sudo コマンドを使う。
    - **ネットワークサービスに必要な設定の多くは特権が必要となる。**
- グループの種類と識別子groupname に加え gid でグループを識別する。
  - グループには複数のユーザが所属できるし、ユーザは複数のグループに所属できる。
  - 習慣的に、グループ名 wheel が管理ユーザのグループ名として使われてきた

Linux のディレクトリ構成

- / 区切りのディレクトリ名で階層化されている、用途によって使い分けられている
  - / : root
    - /boot : OS プログラム（カーネル）が配置
    - /bin, /sbin : （システム上）重要なユーティリティプログラム
    - /etc: システム上欠かせないファイル、システム構成情報
    - /dev : デバイスノード
    - /tmp : 一時ファイル
    - /lib, /lib64 : 共有ライブラリなど
    - /usr : ほとんどのユーティリティ（開発環境を含む）はここに置かれる
    - /var : ログなど頻繁に書き換えられるファイル
    - /home : ユーザのホームディレクトリ

Linux ファイルタイプ

- 以下のようなものがあり、4bits の情報として管理されている。
  - 通常ファイル：プログラム・データ・ソースコードを含む通常のファイル
  - ディレクトリ：ディレクトリもファイルとしてあつかわれる
  - シンボリックリンク：他のファイルへのリンク
- ls -ld を利用すればファイルタイプは先頭文字で示される。ちなみに、3, 4 番目のフィールドには所有者のユーザ名、グループ名が表示される：

```
$ ls -ld /etc/passwd
-rw-r--r-- 1 root root 1304 May 19 09:40 /etc/passwd
$ ls -ld /usr/include/
drwxr-xr-x 3 root root 4096 May  8 17:48 /usr/include/
$ ls -ld /dev/tty0
crw--w---- 1 root tty 4, 0 May 16 02:29 /dev/tty0
$ ls -ld /usr/tmp
lrwxrwxrwx 1 root root 10 May  8 17:48 /usr/tmp -> ../var/tmp
```

通常ファイル

ディレクトリ

デバイスノード

シンボリックリンク

Linux ファイル属性
所有者と権限

- 全てのファイルは 12bits “mode” bits を持つ。
- アクセス許可情報：下位の 9bits
  - 所有者(u), グループ(g), その他のユーザ(o)へそれぞれ 3bit ずつ
  - 3 bits の内訳は、先頭から読み出し(r)、書き込み(w)、実行(x) 。
- 例えば、ls -l コマンドではすべての bit が 1 の場合、rwxrwxrwx のように表示される。0 の場合は文字の代わりに - が、例えば　rwxr-x---
- アクセス許可情報は chmod コマンドで設定することができる。
  - 9bits すべてを指定するときは、8進数表記で指定する：

```
$ ls -l test
-rw-rw-r-- 1 ec2-user ec2-user 0 May 23 02:32 test
[ec2-user@ip-10-10-10-253 ~]$ chmod 666 test
[ec2-user@ip-10-10-10-253 ~]$ ls -l test
-rw-rw-rw- 1 ec2-user ec2-user 0 May 23 02:32 test
$
```


Linux ファイル属性
所有者と権限

- setuid, setgid bits : 12, 11 bit
  - 実行ファイルに設定されていた場合、実行プログラムはファイルの所有者・所有グループとして動作する
  - ディレクトリに setgid が設定されていればディレクトリに作成されるファイルのグループは default でディレクトリの所有者のそれとなる。
- sticky bit : 10 bit
  - ディレクトリに設定されていれば、ディレクトリが書き込み可能であっても、オーナー以外がファイルの削除、ファイル名の変更をおこなえない。

演習：ssh で遠隔ログインしてみる
– 公開鍵認証 –（1）

1.ssh-keygen をクライアントで実行し鍵ペア（公開鍵 / 秘密鍵）を生成する（Windows で有効化した SSH クライアントがベータの場合は、C:¥Program Files¥Git¥usr¥bin¥ssh-keygen で代える）:

```
> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:¥Users¥ikob/.ssh/id_rsa):
```

- 秘密鍵の保存場所は default でかまわない。

2.その後、秘密鍵を安全に管理するためのパスフレーズ（パスワード）を聞かれるので、適当に設定する（後で使います）:

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:¥Users¥ikob/.ssh/id_rsa.
Your public key has been saved in C:¥Users¥ikob/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:DbYwiySwTqAhDuCF4kBCWMhx3NqlNxpNRzSbojKktos ikob@KATSUSHIKOBDC14
The key's randomart image is:
+---[RSA 2048]----+
|#*=o.   o+       |
|&*o. . o .+      |
|*+..+ B.+o       |
|o.o+ =.O.+       |
| + o..= S .      |
|. . o.           |
| .               |
|. .              |
|E.               |
+----[SHA256]-----+
```

演習：ssh で遠隔ログインしてみる
– 公開鍵認証 – (2)

1.生成された公開鍵、.ssh¥id_rsa.pub をサーバに適切に置く必要がある。生成された公開鍵を scp (ssh を利用したファイル転送) によってサーバに転送する。(サーバに適切に置く作業は、教員がおこなう。)

> scp .ssh¥id_rsa.pub demo-pass@ssh-demo.shikob.net:pubkeys/学生証番号
demo-pass@ssh-demo.shikob.net's password:

- パスワードはパスワード認証で使ったものを入力する。

2.教員が公開鍵を適切に配置したのち、以下のコマンドで公開鍵認証でログインする。

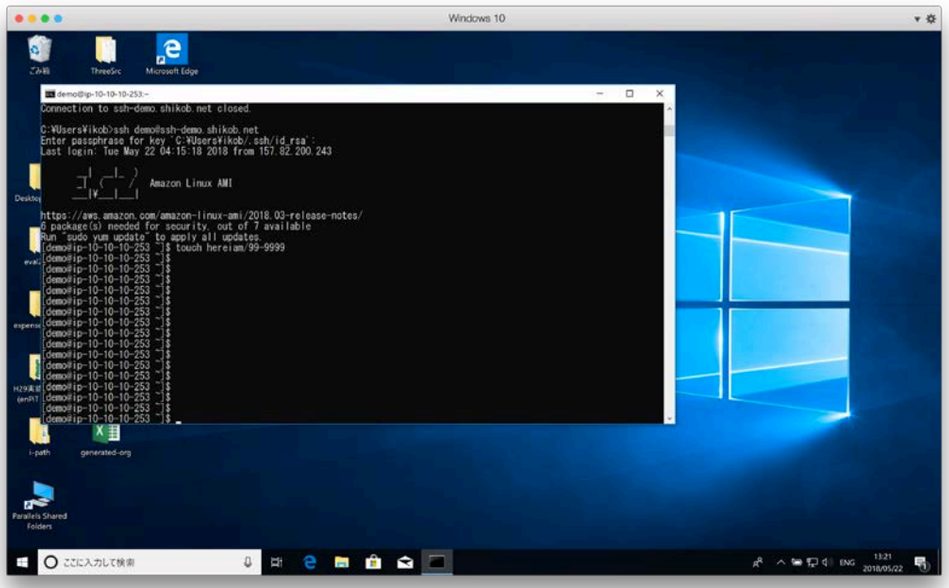
> ssh demo@ssh-demo.shikob.net
Enter passphrase for key 'C:¥Users¥ikob/.ssh/id_rsa':

- パスワードは鍵ペアを作成したときに設定したものを使う

3.以下のコマンドを実行し、爪痕を残す：

aws$ touch hereiam/学生証番号

4. 3. で作ったファイルのアクセス許可情報を以下のように設定する：
所有者：読み、書き、実行可能
グループ：読み、実行可能
その他：実行可能



- scp = C:¥Program Files¥Git¥usr¥bin¥scp
- ssh = C:¥Program Files¥Git¥usr¥bin¥ssh