Command line Environment

[Job Control]

1. Killing a process

- Shell is using a UNIX communication mechanism called a signal to communicate with the process. When a process receives a signal it stops the execution, deals with the signal. So signals are also called software interrupts.
- 3 ways for killing a process (asking a process to exit gracefully):
  Ctrl-C->SIGINT; Ctrl-\->SIGQUIT; kill -TERM <PID> ->SIGTERM

2. Pausing and backgrounding processes

- We can pause the process by Ctrl-Z->SIGSTP/SIGSTOP
- We can then continue the paused job by fg (in the foreground) or bg (in the background).
- the jobs command lists the unfinished jobs associated with the current terminal session
- refer to one job/process using its PID (use pgrep to find that out) or its percent symbol followed by its job number (use jobs to find that out); refer to the last backgrounded job using $!

3. Exercises

- Below are required commands:
  start a process: sleep 10000 (sleep for 10000 ms, or 10s)
  pause the process: Ctrl-Z
  continue the paused process in the background: bg
  get the pid using pgrep: pgrep -af "sleep 10000" (-a: show the pid; -f: show command
name)
  kill the process using pkill: pkill -f "sleep 10000" (-f: show command name)

[Terminal Multiplexers]

- Terminal multiplexers like tmux (the most popular terminal multiplexer these days) allow users to multiplex terminal windows using panes and tabs so they can interact with multiple shell sessions. Also, it lets users detach a current terminal session and reattach at some point later in time.
- the usage of tmux: Ctrl-b+x (Ctrl-b, release, x)
- tmux has the following hierachy of objects:

1. Sessions: a session is an independent workspace with one or more windows
   tmux: starts a new session
   tmux new -s NAME: starts a new session with the given name
   tmux ls: lists the current sessions
   Within tmux typing Ctrl-b+d: detachs the current session
   tmux a: attaches the last session (you can use -t to specify which)
   tmux rename-session -t [session number/name] [newname]: rename a created session
   tmux kill-session -t [session number/name]: kill a session

2. Windows: equivalent to tabs in editors or browsers, they are visually separate parts of the same session

Ctrl-b+c: creates a new window. To close it, you can just terminate the shell by Ctrl-d
Ctrl-b+N: go to the Nth window. Note that they are numbered
Ctrl-b+p: go to the previous window
Ctrl-b+n: go to the next window
Ctrl-b+,: rename the current window
Ctrl-b+w: list the current windows

3. Panes: like vim splits, panes let you have multiple shells in the same visual display
Ctrl-b+": split the current pane horizontally
Ctrl-b+%: split the current pane vertically
Ctrl-b+direction: move the pane in the specified direction. Direction here means arrow keys
Ctrl-b+z: toggle zoom for the current pane
Ctrl-b+[: start scroll back. You can then press <space> to start a selection and <enter> to copy that selection
Ctrl-b+<space>: cycle you through pane arrangements

[Aliases]

- A shell alias is a short form for another command that your shell will replace automatically for you. Below is an example:
```
alias alias_name="command_to_alias arg1 arg2"
```
(note that there is no space between the equal sign, because alias only takes one argument)
- You can ignore an alias by prepending it with \, like "\ls" (then the alias ls will not be used)
- You can disable an alias like "unalias ls"
- You can get an alias definition. Below is an example:
```
#make shorthands for common flags
alias ll="ls -lh"
#get its definition
alias ll
```
- Note that aliases do not persist shell sessions by default.
- Exercises:

1. Create an alias dc that resolves to cd for when you type it wrongly.
```
alias dc="cd"
```

[Dotfiles]

- Many programs are configured using plain-text files known as dotfiles (because the file name begins with a dot, e.g. ~/.vimrc, so that they are hidden in the directory listing ls by default)
- Shells are one example of programs configured with dotfiles.
- Some examples of tool that can be configured through dotfiles are:
bash: ~/.bashrc, ~/.bash_profile
git: ~/.gitconfig
vim: ~/.vimrc and the ~/.vim folder

```
ssh: ~/.ssh/config
tmux: ~/.tmux.conf
```

[Remote machines]

- SSH means a secure shell.
- to ssh into a server: ssh foo@bar.mit.edu
  foo: username
  server: bar.mit.edu (the server can be specified with an URL or an IP)

1. Executing commands
2. SSH keys

- Key-based authentication exploits public-key cryptography to prove to the server that the client owns the secret private key without revealing the key. This way you do not need to reenter your password every time. Nevertheless, the private key (often ~/.ssh/id_rsa) is effectively your password.
- You can generate a pair by running:

```
ssh-keygen -t rsa -C "Your_email_address"
cat ~/.ssh/id_rsa.pub
```

3. Copying files over SSH

- ssh+tee:

```
cat local_file | ssh remote_server tee server_file
```

- scp:

```
scp path/to/local_file remote_host:path/to/remote_file
```

- rsync: it improves upon scp by detecting identical files in local and remote, and preventing copying them again. It also provides more fine-grained control over symlinks, permissions and has extra features. Its syntax is similar to scp.

4. SSH configuration