SKILLPILLS

Mini Course: terminal

Part I: Introduction

Juan Polo

Outlook

SKILLPILLS

1. Terminal
   - Introduction
   - Basic commands
   - Advance commands
2. Vim
   - Jeremie Gillet
3. HPC (high-performance computing)
   - Charles Plessy

For this lecture

Yellow means **you should try to run** the commands
Gray is an example that **you do not need to type** in

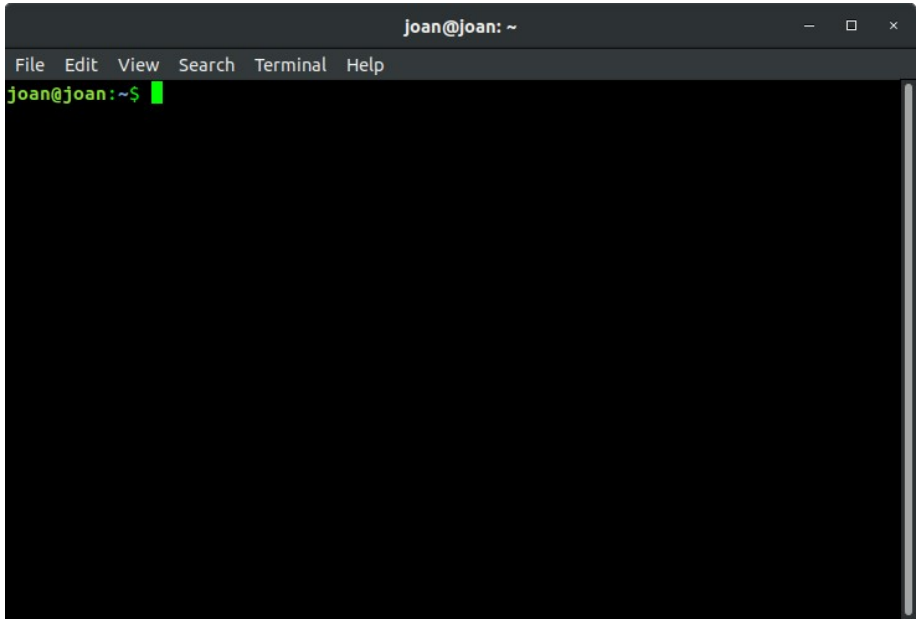
```
ssh -X your-user@deigo.oist.jp
```

```
ssh -X your-user@deigo.oist.jp
```

Why is the terminal important?

SKILLPILLS

- A **terminal** is a **text-only window in a graphical user interface** (GUI) that emulates a console, which in turn is a display mode that contains only text and no images and that occupies the entire screen of the display device.
- Whenever you open a terminal it automatically uses a specific "**shell**". This shell is the actual program that **interprets the commands you input**. In Linux-based systems the shell is usually bash, but others exist cshell (csh), zshell (zsh), korn shell (ksh).



Gnome-terminal in Ubuntu 18

Why is it important and what can you do?

SKILLPILLS

- **Why is it important?**
  - It allows you to access the true power of your computer
  - It can be much faster to complete some tasks using a Terminal than with graphical applications and menus.
- **What will we do?**
  - Move, copy, delete, modify your files and folders
  - Read/Create files
  - Change permission of a file
  - Look for things inside a file
- **What can you do?**
  - Create videos from images using ffmpg
  - Change names of (many) multiple files fast
  - Etc.

How do we do those things?

SKILLPILLS

- **Commands**
  - A command is an instruction given by a user telling a computer to do something, such a run a single program or a group of linked programs.

```
COMMAND space OPTIONS space ARGUMENTS
```

COMMAND: are a type of instructions

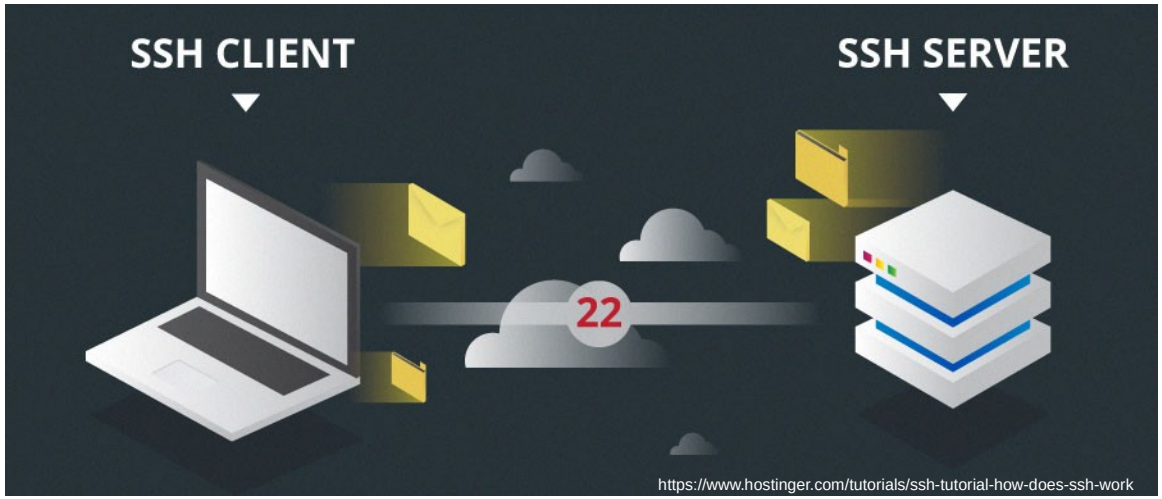
OPTIONS: modifications or extras applied to the basic command

ARGUMENTS: are the “objects” of the command, typically folders and files

Secure Shell connection

SKILLPILLS

- **SSH**
  - A command that creates a secure connection with a remote computer.




```
ssh -X user@server
```

SSH command will open a remote terminal allowing you to run commands in a another computer.

The commands you will write after the ssh connection is established are actually running in **another computer!!!**

Let’s enter the terminal

SKILLPILLS

We all have to enter in the terminal now

Linux

```
ssh -X your-user@deigo.oist.jp
```

Mac

```
ssh -X your-user@deigo.oist.jp
```

Windows (MobaXterm)

```
ssh -X your-user@deigo.oist.jp
```

```
ssh -X your-user@deigo.oist.jp
```

COMMAND: connection to remote server

OPTIONS: allows opening images

ARGUMENTS: the specific server you want to connect

Moving around

SKILLPILLS

Important shortcuts

- Cancel/Stop **Ctrl + c**
- Auto-completion **Tab**
- See previous command **Up/down arrow**

Auto-complete

```
Tab
```

Cancel/Stop

```
Ctrl + c
```

Where am I?

- When connecting to the cluster you will be redirected to **your “home” folder**

Present Working Directory

```
pwd
```

/home/j/juan-polo

Symbol	Name
~	HOME (personal folder)
/	ROOT (first location of the computer)
.	Current directory
..	Parent directory (previous folder)

SKILLPILLS

Typical directory tree

```
/apps/unit/GradschoolD/terminal
└── material
    ├── cat_test
    ├── color_test
    ├── grep_test
    ├── perm_test
    └── remove_test
```

```
/apps/unit/GradschoolD/terminal
└── material
    ├── cat_test
    │   ├── CATchMEifYOUcan.txt
    │   ├── my_cat_mess.txt
    │   └── very_long_table.tsv
    ├── color_test
    ├── grep_test
    ├── perm_test
    └── remove_test
        ├── empty_file1
        ├── empty_file2
        ├── empty_file3
        ├── hello.txt
        ├── hi.txt
        ├── non_empty_file
        ├── non_empty_file2
        ├── non_empty_file3
        ├── non_empty_file4
        └── removeME.txt
```

Deigo root directory

```
/
├── apps
├── bin -> usr/bin
├── boot
├── bucket
├── dev
├── etc
├── flash
├── home
├── hpacquire
├── hpcshare
├── lib -> usr/lib
├── lib64 -> usr/lib64
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── saion_work
├── sango_apps -> /sango_work/.apps
├── sango_work
├── sbin -> usr/sbin
├── scratch
├── srv
├── sys
├── tmp
├── usr
└── var
```

Moving around

SKILLPILLS

Change directory

- It requires an argument
  - Where do you want to go?

```
cd
```

Auto-complete

```
Tab
```

Cancel/Stop

```
Ctrl + c
```

Types of path

- **Absolute**
  - Starts from ROOT
  - Example:

```
cd /home/j/juan-polo
```

Symbol	Name
~	HOME
/	ROOT
.	Current directory
..	Parent directory

- **Relative**
  - Starts from Current directory
  - Example:

```
cd my_folder/my_second_folder
```

(./) might be used to state that we want to start from current directory

Moving around

SKILLPILLS

Change directory

- It requires an argument
  - Where do you want to go?

```
cd
```

Auto-complete

```
Tab
```

Cancel/Stop

```
Ctrl + c
```

```
pwd
cd /
pwd
```

```
pwd
cd ~
pwd
```

```
pwd
cd ..
pwd
```

Symbol	Name
~	HOME
/	ROOT
.	Current directory
..	Parent directory

```
cd
```

SKILLPILLS

Auto-complete

`Tab`

Cancel/Stop

`Ctrl + c`

Go to skill pill directory

- It can be found in /apps/unit/GradschoolD/terminal

- **Hints**
  - Start from ROOT
  - Use “Tab” for auto-completion
  - Check your current path “pwd”

Moving around

SKILLPILLS

Auto-complete

```
Tab
```

Cancel/Stop

```
Ctrl + c
```

Go to skill pill directory

- It can be found in /apps/unit/GradschoolD/terminal

- **Hints**
  - Start from ROOT
  - Use “Tab” for auto-completion
  - Check your current path “**pwd**” at every step

```
cd /apps/unit/GradschoolD/terminal

cd material

cd ..
```

Go back to your home directory

```
cd /home/j/juan-polo/
```

or just

```
cd
```

SKILLPILLS

Listing files

- Shows the contents of the current directory
  - It does not require arguments
  - It can show a lot of information

```
ls
```

ls basic options

-l
    use a long listing format
-h
    human-readable
-a
    all (including hidden files)
-t
    sort by modification time
-S
    sort by size

ls basic arguments

(typically used with options)

file_name
    information about that file/folder
file_nam*
    information about files/folders
    starting by XXX*

*** is somewhat general and can be used to find patterns**

SKILLPILLS

Listing files

- Shows the contents of the current directory
  - It does not require arguments
  - It can show a lot of information

```
ls
```

Listing files in minicourse folder

```
cd /apps/unit/GradschoolD/terminal/material/cat_test

ls

ls -lah
```

size

Permissions
ownership

```
[juan-polo@deigo-login3 Intro]$ ls -lah
total 3.9M
drwxr-xr-x 4 jan-moren scicomsec  150 Oct  2 13:29 .
drwxr-xr-x 3 jan-moren scicomsec   42 Oct  2 13:29 ..
drwxr-xr-x 2 jan-moren scicomsec  490 Oct  2 13:29 code
-rw-r--r-- 1 jan-moren scicomsec  46K Oct  2 13:46 Connect_and_command_line.pdf
-rw-r--r-- 1 jan-moren scicomsec 2.1M Oct  2 13:46 Intro_01.pdf
-rw-r--r-- 1 jan-moren scicomsec 822K Oct  2 13:46 Intro_02.pdf
drwxr-xr-x 2 jan-moren scicomsec   29 Oct  2 13:29 text
```

folders

files

Permissions and Ownership

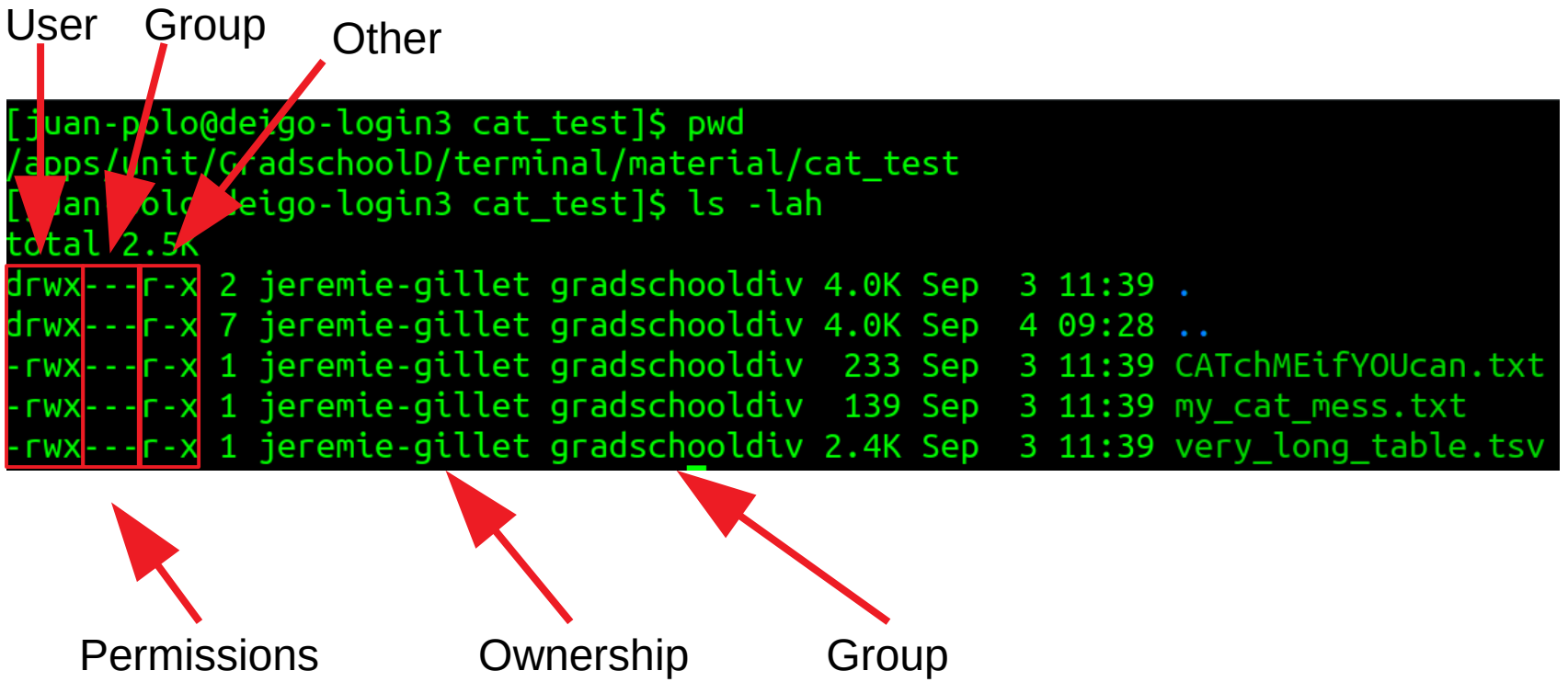
SKILLPILLS

Permission

- Reading **(r)** [4]
- Writing **(w)** [2]
- Executing **(x)** [1]

Ownership

- Current user
- User's group
- Others



Permissions and Ownership

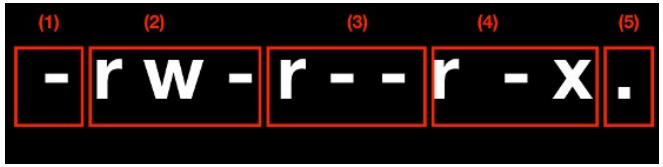
SKILLPILLS

Permission

- Reading **(r)** [4]
- Writing **(w)** [2]
- Executing **(x)** [1]

Ownership

- Current user
- User's group
- Others



https://opensource.com/article/19/6/understanding-linux-permissions
Bryant Jimin Son

Permissions and Ownership

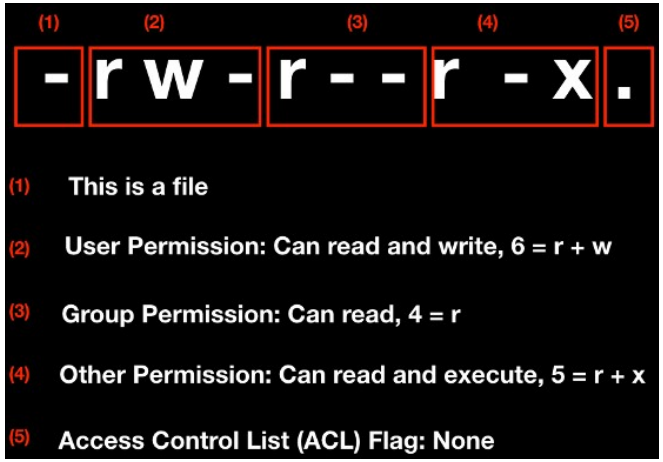
SKILLPILLS

Permission

- Reading **(r)** [4]
- Writing **(w)** [2]
- Executing **(x)** [1]

Ownership

- Current user
- User's group
- Others



https://opensource.com/article/19/6/understanding-linux-permissions
Bryant Jimin Son

Permissions and Ownership

SKILLPILLS

chmod

- change file mode bits
  - Change permissions of folders and files

chown

- change file owner and group

```
chmod 600 test.txt

chmod u=rw,g=-,o=- test.txt
```

```
-rw-------  1 juan-polo buschuni  5 Nov  7 13:08 test.txt
```

Octal Value	Read	Write	Execute
7	r	w	x
6	r	w	-
5	r	-	x
4	r	-	-
3	-	w	x
2	-	w	-
1	-	-	x
0	-	-	-

Creating a folder

SKILLPILLS

Make a directory

- Creates a directory (folder)
  - It requires an argument
  - No spaces is a good practice

```
mkdir ARG1
```

Go back to your home

```
cd
```

Relative path

```
mkdir newfolder
```

Absolute path

```
mkdir /home/j/juan-polo/newfolder2
```

Copying things

SKILLPILLS

Copy

- Copies files and directories from source to destination
  - It requires two arguments

```
cp ARG1 ARG2
```

```
cp SOURCE DESTINATION
```

cp important options

-r

recursive copy (inclduing folders)

Copy exercises to your home

```
/apps/unit/GradschoolD/terminal
```

Copying things

SKILLPILLS

Copy

- Copies files and directories from source to destination
  - It requires two arguments

```
cp ARG1 ARG2
```

```
cp SOURCE DESTINATION
```

cp important options

-r

recursive copy (including folders)

Copy exercises to your home

```
cp -r /apps/unit/GradschoolD/terminal/ /home/j/juan-polo/newfolder2/
```

SKILLPILLS

Move

- Moves files and directories from source to destination
  - It requires two arguments

```
mv ARG1 ARG2
```

```
mv SOURCE DESTINATION
```

SKILLPILLS

Move

- Moves files and directories from source to destination
  - It requires two arguments

```
mv ARG1 ARG2
```

```
mv SOURCE DESTINATION
```

Move exercises to newfolder

```
mv -r /home/j/juan-polo/newfolder2/terminal/ /home/j/juan-polo/newfolder/
```

SKILLPILLS

rm

- Remove files and directories
  - It requires one argument, but it can take multiple

```
rm ARG1
```

It is probably the most dangerous command in terminal.

Careful when using the recursive option “-r” or/and “*” to remove multiple files

Removing things

SKILLPILLS

rm

- Remove files and directories
  - It requires one argument, but it can take multiple

```
rm ARG1
```

rm important options

-r

recursive copy (including folders)

Remove a file from exercise folder

1) We first change directory to the folder remove_test
2) Check the files inside using **ls**
3) remove the file

```
cd  /home/j/juan-polo/newfolder/terminal/material/remove_test

rm empty_file1
```

BREAK

SKILLPILLS

cat

- See file content
  - It requires one argument, but it can take multiple

```
cat ARG1
```

See the contents of a file from the exercise folder

1) We first change directory to the main exercise folder
2) Check the files inside using ls
3) Use “cat” to see the file content (remember tab to autocomplete)

```
cat CATchMEifYOUcan.txt
```

```
cat catmultiple1.txt
```

```
cat catmultiple2.txt
```

```
cat catmultiple*
```

Sorting ordering

SKILLPILLS

sort

- Sorts the content of a file
  - It requires an argument

```
sort ARG1
```

sort important options

-k

sort via a key; typically a column number #

-g

general-numeric-sort. Compare according to general numerical value

Sorting lines of a file

1) Go the the main folder of the exercises
2) Check the files inside using **ls**
3) Use **cat** to see the file content (use **tab** to autocomplete)
4) **sort** the file following column 1 and then 2

```
sort -g -k1 unsorted.txt

sort -g -k1,1 unsorted2.txt

sort -g -k1,1 -k2,2 unsorted2.txt
```

`-k2,2` means use the fields from #2 to #2

SKILLPILLS

Pipe

- Sends the output of one command to another
  - It requires two commands on each side of the symbol “|”

```
|
```

Count lines of a file

1) Go the the main folder of the exercises
2) Check the files inside using **ls**
3) Use **cat** to see the file content (remember tab to autocomplete)
4) Pipe the output to count the line for you
5) We will use the command word count (**wc**) and its option lines (**-l**)

```
cat catmultiple1.txt | wc -l
```

```
cat catmultiple* | wc -l
```

SKILLPILLS

Redirection

- Redirection of the output given by a command
  - “>” Writes output
  - “>>” Appends output
  - It requires a command before and an argument after “>” that says where we want to store the output

```
>
```

```
>>
```

Save results in a file and sort

1) Go the the main folder of the exercises
2) Sort the file following column one
3) Redirect the output to “sorted.txt”

⚠ WARNING ⚠

> will **OVERWRITE**

Permanently, there is NO UNDO

```
cat catmultiple*  > unsortedcat.txt
```

```
sort -g -k1 unsortedcat.txt  > sortedcat.txt
```

```
cat sortedcat.txt
```

SKILLPILLS

grep

- Search within a file
  - It requires an argument

```
grep ARG1
```

grep basic options

-#
print # line before and after the match
-n
line number

Search for a pattern in a file

1) Go the the main folder of the exercises
2) Search for ‘0.0’ in the file

```
grep –n '9' unsorted.txt

grep –n –2 '9' unsorted.txt
```

Other useful commands

SKILLPILLS

head

- See the beginning of a file
  - It requires an argument
  - “-n 10” shows the first 10 lines

```
head ARG1
```

```
head unsorted2.txt
```

tail

- See the end of a file
  - It requires an argument
  - “-n 10” shows the last 10 lines

```
tail ARG1
```

```
tail unsorted2.txt
```

find

- Find files
  - It requires an argument

```
find ./ -name 'cat*'
```

```
find ./ -type f
```

touch

- Create empty file

```
touch  AGRG1
```

Other tools to use in combination

SKILLPILLS

awk

- Tool for processing text file rows and columns

```
awk '{print}' ARG1
```

du -sh

- Size of current folder and subfolders

```
du -sh

du -sh *

du -shc * | sort -h
```

df -h

nohup

- Command that will make another command ignore a hangups or disconnecting.
  - Output will go to “nohup.out”
  - **&** means in general: running in the background

```
nohup COMMAND &
```

SKILLPILLS

man

```
man ls
```

- You can exit by pressing “q”
- You can move with the “arrows” or “Page up/down”
- You can search by “/search_pattern” and find next/previous match with “**n**” / “**N**”

Other tools, commands, etc.

SKILLPILLS

Ctrl+r

- Shortcut to search withing your hystory, it searches the pattern you input. To leave press **Ctrl+c**

Bash is a programming language!!!

- **echo** is your print
- for loops
- Variables
- Scripts starting with a file with
  #!/bin/bash
  Then running **bash file_name.sh**

```
echo "hello"
for i in {1..5}
 do
echo "$i"
 done
```

Sudo

- super user do: run programs with the security privileges
- Sometimes **sudo su**