SKILLPILLS

Skill Pill: Terminal

Lecture 1: the basics

James Schloss

Okinawa Institute of Science and Technology
james.schloss@oist.jp

February 13, 2016

OIST

Overview

1. Navigation
2. File manipulation
3. Permissions
4. Piping
5. System Info
6. Stuff that's good to know...

NAVIGATION

Imagine that you are in a file browser. You have folders (directories) and stuff in the folders.

- **cd** – Moves you to where you want to go

```
cd folder
```

- **pwd** – Tells you where you are

```
pwd
/an/example/folder
```

NAVIGATION

- **ls** – Lists your stuff: -a → hidden files; -l → permissions

```
ls
stuff_in_folder.dat
ls -a
stuff_in_folder.dat   .hidden_stuff_in_folder.dat
ls -l
-rw-r--r-- 1 me me 26978 Feb 14:41 file.dat
```

- **history** – Lists previously run commands

NAVIGATION

EXERCISE

1. Open a terminal
2. Try out all the commands in the navigation block.
3. Move around in your filesystem. Try

```
cd /
```

and then try to navigate back to your home directory.

FILE MANIPULATION

Everything short of text editing

- **mkdir** – Makes a directory
- **rmdir** – Removes a directory

```
rmdir/mkdir folder
```

- **touch** – Initializes file

```
touch file
```

- **rm** – Removes a file

```
rm file
rm -r directory
```

- **cp** – Copies a file

FILE MANIPULATION

- **mv** – Moves a file

```
mv old_file_location new_file_location
```

- **cat** – Prints text from file

```
cat file
```

- **head** –Prints top of file
- **tail** – Prints bottom of file

```
head/tail -n 5 file
```

FILE MANIPULATION

EXERCISE

1. Try out all the commands in the file manipulation block.
2. Move around in your filesystem. Try creating a directory and touching a bunch of files.
3. move and copy the files around, for fun.
4. Remove the files and the directory
5. Find a file with stuff on it. Print out the text.
6. With that same file, print out the top 10 lines, and then the bottom 15

PERMISSIONS

You are not root! You can only access your own data without root access!

- **chmod** – Change access mode

```
chmod +x file
```

- **sudo** – Envokes the power of root for one command.

```
sudo command
[sudo] password for user:
```

- **su** – You become root (use root password)

PERMISSIONS

EXERCISE

Unfortunately, I do not know the configuration of everyone's computer, so this will be quick. There are plenty of cool things we can do and I will show, but no need to do it yourself for now.

1. Use the su command
2. Food for thought: which user has ultimate control over your filesystem?

PIPING

From A to B

- Symbols:
  - | – Parses command output
  - >> / > – Appends to / rewrites file with output
- **echo** – Repeats what you just said

```
echo "hey" >> file
```

- **grep** – Prints lines matching a pattern

```
lspci | grep VGA
```

PIPING

EXERCISE

1. touch a file
2. echo stuff into the file and create the following file:

```
Haikus are easy,
but sometimes they don’t make sense.
Refrigerator.
```

---

3. Use cat, tail, or head to tell me what you put in there.
4. Use grep / piping to read out only the line that starts with “but”

INFORMATION

All the stuff you need to know about your pc

- **man** – Provides manual

```
man program
```

- **top** – Info on what's running
- **free -m** – Info on free memory
- **ps** – Lists active processes

```
ps -ef | grep chromium
```

- **kill, pkill, killall** – kills a process

```
pkill process
```

INFORMATION

- **lsblk** – Info on partitioning scheme
- **du** – Lists size of file

```
du -h file
```

- **df** – Info on free space

```
df -h
```

- **lspci** – Info on all devices

```
lspci | grep VGA
```

INFORMATION

EXERCISE

1. Learn to love your computer. Tell me about your memory usage, and what is currently running.
2. Kill something.
3. What graphics card are you using? (if you are using a mac, try logging into tombo / sango)
4. Use top. Quit from top.

USEFUL SHORTCUTS

Things you might want to know...

- **TAB** autocompletes commands or files. Hit twice to ls.
- "∗" means "everything with this component."

```
ls *bear*
right_to_bear_arms.dat   bear_grills.docx   bear_it.groan
      I_am_beary_funny.not
```

- Stuck program? Press **CTRL+C**!
- Stuck in a text editor? Try the following:
  1. ESC, then q (nano)
  2. ESC, then ":q!", then ENTER (vi / vim)
  3. ESC, then CTRL+C (emacs)

USEFUL SHORTCUTS

- **CTRL+D** logs out
- **CTRL+L** clears the terminal
- **CTRL+A** moves cursor to start of line
- **CTRL+E** moves cursor to end of line
- Spaces are "\ " [BACKSLASH + SPACE] in the terminal
- "$\sim$" means "/home/user"
- "." is your current directory and ".." is the directory under yours

EXTRA

EXERCISE

1. Make a file with a space in it. Delete the file with the space in it.
2. Log out of your terminal.
3. clear your terminal.
4. Touch a bunch of files with similar names and then delete them all with a single command.
5. Type “ls” and then start a filename, press tab to complete it.
6. cd back a few directories until you are in the “/” directory. cd back to home

EXTRA

EXERCISE

1. echo the following into a file called "while":

```
#!/bin/bash
count=1
while [ $count -gt 0 ]; do
    echo The counter is $count
    let count=count+1
done
```

2. Give yourself permission to run the file

```
chmod +x while
```

3. Run the command "./while"
4. Quit the command

FINAL NOTES

- The terminal's a skill and takes time to master.
- Use the cheatcheet (and let me know if you think of other commands)
- Try to use the terminal a few times within the next week. Think about software and imagine doing things gui-less.
- The terminal is good for *certain things*, but not for *everything*. Art's hard to do on an 80×24 grid.