Basics

Exercise 1: ls

`ls` stands for "list".

a) List all files (including hidden files) in your current directory.

```
ls -a
```

b) List all files in your `/bin` directory.

```
ls /bin
```

c) Find the size of your bash executable (`/usr/bin/bash` or `/bin/bash`).

```
ls -la /usr/bin/bash
```

Exercise 2: mv

`mv` stands for move.

a) Make a file `hello` using `touch hello`. Now rename this file in `world`.

```
touch hello
mv hello world
```

b) Move this file into a directory. (e.g. your `Downloads` directory).

```
mv world ~/Downloads
```

Exercise 3: cd

`cd` stands for "change directory".

a) Change into the `/usr/bin` directory, then list all files in there.

```
cd /usr/bin/
ls
```

b) Change into the `/root` directory.

```
cd /root
```

c) Change back to your home directory.

```
cd ~
```

or just

```
cd
```

Exercise 4: cat

a) `cat file`

Exercise 5: rm

`rm` stands for "remove". It directly removes files, it doesn't put them in the trash.

a) Create a file `hello` by typing `touch hello`, then delete it.

```
touch hello
rm hello
```

Exercise 6: mkdir and rmdir

`mkdir` stands for "make directory", `rmdir` for "remove directory".

a) Make a directory called `hello` in your home directory, and then remove it.

```
mkdir hello
rmdir hello
```

b) Can you remove your home directory with rmdir? No, you can only remove empty directories with `rmdir`

Exercise 7: cp

`cp` stands for `copy`.

a) Make a `Backup` directory with `mkdir` and copy your `.bashrc` (bash configuration file) there. Check if it worked with `ls`.

```
mkdir Backup
cp .bashrc Backup
ls Backup
```

b) Copy a directory into `Backup/` (e.g. your `Downloads` directory). What is the difference to copying a file? Difference: `-r` flag (recursive) has to be used.

```
cp -r Downloads Backup
```

c) Remove the `Backup/` directory you just created. Again, the `-r` flag is needed to recursively delete everything in the directory.

```
rm -r Backup
```

Files

Exercise 1

a) `mkdir ~/TempDir`
`cd ~/TempDir`

b) `tee capture.txt`

```
This is some text.
The idea is to test the Linux command.
Now it's time to end the file.
^C
```

c) `wc capture.txt`

d) `cp capture.txt capture-copy.txt`
`diff capture.txt capture-copy.txt`

e) `tee -a capture.txt`

```
some added text into the original file.
^C
```

`diff capture.txt capture-copy.txt`

f) `ls -l`
`chown root:users capture.txt`

- You should have received an error message due to lack of permissions.
- *Hint:* use `sudo`.

```
sudo chown root:users capture.txt
ls -l
```

g) `chmod -w capture-copy.txt`

`tee -a capture-copy.txt`

- This gives an error message because now the file can't be written to.
- Press `^C` (required to close the tee command).
- Extra exercise: Would it be possible to overcome the write restriction using `sudo`?

h) `echo "Hello World!"`

i) `touch -t 201701310001.30 capture.txt`
`ls -l`

j) `ln capture.txt ~/Desktop/capture-link`

k) `file * ~/Downloads/* ~/Pictures/* ~/Music/*`
`file * ~/Downloads/* ~/Pictures/* ~/Music/* > file-types.log`

l) `less file-types.log`

m) `grep directory file-types.log`

n) `find ~/ -name capture-copy.txt`

`find ~/ -name "capture*"`

o) `which ls`
`whereis ls`

p) `cd ~/TempDir`

Make sure you're in the correct directory, then execute:

```
rm *
```

- *Note 1:* If a warning appears asking to confirm removal of write protected files, confirm with `y`.
- *Note 2:* Even file a who's owner was changed to root is also deleted! Ownership doesn't protect the files.

```
cd ~
rmdir TempDir
```

Jobs

Exercise 1

For the first part of the exercise you should have no problem. The script can be run by typing in:

`./infiniteLoop.sh`

For the second part:

a) Run `infiniteLoop.sh &`.
b) Run `fg`.
c) For this it's best to press `Ctrl+Z` to suspend the job and then type `bg` that will move it to the background.

Note: running this script as a background job is pretty ugly because it still spams the console. In order to stop this you would need to redirect the output to somewhere else. However redirection is not the topic we are covering in the exercise and the output helps you to see if the program is still running or not.

d) The PID can be found it using htop, top works similar but is not recommended. In htop you can see the name of the running processes totally on the right and on the left you can see the PID of that process. Remember the PID.
e) One way is to use `kill -9 PID`.

Exercise 2

In order to find the PID we will use the `ps` command. Just type `ps -e` this will list every process in your system. Again find the infiniteLoop.sh and remember it's PID. Since we used `kill -9` last time we will do something else this time. A very nice way to kill a process is using pkill. The nice thing about pkill is that you do not need the PID. Simply type in `pkill -SIGKILL infinite`. Another thing you could use is xkill. However xkill will kill the entire terminal not just the one process.

Remote

Exercise 1

a) `ssh username@supercomputer`
b) `mkdir $yourName`
c) Hit `Ctrl+D`.
d) `scp supercoolNumericalSimulation.c username@supercomputer:$yourname/`
e) Run the ssh command again, compile it as you already know from other exercises.
f) Running you should also know how. Now you should see an output file.
g) `Ctrl+D`, navigate to your remoteExercise Directory `scp username@supercomputer:$yourName/result.t` .

Exercise 2

a) `tmux`
b) `Ctrl+B`, `"` (this is the quotes symbol, typically `SHIFT+2`)
c) `Ctrl+B`, `%` (percent sign, typically `SHITFT+5`) to split and `Ctrl+B`, `Arrow-Key` to navigate.
d) `tmux a` to attach to tmux (note that there is no `-` before the `a`).
e) (no solution)
f) Typically, `Alt+F4` or click the close button on the windows of the terminals. To attach, do again `tmux a`.
g) Repeatedly type `exit` or press `Ctrl+D`.
h) ***TODO!***
i) Simply type `w`.
j) `tmux a` to attach (if it says "no sessions", type `tmux`).
k) `Ctrl+B`, `d`
l) `exit` or `Ctrl+D`

Searching

Exercise 1: `grep`

a) Take a look at the file

```
cd HackingSessionExercises/
cat grep_exercise.md
```

b) Find the word "contains".

```
grep "contains" grep_exercise.md
```

c) Come up with a pattern such that the output are the three lines `Dog`, `Dig` and `Dug`.

```
grep -E "D(o|i|u)g" grep_exercise.md
```

d) Look for all lines beginning with `##`.

```
grep -E "^##" grep_exercise.md
```

e) Print all lines beginning with `##`, but exclude the ones that starts with `###`.

```
grep -E "^##[^#]" grep_exercise.md
```

or you can just use:

```
grep -E "^## " grep_exercise.md
```

f) Search for the lines `Cat`, `Rat`, `Sat` and so on, up until `Bat`, but without the line `Latin`.

```
grep -Eo "^.at" grep_exercise.md
```

g) Produce the same lines as in e), but this time with the line `Latin` included.

```
grep -E "^.at.*" grep_exercise.md
```

h) Try for the lines `Brat` up until `Bat`, but without the line `Latin`.

```
grep -Eo "^..?at" grep_exercise.md
```

i) Adjust your command so that it lists the lines `Brat` up until `Bat`, this time with `Latin`.

```
grep -E "^..?at.*" grep_exercise.md
```

j) Check for the three list elements (these are the lines beginning with `*`).

```
grep "^\*.*" grep_exercise.md
```

k) Investigate for the entire block of Java code (the one at the end of the file).

```
grep -oz "\`\`\`.*\`\`\`" grep_exercise.md
```

or with single quotes:

```
grep -oz '```.*```' grep_exercise.md
```

l) Within that code, output the class that contains the method `out`.

```
grep -Eoz "class[^}]*void out.*}" grep_exercise.md
```

or, with slight cheating:

```
grep -Eoz "class Out.*}" grep_exercise.md
```

m) Pry for all animals that consist of exactly two words.

```
grep -Eo "\+ [A-Z]\w* \w+" grep_exercise.md
```

If you don't want the plus signs, we need to use the pearl regex engine with `-P` and use a technique called "lookbehind". This is rather advanced. If you're more interested, you can look for "lookbehind regex" online.

```
grep -Po "(?<=\+ )[A-Z]\w* \w+" grep_exercise.md
```

n) Probe for all animals that begin with `Danger`.

```
grep -Eo "Danger\w* ?(\w+[- ]?)*" grep_exercise.md
```

o) Finally, if you're still motivated: All animals that contain the word `Danger` *or* `danger`.

```
grep -Eo "([A-Z]\w*|D)anger\w* ?([A-Z]\w*[- ]?)*" grep_exercise.md
```

System management

Exercise 1: Power on/off

a) Try to put your computer into suspend, and wake it back up.

```
systemctl suspend
```

b) Reboot your computer from the command line.

```
reboot
```

Exercise 2: Space left on disk

a) Find out what size (in MB) the exercise directory has.

```
du -sh HackingSessionExercises
```

b) Find out how much space is used by your root directory `/`.

```
df -h /
```

Exercise 3: User management

a) Add a new user called `ExerciseUser`.

```
sudo useradd exercise_user
```

b) Add a new group called `exercise_group`.

```
sudo groupadd exercise_group
```

c) Add your new user to the new group.

```
sudo gpasswd -a exercise_user exercise_group
```

d) Remove the user and the group again.

```
sudo userdel exercise_user
sudo groupdel exercise_group
```

Block devices and file systems

Exercise 1: Partition a disk

a) Create two partitions: One with 1MB size, one with the rest of the available space. Both should be of type `Linux filesystem`. Two options:

(1) `cfdisk`

- Run `cfdisk TestDisk`.
- Select `gpt/dos` label (both will work), in an actual system you will probably want `gpt`.

- Select `New` with the arrow keys, hit enter.
- Edit the size to 1M, hit enter.
- Select `Free space` with the arrow keys.
- Same thing again: New, use the default size (all space left), hit enter.
- Select `Write` with the arrow keys.
- Type `Quit`.

(2) `gdisk`

- Run `gdisk TestDisk`.
- Enter `n` for new partition.
- Select number 1.
- Set first sector to default (just hit enter).
- Set second sector to +1M.
- Use the default partition type.
- Do the same thing for the second partition again, but use the defaults for both sectors, and use partition number 2.
- Enter `w` to write your changes.

Exercise 2: Create a file system

a) Create a FAT file system on `partition1`.

```
mkfs.vfat partition1
```

b) Create a ext4 file system on `partition2`.

```
mkfs.ext4 partition2
```

- You can check if this worked by looking at the output of `file partition1` and `file partition2`.

Exercise 3: Using dd

a) Write `image.iso` to `TestDisk` using `dd`.

```
dd if=image.iso of=TestDisk
```

Software management

Exercise 1: Package manager

This section is specific to your distribution. In the following we will present solutions for Ubuntu and OpenSUSE based distributions. If you're in trouble running another distro, please contact a helper.

Under Ubuntu based systems

a) `sudo apt update`
b) `sudo apt upgrade`, then press ENTER when asked `[Y/n]`. Note the capital Y: it means that if you simply press ENTER (without typing anything) this option will be chosen.
c) `sudo apt autoremove`, press ENTER when asked as above. This will uninstall old kernels from your system and free space.

d) `apt search youtube-dl`
e) `sudo apt install youtube-dl`
f) `youtube-dl` should display something like "Usage: …"
g) `man youtube-dl`, quit with `q`
h) `sudo apt autoremove youtube-dl`

i) `df` shows your current disk usage (use `df -h` for human-readable numbers). On the right hand side, you can see "Mounted on". Look for `/`, which stands for your system partition. The used and free space can be seen in the other columns. To clean the package cache: `sudo apt clean`


Under OpenSUSE

a) `sudo zypper ref` (or `sudo zypper refresh`)
b) `sudo zypper up` (or `sudo zypper update`), then press ENTER when asked `[y/n/?] (y)`. Note the y in parentheses: it means that if you simply press ENTER (without typing anything) this option will be chosen.
c) (skip this under OpenSUSE)
d) `zypper se youtube-dl` (or `zypper search youtube-dl`)
e) `sudo zypper in youtube-dl` (or `sudo zypper install youtube-dl`)
f) `youtube-dl` should display something like "Usage: …"
g) `man youtube-dl`, (if asked, press ENTER), finally quit with `q`
h) `sudo zypper rm youtube-dl` (or `sudo zypper remove youtube-dl`)
i) `df` shows your current disk usage (use `df -h` for human-readable numbers). On the right hand side, you can see "Mounted on". Look for `/`, which stands for your system partition. The used and free space can be seen in the other columns. To clean the package cache: `sudo zypper clean`

Exercise 2: Ubuntu only: Installing software from PPA

a) The page we're looking for is https://launchpad.net/~stebbins/+archive/ubuntu/handbrake-releases
b) `sudo add-apt-repository ppa:stebbins/handbrake-releases`
c) `sudo apt update`
d) `sudo apt install handbrake-gtk`
e) (no solution)
f) `sudo apt autoremove handbrake-gtk`
g) Open up your favourite text editor (e.g. `nano /etc/apt/sources.list`), remove the two lines described below and save:

```
deb http://ppa.launchpad.net/stebbins/handbrake-releases/ubuntu YOUR_UBUNTU_VERSION_HER
deb-src http://ppa.launchpad.net/stebbins/handbrake-releases/ubuntu YOUR_UBUNTU_VERSION
```

h) `sudo apt update`
i) `sudo apt install handbrake-gtk` or `apt search handbrake`

Exercise 3: Installing a package manually

a) (no solution)
b) (individual)
c) `cd Downloads; ls`

d) Depends on your distro, at `<TAB>` press the tabulator key for autocompletion:

- Ubuntu: `sudo apt install teamviewer<TAB>`, will complete with …deb
- OpenSUSE: `sudo zypper in teamviewer<TAB>`, will complete with …rpm

e) (no solution)

f) Depends on your distro:

- Ubuntu: `sudo apt autoremove teamviewer`
- OpenSUSE: `sudo zypper rm teamviewer`

Exercise 4: Compile from source using git

a) (no solution)

b) The desired URL is https://github.com/AltraMayor/f3.git

c)
```
cd Downloads
git clone https://github.com/AltraMayor/f3.git
```

d) `cd f3`

(On Ubuntu, according to the section "Install Dependencies" in the README): `sudo apt-get install libudev1 libudev-dev libparted0-dev`

`make`

`sudo make install` (Without sudo, the command fails due to missing permissions, you are supposed to figure that out on your own)

e) `f3read` should complain "f3read: The disk path was not specified".

f) We will first look around, then delete the files:

```
ls /usr/local/bin/f3*
```

```
/usr/local/bin/f3read
/usr/local/bin/f3write
```

```
ls /usr/local/share/man/man1/f3*
```

```
/usr/local/share/man/man1/f3read.1
/usr/local/share/man/man1/f3write.1
```

```
rm /usr/local/bin/f3*
rm /usr/local/share/man/man1/f3*
```

Exercise 5: Installing from a self-containing install script

a) (no solution)

b) `cd Downloads`

c) `chmod a+x postgresql<TAB>` where TAB is autocompletion by pressing the Tabulator key on your keyboard.

d) `./postresql<TAB>`

e) `sudo ./postresql<TAB>`

f) (no solution)

g) `/opt/PostgreSQL/YOUR.VERSION/bin/postgres` should complain that it does not know where to find the server configuration file.

h) `sudo ln /opt/PostgreSQL/YOUR.VERSION/bin/postgres /usr/bin/`

i) `postgres` should now print the error message about the missing config file.

j)
```
cd /opt/PostgreSQL/YOUR.VERSION
sudo ./uninstall-postgresql
```

Press "Yes"

k) `cd` (Leave the `/opt/PostgreSQL` directory since you are going to delete it)

```
sudo rm -rf /opt/PostgreSQL
```

l) `sudo userdel postgres`