Basics

This aims to train usage in basic commands.

Flags

flags are options you can add to a command: for example, if you want to list all files including hidden files, you will use the corresponding flag (that you will find in the manual of the program). Flags start with `-` or `--`. For example, `ls -r` lists all files in reverse order, `ls --reverse` does the same. `ls -rR` (yes, bash is case-sensitive) lists all files in reverse order and recursively and is equivalent to `ls -R -r` and to `ls --reverse --recursive`.

man

`man` is the command that you will need all along this hacking session. It stands for 'manual' and displays the manual of a command, i.e. what the command does/can do, which options are available or not, what its input and output are etc. The content that `man` displays is also called a manpage. Almost all commands (all that we will deal with today) have manpages that are provided together with the program. The `man` command has its own manpage! To learn how the `man` command works, you can type `man man`. For every command unknown to you, you should take a look at the manpage.

Wildcards

Wildcards are a very useful tool. They allow you to apply a command to all files that have a certain structure. There are many different wildcards, the most useful one is `*`. `*` means: any set of characters. So for example `*.jpg` designates all files that end with `.jpg` (`a.jpg`, `abcd.png.jpg` and so on).

sudo

Not every user can do anything on your computer, some users do not have read or write access to some files or directories. Only the superuser can do everything. To execute `command` as a superuser, type `sudo command`. The program then asks the user for his password (if he is in the sudo group) before executing the command. Superusers have all rights, including the right to scrap the computer, so be careful if you execute something with sudo since your computer may warn you (sometimes) but he will not stop you from damaging the computer.

Exercise 0: man

Before you can start to learn anything, you should make yourself familiar with the `man` command. Type `man man` in the commandshell and hit enter. You don't have to read the whole thing, but skimming over it is definitely a good idea. Also, while the manual of `man` is open, you can hit `h` to make a window with keyboard shortcuts pop up. This is a good opportunity to take a look at them, because it is vital that you know how to search a manpage and how to move around within it quickly!

Exercise 1: ls

`ls` stands for "list".

a) List all files (including hidden files) in your current directory.
b) List all files in your `/bin` directory.
c) Find the size of your bash executable (`/usr/bin/bash` or `/bin/bash`).

Exercise 2: mv

`mv` stands for move.

a) Make a file `hello` using `touch hello`. Now rename this file into `world`.
b) Move this file into a different directory. (e.g. your `Downloads` folder).

Exercise 3: cd

`cd` stands for "change directory".

a) Change into the `/usr/bin` directory, then list all files in there.
b) Change into the `/root` directory.
c) Change back to your home directory.

Exercise 4: cat

`cat` stands for con*cat*enate files and print to the standard output.

a) Look at the output of whatever file you desire.

Exercise 5: rm

`rm` stands for "remove". It directly removes files, it doesn't put them in the trash. Any data that you remove with `rm` is gone forever.

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

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.
b) Can you remove your home directory 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`.
b) Copy a directory into `Backup/` (e.g. your `Downloads` directory). What is the differece to copying a file?
difference
c) Remove the `Backup/` directory you just created.

Files

This section aims to train usage with basic commands for files. Here you will find a basic use and most common syntax of the commands, as well as some exercises at the end. However the commands are much more powerful and you are invited to self-explore and experiment with other option attributes and alternative syntaxes (Hint: remember command `man` from the section *basics*).

tee

`tee` - read from standard input and write to standard output **and** files.

Syntax:

```
tee [OPTION]... [FILE]...
-a  --append    append input to the given file(s), do not overwrite.
```

A little hard to follow if this is supposed to be a separate command or if it is supposed to go after

wc

`wc` stands for "wordcount". It prints newline-, word-, and byte-counts for each file, and a summary line if more than one FILE is specified.

Syntax:

```
wc [OPTION]... [FILE]...
```

diff

`diff` stand for "difference", compare files line by line.

Syntax:

```
diff [OPTION]... [FILE]...
```

chown

`chown` stands for "change owner". It can change also group owner.

Syntax:

```
chown [OPTION]... [OWNER][:[GROUP]] FILE...
```

chmod

`chmod` stands for "change file mode bits" and controls the permissions: who can read and write a given file? Is it an executable? Permissions can be read using `ls -l`.

Syntax:

```
chmod [OPTION]... MODE[,MODE]... FILE...
```

echo

`echo` - display a line of text

Syntax:

```
echo [SHORT-OPTION]... [STRING]...
```

touch

`touch` - change file timestamps (access and modification times) to the current time or to a specific one defined through the options. Most useful to create new, empty files.

Syntax:

```
touch [OPTION]... FILE...
-t STAMP`  use [[CC]YY]MMDDhhmm[.ss] instead of current time
```

Note: a FILE argument that does not exist is created empty (this mode of operation was already used in the section *basics*).

ln

`ln` - make links between files.

Syntax to create a link to TARGET with the name LINK_NAME:

```
ln [OPTION]... [-T] TARGET LINK_NAME
```

file

`file` - determine file type.

Syntax:

```
file [OPTION]... FILE...
```

less

`less` - A file viewer. `less` allows to scroll forward and backward in a file with the arrow keys. The current file can be opened in an editor by pressing `v`, and `less` can be quit by pressing `q`.

Syntax:

```
less [OPTION]... [filename]...
```

grep

`grep` - searches the named input FILEs for lines containing a match to the given PATTERN (a regular expression). By default, grep prints the matching lines.

Syntax:

```
grep [OPTIONS] PATTERN [FILE...]
```

find

`find` - search for files in a directory hierarchy with a regular expression.

Syntax:

```
find [starting-point...] [expression]
```

which (whereis)

`which` - shows the full path of (shell) commands.

Syntax:

```
which [options] [--] programname [...]
```

`whereis` - locate the binary, source, and manual page files for a command.

Syntax:

```
whereis [OPTIONS] name...
```

Exercise 1

a) For this set of exercises, create the temporary working directory `TempDir` in your home directory. Now, make it your current working directory.

b) Create the file `capture.txt` containing some text typed in the keyboard.
Hint 1: Once you have found the correct command, you can type some words in the keyboard over several lines using `<Enter>`
Hint 2: Press `^C` to end the command
c) How many lines, words and characters does this file have?
d) Make a copy of the file and compare both files.
e) Append text to the copied file and compare with the original.
f) Assign user `root` and group `users` as new owner of the original file and check the change. *Hint:* Check the current owners of the files first
g) Change the file attributes to make it write protected and try to append text again.
h) Get in the terminal a new line which contains "Hello World!".
i) Change the date and time of the file `capture.txt` to 31.01.2017 00:01 and check the result.
j) Create a link to the `capture.txt` file in the desktop, minimize the terminal and look for the new link on the desktop. Make double-click on the link, does the text editor open the file?
k) Check the file types in the following directories: your test directory, downloads, pictures and music. Make a list of all those file types creating the `file-types.log` file.
l) Explore the log file `file-types.log` with the command `less`.
m) List all lines which contain a directory in the previous log file.
n) Starting from your home directory, search for the file `capture.txt`.
o) Obtain the location of the command `ls`, as well as the location of it's manual.
p) Delete the content of `TempDir` directory and check it was deleted. Then delete the directory.

Maybe "Generate" would be a clearer instruction for this step

Jobs

Exercise 1: fg, bg, &, top, htop, kill, pkill, killall

a) Install `htop` if you do not already have it.
b) Navigate to the `jobs` directory.
c) Run the `HackingSessionExercises/infiniteloop.sh`
d) Now your terminal is pretty unusable.
e) You can kill the running process by pressing Ctrl+C.

Now we want to learn how to kill processes running in the background. This is helpful when you have programs that have hung themselves up and have to be shut down. Collectively, the commands you will be using are known to provide "process control", that is, functionality similar to task managers known from Windows/OS X. So we will basically learn how to use Ctrl+Alt+Delete under Linux ;)

f) Run the script used in exercise c) again, but in a way such that it becomes a background job/
g) Make it to be a foreground job. Do you notice a difference?
h) Now move it back to be a background job. Can you still kill it with Ctrl+C?
i) Try and find out the pid (process id) of your started process (you might want to do this in a different terminal).
j) Use the kill command to get rid of the process.

Exercise 2: ps, kill, killall, pkill

Do the same as above but this time without using top or htop and use a different method to kill it. You also do not need to do all that background foreground stuff again.

Can you find the parent of the process you started?

Remote

Exercise 1: ssh, scp

In this exercise you will learn how to access a compute cluster and do stuff on it and get results. For this please first find the C program called `supercoolNumercialSimulation.c`. `ssh` and `scp` are the most important commands you will use within this exercise. `ssh` is used to connect to a machine remotely, while `scp` is used to copy files. For this exercise, it's probably best if you don't lose yourself in trying to understand how `ssh`, resp. `scp` works exactly, because there is quite a bit of cryptography and complicated network stuff involved. To get a general idea of how it works: `ssh` stands for "Secure Shell" and `scp` stands for "Secure Copy". Both make use of the cryptographic library `openSSL` in order to be able to exchange your data securely.

a) Connect to $Supercomputer via ssh.
b) Make yourself familiar on the remote machine and make a directory with your name as the directory name.
c) Return to your local machine.
d) Copy the C program you found before to your directory on the remote machine.
e) Again access the machine and compile the program.
f) Run it.
g) Find it's output and copy it to your remoteExercise directory on your local machine.

Exercise 2: tmux

This exercise is about using the terminal multiplexer tmux. Using it gives you several advantages, here are a few:

- If the connection gets lost, the process on the remote machine keeps running.
- You can turn off your local computer without interrupting the remote calculation. You may re-connect to the remote tmux session any time to check how your calculations are doing.
- Tmux allows you to tile your terminals and have several terminals open through the same ssh connection.
- You may collaborate with someone else and you both see the same content.

Consider looking up tmux keyboard shortcuts online.

a) Start tmux on your local machine.
b) Split your tmux horizontally (so that you have one terminal on top of the other).
c) Split the upper half vertically and try to navigate into each of the three parts.

d) Open up another terminal and connect to tmux (Caution: this is not the same thing as starting tmux).
e) Verify that whatever you do in one terminal is also reproduced in the other one.
f) Close both terminal windows, open up a terminal and attach. Verify that tmux kept running in the background and that it's still in the state in which you left it..
g) Close every bit of tmux until tmux is not running any more.
h) We have prepared a machine running that you may now connect to using SSH: URL: ***TODO!*** The user name is ***TODO!*** and the password ***TODO!***.
i) Type the command `w` to see how many users are currently logged in.
j) Attach to tmux (if none is running, start it). Note that there might be an arbitrary number of people already logged in and all of you type in the same window. Have fun ;-)
k) Detach from tmux, but not by closing the terminal nor by exiting tmux. Check online for how to do this.
l) Log out from the remote server.

Searching

These exercises cover the `grep` *and* `find` *commands in detail.*

We're going look at two of the most important commands that perform "search-like" operations. Their names are `grep` and `find`. `find` is a command that is used for finding certain files. `grep` is a bit more complex and allows you to search a file's contents.

I would recommend that you do the exercises on both `grep` and `find` - they are very powerful when used in combination. If you have to choose only one, I recommend `grep` because I use it much more often.

Exercise 1: `grep`

`grep` searches the text using so-called Regular Expressions (commonly referred to as Regex). Regexes are textual patterns that encode a set of strings. In other words, a Regex is simply an efficient way to specify many strings. For example, `un(b|f)ounded` is a Regex that stands for both "unbounded" and "unfounded". There are many Regex standards, and they may differ in the special characters they use. It's probably a good idea to read https://en.wikipedia.org/wiki/Regular_expression#Basic_concepts. After that, the `grep` manual provides some further information.

Fun fact: `grep`'s name comes from *g*lobally search a *r*egular *e*xpression and *p*rint. This is a wordplay on the `ed` command g/re/p, which searches for a Regex and prints it. `ed` is one of the very first text editors created. Nowadays, no one uses `ed` anymore because there is no reason to do so (maybe apart from bragging rights), but `ed` has influenced a *lot* of commands commonly used in Unix.

Should clarify where this file is located / how to access it

a) Take a look at the file `HackingSessionExercises/grep_exercise.md`. In this exercise, you will search the file for various patterns using `grep`. The goal is that you use `grep` to output *only* what is written in the exercises.
b) Find the word "contains".
c) Come up with a pattern such that the output are the three lines `Dog`, `Dig` and `Dug`.
   Hint: Take a look at the -E or the -P flag for `grep`. You will need one of them here.
d) Look for all lines beginning with `##`.
e) Print all lines beginning with `##`, but exclude the ones that starts with `###`.
f) Search for the lines `Cat`, `Rat`, `Sat` and so on, up until `Bat`, but without the line `Latin`.
g) Produce the same lines as in e), but this time with the line `Latin` included.
h) Try for the lines `Brat` up until `Bat`, but without the line `Latin`.
i) Adjust your command so that it lists the lines `Brat` up until `Bat`, this time with `Latin`.

j) Check for the three list elements (these are the lines beginning with `*`).
k) Investigate for the entire block of Java code (the one at the end of the file).
l) Within that code, output the class that contains the method `out`.
m) Pry for all animals that consist of exactly two words.
n) Probe for all animals that begin with `Danger`. *Note:* Take particular care that "Booplesnoot" and "Trash Panda" aren't part of your output. This makes it more tricky.
o) Finally, if you're still motivated: All animals that contain the word `Danger` *or* `danger`.

Exercise 2: `find`

In this exercise you'll use the `find` command to do some complicated file finding. You can "chain" filters, so for searching for a file that is called "asdf" and has size less than 50MB, you can do `find -name asdf -size -50M`. As usual, there are some more examples at the bottom of the manpage.

a) Navigate to your home directory.
b) Using `find`, search for all files that were created less than 5 days ago.
c) Using `find`, search for all the files that you have write access for.
d) Search for all files that start with the letter `d`.
e) Search for all files smaller than 1MB, but larger than 10KB.
f) Find all files in the directory `Desktop` (or some different directory).
g) Search for all files starting with the letter `d` and execute `cat` on them.

No solutions to these

Exercise 3

Combining the knowledge of `grep` and `find`, we can search for some incredibly specific stuff! Look at the `exec` flag in the manual for `find`.

a) Navigate to your etc directory (`/etc`, on most distributions).
b) Within that directory, search for all lines of text containing the word "root".
c) If you received a lot of error messages in exercise b), try to limit your search to files which you have read permissions for.
d) Search for all files containing an IP address.
e) Output all comments (lines starting with a `#`) of all files with a name ending in `.conf`.

System Management

Exercise 1: Power on/off

You can use the shell to do various system management tasks. To shut down, reboot, or put your computer in suspend mode, you can use `reboot`, `poweroff`, or `systemctl suspend` respectively. (You might need to use `sudo` on your system)

a) Try to put your computer into suspend, and wake it back up.
b) Reboot your computer from the command line.

Exercise 2: Space Left on Disk

Useful commands: `df`, `dh`

a) Find out what size (in MB) the exercise directory has.
b) Find out how much space is used by your root directory `/`.

Exercise 3: User Management

Useful commands: `useradd`, `userdel`, `groupadd`, `groupdel`, `gpasswd`, `passwd`

a) Add a new user called `ExerciseUser`.

b) Add a new group called `ExerciseGroup`.

c) Add your new user to the new group.

d) Remove the user and the group again.

   Hint: If you are confused by `gpasswd` and `passwd`, consider the following: One of them is used to set the password for a user, the other is used to add a user to a group.

Block devices and file systems

This exercise is a bit more advanced! It will involve creating/editing partitions from the command line and manipulating block devices and file systems.

WARNING: If any of the following exercises require you to run them as root, you should not be doing that!

Exercise 1: Partition a disk

Useful tools: `cfdisk`, `gdisk`

For the purpose of this exercise, we provided a file called `TestDisk`. You can treat it like a regular block device (like `/dev/sda`), but using this file you will not damage your live system.

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

Exercise 2: Create a file system

Useful tools: `mkfs.ext4`, `mkfs.vfat`

For the purpose of this exercise, we provided two files: `partition1` and `partition2`. This will again not change your actual system.

a) Create a FAT file system on `partition1`.
b) Create a ext4 file system on `partition2`.

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

Exercise 3: Using dd

WARNING: Don't do the following to your actual file systems! You can destroy your file systems! Especially don't run any of the commands as root.
With `dd` you can write images to a disk, like an image of a Linux distribution.

a) Write `image.iso` to `TestDisk` using `dd`.
   You can check if this worked by looking at the output of `cat TestDisk`.

Software management

This exercise is about installing and managing software.

Exercise 1: Package manager

a) Refresh the sources of your package manager
b) Install updates for your system
c) If you are under Ubuntu, remove old software by using `autoremove` (otherwise skip this step)
d) Search for a package called `youtube-dl`
e) Install the package you found in the previous step
f) Run `youtube-dl` to see if it works. It should display something like "Usage: ..."
g) Open the man page for `youtube-dl`
h) Remove `youtube-dl` from your system
i) Look at the disk usage of your system partition (Hint: `df`). Then clean the cache of your package manager. Look at the disk usage again: some space should have been freed.

Exercise 2: Ubuntu only: Installing software from PPA

Skip this exercise if your distribution is not Ubuntu-based.

If a program is not in the official package sources, you might still be able to install it using your package manager. Under Ubuntu, a developer can create a Personal Package Archive (PPA), sort of a personal software channel which you can integrate into your sources list. After refreshing your sources, you may then install and upgrade software from the PPA as if it was an official source.

Caution: Anyone can own a PPA and distribute arbitrary software over it. Make sure you trust the developer!

In this exercise, we are going to install a video transcoder called Handbrake.

a) Google for "Handbrake PPA" and pick the Launchpad link
b) Use the command `add-apt-repository` to add the PPA to your sources as described on Launchpad
c) Update your software sources
d) Install `handbrake-gtk` using your package manager
e) In your start menu, search for "Handbrake" and open the program. Its logo shows a pineapple. The program should open.
f) Uninstall `handbrake-gtk`
g) We are now going to remove the PPA from the system. There is no command to do this. Edit the file `/etc/apt/sources.list` and remove the two lines that contain "stebbins".

h) Refresh your software sources again.
i) Attempt to install `handbrake-gtk`. There should be no such package.

Exercise 3: Installing a package manually

Event if there is no package available in the software source and there is no PPA available, there might be a package available online that you can download manually and install. This means that you download the package file using your browser or the `wget` command and then tell your package manager to install it. This way, the package can be removed **but not upgraded automatically** by your package manager.

Note: In this method you need to trust the developer since the package manager only verifies the integrity of the package, not the security.

In this exercise we are going to install the software TeamViewer. Note that this is a proprietary (non-free) software.

a) In your browser, go to https://www.teamviewer.com and visit the Downloads page
b) Select the version suitable for your OS and architecture (32- vs. 64-bit) and download it
c) Open a terminal, navigate to your Downloads folder and check that the file exists
d) Tell your package manager to install the file (use Tab completion when typing the filename teamviewer…)
e) Start `teamviewer` to check that the program works
f) Use your package manager to remove the package.

Exercise 4: Compile from source using git

Sometimes there is no package at all available for the software you are looking for. In this case, you need to go online and download a program manually. For Free and Open Source Software, you often get the sources and compile the program suitable for your OS and architecture.

In this exercise, we are going to download the source of a software called `f3` from Github, compile it into an executable program and install it. **Note that programs installed that way are not managed by your package manager.** You will need to update and uninstall them manually. Only use this option if you are sure that there is no package with your software for your distro. **Note:** When installing software this way, you will download and run code from a source that is not verified by your distro's community. **Only do this with software you trust.**

Before you start, you need to make sure your system has the required utilities to compile a program:

- Ubuntu-based: `apt-get install build-essential`
- OpenSUSE: `sudo zypper in --type pattern devel_basis`

a) In your browser, visit the GitHub page of `f3`: https://github.com/AltraMayor/f3

b) In the top right, there is a green button saying "Clone or Download". Copy the https URL to your clipboard.

c) In a terminal, go to your Downloads folder and type `git clone URL` where you *replace* `URL` by the URL you just copied in the previous step. This will create a new directory `f3` with the contents of the Github repository you just cloned.

d) Go into the `f3` directory and follow the instructions of the README to compile (make) and install the main software without the extras (you will type 2-3 commands).

e) Type `f3read` to verify that the program has been installed correctly. It should complain "f3read: The disk path was not specified"

f) We will now uninstall f3. Unfortunately, the developer has not included a script to uninstall f3, meaning that we must manually delete the files that were created by the install step. You probably get a feeling now why the package manager is always the preferrable option to install software. To locate the files we want to delete, we look at the output of the previously run install command. To help you, there are 4 files to be deleted:

   - 2 files in `/usr/local/bin/`
   - 2 files in `/usr/local/share/man/man1/`

Note: Every `sudo make install` is different and you need to figure out on your own how to undo its effects. Nice developers provide an option `sudo make uninstall`.

Exercise 5: Installing from a self-containing install script

It is possible to build setups for Linux. This means that you download some file (typically the file name ends with `.run`) and blindly execute it. The file self-contains all the data it needs and installs the program on your computer. Just like under Windows, **you don't know and have no control over what's going to happen when running such a file**. Outdated software may destroy your computer and you need to trust the developer.

In this exercise, we will install PostgreSQL from a `.run` archive. PostgreSQL can be installed with the package manager on most distros, but for the sake of training, we'll do the `.run` Method here.

a) In your browser, visit https://www.enterprisedb.com/downloads/postgres-postgresql-downloads#linux and download the newest version of the installer for your distro and architecture.
b) Open a terminal and navigate to your Downloads folder
c) Use `chmod` to make the freshly downloaded file executable
d) Attempt to run the installer as an unprivileged user (using `./postgres<TAB>`) where TAB is autocompletion by pressing the Tabulator key on your keyboard
e) Do the same thing as root. **Note that now you are executing an unknown program with super user privileges. This is bad practice and should be done as your very last choice!**
f) Click yourself through the installer. Leave default values as they are. Type any password when asked. Do not run Stack Builder.
g) The software is now installed in your `/opt` directory. To run it, execute the program `postgresql` which you can find in `/opt/PostgreSQL/YOUR.VERSION/bin/`. It should complain that it does not know where to find the server configuration file.
h) It's cumbersome to have to type the full path every time we want to start up the program. We can either add PostgreSQL's bin folder to the $PATH variable (out of scope of this exercise) or create a symlink to it. For the sake of the exercise, create a symlink to the executable under `usr/bin/postgres`
i) Verify that the program can now be run without typing the entire path
j) PostgreSQL comes with an uninstaller. You can find it right above the `bin/` directory, it's called `uninstall-postgresql`. Run it as root.
k) The installer tells you that the data directory and service user account have not been removed. Remove `opt/PostgreSQL` manually.
l) Remove the user `postgres` that the installer created

Note: Every `.run` is different and you need to figure out on your own what needs to be done to undo it (if possible).