Lab 1

Members: Fangwen Liao (3439869), Yijin Wang (3476217), Moritz Pfeffer (3261671)

Exercise 1

(a) Use the lsb_release command to print all the distribution details.
According to the man page of *lsb_release* it is used to print information about distribution.
With the **-a** option it can print all the distribution details.

```
moritzpfeffer@debian:~$ man lsb_release
Lots of text
```

```
moritzpfeffer@debian:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description: Debian GNU/Linux 9.13 (stretch)
Release: 9.13
Codename: stretch
```

(b) Use the uname command to print the kernel release info.
In man page of *uname* the **-r** option is for printing the kernel release info.

```
moritzpfeffer@debian:~$ man uname
Lots of text
```

```
moritzpfeffer@debian:~$ uname -r
4.9.0-16-686
```

Exercise 2

(a) Determine the shell that is used by default by using the echo command and the $SHELL environment variable.
The value of an environment variable can be simply accessed by command *echo*.

```
moritzpfeffer@debian:~$ echo $SHELL
/bin/bash
```

This shows that the default shell is *bash* located at */bin/bash*.

(b) List all the directories found in the $PATH environment variable.

```
moritzpfeffer@debian:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

Thus, the directories in $PATH are:

- /usr/local/bin
- /usr/bin
- /bin
- /usr/local/games
- /usr/games

Exercise 3

List the number of scripts that run
(a) at run level S, (b) run level 2, and (c) run level 5.
In the man page of *wc* the **-l** option shows the number of lines of data fed to the command.
To count the files in a directory, pipe *ls* with *wc -l*.
Scripts of each run level can be found in */etc/rc.x* files.

```
moritzpfeffer@debian:/etc$ ls /etc/rcS.d/ | wc -l
10
moritzpfeffer@debian:/etc$ ls /etc/rc2.d/ | wc -l
20
moritzpfeffer@debian:/etc$ ls /etc/rc5.d/ | wc -l
20
```

Therefore, at run level S **10** scripts are running, at run level 2 **20** scripts and at level 5 **20** scripts.

Exercise 4

Research the difference between systemd and init (System V init)
(a) describe in your own words the difference between these systems
I will describe four differences between systemd and init.
One important motivation for systemd was to speed up boot times. To achieve this systemd starts services **on demand** and in **parallel**, while init starts services **serially**. [1]
Secondly init starts services through shell script, while systemd recommend .service files. Thus, I would say that init uses an **imperative** approach (scripts), whereas systemd prefers a **declarative** one. [2]

Thirdly, systemd's .service files and other unit files can be grouped into **targets**, which **replace init's runlevels**.
Furthermore, systemd contains many more components than init. For example, it features a ntp-implementation called systemd-timesyncd and systemd-timer which can run recurring tasks like cron does.
Thus, the fourth difference is that **systemd is less focused and larger** than the original init. [3]
(b) determine which of the two is used by the operating system you have installed in VirtualBox. How can you tell?
The archlinux wiki on systemd tells us that systemctl is the "main command used to introspect and control systemd is systemctl". By entering "systemctl" into the terminal and executing it i confirm that it is present in the VM. From that i infer that systemd is used.

Additionally, systemd or init will run as the first process in operating system and according to man page, the PID should be 1.

```
fangwenliao@debian:~$ ps -e | less -X
  PID TTY          TIME CMD
    1 ?        00:00:01 systemd
```

or

```
fangwenliao@debian:~$ ps -ef | less -X
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 21:22 ?        00:00:01 /sbin/init
```

```
fangwenliao@debian:/sbin$ ls -l init
lrwxrwxrwx 1 root root 20 Jul  8 15:07 init -> /lib/systemd/systemd
```