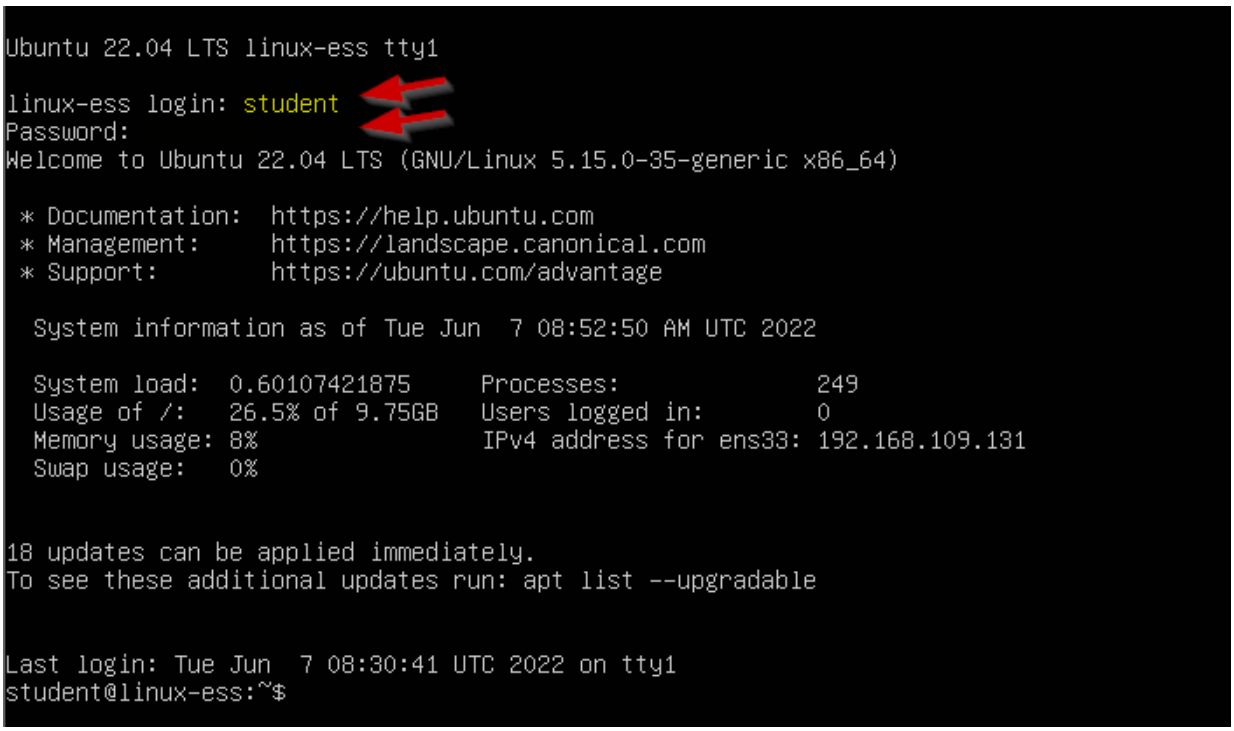
Command line interface

When booting the virtual machine, all that will show is a black screen with some white text:



You will notice that there is no mouse pointer available. We will only use our keyboard as input device. This environment is called a *command line interface (CLI)*. There is no *graphical user interface (GUI)* present in Ubuntu server. One of the reasons why they chose this is because having no GUI present will save system resources and narrows the attack surface. A CLI is also proven to be a very efficient & trustworthy way of working & interacting with an operating system and its services.

> You can login with you username `student` and the password you have set `pxl`. Notice that you do not see what you are typing for the password. Just type the password and press `enter`.

CLIs will be a lot more intersting towards automation, something that is harder

when using a GUI. We can see this trend in Windows systems as well, where *Powershell* is becoming more and more popular to configure Windows servers and performing automation tasks.

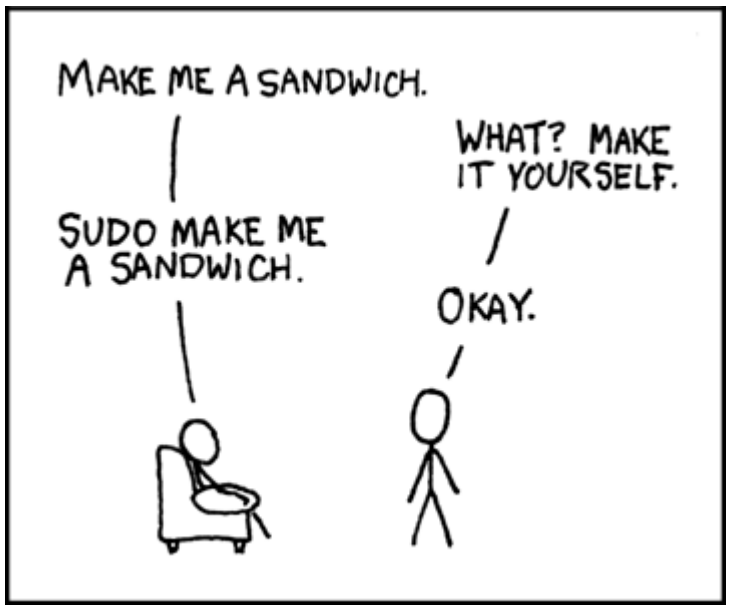
The prompt

After logging in, you are shown the following line in the CLI. This is called the *prompt*:

The prompt exists of multiple parts that give us more information about the system we are using. We see information about our user and the hostname of the server that we are logged into.

The `~` symbol is an abbreviation of the homefolder of the logged in user (in our case the folder `/home/student`). We will learn about paths & folders in a later chapter. For now you can compare this to the path `C:\Users\student` in Windows. So what we actually see in between the `:` and the `$` sign is a path pointing to the folder we are currently working in.

Linux prompt definition



commands and options

Whenever you type something using the keyboard the input will apear after the `$` sign. The `$` sign indicates the end of the prompt and the start of the user's input (= a command). To use the operating system using a CLI, we will have to use commands. The first command we will use is the `echo` command:

```bash
echo hello world
```

The string `hello world` is considered an *argument* of the `echo` command. This command just prints out whatever argument we provide.

The second command we will learn to use is the `shutdown` command:

```bash
sudo shutdown now
```

This command will shut down the Ubuntu server machine. This is also the propper way of shutting down the virtual machine! The `sudo` command stands for *super user do*. Some commands require administrator rights. By adding the `sudo` command in front, you will run this command as the *super user*. This user, in Linux, is called *root* (compared to the administrator user in Windows). The example actually exists out of 2 commands. The `sudo` command followed by the `shutdown command` . The `shutdown` command will use *the parameter* `now` .

Commands often have all kinds of options (beginning with a hyphen (-)) and paramaters (separated with spaces) to extend the functionality of a command. The `shutdown` command will take an (optional) value that defines when the server needs to actually shutdown. If no value is given, it will plan the shutdown task 1 minute after running the command. Let's look at some other options with the command `man shutdown` :

```bash
SHUTDOWN(8)                                    shutdown
```

```
NAME
       shutdown - Halt, power-off or reboot the machine

SYNOPSIS
       shutdown [OPTIONS...] [TIME] [WALL...]

DESCRIPTION
       ...

OPTIONS
       The following options are understood:

       --help
           Print a short help text and exit.
       -H, --halt
           Halt the machine.
       -P, --poweroff
           Power-off the machine (the default).
       -r, --reboot
           Reboot the machine.
       -h
           Equivalent to --poweroff, unless --halt is specified.
       -k
           Do not halt, power-off, reboot, just write wall message.
       --no-wall
           Do not send wall message before halt, power-off, reboot.
       -c
           Cancel a pending shutdown. This may be used to cancel the effe
           argument that is not "+0" or "now".
```

As you can see the `shutdown` command has a lot of options that we can use to extend the basic functionality of the command. Every command has his own set of unique options.

> ℹ Everything in Linux is case sensitive: commands, options, arguments, file- and foldernames, ...

Linux command structure

manpages

man

In Linux we use a CLI. Therefore we will have to work with various commands. Our Ubuntu installation has all kinds of commands built-in. To find commands that we can use we could use Google, but the operating system itself also has information about all installed commands. This info is bundled in *manpages* (short for manual pages). You can access these manpages through the `man` command.

Type man followed by a command (for which you want help) and start reading:

```bash
man shutdown
```

Not only commands have their own manpage, config/system files might have a manpage as well. The command below shows the manpage of the `sudo.conf` file:

```bash
man sudo.conf
```

manpages are pretty big and exist out of multiple pages and/or sections. To view the next page in a manpage you can press the `spacebar` or you can use the `arrow keys`. Manpages are pretty easy to search. Just type `/` followed by a keyword. The manpage will highlight the first occurence of that keyword. You can use the key `n` (*next*) to go to the next occurence of the keyword. Exiting a manpage can be done by pressing the `q` (*quit*) key. Want to know more about manpages? type `man man`!

You can search in the description of installed commands by using the `-k` option (or use the command apropos) as follows:

```bash
man -k shutdown
```

or

```bash
apropos shutdo
```

manpage sections

Sometimes `man <keyword>` is used to get help of a command, but it's also used to get the help of a (config)file, daemon, ... This is an issue because `man <keyword>` needs to open the manpage you are looking for. What if the keyword exists as a command and as a configuration file? A perfect example for this is `passwd` :

```
student@linux-ess:~$ apropos passwd
...
passwd (1)           - change user password
passwd (1ssl)        - compute password hashes
passwd (5)           - the password file
```

Looking at the output above we see 3 `passwd` entries containing different numbers between the round brackets. The numbers refer to the *sections* of the manpage. By default, the `man` command will open the first section it finds. In this case *section 1*, which contains information about the command `passwd` . We can see that there is also a *section 5* which contains information about the configuration file `/etc/passwd` . We can open this section by specifying it as follows:

```bash
man 5 passwd
```

This command will now show the manpage of the configuration file rather than the `passwd` command.

Handy man page shortcuts

You can go to the first line by typing *g*. To go to the last line you will have to push capital *G*. To get help you can always push *h*.

Searching through a man page

You can search through a man page by typing a slash (/) followed by a string (word, letter,...) and then pushing the enter key. To go to the next occurence you can push the letter *n*. To go to the previous occurence you need to push the capital *N*.

> ℹ It's allways a good idea to first type *g* to go to the first line before starting your search!

whereis & whatis

We can quickly view the description of a command without opening the full manpage by using the `whatis` command:

```bash
student@linux-ess:~$ whatis route
route (8) - show / manipulate the IP routing table
```

To view the location of the manpage itself we can use the `whereis` command:

```bash
student@linux-ess:~$ whereis -m ls
whois: /usr/share/man/man1/ls.1.gz
```

The manpages are stored in archives with a `.gz` extention. This is comparable to a `zip` file containing text files. When typing the command `man ls` it will actually open the text file in the archive `/usr/share/man/man1/ls.1.gz` .

Shell history

A command line interface in Linux environments is called a *shell*. A shell often keeps track of all the commands we have used in the past. This means we can use this to easily repeat, edit or lookup previously used commands.

> Try using the `arrow up` and `arrow down` keys after using some commands. You will notice that these commands will reappear after the prompt.

Repeating the last command is very easy as well. We can type `!!` (often referred to as *bang bang*) and this will rerun the command that we last used. We often use it when we don't have enough privileges and we want to run the same command again with sudo:

```bash
student@linux-ess:~$ cat /etc/shadow
cat: /etc/shadow: Permission denied
student@linux-ess:~$ sudo !!
sudo cat /etc/shadow
[sudo] password for student:
root:*:19103:0:99999:7:::
daemon:*:19103:0:99999:7:::
bin:*:19103:0:99999:7:::
...
usbmux:*:19150:0:99999:7:::
student:$6$2YcjTQ1OiAVeexi5$MgKJ3MAZBx5P2ZfGIkIjbYYLtcPjxKBVAJx.RnuzPn.EJ
lxd:!:19150::::::
```

> Note that the shadow file holds the passwords of the users and is only viewable by root for security reasons!

To view the history of our last used commands we can use the `history` command:

bash

```
student@linux-ess:~/linuscraft$ history 10
  257  ls
  258  pwd
  259  cd
  260  cd linuscraft
  261  mkdir testfolder
  262  rm -rf testfolder
  263  pwd
  264  ls /
  265  ls -alh /
  266  history 10
```

The number that we use as an argument is the amount of commands the output will show. We can run any of these commands by using the identifier listed before the command as follows: `!n` . So for example running `!261` will run the command `mkdir testfolder` .

> Note that in some distros by default command lines that start with a space are not added to the history. If the command is identical to the previous command it is also witheld:

bash

```
student@linux-ess:~$ echo Start
Start
student@linux-ess:~$ echo "echo command not beginning with a space"
echo command not beginning with a space
student@linux-ess:~$ echo "echo command not beginning with a space"
echo command not beginning with a space
student@linux-ess:~$  echo "echo command beginning with a space"
echo command beginning with a space
student@linux-ess:~$ echo End
End
student@linux-ess:~$ history 4
  220  echo Start
  221  echo "echo command not beginning with a space"
  222  echo End
```

```
223  history 4
```

It's a good habbit to use `CTRL-R` to do a reverse search (newest to oldest) through your history. Just press `CTRL-R` and type your search string. Use `CTRL-R` again to search for the next command. Use the arrow keys to go into the command line and modify it. Use `CTRL-C` to quit and go back to an empty command line. Use `enter` to run the command.

Extra course material

Linux man pages guide

[Pluralsight] Using linux help resources

Previous
2 Installation

Next
Lab