KU LEUVEN

ICTS

Linux: an introduction

A cheat sheet approach

Preliminary version - September 2021

Content

1 Introduction

Some warning on the linux command line interface

- The command line interface is case-sensitive, one wrongly typed character can cause a lot of problems.
- There is no Recycle Bin, anything that you execute (run) in the command prompt would need to be fixed manually. Double Check EVERY command that you type, to make sure it is correct. The command line is a very powerful tool, and it requires a little more finesse and learning than the GUI (Graphic User Interface).

2 Linux

The Linux operating system is not a monolithic block fixed once and for all. In reality, many components work together, written by different people and assembled into distributions. It is only from the outside that the Linux kernel appears to be an indivisible unit. However, the distributions all have the same operating system kernel, and many common applications.

2.1 Kernel

The kernel is the part of the code in memory. The kernel takes care of :

- Memory management. Physical memory is extended virtually. Unused programs or program sections are offloaded to the hard disk and loaded into RAM when required for execution.
- File management. Linux has a hierarchical file system whose internal structure may vary. Disk space from other systems can be mounted.
- Device management. Access to devices is through files that form the interface between the device drivers and the kernel.
- Program and process management. Linux ensures that each program runs independently.
- Data access. Data stored in the file system must be protected from unauthorised access.

2.2 Shell

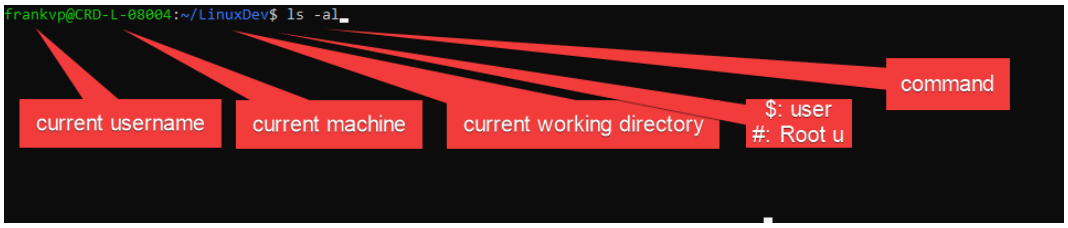
The shell is the most basic link between the user and the operating system. The commands entered on the command line will be interpreted by the shell and passed on to the operating system. There are several versions of the shell:

- **bash** This is the standard Linux shell the Bourne Again Shell.
- **sh** The original Bourne shell.
- **csh** The C shell uses a different programming interface to bash.
- **ksh** The Korn shell is one of the most popular shells on Unix. It is compatible with bash .
- **tcsh** The enhanced C shell.
- **zsh** A bash compatible shell.

2.3 Linux command

Telling the computer what to do is done by passing commands to the computer.

The anatomy of a command line interface is as follows:



A Linux command consists of 3 parts: the command itself, the command options, and its arguments.

```
command [OPTIONS prefixed with - or --] [ARG1] […ARGX]
```

3 Getting help

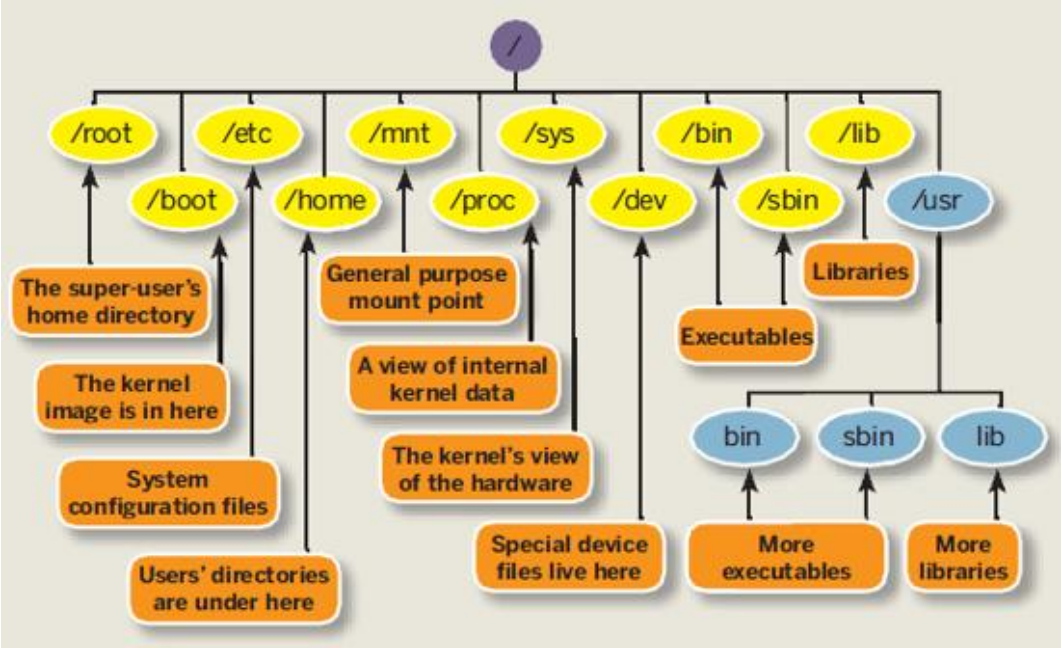
Source: https://www.howtogeek.com/108890/how-to-get-help-with-a-command-from-the-linux-terminal-8-tricks-for-beginners-pros-alike/

Command	Useful options	Common usage	What does it?
run the command with the -h or –help switches.	`-h` `--help`	`anycommand -h` `anycommand --help`	Show usage information and a list of options to be used with the command
Tab completion			When in doubt about about a specific command's name, an option, or a file name, use tab completion to help.
man		`man anycommand`	Loads manual page for the anycommand command The traditional documentation system
help		`help anycommand`	Shows help on the bash built-in commands
info		`info anycommand`	Another documentation system, behaving more like a digital book
apropos		`apropos keyword`	searches for man pages containing the keyword
whatis		`whatis anycommand`	a one-line summary of a command, taken from its man page.
type		`type anycommand`	It shows how the command would be interpreted when entered on the command line.

4 Working with files and directories

4.1 The file system

Common Linux directories



Source: http://linuxsuperuser07.blogspot.be/2011/09/rhel-6-file-system.html

directory	
/	The root directory contains all subdirectories.
/boot	When the system boots, the boot program will examine the /boot. Among the objects it looks for is the map file, by which the Linux boot manager will determine the location of the kernel on the hard disk.
/bin	The binaries directory contains the most important commands.
/dev	The device directory contains device files through which you communicate with the devices connected to the computer.
/etc	Only the configuration files should be there: passwd, group, hosts, etc.
/home	The users home directory will often be under the /home/yourname directory.The advantage of this is that the user will have their own file system.
/lib	Directory for shared libraries.

/opt	Directory for additional programs that are not part of Linux.
/root	The administrator's directory. Not to be confused with the root /.
/sbin	This is the directory for administrative commands.
/tmp	The temporary files directory. It is accessible for read and write.
/var	This is the directory for variable data in which Linux stores variable, rapidly or frequently changing data.
/usr	The sensitive data directory is a directory that contains a series of directories in which where Linux keeps very important data. Note the presence of: /usr/doc which contains the Linux documentation

4.2 `ls` command

Source: https://cheatography.com/davechild/cheat-sheets/linux-command-line/

Command	Options	Common usage	What does it?
ls	-a		Show all (including hidden)
ls	-R		Recursive list
ls	-r		Reverse order
ls	-t		Sort by last modified
ls	-S		Sort by file size
ls	-l		Long listing format
ls	-1		One file per line
ls	-m		Comma-separated output
ls	-Q		Quoted output

Tip:

- options can be combined
  ls -ltr: long listing, most recent files at the end

- use wildcards

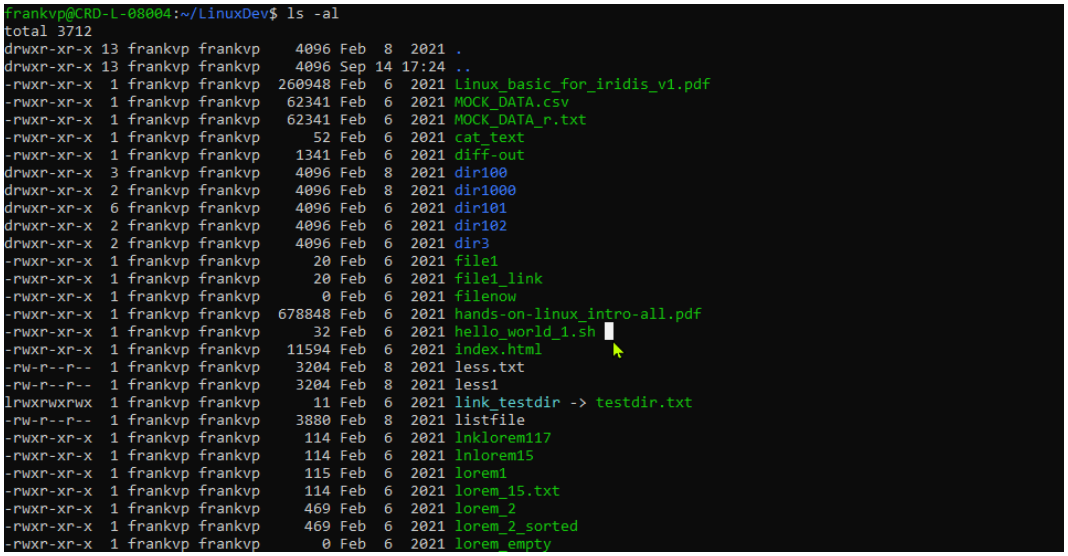
source: https://tldp.org/LDP/GNU-Linux-Tools-Summary/html/x11655.htm

Wildcard	Common usage	What does it ?
?	`hel?`	any *single* character (a-z, 0-9) .
*****	`hel*`	any number of characters (zero or more characters).

[ ]	`k[a,e,x]m`
	
	`k[f-h]m`
		specifies a range. This is an OR operation

[!]	`list[!9]`	similar to the [ ] construct, except rather than matching any characters inside the brackets, it'll match any character, as long as it is not listed between the [and]. This is a logical NOT.

{ }	`ls {*.doc,*.pdf}`
	
	`list anything ending with .doc or .pdf.`
		terms are separated by commas and each term must be the name of something or a wildcard.
		
		No spaces are not allowed after the commas.



Column	
1	File type + Permissions File type : (https://linuxconfig.org/identifying-file-types-in-linux) – : regular file d: directory c: character device file b: block device file s: local socket file p: named pipe l: symbolic link Permissions: User Group Other w: read w: write x: execute
2	#of links
3	Owner
4	Group
5	Size
6	Modification date and time
7	Name

4.3 Navigating the file system

4.3.1 Working with directories

A **directory**: a directory is just a file containing names of other files.

A **path** is a sequence of nested directories with a file or directory at the end, separated by the / character

Relative path: documents/fun/file1
Relative to the current directory

Absolute path: /home/user/leuven/file2

/ : root directory.
Start of absolute paths for all files on the system

Symbol	What does it mean?
.	Current directory
..	Parent directory
~	Home directory
/	Root directory

Command	Options	Common usage	What does it?
pwd		`pwd`	Print working (current) directory
cd		`cd dir`	Change directory Changes the current position to the specified directory
		`cd ..`	Go up one level (parent)
		`cd dir1/dir2`	Relative pathname, relative to the current directory

		`cd /dir1/dir2`	Absolute path, starting from the root
		`cd -`	Switch between current dir and previous dir
mkdir		`mkdir dir`	Make a directory dir. Relative and absolute path specification possible.
rmdir		`rmdir dir`	Remove a directory dir, this directory must be empty.

4.4 Viewing file content

4.4.1 Viewing commands

Command	Options	Common usage	What does it?
cat		`cat anyfile anotherfile`	Concatenate files and output
less		`less anyfile`	View and paginate anyfile
head		`head anyfile`	Show first 10 lines of anyfile
tail		`tail anyfile`	Show last 10 lines of anyfile

4.4.2 Less command

Source: https://linuxize.com/post/less-command-in-linux/

Action	What does it?
Down arrow, enter, e, j	Move forward one line.
Up arrow, y, k	Move backward one line.
Space bar, f	Move Forward one page.
b	Move Backward one page.
/pattern	Search forward for matching patterns.
?pattern	Search backward for matching patterns.
n	Repeat previous search.
N	Repeat previous search in reverse direction.
g	Go to the first line in the file.
G	Go to the last line in the file.
q	Exit less

4.5 File manipulation

Command	Options	Common usage	What does it?
touch		`touch anyfile`	Create anyfile
cp		`cp anyfile anotherfile`	Copy anyfile to anotherfile, overwriting if the file already exists or multiple sources to directory
	`-i`	`cp -i anyfile anotherfile`	Interactive option, prompts the user before overwriting the destination file

	`-v`		Verbose, displays what is being copied
	`-r`		
mv		`mv old new`	Move or rename files and directories
rm		`rm anyfile`	Delete anyfile

4.6 Search files

Command	Options	Common usage	What does it?
grep		`grep` *`pattern files`*	Search for *pattern* in *files*
		`grep -i`	Case insensitive search
		`grep -r`	Recursive search
		`grep -v`	Inverted search
		`grep -o`	Show matched part of file only
find		`find` *`/dir/`* `-name` *`name`*`*`	Find files starting with *name* in *dir*
		`find` *`/dir/`* `-user` *`name`*	Find files owned by *name* in *dir*
		`find` *`/dir/`* `-mmin` *`num`*	Find files modifed less than *num* minutes ago in *dir*
whereis		`whereis` *`command`*	Find binary / source / manual for *command*
locate		`locate` *`file`*	Find *file* (quick search of system index)
file		`file anyfile`	Get type of anyfile
stat		`stat anyfile`	

4.7 Archiving and compressing files

Tar stands for tape archive, the archive itself is a single file that can represent many files. tar files do not have to be compressed but they often are, even a not zipped tar file will often use less disk space than the files stored individually.

Command	Options	Common usage	What does it?
tar	-c		create a new archive.
	-x		extract an archive. The files will expand within the current directory.
	-t		verify or test; list the contents of the archive.
	-f		specifies the file name and must be followed by this name. Use the extension .tar to clearly indicate the file type.
	-k		prevents existing files being overwritten.
	-z		use gzip for compression and gunzip for decompressing the file.
	-j		uses bzip2 for compression and bunzip2 for decompressing.
	-r	`tar rf my_archive.tar example.txt`	Files or directories can be added/updated to the existing archives

	`--delete`	`tar –delete my_archive.tar workshop/file1`	To remove files/directories from an existing tar file.Make sure the path of the file/directory to delete is the same as the pathname displayed by tar -tvf

Recipes:

Explicitly archive and compress afterwards

1. Put all the files to archive in the same folder.
2. Create the tar file:
    a. `tar –cvf my_archive.tar workshop/` (archive the workshop directory within the current directory)
        i. -c : creates a .tar archive
        ii. -v : tells what is happening (verbose)
        iii. -f : assembles the archive into my_archive.tar
3. Once the tar file created, verify the file contents with the -t option
    a. `tar –tf my_archive.tar`
4. Compress the tar, create gzip file (most current)
    a. `gzip my_archive.tar`
    b. to decompress:
    `gunzip my_archive.tar.gz`

Archive and compress data the fast way

1. Compress with gzip:
    a. `tar –zcvf my_archive.tar.gz workshop/`
    b. decompress:
        i. `tar –zcvf my_archive.tar.gz  destination/`
2. Compress with bzip2:
    a. `tar –jcvf my_archive.tar.gz workshop/`
    b. decompress:
        i. `tar –jxvf archive.tar.bz2 destination/`

Extract a file from a tar file

1. Check with -t option the path of the file
2. `tar –xf my_archive.tar workshop/file1`

tip:

the content of a tar file may be read directly with `less` , without extracting the file.

4.8 Check disk space

Command	Options	Common usage	What does it?
du			Disk usage Be careful when running this command on directories with large amounts of data
	`-s`		Simplify the output
	`-h`	`du -sh`	Human readable output
		`du -sh directory`	Specify a directory
df			disk free, shows the amount of space taken up by different drives.
		`df -h`	Human readable output

4.9 Links

Source: https://ostechnix.com/explaining-soft-link-and-hard-link-in-linux-with-examples/

A symbolic or soft link is an actual link to the original file (similar to a shortcut in Windows), whereas a hard link is a mirror copy of the original file.

If you delete the original file, the soft link will be useless since it points to a non-existent file, the hard link will still have the data of the original file because it acts as a mirror copy of the original file.

Command	Options	Common usage	What does it?
ln	`-s`	`ln -s source_file link_2_file`	Create a soft (symbolic) link
		`ln source_file link_2_file`	Create a hard link

Tip:
what is the difference between Hard link and the normal copied file?

- If you copy a file, it will just duplicate the content. So if you modify the content of a one file (either original or hard link), it has no effect on the other one.
- However if you create a hard link to a file and change the content of either of the files, the change will be seen on both.

5 The shell revisited

Some useful features

5.1 Various commands

Source: https://cheatography.com/davechild/cheat-sheets/linux-command-line/

Command	Options	Common usage	What does it?
uname	`-a`		Show system and kernel
date			Show system date
uptime			Show uptime
whoami			Show your username
clear			Clear terminal
history			See your previously entered commands, the commands are saved to a file in your home directory (.bash_history) Use the up/down arrows to scroll through your previous commands
		`history | tail`	
		`history | grep keyword`	Search for a keyword in the commands

5.2 Bash shortcuts

Source: https://cheatography.com/davechild/cheat-sheets/linux-command-line/

Command	Options	Common usage	What does it?
CTRL-c			Stop current command
CTRL-r			Recall the last command matching the characters you provide
!!			Repeat last command
!*abc*			Run last command starting with *abc*
!*number*			Retrieve command on line *number* in the history

5.3 Bash variables

Source: https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-linux

Variables stores any user-defined or system-defined information that can be accessible within the shell.

- useful for passing data to programs or being used in shell scripts.
- Defining a variable is very simple (do not put spaces around = symbol)

Command	Options	Common usage	What does it?
env			Show environment variables
echo *$NAME*			Output value of *$NAME* variable
export *NAME=value*			Set *$NAME* to *value* export command promotes a shell variable to an environment variable

$PATH			Executable search path
$HOME			Home directory
$SHELL			Current shell

5.4 IO Redirection

Source: https://www.howtogeek.com/435903/what-are-stdin-stdout-and-stderr-on-linux/

`stdin`, `stdout`, and `stderr` are three data streams.

- stdin: standard input stream. This accepts text as its input.
- stdout: text output from the command to the shell is delivered via the stdout (standard out) stream.
- stderr: Error messages from the command are sent through the stderr (standard error) stream.

The > redirection symbol works with stdout by default. You can use one of the numeric file descriptors to indicate which standard output stream you wish to redirect.

Command	Options	Common usage	What does it?
>		*cmd* > *file*	redirect standard output of *cmd* to *file*
		cmd > /dev/null	Discard stdout of *cmd*
>>		*cmd* >> *file*	Append stdout to *file*
2>		*cmd* 2> *file*	Error output (stderr) of *cmd* to *file*
1>&2		*cmd* 1>&2	stdout to same place as stderr
2>&1		*cmd* 2>&1	stderr to same place as stdout
&>		*cmd* &> *file*	Every output of *cmd* to *file*
<		cmd < file	Input of *cmd* from *file*
		cmd1 <(*cmd2*)	Output of *cmd2* as file input to *cmd1*

5.5 Pipes

Pipe is used to pass output to another program or utility.

Redirect is used to pass output to either a file or stream.

Command	Options	Common usage	What does it?
cmd1 | cmd2		`ls | grep keyword`	stdout of *cmd1* piped to *cmd2*

Command lists

Command	Options	Common usage	What does it?
cmd1* ; *cmd2			Run *cmd1* then *cmd2*
cmd1* && *cmd2			Run *cmd2* if *cmd1* is successful
cmd1* || *cmd2			Run *cmd2* if *cmd1* is not successful
***cmd* &**			Run *cmd* in a subshell

6 Text editing

Nano is very much like ‘Notepad’ but without mouse support. All commands are executed through the use of the keyboard, using the Ctrl-key.

6.1 Nano Quick Reference

- ***Ctrl+X*** Exit the editor. If you’ve edited text without saving, you’ll be prompted as to whether you really want to exit.
- ***Ctrl+O*** Write (output) the current contents of the text buffer to a file. A filename prompt will appear; press Ctrl+T to open the file navigator shown above.
- ***Ctrl+R*** Read a text file into the current editing session. At the filename prompt, hit Ctrl+T for the file navigator.
- ***Ctrl+K*** Cut a line into the clipboard. You can press this repeatedly to copy multiple lines, which are then stored as one chunk.
- ***Ctrl+J*** Justify (fill out) a paragraph of text. By default, this reflows text to match the width of the editing window.
- ***Ctrl+U*** Uncut text, or rather, paste it from the clipboard. Note that after a Justify operation, this turns into unjustify.
- ***Ctrl+T*** Check spelling.
- ***Ctrl+W*** Find a word or phrase. At the prompt, use the cursor keys to go through previous search terms, or hit Ctrl+R to move into replace mode. Alternatively you can hit Ctrl+T to go to a specific line.

- ***Ctrl+C*** Show current line number and file information.
- ***Ctrl+G*** Get help; this provides information on navigating through files and common keyboard commands.

Manual: https://www.nano-editor.org/dist/v4/nano.pdf

Cheat sheet: https://www.nano-editor.org/dist/latest/cheatsheet.html

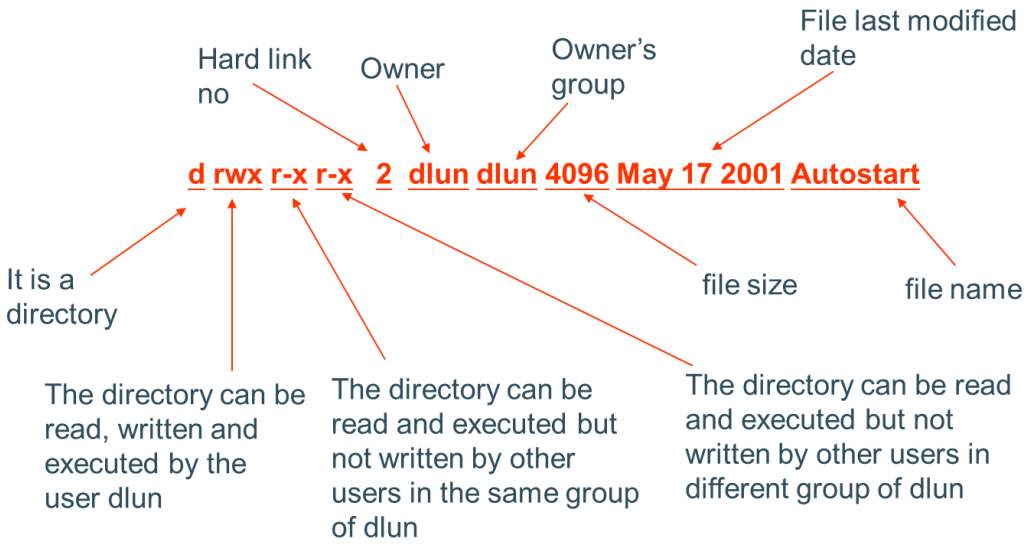
7 File security

Linux File Access permissions:

- Linux is a multiuser system, the files of all users are stored in a single file structure
- Mechanism is required to restrict one user to access the files of another user. User can impose access permission to each file to restrict its access

3 types of **access rights**

- Read access (r)
  - reading, opening, viewing, and copying the file is allowed
- Write access (w)
  - writing, changing, deleting, and saving the file is allowed
- Execute rights (x)
  - executing and invoking the file is allowed. This is required for directories to allow searching and access.



7.1 File Permissions

Command	Options	Common usage	What does it?
chmod		chmod 775 file	Change mode of *file* to 775
	-R	chmod -R 600 folder	Recursively chmod folder to 600
		chmod go+r	add read permissions to group and others.
		chmod u-w	remove write permissions from user.
		chmod a-x	remove execute permission from all (a: all).
chown		chown user:group file	Change file owner to user and group to group

2 formats for permissions:

- octal format (3 digit octal form)

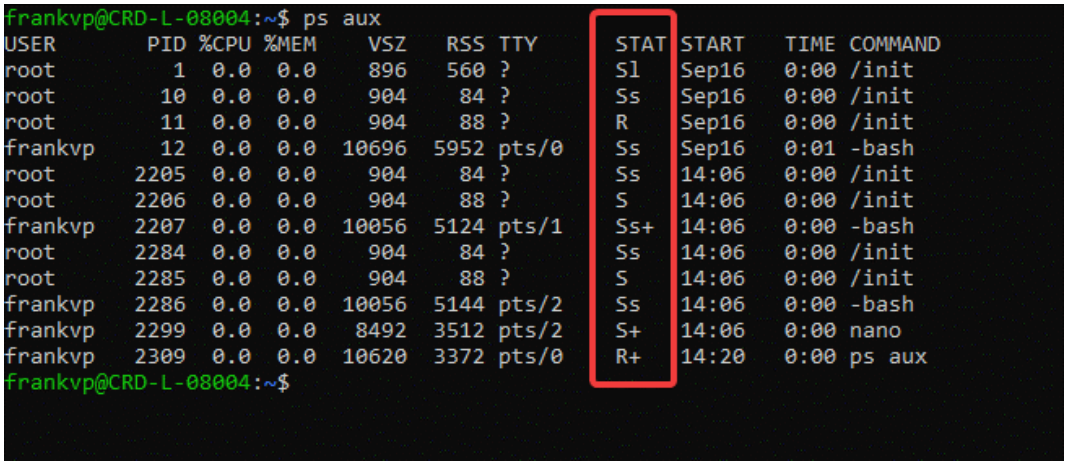
Calculate permission digits by adding numbers below.	
4	read (r)
2	write (w)
1	execute (x)

- symbolic format
    - action: r (read), w (write), x (execute
    - u (user), g (group), o (other), a (all)

8 Process Management

Processes

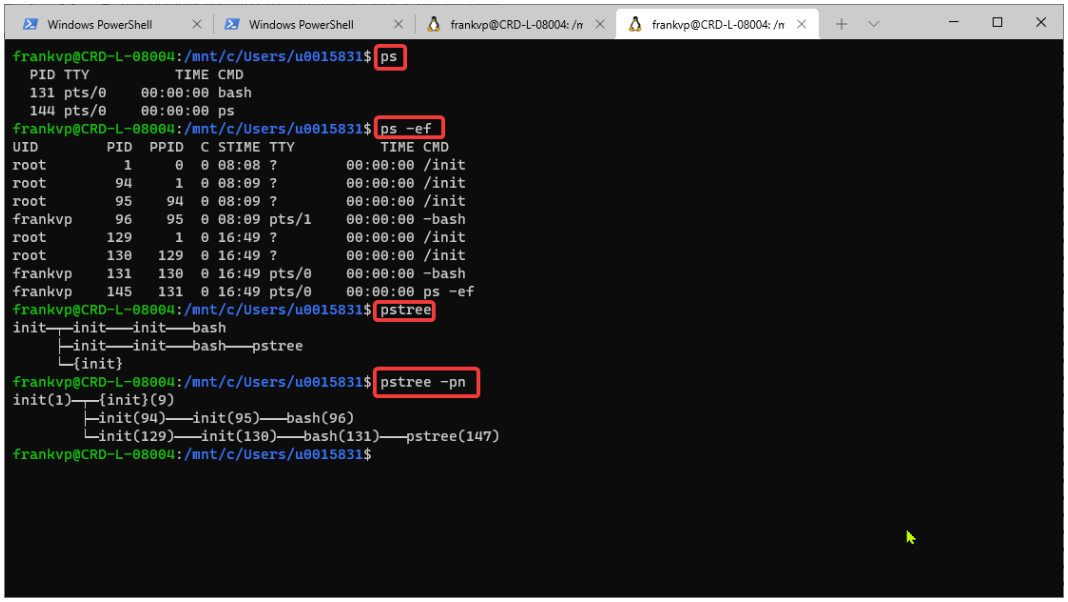
Source: https://linuxjourney.com/lesson/process-states



in the STAT column, you'll see lots of values. A linux process can be in a number of different states. The most common state codes you'll see are described below:

- R: running or runnable, it is just waiting for the CPU to process it
- S: Interruptible sleep, waiting for an event to complete, such as input from the terminal
- D: Uninterruptible sleep, processes that cannot be killed or interrupted with a signal, usually to make them go away you have to reboot or fix the issue
- Z: Zombie, we discussed in a previous lesson that zombies are terminated processes that are waiting to have their statuses collected
- T: Stopped, a process that has been suspended/stopped

Command	Options	Common usage	What does it?
ps			Process status, showing a snapshot of processes, By default, it will only show you the processes running in the local shell
		`ps -elf`	Display detailed information about running processes
		`ps -auxf`	a = show processes for all users u = display the process's user/owner x = also show processes not attached to a terminal f = forest view
pstree			Information presented in a forest view, less informative than ps -auxf
		`pstree -pn`	Show extra informatio
top			Show real time processes
kill		`kill` *`pid`*	Kill process with id *pid*
pkill		`pkill` *`name`*	Kill process with name *name*
killall		`killall` *`name`*	Kill all processes with names beginning *name*



9 Useful links

- William Shotts: The Linux Command Line - https://linuxcommand.org/tlcl.php
- The Linux Documentation Project - https://tldp.org/

UNIT
Street no. bus 0000
3000 LEUVEN, België
phone + 32 16 00 00 00
fax + 32 16 00 00 00
@kuleuven.be
www.kuleuven.be