Command line tutorial

Introduction to Linux

Introduction

Philosophy: Many small programs that work together efficiently, with as goal usability and maintainability

Linux is a kernel, the operating system is GNU. But when people say Linux, they mean Linux Kernel + GNU-utils, which are a set of programs that are very well known and widely used, such as `ls`. There are also other kernels such as the BSD kernel, with a similar set of tools comparable to the GNU-utils.

The Kernel manages:

- System hardware such as the CPU and other input/output (IO)
- File system
- Processes scheduling

File system overview

Folder	Description
`/`	Root of the file system
`/bin`	Contains programs (binaries)
`/usr/bin`	Non essential programs
`/etc`	Contains system config files
`/tmp`	Stores temporary files
`/home`	Contains personal data of users
`/home/patrick`	Patricks home folder

Key features of Linux

- Comes with a package manager which installs and updates software from trusted sources. Updating the system is very easy.
- Very flexible: you can install desktops, GUI's, window managers etc on top of linux
- Lightweight and very fast
- Free

The terminal and the shell

- The terminal is program that outputs text to user
- Inside the terminal runs a shell, which is its own program which has certain functionalities such as: variables, functions, auto-completion
- The shell is a program that runs as a REPL (Read Evaluate Print Loop, same as a console in R). It reads your input and evaluates it.
- Examples of shells are: csh, zsh, bash, fish, bash is the most common

Command line overview: the essentials

Some commands to start with

Command	Description
`whoami`	who am I
`w`	Who is doing what
`uname -a`	Shows OS info
`pwd`	Shows primary working directory

Moving on the command line

Command	Description
`cd`	Change directory to home folder
`cd ./somefolder`	Change directory to somefolder
`cd ~/somefolder`	Change directory to some folder from homefolder
`cd ../`	Change to the directory above the current directory
`cd ../../`	Change to 2 directories above the current directory
`mv  ./file1 ./file2`	Move files from A to B, also to rename files
`mkdir`	Create a directory
`rmdir`	Removes an empty directory
`cp  ./original ./copy`	copies a file
`cp -R ./folder ./copyOffolder`	copies a folder recursively
`rm ./file`	Removes a file
`rm ./file*`	Removes every file that starts with “file”
`rm -R ./folder`	Removes a folder recursively
`rm -Rf`	Removes a folder recursively with FORCE (useful for folders with git repo’s)

List files and directories

Command	Description
`ls`	List files in the current folder
`ls -l`	Show a longer list
`ls -a`	Show hidden files and folders
`ls -lah`	Add human readability
`ls -R`	list files Recusively recursively

File permission system

Overview of the `ls -lah` output

Command	Description
`chmod +x`	Makes a file executable
`sudo`	Execute a command as root user for full rights
`sudo vim /etc/hosts`	Execute a command as root user for full rights, useful for installing programs, editing files that are not yours etc.

Pipes

Extremely useful and often used. Can be used to chain programs together that are on PATH variable (see `echo $PATH`, run `env` to see all variables)

Command	Description
`ls -la | less`	View the output of the ls command in the less pager
`history | grep ssh | wc -l`	Count how many times the ssh program has been execute
`ls -R ~/ > bigFile.txt`	Stream the output of the less command to bigFile.txt
`cat bigFile.txt | wc -l`	Count the lines of bigFile

use `|` to stream the output of a program to the input of a program
use >> for appending to a file
use > for overwriting the file, it gives no warnings
use < for reading from a file (example `less < bigFile.txt`)

Useful commands

Command	Description
`man man`	Opens the manual of the man command
`less`	Pager, up/down j/k, seach with "/", quit with q
`top`	View all running processes
`top -o %CPU`	Sort by CPU usage
`top -o %MEM`	Sort by memory usage
`kill PID`	kill a process with PID (process ID) number
`pkill firefox`	kill all processes matching the name firefox
`echo "hi"`	outputs the string "hi" to standard out (Can be redirected using pipes)
`yes`	output a string repeatedly until killed
`Control + C`	Keycombination to press in order to kill a running program
`nano`	A "userfriendly" text editor
`ps -aux`	Outputs processes of all users
`grep`	Print lines that match patterns
`history`	Outputs a history of all run commands
`history | grep command`	Find the specifics of a command that matches the string "command"
`cat file1 file2`	Concatenates the output of 2 files, is often used to output a file to standard out
`head`	Shows the first 10 lines of a file
`tail`	Shows the last 10 lines of a file

`exit`	exit the current terminal, useful if you are going from one shell to the other
`clear`	clear the screen (Control + L)
`touch`	Change the modification date of a file, if file does not exist, create it
`touch file{1..10}`	Creates 10 files: file1 ... file10
`find . -name '*.pdf' -type f`	Find all pdf files in the current folder recusively
`file`	Determines the file type

Remote access

Command	Description
`ping www.google.nl`	See if you can reach a host (www.google.nl is translated to an ipaddr) from current shell
`ping ipaddr`	See if you can reach a host matching: ipaddr
`ping hostname`	See if you can reach a host matching: hostname, the hostname is linked to an ipaddr
`ssh 192.168.0.1`	Remote login to a host with ip adress: 192.168.0.1
`ssh user@192.168.0.1`	Remote login to a host with ip adress: 192.168.0.1 as the user: user

Bash and shell scripting with bash

Introduction

Bash is often the default shell, when you log onto a system. The config of bash is stored in ~/.bashrc, when starting up a terminal with bash as a shell, the contents in .bashrc are sourced. Most programs have similar config files.

Bash has its own programming language, which is similar to but not the same as the programming languages of other shells. When writing scripts comply to the POSIX standard whenever possible.

The combination of shell scripting and pipes is very powerful, you can write short and powerful programs with it. The power lies in the fact that you can create an new programs by combining other programs that are in the /bin directory (or one of the other directories on the PATH variable)

Use aliases to create your own shortcuts (run your shorcuts as if they are programs in the /bin folder). Aliases are very powerful, they can refer to scripts that do non-trivial things. Example: `alias tu='top -o %CPU'`

Example of shell script

Bash has its own scripting language, below is an example of a shell script:

```
#!/bin/bash

# The first line of the script tells the OS,
# that the script has the be execute with bash.
# Bash is also a program. Try executing bash inside bash.

# Define the variable COUNTER and set to 1
COUNTER=1

while true; do

    # Append a string to the file called counter.log
    echo "Dit is mijn counter: ""$COUNTER" >> counter.log

    # Add one to counter
    COUNTER=$((COUNTER+1))

    # Let the CPU sleep for half a second
    sleep 0.5;

done & echo "Process has been send to the background, please kill
        me! run: kill ""$!"

# Note: the ampersand sends a program to the background and frees up
        the terminal.
# Do not forget to kill it
```

To run this script, make it executable: `chmod +x ./myscript.sh`. Then you can run the program as follows: `user@host $ /path/to/myscript.sh`

See this overview for more an extensive overview of bash.

Tip: You can run shell commands from with `python` and `R` (In `R` this can be done with the `system()` function)