INTRODUCTION TO SUPERCOMPUTING

TMA4280 · Introduction to UNIX environment and tools

0.1 Getting started with the environment and the `bash` shell interpreter

Desktop computers are usually operated from a *graphical user interface* (GUI), but they have historically been interacted with by using a command interpreter: a sequence of commands is enter in a *terminal*. Since supercomputers do not run a graphical interface, As of November 2017, 100% of the supercomputers listed on the Top500 run a flavour of GNU/Linux system and they are exclusively operated without a GUI. The most common interactive command interpreter, a *shell*, is `bash`, the Bourne Again SHell. Other *shell* programs exist and offer their own scripting language: *csh*, *tcsh*, *ksh*, *zsh*...

The *shell* can be used interactively in a terminal and to run scripts: it is a very powerful tool which allows the user to interact with all the aspects of the system and perform tasks automatically.

This section is a basic introduction to the `UNIX` environment which covers:

- getting information from the system,
- navigating in directories,
- manipulating files,
- and processing text.

At first learning `shell` can seem difficult and may have a steep learning curve, but in the long term it provides an elegant tool to perform tasks in a very efficient way.

Some aspects of the interaction with the `bash` shell

The computer is operated by entering commands at the *prompt*: it is called a *command line*. By convention the *prompt* can be set to either:

1. $ for a normal user,
2. # for an administrator.

The behaviour of commands can be modified with flags,and arguments can be provided. For example, the following line can be entered:

```
$ <command> -f0 -f1 <argument0> <argument1>
```

to call the command with two flags `-f0`, `-f1`, and two arguments.

Interacting with the *bash* command line consists of:

- typing sequences of commands and arguments on the *command line*,
- auto-completion of arguments with the `TAB` key to expand the name of a command or file,
- navigation in the history of commands entered can be done with `UP/DOWN` arrow keys, or searched with `CTRL-R`,
- built-in functions that can be used for scripting.

All commands are documented and the manual pages can be displayed using the tools listed in Table 0.1.

Table 0.1: Basic commands: documentation

Command	Description
`info`	Display manual pages in the `GNU Texinfo` format.
`man`	Display **man**ual pages in the `troff` format.

The `man` utility is definitely the most important command on the system.

Basic commands to get information on the system

The *command line* offers several tools to get various information of the computer and the users. Before using commands that perform actions, it is a good idea to start with a few commands that will help you understand the environment.

Open a terminal and look at the information provided by the *prompt*, for example:

```
[aurelila@lille-login2 ~]$
```

and let us discuss the information presented. The first concept to understand is that:

- you are identified by a *user* on a system identified by his *hostname*,
- this *user* belongs to one or more *groups*,
- which define the *permissions* you possess on the system.

Table 0.2: Basic commands: information

Command	Description
`df`	Display status of **d**isk space on **f**ile systems.
`du`	Summarize **d**isk **u**sage.
`env`	Print **env**ironment variables.
`groups`	Print **group** membership of user.
`hostname`	Set or print **name** of current **host** system.
`top`	Display system usage information.
`uname`	Print **name** of current system.
`uptime`	Show how long the system has been **up**.
`which`	Which - locate a command and display its pathname or alias.
`who`	Who is logged on the system.
`whoami`	Print effective user identity.

Exercise 1. Before starting to navigate, let us inspect the work environment using the commands listed in Table 0.2.

Open a text editor and create an empty document that you will use to describe your steps and copy sample outputs from the terminal. From the graphical interface save the file as `unix.txt`.

1. Open a terminal and inspect information about your user and the system:
   - Print your user name, the machine name (`hostname`).
   - Use the manual page of `uname` to find how to print the machine name.
   - Find out to which groups your user belongs to.
2. Inspect the environment variables:
   - Display the environment variables and note the value of `SHELL`, `PATH` and `HOME`.
   - For each of these variables, type `echo $VARIABLE`, this is how to return the value of an environment variable.
   - Print the current working directory and observe that it is your home directory, i.e the value returned.
3. Find out more about resources:
   - Display the time elapsed since the machine was booted and inspect the resources (memory, processes, disk space).
   - Look at the `du` manual page and determine the current disk usage of your home directory.

Permissions

A set of *permissions* can be applied to files: in the UNIX philosophy, everything is represented as a file (text files, directories, devices). Any file is attributed to a *user* and a *group*.

The set is composed of three subsets describing permissions for:

1. *user* (**u**)
2. *group* (**g**)
3. *others* (**o**)

and is encoded using an octal triplet. For example, the triplet `644`, will give read/write permissions to the *user*, and only read permissions to the *group* and *others*.

Table 0.3: Permissions

Type	Description	Octal value
r	**R**ead from the file.	4
w	**W**rite to the file.	2
x	Execute the file.	1

Table 0.4: Basic commands: permissions

Command	Description
`chmod`	Change the permission set.
`chown`	Change the user ownership.
`chgrp`	Change the group ownership.

Exercise 2. Use the command `ls -lha` to list the files in the current working directory and understand the information printed.

Basic commands to navigate on the system

The set of commands essential to navigate in a *terminal* is small but provides many possibilities.

Exercise 3. Now that you are familiar with your system, let us navigate and create directories/files.

- Create a directory named `TMA4280` and move inside it.
- Create a directory named `shell` and move inside it.
- Find your `unix.txt` file and copy it to the current directory.

Table 0.5: Basic commands: navigation

Command	Description
`cd`	**C**hange working **d**irectory.
`cp`	**Cop**y files.
`find`	**Find** files in the directory hierarchy.
`ls`	**L**i**s**t contents of directory.
`mkdir`	**M**a**k**e **dir**ectories
`mv`	**M**o**v**e files.
`pwd`	**P**rint **w**orking **d**irectory name.
`rm, rmdir`	**Rem**ove (directory) entries.
`touch, settime`	Change file access and modification times.

- Move one level up (two options possible) and create a directory named `prog`.
- Move one level up again and type the command `ls -lha`, inspect the different folders.
- Open the manual pages of `chown`, `chgrp` and `chmod`: which one should you use to change the read permission of *others*?
- Create a `README` file using the command `touch`.

Table 0.6: Basic commands: text utilities

Command	Description
`awk`	Pattern scanning and processing language.
`cat`	Concatenate and display files.
`grep`	Search a file for a pattern.
`more, less`	Display text content with pagination.
`sed`	Stream editor for text manipulation.

Table 0.7: Basic commands: pipes and redirections

Command	Description
|	Pipe operator.
>	Redirection.
<	Redirection.

Exercise 4. Just introducing a few concept for text and file manipulation:

- Enter the `shell` directory and create a `cat` subdirectory.

- Create three files named `1.txt`, `2.txt` and `3.txt` and write the text `foo`, `bar` and `braz` into them using *echo*.
- Concatenate the content of the three files into the file `all.txt`.
- Display it in a pager.
- Together let us use the different utilities to search and replace text in these files: substitute `braz` with `babar`.