Tlemcen University

Department of computer science

1st Year Engineer

"Introduction to operating system 1"

Sessions 3 and 4: Basic Linux commands

Academic year: 2023-2024

Outline

- UNIX commands
  - *Command syntax*
  - *Basic commands*
- Redirection and pipe
  - *Input/output*
  - *I/O redirection*
  - *Error redirection*
  - *Pipes*

UNIX commands

- The general syntax of a command is :

`cmd_name[options] [[argument1][argument2] …]`

- *cmd_name: name of the command (in lower case)*
  - *options: these allow variants of the command. An option is usually preceded by a hyphen ('-').*
  - *arguments: generally the names of the objects targeted by the command.*
  - *The space separates the arguments. Exemple :*
- *Example:*
  - *ls -l Folder/*

  Display" in long format (with details) the contents of the directory (Folder)

- There are 2 types of commands:
  - *Internal commands: sub-programs of the command interpreter (shell).*
    - *They can be run directly without creating a child shell.*
    - *Examples: cd, echo, ...*
  - *External commands: executable files.*
    - *To execute them, you need to create a child process.*
    - *Examples: mkdir, mv, chmod, ...*

Basic commands

- The ***man*** command (help)
  - *The man command in Linux is used to display the user manual for any command we may run on the terminal.*
  - *Its syntax is :*

    `man [OPTION]... [COMMAND NAME]...`

    - *Without options, it displays the entire command manual*
    - *"q" to quit*
  - *Example of use:*
    - *man man (displays information on using man)*
    - *man ls (displays the full syntax of the ls command)*
- The ***pwd*** command (print working directory)
  - *Displays the current working directory*

- The ***ls (list)*** command
  - *The ls command lists the files and directories in the file system and displays detailed information about them.*
  - *Its syntax is :*

    ```
    ls [OPTIONS] [FILES]
    ```

  - *When used without options and arguments, ls displays a list of the names of all the files and directories in the current working directory.*
  - *To display the files in a list, use the -1 (one) option.*

    ls -1
  - *To list the files in such a directory, enter the directory name.*

    ls /home
  - *You can specify several directories in a row to list their contents*

    ls /etc /home

- *Format of the long list in the ls command (ls -l)*
  - *This displays precise information about the files in one or more directories*
  - *Example:*

    ls –l /etc
- *List only directories without files*

  ls -d */
- *To make the output human-readable, we add the -h option where file sizes are displayed in Kilobytes, Megabytes, ...*

  ls -lh
- *To display all the contents of the current directory, including hidden files.*

  ls –a
- *To list files in reverse alphabetical order*

  ls -1r

- *with the list of long formats :*

  ls –lr

- *If you prefer to sort folders separately and have them displayed before files, use the command :*

  ls -lr /etc/ --group-directories-first

- *You can sort the contents by file size, with the largest size listed first, using the -S option.*

  ls –S

- *By file modification date*

  ls –lht

- *By file extension*

  ls -lX

Directory commands

- The ***mkdir (make directory)*** command
  - *This command creates one or more directories*
  - *Its syntax is :*

    mkdir [OPTIONS] [DIRECTORY]
  - *To use the **mkdir** command, the user must have permission to create directories in the parent directory.*
  - *Create several directories*

    mkdir directory1 directory2 directory3
  - *Create sub-directories (tree structure)*
    - *The -p option can be used to create directories even if their parents do not exist.*

      mkdir -p directory1/directory2/directory3
    - *the "diretory1" directory will be created and even "directory2".*
  - *Define default permissions (umask) but you can bypass the umask*

    mkdir -m 777 directory4

- The ***cd (change directory)*** command
  - *cd allows you to navigate between directories*
  - *Its syntax is :*

    ```
    cd [options] [Directory_name]
    ```

  - *Change to a specific directory*

    cd /usr/bin

  - *You can also use the ~ (tilde) character directly. Instead of specifying the full path to home (/home/username)*

    cd ~

  - *To go to the user's home, simply enter the cd command with no arguments:*

    cd

  - *To go to the root folder represented by (/)*

    cd /

- *To go to the previous working directory :*

  cd –

- *To move to the parent directory :*

  cd ..

- *To change to a directory two levels above :*

  cd ../../

- The ***rmdir (Remove directory)*** command
  - *This command is used to remove empty folders.*
  - *Its syntax is :*

    rmdir [option] FolderName
  - *Deleting multiple directories (Dir1, Dir2 and Dir3)*

    rmdir Dir1 Dir2 Dir3
  - *The -p option allows you to delete a folder and its directories.*

    rmdir -p Dir1/Dir2/Dir3

File commands

- The ***touch*** command
  - *Allows you to create or modify the modification date of a file.*
  - *Its syntax :*

```
touch <options> <File>
```

    - *-a: Change the file access date*
    - *-d=<text>: Change the file date by specifying a timestamp*
    - *-c: Avoid creating a new file*

- The ***cp (Copy)*** command
  - *Its syntax is :*

    cp [OPTIONS] SOURCE... DESTINATION

    - *-a: Preserve as far as possible the structure and attributes of the original files in the copy*
    - *-f: This option forces the copy even if the destination folder is not available for writing.*
    - *-i : Display a message each time a file is to be overwritten.*
    - *-r or -R: Recursive tree copying.*
  - *-u: This option does not copy files that have an identical or more recent timestamp modification in the destination folder (this is an update of a copy).*
  - *If you wish to copy the file under a different name, you must specify the desired file name.*
    - *Example:*

      cp file1.txt /tmp/file2.txt

- The ***mv (move)*** command
  - *Moves a file or directory from one directory to another.*
  - *Syntax:*

```
mv [OPTIONS] Source Destination
```

    - *-b: Make a backup of each existing destination file*
    - *-f, -force: Don't ask for anything before overwriting*
    - *-n: Do not overwrite an existing file*
    - *-u: Move only when the source file is newer than the destination file or when the destination file is missing.*
  - *Examples:*
    - *mv file1.txt file2.txt (Move the file to the same location under a new name)*
    - *mv file.txt /tmp/ (Move file.txt to /tmp)*
    - *mv * destination/ (Move all files in the current directory to the "destination" directory)*

- The ***rm (Remove)*** command
  - *Allows you to remove a file or directory:*
  - *Its syntax is :*

    rm [OPTIONS]... FILE/DIRECTORY...

    - *rm -i file (interactively, with confirmation prompt)*
    - *rm -f file (forcefully, without confirmation prompt)*
    - *rm -r directory-not-empty (recursively, with subdirectories)*
    - *rm -rf folder (removes the directory and all its contents, without confirmation)*
    - *rm -d folder (removes empty directories)*
    - *rm file1.txt file2.txt file3.txt (rm accepts several filenames in a row to delete several files at once)*

Editing commands

- The ***cat*** command
  - *Displays the contents of a file*
  - *Its syntax is :*

```
cat [options] file(s)
```

    - *-n : activates line numbering*
    - *-nb: number even empty lines*
  - *The **tac** command reads the file in reverse order*

- The ***more*** command
  - *Displays the contents of a file page by page*
  - *Its syntax is :*

```
more [options] file
```

    - *-s: Groups consecutive blank lines into a single line.*
    - *-f: Logically counts the lines (rather than counting them on screen), i.e. long lines are not cut.*

- The ***less*** command
  - *less is a command that displays the contents of a file or command output, one page at a time.*
  - *It is similar to more, but has more advanced functionalities and allows us to navigate both forwards and backwards through the file.*
  - *Its syntax is :*

```
less [OPTIONS] filename
```

  - *less lets you browse a file and search for a string of characters. To see all the actions associated with less, simply type h.*
    - *move by line* ↓ ↑
    - *go to the first line (g or <)*
    - *go to the last line (G or >)*
    - *move down a page (space bar)*

- The ***echo*** command
  - *The echo command is one of the most basic and frequently used commands in Linux wherein "Arguments" passed to echo are displayed on the standard output.*
  - *Its syntax is as follows:*

```
echo [OPTIONS] [TEXTE]
```

  - ***Example :***

```
echo "Hello World "
```

  - ***Some OPTIONS:***
    - *-n : does not display the new end line*
    - *-e: is used to interpret backslash escapes*
    - *-E: disable backslash interpretation (default)*

- The echo command can also display the values of a variable. To do this, simply specify the variable without using single quotes:
- Examples :

```
echo $PWD or echo "$PWD"
```

- The command ***tr***
  - *tr is a command used to translate, convert and/or delete characters from standard input, written to standard output.*
  - *Its syntax is :*

```
tr [OPTION] SET1 [SET2]
```

  - *tr works with an input text that can be supplied with the echo command or the cat command, for example:*

```
echo text | tr [OPTION] SET1 [SET2]
cat file1 | tr [OPTION] SET1 [SET2]
```

  - *But you can also specify the file directly like this:*

```
tr [OPTION] SET1 [SET2] fichier.txt
```

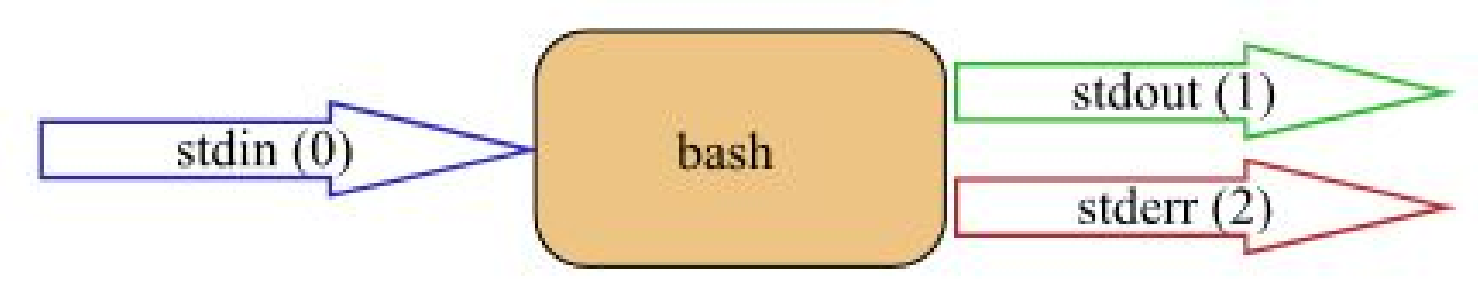
  - ***Example***

```
echo "tilekil.cot"|tr it am

malekal.com
```

Redirection and pipes

- **stdin, stdout, and stderr**
  - *The bash shell has three basic streams;*
    - *it takes input from **stdin** (stream **0**),*
    - *it sends output to **stdout** (stream **1**),*
    - *it sends error messages to **stderr** (stream **2**).*



- The keyboard is often used as stdin, while stdout and stderr both go to the screen.
  - *This can be confusing for new Linux users, as there is no clear way to distinguish stdout from stderr.*
  - *Experienced users will know that it can be very useful to separate error output.*
- **output redirection**
  - *stdout can be redirected using a greater than sign. When parsing the line, the shell will see the* **>** *sign and clear the file.*
    - *The > notation is in fact the abbreviation of* ***1>*** *(**stdout** being referred to as stream **1**).*
  - ***Example** :*

```
echo "Hello Students" > test.txt
```

❑ output file is erased

- *When the line is parsed, the command interpreter will see the > sign and will be clear the file.*
  - ➢ *This happens before argument 0 (the name of the command) is resolved, this means that even if the command fails, the file will have been cleared.*
- ***Example***

mohamed@mohamed-VirtualBox:~$ cat test.txt

Hello students

mohamed@mohamed-VirtualBox:~$ zcho Hello students > test.txt

-bash: zcho: command not found

mohamed@mohamed-VirtualBox:~$ cat test.txt

mohamed@mohamed-VirtualBox:~$

noclobber

- *Erasing a file when using > can be avoided by setting the **noclobber** option.*
- *Example :*

mohamed@mohamed-VirtualBox:~$cat test2.txt

Hello students

mohamed@mohamed-VirtualBox:~$ set -o noclobber

mohamed@mohamed-VirtualBox:~$ echo Hello students > test2.txt

-bash: test2.txt: cannot overwrite existing file

mohamed@mohamed-VirtualBox:~$ set +o noclobber

mohamed@mohamed-VirtualBox:~$

❑ overruling noclobber

- *The **noclobber** can be overruled by using the symbol >|.*
- ***Example :***

mohamed@mohamed-VirtualBox:~$ ***set -o noclobber***

mohamed@mohamed-VirtualBox:~$ ***echo Hello students! > test.txt***

-bash: test.txt: cannot overwrite existing file

mohamed@mohamed-VirtualBox:~$ ***echo Hello World >| test.txt***

mohamed@mohamed-VirtualBox:~$ ***cat test.txt***

Hello World

mohamed@mohamed-VirtualBox:~$

- **>> append**
  - ***Use >> to append output to a file***
  - ***Example:***

mohamed@mohamed-VirtualBox:~$ ***echo Hello students! > test.txt***

mohamed@mohamed-VirtualBox:~$ ***cat test.txt***

Hello students!

mohamed@mohamed-VirtualBox:~$ ***echo Hello World! >> test.txt***

mohamed@mohamed-VirtualBox:~$ ***cat test.txt***

Hello students!

Hello World!

mohamed@mohamed-VirtualBox:~$

error redirection

- **2> stderr**
  - *Redirection of stderr is done with 2>. This can be very useful for avoiding error messages cluttering up your screen.*
  - *The following command shows the redirection of stdout to a file and stderr to /dev/null. Writing 1> is identical to writing > :*

```
$find / > allfiles.txt 2> /dev/null
```

- **2>&1**
  - *To redirect stdout and stderr to the same file, we use 2>&1.*
  - ***Example:***

```
$find / > file_and_error.txt 2>&1
```

pipe

- A pipe is a form of redirection in Linux used to connect the **STDOUT** of one command into the **STDIN** of a second command.
- It allows us to narrow the output of a string of commands until we have an easily digestible amount of data.
- The pipe character is the | symbol and is placed between any two commands
- Example:

```
cmd1 arguments1  |  cmd2 qrguments2
```

```
echo "tilekil.cot"|tr it am
malekal.com
```

input redirection

- **< stdin**
  - *Redirecting **stdin** is done with < (short for 0<).*
  - *Example*

```
mohamed@mohamed-VirtualBox:~$ cat text.txt
one
two
mohamed@mohamed-VirtualBox:~$ tr 'onetw' 'ONEZZ' < text.txt
ONE
ZZO
```