Basics of Linux Shell

UManitoba Fall 2026
High-Performance Computing And Cloud
Workshop

Stefano Ansaloni

University of Manitoba

October 21, 2026





Stefano Ansaloni

Cloud Computing Specialist at University of Manitoba

Software Developer and DevOps Specialist since 2017

Linux User/Admin since 2005



What is a shell?

From Wikipedia (Unix Shell):

> *A Unix shell is a command-line interpreter or shell that provides a command line user interface (CLI) for Unix-like operating systems. The shell is both an interactive command language and a scripting language, and is used by the operating system to control the execution of the system using shell scripts.*

Or, simply put, the *shell* is a program that takes commands from the keyboard and gives them to the operating system to execute.

University of Manitoba

Common shells

Shells commonly shipped with Unix-like systems:

- Bourne Shell - **sh** (probably the most widely distributed and influential of the early Unix shells)
- Bourne-Again Shell - **bash** (usually the default interactive shell for users on most GNU/Linux systems)

University of Manitoba

What is a terminal?

A *terminal* (or *terminal emulator*) is a graphical program that allows you to interact with the shell.

University of Manitoba

Common terminals

Terminal emulators commonly shipped with GNU/Linux systems:

- Gnome Terminal (default terminal for the Gnome desktop environment)
- Konsole (default terminal for the KDE desktop environment)
- XTerm (generic terminal for the X window system)
- MobaXterm (enhanced terminal for Windows with X11 server and SSH client)
- MacOS Terminal (default terminal on MacOS)
- iTerm2 (enhanced terminal for MacOS)

University of Manitoba

Command prompt

The *prompt* (or *command prompt*) is issued by the shell, and while it is displayed, you can type commands.

Prompt examples:

- `user@hostname:/path/to/dir$`
- `[user@hostname dir]$`

University of Manitoba

Tab key

When typing commands inside a shell, you can use the *tab key* () to autocomplete commands.

For example: if you write "`ech`" and then press , the shell will complete the command to "`echo`".

(Some conditions apply)

University of Manitoba

Arrow keys

You can navigate the command history using the *up* and *down* arrow keys (↑, ↓).

This allows you to easily re-execute a previous command without retyping it.

University of Manitoba

Working directory

Usually the prompt informs you about the *current working directory* (or *current directory*, or *working directory*) using the `/path/to/dir` or `dir` string at the end of the prompt itself.

This is the directory where the shell is executing the commands you enter in the prompt.

You are **always** operating inside a *working directory*.

University of Manitoba

Home directory

The default working directory when you open a terminal, is your *home directory*.

Usually the *home directory* has the form "`/home/username`".

The *home directory* can be abbreviated using the tilde symbol "~".

Prompt example for user "goofy": `[goofy@hostname ~]$`

University of Manitoba

Special directories

On a GNU/Linux system every directory contains two special directories:

- `.` (single dot) – this is the current directory itself
- `..` (two dots) – this is the parent directory

University of Manitoba

Absolute paths

An *absolute path* is defined as specifying the location of a file or directory from the root directory "/" (it **always** starts with "/").

For example: `/home/goofy/file1.txt`

The specified path must exist on the current system.

University of Manitoba

Relative paths

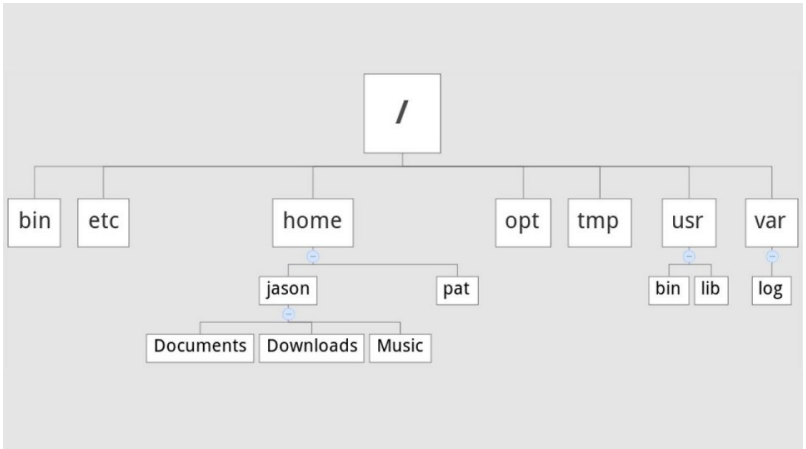
A *relative path* is defined as the path related to the current working directory and **never** starts with a "/".

For example: `goofy/file1.txt`

The specified path must exist inside the current directory.

University of Manitoba

Linux file system hierarchy



University of Manitoba

Execute a command

Commands and programs are regular files that reside inside some directories.

To execute a command, you can write its full path (absolute or relative to current directory).

You can invoke a command only by its name if it resides inside a directory included into the "`PATH`" variable.

You can execute multiple commands on a single row, by separating them with semicolon "`;`".

University of Manitoba

Commands manual pages

The commands **man** and **info** can be used to print respectively the manual pages (if present) and the info documents (if present) about a command.

```
[goofy@hostname ~]$ man pwd
```

University of Manitoba

Print current directory

The command **pwd** can be used to ask the shell to print the current working directory.

```
[goofy@hostname ~]$ pwd
/home/goofy
```

pwd will print the working directory as absolute path.

University of Manitoba

List directory content

The command **ls** can be used to list the files and directories inside a given directory (or the current directory if nothing is specified).

```
[goofy@hostname ~]$ ls
file1.txt
[goofy@hostname ~]$ ls /home/goofy
file1.txt
```

University of Manitoba

Command arguments

Shell commands often allow (or require) one or more *arguments* in order to be executed.

This is the case of *ls*, that allows you to optionally specify the directory path of which you want the content listed.

University of Manitoba

Command options

A special case for arguments are the command *options* (or *flags*), that are typically prefixed with one "-" or two "--" dashes.

Every command can have its own specific options, but usually a common one is the "`--help`" (or "`-h`") option that prints some information about the command itself.

For example: `[goofy@hostname ~]$ ls --help`

University of Manitoba

Change working directory

The command **cd** can be used to change working directory, navigating the filesystem.

```
[goofy@hostname ~]$ cd other_dir
[goofy@hostname other_dir]$ pwd
/home/goofy/other_dir
[goofy@hostname other_dir]$ cd
[goofy@hostname ~]$ pwd
/home/goofy
```

University of Manitoba

Copy files and directories

The command **cp** can be used to copy files and directories.

```
[goofy@hostname ~]$ cp file1.txt file2.txt
[goofy@hostname ~]$ ls
file1.txt    file2.txt    other_dir
[goofy@hostname ~]$ cp -R other_dir other_dir2
[goofy@hostname ~]$ ls
file1.txt  file2.txt  other_dir  other_dir2
```

University of Manitoba

Move files and directories

The command **mv** can be used to move (or rename) files and directories.

```
[goofy@hostname ~]$ mv file1.txt file3.txt
[goofy@hostname ~]$ ls
file2.txt  file3.txt  other_dir  other_dir2
[goofy@hostname ~]$ mv other_dir other_dir3
[goofy@hostname ~]$ ls
file2.txt  file3.txt  other_dir2  other_dir3
[goofy@hostname ~]$ mv file3.txt other_dir3
[goofy@hostname ~]$ ls
file2.txt  other_dir2  other_dir3
[goofy@hostname ~]$ ls other_dir3
file3.txt
```

University of Manitoba

Delete files and directories

The command **rm** can be used to delete files and directories.

The option "-R" is necessary when deleting directories.

```
[goofy@hostname ~]$ rm other_dir3/file3.txt
[goofy@hostname ~]$ ls other_dir3
[goofy@hostname ~]$ rm other_dir3
rm: cannot remove 'other_dir3': Is a directory
[goofy@hostname ~]$ rm -R other_dir3
[goofy@hostname ~]$ ls
file2.txt  other_dir2
```

University of Manitoba

Compress/Decompress files and directories

The command **tar** can be used to compress/decompress files and directories.

The options "-c", "-z" and "-f" are used to create a *tar archive* and compress it using the *gzip* format.

To extract files and directories from a *tar archive*, the options "-x" and "-f" are needed.

```
[goofy@hostname ~]$ tar -czf archive.tar.gz other_dir2
[goofy@hostname ~]$ rm -Rf other_dir2
[goofy@hostname ~]$ tar -xf archive.tar.gz
```

University of Manitoba

Print a message

The command **echo** can be used to print a message.

```
[goofy@hostname ~]$ echo "Some message"
Some message
```

University of Manitoba

Print file content

The command **cat** can be used to print the content of a file.

```
[goofy@hostname ~]$ cat file2.txt
Hello,
this is a test file.

Greetings
```

University of Manitoba

View file content in a pager

When a file contains a lot of lines, it could be more convenient to use a pager to visualize the file content.

The command **less** can be used to view the content of a file inside a pager.

A pager allows you to navigate the file content back and forth using the arrow keys (↑, ↓).

University of Manitoba

Filter file content

The command **grep** can be used to filter the content of a file.

```
[goofy@hostname ~]$ grep "lo" file2.txt
Hello,
[goofy@hostname ~]$ grep "is a" file2.txt
this is a test file.
```

University of Manitoba

Create empty file

The command **touch** can be used to create a new empty file (if the file already exists, its access and modification times are updated).

```
[goofy@hostname ~]$ touch file1.txt
[goofy@hostname ~]$ ls
file1.txt  file2.txt  other_dir2
[goofy@hostname ~]$ cat file1.txt
```

University of Manitoba

Create directory

The command **mkdir** can be used to create a new directory.

```
[goofy@hostname ~]$ mkdir other_dir1
[goofy@hostname ~]$ ls
file1.txt  file2.txt  other_dir1  other_dir2
[goofy@hostname ~]$ ls other_dir1
```

University of Manitoba

Print past commands

The command **history** can be used to print the list of previously executed commands (usually only the last 500-1000 commands are retained).

```
[goofy@hostname ~]$ history
-- REDACTED --
  134  grep "is a" file2.txt
  135  touch file1.txt
  136  ls
  137  cat file1.txt
  138  mkdir other_dir1
  139  ls
  140  ls other_dir1
  141  history
```

University of Manitoba

Quoting special characters

To create a file with spaces in its name, you must *quote* the file name, or *escape* the spaces.

To *quote*, you can use double or single quotes (", ').

Note that single quotes will preserve the literal value of all characters, while double quotes will preserve all characters except for $, `, \, and !.

To *escape*, you can use the backslash symbol ("\") to prepend each special character.

University of Manitoba

Quoting special characters

Example

```
[goofy@hostname ~]$ touch single file

[goofy@hostname ~]$ ls
file  file1.txt  file2.txt  other_dir2  single

[goofy@hostname ~]$ rm single file

[goofy@hostname ~]$ touch "single file"

[goofy@hostname ~]$ ls
file1.txt  file2.txt  other_dir2  'single file'
```

University of Manitoba

Print environment variables

The *environment* is a set of variables created by the system when a shell starts.

Each shell instance has its own environment.

The starting environment of each shell instance is the same.

The command **env** can be used to print the environment variables.

```
[goofy@hostname ~]$ env
-- REDACTED --
HOME=/home/goofy
LANG=en_US.UTF-8
PATH=/usr/local/bin:/usr/bin:/usr/local/sbin:
   /usr/sbin
PWD=/home/goofy
```

University of Manitoba

Variables scopes

Depending on how a variable is created, it can be available inside the current shell only, or inside the current shell and all its child processes created from that shell.

In the former case we talk about *shell variables*, in the latter *environment variables*.

University of Manitoba

Create shell variables

To create a *shell variable*, write the desired variable name, followed by the equal sign ("=") and the value you want to assign.

To avoid problems, only use alphanumeric characters for the variable name.

To access a variable content, use the dollar sign ("$") followed by the variable name.

```
[goofy@hostname ~]$ my_var="important value"
[goofy@hostname ~]$ echo $my_var
important value
[goofy@hostname ~]$ bash -c 'echo $my_var'
```

University of Manitoba

Create environment variables

The command **export** can be used to create an *environment variable* from a *shell variable*, or define a new one.

```
[goofy@hostname ~]$ export my_var
[goofy@hostname ~]$ export my_other_var="other important value"
[goofy@hostname ~]$ bash -c 'echo $my_var'
important value
[goofy@hostname ~]$ bash -c 'echo $my_other_var'
other important value
```

University of Manitoba

Text editors

The commands **vim** or **nano** can be used to open/edit text files directly inside the shell.

Some key bindings available inside each editor:

Editor	Enter Insert Mode	Exit Insert Mode	Save File	Exit Editor	Show Help
Vim	i	Esc	:w	:q	:h or F1
Nano			Ctrl + o	Ctrl + x	Ctrl + g

University of Manitoba

Scripting

A *script* (or *shell script*) is a text file that starts with a *shebang* ("`#!`"), followed by the shell interpreter ("`/bin/bash`"), and contains one or more commands (one per line).

Usually a *shell script* has the "`.sh`" extension.

```
[goofy@hostname ~]$ cat test_script.sh
#!/bin/bash
pwd
ls
```

University of Manitoba

Scripting

Conditional statements

You can decide to execute a command only when a certain condition is verified.

```
if <command_1> ; then
  echo "First condition verified"
elif <command_2> ; then
  echo "Second condition verified"
else
  echo "No conditions verified"
fi
```

University of Manitoba

Scripting

Loop statements

You can execute one or more commands multiple times using a loop.

The most used loop is the **for** loop.

```
for i in {1..10} ; do
  echo "$i"
done
```

University of Manitoba

Redirecting and piping

You can use a single or double greater-than symbol (“>”, “>>”) to redirect the output of a command to a file (the file will be created if it doesn't exist).

Using “>” will cause all the previous file contents to be deleted.
Using “>>” will cause the command output to be appended to the previous file contents.

You can use pipe symbol (“|”) to chain the output of a command to the input of a second command.

```
[goofy@hostname ~]$ ls > ls_output.txt
[goofy@hostname ~]$ ls >> ls_output.txt
[goofy@hostname ~]$ ls | grep er_d
other_dir2
```

University of Manitoba

Download files from websites

The command **wget** can be used to download files from websites through HTTP(S).

The option "-O" can be used to specify the filename for the downloaded data.

```
[goofy@hostname ~]$ wget https://www.kernel.org/pub/software/
    scm/git/git-2.49.0.tar.gz
```

University of Manitoba

Remote shell – SSH

The command **ssh** can be used to connect to a remote system that is running the *SSH server*.

After the connection, all the commands issued to the shell, will be executed on the remote host.

```
[goofy@hostname ~]$ ssh remotegoofy@remote-host
remotegoofy@remote-host's password:
[remotegoofy@remote-host ~]$
```

University of Manitoba

SSH keypairs

The command **ssh-keygen** can be used to create a *SSH keypair* that can be used for a password-less login on the remote SSH server.

The "`-t`" option can be used to select the key algorithm (usually `rsa`, `dsa`, `ecdsa` and `ed25519` are available).

The "`-f`" option can be used to specify the filename where to save the keypair.

```
[goofy@hostname ~]$ ssh-keygen -t ed25519 -f my_ssh_key
Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in my_ssh_key
Your public key has been saved in my_ssh_key.pub
```

University of Manitoba

Copying SSH public keys to remote hosts

The command **ssh-copy-id** can be used to copy a local SSH public key to a remote SSH server.

The "`-i`" option can be used to specify the SSH public key to copy.

```
[goofy@hostname ~]$ ssh-copy-id -i my_ssh_key.pub remotegoofy@remote-
    host
```

University of Manitoba

SSH login with private keys

You can use the "`-i`" option to specify the SSH private key to use to login on the remote SSH server.

```
[goofy@hostname ~]$ ssh -i my_ssh_key remotegoofy@remote-host
[remotegoofy@remote-host ~]$
```

University of Manitoba

Copying files from/to remote hosts

The command **scp** can be used to copy files from/to a remote SSH server.

The "-i" option can be used to specify the SSH private key to use for the login process.

```
[goofy@hostname ~]$ scp -i my_ssh_key
    remotegoofy@remote-host:file_from_remote
    file_to_local
[goofy@hostname ~]$ scp -i my_ssh_key
    file_from_local
    remotegoofy@remote-host:file_to_remote
```

University of Manitoba

Other commands

- File permissions & ownership: `chmod`, `chown`
- Compression: `gzip`, `bzip2`, `xz`, `zstd`
- Compression & print: `zcat`, `bzcat`, `xzcat`
- Text manipulation: `cut`, `sort`, `uniq`, `sed`
- Integrity check: `md5sum`, `sha256sum`, `sha512sum`

University of Manitoba

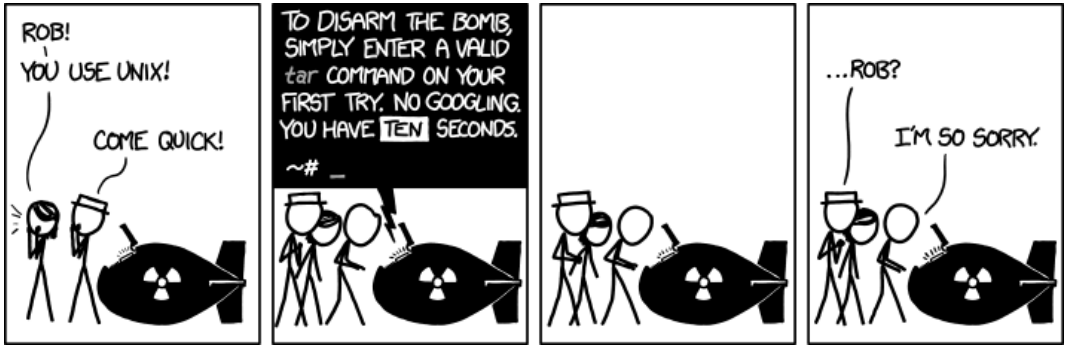
Final thoughts

The commands presented are only few of those available on a standard GNU/Linux system (the same applies to their options).

When exploring new commands, keep in mind the "`-h`" / "`--help`" options to print help information (and also the "`man`" command).

Avoid blindly running commands by copy–pasting them from some obscure internet webpage.

University of Manitoba



XKCD [1168]

University of Manitoba

Questions?

University of Manitoba

Thank you

