# LPI 105.1 - Customize and use the shell environment

Curs 2021 - 2022

ASIX M01-ISO 105 Shells and shell scripting

---



---

# Customize and use the shell environment

## Description

**Key concepts:**

- ❏ Set environment variables (e.g. PATH) at login or when spawning a new shell.
- ❏ Write Bash functions for frequently used sequences of commands.
- ❏ Maintain skeleton directories for new user accounts.
- ❏ Set command search path with the proper directory.

**Commands and files:**

- ❏ .
- ❏ source
- ❏ /etc/bash.bashrc
- ❏ /etc/profile
- ❏ env
- ❏ export
- ❏ set
- ❏ unset
- ❏ ~/.bash_profile
- ❏ ~/.bash_login
- ❏ ~/.profile
- ❏ ~/.bashrc
- ❏ ~/.bash_logout
- ❏ function
- ❏ alias

## Shell Variables

A variable is a name or identifier that can be assigned a value. Variable names should start with a letter (alpha character) or underscore and contain only letters, numbers, and the underscore character. Variable names are case-sensitive.

- local variables
- environment variables

A local variable is only available to the shell in which it was created. An environment variable is available to the shell in which it was created, and it is passed into all other commands/programs started by the shell. By convention, lowercase characters are used to

create local variable names, and uppercase characters are used when naming an environment variable.

Commands:

- set
- env (declare -x, typeset -x, export -p)
- export
- env  var command

- var=value
- var=”value with expansions”
- var=’literal’
- $var
- ${var}

- unset

```
#1
$ set | head
BASH=/usr/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquote:force_f
ignore:globasciiranges:histappend:interactive_comments:progcomp:promptvars:sourcepath
BASHRCSOURCED=Y
BASH_ALIASES=()
BASH_ARGC=([0]="0")
BASH_ARGV=()
BASH_CMDS=()
BASH_COMPLETION_VERSINFO=([0]="2" [1]="8")
BASH_LINENO=()
BASH_REMATCH=()

$ env | head
SHELL=/bin/bash
SESSION_MANAGER=local/unix:@/tmp/.ICE-unix/2055,unix/unix:/tmp/.ICE-unix/2055
COLORTERM=truecolor
HISTCONTROL=ignoredups
XDG_MENU_PREFIX=gnome-
HISTSIZE=1000
HOSTNAME=localhost.localdomain
EDIT=vim
LAN=C
GUESTFISH_OUTPUT=\e[0m
```

```
#2
$ age=25

$ name="pere pou prat"

$ msg="my home is $HOME"

$ literal='my uid is $UID'

$ echo $age $name $msg $literal
25 pere pou prat my home is /home/ecanet my uid is $UID
```

```
#3
$ age=23
$ export age

$ export name="pere pou prat"
```

```
$ env | grep age
age=23

$ env | grep name
name=pere pou prat

$ env TZ="Europe/Moscow" date
Mon Nov  1 19:19:14 MSK 2021

$ env TZ="Europe/Andorra" date
Mon Nov  1 17:19:20 CET 2021
```

```
#4
$ echo $age $name
23 pere pou prat

$ unset age

$ echo $age $name
pere pou prat
```

## PATH variable / Execute commands

Priority order:

- Aliases or functions
- Shell built-in commands
- External commands found in the PATH (in precedence order)

- Absolute (or relative) path to the command

The PATH variable contains a list of directories that are used to search for commands entered by the user. Processing works through the list of directories from left to right; the first executable file that matches what is typed is the command the shell will try to execute.

PATH changes affect only the session unless a permanent change in user's configuration files is made.

```
#5

$ which -a ls
alias ls='ls --color=auto'
        /usr/bin/ls

$ cd

$ /usr/bin/date

$ uptime

$ ./myprog
```

```
BASH_BUILTINS(1)                                          General Commands
Manual                                                BASH_BUILTINS(1)

NAME
```

```
bash, :, ., [, alias, bg, bind, break, builtin, caller, cd, command, compgen, complete, compopt, continue, declare,
dirs, disown, echo, enable, eval, exec, exit, export, false, fc, fg, getopts, hash, help, history, jobs, kill, let, local,
logout, mapfile, popd, printf, pushd, pwd, read,readonly, return, set, shift, shopt, source, suspend, test, times,
trap, true, type, typeset, ulimit, umask, unalias, unset, wait - bash built-in commands, see bash(1)
```

PATH directories

/sbin

Contains the essential administrative commands.

/usr/sbin

Contains the majority of the administrative command files.

/bin

Contains the most fundamental commands that are essential for the operating system to function.

/usr/bin

Contains the majority of the commands that are available for regular users to execute.

/usr/local/sbin

Normally empty, but may have administrative commands that have been compiled from local sources.

/usr/local/bin

Normally empty, but may have commands that have been compiled from local sources.

**Execute command not in the PATH**

To execute commands that are not contained in the directories that are listed in the PATH variable, several options exist:

- The command may be executed by typing the absolute path to the command.
- The command may be executed with a relative path to the command.
- The PATH variable can be set to include the directory where the command is located.
- The command can be relocated or copied to a directory that is listed in the PATH variable.

```
#6
$ pwd
/home/ecanet

$ mkdir bin
$ cd bin/

$ pwd
/home/ecanet/bin

$ cp /usr/bin/date mydate

$ /home/ecanet/bin/mydate
Mon 01 Nov 2021 05:38:09 PM CET
```

```
$ ./mydate
Mon 01 Nov 2021 05:38:58 PM CET
```

```
$ OLDPATH=$PATH

$ PATH=/home/ecanet/bin
```

```
$ mydate
Mon 01 Nov 2021 05:40:02 PM CET

$ uname -a
bash: uname: command not found...
```

```
$ PATH=$OLDPATH:$PATH

$ mydate
Mon 01 Nov 2021 05:41:10 PM CET

$ date
Mon 01 Nov 2021 05:41:13 PM CET

$ uname -a
Linux localhost.localdomain 5.11.22-100.fc32.x86_64 #1 SMP Wed May 19 18:58:25 UTC 2021
x86_64 x86_64 x86_64 GNU/Linux
```

```
$ PATH=/usr/share/Modules/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin

$ mydate
bash: mydate: command not found...

# cp /home/ecanet/bin/mydate /usr/bin

$ mydate
Mon Nov  1 17:45:00 CET 2021
```

```
#7
$ type cd
cd is a shell builtin

$ type echo
echo is a shell builtin

$ type ls
ls is aliased to `ls --color=auto'

$ type date
date is hashed (/home/ecanet/bin/date)

$ type uname
uname is /usr/bin/uname
```

Path priority

- from left to right.
- left: more priority:  PATH=new-dir:$PATH
- right: less priority:  PATH=$PATH:new-dir
- no PATH variable means no external orders by name, only by absolute path.

```
#8
$ echo $PATH
/usr/share/Modules/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin

$ which date
/usr/bin/date

$ cp /usr/bin/cal /home/ecanet/bin/date

$ echo $PATH
/usr/share/Modules/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin

$ OLDPATH=$PATH
```

```
$ PATH=$PATH:/home/ecanet/bin

$ echo $PATH
/usr/share/Modules/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/home/ecanet/bi
n

$ which -a date
/usr/bin/date
~/bin/date

$ date
Mon 01 Nov 2021 05:51:01 PM CET
```

```
$ PATH=/home/ecanet/bin:$OLDPATH

$ echo $PATH
/home/ecanet/bin:/usr/share/Modules/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbi
n

$ which -a date
~/bin/date
/usr/bin/date

$ date
    November 2021
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

**Prompt Variable**

- PS1
- PS2
- PS3

Note that changes made to the prompt only affect the current shell. Closing the shell and opening a new one will result in a return to the original prompt. To make changes to a variable permanent, define it in an initialization file.

Prompt values (man bash PROMPTING)

\a an ASCII bell character (07)

\d the date in "Weekday Month Date" format (e.g., "Tue May 26")

\D{format}

the format is passed to strftime(3) and the result is inserted into the prompt string; an empty format results in a locale-specific time representation. The braces are required

\e an ASCII escape character (033)

\h the hostname up to the first `.'

\H the hostname

\j the number of jobs currently managed by the shell

\l the basename of the shell's terminal device name

\n newline

\r carriage return
\s the name of the shell, the basename of $0 (the portion following the final slash)
\t the current time in 24-hour HH:MM:SS format
\T the current time in 12-hour HH:MM:SS format
\@ the current time in 12-hour am/pm format
\A the current time in 24-hour HH:MM format
\u the username of the current user
\v the version of bash (e.g., 2.00)
\V the release of bash, version + patch level (e.g., 2.00.0)
\w the current working directory, with $HOME abbreviated with a tilde (uses the value of the PROMPT_DIRTRIM variable)
\W the basename of the current working directory, with $HOME abbreviated with a tilde
\! the history number of this command
\# the command number of this command
\$ if the effective UID is 0, a #, otherwise a $
\nnn the character corresponding to the octal number nnn
\\ a backslash
\[ begin a sequence of non-printing characters, which could be used to embed a terminal control sequence into the prompt
\] end a sequence of non-printing characters

```
#9
[ecanet@localhost ~]$ echo $PS1
[\u@\h \W]\$


$ echo $PS2
>

$ echo $PS3


$ echo "my name
> is pere"
my name
is pere
```

```
#10
ecanet@localhost ~]$ PS1="$ "

$ echo $PS1
$

$ PS1="\w\$ "
~$

~$ PS1="\w\$ "

[~]$ PS1="[\w]\$ "

[~]$ cd /tmp/
[/tmp]$

[/tmp]$ PS1="[\u \W]\$"
[ecanet tmp]$
```

# Aliases

An alias is essentially a nickname for a command or series of commands. Some aliases may be created automatically as the result of commands executed in initialization shells. Aliases can also be avoided by using either an absolute or relative path to the command.

- alias
- alias name='command…'
- unalias name

Characteristics:

- By default in the current shell
- To be permanent should be included in the configuration files.
- To personalize a command (select options)
- To create new command

```
#11
$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias hola='echo hola'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mygit='# eval "$(ssh-agent -s)"; ssh-add ~/.ssh/KeySSH-m01\'
alias which='(alias; declare -f) | /usr/bin/which --tty-only --read-alias
--read-functions --show-tilde --show-dot'
alias xzegrep='xzegrep --color=auto'
alias xzfgrep='xzfgrep --color=auto'
alias xzgrep='xzgrep --color=auto'
alias zegrep='zegrep --color=auto'
alias zfgrep='zfgrep --color=auto'
alias zgrep='zgrep --color=auto'
```

```
#12

$ alias lh='ls -lh'

$ lh /usr/bin/date
-rwxr-xr-x. 1 root root 114K Feb  2  2021 /usr/bin/date
```

```
$ alias uname='uname -a'

$ uname
Linux localhost.localdomain 5.11.22-100.fc32.x86_64 #1 SMP Wed May 19 18:58:25 UTC 2021
x86_64 x86_64 x86_64 GNU/Linux

$ which -a uname
alias uname='uname -a'
       /usr/bin/uname
```

```
$ alias crazyday='whoami; date; id'

$ crazyday
ecanet
```

```
Mon 01 Nov 2021 06:12:51 PM CET
uid=1001(ecanet) gid=1001(ecanet) groups=1001(ecanet),18(dialout),974(docker)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

```
$ alias
alias crazyday='whoami; date; id'
alias lh='ls -lh'
alias uname='uname -a'
...

$ unalias uname

$ unalias lh
```

# Functions

Functions are a bit more advanced than aliases and typically are used in Bash shell scripts.

```
function_name ()
{
   commands_here
}
```

```
#13
$ set | grep function
    function is_valid_command ()
    function get_command ()
    function get_command_param ()
    function get_profile ()
    function get_command_keywords ()
    function get_command_options ()
    function get_global_options ()
    function get_option_params ()
    function get_command_params ()
```

```
#14
$ function hola() {
> echo "hola"
> date
> }


$ hola
hola
Mon 01 Nov 2021 06:18:26 PM CET
```

```
#15
$ function noms(){
> echo "nom: $1"
> echo "cognom: $2"
> }

$ noms pere pou
nom: pere
cognom: pou

$ unset noms
```

# Compound commands

- man bash: Compound Commands
- Lists
- &&
- ||
- ;

Operators:

;

The commands within the list are executed sequentially, where the shell will execute the first command and wait for it to terminate before executing the next command. The exit status of the list is based upon the exit status of the last command that is executed.

&

Each command within the list is executed asynchronously within a subshell, or in the background. The shell does not wait for the commands to terminate and returns an exit status of zero.

&&

This is an AND list, so if the command on the left side of && does execute successfully, then the command on the right side of && will execute. The exit status of the list is based upon the exit status of the last command that is executed.

||

This is an OR list, so if the command on the left side of || does NOT execute successfully, then the command on the right side of || is executed. The exit status of the list is based upon the exit status of the last command that is executed.

```
#16
$ date; who; uptime
Mon 01 Nov 2021 06:23:59 PM CET
ecanet   :0           2021-11-01 11:11 (:0)
 18:23:59 up  7:13,  1 user,  load average: 0.75, 0.64, 0.57


$ sleep 12345 &
[1] 21879

$ sleep 12345 & sleep 4444 &
[2] 21892
[3] 21893

$ mkdir /tmp/dir && echo "created OK"
created OK

$ mkdir /tmp/dir && echo "created OK"
mkdir: cannot create directory ‘/tmp/dir’: File exists

$ mkdir /tmp/dir || echo "ERROR"
mkdir: cannot create directory ‘/tmp/dir’: File exists
ERROR
```

## Initialization / logout files

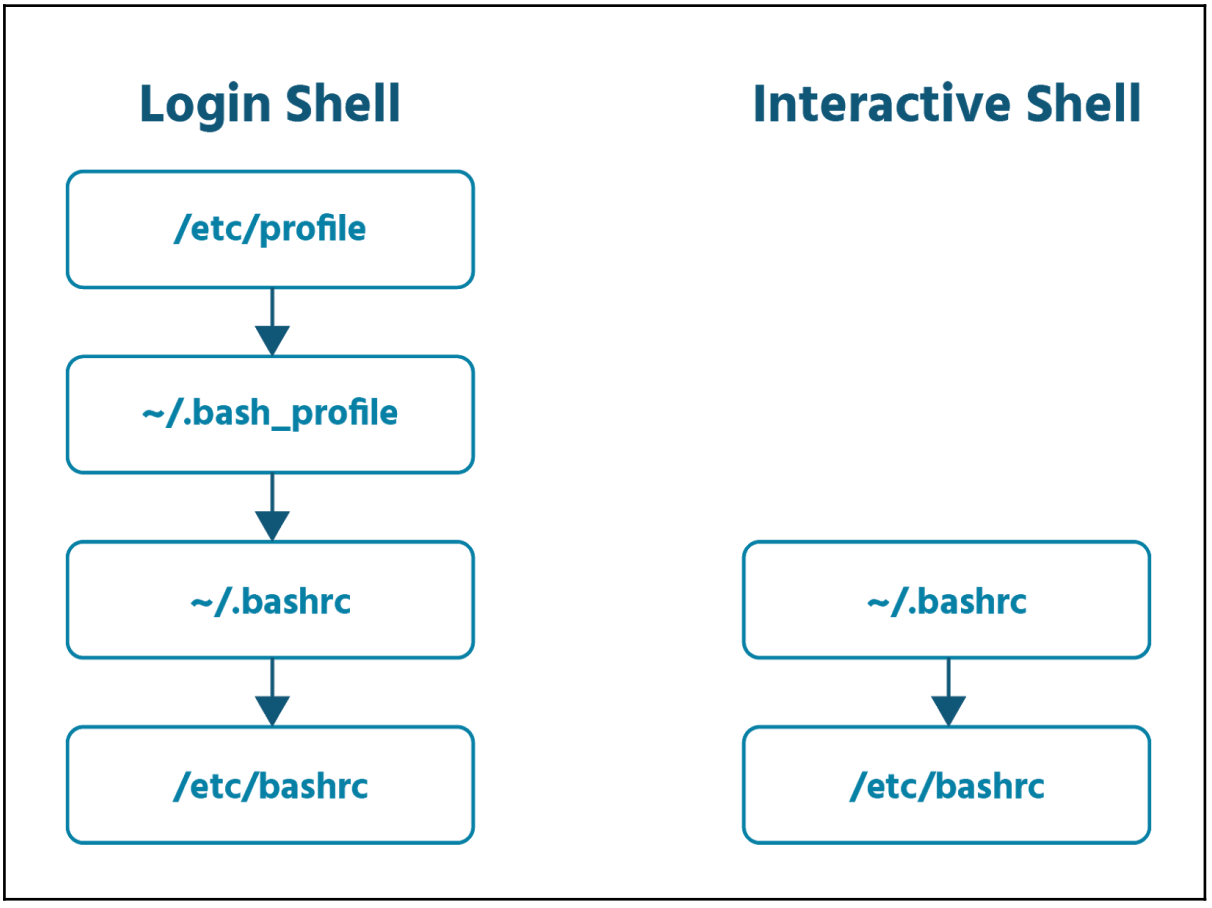
When a user opens a new shell, either during login or when they run a terminal that starts a shell, the shell is customized by files called initialization (or configuration) files. These initialization files set the value of variables, create aliases and functions, and execute other commands that are useful in starting the shell.

Type:

- global (system wide): /etc/profile /etc/bashrc /etc/logout
- user local: ~/.bash_profile ~/.bashrc ~/.bash_logout

There are two types of initialization files: global initialization files that affect all users on the system and local initialization files that are specific to an individual user.

Most shells execute different initialization files when the shell is started via the login process (called a login shell) versus when a shell is started by a terminal (called a non-login shell or an interactive shell) (for example graphical desktops).



When Bash is started as an interactive shell, it executes the ~/.bashrc file, which may also execute the /etc/bashrc file, if it exists. Again, since the ~/.bashrc file is owned by the user who is logging in, the user can prevent execution of the /etc/bashrc file.

Initialization files:

/etc/profile

This file can only be modified by the administrator and will be executed by every user who logs in. Administrators use this file to create key environment variables, display messages to users as they log in, and set key system values. System wide. Environment and startup programs.

~/.bash_profile ~/.profile

Each user has their own .bash_profile file in their home directory. The purpose of this file is the same as the /etc/profile file, but having this file allows a user to customize the shell to their own tastes. This file is typically used to create customized environment variables and startup programs.

~/.bashrc

Each user has their own .bashrc file in their home directory. The purpose of this file is to generate items that need to be created for each shell, such as local variables, aliases and functions.

/etc/bashrc /etc/bash.bashrc

This file may affect every user on the system. Only the administrator can modify this file. Like the .bashrc file, the purpose of this file is to generate items that need to be created for each shell, such as local variables, aliases and functions. system wide.

/etc/profile.d

This directory may contain scripts that get executed by /etc/profile.

```
#17

$ cat /etc/profile

$ cat /etc/profile | grep export
    export HISTCONTROL=ignoreboth
    export HISTCONTROL=ignoredups
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL


$ cat /etc/profile | grep umask
# By default, we want umask to get set. This sets it for login shell
    umask 002
    umask 022
```

```
$ cat ~/.bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/.local/bin:$HOME/bin
NAME="pere pou prat"
export PATH NAME
```

```
#18
$ head /etc/bashrc
# /etc/bashrc

# System wide functions and aliases
# Environment stuff goes in /etc/profile

# It's NOT a good idea to change this file unless you know what you
# are doing. It's much better to create a custom.sh shell script in
# /etc/profile.d/ to make custom changes to your environment, as this
# will prevent the need for merging in future updates.
```

```
$ cat ~/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
alias hola='echo hola'
alias lh='ls -lh'
```

```
#19
$ ls /etc/profile.d/
bash_completion.sh  colorls.csh      colorxzgrep.sh  csh.local   gawk.sh
lang.csh  less.sh      PackageKit.sh  sh.local  which2.csh
colorgrep.csh       colorls.sh       colorzgrep.csh  flatpak.sh  gmpopenh264.sh  lang.sh
modules.csh  scl-init.csh   vte.csh   which2.sh
colorgrep.sh        colorxzgrep.csh  colorzgrep.sh   gawk.csh    guestfish.sh
less.csh  modules.sh   scl-init.sh    vte.sh
```

Organization:

- Variables and startup programs:
    - /etc/profile (system wide)
    - ~/.bash_profile (user local)
- Aliases and functions
    - /etc/bashrc (system wide)
    - ~/.bashrc (user local)

Interactive (non login) Shells

This kind of shell refers to the interaction that takes place between the user and the shell: The user provides input by typing commands into the terminal using the keyboard; the shell provides output by printing messages on the screen.

Are non login shells when the user opens múltiples tabs in a graphical console terminal or when the user opens a new shell using the command `bash`.

Login Shells

This kind of shell refers to the event of a user accessing a computer system providing its credentials, such as username and password. The user can open a login shell also with the command `sudo login user`.

**Logout script**

- ~/.bash_logout
- /etc/bash_logout

  Just as Bash executes one or more files upon starting up, it also may execute one or more files upon exiting. As Bash exits, it will execute the ~/.bash_logout and /etc/bash_logout files, if they exist.

```
#20
$ cat ~/.bash_logout
# ~/.bash_logout
kdestroy
echo "sayonara baby"
```

When a new user is created, the files from the /etc/skel directory are automatically copied into the new user's home directory. As an administrator, you can modify the files in the /etc/skel directory to provide custom features to new users.

*Note*
source and . postponed to the next lesson

## Example Exercises

[variables]

1. Assign your name to the variable *name*, and show it.
2. Assign your age to the variable *age* and show it.
3. Enter in a subshell and show the variables *name* and *age*. Then exit the subshell.
4. Export the variables *name* and *age*.
5. Enter in a subshell and show the variables *name* and *age*.

[aliases]

6. Create an alias (uname) for *uname -a* and test it.
7. Create an alias (thisyear) for the calendar of 2022 and test it.
8. Create an alias called today with date, cal and uname. test it.
9. Delete the previous aliases.

[configuration files]

10. For all system users, configure the *class* variable with the value PUE and the variable city with the value Barcelona. In another session test it using `sudo login user`, and show the values.
11. For all system users, configure the alias *myself* (show the id and whoiam) and the alias *isday* (show the date and uptime). In another session test it using sudo login user, and show the alias.

12. Only for your user, configure the variable *name* with your name and the *city* variable with the value “Girona”. Start a non login shell with `bash` and show the variables: class, city and name.
13. Only for your user, configure the alias *processes* with the command “`ps ax | wc -l`”. Also redefine the alias *isday* showing the cal. Start a non login shell with *bash* and list the aliases.
14. Configure to run the *date* and *uptime* commands on every user login.

[path]

15. Create the /tmp/bin directory and copy the date command inside calling it *cal*. Copy also the cal command and call it *mycal*.
16. Executa mycal using an absolute path
17. Execute mycal using a relative path.
18. Configure the PATH to use /tmp/bin with less precedence.
19. Locate with *which -a* all the *cal* commands. Which has the most precedence?. Execute it.
20. Configure the PATH to use /tmp/bin with higher precedence.
21. Locate with *which -a* all the *cal* commands. Which has the most precedence?. Execute it.

22. Realitza els exercicis indicats a: 105.1 Customize and use the shell environment
23. Realitza els exercicis del Question-Topics 107.3.