Programación con Shell Scripting: Sesión 7

Máster Universitario en Bioinformática

Dra. Paula Soler Vila
paula.solerv@professor.universidadviu.com

De:
Planeta Formación y Universidades

Aspectos a tratar

1. Partes básicas de la estructura de un script

2. Elementos de la sintaxis del Shell scripting: **Variables**
   - Variables locales
   - Variables globales
     - Variables de entorno global
   - Operaciones aritméticas
   - Expansión de comandos

Partes básicas de un Shell script

Un script no es más que un archivo que contiene un conjunto de órdenes para realizar una tarea

```
#!/usr/bin/bash
```

script.sh

Shebang (#!)

 # Definición del intérprete

- La primera línea del script le indica al sistema que tiene que usar ***shell* BASH**

```
$ which bash
/usr/bin/bash
```

- El shebang no es obligatorio **Pero es recomendable usarla siempre**

Partes básicas de un Shell script

Un script no es más que un archivo que contiene un conjunto de órdenes para realizar una tarea

```
#!/usr/bin/bash

#Fecha de escritura: 07/05/2024
#Este programa te informa en que directorio se encuentra trabajando la terminal.
```

script.sh

Comentarios

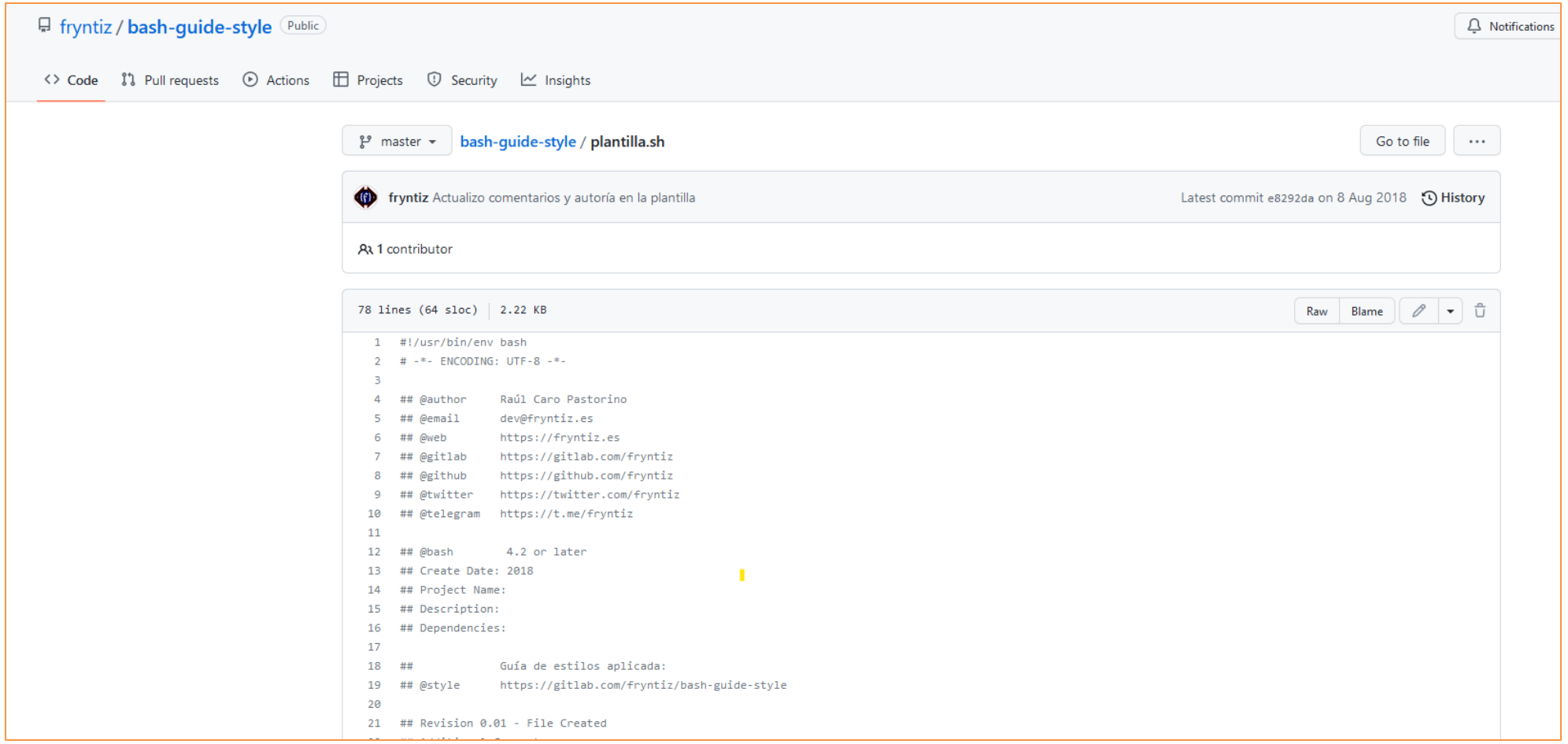
- Las siguientes líneas son **comentarios**
- Todo lo que está **después del carácter #** es **ignorado** por el interpretador de comandos **salvo la que tiene un #!, que esa no se ignora**

GUÍAS DE ESTILO

Partes básicas de un Shell script: Guías de estilo

- *The Google Shell Style Guide* The coding standard for scripts developed at Google. It's among the most sensible standards. https://google.github.io/styleguide/shellguide.html
- *Anyone Can Write Good Bash (with a little effort)* https://blog.yossarian.net/2020/01/23/Anybody-can-write-good-bash-with-a-little-effort
- *Some Bash coding conventions and good practices* https://github.com/icy/bash-coding-style
- *Bash Style Guide and Coding Standard* https://lug.fh-swf.de/vim/vim-bash/StyleGuideShell.en.pdf
- *Shell Script Standards* https://engineering.vokal.io/Systems/sh.md.html
- *Unix/Linux Shell Script Programming Conventions and Style* http://teaching.idallen.com/cst8177/13w/notes/000_script_style.html.

Partes básicas de un Shell script: Guías de estilo



https://github.com/fryntiz/bash-guide-style/blob/master/plantilla.sh

Partes básicas de un Shell script

Un script no es más que un archivo que contiene un conjunto de órdenes para realizar una tarea

```
#!/usr/bin/bash

#Fecha de escritura: 07/05/2024
#Este programa te informa en que directorio se encuentra trabajando la terminal.

lugar=$(pwd)
echo "En este momento te encuentras trabajando en este $lugar, no vayas a perderte"
```

script.sh

Bloque de código

- Comandos, funciones, variables...
- Orden secuencial -> Interpretación línea por línea

¿Qué es una variable?

Una **variable** es una «**ubicación**» en memoria donde se almacena un valor.

Crear
Modificar
Guardar
Eliminar

Naturaleza múltiple
Dinámicos

<nombre_variable>=<valor_variable>

Tenga en cuenta que no hay espacios entre el nombre de la variable, el signo igual o el valor asignado.

¿Qué es una variable?

Una **variable** es una «**ubicación**» en memoria donde se almacena un valor.

Crear
Modificar
Guardar
Eliminar

Naturaleza múltiple
Dinámicos

```
nombre=Juan
edad=25
ruta_archivos="/home/usuario/archivos locales"
```

la ruta se pone entre comillas porque tiene un espacio en blanco. Hay que ponerlo entre comillas cuando tiene carateres especiales

```
int main(){
    int var1 = 1;
    char var2 = 'a';
    float var3 = 13.5;
    double var4 = 11.3;
    bool var5 = true;

    return 0;
}
```

Tipos de variables

Pregunta examen

En el ***Shell*** de ***Bash*** podemos diferenciar dos tipos de variables:

- **Variables globales**

  Accesibles para **cualquier proceso en ejecución desde el *shell* o *desde* *cualquier* *sub-shell*** generada a partir de ella.

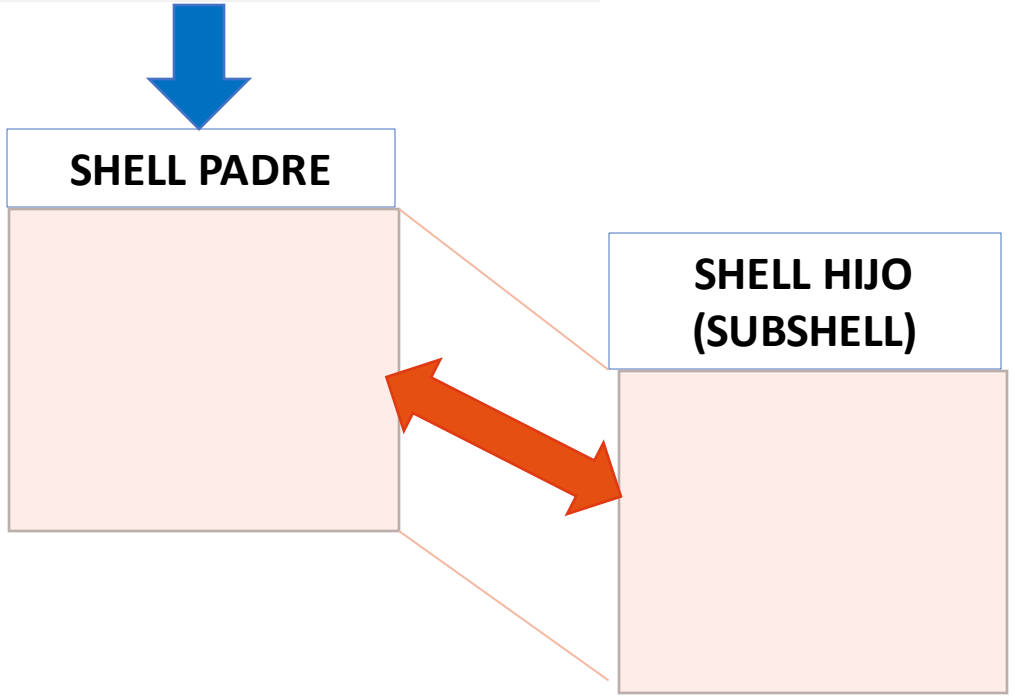
- **Variables locales**

  Accesibles **solo en el *shell* o *sub-shell* donde se definió.**

Shell vs subshell

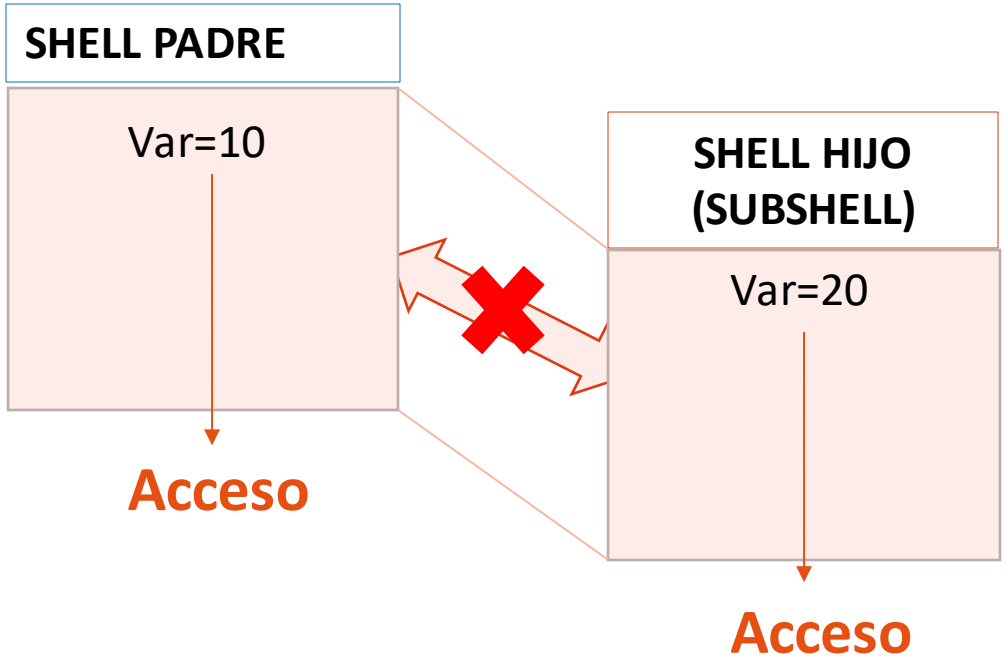
Shell inicial

Ejecuta cualquier comando simple



- Comandos agrupados *(ls;pwd;date)*
- Scripts de forma explícita
- Instrucciones en segundo plano

Variables locales



Creando *variables locales*

Para **declarar** una variable local solo debes escribir el **nombre de la variable** seguido de un **signo igual(=)** y el valor **SIN espacios en blanco.**

```
$ distro=centos

$ distro= centos
bash: centos: command not found
$ distro =centos
bash: distro: command not found
```

```
$ echo $distro
centos

$echo ${distro}
centos
echo ${distro}_v10
centos_v10
```

Estas son formas de acceder al contenido de una variable. En este caso la variable es DISTRO

Si se quiere añadir algún tipo de caracter asociado a la variable tiene que hacerse con esos corchetes, porque si no va a buscar una variable que se llame distro_v10 y no lo va a encontrar

Acceso a las *variables locales*

```
$
$ bioinfo=9.2
$ echo $bioinfo
$ 9.2
```

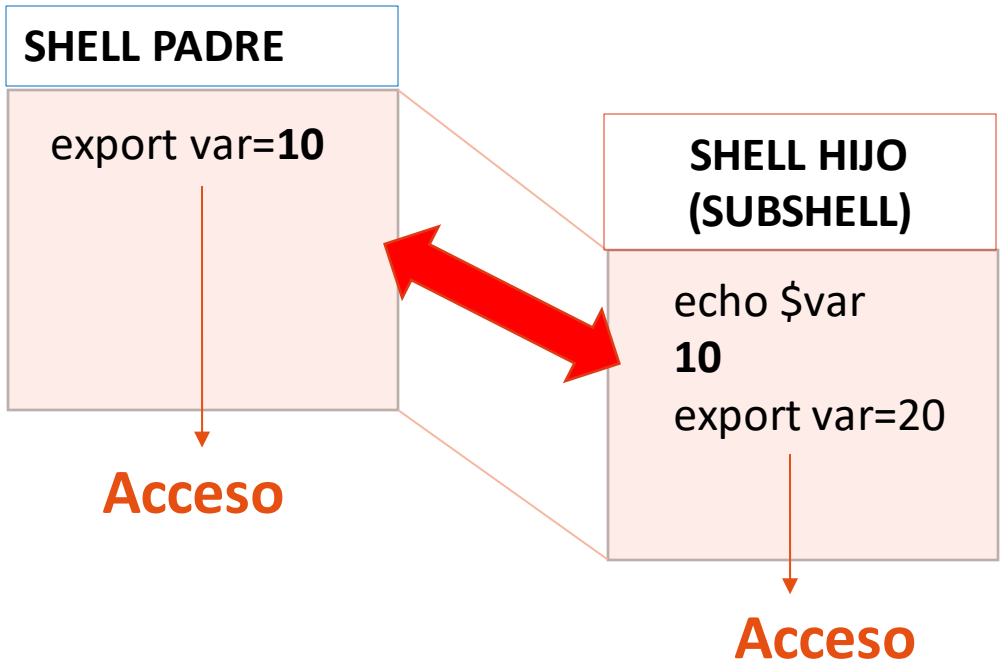
Shell principal

```
$ bash
$ echo $bioinfo

$ exit
exit

$ echo $bioinfo
9.2
```

Shell secundaria

(sub-Shell)

Variables globales



Creando *variables globales:*

export var=valor

- **export**: el comando utilizado para modificar el alcance de la variable.
- **var**: el nombre de la variable.
- **=** indica que la siguiente sección es el valor.
- **«valor»**: el valor real

para que sea una variable global y se pueda acceder desde otros sitios

```
$ distro=centos
$ export distro

$ export distro=centos
```

Acceso a las *variables globales*

```
$ export gen=CTCF
```

SHELL PRINCIPAL

```
$ bash
$ echo $gen
CTCF
$ gen=LCT
```

sub-shell -1

```
$ bash
$ echo $gen
LCT
$ exit
exit
```

sub-shell -2

```
$ echo $gen
LCT
$ exit
exit
```

sub-shell -1

```
$ echo $gen
CTCF
```

SHELL PRINCIPAL

Acceso a las *variables globales*

```
$ export gen=CTCF

$ bash
$ echo $gen
CTCF
$ gen=LCT

$ bash
$ echo $gen
LCT
$ exit
exit
$ echo $gen
LCT
$ exit
exit
$ echo $gen
CTCF
```

utilizar comando ps para ver donde estamos

1. Creación de subshells anidadas.
2. Después de exportar una variable global, permanece exportada a todas las subcapas creadas posteriormente.
3. Puede cambiar el valor de la variable exportada en una subcapa. El valor modificado se pasará a las subcapas posteriores, pero si sale y vuelve a la capa original, se conserva el valor original.

Comparando *variables locales* vs *globales (history)*

```
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ drink="tea"
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ echo $drink
tea
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ bash # Abriendo una sub-shell
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ echo $drink

[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ exit # ctrl + D
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ echo $drink
tea
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ export drink
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ bash
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ printenv drink
tea
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ exit
```

PREGUNTA DE EXAMEN

PRACTIQUEMOS

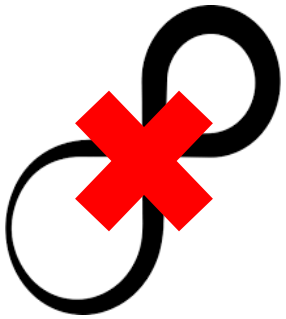
Eliminando *variables (globales y locales)*

unset nombre_variables

```
$ echo $distro
centos

$ unset distro
$ echo $distro
```

ojo aquí estamos eliminando solo el nombre de la variable porque no lleva el $ delante.

unset requiere el nombre de la variable como argumento. Por lo tanto, no puede agregar $ al nombre, ya que pasaría el valor de la variable a unset en lugar del nombre de la variable.



Las variables **no** son persistentes.

- Cuando se cierra la shell donde se establecieron, se pierden todas las variables y su contenido.
- Existen archivos de configuración que pueden modificarse para establecer estas variables como permanentemente.

Nomenclatura de las variables

- Sólo puede contener caracteres **alfanuméricos y guiones bajos** (_).
- El **primer carácter** debe ser una letra del **alfabeto** o un **guión bajo.**

```
$ distro=centos
$ _distro=centos
$ 1distro=centos
bash: 1distro=centos: command not found
```

- No pueden contener espacios en blanco.

```
$ my distro=centos
bash: my: command not found (base)
$ "my distro"=centos
 bash: my distro=centos: command not found
```

- Las **mayúsculas y las minúsculas importan**, “a” es distinto de “A”

Valores de las variables

- Puede contener cualquier carácter **alfanumérico,** así como la mayoría de los caracteres (?.!;*.etc)

```
$ distro=centos12.4?
$ echo $distro
centos12.4?
```

- Los valores de las variables deben ser encerrados **entre comillas** si contienen espacios en blanco

```
$ distro=centos12.4? 123
$ bash: 123: command not found
$ distro="centos12.4? 123"
$ echo $distro centos12.4? 123
$ distro='centos12.4? 123'
$ echo $distro centos12.4? 123
```

Aquí las comillas simples ‘ ‘ toman todos los valores literalmente, mientras que las comillas dobles “ “ van a permitir sustitución o expansión de la variable.

Valores de las variables

```
$ gen=SIX5
$ anotacion='Mi gen es $gen'
$ echo $anotacion
Mi gen es $gen

$ anotacion="Mi gen es $gen"
$ echo $anotacion
Mi gen es SIX5
```

Comillas dobles " "

- En este caso se hacen **Expansiones** (una sustitución de una variable por su valor)

Comillas simples ' '

- Todo texto entrecomillado con **comillas simples** mantendrá su valor literal, no se producirá ninguna expansión y será considerado como una única palabra.

Variables de entorno globales

- Las **variables de entorno globales se establecen** cuando **se inicia una sesión.**
- Por convención, se suelen definir en **letras mayúsculas** para diferenciarlas de las variables que define el usuario.

Variable	Descripción
DISPLAY	Donde aparecen las salidas de X-Windows
HOME	Directorio personal
HOSTNAME	Nombre de la máquina
MAIL	Archivo de correo
PATH	Lista de directorio en donde buscará programas y comandos
PS1	Prompt o indicador del shell que aparece en la línea de comandos
SHELL	Interprete de comandos por defecto
TERM	Tipo de terminal
USER	Nombre de usuario

¿Cómo conocemos el contenido de una variable de entorno global?

1. echo $HOME / echo ${HOME}

```
$ echo $HOME
/home/paula.soler

$ echo ${HOME}
/home/paula.soler
```

2. printenv HOME

```
$ printenv HOME
/home/paula.soler
```

```
$ printenv | head
XDG_VTNR=7
MATE_DESKTOP_SESSION_ID=this-is-deprecated
SSH_AGENT_PID=3117
XDG_SESSION_ID=11
```

Comando *printenv*

```
PRINTENV(1)                                User Commands                                PRINTENV(1)

NAME
       printenv - print all or part of environment

SYNOPSIS
       printenv [OPTION]... [VARIABLE]...

DESCRIPTION
       Print  the values of the specified environment VARIABLE(s).  If no VARIABLE is specified, print name and value pairs for them
       all.

       -0, --null
              end each output line with 0 byte rather than newline

       --help display this help and exit

       --version
              output version information and exit
```

PRACTIQUEMOS

VARIABLES NUMÉRICAS Y SU EXPANSIÓN

Operaciones aritméticas

$((expresion))
o
$[expresion]
O
`expr expresion`

```
(base) [UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ x=30
(base) [UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ y=15
(base) [UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$
(base) [UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ x + y
bash: x: command not found

(base) [UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ echo $[x+y]
45
(base) [UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ echo $((x+y))
45
(base) [UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ echo `expr $x + $y`
45
```

esto se llama acento grave, el de al lado de la tecla P

VARIABLES Y EXPANSIÓN DE COMANDOS

Expansión de comandos

La salida estándar de un comando se almacena en una variable.

Dos sintaxis diferentes:

$(comando)

Output=$(comando)

`comando`

Output=`comando`

Se **usa el acento grave**, que no debe confundirse con las comillas simples

El shell ejecuta el comando, captura su salida estándar y sustituye $(comando) o `comando` por la salida capturada.

Sustitución de comandos (*history*)

```
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ user=$(whoami)
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ echo "El usuario logeado es $user"
El usuario logeado es UNIVERSIDADVIU\paula.soler

[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ user=`whoami`
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ echo "El usuario logeado es $user"
El usuario logeado es UNIVERSIDADVIU\paula.soler

[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ files=$(ls -lh .)
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ echo "$files"
drwxr-xr-x 4 UNIVERSIDADVIU\paula.soler UNIVERSIDADVIU\domain users 4.0K Oct 23 09:21 comandos
-rw-r--r-- 1 UNIVERSIDADVIU\paula.soler UNIVERSIDADVIU\domain users  26K Oct 24 13:52 comandos.txt
-rw------- 1 UNIVERSIDADVIU\paula.soler UNIVERSIDADVIU\domain users 318M Jul 13 15:49 core.31938
-rw------- 1 UNIVERSIDADVIU\paula.soler UNIVERSIDADVIU\domain users 204M Aug 17 16:56 core.4554
-rw-r--r-- 1 UNIVERSIDADVIU\paula.soler UNIVERSIDADVIU\domain users   72 Oct 26 10:26 data.txt
...

[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ files=`ls -lh .`
[UNIVERSIDADVIU\paula.soler@a-3edhijmqygwxr ~]$ echo "$files"
total 415M
drwxr-xr-x 4 UNIVERSIDADVIU\paula.soler UNIVERSIDADVIU\domain users 4.0K Oct 23 09:21 comandos
-rw-r--r-- 1 UNIVERSIDADVIU\paula.soler UNIVERSIDADVIU\domain users  26K Oct 24 13:52 comandos.txt
-rw------- 1 UNIVERSIDADVIU\paula.soler UNIVERSIDADVIU\domain users 318M Jul 13 15:49 core.31938
-rw------- 1 UNIVERSIDADVIU\paula.soler UNIVERSIDADVIU\domain users 204M Aug 17 16:56 core.4554
-rw-r--r-- 1 UNIVERSIDADVIU\paula.soler UNIVERSIDADVIU\domain users   72 Oct 26 10:26 data.txt
```

Universidad
Internacional
de Valencia

universidadviu.com

De:
Planeta Formación y Universidades