



El sistema operativo UNIX

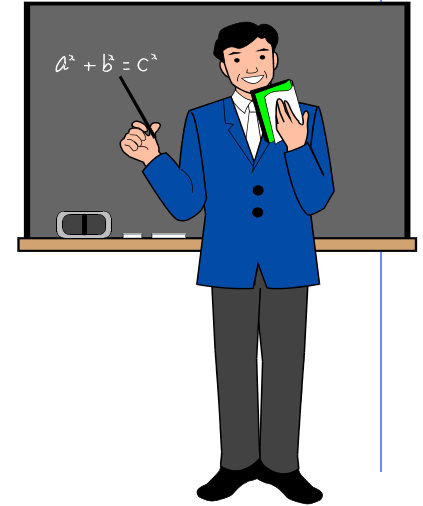
Introducción a la programación Shell I

Juan Carlos Yelmo

Contenidos

4. Programación en shell

- Introducción
- Programación en bash



Introducción

- ◆ La shell es un lenguaje de programación de alto nivel para escribir comandos nuevos o combinaciones frecuentes en base a comandos preexistentes
- ◆ Los comandos se combinan con estructuras de control y el uso de variables en ficheros denominados shell scripts
- ◆ Estos shell scripts son más cortos y sencillos que los programas convencionales pero menos eficientes

Introducción

- ◆ Cada una de las shells UNIX tiene comandos internos para permitir la programación de shells scripts
- ◆ Estos comandos internos y su sintaxis son diferentes en cada una de las shells
- ◆ Las shells más frecuentes para programación son sh y bash, siendo ksh, csh o tcsh más utilizadas en uso interactivo

Estructuras de programación en la shell

- ◆ Bucles
- ◆ Iteraciones
- ◆ Bifurcación
- ◆ Definición de funciones
- ◆ Evaluación de expresiones
- ◆ Manejo de entrada/salida
- ◆ Manejo de interrupciones y control de procesos

Cómo hacer un shell script

- ◆ Editar el fichero (e.g. script)
 - `vi script`
- ◆ Dar permiso de ejecución al fichero
 - `chmod +x script`
- ◆ Ejecutar y probar el script
 - `bash -x script [opciones][argumentos]`
- ◆ Usar el script
 - `script [opciones][argumentos]`

Argumentos de shell scripts

- ◆ Las variables posicionales de la shell se utilizan para pasar opciones y argumentos a un shell script
 - \$ <comando> <arg1> arg2> ...
- ◆ En el shell scripts estas cadenas se pueden utilizar mediante \$0 (el nombre del comando), \$1, \$2, ...
- ◆ Otras variables especiales
 - \$*: Todos los argumentos en un string
 - \$?: Valor de retorno del último comando ejecutado
 - \$#: Número de argumentos

Ejemplo sencillo

- ◆ Lista los 10 directorios más ocupados a partir de un path dado
 - gordos path

```
$ cat gordos
#!/bin/bash
# Lista directorios mas ocupados por debajo de $1

du $1 | sort -n | tail
$
```




El sistema operativo UNIX

Introducción a la programación Shell I

Fin del tema

Juan Carlos Yelmo



El sistema operativo UNIX

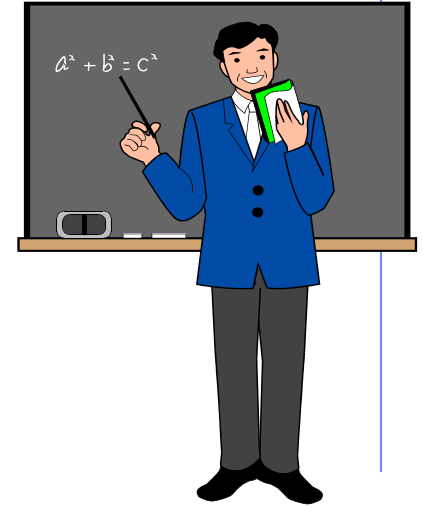
Introducción a la programación Shell II

Juan Carlos Yelmo

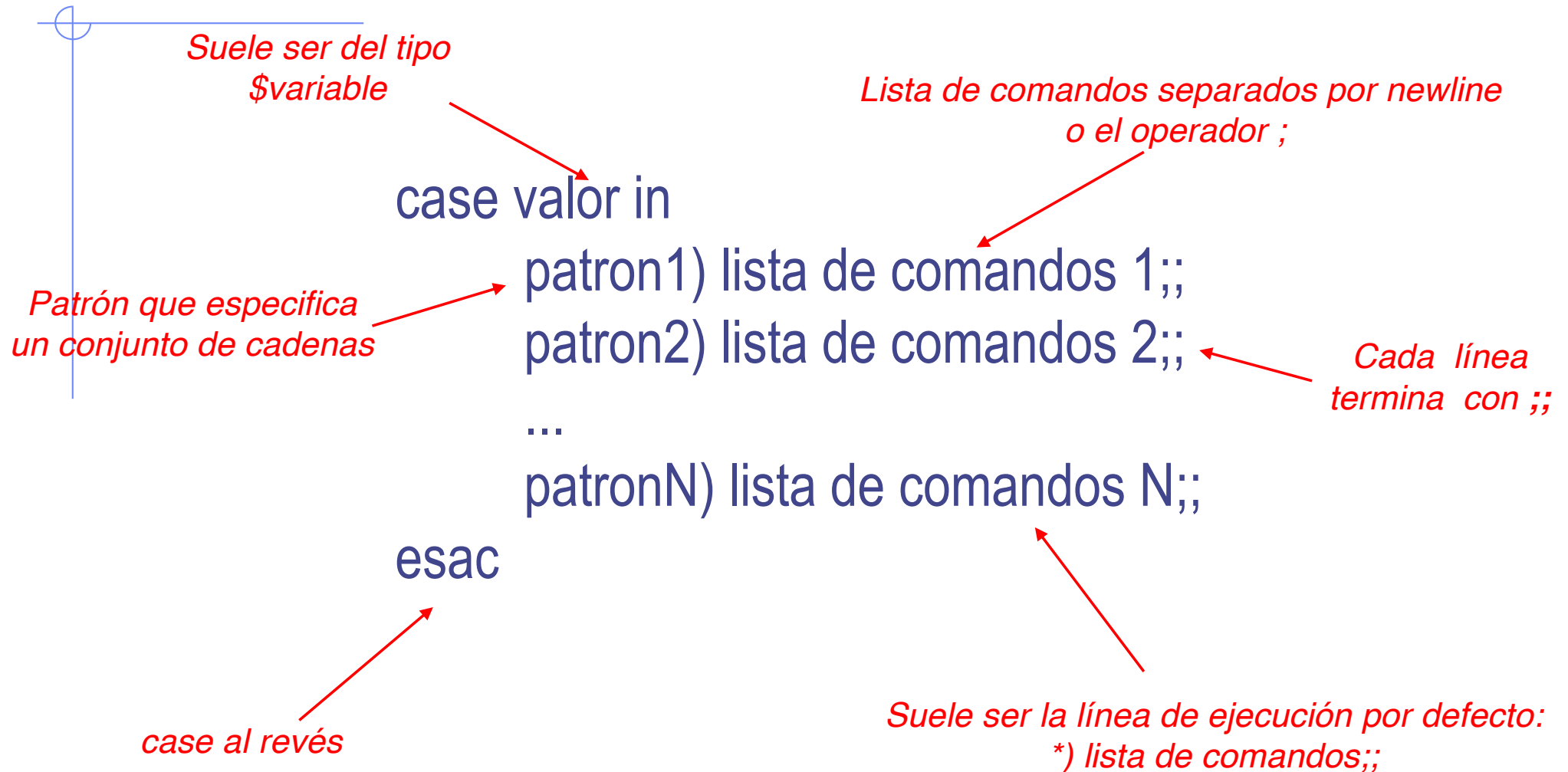
Contenidos

4. Programación en shell

- Bifurcaciones con `case` e `if`
- Comprobaciones con `test`
- Sustitución de variables

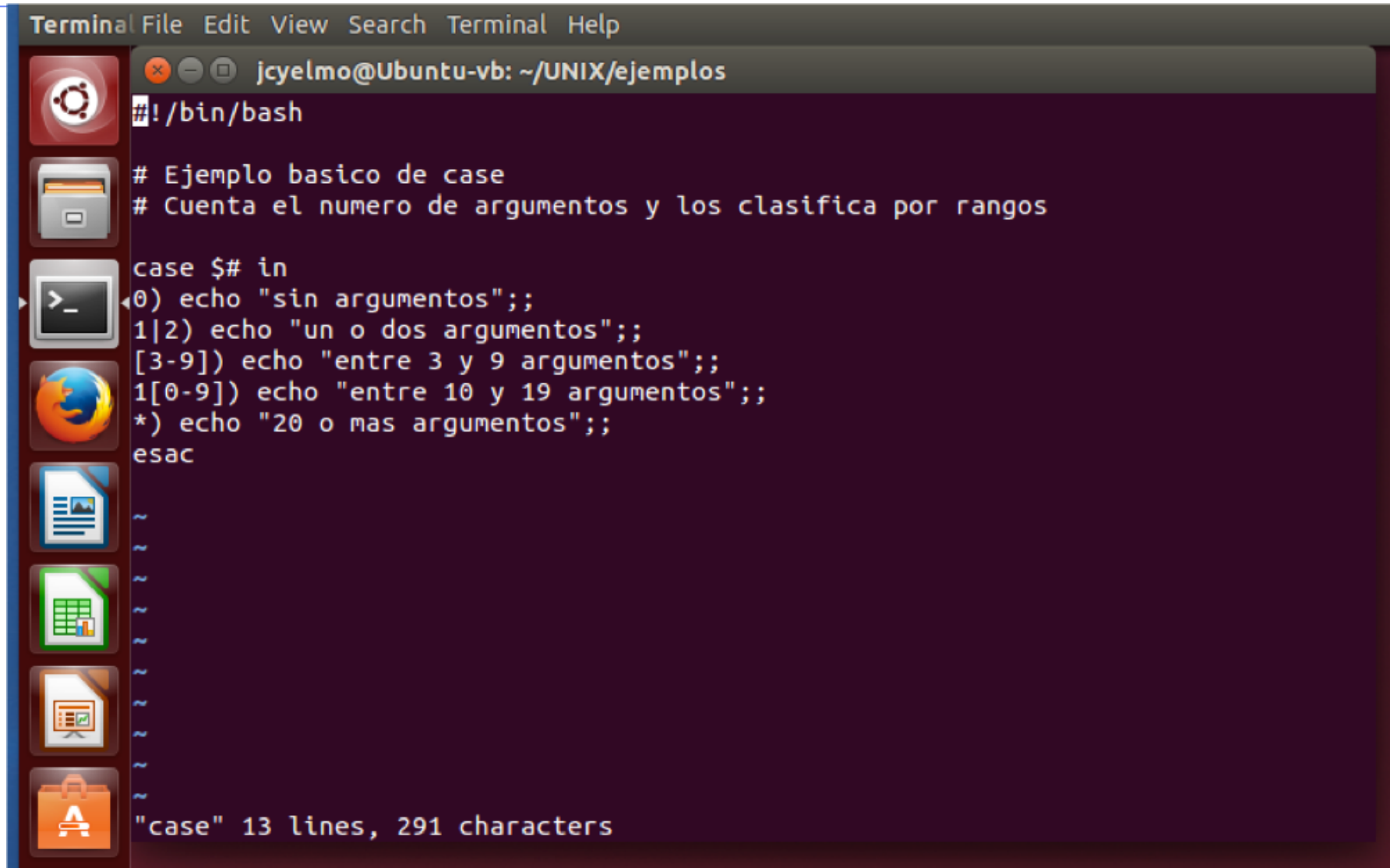


case



Patrones en bash y case

<code>*</code>	Coincide con cualquier cadena
<code>?</code>	Coincide con cualquier carácter
<code>[xyz]</code>	Uno de los caracteres de la enumeración
<code>[a-z]</code>	Uno de los caracteres del rango
<code>\c</code>	Coincide con <i>c</i> literalmente
<code>'cadena'</code>	Coincide con <i>cadena</i> literalmente
<code>"cadena"</code>	Coincide con <i>cadena</i> literalmente salvo variables de shell que se sustituyen por su valor
<code>patron1 patron2</code>	Uno cualquiera de los dos patrones. Sólo en case



if

```
If condicion
then
    lista de comandos 1
[else
    lista de comandos 2]
fi
```

*Comando.
Normalmente evaluación de
expresión con el comando test*

if al revés

if

If condicion1

then

lista de comandos 1

[elif condicion2

then

lista de comandos 2]

...

[else

lista de comandos N]

fi

Comando.

*Normalmente evaluación de
expresión con el comando test*

Línea de ejecución por defecto

if al revés

Comprobaciones con test

- ◆ Las comprobaciones y expresiones lógicas en shell scripts se realizan principalmente con el comando `test`
- ◆ El comando `test` devuelve cero en caso de evaluación a cierto y un valor distinto de cero en caso de falso.
- ◆ Dos sintaxis alternativas
 - `test condicion`
 - `[condicion]`

Espacio en blanco

test

test condicion | [condicion]

Evalúa la condición y si es cierta devuelve cero como valor de retorno. En caso de condición falsa devuelve un valor distinto de cero

Comprobaciones de ficheros

-d *fichero*

El fichero existe y es un directorio

-f *fichero*

El fichero existe y es un fichero ordinario

-r *fichero*

El fichero existe y tiene permiso de lectura

-s *fichero*

El fichero existe y tiene una longitud mayor de cero

-w *fichero*

El fichero existe y tiene permiso de escritura

-x *fichero*

El fichero existe y tiene permiso de ejecución

test

test condicion | [condicion]

Evalúa la condición y si es cierta devuelve cero como valor de retorno. En caso de condición falsa devuelve un valor distinto de cero

Comprobaciones con cadenas

cadena

cadena es no nula

-n cadena

La cadena es de longitud mayor que cero

-z cadena

La cadena es de longitud cero

cadena1 = cadena2

Las dos cadenas son idénticas

cadena1 != cadena2

Las dos cadenas no son idénticas

test

test condicion | [condicion]

Evalúa la condición y si es cierta devuelve cero como valor de retorno. En caso de condición falsa devuelve un valor distinto de cero

Comprobaciones con enteros

n1 -eq n2

n1 es igual a *n2*

n1 -neq n2

n1 no es igual a *n2*

n1 -gt n2

n1 es mayor que *n2*

n1 -ge n2

n1 es mayor o igual que *n2*

n1 -lt n2

n1 es menor que *n2*

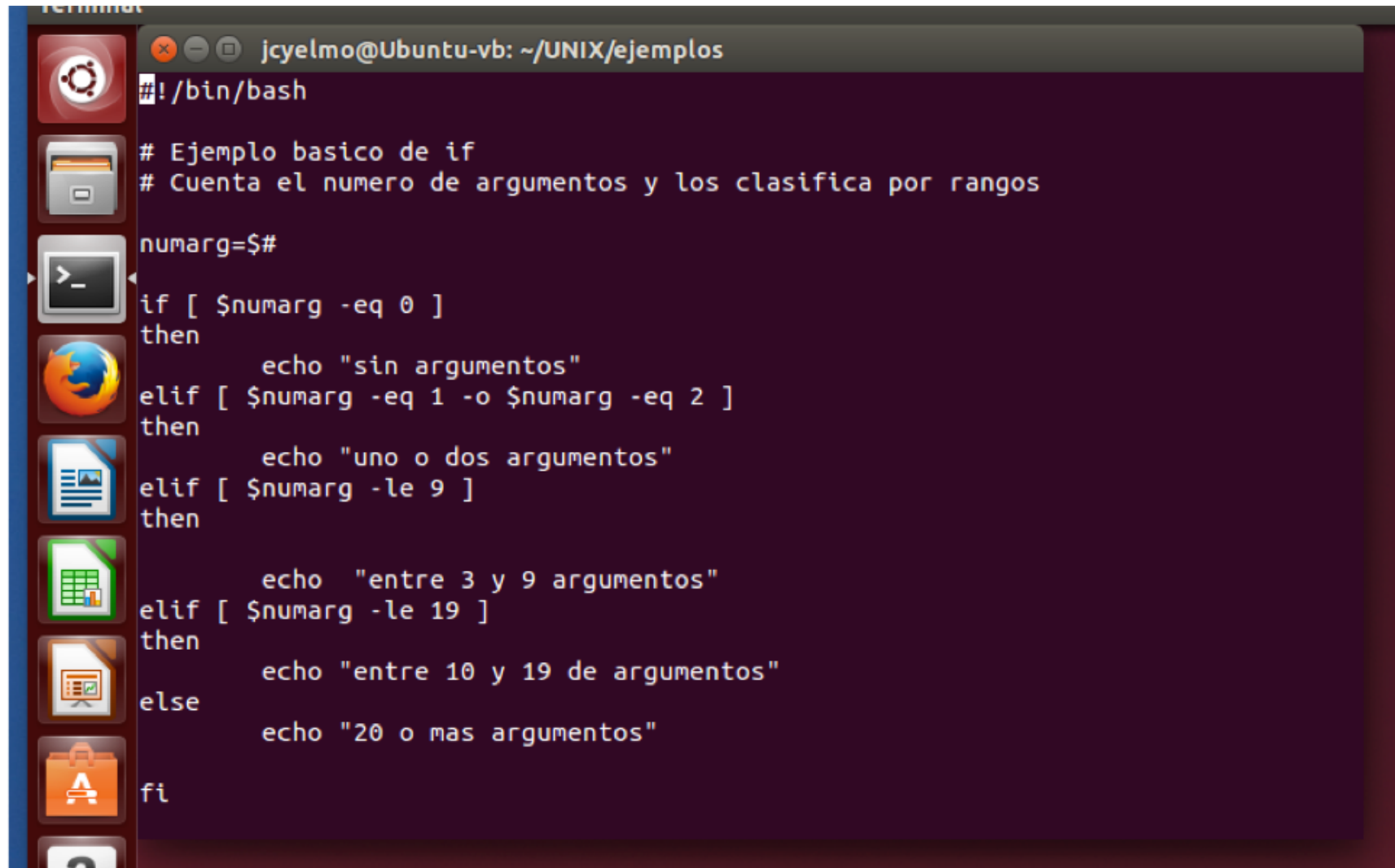
n1 -le n2

n1 es menor o igual que *n2*

test

test condicion [condicion]	
Evalúa la condición y si es cierta devuelve cero como valor de retorno. En caso de condición falsa devuelve un valor distinto de cero	
Expresiones	
! expr	Negación
(...)	Agrupación de expresiones
e1 -a e2	And lógico
e1 -o e2	Or lógico

Ejemplo de if y test



```
Terminal
jcyelmo@Ubuntu-vb: ~/UNIX/ejemplos
#!/bin/bash

# Ejemplo basico de if
# Cuenta el numero de argumentos y los clasifica por rangos

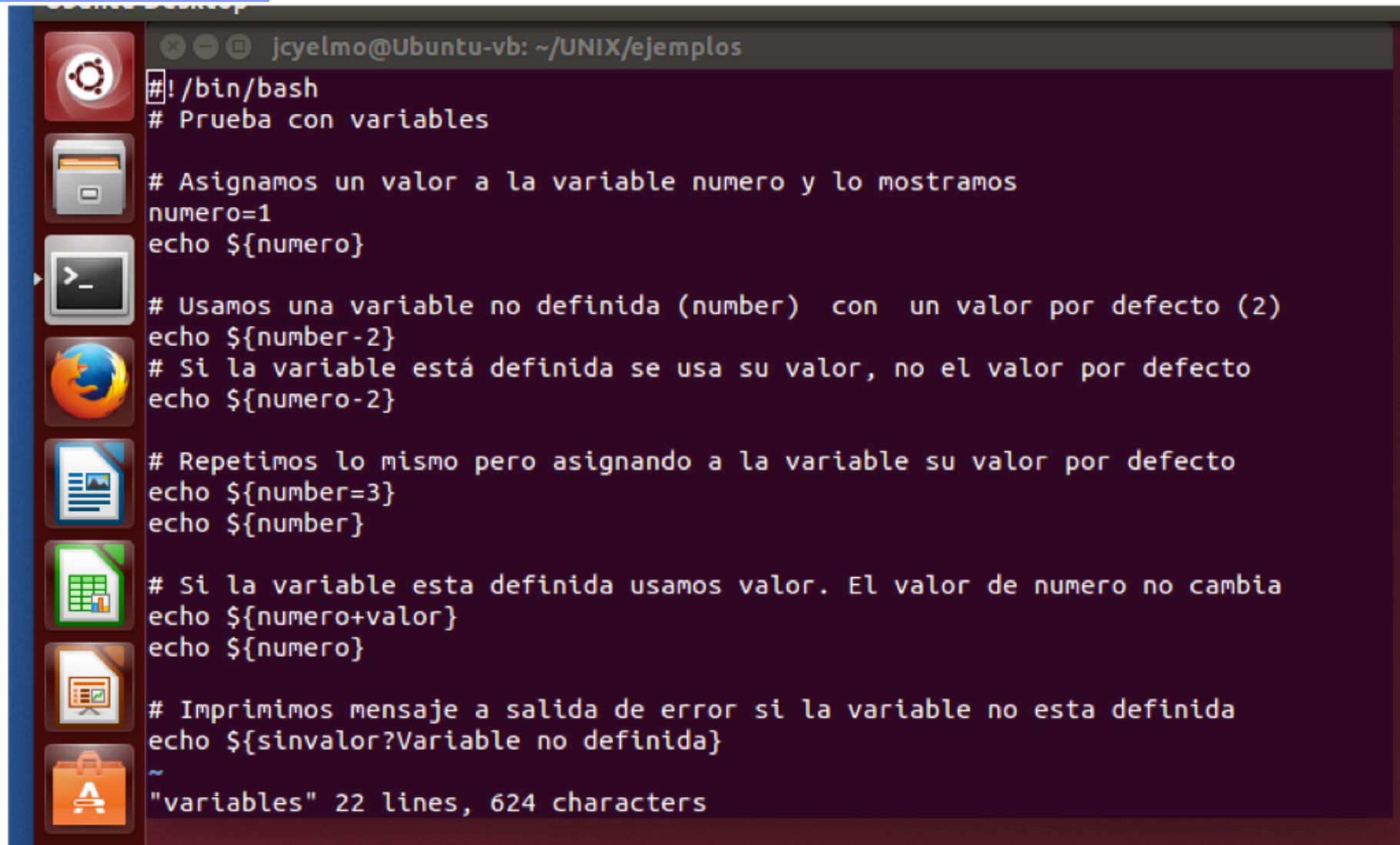
numarg=$#

if [ $numarg -eq 0 ]
then
    echo "sin argumentos"
elif [ $numarg -eq 1 -o $numarg -eq 2 ]
then
    echo "uno o dos argumentos"
elif [ $numarg -le 9 ]
then
    echo "entre 3 y 9 argumentos"
elif [ $numarg -le 19 ]
then
    echo "entre 10 y 19 de argumentos"
else
    echo "20 o mas argumentos"
fi
```

Sustitución de variables

<code>\$var</code>	Valor de la variable <i>var</i> , si está definida
<code>\${var}</code>	Lo mismo pero delimita el nombre de la variable cuando está inserta en una cadena mayor
<code>\${var-valor}</code>	Valor de la variable <i>var</i> , si está definida. Si no se usa <i>valor</i>
<code>\${var=valor}</code>	Valor de la variable <i>var</i> , si está definida. Si no se usa <i>valor</i> y se asigna <i>valor</i> a <i>var</i>
<code>\${var?mensaje}</code>	Valor de la variable <i>var</i> , si está definida. Si no imprime mensaje en salida de error
<code>\${var+valor}</code>	Usa <i>valor</i> si la variable <i>var</i> está definida

Ejemplo de variables



```
jcyelmo@Ubuntu-vb: ~/UNIX/ejemplos
#!/bin/bash
# Prueba con variables

# Asignamos un valor a la variable numero y lo mostramos
numero=1
echo ${numero}

# Usamos una variable no definida (number) con un valor por defecto (2)
echo ${number-2}
# Si la variable está definida se usa su valor, no el valor por defecto
echo ${numero-2}

# Repetimos lo mismo pero asignando a la variable su valor por defecto
echo ${number=3}
echo ${number}

# Si la variable esta definida usamos valor. El valor de numero no cambia
echo ${numero+valor}
echo ${numero}

# Imprimimos mensaje a salida de error si la variable no esta definida
echo ${sinvalor?Variable no definida}

~
"variables" 22 lines, 624 characters
```




El sistema operativo UNIX

Introducción a la programación Shell II

Fin del tema

Juan Carlos Yelmo



El sistema operativo UNIX

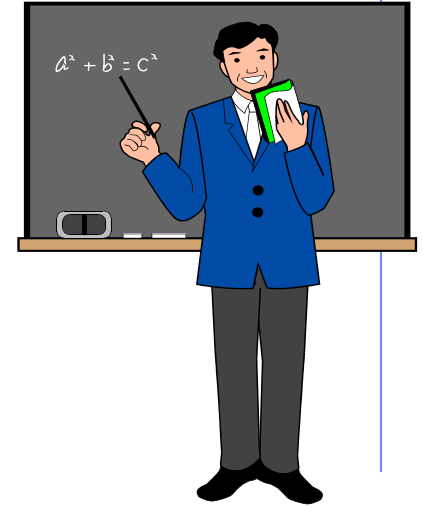
Introducción a la programación Shell III

Juan Carlos Yelmo

Contenidos

4. Programación en shell

- Iteraciones con `for`
- Bucles con `while/do-until`
- Funciones



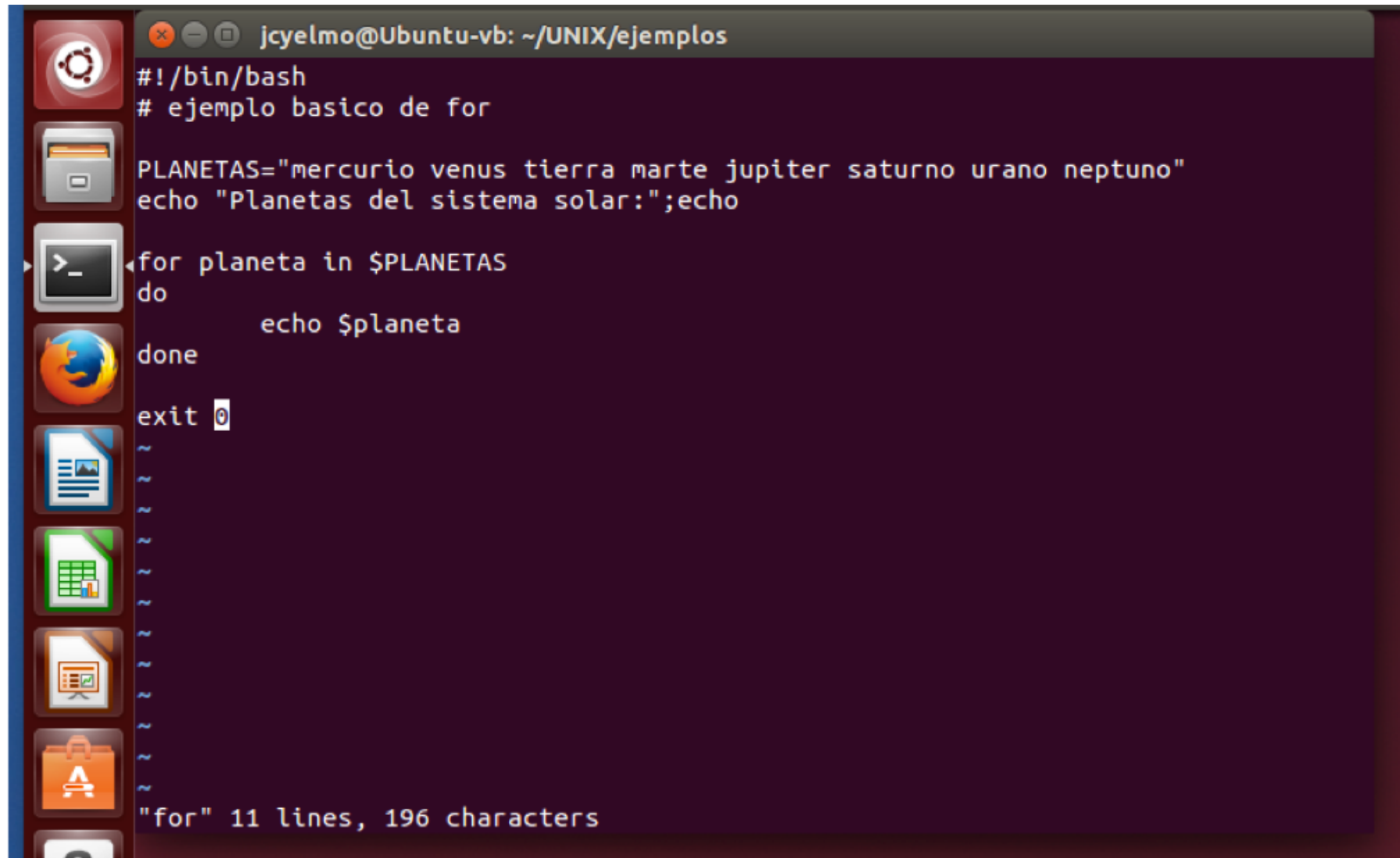
for

*var asume sucesivamente los
valores: valor1 ... valorN*

*Si la lista de valores no está
presente se toman de \$@ (variable
especial similar a \$*)*

```
for var [in valor1 valor2 ... valorN]
do
    comandos
done
```

Ejemplo de for



The screenshot shows a terminal window titled "jcyelmo@Ubuntu-vb: ~/UNIX/ejemplos". The terminal contains the following text:

```
#!/bin/bash
# ejemplo basico de for

PLANETAS="mercurio venus tierra marte jupiter saturno urano neptuno"
echo "Planetas del sistema solar: ";echo

for planeta in $PLANETAS
do
    echo $planeta
done

exit 0
```

Below the code, there are several tilde (~) characters representing line numbers. At the bottom, a status message reads: "for" 11 lines, 196 characters.

Ejemplo de for

A terminal window titled "Terminal" with a dark background. The window shows a bash shell prompt and a script for demonstrating a for loop. The script lists files in the current directory using 'ls -l' for each file. The terminal output shows the script being executed, with file details printed for each file in the directory. The window has a sidebar with icons for various applications like a file manager, web browser, and terminal.

```
Terminal
jcyelmo@Ubuntu-vb: ~/UNIX/ejemplos
#!/bin/bash
# ejemplo basico de for
# El asterisco en for expande a la lista de ficheros en $PWD
# Este script es equivalente a ls -l .

for file in *
do
    ls -l $file
done

exit 0

~
~
~
~
~
~
~
~
~
~
"for2" 11 lines, 181 characters
```

while

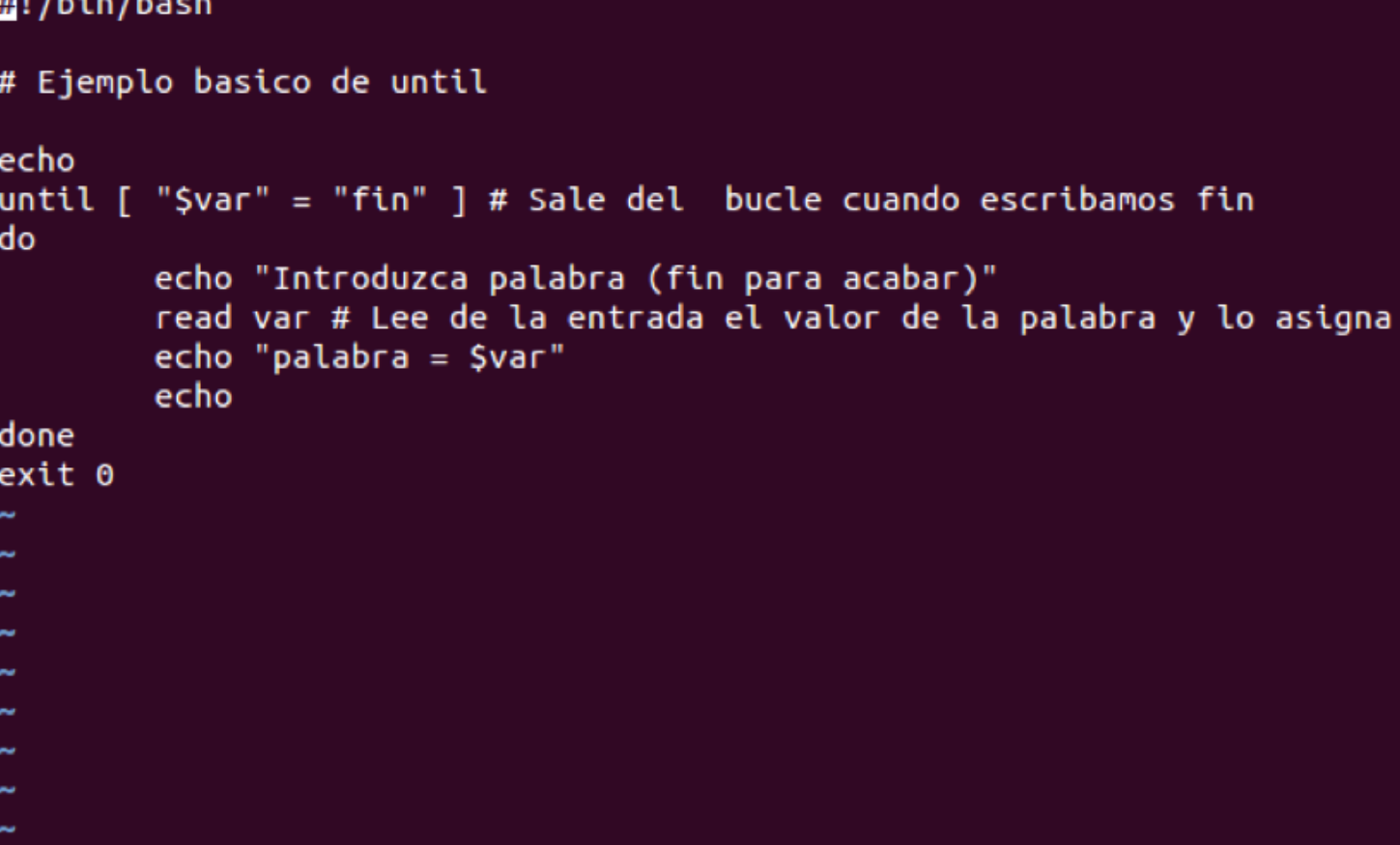
```
while condicion  
do  
    comandos  
done
```

*Comando.
Normalmente
evaluación de
expresión con test*

until

until condicion
do
comandos
done

*Comando.
Normalmente
evaluación de
expresión con test*



```

jcyelmo@Ubuntu-vb: ~/UNIX/ejemplos
#!/bin/bash

# Ejemplo basico de until

echo
until [ "$var" = "fin" ] # Sale del bucle cuando escribamos fin
do
    echo "Introduzca palabra (fin para acabar)"
    read var # Lee de la entrada el valor de la palabra y lo asigna a var
    echo "palabra = $var"
    echo
done
exit 0

~
~
~
~
~
~
~
~
~
"until" 13 lines, 272 characters

```

Funciones

Definición de función o alias



nombre () {


En sh comparte captura de señales del padre y no puede ser recursiva



comandos

}

Ejemplo de función



A terminal window titled "Terminal" with the user "jcyelmo@Ubuntu-vb" in the directory "~/UNIX/ejemplos". The terminal shows a shell script for a basic function example. The script defines a function named "fun" that prints a message, sleeps for a second, and repeats this process 30 times. The function is then invoked, and the terminal shows the output of the function, including a series of dashes and the word "FUNCTIONS".

```
#!/bin/bash

# Ejemplo basico de funciones

RATITO=1

fun () # Se declara antes de usarla
{
    i=0
    REPETICIONES=30
    echo
    echo "Comienza a ejecutar la funcion"
    echo
    sleep $RATITO # Hey, wait a second!
    while [ $i -lt $REPETICIONES ]
    do
        echo "-----FUNCTIONS----->"
        echo "<-----ARE-----"
        echo "<-----FUN----->"
        echo
        let "i+=1"
    done
}

# Invocamos la funcion

fun

exit $?
```



El sistema operativo UNIX

Introducción a la programación Shell III

Fin del tema

Juan Carlos Yelmo



El sistema operativo UNIX

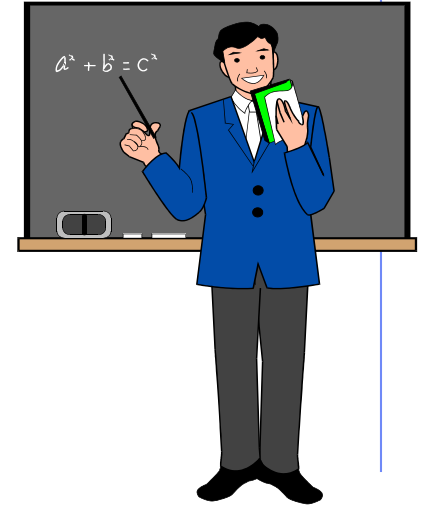
Introducción a la programación Shell IV

Juan Carlos Yelmo

Contenidos

4. Programación en shell

- Ejemplo de Shell Script



Ejemplo sencillo

- ◆ Lista los 10 directorios más ocupados a partir de un path dado
 - gordos path

```
$ cat gordos
#!/bin/bash
# Lista directorios mas ocupados por debajo de $1

du $1 | sort -n | tail
$
```


Ejercicio

- ◆ Estudie el ejemplo: páginas de manual de cada uno de los comandos utilizados en el ejemplo y su ejecución manual por separado y en combinación
- ◆ Edite un fichero de nombre *gordos* con el contenido que se muestra en el ejemplo. Guárdelo, añádale permiso de ejecución y ejecute el comando de las siguientes formas:

```
$ bash -x gordos $HOME
```

```
$ ./gordos $HOME
```

- ◆ Complete el Shell script para que reúna las características que se indican a continuación

Ejercicio

*Modificador opcional para indicar
El número de directorios a listar.
Por defecto, 10*

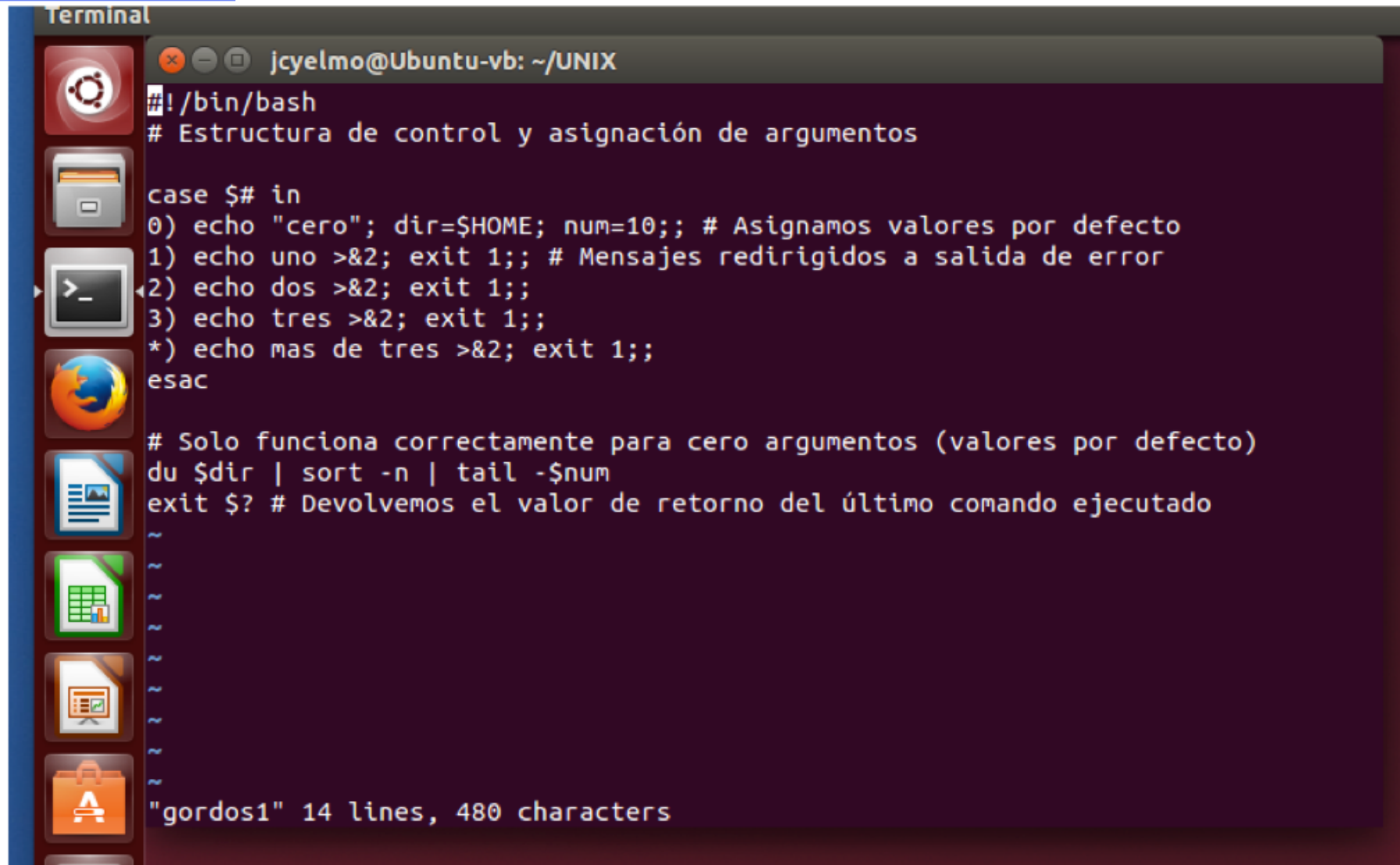
*Argumento opcional para indicar
el directorio de partida para la búsqueda
de subdirectorios más ocupados.
Por defecto, \$HOME*

\$ gordos [-n k] [DIR]

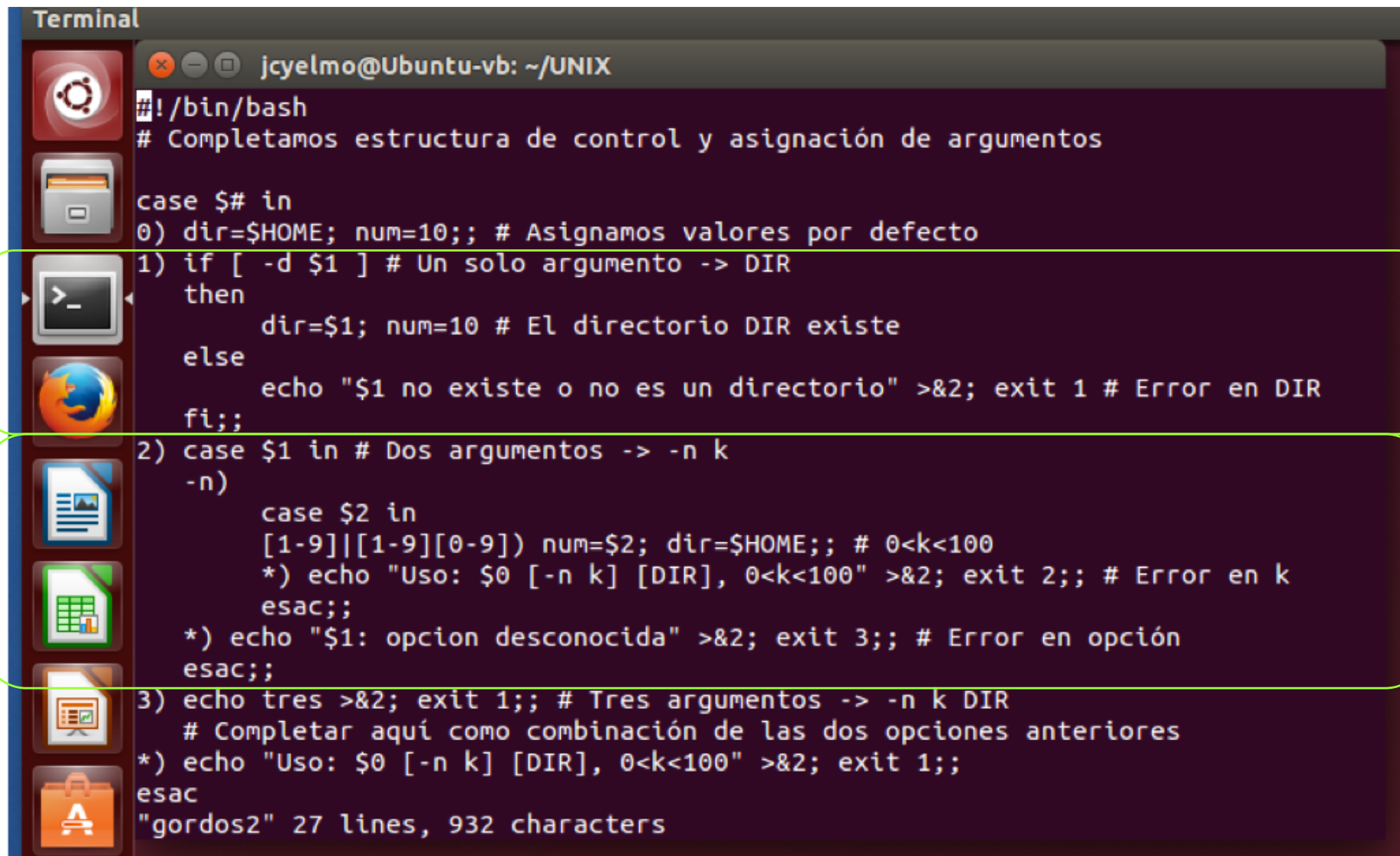
Muestra un listado de los k subdirectorios más ocupados por debajo del directorio DIR

Ejercicio

- ◆ Muestra un listado de los k ($0 < k < 100$) subdirectorios más ocupados por debajo del directorio DIR. Si no se proporciona DIR, el directorio de referencia será el directorio HOME del usuario que invoca el script. Si no se utiliza la opción `-n`, se listan los 10 subdirectorios más ocupados.
- ◆ Implemente un control básico de errores: El script no puede tener más de una opción (`-n k`) ni más de un argumento (DIR) y si se le proporciona uno ha de ser un directorio existente.
- ◆ Si el comando se utiliza mal, se imprimirá un mensaje de error a través de la salida estándar de error informando sobre el error concreto: número de argumentos incorrecto, directorio no existe, el fichero no es un directorio, uso de una opción no contemplada, k no es un número en el rango contemplado, etc.
- ◆ El script devuelve al entorno un valor de éxito o error de ejecución
- ◆ Documente el script con líneas de comentario para que resulte legible y fácilmente mantenible.



Gordos V2



```
Terminal
jcyelmo@Ubuntu-vb: ~/UNIX
#!/bin/bash
# Completamos estructura de control y asignación de argumentos

case $# in
0) dir=$HOME; num=10;; # Asignamos valores por defecto
1) if [ -d $1 ] # Un solo argumento -> DIR
    then
        dir=$1; num=10 # El directorio DIR existe
    else
        echo "$1 no existe o no es un directorio" >&2; exit 1 # Error en DIR
    fi;;
2) case $1 in # Dos argumentos -> -n k
    -n)
        case $2 in
        [1-9]|[1-9][0-9]) num=$2; dir=$HOME;; # 0<k<100
        *) echo "Uso: $0 [-n k] [DIR], 0<k<100" >&2; exit 2;; # Error en k
        esac;;
        *) echo "$1: opcion desconocida" >&2; exit 3;; # Error en opción
        esac;;
3) echo tres >&2; exit 1;; # Tres argumentos -> -n k DIR
    # Completar aquí como combinación de las dos opciones anteriores
    *) echo "Uso: $0 [-n k] [DIR], 0<k<100" >&2; exit 1;;
    esac
"gordos2" 27 lines, 932 characters
```

Casos de prueba

Casos de prueba

```
$ ./gordos
$ ./gordos $HOME
$ ./gordos /home
$ ./gordos -n 8
$ ./gordos -x 8
$ ./gordos -n 8 $HOME
$ ./gordos -n 5 /home
$ ./gordos -n 5 /home $HOME
$ ./gordos kk
```



El sistema operativo UNIX

Introducción a la programación Shell IV

Fin del tema

Juan Carlos Yelmo