Inmersión Linux: de 0 a 100 en 30 Horas

Tema 4: Shell Scripting

J. Félix Ontañón <felixonta@gmail.com>

Indice

- Espacio de usuario y espacio de kernel
- ¿Qué es una shell?
- Breve historia
- Comandos básicos
- Tuberías y redirecciones
- Variables bash y variables de entorno
- Sentencias de control
- Scripts
- Funciones
- Comandos avanzados

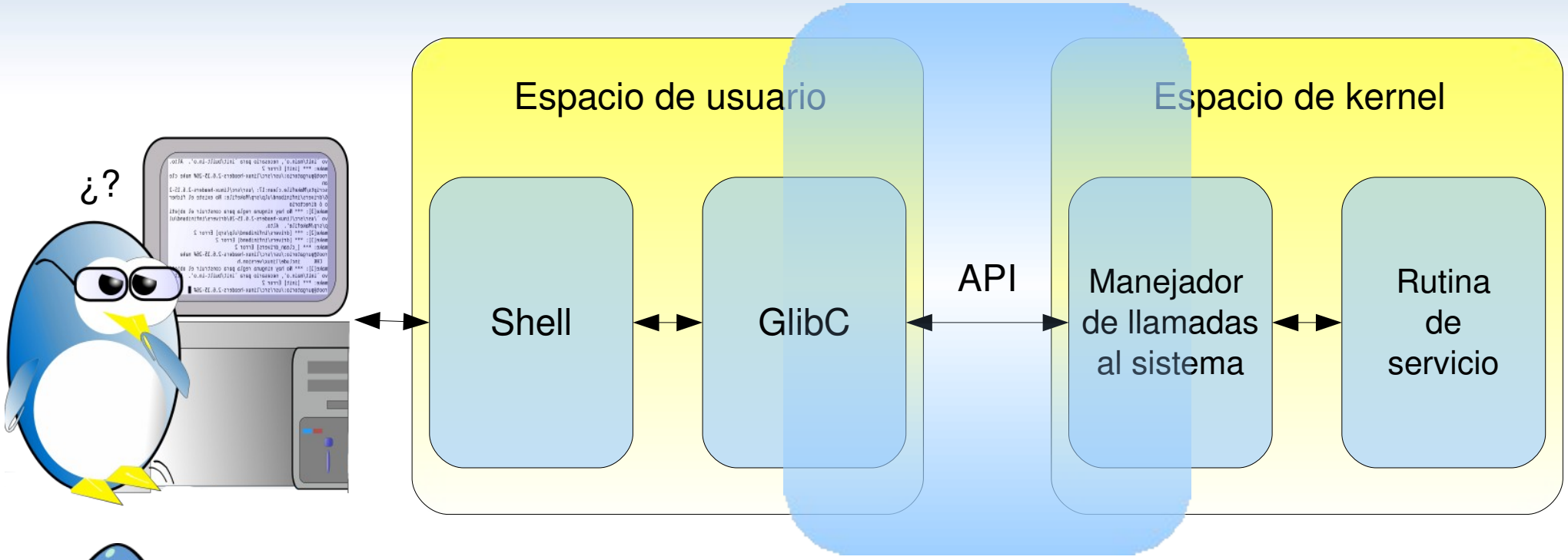
Espacio de kernel

El kernel se ocupan de gestionar los recursos de hardware de la máquina de una forma eficiente y sencilla, ofreciendo al usuario una interfaz de programación simple y uniforme. Toda subrutina que forma parte del kernel tales como los módulos o drivers se consideran que están en el espacio del kernel.

Espacio usuario

Los programas que utiliza el usuario final, tales como las "shell", residen en el espacio de usuario. Estas aplicaciones necesitan interaccionar con el hardware del sistema, pero no lo hacen directamente, sino a través de las funciones que soporta el kernel.

- Intérprete de comandos. Interfaz modo-texto al S.O.
- Se comunica con las rutinas de servicio del núcleo a través de la librería estándar de C GlibC.
- Potente lenguaje de scripting, según qué shell.



- **sh**: Bourne shell. Shell por defecto en Unix desde 1977
- **csh**: C shell. Es una shell dirigida a programadores.
  Integra la sintaxis del lenguaje C. Principios de los 80.
- **ksh**: Korn shell. Integra elementos de sh y csh. Creada en los 80
- **bash:** Bourne-again shell. Creada para el proyecto Gnu.
  Evolución de ksh. Ampliamente utilizada en el mundo Linux.

La potencia del shell scripting reside en la combinación de los comandos del sistema sumada a las built-in features de la shell.

`ls`	Lista el contenido de un directorio
`cd`	Cambia de directorio
`mkdir`	Crea un directorio
`rm`	Borra ficheros y directorios
`cat`	Muestra el contenido de un fichero
`echo`	Imprime cadenas de caracteres por pantalla
`cp`	Copia ficheros y directorios
`mv`	Mueve ficheros y directorios
`find`	Busca nombres de ficheros
`grep`	Busca cadenas de texto que siguen un patrón
`wc`	Cuenta líneas, palabras y letras
`ln`	Crea enlaces duros y simbólicos
`who`	Muestra información sobre los usuarios en el sistema
`uptime`	Muestra información interesante sobre el sistema
`ps`	Muestra información sobre los procesos en ejecución
`test`	Realiza comparaciones lógicas con cadenas, enteros y ficheros

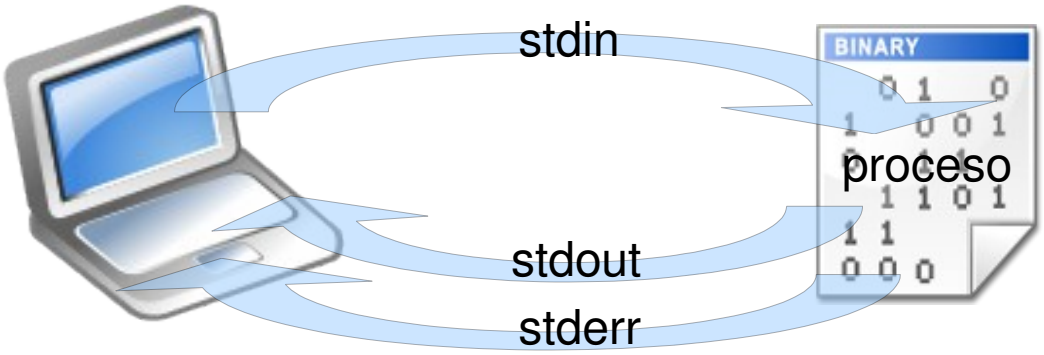
Nota:
Para información detallada sobre un comando tecléese

```
man comando
```

Entrada estándar, salida estándar y salida de error

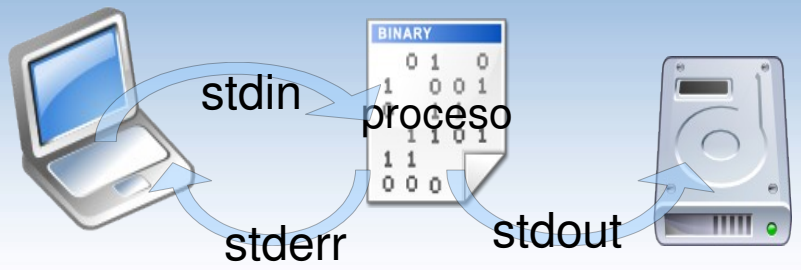
*Cada proceso abre tres "archivos" estándar: entrada estándar (**stdin**), salida estándar (**stdout**) y error estándar (**stderr**).*

- **stdin**: Lugar desde donde los procesos reciben la entrada.
  Por defecto el teclado.
- **stdout**: Lugar por donde los procesos envían la salida.
  Por defecto pantalla.
- **stdout**: Lugar por donde los procesos envían los mensajes de error.
  Por defecto la pantalla.

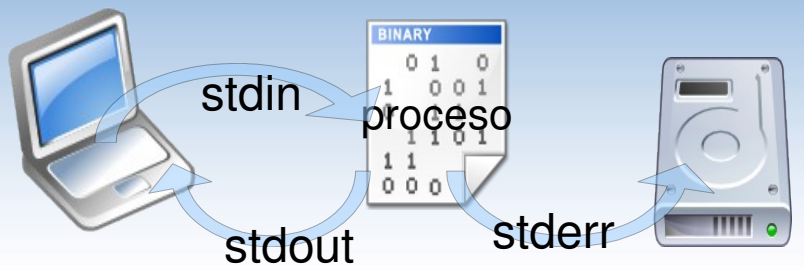


Redirecciones

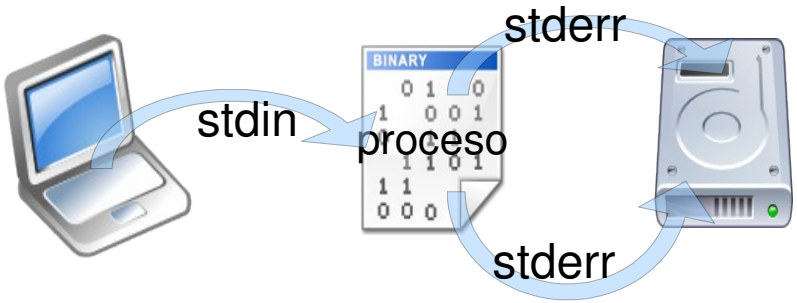
Desde la shell podemos desviar la salida estándar o de error de un proceso a un archivo. También podemos usar un archivo como entrada estándar de una aplicación



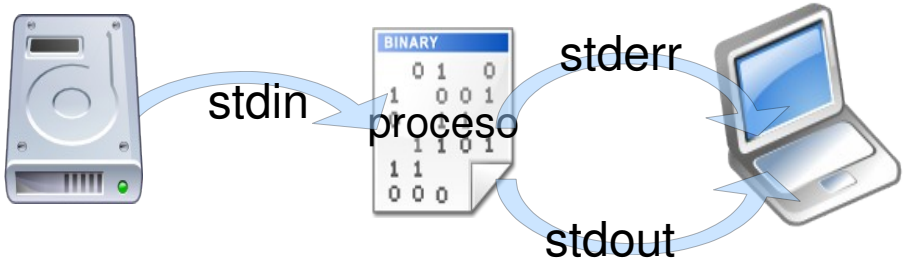

```
$ ls -a > mis_ficheros.txt
```




```
$ make 2> errores_compilando.txt
```



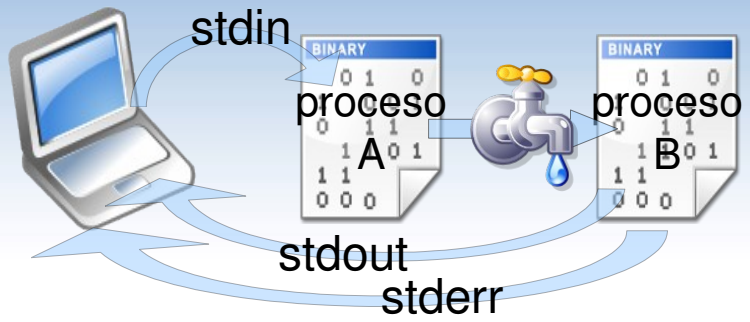

```
$ make > salida.txt 2> errores.txt
```



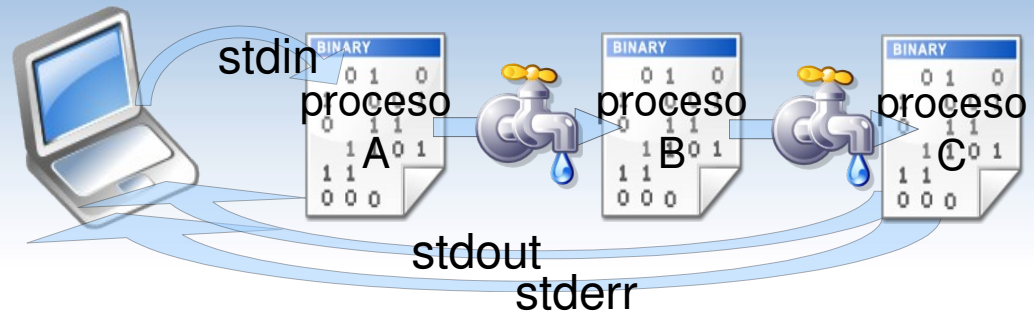

```
$ wc -l < fichero
```

Tuberías

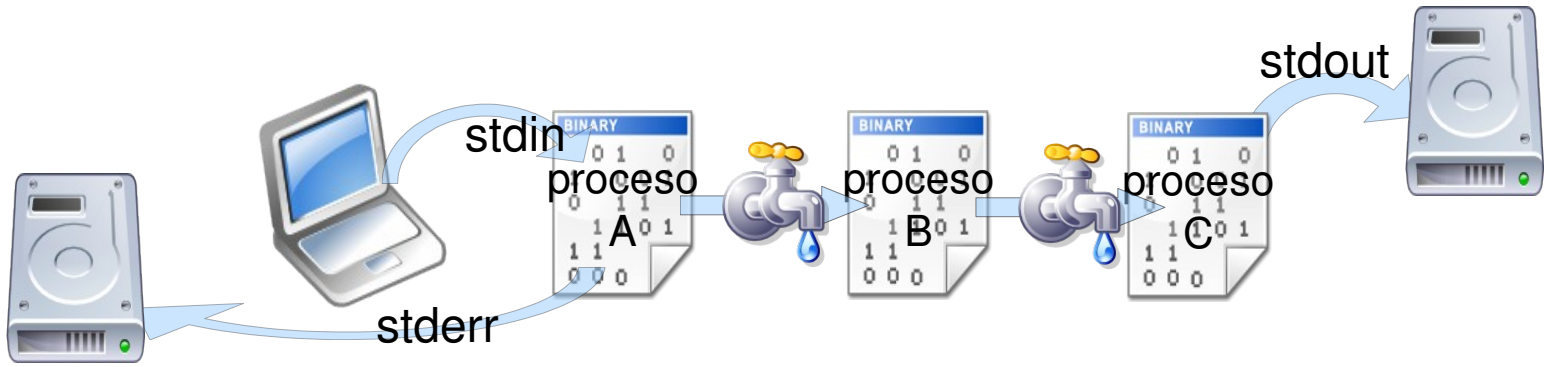
También podemos canalizar la salida de un proceso como entrada de otro proceso. De este modo podríamos decir que "conectamos" comandos. Este es el concepto de tubería.




```
$ find . | grep informes
```




```
$ find . | grep informes | wc -l
```




```
$ find .  2> errores | grep informes | wc -l > num_informes.txt
```

Variables en bash

- Pueden crearse variables y arrays de variables.
- Consideraremos a bash como lenguaje débilemente tipado.
- Podemos asignar la salida de un comando a una variable.

```
yo@mipc:~$ una_variable="Hola Mundo"
yo@mipc:~$ echo $una_variable
Hola Mundo
yo@mipc:~$ una_variable=$(whoami)
yo@mipc:~$ echo $una_variable
yo
yo@mipc:~$ un_array=( hola adios si no)
yo@mipc:~$ echo ${un_array[1]}
adios
```

Variables de entorno

Contamos con un conjunto de variables definidas cuando iniciamos una sesión de shell, algunas definidas por el S.O y otras por las aplicaciones lanzadas y otras asignadas en nuestro fichero de configuración de shell.

`$PATH`	Caminos de búsqueda de ejecutables
`$LANG`	Idioma de nuestra sesión de shell
`$PWD`	Ruta actual
`$HOME`	Directorio *home* del usuario (carpeta personal)
`$HOSTNAME`	Nombre de la máquina
`$PS1`	Formato del prompt
`$?`	Código de retorno del comando anterior
`$RANDOM`	Entero aleatorio

Nota:
Podemos ver todas las variables de entorno definidas ejecutando el comando `set`

Estructuras condicionales

Con estructuras condicionales nos referimos a sentencias del tipo `if then else` *o* `switch case`*. Este tipo de estructuras son built-in functions de bash.*

```
yo@mipc:~/$ T1="eso"
yo@mipc:~/$ T2="aquello"
yo@mipc:~/$ if [ $T1 = $T2 ]; then echo igual; else echo distinto; fi
```

```
yo@mipc:~/$ uno=1
yo@mipc:~/$ dos=2
yo@mipc:~/$ if [ $uno -lt $dos ]
> then echo menor
> else echo mayor
> fi
menor
```

```
yo@mipc:~/$ case $(wc -l fichero) in
> 2 ) echo "dos líneas"; break;;
> 3 ) echo "tres líneas" ; break;;
> * ) echo “no lo se”;;
> esac
dos líneas
```

Bucles I

*Contamos con los bucles tipo: **for, while, until**.*
El bucle for es algo particular en bash porque opera sobre series de "palabras"

```
yo@mipc:~/$ echo -n "Mis amigos son:"; for i in Maria Pedro Juan;
do echo -n " $i"; done; echo
Mis amigos son: Maria Pedro Juan
```

```
yo@mipc:~/$ for i in $(seq 1 4)
> do
> echo $i- $RANDOM
> done
1- 23242
2- 23123
3- 232
4- 2312
```

```
yo@mipc:~/$ array=( a b c )
yo@mipc:~/$ for i in ${array[*]}
> do
> echo $i
> done
a
b
c
```

Bucles II

```
yo@mipc:~/$ CONTADOR=0
yo@mipc:~/$ while [ CONTADOR -lt 10 ]; do
> echo -n $CONTADOR; done
> let CONTADOR=$CONTADOR+1
> done
0123456789

yo@mipc:~/$ CONTADOR=0
yo@mipc:~/$ until [ CONTADOR -gt 9 ]; do
> echo -n $CONTADOR
> let CONTADOR=$CONTADOR+1
> done
0123456789
```

En informática, un script es un guión o conjunto de instrucciones. Permiten la automatización de tareas creando pequeñas utilidades. Es muy utilizado para la administración de sistemas UNIX. Son ejecutados por un intérprete de línea de comandos y usualmente son archivos de texto.

Wikipedia

$ chmod +x

```
#!/bin/bash

for i in $(seq 1 10)
do
  echo Saludo nº $i
done
```

script bash

Podemos organizar en código en funciones. Las peculiaridades al respecto son el nombre de los argumentos que recibe una función. Éstos se llaman $1 $2 $3 y así sucesivamente.

```
#!/bin/bash

function dos_parametros {
    echo son $1 y $2
}

function archivo {
    echo mi archivo es: $1
}

dos_parametros este $1
for i in $(ls); do
    archivo $PWD/$i
done
```

```
yo@mipc:~/$ chmod +x mi_script.sh
yo@mipc:~/$ ./mi_script.sh aquel
son este y aquel
mi archivo es: /home/yo/Audio
mi archivo es: /home/yo/Desktop
mi archivo es: /home/yo/mi_script.sh
mi archivo es: /home/yo/Documentos
mi archivo es: /home/yo/informe.txt
...
...
...
```

awk

*Usaremos awk como comando para el **tratamiento de texto**. awk es capaz de realizar búsquedas por líneas que cumplan ciertos criterios (o expresiones regulares) y aplicar procesamiento a los campos de las líneas que validan los criterios.*

```
yo@mipc:~/$ cat personas | awk '/Maria/ {print $3}' | sort | uniq
Abogado
Profesora
yo@mipc:~/$ awk '.*a.*u.*' {print $1,$2}' personas
Felipe Marquez
Jose Lopez
Manuel Dominguez
yo@mipc:~/$ who | awk '// {print $1}'
yo
yo@mipc:~/$ ps -A | awk '// {print $NF}'
```

sed

sed es otra potente herramienta de tratamiento de texto. Permite cortar líneas y realizar sustituciones de palabras que cumplan criterios (o expresiones regulares) a partir de un flujo de datos.

```
yo@mipc:~/$ cat personas | grep Pepe | sed 's/[a,e,o,u]/i/g'
Pipi Sinchiz Fliristi
yo@mipc:~/$ sed 2, 5d personas
Maria Perez Abogado
Maria García Profesora
Manuel Dominguez Ejecutivo
yo@mipc:~/$ cat /var/log/auth.log | grep Failed | awk '// {print
$9,$1,$2,$3}' | sed 's/^/POSIBLE INTRUSO /g' > INTRUSOS.txt
```

Bibliografía

http://es.wikipedia.org/wiki/Bash
http://es.tldp.org/COMO-INSFLUG/COMOs/Bash-Prog-Intro-COMO/Bash-Prog-Intro-COMO.html
http://www.freeos.com/guides/lsst/
http://docs.hp.com/es/5187-2217/ch03s03.html