

Práctica 2 del Laboratorio de Sistemas Operativos

Departamento de Automática
Universidad de Alcalá

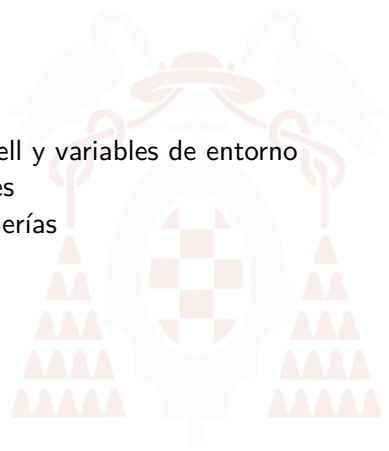


/gso>

Índice

1 Práctica 2

- Variables shell y variables de entorno
- Redirecciones
- Filtros y tuberías



Variables shell (I) - Definición

Variable shell

En sistemas Unix/Linux, una variable shell es aquella que caracteriza **únicamente** al entorno de trabajo (shell), es decir, funciona como variable dentro de una instancia del intérprete.

- Definición -> `identificador=valor`
- Consulta -> `echo $(identificador) / set`
- Modificación -> `identificadorExistente=valor`
- Eliminación -> `unset identificador`

Variables shell (II) - Características

- En la definición, no deben teclearse espacios en blanco en torno a la asignación (=).
- La variable es eliminada cuando muere el proceso.
- Los nombres de las variables solo pueden contener letras (a-zA-Z), números (0-9) y el guión bajo (_).
- Por convención, las variables shell suelen escribirse en mayúscula.
- Las variables shell pueden convertirse en variables de entorno mediante el comando `export`.

Variables de entorno (I) - Definición

Variable de entorno

En sistemas Unix/Linux, una variable de entorno es aquella que caracteriza tanto al entorno de trabajo del intérprete como a todos los subprocesos que genere.

- Definición -> `export identificadorExistente`
- Consulta -> `echo $(identificador) / env`
- Modificación -> `identificadorExistente=valor`
- Eliminación -> `unset identificador`

Variables de entorno (II) - Características

- Si la variable es modificada, los cambios solo se visualizarán en el proceso padre y en los hijos futuros.
- La variable es eliminada cuando muere el proceso padre y los subprocesos hijos.
- Existen variables de entorno muy importantes como HOME, PWD o PATH.

Utilidad

- Crear una copia de seguridad de un directorio.
- Modificar el mensaje por defecto del terminal.
- Utilización de *shell scripts*.

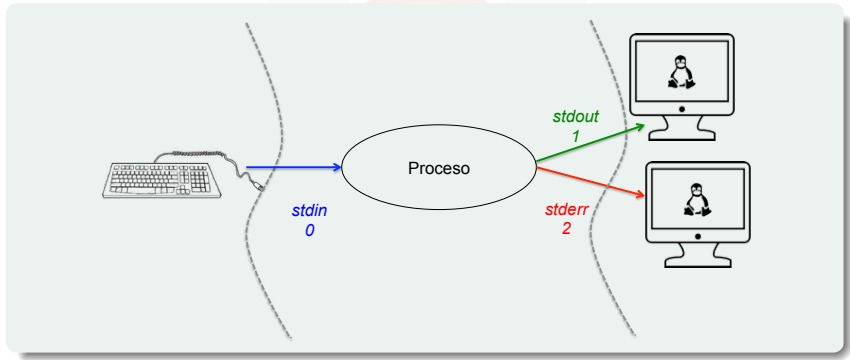
Ejemplo 1: Uso de variables shell.

```
#!/bin/bash  
OF=/var/my-backup-$(date +%Y%m%d).tgz  
tar -cZf $OF /home/me/
```

Ejercicios

- 1 Crear dos variables shell: SALUDO (hola) y DESPEDIDA (adios).
- 2 Exportar la variable DESPEDIDA para convertirla en una variable de entorno.
- 3 Mostrar todas las variables shell. (PISTA: investigue cómo puede guardar el resultado de un comando en un archivo). ¿Muestra las dos variables creadas?
- 4 Mostrar todas las variables de entorno. ¿Pueden visualizarse las dos variables creadas?
- 5 Desde el mismo terminal, abra otro terminal (investigue cómo puede hacerlo). Ahora repita los dos ejercicios anteriores. ¿Qué resultado encuentra?

Redirecciones (I) - Archivos de entrada y salida estándar



Redirecciones (II) - Características

- En Unix casi todo se representa mediante un archivo (teclado y pantalla inclusive).
- Todos los procesos arrancados disponen de tres archivos por parte del sistema operativo:
 - *stdin* (entrada estándar) -> **<**.
 - *stdout* (salida estándar)-> **>** o **»**.
 - *stderr* (salida de errores estándar) -> **2>**.
- Los operadores de redirección se evalúan de izquierda a derecha.
- Las redirecciones sólo pueden ser aplicadas a archivos, **NUNCA A PROCESOS**.

Redirecciones (III) - Entrada estándar

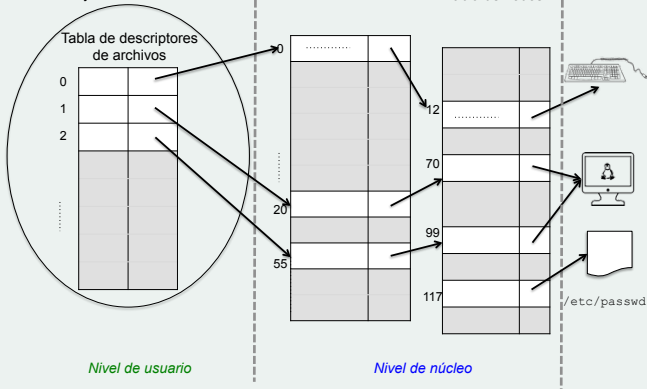
- Teclado → entrada de datos estándar que utilizan los procesos.
- La entrada del sistema puede ser modificada mediante el operador `<`.

```
sort < /etc/passwd
```

Muestra el contenido del archivo `/etc/passwd` ordenado alfabéticamente.

Redirecciones (IV) - Tablas de control de acceso a archivos

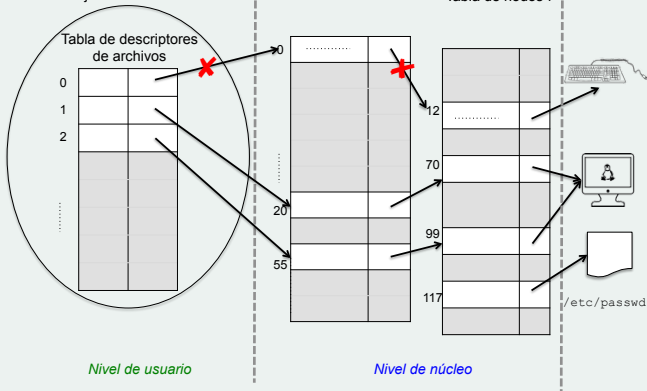
Proceso: ejecución de orden `sort`



Redirecciones (V) - Entrando en algunos detalles del SO

Cerrar archivo de entrada estándar stdin (uso de close())

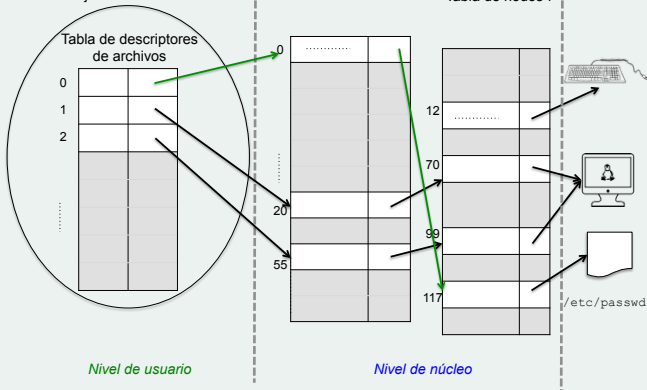
Proceso: ejecución de orden `sort`



Redirecciones (V) - Entrando en algunos detalles del SO

Abrir archivo /etc/passwd) (uso de open())

Proceso: ejecución de orden `sort`



Redirecciones (VI) - Salida estándar

- Pantalla → salida de datos estándar que utilizan los procesos.
- La salida del sistema puede ser modificada mediante:
 - El operador `>` elimina el contenido actual del archivo y después añade la nueva información.
 - El operador `>>` añade la nueva información al final del archivo.

```
cat hola > saludo.txt
```

Elimina el contenido del archivo `saludo.txt` y luego inserta en él el mensaje: `hola`

En este caso, ¿cómo realizaría el Sistema Operativo la redirección?

Redirecciones (VII) - Salida de errores estándar

- Pantalla -> salida de errores estándar que utilizan los procesos.
- La salida de errores del sistema puede ser modificada mediante el operador **2>**.

```
mv noexisto.txt fichero.txt 2> salida.txt
```

Si el archivo `noexisto.txt` no existe, se generará en la ruta actual el archivo `salida.txt` que contendrá el mensaje de error correspondiente.

Redirecciones (VIII) - Archivo `/dev/null`

- En Linux existe un archivo denominado `/dev/null` similar a una papelera.
- Este archivo es gestionado por el sistema operativo.
- Tanto la salida estándar como la salida de errores pueden redirigirse a este archivo para no tener que preocuparse de gestionar esa información.

```
mv noexistto.txt fichero.txt 2> /dev/null
```

Si el archivo `noexistto.txt` no existe, el mensaje de error correspondiente será enviado al archivo `/dev/null`.

Tuberías o *pipelines* (I)

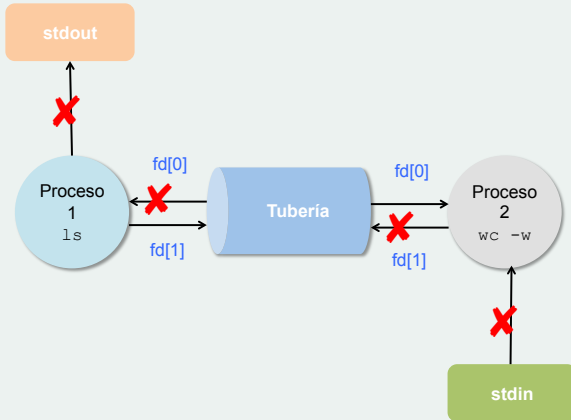
- Permiten conectar la salida estándar de un proceso con la entrada estándar de otro.
- Se especifica mediante el carácter `|`.
- Actúa como un tubo con dos extremos con funcionamiento FIFO (*First In First Out*).
- Esquema genérico de línea de órdenes:
`orden_1 | orden_2 | ... | orden_n`
- **Muy útil** si se combinan con filtros.

```
ls | wc -w //este es un ejemplo muy simple  
ls > tmp; wc -w < tmp; rm tmp //peor alternativa
```

Ejemplos más complejos en la pág. 13 del enunciado de la práctica.

Tuberías o *pipelines* (II). Entrando en algunos detalles del SO

Comunicación entre dos procesos mediante tubería



Filtros

Filtro

Todo proceso que lee de su entrada estándar y escribe en su salida estándar.

- Individualmente: cada filtro hace una tarea muy sencilla.
- Colectivamente: los filtros tienen una gran potencia.
- En Linux existen muchos filtros: `cat`, `wc`, `tee`, `sort`, `cut`, ...

Filtro cat

cat

Permite visualizar el contenido de archivo(s) de texto por pantalla.

Sintaxis: `cat [opciones] [archivo]`

`-b`: muestra sólo numeradas las líneas sin blancos de archivo.

`-n`: muestra numeradas todas las líneas de archivo.

Si no se proporciona ningún archivo, lee la entrada estándar hasta pulsar Ctrl-D.

Ejemplo 2: Uso de cat.

```
cat -n archivo1.txt
```

Filtro `wc`

`wc`

Cuenta el número de caracteres, palabras y líneas de un archivo pasado como parámetro y el total de cada una de las categorías para todos los archivos.

Sintaxis: `wc [opciones] [archivo(s)]`

- c: cuenta únicamente el número de caracteres.
- l: cuenta sólo el número de líneas.
- w: cuenta sólo el número de palabras.

Ejemplo 3: Uso de `wc`.

```
wc -lw archivo1.txt
```

Filtro tee

tee

Transcribe su entrada estándar a su salida estándar copiándola además en los archivos especificados.

Sintaxis: `tee [opciones] [archivo]`

-a: añade la entrada a los archivos sin sobreescribirlos.

Ejemplo 4: Uso de tee.

```
who | tee -a archivo1.txt usuarios
```

Filtro sort

sort

Ordena las líneas de la entrada y las envía a la salida estándar.

Sintaxis: `sort [opciones] [archivo]`

- n: ordenación numérica.
- r: invierte el resultado de la ordenación.
- o: escribe el resultado de la ordenación en un archivo.

Ejemplo 5: Uso de sort.

```
echo -e "3\n 5\n 10\n 31\n 4\n 0\n -3" | sort  
-r
```


Filtro cut





cut

Extrae una o varias columnas (o campos) para cada línea del archivo especificado.

Sintaxis:

- ❶ `cut -f lista [-d carácter] [-s] [archivo]`
- ❷ `cut -c lista [archivo]`

-c lista: extrae los caracteres que ocupa lista en cada línea¹.
-f lista: la separación se hará mediante delimitadores (-d o -s).
-d carácter: especifica el carácter delimitador.
-s: con la opción -f, elimina de la salida estándar las líneas sin delimitadores.

¹Lista creciente y separada por comas o guiones con las posiciones.    

Filtro cut

Ejemplo 6: Uso de cut.

```
echo -e "hi \n bye" | cut -c 2
```

Resultado:

- 1 i
- 2 b

Filtro cut

Ejemplo 7: Uso de cut.

```
echo -e "en un lugar de la mancha \n de cuyo  
nombre no quiero acordarme\n..." | cut -f 3  
-d 'u' -s
```

Resultado:

- ① gar de la mancha
- ② iero acordarme

Filtro grep

grep

Busca en los archivos los patrones de texto especificados.

Sintaxis: `grep [opciones] [archivo(s)]`

- f `archivo_patrones`: indica los patrones a buscar en el archivo (`archivo_patrones`).
- e (`patron`): especifica `patron(es)` en el propio comando.
- i: ignora la diferencia entre mayúscula y minúscula.
- v: se muestran sólo las líneas que no se emparejan.

Ejemplo 8: Uso de grep.

```
echo -e "hi \n bye" | grep -e "h" -e "E"  
echo -e "hi \n bye" | grep -e "^ b"
```