Linux Administrator

Tema 3: Comandos Shell de administración

Sonia Fernández Sapena

COMANDOS DE ADMINISTRACIÓN

El shell bash

El intérprete de comandos ejecuta las instrucciones introducidas con el teclado o en un script y le devuelve los resultados. Este intérprete es un programa comúnmente llamado shell.

Como «shell» significa concha, viene a decir que es lo que rodea al **núcleo** de Linux: se utiliza mediante comandos. Por lo tanto, es una interfaz que funciona en modo texto entre el núcleo de Linux y los usuarios (avanzados) o las aplicaciones.

Hay varios shells: cada uno dispone de especificaciones propias.

1. El Bourne Shell (sh) es el shell más conocido y habitual en los Unix.
2. El C-Shell (csh) retoma la estructura del lenguaje C.
3. El Korn Shell (ksh) es una evolución del Bourne Shell.
4. El Z-Shell (zsh) es a su vez una evolución del Korn Shell.
5. El shell de referencia en Linux se llama Bourne Again Shell (bash). A continuación le presentamos una lista exhaustiva de intérpretes de comandos que puede encontrar en Linux:
6. tcsh: Tenex C Shell;
7. ash: A Shell;
8. dash: Debian Almquist Shell.

COMANDOS DE ADMINISTRACIÓN

Bash: el shell por defecto

a. Un shell potente y libre

El bash no sólo está en Linux. Al ser un programa libre, se puede compilar o ejecutar en numerosas plataformas.

Se distinguen dos tipos de ellos en Linux:

- las consolas virtuales de texto, el modo por defecto de Linux cuando arranca o funciona bajo entorno gráfico;
- las consolas o terminales gráficos, como xterm, eterm o konsole, que son emuladores de terminales en el seno de ventanas gráficas.

El shell funciona en un terminal. Espera entradas por teclado en la consola o la ventana, y visualiza sus resultados en el mismo lugar.

COMANDOS DE ADMINISTRACIÓN

Línea de comandos

- El shell espera entradas por el teclado en una línea llamada **línea de comandos** o **prompt**
- La línea (prompt) proporciona información en el terminal y su posición en el sistema de archivos.

    - **seb@slyserver:/home/public>**

    o

    - **seb@slyserver:/home/public$**

    o

    - **[seb@slyserver public]$**

En esta clásica línea, obtiene cuatro datos:

- **seb**: es el nombre de inicio de sesión o login del usuario actualmente conectado al terminal;
- **slyserver**: es el nombre de anfitrión (hostname), el nombre lógico de la máquina conectada al terminal;
- **/home/public**: es la posición actual del shell en el sistema de archivos. Si solo se indica un directorio, es el último del árbol donde se esté ubicado.
- **> o $:** es la terminación estándar del bash para un usuario sin privilegios.
- **#** indica que el usuario es el administrador root que tiene todos los privilegios.

Esta línea le informa de que el usuario sin privilegios de administración seb es el que utiliza el terminal (está conectado) en la máquina slyserver y que se encuentra actualmente en /home/public.

COMANDOS DE ADMINISTRACIÓN

Utilizar el Shell

La introducción de datos

En el terminal, el teclado se utiliza de forma intuitiva. Puede desplazarse en la línea con las flechas de derecha e izquierda del teclado y borrar caracteres con las teclas [Tab] y [Supr].

Hay varios métodos abreviados que son prácticos:

- **[Ctrl] a**: ir al principio de la línea.
- **[Ctrl] e**: ir al final de la línea.
- **[Ctrl] l**: borrar el contenido del terminal, y mostrar la línea de comandos en la parte superior.
- **[Ctrl] u**: borrar la línea hasta el principio.
- **[Ctrl] k**: borrar la línea hasta el final.

```
$ date
mié nov 19 18:46:37 CET 2014
```

```
$ pwd
/home/jolivares
```

Sintaxis general de los comandos

Los comandos o instrucciones (las dos palabras son sinónimas en este caso) GNU tienen a menudo una sintaxis con la misma estructura:

```
Comando [parámetros] [argumentos]
```

COMANDOS DE ADMINISTRACIÓN

Utilizar el Shell

Encadenar los comandos
Puede ejecutar varios comandos en una sola línea, unos tras otros. Para ello, basta con separarlos con un punto y coma.

```
root@ubuntu:~# date;pwd;cal -m  ene 2017
```

Visualizar texto con echo
No hay nada más sencillo que visualizar texto. El comando **echo** está hecho para ello. Como casi todos los comandos, acepta los parámetros, además de los argumentos, en forma de texto. Para visualizar un texto sencillo:

```
$ echo Hola amigos
Hola amigos
```

Secuencia	Acción
\n	Pasar a la línea
\t	Tabulación horizontal
\c	Suprimir el salto de línea final
\b	Retorno de un carácter atrás
\\	Visualiza la barra oblícua
\nnn	Visualiza el carácter especificado en octal

Para utilizar estas secuencias, añada el argumento -e:

```
$ echo -e "Hola.\tMe llamo Javier\b\b\bNadie\n"
Hola. Me llamo Nadie
```

COMANDOS DE ADMINISTRACIÓN

Utilizar el Shell

Comandos internos y externos

Existen dos tipos de comandos:

- Los comandos externos son programas binarios presentes como archivos en su disco duro (o cualquier otro soporte de datos). Cuando ejecuta el comando, se carga este archivo en memoria y se inicia como proceso (esta noción se explicará en este mismo capítulo).

- Los comandos internos son del proprio shell y se ejecutan en él. Estos comandos forman parte del programa shell, el bash. Los comandos **cd** o **pwd** son dos ejemplos de comandos internos. Cuando los ejecuta, el shell ejecuta las funciones definidas en su interior que les corresponden.

Puede distinguir un comando interno de un comando externo con la ayuda del comando interno **type**. Así, **date** es un comando externo. Puede observar que es un archivo presente en /bin, mientras que **pwd** es interno al shell.

```
$ type date
date is /bin/date
$ type pwd
pwd es una orden interna del shell
```

COMANDOS DE ADMINISTRACIÓN

Utilizar el Shell

Comandos internos y externos

Existen dos tipos de comandos:

- Los comandos externos son programas binarios presentes como archivos en su disco duro (o cualquier otro soporte de datos). Cuando ejecuta el comando, se carga este archivo en memoria y se inicia como proceso (esta noción se explicará en este mismo capítulo).

- Los comandos internos son del proprio shell y se ejecutan en él. Estos comandos forman parte del programa shell, el bash. Los comandos **cd** o **pwd** son dos ejemplos de comandos internos. Cuando los ejecuta, el shell ejecuta las funciones definidas en su interior que les corresponden.

Puede distinguir un comando interno de un comando externo con la ayuda del comando interno **type**. Así, **date** es un comando externo. Puede observar que es un archivo presente en /bin, mientras que **pwd** es interno al shell.

```
$ type date
date is /bin/date
$ type pwd
pwd es una orden interna del shell
```

COMANDOS DE ADMINISTRACIÓN

Utilizar el Shell

Algunos atajos útiles
Se deben conocer algunas secuencias de atajos de comandos:
[Ctrl] c: interrupción del programa: se termina.
[Ctrl] z: para el programa (ver los procesos).
[Ctrl] d: interrumpe una introducción de datos en un símbolo del sistema >.

El historial de comandos

Puede ver el contenido del historial con el comando **history**. El resultado siguiente está truncado de manera voluntaria, ya que la lista es demasiado larga.

El comando **fc** funciona como history cuando se utiliza con el parámetro -l. Por defecto, se limita a los últimos quince comandos.

```
$ history
...
 1000  date
 1001  pwd
 1002  uname -a
 1003  ls
 1004  fc -l -5
 1005  history
```

```
$ fc -l -10
109     cal -m -3 12 1975
110     date;pwd;cal
111     date;pwd;cal -m
112     echo "toto\n"
113     echo -e "toto\n"
114     type date
115     type pwd
116     type ll
117     fc -l -10
118     uname -a
```

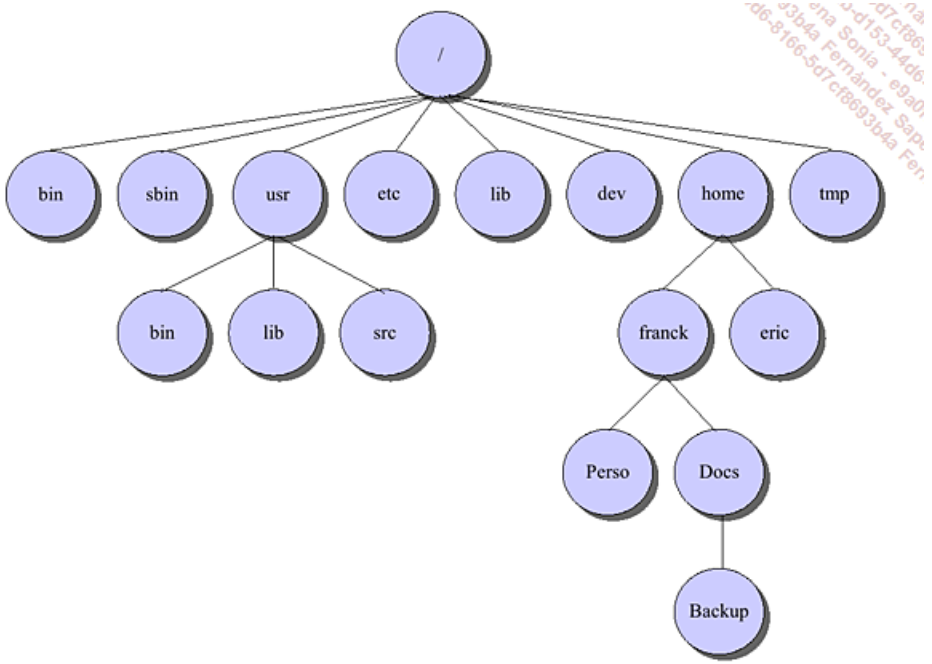
COMANDOS DE ADMINISTRACIÓN

PRACTICAS BASICAS

Practica 1 de clase tema 4

COMANDOS DE ADMINISTRACIÓN

La gestión de los archivos



El sistema de archivos
Un sistema de archivos, llamado comúnmente File System o FS, determina la organización de los datos en un soporte de almacenamiento, y por tanto, cómo gestiona y organiza el sistema operativo los archivos

Linux es, como todo Unix, un sistema operativo completamente orientado a archivos. Se representa todo (o casi todo) con un archivo, tanto los datos (archivos de datos de cualquier tipo, como una imagen o un programa) como los periféricos (terminales, ratones, teclado, tarjeta sonido, etc.) o incluso los medios de comunicación (sockets, tuberías nombradas, etc.). Se puede decir que el sistema de archivos es el corazón de cualquier sistema Unix.

COMANDOS DE ADMINISTRACIÓN

La gestión de los archivos

Los diferentes tipos de archivos
Distinguimos tres tipos de archivos: ordinarios, catálogo, especiales.

a) **Los archivos ordinarios o regulares.** Los archivos ordinarios se llaman también archivos regulares, ordinary files o regular files. Son archivos totalmente clásicos que contienen datos. Por datos se debe entender cualquier contenido.
b) **Los catálogos.** Los archivos catálogo son los directorios o carpetas. Los directorios permiten organizar el disco duro creando una jerarquía. Un directorio puede contener archivos normales, archivos especiales y otros directorios de manera recursiva. Un directorio no es más que un archivo particular que contiene la lista de los propios archivos presentes en este directorio, como un índice. Esta noción resultará muy útil cuando se trate el tema de los permisos.
c) **Los archivos especiales.** El tercer tipo de archivos es el especial. Existen varios tipos de archivos especiales. Por ejemplo, los drivers de los periféricos están representados por archivos especiales de la carpeta /dev. Son principalmente archivos que sirven de interfaz para los diversos periféricos.

COMANDOS DE ADMINISTRACIÓN

La gestión de los archivos

Nomenclatura de los archivos

No se puede dar cualquier nombre a un archivo; hay que seguir unas simples reglas, válidas para todos los tipos de archivos.

- En los sistemas actuales se puede llegar hasta 255 caracteres. La posible extensión está incluida en la longitud del nombre del archivo.
- Un punto extremamente importante: Linux respeta la distinción entre los nombres de archivos en minúsculas y en mayúsculas.
- Se acepta la mayoría de los caracteres (las cifras, las letras, las mayúsculas, las minúsculas, ciertos signos, los caracteres acentuados), incluido el espacio. Sin embargo, se deben evitar algunos caracteres, ya que tienen un significado particular dentro del shell: & ; ( ) ~ <espacio> \ / | ` ? - (al principio del nombre).

COMANDOS DE ADMINISTRACIÓN

La gestión de los archivos

Estructura y nombre de ruta

Las rutas permiten definir una ubicación en el sistema de archivos. Es la lista de los directorios y subdirectorios utilizados para acceder a un sitio determinado de la estructura hasta la posición deseada (directorio, archivo). Se suele completar un nombre de archivo con su ruta de acceso. Por eso el archivo pepito del directorio dir1 es diferente del archivo pepito del directorio dir2. Al ser jerárquico, el sistema de archivos de Unix tiene forma de estructura en árbol.

COMANDOS DE ADMINISTRACIÓN

La gestión de los archivos

Rutas absolutas

- empieza desde la raíz. Por lo tanto, comienza con una /,
- describe todos los directorios que hay que cruzar para acceder al sitio deseado,
- no contiene ni . ni ...

/home/pepito/Docs/Backup/fic.bak

Rutas relativas

- El punto . representa el directorio corriente, activo. Suele estar implícito.
- Los dos puntos .. representan el directorio de nivel superior.

Una ruta relativa:

- describe una ruta relativa a una posición determinada en la estructura, en general (pero no siempre) desde la posición actual;
- describe en principio la ruta más corta para ir de un punto a otro;
- puede contener puntos o dos puntos.

./Documents/Photos

- *Documents/Photos es una ruta relativa*
- *./Documents/Photos es una ruta relativa*
- */usr/local/../bin es una ruta relativa*

COMANDOS DE ADMINISTRACIÓN

La gestión de los archivos

Directorio personal
Al crear un nuevo usuario, el administrador le asigna un directorio personal llamado home directory. Cuando inicia sesión, el usuario es dirigido directamente a ese directorio, que es el suyo personal, en el que podrá crear sus propios archivos y subdirectorios.
La virgulilla
El bash interpreta el carácter virgulilla ~ como un alias del directorio personal. Para sacar la virgulilla por teclado es Alt Gr + 4

cd
Para desplazarse por los directorios, utilice el comando **cd** (change directory). El comando **pwd** (print working directory), que ya hemos comentado, muestra la ruta completa del directorio actual.
Si introduce **cd .**, no se mueve. El punto será muy útil cuando tenga que especificar rutas explícitas a comandos ubicados en el directorio donde está ubicado.
Cd .. sube un nivel. Si se encontraba en /home/seb, ahora estará en home.

COMANDOS DE ADMINISTRACIÓN

Los comandos básicos.

ls: Listar los archivos y los directorios
El comando **ls** permite listar el contenido de un directorio (catálogo) en líneas o columnas

Parámetro	Significado
-l	Para cada archivo o carpeta, facilita información detallada.
-a	Se visualizan los archivos escondidos (empiezan por un punto).
-d	En un directorio, precisa el propio directorio, y no su contenido.
-F	Añade un carácter al final del nombre para especificar el tipo: / para un directorio, * para un ejecutable, @ para un vínculo simbólico, etc.
-R	Si el comando detecta subdirectorios, entra en ellos de manera recursiva.
-t	Se filtra la salida por fecha de modificación del más reciente al más antiguo. Se visualiza esta fecha.
-c	Muestra/ordena (con -t) por fecha de cambio de estado del archivo.
-u	Muestra/ordena (con -t) por fecha de acceso del archivo.
-r	Se invierte el orden de salida.
-i	Muestra el inodo del archivo.
-C	La visualización se hace en varias columnas (por defecto).
-1	La visualización se hace en una sola columna.
-Z	Muestra los atributos del archivo vinculado al tipo de sistema de archivo o al contexto de seguridad (selinux).

COMANDOS DE ADMINISTRACIÓN

Los comandos básicos.

Gestionar los archivos y los directorios

1. **touch**: Crear archivos vacíos. Utilizado con el nombre de un archivo como argumento únicamente, crea un archivo con un tamaño cero. Ej: ***$ touch fictest***
2. **mkdir**: Crear directorios. Permite crear uno o varios directorios, o una estructura completa. Por defecto, el comando no crea una estructura. Si pasa como argumentos dir1/dir2 y dir1 no existe, el comando devuelve un error. En este caso, utilice el parámetro –p.: Ej: ***mkdir [-p] dir1 [dir2] ... [dirn]***
3. **rmdir**: suprime uno o varios directorios. No puede suprimir una estructura. el directorio no debe contener ni archivos ni directorios, y ello aunque los propios subdirectorios estén vacíos.: ***rmdir dir1 [dir2] ... [dirn]***
4. **rm**: Surpimir una estructura de directorios.

Parámetro	Significado
-i	El comando requerirá una confirmación para cada uno de los archivos que desea suprimir.
-r	Espera es un directorio. La supresión es recursiva: se suprimen todos los niveles inferiores, tanto los directorios como los archivos.
-f	Fuerza la supresión

COMANDOS DE ADMINISTRACIÓN

Los comandos básicos.

Gestionar los archivos y los directorios

5. **cp:** Copiar archivos. Copia uno o varios archivos en otro archivo o en un directorio.

Ej: cp fic1 [fic2 ... ficn] Destino

Parámetro	Significado
-i	Pide confirmación de copia para cada archivo.
-r	Recursivo: copia un directorio y todo su contenido.
-p	Se preservan los permisos y fechas.
-f	Forzar la copia.
-a	Copia de archivo: el destino es en la medida de lo posible idéntico al origen. La copia es recursiva.

COMANDOS DE ADMINISTRACIÓN

Los comandos básicos.

Gestionar los archivos y los directorios

6. **mv**: permite mover, volver a nombrar un archivo o las dos cosas a la vez. Funciona como el comando **cp**. Los parámetros -f e -i tienen el mismo efecto. Con los tres comandos mv sucesivos siguientes:

    - $ touch txt1 txt2 txt3 # Creamos los tres archivos
    - $ mv txt1 txt1.old # renombramos txt1 como txt1.old
    - $ mv txt2 dir1/txt2 # movemos txt2 al directorio dir1
    - $ mv txt3 dir1/txt3.old #movemos txt3 a dir y se renombra como txt3.old

COMANDOS DE ADMINISTRACIÓN

Los comandos básicos.

Los vínculos simbólicos

Puede crear vínculos simbólicos, que son un poco como atajos a un archivo. Un vínculo es un archivo especial que contiene como información la ruta hacia otro archivo. Es un tipo de alias. Existen dos tipos de vínculos:

- el vínculo duro (hard link), que veremos más adelante
- el vínculo simbólico (soft link), que corresponde a la definición dada.

Es posible crear vínculos simbólicos hacia cualquier tipo de archivo, sea cual sea y esté donde esté. El comando de creación de vínculos simbólicos no comprueba la existencia del archivo al que se apunta. Es posible crear vínculos a archivos que no existen con el parámetro -f.

ln -s archivo vínculo

Si fuera necesario, el vínculo se comportará como el archivo al que se apunta, con los mismos permisos y las mismas propiedades:

- si el archivo al que se apunta es un programa, ejecutar el vínculo lleva a ejecutar el programa;
- si el archivo al que se apunta es un directorio, un cd sobre el vínculo entra en este directorio;
- si el archivo al que se apunta es un archivo especial (periférico), se ve el vínculo como periférico;

```
$ touch fic1
$ ln -s fic1 vinculofic1
$ ls -l
-rw-r--r-- 1 seb users    0 mar  4 19:16 fic1
lrwxrwxrwx 1 seb users    4 mar  4 19:17 vinculofic1 -> fic1
$ ls -F
fic1  vinculofic1@
$ echo titi>fic1
$ cat vinculofic1
titi
```

COMANDOS DE ADMINISTRACIÓN

Los comandos básicos.

Comodines: carácter de sustitución
Al utilizar los comandos con el sistema de archivos, puede resultar interesante filtrar la salida de nombres de archivos con ayuda de determinados criterios, por ejemplo con el comando **ls**. En vez de visualizar toda la lista de archivos, se puede filtrar la visualización de varios criterios y caracteres especiales.

Caracteres(s)	Función
*	Sustituye una cadena de longitud variable, incluso vacía.
?	Sustituye cualquier carácter único.
[...]	Una serie o un rango de caracteres.
[a-b]	Un carácter entre el rango indicado (de a a b incluida).
[!...]	Inversión de la búsqueda.
[^...]	Ídem.

```
$ ls
afic   afic2  bfic   bfic2  cfic   cfic2  dfic   dfic2
afic1  afic3  bfic1  bfic3  cfic1  cfic3  dfic1  dfic3
```

```
$ ls a*
afic1  afic2  afic3
```

```
$ ls a???
afic
```

```
$ ls b??*
bfic  bfic1  bfic2  bfic3
```

```
$ ls *[12]
afic1  afic2  bfic1  bfic2  cfic1  cfic2  dfic1  dfic2
```

COMANDOS DE ADMINISTRACIÓN

Los comandos básicos.

Cierre de caracteres
Se deben cerrar algunos caracteres especiales; por ejemplo, en caso de caracteres poco corrientes en un nombre de archivo.

- La **contrabarra ** permite cerrar un carácter único. ls paga\ *.xls va a listar todos los archivos que contienen un espacio después de paga.
- Las **comillas** "..." permiten la interpretación de los caracteres especiales, de las variables, dentro de una cadena.
- Los **apóstrofes** ’...’ cierran todos los caracteres especiales en una cadena o un archivo.

COMANDOS DE ADMINISTRACIÓN

Los comandos básicos.

Buscar archivos

Find: permite buscar archivos dentro de la estructura del sistema de archivos con la ayuda de criterios, y da la posibilidad de actuar sobre los resultados devueltos.

```
find ruta criterios opciones
```

-**name** permite una selección por nombres de archivos
-type permite una selección por tipo de archivos.

```
$ find . -name "re*" -tipo d -print
./dir1
./dir2
```

-user y -group permiten una búsqueda sobre el propietario y el grupo de pertenencia de los archivos.

Código	Tipo de archivo
b	Archivo especial en modo bloque
c	Archivo especial en modo carácter
d	Directorio (directory)
f	Archivo ordinario
l	Vínculo simbólico
p	Tubería con nombre (pipe)
s	Socket (Conexión de red)

COMANDOS DE ADMINISTRACIÓN

Los comandos básicos.

Buscar archivos

Find: permite buscar archivos dentro de la estructura del sistema de archivos con la ayuda de criterios, y da la posibilidad de actuar sobre los resultados devueltos.

-**atime**: busca en la fecha del último acceso (access time). Un acceso puede ser la lectura del archivo.
-**mtime**: busca en la fecha de la última modificación (modification time). Se trata de la modificación del contenido.
-**ctime**: busca en la fecha de modificación (change time, en realidad la fecha de última modificación del número de inodo).

.

```
root@client log]# find . -mtime -1
./lastlog
```

-**perm** permite efectuar búsquedas en las autorizaciones de acceso (derechos, SUID, SGID, Sticky)
-**regex** devolverá los valores correspondientes a las expresiones regulares proporcionadas.
-**iregex** hace lo mismo sin discriminar entre mayúsculas y minúsculas.

COMANDOS DE ADMINISTRACIÓN

Los comandos básicos.

Buscar archivos

Find: permite buscar archivos dentro de la estructura del sistema de archivos con la ayuda de criterios, y da la posibilidad de actuar sobre los resultados devueltos.

Comandos

Además de la opción -print, hay otras opciones que permiten efectuar una acción en los archivos encontrados.

-ls: El criterio muestra información detallada en los archivos encontrados que corresponden al criterio, en lugar del simple nombre de archivo.

-exec: El criterio -exec va a ejecutar el comando colocado justo después con cada coincidencia encontrada.

-ok: El criterio -ok es idéntico a la opción -exec, pero, para cada coincidencia, se le requiere una confirmación al usuario

Criterios AND / OR / NOT

Es posible combinar las opciones de criterio de selección. Si no se especifica ninguno, el Y lógico está implícito.

Criterio	Acción
-a, -and	AND, Y lógico, por defecto
-o, -or	OR, O lógico
!	Negación del criterio

COMANDOS DE ADMINISTRACIÓN

Los comandos básicos.

Encontrar ejecutables
Whereis: busca en las rutas de archivos binarios, del manual y de las fuentes los archivos que corresponden a los criterios facilitados. Puede precisar algunos parámetros:

- -b únicamente para los binarios,
- -m únicamente para los manuales,
- -s únicamente para las fuentes.

Which: busca un comando en el PATH (ruta de los ejecutables) y le facilita el primero que encuentra,
Locate: busca un archivo según el modelo dado en una base de datos de archivos construida por el comando **updatedb**. El comando **updatedb** recorre una serie de rutas en las que ejecuta un **find** y almacena todos los resultados en una base indexada. Esto evita, por lo tanto, efectuar de nuevo un **find** para las búsquedas clásicas.

COMANDOS DE ADMINISTRACIÓN

Redirecciones

Fundamentos
Una redirección es la posibilidad de redireccionar la visualización de la pantalla hacia un archivo, una impresora o cualquier otro periférico, los mensajes de errores hacia otro archivo, de sustituir la introducción vía teclado por el contenido de un archivo.

De salida
Se puede utilizar el carácter **>** para redireccionar la salida estándar (la que va normalmente en la pantalla). Luego se indica el nombre del archivo donde se colocarán los resultados de salida.
El carácter **>>** es para añadir datos al fichero de salida.

```
$ ls -l > resultado.txt
$ date >> resultado.txt
$ cat resultado.txt
total 1
-rw-r--r--    1 Administ ssh_user        0 Jul  4 12:04 PEPITO
-rw-r--r--    1 Administ ssh_user        0 Jul 25 15:13 resultado.txt
-rw-r--r--    1 Administ ssh_user      171 Jul 25 15:13 test.txt
Thu Jul 25 15:20:12  2002
```

COMANDOS DE ADMINISTRACIÓN

Redirecciones

Tipos de canales estándar:

Canal 0 : Canal con descriptor 0; se denomina stdin y lleva descriptor 0
Canal 1 : Canal con descriptor 1; se denomina stdout y lleva descriptor 1
Canal 2 : Canal con descriptor 2; se denomina stderr y lleva descriptor 2

Se puede utilizar el carácter **>&** para redireccionar los dos canales a un mismo archivo.

Por ejemplo: ***ls –l >resultado.txt 2>&1***
[se redirecciona la salida 2 a la salida 1; por lo tanto, los mensajes de error pasarán por la salida estándar, luego se redirecciona la salida estándar a resultado.txt.

COMANDOS DE ADMINISTRACIÓN

Redirecciones

En entrada
Los comandos que esperan datos o parámetros desde el teclado pueden también recibirlos desde un archivo usando el carácter <. Un ejemplo con el comando **wc** (word count), que permite contar el número de líneas, de palabras y de caracteres de un archivo

Documento en línea
La redirección << es algo particular. Permite la utilización de los documentos en línea. Encontrará a veces el término Herescript o Here Document. Esto permite la inserción de un texto hasta un punto dado y el envío de su resultado a un comando o un filtro. Se autorizan las redirecciones clásicas. Después del <<, ha de indicar una cadena que define el final de la inserción; por ejemplo, aquí 'end'.

```
$ tr "[a-z]" "[A-Z]" << end
> hola amigos
> esto es un ejemplo
> de herescript
> end
HOLA AMIGOS
ESTO ES UN EJEMPLO
DE HERESCRIPT
```

```
$ cat <<< toto
toto
```

COMANDOS DE ADMINISTRACIÓN

PRACTICA 2 DE CLASE TEMA 4

COMANDOS DE ADMINISTRACIÓN

Filtros y herramientas

Un **filtro** (o un comando filtro) es un programa que sabe escribir y leer datos por los canales estándares de entrada y salida.

- **basename** permite extraer el nombre del archivo en una ruta.
- **dirname** efectúa lo contrario, extrae la ruta.

Búsqueda de líneas

Se trata de extraer líneas de un archivo según varios criterios.

grep

La sintaxis del comando **grep** es: **grep [Opciones] modelo [Archivo1...]**

El comando **grep** también puede tomar algunas opciones interesantes.

- **-v** efectúa la búsqueda inversa: se visualizan todas las líneas que no corresponden a los criterios.
- **-c** sólo devuelve el número de líneas encontradas, sin mostrarlas.
- **-i** no diferencia las mayúsculas de las minúsculas.
- **-n** indica el número de línea para cada línea encontrada.
- **-l** en el caso de archivos múltiples, indica en qué archivo se ha encontrado la línea.

```
$ grep -i "^b" fic4
Buey
buey
```

COMANDOS DE ADMINISTRACIÓN

Filtros y herramientas

Egrep: extiende los criterios de búsqueda y puede aceptar un archivo de criterios en entrada. Equivale a un grep -E. Emplea como criterios expresiones regulares.

egrep -f archivo_criterio archivo_búsqueda

echo $IP | egrep '([0-9]{1,3}\.){3}[0-9]{1,3}'

Esta línea se descompone de la manera siguiente:

- ([0-9]{1,3}\.){3}: www.xxx.yyy.
- [0-9]: un carácter entre 0 y 9
- {1,3}: repetido entre una y tres veces, por lo tanto: x, xx o xxx
- \.: seguido de un punto
- {3}: el conjunto tres veces
- Luego [0-9]{1,3}: .zzz
- [0-9]: un carácter entre 0 y 9
- {1,3}: repetido entre una y tres veces

Carácter especial	Significado
|	O lógico, la expresión colocada antes o después debe desaparecer.
(...)	Agrupación de caracteres.
[...]	Un carácter tiene esta posición entre los indicados.
. (punto)	Cualquier carácter.
+	Repetición, el carácter colocado antes debe aparecer al menos una vez.
*	Repetición, el carácter colocado antes debe aparecer de cero a n veces.
?	El carácter colocado antes debe aparecer una vez como máximo.
{n}	El carácter colocado antes debe aparecer exactamente n veces.
{n,}	Aparece n veces o más.
{n,m}	Aparece entre n y m veces.
^	En principio de cadena.
$	En final de cadena.

COMANDOS DE ADMINISTRACIÓN

Filtros y herramientas

Fgrep: es un grep simplificado y rápido (fast grep) y equivale a un grep -F. Acepta también un archivo de criterios de búsqueda, pero debe tratarse de criterios simples, sin caracteres especiales. Introduzca en el archivo de criterios líneas sencillas (texto y cifras), una búsqueda por línea. Fgrep va a buscar en un archivo meta o un flujo en entrada las líneas que corresponden a cada uno de los criterios.
Sed: Sed es un editor de flujo (Stream Editor) que permite filtrar y transformar texto. Es un poco como un editor que permite modificar texto vía comandos de scripts, pero en un paso y sin edición interactiva.

sed -e '<cmd>' arch

Sed se utiliza muy a menudo para sustituir valores por otros (sustitución) o suprimir líneas particulares (aunque se podría utilizar grep en este caso). La sintaxis básica de sustitución es la siguiente:

s/<antiguo>/<nuevo>/[g]

a g final permite realizar una sustitución en toda la línea en caso de haya varias coincidencias. Aquí tiene un ejemplo que sustituye __NOMBRE__ por Pepito:

```
$ echo "Me llamo __NOMBRE__. ¿Te llamas __NOMBRE__?" | sed -e 's/__
NOMBRE__/Pepito/'
Me llamo Pepito. ¿Te llamas __NOMBRE?
$ echo "Me llamo __NOMBRE__. ¿Te llamas __NOMBRE__?" | sed -e 's/__
NOMBRE__/Pepito/g'
Me llamo Pepito. ¿Te llamas Pepito?
```

COMANDOS DE ADMINISTRACIÓN

Columnas y campos

El comando **cut** permite seleccionar columnas y campos en un archivo.
Columnas
La sintaxis es la siguiente:

cut -cColumnas [fic1...]

Una columna es la posición de un carácter en la línea. El primer carácter es la columna 1; el segundo, la columna 2, y así sucesivamente. Una línea de 80 caracteres dispone de 80 columnas. La numeración empieza en 1. Es el método ideal para archivos planos y con formato fijo, donde cada campo empieza y termina con posiciones dadas.

El formato de selección de columna es el siguiente:

- una columna sola (p. ej. -c2 para la columna 2);
- un intervalo (p. ej. -c2-4 para las columnas 2, 3 y 4);
- una lista de columnas (p. ej. -c1,3,6 para las columnas 1, 3 y 6);
- los res a la vez (p. ej. -c1-3,5,6,12-).

```
$ cat lista
Producto  precio  cantidades
ratón     30      15
disco     100     30
pantalla  300     20
teclado   45      30

$ cut -c1-5 lista
Produ
ratón
disco
panta
tecla
```

COMANDOS DE ADMINISTRACIÓN

Columnas y campos

El comando **cut** permite seleccionar columnas y campos en un archivo.
Columnas
La sintaxis es la siguiente:

cut -cColumnas [fic1...]

Una columna es la posición de un carácter en la línea. El primer carácter es la columna 1; el segundo, la columna 2, y así sucesivamente. Una línea de 80 caracteres dispone de 80 columnas. La numeración empieza en 1. Es el método ideal para archivos planos y con formato fijo, donde cada campo empieza y termina con posiciones dadas.

El formato de selección de columna es el siguiente:

- una columna sola (p. ej. -c2 para la columna 2);
- un intervalo (p. ej. -c2-4 para las columnas 2, 3 y 4);
- una lista de columnas (p. ej. -c1,3,6 para las columnas 1, 3 y 6);
- los res a la vez (p. ej. -c1-3,5,6,12-).

```
$ cat lista
Producto  precio  cantidades
ratón     30      15
disco     100     30
pantalla  300     20
teclado   45      30

$ cut -c1-5 lista
Produ
ratón
disco
panta
tecla
```

COMANDOS DE ADMINISTRACIÓN

Columnas y campos

Campos
El comando **cut** también permite seleccionar campos. Se deben delimitar estos campos por defecto por una tabulación, pero el parámetro -d permite seleccionar otro carácter (espacio, ;). La selección de los campos es idéntica a la de las columnas. Con –d escribimos el carácter separador de los campos.

cut -dc -fCampos [fic1...]

En el ejemplo podemos ver el archivo group que incluye los campos delimitados por :
escribimos el comando cut
cut –d: -f1,3 /etc/group

decimos que el archivo delimitador del campo es : con -d: y
que nos visualice los campos 1 y 3

```
$ cat /etc/group
seb@slyserver:~> cat /etc/group
at:!:25:
audio:x:17:
avahi:!:106:
beagleindex:!:107:
bin:x:1:daemon
cdrom:x:20:
console:x:21:
daemon:x:2:
dialout:x:16:seb,esteban,enrique,public
disk:x:6:

$ cut -d: -f1,3 /etc/group
at:25
audio:17
avahi:106
beagleindex:107
bin:1
cdrom:20
console:21
```

COMANDOS DE ADMINISTRACIÓN

Recuento de líneas

wc (word count) permite contar las líneas, las palabras y los caracteres.

wc [-l] [-c] [-w] [-w] fic1

- -l: cuenta el número de líneas
- -c: cuenta el número de bytes
- -w: cuenta el número de palabras
- -m: cuenta el número de caracteres

```
$ wc lista
        12        48       234 lista
```

El archivo lista contiene 12 líneas, 48 palabras y 234 caracteres.

COMANDOS DE ADMINISTRACIÓN

Ordenación de líneas

El comando **sort** permite ordenar las líneas. Por defecto, la ordenación se hace sobre toda la tabla en orden creciente. La ordenación es posible a partir de uno o varios campos. El separador de campos por defecto es la tabulación o, al menos, un espacio. Si hay varios campos, el primero es el separador; los demás son caracteres del campo.

Opción	Función
-d	Dictionnary sort (ordenación de diccionario). Sólo toma como criterio de ordenación las letras, las cifras y los espacios.
-n	Ordenación numérica, ideal para las columnas de cifras.
-b	Ignora los espacios al principio del campo.
-f	No hay diferencias entre mayúsculas y minúsculas (conversión en minúsculas y luego ordenación).
-r	Reverse, ordenación en orden decreciente.
-tc	Nuevo delimitador de campo c.

COMANDOS DE ADMINISTRACIÓN

Eliminación de las líneas repetidas

El comando **uniq** permite suprimir las líneas repetidas en flujos de entrada o archivos ordenados. Por ejemplo, a continuación se muestra cómo sacar únicamente la lista de los GID realmente utilizados como grupo principal de usuarios

```
$ cut -d: -f4 /etc/passwd | sort -n | uniq
0
```

Unión de dos archivos

El comando **join** permite efectuar la unión de dos archivos en función de un campo común. Se deben ordenar los dos archivos en los campos especificados en la unión.

join [-tc] [-1 n] [-2 m] fic1 fic

La opción -t indica los separadores, -1 el campo del primer archivo y -2 el campo del segundo archivo, en los cuales efectuar la unión.

COMANDOS DE ADMINISTRACIÓN

Línea a línea

El comando **paste** agrupa n archivos en uno. Para ello, concatena las líneas de cada uno de los archivos en una sola línea: línea1 de fic1 con línea2 de fic2, línea3 de fic 3, y así sucesivamente. Es un poco lo contrario de cut. El separador por defecto es la tabulación, pero puede precisar un delimitador con -d.

```
$ cat fic1
lista_a
lista_b
lista_c

$ cat fic2
lista_a2
lista_b2
lista_c2

$ paste -d: fic1 fic2
lista_a:lista_a2
lista_b:lista_b2
lista_c:lista_c2
```

COMANDOS DE ADMINISTRACIÓN

División de un archivo en partes

Recortar

Aquí tenemos un comando muy práctico, **split**, que permite recortar un gran archivo en varios trozos con un tamaño determinado. Los sistemas de archivos no son todos iguales frente al tamaño máximo de un archivo. En Linux, el problema no es habitual, ya que un sistema de archivos de tipo ext4 puede soportar archivos de varias decenas de TB. Pero las bandas magnéticas, o en menor medida los discos removibles, no disponen de esta posibilidad.

split [-l n] [-b n[bkm] [archivo [prefijo]]

El comando puede funcionar según dos modos:

- recorte por líneas con -l: los archivos de salida tendrán todos n líneas de texto
- recorte a tamaño fijo con -b: los archivos tendrán todos un tamaño fijo de n bytes

COMANDOS DE ADMINISTRACIÓN

División de un archivo en partes

Recortar

Aquí tenemos un comando muy práctico, **split**, que permite recortar un gran archivo en varios trozos con un tamaño determinado. Los sistemas de archivos no son todos iguales frente al tamaño máximo de un archivo. En Linux, el problema no es habitual, ya que un sistema de archivos de tipo ext4 puede soportar archivos de varias decenas de TB. Pero las bandas magnéticas, o en menor medida los discos removibles, no disponen de esta posibilidad.

split [-l n] [-b n[bkm] [archivo [prefijo]]

El comando puede funcionar según dos modos:

- recorte por líneas con -l: los archivos de salida tendrán todos n líneas de texto
- recorte a tamaño fijo con -b: los archivos tendrán todos un tamaño fijo de n bytes

```
$ ls -l granarchivo
-rw-r--r-- 1 seb users 1073741824 mar 12 19:47 granarchivo
$ split -b 150m granarchivo fic
$ ls -l fic*
-rw-r--r-- 1 seb users 157286400 mar 12 20:15 ficaa
-rw-r--r-- 1 seb users 157286400 mar 12 20:15 ficab
-rw-r--r-- 1 seb users 157286400 mar 12 20:15 ficac
-rw-r--r-- 1 seb users 157286400 mar 12 20:16 ficad
-rw-r--r-- 1 seb users 157286400 mar 12 20:16 ficae
-rw-r--r-- 1 seb users 157286400 mar 12 20:16 ficaf
-rw-r--r-- 1 seb users 130023424 mar 12 20:16 ficag
```

COMANDOS DE ADMINISTRACIÓN

Reconstruir

Una línea basta para reconstruir un archivo dividido con la ayuda de las redirecciones:

```
$ cat fic* > newfic
$ ls -l newfic
-rw-r--r-- 1 seb users 1073741824 mar 12 20:47 newfic
```

COMANDOS DE ADMINISTRACIÓN

Sustitución de caracteres

Lista de caracteres

El comando **tr** permite sustituir unos caracteres con otros y sólo acepta datos que provengan del canal de entrada estándar, no de los archivos.

Tabulaciones y espacios

La mayoría de los editores sustituyen las tabulaciones por espacios. Ahora bien, algunos comandos esperan a obtener tabulaciones como delimitadores de campos (es el caso de cut). Si no puede apañarse con tr, tiene a su disposición dos comandos para este caso específico.

El comando **expand** convierte las tabulaciones en espacios (muy útil para el archivo yaml). El comando **unexpand** convierte los espacios en tabulaciones.

xargs

El comando **xargs** permite leer los elementos de la entrada estándar (pipe, redirección), delimitados por defecto por un espacio o un retorno de línea, y luego ejecutar un comando, por defecto **echo**, con esos mismos elementos, uno por uno o reformateados.

COMANDOS DE ADMINISTRACIÓN

Visualización de texto

En pantalla completa

Nada impide desviar cualquier flujo para visualizarlo en la pantalla o por impresora. Aquí presentamos algunos comandos.

- página por página: **pg**, **more**, **less**
- en bloque: **cat**
- al revés: **tac**
- en dump hexadecimal: **hexdump**
- en dump octal (por defecto): **od**
- creación de un banner: **banner**
- formateo para impresión: **pr**
- formateo de párrafo: **fmt**
- numerar las líneas: **cat -n** o **nl**

COMANDOS DE ADMINISTRACIÓN

El principio de un archivo

Para ver el principio del contenido de un archivo, utilice el comando **head**.

head [-c nbcar] [-n nblíneas] [fic1...]

El parámetro -c permite precisar el número de bytes de encabezamiento que visualizar. Por defecto se visualizan diez líneas.
El parámetro -n permite indicar el número de líneas que visualizar. Es posible indicar directamente el número de líneas:

Fin y modo de espera de archivo

Para ver las últimas líneas de un archivo, utilice el comando **tail**.

tail [+/-valor[b/c]] [-f] [fic1...

Dar formato a una salida

El comando **column** permite dar formato de tabla a la salida de un comando. La opción -t determina cuántas columnas se mostrarán en la salida y añade espacios para alinearlas. La opción -s permite indicar cuál es el separador.

COMANDOS DE ADMINISTRACIÓN

Comparación de archivos

Los dos comandos que permiten comparar el contenido de dos archivos, o de un archivo y de un flujo, son los comandos **diff** y **cmp**.

a. **Diff**
b. El comando **diff** indica las modificaciones que hay que aportar a los dos archivos en entrada para que su contenido sea idéntico. ***diff [-b] [-e] fic1 fic2***

La opción -b permite ignorar los espacios (blank), y la opción -e permite generar un script ed (no lo utilizaremos). Este comando devuelve tres tipos de mensajes:

- APPEND: línea1 a línea3,línea4, ex 5 a 6,8 significa: en la línea 5 de fic1 hay que insertar las líneas 6 a 8 de fic2 para que sus contenidos sean idénticos.
- DELETE: línea1,línea2 d línea3, ex 7,9 d 6 significa: se deben suprimir las líneas 7 a 9 de fic1, no existen detrás de la línea 6 de fic2.
- CHANGE: línea1,línea2 c línea3,línea4, ex 8,12 c 9,13 significa: se debe intercambiar las líneas 8 a 12 de fic1 contra las líneas 9 a 13 de fic2.

cmp

El comando **cmp** compara los archivos carácter por carácter. Por defecto, el comando se para en cuanto encuentra la primera diferencia e indica la posición. cmp [-l] [-s] fic1 fic2

COMANDOS DE ADMINISTRACIÓN

PRACTICA 3 DE CLASE TEMA 4

COMANDOS DE ADMINISTRACIÓN

LOS PROCESOS

Un **proceso** representa un programa en curso de ejecución y al mismo tiempo todo su entorno de ejecución (memoria, estado, identificación, propietario, padre…)
Los datos de identificación de un proceso son:

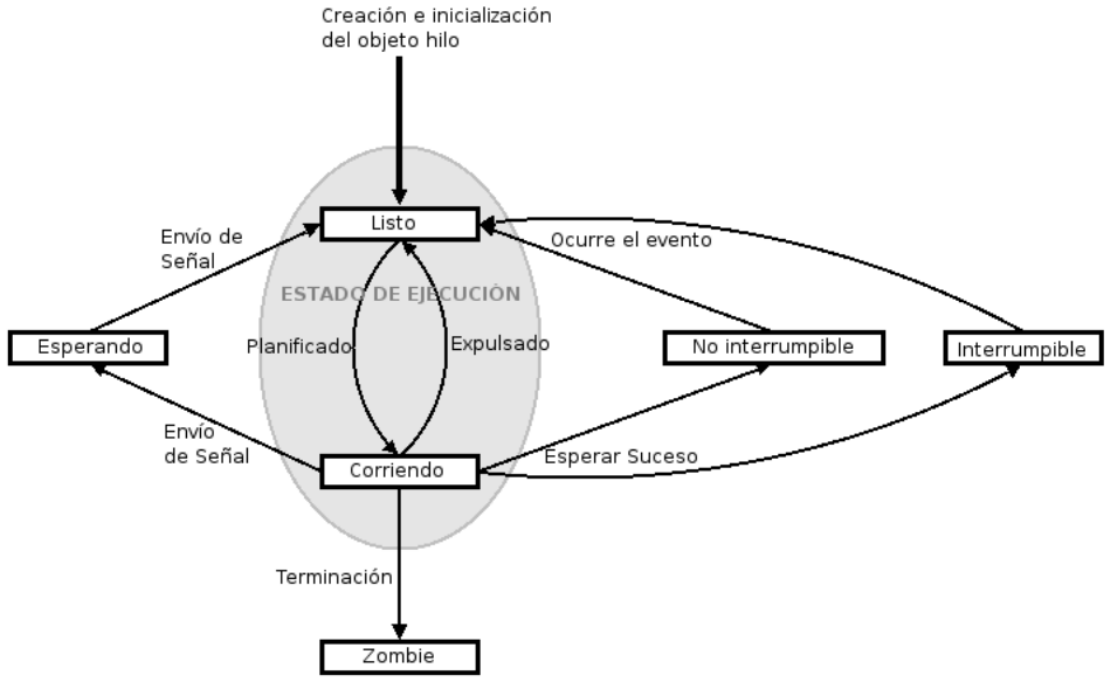
- **Número de proceso único (PID, process ID**): se numera cada proceso con el fin de poder diferenciarlos de los otros. El primer proceso iniciado por el sistema es el 1.
- **Número de proceso padre (PPID, parent process ID):** cada proceso puede iniciar otros procesos, sus procesos hijos. Cada proceso hijo debe contener el PID del proceso padre que lo inició.
- **Un número de usuario y uno de grupo**: UID y GID de la cuenta de usuario que inicia el proceso.
- **Duración y prioridad del proceso**: la duración del proceso corresponde al tiempo de ejecución consumido desde la última invocación. La prioridad de un proceso aumenta cuando está inactivo y baja al estar en ejecución para dar paso a otros procesos.
- **Directorio del trabajo activo**: Directorio actual de proceso para poder resolver direcciones relativas.
- **Archivos abiertos**: tabla de los descriptores de archivos abiertes. Por defecto al principio hay tres abiertos: 0,1 y 2 (los canales estándar)
- Tamaño de memoria asignada, la fecha de inicio del proceso, el terminal de atribución, el estado del proceso, los UID efectivo y real así como los GID efectivos y reales.

COMANDOS DE ADMINISTRACIÓN

LOS ESTADOS DE UN PROCESO

Durante su vida (tiempo entre el inicio y la salida) un proceso puede pasar por diversos estados o process state:

- Ejecución en modo usuario (user mode)
- Ejecución en modo núcleo (kernel mode)
- En espera E/S (waiting)
- Dormido (sleeping)
- Listo para la ejecución (runnable)
- Domido en el swap (memoria virtual)
- Nuevo proceso
- Fin de proceso (zombie)

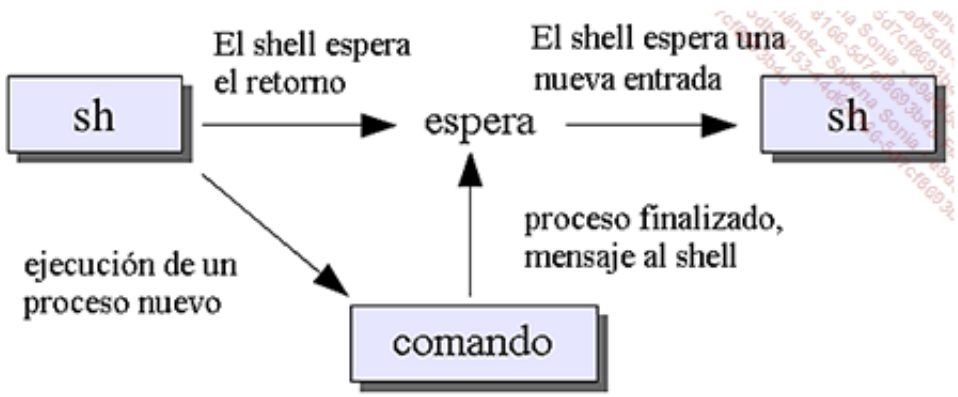


COMANDOS DE ADMINISTRACIÓN

PROCESOS EJECUCIÓN EN SEGUNDO PLANO

La Shell en sí es un proceso, cuando insertamos un comando se convierte en un proceso hijo de la Shell. Una vez iniciado, hay que esperar al final de su ejecución para iniciar el siguiente.
Nada impide al Shell esperar el mensaje de proceso terminado para dejar paso.
Si queremos que la Shell de paso al comando introducido sin esperar el término del anterior deberemos utilizar el comando & al final.
Existen unas pautas para el uso de segundo plano:

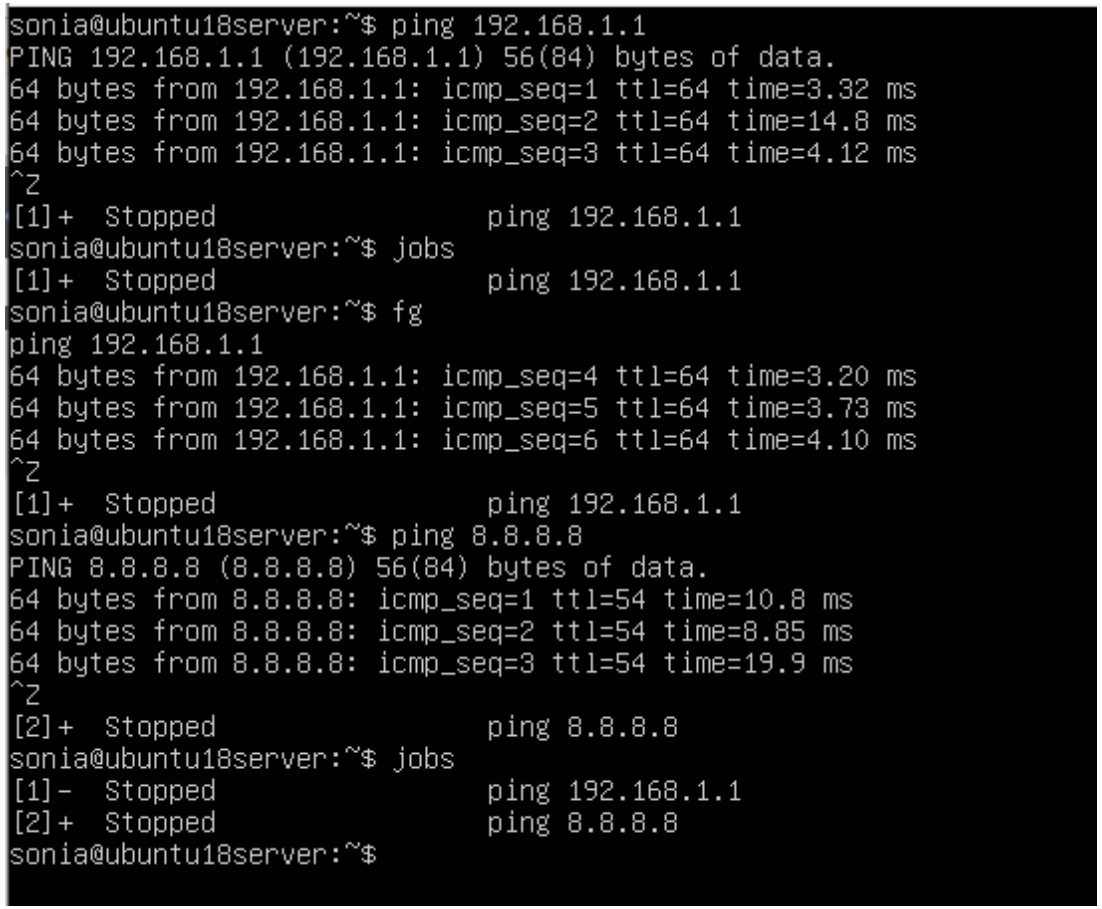
- El segundo comando para trabajar en paralelo no podrá ser otra Shell.
- El proceso iniciado no deberá mostrar resultados en la pantalla, por que entraría en conflicto con los de la Shell.
- Cuando se sale del Shell, se sale también de todos los procesos hijos, no cerrarla a la mitad de algún proceso en segundo plano.



COMANDOS DE ADMINISTRACIÓN

PROCESOS: BACKGROUND, FOREGROUND, JOBS

- **[Ctrl]+Z**: Parar el control temporal de un proceso.
- **fg**: después de suspender un proceso, volver a ponerlo en primer plano.
- **Jobs**: lista de tareas
- **bg**: se ejecuta en un job parado para iniciarlo de nuevo en segundo plano.



COMANDOS DE ADMINISTRACIÓN

PROCESOS: LISTA DE LOS PROCESOS

- **ps**: (process status) información sobre procesos en curso. Tiene las siguientes opciones:
  - si se ejecuta solo, visualiza los procesos en curso iniciados por el usuario.
  - **-f** : mas información
  - -**ef** : sólo los procesos en curso
  - -**u** (usuario): filtra por usuario
  - -**g**: filtra por grupo
  - -**t**: filtra por terminal
  - -**p**: filtra por PID
  - **-l** : muestra información más técnica

```
sonia@ubuntu18server:~$ ps
   PID TTY          TIME CMD
  1368 tty1     00:00:00 bash
  1380 tty1     00:00:00 ping
  1381 tty1     00:00:00 ping
  1469 tty1     00:00:00 ps
sonia@ubuntu18server:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
sonia       1368     954  0 12:20 tty1     00:00:00 -bash
sonia       1380    1368  0 12:21 tty1     00:00:00 ping 192.168.1.1
sonia       1381    1368  0 12:22 tty1     00:00:00 ping 8.8.8.8
sonia       1470    1368  0 12:39 tty1     00:00:00 ps -f
```

Columna	Definición
UID	User ID, nombre del usuario.
PID	Process ID, número del proceso.
PPID	Parent Process ID, número del proceso padre.
C	Factor de prioridad, cuanto más grande es el valor, más elevada es la prioridad.
STIME	Hora de inicio del proceso.
TTY	Nombre del terminal desde el cual se ejecutó el proceso.
TIME	Duración de tratamiento del proceso.
CMD	Comando ejecutado.
F	Banderas del proceso (sale del ámbito de este libro).
S	Estado del proceso S (sleeping) R (running) Z (zombie).
PRI	Prioridad del proceso.
NI	Nice, incremento para el scheduler.

COMANDOS DE ADMINISTRACIÓN

PROCESOS: LISTA DE LOS PROCESOS

fuser: Nos permite identificar que procesos están haciendo uso determinados archivos, *sockets* o directorios, o dicho de otra manera por que procesos están siendo controlados determinados archivos. También nos permite matar procesos.

fuser [opciones] archivo

Opciones:

- Mostrar los procesos para todos los archivos pasados desde la línea de comandos: -a, –all
- Mata un proceso: -k
- Activa el modo interactivo: -i
- Muestra mas información: -v
- Mostrar todas las señales conocidas -l
- Matar solo procesos que tienen acceso de escritura: -w
- Activar el modo silencioso: -s
- Añade el nombre del usuario propietario para cada PID: -u, –user

COMANDOS DE ADMINISTRACIÓN

PROCESOS: LISTA DE LOS PROCESOS

top : Muestra un resumen de la información del sistema y de los procesos del ordenador que mas CPU consumen.

$top -opciones

[Opciones]:

- Especificar un retardo diferente al de por defecto (5s) para la actualización de la info: **-d**
- Monitorizar procesos específicos (hasta 20): **-p** *PID*
- Mostrar un número concreto de actualizaciones y luego cerrarse: **-n** *num_iteraciones*
- Utilizar un archivo para recoger el uso de CPU de determinados programas: **-b**

[Teclas para interaccionar con top]:

- Mostrar información de ayuda: **h**, **y**, **?**
- Destruir un proceso conociendo su PID: **k**
- Cambiar la prioridad de un proceso: **r**
- Modificar el intervalo de refresco de información: **s**
- Ordenar los datos mostrados por uso de CPU (top por defecto actúa así): **P**
- Ordenar los datos por uso de memoria: **M**
- Salir de top: **q**

COMANDOS DE ADMINISTRACIÓN

PROCESOS: LISTA DE LOS PROCESOS

uptime : Nos permite conocer el tiempo que lleva el sistema sin ser reiniciado, así como la carga media (load average).

$uptime

Nota: El comando *uptime* desplegará una línea con 4 campos: la hora actual, el tiempo que lleva el sistema iniciado sin ser reiniciado, el número de usuarios conectados y el promedio de carga del sistema en los últimos 1, 5 y 15 minutos. También podemos ver el tiempo sin reinicio del sistema en segundos desplegando *$cat /proc/uptime* o el promedio de carga a través del archivo */proc/loadavg*

COMANDOS DE ADMINISTRACIÓN

PROCESOS: LISTA DE LOS PROCESOS

free : Muestra la cantidad de memoria física y de intercambio libre y usada en el sistema, así como el búfer utilizado por el kernel.

$free [opciones]

[Opciones]:

- Mostrar la memoria en bytes, kilobytes, megabytes o gigabytes: **-b -k -m -g** (*respectivamente*)
- Mostrar la cantidad de memoria en unidades (B, M, K, G o T) directamente abreviadas según la cantidad: **-h**
- Mostrar la salida del comando durante un tiempo deseado: **-s** *<num_seg>*
- Mostrar la salida de *free* tantas veces como deseemos. Es necesario acompañarlo de la opción s: **-c** *<num_veces>*

COMANDOS DE ADMINISTRACIÓN

PROCESOS: LISTA DE LOS PROCESOS

& : Si lo añadimos al final de la línea de ejecución del comando nos permitirá que este se ejecute en segundo plano, dejando libre la terminal.
wait : Nos permite elegir cuando queremos retomar el control de la terminal en el caso de que haya varios programas en background.

$wait

[Opciones]:

- Si queremos esperar a que terminen todos los trabajos, no pasaremos ninguna opción.
- Si queremos retomar el control cuando termine el trabajo [2]: **%**2
- Si queremos retomar el control cuando termine el proceso 4563: <num_proc>

nohup : Permite la continuidad en la ejecución de un programa/comando aun habiendo cerrado la terminal desde la que se ejecutó.

$nohup [comando/programa] [opciones comando/programa]

COMANDOS DE ADMINISTRACIÓN

PROCESOS: LISTA DE LOS PROCESOS

screen : Crea terminales virtuales para poder ejecutar un comando pudiendo cerrar la ventana y que este siga ejecutándose, además poder abrir de nuevo esa misma terminal virtual y retomar el control.

$screen [opciones]

[Opciones]:

- Si escribimos ‘screen’ a secas, abriremos una nueva terminal virtual
- Si queremos listar las terminales virtuales existentes en el equipo: **-ls**
- Para retomar el control de una determinada terminal virtual: **-r** *ID*
- **Nota**: Si solo hay una terminal virtual abierta, basta con escribir screen *-r*

COMANDOS DE ADMINISTRACIÓN

PROCESOS: PARADA DE UN PROCESO/SEÑALES

- **kill :** manda señales al kernel para actuar con los procesos

Nota: El comando *kill* también acepta el parámetro *%id_job* para pasar una señal a determinado trabajo. **$kill %3**

$kill [parametro] [señal]

- Para indicar la señal por su nombre completo: **-s**
- Para indica la señal por su nombre (sin SIG): **–*señal***
- Para pasarle el número de señal: **–*num***

Señal	Función
1 (SIGHUP)	El padre manda Hang Up a todos sus hijos cuando termina.
2 (SIGINT)	Interrupción del proceso pedido (tecla [Supr], [Ctrl] **C**).
3 (SIGQUIT)	Ídem SIGINT, pero con generación de un Core Dump (archivo de depuración).
9 (SIGKILL)	Señal que no se puede ignorar, fuerza el proceso a terminar de manera «expeditiva».
15 (SIGTERM)	Señal mandada por defecto por el comando **kill**. Pide al proceso terminar con normalidad.

```
sonia@ubuntu18server:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

COMANDOS DE ADMINISTRACIÓN

PROCESOS: PARADA DE UN PROCESO/SEÑALES

- **killall** : Nos permite matar un proceso por su nombre y no por su *PID*. También podemos pasarles señales como a *kill*. Existe una variante de Unix que mata todos los procesos iniciados por un determinado usuario, en caso de pasarle como argumento el nombre del usuario.

```
$killall –i vi
```

[Opciones]:

- Para pedir confirmación de finalización de proceso: **-i**
- Todos los procesos de un determinado usuario: **-u**
- Todos los procesos de un determinado grupo: **-g**
- Incluir expresión regular para la búsqueda de los procesos: **-r**
- Mandar una señal de finalización: **-s**

)

COMANDOS DE ADMINISTRACIÓN

PROCESOS: PARADA DE UN PROCESO/SEÑALES

- **Pgrep:** listados de comandos a los que se puede enviar una señal con pkill.

Ejemplo: El comando sleep se utiliza para temporizar un intervalo de tiempo determinado.

ejemplo $ sleep 100 & (temporiza 100 segundos y en multiproceso)

```
sonia@ubuntu18server:~$ sleep 100&
[5] 24330
sonia@ubuntu18server:~$ sleep 100 &
[6] 24331
sonia@ubuntu18server:~$ sleep 100 &
[7] 24332
sonia@ubuntu18server:~$ pgrep sleep
24329
24330
24331
24332
```

)

COMANDOS DE ADMINISTRACIÓN

PROCESOS: PARADA DE UN PROCESO/SEÑALES

- **Pkill:** la señal no se envia al PID indicado, sino a los comandos, se pasa un nombre de programa como patrón, un usuario o grupo de usuarios y mata estos procesos. Además permite pasar señales a procesos.

$pkill [opciones] expresión regular

Ejemplo: ***$pkill -HUP syslog*** (Este comando le pasa la señal de finalización a *syslog)*

[Opciones]:

- Aquellas que se le pasen para identificar a un proceso, serán las que sirvan para identificar y matarlo.
- Enviar una determinada señal al proceso: **–**SEÑAL (sin SIG

COMANDOS DE ADMINISTRACIÓN

PROCESOS

- **Nohub**: cuando queremos ejecutar un comando que siga en marcha cuando salimos de la Shell o enviamos una señal 1 SIGHUP (exit, Ctrl[z])).

  $nohup [comando/programa] [opciones comando/programa]

- **Nice**: El comando **nice** nos permite ejecutar programas con una determinada prioridad en la CPU.

$nice [prioridad] [programa] [argumentos del programa]

```
$nice -12 [programa] [argumentos...]
```

Nota: Estos 3 comando ejecutan “programa” con prioridad 12 positiva. Para prioridades negativas solo podemos usar las 2 últimas. Si no le pasamos prioridad a *nice*, inicia el programa con prioridad 10.
[Prioridad]:

- La prioridad va desde -20 a 19, siendo los números negativos los de prioridad mas elevada y solo pueden se utilizados por root.

COMANDOS DE ADMINISTRACIÓN

PROCESOS

renice Permite modificar la prioridad de un programa en ejecución. La sintaxis es igual a la de *nice*, pero además nos permite modificar la prioridad de programa(s) por PIDs, por GUIDs y por nombres de usuarios.

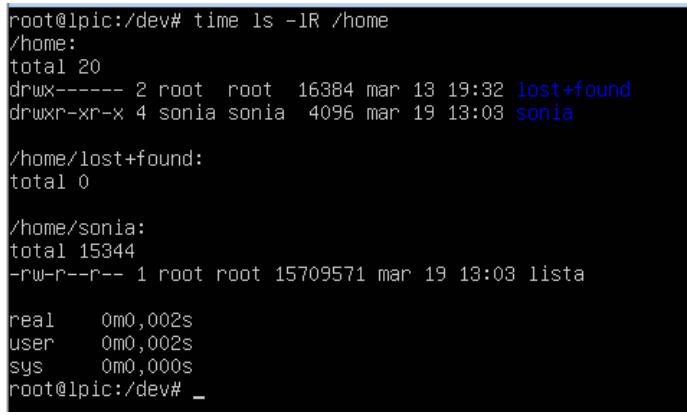
$renice [prioridad] [-p PIDs | -g grps | -u usuarios] [programa] [argumentos…]

Nota: Solo *root* puede modificar la prioridad por GUIs y nombre de usuarios. Si no se le pasa prioridad a *renice*, entiende que el primer número es un *PID***.**

- la opción –p precisa un PID, un –g un GID y –u un UID

Time: Mide las duraciones de ejecución de un comando, lo que es idean para conocer los tiempos de ejecución y devuelve tres valores:

- **Real**: duración total de ejecución del comando.
- **User**: duración del tiempo de CPU necesario para ejecutar el programa.
- **System**: duración del tiempo de CPU necesario para ejecutar los comandos relacionados con el OS



COMANDOS DE ADMINISTRACIÓN

PROCESOS

- **Exec:** Reemplaza el proceso actual por otro. Su Shell por defecto es bash. Si deseamos cambiarlo por el Shell ksh, basta con ejecutar exec ksh. He aquí lo que sucede.

```
$ ps | grep bash
 2375 pts/0    00:00:00 bash
$ exec ksh
$
$ ps | grep ksh
 2375 pts/0    00:00:00 ksh
```

COMANDOS DE ADMINISTRACIÓN

EDITOR VI

- Es el editor de Unix por defecto. En Linux se denomina Vim
- Existen tres modos de funcionamiento:
  - Modo **comando**: Las inserciones representan comandos. Se accede pulsando ESC.
  - Modo **inserción**: Inserción en modo clásico.
  - Modo **línea de comandos**: Una línea en la parte inferior nos permite introducir comandos especiales, para ello tenemos que teclear ESC + :

  Cuando entramos en vi se entra en modo comando por lo que para:
  - Empezar a escribir: ESC + i
  - Para salir: ESC + : y luego q + INTRO

COMANDOS DE ADMINISTRACIÓN

EDITOR VI : COMANDOS DE INTRODUCCION DE DATOS

Hay que introducir previamente el comando [ESC]

Comando	Acción
a	Añadir después del carácter actual
A	Añadir texto al final de línea.
i	Insertar delante del carácter actual, como en un procesador
I	Insertar texto al principio de línea
o	Añadir una línea debajo de la línea actual
O	Insertar una línea encima de la línea actual.

COMANDOS DE ADMINISTRACIÓN

EDITOR VI : COMANDOS DE SALIR Y GUARDAR

Hay que introducir previamente el comando [ESC] :

Comando	Acción
ZZ	Guarda el archivo y sale.
:q!	Salir sin guardar
:q	Salir si no se modificó el archivo
:w	Guardar el archivo. Puede especificar un nombre para seguir.
:wq o :x	Guardar y salir
1, 10w fic	Guarda la líneas de 1 a 10 fic.

COMANDOS DE ADMINISTRACIÓN

EDITOR VI : DESPLAZAMIENTO

Hay que introducir previamente el comando [ESC] :

Comando	Acción	Comando	Acción
h	Ir a la izquierda	w	Ir a la palabra siguiente
l l minúscula	Ir a la derecha	b	Ir a la palabra anterior
k	Ir hacia arriba	f<c>	Saltar al carácter c siguiente
j	Ir hacia abajo	Ctrol+f	Avanzar una pantalla
0	Principio de línea	Ctrol+b	Retrodecer una pantalla
:0	Principio de archivo	G	Ultima línea del archivo
$	Fin de línea	<n>G	Salta a la línea <<n>>
:$	Fin de archivo	:<n>	Idem

COMANDOS DE ADMINISTRACIÓN

EDITOR VI : LA CORRECION

Hay que introducir previamente el comando [ESC] :

Comando	Acción	Comando	Acción
x	Borrar el carácter bajo el cursor	df	Ir a la palabra siguiente
X	Borrar el carácter delante del cursor	b	Ir a la palabra anterior
r<c>	Sustituir el carácter bajo el cursos por el carácter c	f<c>	Saltar al carácter c siguiente
dw	Hasta el final de la palabra	Ctrol+f	Avanzar una pantalla
d$ o D	Borrar hasta el final de la linea	Ctrol+b	Retrodecer una pantalla
dO	Borrar desde el principio de línea hasta el cursor	G	Ultima línea del archivo

COMANDOS DE ADMINISTRACIÓN

EDITOR VI : BÚSQUEDA DE TEXTO

El comando de búsqueda es el carácter /. La búsqueda arranca desde el carácter actual hasta el final de archivo

Comando	Acción
/[CcBb]ola	Ola con C,c,B o b delante
/[A-Z]e	Todo lo que empieza por una may. Min. y una e después
/[^a-z]	Todo lo que no empieza por minúscula
.	El punto sustituye un carácter
*	Un carácter o un conjunto de ellos
/^Auto	^ indica que tiene que estar al principio de línea
/Auto$	El $ indica que la cadena buscada deberá estar al final de línea

COMANDOS DE ADMINISTRACIÓN

EDITOR VI : COMANDOS DE SUSTITUCIÓN

Para sustituir texto, hay que colocarse al principio de la cadena que desea modificar y teclear uno de los comandos siguientes.

Comando	Acción
cw	Sustituye la palabra corriente.
c$	Sustituir hasta el final de la linea
c0	Sustituir hasta el principio de la linea
cf<x>	Sustituir hasta el próximo carácter <x>
c/<rech>	Sustituir hasta la próxima coincidencia de la cadena <rech>

COMANDOS DE ADMINISTRACIÓN

EDITOR VI : COPIAR-PEGAR

El comando **v** permite una selección visual. Se subraya el texto y se puede desplazar el cursos para seleccionar el texto. Luego

Comando	Acción
d	cortar
c	Cortar pero en modo edición
p	Pegar detrás del carácter
P	Pegar delante del carácter
[ESC]+yy	Para copiar una línea
[ESC]+5yy	Para copiar 5 líneas
[ESC]+m1y5w	Para copiar cinco palabras en la memoria m1
[ESC]+m1p	Para pegar el contenido de la memoria m1 en un lugar determinado

COMANDOS DE ADMINISTRACIÓN

EDITOR VI : SUSTITUCIÓN

La sustitución permite reemplazar de manera automática varias coincidencias por otra cadena

: [1ª línea, última línea] s/Modelo/Sustitución/[gc]

Los números de líneas son opcionales. En este caso, la sustitución se hace en la línea corriente únicamente.

Ejemplo: ***:1,$ s/ [Uu]nix/UNIX/gc*** Sustituye todas las coincidencias de Unix o unix en UNIX

COMANDOS DE ADMINISTRACIÓN

EDITOR VI : OTROS

Edición avanzada

Comando	Acción
:r fic	Insertar el contenido de fic desde el lugar actual
:! cmd	Ejecutar el comando. Pulse [Intro] para volver a vi
:r! cmd	Insertar el resultado del comando en el lugar actual
:e fic	Cargar el archivo fic para edición
:e#	Conmutar entre los diferentes archivos abiertos.