1a) Existen vim, emacs, ed

1b) Se diferencian en que con los comandos solo puedo visualizar el texto o concatenarlo, en cambio con los editores puedo hacer muchas más modificaciones. El editor de texto vi tiene tres modos de operación:

- Modo comandos: Vim empieza en modo comando. En este modo se pueden emplear combinaciones de teclas para, por ejemplo, copiar líneas y trabajar en el formato del texto.

- Modo inserción: En modo inserción cuando se pulsan las teclas se edita el texto como en otros editores. Se puede cambiar del modo comandos al modo inserción pulsando la tecla i.

- Modo línea de órdenes: A este modo se accede pulsando la tecla dos puntos :. Tras los dos puntos se pueden introducir órdenes complejas, como por ejemplo buscar y reemplazar con expresiones regulares.

1c)

- dd: Cortar.

- y: Copiar al portapapeles.

- p: Pegar desde el portapapeles.

- u: Deshacer.

- /frase: Busca "frase" dentro del archivo.

Github

- i: Inserta texto antes del carácter sobre el que está el cursor.
- a: Inserta texto después del carácter sobre el que está el cursor.
- I: Inserta texto al comienzo de la línea en la que está el cursor.
- A: Inserta texto al final de la línea en la que está el cursor.
- o: Abre espacio para una nueva línea después de la línea en la que está el cursor y permite insertar texto en la nueva línea.
- O: Igual al anterior, pero abre espacio en la línea anterior.
- ESC: Abandonar el modo linea de orden para volver al modo de comandos; también se usa para cancelar comandos. (Usarlo en caso de duda)
- Ctrl-F: Avanzar una página hacia adelante.
- Ctrl-B: Avanzar una página hacia atrás.
- Ctrl-L: Refrescar la pantalla.
- G: Cursor al final del fichero.
- 1G: Cursor al principio del fichero.
- $: Cursor al final de la línea.

Github

- 0: Cursor al principio de la línea.

- q o q!: Salir del editor.

2a)

1. Se empieza a ejecutar el código del BIOS.
2. El BIOS ejecuta el POST.
3. El BIOS lee el sector de arranque (MBR).
4. Se carga el gestor de arranque (MBC).
5. El bootloader carga el kernel y el initrd.
6. Se monta el initrd como sistema de archivos raíz y se inicializan componentes esenciales.
7. El kernel ejecuta el proceso init y se desmonta el initrd.
8. Se lee el "/etc/inittab".
9. Se ejecutan los scripts apuntados por el runlevel 1.
10. El final del runlevel 1 le indica que vaya al runlevel por defecto.
11. Se ejecutan los scripts apuntados por el runlevel por defecto.
12. El sistema está listo para usarse.

2b) Al proceso init lo ejecuta el kernel, su función es cargar todos los procesos para el correcto funcionamiento del sistema operativo, como por ejemplo montar los filesystem, y hacer disponibles los demás dispositivos.

2d) Los runlevels son el modo en el que arranca Linux siempre (2 en Debian), cada runlevel es responsable de levantar o bajar una serie de servicios.

2e)

- 0: Halt (Parada)
- 1: Single user mode (Modo monousuario)

Github

- 2: Multiuser, without NFS (Modo multiusuario sin soporte de red)
- 3: Full multiuser mode (Modo multiusuario completo, console)
- 4: No se utiliza.
- 5: X11 (Modo multiusuario completo, con login gráfico basado en X)
- 6: Reboot (Reiniciar)

En inittab se define que runlevel ejecutar.
No todas las distribuciones respetan el estándar.

2f) En el archivo "/etc/inittab" se guarda la configuración del proceso init con respecto a que runlevels ejecutar, y en qué acciones consisten cada uno de ellos (donde buscar los procesos referentes a cada runlevel). Su formato es:

- *id:runlevels:acción:proceso*

- *id*: Identifica la entrada en inittab (1 a 4 caracteres).
- *runlevels*: El/Los niveles de ejecución en los que se realiza la acción.
- *acción*: Describe la acción a realizar:

    - wait: Inicia cuando entra al runlevel e init espera a que termine.
    - initdefault:
    - ctrlaltdel: se ejecutará cuando init reciba la señal SIGINT off, respawn, once, sysinit, boot, bootwait, powerwait, etc...

- *proceso*: el proceso exacto que será ejecutado.

2h) Los Scripts RC se guardan en los directorios "etc/rcX.d/", donde X es un runlevel. Son enlaces a las funciones que crean los servicios que están en "etc/init.d/". Los nombres en estos directorios empiezan por una letra (S o K) seguidos de un número y el nombre del servicio. La letra S significa iniciar (S de start). La letra K significa acabar (K de kill). El número es de dos dígitos, de 00 a 99 e indica el orden en el que se arrancará el servicio. Por ej: "S20lpd".

2i) “inserv” se utiliza para manejar y actualizar el orden de los enlaces simbólicos de los RC de una forma más sencilla, en vez de tener que editar muchos archivos en los distintos runlevel para que sean enlaces simbólicos a mi servicio, solo tengo que modificar la cabecera de mi archivo, explicando en que runlevels quiero que se ejecute, y otras relaciones con los Scripts RC.

Es más rápido que un arranque tradicional. La efectividad de esta técnica depende del número de servicios a iniciar, así como de la red de dependencia que haya entre ellos. Cuantos más servicios independientes haya, más se acelerará el arranque (más servicios podrán ser arrancados en paralelo).

2j) “Upstart” es un sistema de arranque compatible SystemV pero que ejecuta las tareas de forma asincrónica permitiendo así una mejora en las prestaciones. Las tareas y servicios son ejecutados ante eventos (arranque del equipo o inserción de un dispositivo USB) definidos como Tareas o Jobs.

2n) El systemd es un sistema que centraliza la administración de demonios y librerías del sistema. Mejora el paralelismo de booteo.

Github

2ñ) La activación por socket es un mecanismo de iniciación bajo demanda. Podemos ofrecer una variedad de servicios sin que realmente estén iniciados. Cuando el socket recibe una conexión spawnea el servicio y le pasa el socket. No hay necesidad de definir dependencias entre servicios, ya que se inician todos los sockets en primer medida.

2o) “cgoups” permite organizar un grupo de procesos en forma jerárquica. Agrupa conjuntos de procesos relacionados.

3a)

- etc/passwd: Se usa para guardar el UID, el GID, su directorio de usuario, su nombre, etc. Tiene el siguiente formato:

- *nombre_de_cuenta : contraseña : número_de_usuario : número_de_grupo : comentario : directorio : programa_de_inicio*

- etc/group: Se usa para guardar los datos del grupo, como su GID, sus miembros, etc. Su formato es:

- *nombre_de_grupo : campo_especial : número_de_grupo: miembro1, miembro2*

Con frecuencia, el campo especial está vacío. El número de grupo corresponde al número del vínculo entre los archivos "/etc/group" y los archivos "/etc/passwd".

- etc/shadow: Se usa para guardar la contraseña de forma encriptada, junto con la fecha de expiración.

Github

3b) El UID es el identificador único de usuario y el GID es el identificador único de grupo. En versiones anteriores, podía haber varios usuarios con el mismo UID, pero luego esto fue corregido, ya que de no hacerlo se podía modificar el UID de un usuario a 0 y así este tendría los mismos permisos que el root (ya que este último tiene UID= 0).

3c) El usuario root es el usuario que tiene permiso para manejar y modificar todas las cosas importantes. Puede crear usuarios, modificarlos, modificar los permisos de estos, modificar cualquier archivo, etc. Su UID es igual a 0. No puede existir más de un root en GNU/Linux.

3e)

- adduser: Agrega el usuario. Modifica el archivo "etc/passwd" (va a pedir todos los datos).

- usermod <nombre usuario> …
  - … -g: Modifica el grupo inicial. Modifica "etc/passwd".
  - … -G: Modifica grupos adicionales "etc/group".
  - … -d: Modifica el directorio home del usuario. Modifica "etc/password".

- userdel <nombre usuario>: Elimina el usuario.

- su <nombre usuario>: Cambia al usuario, antes pidiendo su contraseña. Si no se pone <nombre usuario> se establece que se quiere cambiar a ser super usuario.

- groupadd <nombre grupo>: Crea un nuevo grupo.

- who: Muestra quién está logueado.

Github

- groupdel <nombre grupo>: Elimina el grupo.

- passwd <nombre usuario>: Asigna o cambia la contraseña. Modifica los archivos "etc/shadow".

4a) Los permisos están divididos en tres tipos: lectura, escritura y ejecución. Estos permisos pueden ser fijados para tres clases de usuarios: el propietario del archivo o directorio, los integrantes del grupo al que pertenece y todos los demás usuarios.

- El permiso de lectura permite a un usuario leer el contenido del archivo o en el caso de un directorio, listar el contenido del mismo (usando “ls”).

- El permiso de escritura permite a un usuario escribir y modificar el archivo (inclusive, eliminarlo). Para directorios, el permiso de escritura permite crear nuevos archivos o borrar archivos ya existentes en el mismo.

- Por último, el permiso de ejecución permite a un usuario ejecutar el archivo si es un programa. Para directorios, el permiso de ejecución permite al usuario ingresar al mismo (por ejemplo, con el comando cd).

4b)

- chmod: Permite cambiar los permisos de un archivo.

- chown: Permite cambiar qué usuario es el dueño del archivo. (owner)

Github

- chgrp: Permite cambiar qué grupo es el dueño del archivo. (group)

4c) El valor octal que se usa para definir permisos con chmod permite asignar permisos para usuarios, grupos, y otros a la vez. Por ej, si los permisos actuales son de la forma:

- -rw-rw-r--

Con "chmod 764", nos quedarán de la forma:

- -rwxrw-r--

Porque cada cifra del octal representa un valor binario que se le asigna a cada una de las tres partes de los permisos.

4e) Path absoluto: Empieza desde el directorio raíz "/" (sin comillas). Ej: /home/user/programas/script.sh

Path relativo: Empieza desde donde estamos posicionados. Mismo ej, si estoy en programas: ./script.sh

4f) Con el comando "pwd" se puede determinar la ruta del directorio actual.

Con $HOME se puede acceder al directorio personal del usuario, y de ahí concatenamos las carpetas o archivos que queremos acceder.

4g)

- cd: Acceder a un directorio. Con ".." podemos ir un directorio hacia atrás.

Github

- mount: Cuando conectamos una unidad de almacenamiento o un lector de DVD, el sistema tiene que crear lo que se denomina un punto de montaje para poderlo utilizar. Este punto de montaje en casos de discos duros y pendrives, suele ser una carpeta que creamos manualmente en el sistema con este comando. Ej:

    - mount -t (tipo de dispositivo, fat, ext3, ext4, etc) Directorio de destino (debe existir)

- umount: Para desmontar un disco del sistema. Ej:

    - umount Directorio de destino (debe existir)

- mkdir: Crear un directorio.

- rmdir: Eliminar un directorio.

- du: Nos permite ver el tamaño de un archivo o carpeta.

- df: Nos permite ver el uso de espacio en disco.

- ln: Nos permite crear un enlace simbolico (acceso directo) a archivos o directorios. Formato: ln -s ORIGEN DESTINO. Ej:

    - ln -s /var/www/index.html /home/asolano/index.html

- ls: Lista todo el contenido de un directorio.

- pwd: Informa el nombre del directorio actual.

- cp: Copiar archivos o directorios. Formato: cp ORIGEN DESTINO

- mv: Mover o renombrar un archivo o directorio. Formato: mv ORIGEN DESTINO.

5a) Un proceso es básicamente un programa en ejecución. PID (Process ID) es un identificador numérico único para cada proceso. PPID (Process Parent ID) es el identificador del proceso padre.

5b) Con el comando ps podemos ver los procesos en ejecución en el momento en que se ingresa el comando.

5c) Un proceso que se ejecuta en Foreground es un proceso que se conecta con la terminal, y puede comunicarse con el usuario a través de la pantalla y el teclado. Al contrario, un proceso que se ejecuta en Background es un proceso que no se conecta con la terminal y no puede comunicarse con el usuario de ninguna manera.

5d) Un proceso puede ejecutarse directamente en Background agregando un “&” al final del comando que ejecuta al proceso. Ej:

- sleep 60 &

Un proceso en ejecucion se puede enviar al Background, primero deteniendo su ejecución con Ctrl+Z y luego con el comando bg.

Github

Para traer un proceso al Foreground se puede usar el comando fg, el cual traerá al último proceso en la fila de procesos. Para traer un proceso específico, hay que agregar al comando fg el "job id".

El "job id" es el número que se ve al principio de cada línea del output que provee el comando "jobs".

5e) El Pipe “|” es una herramienta que permite procesar secuencialmente una serie de comandos referentes a un conjunto de datos, conectando la salida de un comando con la entrada de otro. Ej:

- cat /etc/passwd | cut -d: -f1 | grep a | wc -l

Del archivo etc/passwd, cortar con delimitador ":" y quedarse con la fila uno, buscar cadenas que contengan "a", y finalmente contar las líneas.

5f)

- Re-dirección destructiva (>): Si el archivo no existe, lo crea. Si existe, lo sobreescribe. Ej:
    - ls > /tmp/lista.txt
- Re-dirección NO destructiva (>>): Si el archivo no existe, lo crea. Si existe, agrega el resultado de comando al final del mismo. Ej:
    - ls >> /tmp/lista.txt

Github

5g) El comando kill permite terminar procesos manualmente. Envía una señal a un proceso que provoca que el proceso se termine. Formato: kill [signal] PID. Ej:

- kill -9 1212

Signal o Señal es un parámetro del comando kill que especifica el modo en que se realiza la terminación. Con "kill -l" podemos ver una lista de Señales disponibles.

5h)

- ps: Ver los procesos en ejecucion en el momento en que se ingresa el comando.

- pstree: Ver los procesos en ejecucion en el momento en que se ingresa el comando, en formato de árbol.

- kill: Terminar procesos manualmente.

- killall: Terminar procesos con un nombre específico manualmente.

6a) Empaquetar archivos es unir varios archivos en uno solo. Para esto se usa el comando "tar":

- Parámetros -cvf para empaquetar:
  - tar -cvf archivo.tar archivo1 archivo2 archivo3
- Parámetros -xvf para desempaquetar:
  - tar -xvf archivo.tar

Github

6c)

- Genera archivo.tar.gz comprimido:
    - gzip archivo.tar

- Descomprime archivo.tar:
    - gzip -d archivo.tar.gz

6d) Si deseamos comprimir más de un archivo, podemos usar el comando "cat" y gzip utilizando pipes. Ej:

- cat fichero1.txt fichero2.txt | gzip > final.gz

6e)

- tar: Empaquetar varios archivos en uno solo.

- gzip: Comprimir archivos.

- grep: Buscar líneas que contengan frases, números, cadenas, o datos específicos en un archivo.

- zgrep: Buscar igual que "grep", incluso si el archivo en el que se busca está comprimido.

- wc: Buscar número de líneas, caracteres, palabras en un archivo. Con el parámetro -l obtenemos el número total de líneas. Si ejecutamos el comando sin opciones, podremos ver el número de líneas, número de palabras y número de bytes. Ej:

    - wc /ruta/archivo.txt
    - 60 237 2059 archivo.txt

Github