UD 05.- Administración Básica de Linux

1. Administración de usuarios y grupos locales.

GNU/Linux es un sistema multiusuario donde varios usuarios pueden iniciar sesión simultáneamente desde otros terminales o estaciones de trabajo a un ordenador siempre que estén en un entorno de trabajo en red. También, los usuarios creados en el sistema, pueden acceder localmente en el ordenador, abriendo diferentes sesiones.

La administración y gestión de los usuarios que acceden al sistema, es una de las tareas que controla el usuario administrador. El sistema operativo tiene que aportar funciones que permitan la seguridad del acceso mediante usuarios al sistema. Las tareas que realiza el administrador de usuarios:

Añadir, modificar y eliminar usuarios en el sistema.

Añadir, modificar y eliminar grupos locales o globales.

Fijar el plan de cuentas y contraseñas en el equipo junto con una política de derechos de usuario.

Establecer el sistema de auditorías.

Generalmente la entrada al sistema o login (también se denomina loguearse) se realiza con la identificación del nombre de usuario y su contraseña de acceso (existen otros mecanismos como tarjeta inteligente identificativa, reconocimiento de huellas, voz, etc). Cada usuario, dentro del sistema, pertenece a un tipo de conjunto de usuarios denominado grupo de usuario y podrá pertenecer a tantos como sea necesario, adquiriendo los permisos de todos ellos.

El comando para salir del sistema es **logout**, para terminales en modo texto, y **exit**, para salir del programa emulador de terminal en modo gráfico.

GNU/Linux utiliza un sistema de cuentas de usuarios para poder utilizar el sistema operativo de forma eficiente y seguro

Usuarios

Los usuarios en Linux se identifican por un número único de usuario (comprendido entre 0 y 65534) denominado UID (User Identifier), y deben pertenecer a un grupo principal de usuario, identificado también por un número único de grupo denominado GID (Group Identifier).

En los sistemas Linux podemos identificar tres tipos de usuarios:

Usuario root: tiene como UID el valor cero y se encarga de todas las tareas de administración del sistema (usuarios, aplicaciones, controladores,etc.) siendo la única cuenta con privilegios sobre todo el sistema. A este usuario también se le conoce como administrador o superusuario. La cuenta root, por seguridad, está deshabilitada en la mayoría de distribuciones Linux (aunque ya hemos visto como habilitarla).

Usuarios del sistema, como por ejemplo: daemon (procesos y servicios del sistema), nobody (usuario sin privilegios), backup, bin, lp, mail, etc: tienen como UID un valor inferior al 1000 (en Ubuntu), no tienen contraseñas y dependiendo de la cuenta tienen distintos privilegios de root. Se crean normalmente de forma automática en el momento de la instalación del sistema o de alguna aplicación.

Usuarios normales: se usan para usuarios individuales. Tienen un UID a partir del 1000 (en Ubuntu), cada usuario tiene una carpeta personal, ubicada en “/home” y con el mismo nombre de usuario, teniendo solo privilegios completos en su carpeta personal. Cada usuario puede personalizar su entorno de trabajo.

En Ubuntu, el usuario root está desactivado por defecto, siendo el usuario creado en la instalación el que puede realizar todas las tareas del administrador usando para ello el comando “sudo”.

Grupos

Para poder administrar los permisos de los usuarios de una forma más cómoda, Linux, al igual que Windows, permite agrupar los usuarios en grupos, los cuáles se identifican por un identificador numérico que es utilizado por el sistema.

También debemos tener en cuenta, que en Linux todos los usuarios tienen que pertenecer a la fuerza a un grupo, llamado "grupo principal", que será utilizado a la hora de asignar los permisos a los ficheros.

Además, cada usuario opcionalmente puede pertenecer a varios grupos, siendo estos conocidos como grupos secundarios.

El sistema dispone de una serie de grupos predeterminados, que dependen de los recursos y servicios activados en el servidor. Cuando se da de alta un usuario se crea un grupo con el mismo nombre que el usuario y con los mismos derechos y privilegios que se conceden al usuario. De la misma forma, cada vez que se instala un servicio en el ordenador se crea un grupo perteneciente al servicio, se considera que cuando se incluye un usuario a dicho grupo ya dispone de derechos para usar el servicio. Por ejemplo, si se habilita el servicio ssh todos los usuarios que estén incluidos tienen derecho a acceder al servidor remotamente (desde otro terminal).

La información de los grupos se almacena en el fichero "/etc/group".

Nota: un grupo no puede contener a otro grupo.

1.1.- Ficheros Linux de la gestión de usuarios y grupos locales.

Un usuario es identificado por un número de usuario, el uid (user ID) y por un número de grupo el gid (group ID), que le permite al sistema asociar los procesos mediante esos números identificativos.

Los ficheros relacionados con las cuentas de usuario y grupos son los siguientes:

FICHERO	LOCALIZACION	DESCRIPCIÓN
passwd	/etc/passwd	Se encuentran definidas las cuentas de usuario
shadow	/etc/shadow	Contiene las contraseñas de usuario encriptadas
group	/etc/group	Contiene una relación de los grupos a los que pertenecen los usuarios
gshadow	/etc/gshadow	Contiene las contraseñas de grupo encriptadas
login.defs	/etc/login.defs	Están definidas las variables que controlan los aspectos de la creación de usuarios y de los campos de shadow usadas por defecto

La información de **las cuentas de usuario en Linux se almacena en dos archivos**:

- ***/etc/passwd***: contiene las cuentas de usuario existentes. En cada línea representa un usuario con la siguiente información, separada por dos puntos:

<nombre usuario>:<contraseña encriptada>:<uid>:<gid>:<descripción de la cuenta>:<el directorio local [home]>:<shell>

- **/etc/shadow**: contiene las contraseñas encriptadas de los usuarios e información relacionada con la gestión de las contraseñas.. Cada línea representa un usuario con la siguiente información separada por dos puntos:

<nombre usuario><contraseña encriptada><1><2><3><4><5><6><7>

Donde:

1-Días transcurridos desde 1-1-1970 donde el password fue cambiado por última vez.

2-El mínimo número de días entre cambios de contraseña.

3-Días máximos de validez de la cuenta.

4-Días que avisa antes de caducar la contraseña.

5- Días después de que un password caduque para deshabiltar la cuenta

6- Fecha de caducidad. días desde 1-1-1970, donde la cuenta es deshabilatada y el usuario no podrá iniciar sesión

7- Campo reservado para usos futuros.

Podemos ver los campos traducidos mediante el comando chage: **chage -l <usuario>**

La información de los grupos de usuario en Linux se almacena en los archivos:

- **/etc/group**: cada línea representa a un grupo con la siguiente información:

<nombre grupo>: <contraseña encriptada o "x" si no tiene contraseña>: <gid o identif. del grupo>:<lista de los miembros del grupo>

- **/etc/gshadow:** Al igual que el fichero /etc/shadow de las contraseñas encriptadas para usuarios, también se puede usar un fichero /etc/gshadow de contraseñas encriptadas para grupos. Se suele usar para permitir el acceso al grupo, a un usuario que no es miembro del grupo. Ese usuario tendría entonces los mismos privilegios que los miembros de su nuevo grupo.

Además, en el fichero de configuración "**/etc/login.defs**" están definidas diferentes variables que controlan aspectos de la creación de usuarios y contraseñas (como la longitud de las contraseñas, número de días que una contraseña es válida, etc).

Otros directorios y ficheros

El directorio **/etc/skel** proporciona una forma de estar seguro de que todos los nuevos usuarios de tu sistema tienen la misma configuración inicial. El administrador del sistema puede crear archivos dentro de /etc/skel que proveerán un amable entorno predeterminado para los usuarios. Por ejemplo, puede crear un **/etc/skel/.profile** que configura las variables de entorno de algún editor más amigable para los usuarios nuevos.

El fichero **/etc/shells** contiene una lista de shell válidos. Si la shell de usuario no está en este fichero, o bien el usuario tiene la shell /bin/false, no puede acceder al sistema mediante login.

1.2.- Administración de usuarios y grupos en modo comando

Para gestionar usuarios y grupos en una terminal o consola disponemos de los siguientes comandos:

adduser <nombreusuario> → crea un usuario.

useradd <nombreusuario> → crea un usuario. No se recomienda su uso en Debian.

deluser <nombreusuario> → elimina un usuario.

userdel <nombreusuario> → elimina un usuario. No se recomienda su uso en Debian.

usermod <nombreusuario> → modifica el usuario.

passwd <nombreusuario> → crea o cambia la contraseña a un usuario.

chfn <nombreusuario> → cambia la información de un usuario.

chsh <nombreusuario> → cambia la shell a utilizar por un usuario.

chage <opciones ><nombreusuario> → cambia la información de caducidad de contraseñas.

Comandos para gestionar grupos:

groups <nombrusuario> → ver los grupos a los que pertenece el usuario pasado como argumento.

addgroup <nombregrupo> → crea un grupo.

delgroup <nombregrupo> → elimina un grupo. No elimina un grupo primario de un usuario existente.

groupdel <nombregrupo> → elimina un grupo.

groupmod –n <nombrenuevo> <nombreactual> → cambia el nombre a un grupo.

gpasswd –a <usuario> <grupo> → añade un usuario a un grupo.

gpasswd –d <usuario> <grupo> → elimina un usuario de un grupo.

newgrp <grupo> → cambia el grupo activo al usuario actual.

1.3.- Perfiles de usuario en GNU/Linux

Para cualquier usuario dado de alta en el sistema se creará una carpeta dentro del directorio **/home** con el nombre del usuario que contendrá el perfil que se le aplicará cuando inicie sesión en Linux. El usuario será el único que tendrá todos los derechos de uso.

Por seguridad el usuario **root** tiene su propio perfil local de usuario ubicado en el directorio **/root**. Se pueden crear scripts y ficheros de inicio que se ejecutarán al entrar en sesión un usuario de manera que podamos configurar el perfil de trabajo del usuario dentro del sistema.

El fichero **/etc/profile** contiene el perfil igual para todos los usuarios, en su interior podemos poner comandos que se ejecutarán al iniciar sesión cualquier usuario, también ejecuta todos los script que se encuentran en el directorio **/etc/profile.d**.

Cada vez que se inicia sesión de un usuario con el comando, se ejecutarán los siguientes ficheros ocultos (llevan el identificativo del punto) relacionados con el perfil de acceso al sistema de un usuario:

1	/etc/profile	Ejecutar el perfil genérico para todos los usuarios
2	/home/nombre_usuario/.profile	Ejecuta el .bashrc
3	/home/nombre_usuario/.bashrc	Contiene comandos que se ejecutan al inicio del Shell de forma interactiva
4	/home/nombre_usuario/.bash_history	Almacena el histórico de comandos que introduce el usuario en consola
5	/home/nombre_usuario/.bash_logout	Se ejecuta cuando el usuario sale de la sesión

Hay que destacar que cuando se inicia sesión desde un terminal para cambiar de usuario solamente se ejecuta el fichero ***.bashrc***

Algunas operaciones con ficheros de perfil:

alumno@sistemaubuntu:~$ ls -lrta | grep .profile	Busca los fichero .profile en la ubicación actual
alumno@sistemaubuntu:~$ ls -a	Lista todos los ficheros ocultos
alumno@sistemaubuntu:~$ more .profile	Visualizamos el contenido del fichero .profile
alumno@sistemaubuntu:~$ gedit .profile	Editamos el fichero .profile
alumno@sistemaubuntu:~$ su – nombre_usuario	Cambiamos de usuario y se ejecuta su .profile
alumno@sistemaubuntu:~$ su nombre_usuario	Cambiamos de usuario pero no ejecuta el .profile

Debemos de tener en cuenta que cada vez que modifiquemos el fichero ***.profile*** y ejecutemos el comando ***su – Nombre_usuario*** se ejecutara lo que éste contenga ya que es un script de inicio de sesión del usuario. Un ejemplo para hacer que se muestre un mensaje de bienvenida cada vez que inicia sesión un usuario en el sistema puede ser el siguiente:

1	alumno@sistemaubuntu:~$ gedit .profile	Editar el fichero .profile
2	echo “Hola ya estas en el sistema” echo “estás en el directorio” pwd	Añadir al final las líneas siguientes
3		Guardar el archivo y salir
4	exit	Salir de la sesión del usuario
5	alumno@sistemaubuntu:~$ su – carlos	Entrar al sistema como usuario carlos

2. Gestión de permisos en GNU/Linux.

En Linux, el sistema asigna permisos a los distintos objetos del sistema de ficheros en base a tres niveles:

- Nivel del usuario propietario del archivo.
- Nivel del grupo propietario del archivo.
- Nivel del resto de usuarios.

Para cada uno de estos tres niveles, se pueden asignar los siguientes permisos básicos:

- Lectura (r):

  En archivos significa que se puede leer.

  En carpetas significa que podemos ver el contenido (comando ls).

- Escritura (w):

  En archivos significa que se puede modificar el contenido.

  En carpetas significa que podemos añadir o borrar ficheros dentro de ese directorio.

- Ejecución (x):

  En archivos significa que se puede ejecutar el contenido.

  En carpetas significa que podemos entrar en la carpeta (comando cd).

El comando ls –l nos muestra los permisos de un fichero atendiendo a estos 3 niveles y con los permisos antes descritos. A modo de ejemplo:

El primer guión por la izquierda (-) nos indica el tipo de fichero que es, pudiendo ser uno de los siguientes:

- fichero normal.
- d directorio o carpeta.
- l enlace simbólico.
- c dispositivo de caracteres.
- b dispositivo de bloques.
- p tubería.
- s socket.

Los 3 primeros tipos de ficheros son los mas habituales.

2.1. Cambiando permisos.

Para cambiar los permisos a un fichero debemos usar el comando “chmod”. Este comando podemos usarlo de dos maneras.

Método 1: usando notación octal

La sintaxis básica es:

chmod <permisos en octal> <fichero>

<permisos en octal> es un valor de 3 o 4 dígitos en octal que representa lo siguiente:

Octal	Binario	Permisos
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Método 2: usando notación simbólica

La sintaxis básica es:

chmod <usuario><operador><permiso> <fichero>

Donde <usuario> puede ser:

u → propietario (user)
g → grupo (group)
o → el resto (other)
a → todos (all)

Donde <permisos> puede ser:

r → lectura
w → escritura
x → ejecución
s → set uid o gid
t → sticky bit

Donde <operador> puede ser:

+ → añade el permiso.

- → quita el permiso.

= → para asignar exactamente un permiso concreto (eliminando cualquier otro que ya pudiese tener el fichero.

El mandato **chmod** jamás cambia los permisos de enlaces simbólicos, sin embargo no representa un problema dado que jamás se utilizan los permisos de los enlaces simbólicos. Si se aplica el mandato **chmod** sobre un enlace simbólico, se cambiará el permiso del fichero o directorio hacia el cual apunta. Cuando se aplica **chmod** de forma descendente en un directorio, éste ignora los enlaces simbólicos que pudiera encontrar en el recorrido.

2.2. Permisos de archivos por defecto.

Cuando creamos un fichero, este se crea con unos permisos por defecto (llamado máscara) que pueden depender del usuario que lo crea.

El comando **umask**, es la abreviatura de user file-creation mode mask, y sirve para establecer los permisos por defecto que tendrán los nuevos ficheros y directorios que creemos.

Para saber la máscara que tiene un usuario al crear ficheros usamos el comando umask de las siguientes formas:

$umask **$umask -S**

La máscara que tiene un usuario se define en el fichero "**/etc/profile**".

Los permisos iniciales por defecto para un archivo son 666 y para una carpeta 777. Estos permisos son modificados por la máscara que tiene el usuario de la siguiente manera:

<Permiso inicial> – <máscara del usuario> = <permiso del fichero>

Ejemplo: si un programa crea ficheros con permisos 666 (rw-rw-rw-) y la umask es de 002 el fichero finalmente tendría de permisos 666-002=664, es decir, rw-rw-r--. Si embargo, si para otro usuario la umask es de 022, el mismo programa crearía los ficheros con permisos 666-022=644, es decir, rw-r--r--.

Para cambiar la máscara que se aplica a los ficheros podemos:

Usar el comando umask:

$**umask <máscara en octal>**. Este cambio se perderá al finalizar la sesión activa.

Modificar el fichero "**/etc/profile**" e indicar el valor a umask al final del fichero. Este cambio es permanente y para todos los usuarios.

Agregar el umask en el fichero .profile o en el .bashrc de nuestro home y afectar únicamente a nuestro usuario.

Actualmente la mayoría de las distribuciones tienen un umask por defecto de 022 o 002. Si el nombre del usuario y del grupo es el mismo, y el uid es mayor que 100 se establece un umask de 002, y sino de 022.

2.3. Otros comandos relacionados con permisos.

Para cambiar el propietario o el grupo de un fichero debemos ser el propietario o el usuario root. Los comandos que nos permiten hacer esto son:

Comandos para cambiar el propietario y el grupo de un fichero:
chown <propietarionuevo> <nombrefichero> -> cambia el propietario de un fichero.
chown –R <propietarionuevo> <nombredirectorio> -> cambia el propietario de un directorio
chgrp <gruponuevo> <nombrefichero> -> cambia el grupo de un fichero.
chgrp –R <gruponuevo> <nombredirectorio> -> cambia el grupo de un directorio.

Comando chown.

El comando chown se utiliza para cambiar el propietario al que pertenece un fichero o directorio. Puede especificarse tanto el nombre de un usuario, así como un número de identidad de usuario (UID). Opcionalmente, utilizando un signo de dos puntos (:), o bien un punto (.), permite especificar también un nombre de grupo.

Utilización:

chown [opciones] usuario[:grupo] fichero(s) o directorio(s)

Opciones:

-R	Cambia recursivamente el propietario (y, opcionalmente, el grupo al que pertenecen los directorios, junto con todos sus contenidos.
-v (o --verbose)	Salida de chown más descriptiva.
--version	Ver el número de versión del programa.
--dereference	Actúa sobre enlaces simbólicos en lugar de hacerlo sobre el destino.
-h (o --no-dereference)	En el caso de enlaces simbólicos, cambia el propietario del destino en lugar del propio enlace.
--reference	Cambia el el propietario de un fichero, tomando como referencia el propietario de otro.

Comando chgrp.

El comando chgrp se utiliza para cambiar el grupo al que pertenece un fichero o directorio. Puede especificarse tanto el nombre de un grupo, así como un número de identidad de grupo (GID).

Utilización:

chgrp [opciones] fichero(s) o directorio(s)

Opciones:

-R	Cambia recursivamente el grupo al que pertenecen los directorios, junto con todos sus contenidos.
-v (o --verbose)	Salida de chgrp más descriptiva.
--version	Ver el número de versión del programa.
--dereference	Actúa sobre enlaces simbólicos en lugar de hacerlo sobre el destino.
-h (o --no-dereference)	En el caso de enlaces simbólicos, cambia el grupo del destino en lugar del propio enlace.
--reference	Cambia el grupo de un fichero, tomando como referencia el propietario de otro.

3. Gestión de procesos en GNU/Linux.

Linux no hace una distinción tan clara como Windows entre procesos y demonios. De hecho, la gestión de procesos y la de demonios sigue las mismas directrices.

Los sistemas Linux siempre se han destacado por la permisividad de manipulación de los objetos del sistema, por lo que disponemos de un gran abanico de herramientas (sobre todo a nivel de consola) para gestionar los procesos.

Antes de abordar la gestión de los procesos en sistemas Linux conviene tratar un concepto bastante interesante: El plano de un proceso. Cuando el usuario está interactuando con un proceso, se dice que este proceso está en primer plano. Cuando el proceso se ejecuta "en silencio", sin necesidad de que el usuario interactúe con él se dice que el proceso está en segundo plano. Un ejemplo típico de procesos en segundo plano son los demonios.

Sólo un proceso puede estar en primer plano en un momento dado. Por el contrario, en segundo plano pueden existir varios procesos a la vez.

Por otro lado, un proceso puede pasarse de primer plano a segundo plano, y viceversa, cuantas veces se desee.

3.1.- Gestión de procesos por interfaz gráfico.

Linux proporciona una herramienta similar al Administrador del sistema de Windows llamada Monitor del sistema. De esta herramienta vemos la pestaña Procesos.

Esta pestaña muestra, por defecto, todos los procesos (incluidos los demonios) que están asociados al usuario. A través de la opción Ver del Monitor del sistema podemos seleccionar otro criterio de visualización, pudiendo mostrar todos los procesos del sistema o solamente aquellos que están en estado activo.

Para cada proceso, como sucedía en Windows, se pueden aplicar una serie de operaciones (con clic derecho sobre el proceso en cuestión), de entre las que destacamos estas:

- Detener el proceso: pasa el proceso a estado suspendido.
- Continuar el proceso: reanuda un proceso que estaba detenido.
- Finalizar el proceso: pasa el proceso a estado terminado.
- Matar el proceso: pasa el proceso a estado terminado de forma inmediata.
- Cambiar la prioridad: permite variar la prioridad del proceso siendo 0 la prioridad normal y pudiendo oscilar entre los valores -20 (máxima prioridad) y 20 (mínima prioridad).

En la opción Ver también podemos activar la funcionalidad Dependencias. Al hacerlo, en el listado también se muestran los procesos hijo. Aquellos procesos que tienen procesos hijo asociados aparecen con un triángulo a su izquierda sobre el que se puede pinchar para mostrarlos/ocultarlos. Sobre los procesos hijo se pueden realizar las mismas operaciones que sobre los procesos padre.

3.2.- Gestión de procesos por línea de comandos.

Las principales acciones que pueden realizarse sobre procesos desde la línea de comandos son estas:

Mostrar información sobre los procesos.

El comando ps (abreviatura de Process Status) muestra un listado con el estado de los procesos.

El comando pstree es una variante de ps en la que los procesos se muestran en forma de árbol.

El comando jobs se utiliza para mostrar los procesos que se están ejecutando en primer y en segundo plano.

Cambiar el estado de los procesos.

El comando kill permite enviar señales a los procesos para cambiar su estado. La señal por defecto de este comando es terminar (matar) el proceso, de ahí su nombre.

Cambiar la prioridad de los procesos.

El comando nice se emplea para cambiar la prioridad de un proceso cuando se inicia su ejecución. Por defecto el valor de la prioridad es 0. Sólo los usuarios root tienen privilegios para asignar prioridades negativas.

El comando renice permite cambiar la prioridad de un proceso que ya está en ejecución.

Cambios de plano de los procesos.

El operador & se utiliza para lanzar un proceso directamente a segundo plano.

El comando nohup se usa para lanzar un proceso haciéndolo inmune a los hangup (cuelgues).

Cuando un proceso está detenido, se puede utilizar los comandos fg y bg para lanzarlo en primer plano o en segundo plano, respectivamente.

Mostrar información: Comando ps

El comando ps muestra por pantalla un listado de los procesos que se están ejecutando en el sistema.

Si no añadimos ningún parámetro, ps mostrará los procesos del usuario con el que estamos logueados. Por otra parte, los parámetros más básicos son los siguientes:

-aux Lista los procesos de todos los usuarios con información añadida (destacamos más abajo).

-a Lista los procesos de todos los usuarios.

-u Lista información del proceso como por ejemplo el usuario que lo está corriendo, la utilización de Cpu y memoria, etc.

-x Lista procesos de todas las terminales y usuarios

-l Muestra información que incluye el UID y el valor “nice“.

–forest – Muestra el listado procesos en un formato tipo árbol que permite ver como los procesos interactuan entre si, podría ser algo similar al comando pstree.

-e visualiza información sobre "todos" los procesos del sistema.

-A idem a la opción -e

-j visualiza información sobre el PGID y SID.

-f visualiza los parámetros con los que se levanto el proceso.

-N niega el efecto de cualquier opción que se haya especificado.

-u pepe visualiza información de los procesos del usuario pepe.

Ejemplo de uso del comando ps:

- Ver todos los procesos del sistema (en formato completo)

  #ps –ef

- Ver todos los procesos en nuestro shell activo

  #ps

Mostrar información. Comando pstree

Visualizar el árbol de procesos, mostrando la relación padre hijo

pstree -a muestra los procesos junto con sus parámetros.

Mostrar información. Comandos jobs

El comando jobs se utiliza para listar procesos que estés ejecutando en segundo plano o en primer plano. Si la respuesta se devuelve sin información es que no hay procesos presentes.

La sintaxis es

jobs [opciones]

OPCIONES:

-l Informa del identificador del grupo de proceso y la carpeta de trabajo de las operaciones.

-n Muestra sólo los trabajos que se han detenido o cerrado desde la última notificación.

-p Muestra sólo el identificador de proceso para los líderes de grupo de procesos de los trabajos seleccionados.

EJEMPLO:

jobs –l : Muestra los trabajos que estás ejecutando en primer plano (o) en segundo plano.

jobs –p : Muestra sólo el identificador de proceso para los trabajos listados.

Cambiar el estado de los procesos. Comando kill:

El comando kill permite enviar un mensaje arbitrario a un proceso, o varios, con un PID igual a pids. El parámetro sigspec es el valor entero de la señal o el nombre de la señal que enviaremos al proceso. El valor de sigspec se puede especificar en minúsculas o mayúsculas, pero normalmente se especifica en mayúsculas.

Sintaxis**:** kill [-s sigspec | –sigspec] [pids]

Para especificar el tipo de señal se pueden usar –s sigspec o simplemente

–sigspec.

Si se omite sigspec en el comando kill, se toma por defecto el valor SIGTERM (señal 15, salir de manera correcta).

El valor de pids tendremos que averiguarlos utilizando la orden ps.

Ejemplos: $kill-9 337 $kill-KILL 337

El grupo de señales que podemos enviar a kill son:

Nombre	Entero	Descripción
SIGHUP	1	Colgar. Esta señal es enviada automáticamente cuando un modem se desconecta. También se usa por muchos demonios para forzar la relectura del archivo de configuración. Por ejemplo, en los procesos init y inetd.
SIGINT	2	Parar el proceso y desaparece. Esta señal se envía con la secuencia de teclas Ctrl-C.
SIGKILL	9	Kill. Mata un proceso incondicionalmente e inmediatamente. Enviar esta señal es un método drástico de terminar el proceso, ya que no se puede ignorar
SIGTERM	15	Terminar. Mata un proceso de forma controlada.
SIGCONT	18	Continuar. Cuando un proceso detenido recibe esta señal continúa su ejecución.
SIGSTOP	19	Parar, pero preparado para continuar. Esta señal se envía con la secuencia de teclas Ctrl-Z.

Cambiar prioridad. Comando nice

Permite cambiar la prioridad de un proceso. Por defecto, todos los procesos tienen una prioridad igual ante el CPU que es de 0. Con nice es posible iniciar un programa (proceso) con la prioridad modificada, más alta o más baja según se requiera. Las prioridades van de -20 (la más alta) a 19 la más baja. Solo root o el superusuario puede establecer prioridades negativas que son más altas. Con la opción -l de ps es posible observar la columna NI que muestra este valor.

Ejemplo :

#> nice (sin argumentos, devuelve la prioridad por defecto)

o

#> nice -n -5 comando (inicia comando con una prioridad de -5, lo que le da más tiempo de cpu)

Cambiar prioridad. Comando renice

Así como nice establece la prioridad de un proceso cuando se inicia su ejecución, renice permite alterarla en tiempo real, sin necesidad de detener el proceso.

Ejemplos:

#> nice -n -5 yes (se ejecuta el programa 'yes' con prioridad -5)

(dejar ejecutando 'yes' y en otra terminal se analiza con 'ps')

#> ps -el

F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD

4 S 0 12826 12208 4 75 -5 - 708 write_ pts/2 00:00:00 yes

#> renice 7 12826

12826: prioridad antigua -5, nueva prioridad 7

#> ps -el

F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD

4 S 0 12826 12208 4 87 7 - 708 write_ pts/2 00:00:15 yes

(obsérvese el campo NI en el primer caso en -5, y en el segundo con renice quedó en 7, en tiempo real)

Cambios de plano de los procesos. Operador &, comando bg , comando fg y comando nohup

El operador **&** se utiliza para iniciar un proceso en segundo plano. Se coloca detrás del proceso.

 Iniciar un proceso a segundo plano.

El comando **bg** se emplea para ejecutar un proceso en segundo plano. En lugar de utilizar el PID se hace uso del número de tarea, que es visible ejecutando el comando jobs.

 Ejecutar un proceso en segundo plano. (bg %numeroproceso)

El comando **fg** se emplea para ejecutar un proceso en primer plano. En lugar de utilizar el PID se hace uso del número de tarea, que es visible ejecutando el comando jobs.

 Ejecutar un proceso en primer plano . (fg %numeroproceso)

El comando **nohup** se usa para ejecutar un proceso haciéndolo inmune a los hang ups (cuelgues). De una forma simple, se evita que el proceso se cuelgue cuando se cierra la consola.

 Iniciar un proceso haciéndolo inmune a hang ups. (nohup proceso)

4. Gestión de servicios en GNU/Linux

Los servicios en Linux al igual que en Windows son programas que se ejecutan en segundo plano y que no requieren de la intervención del usuario, iniciándose antes de que se valide cualquier usuario.

Normalmente, a los servicios en Linux se les llama demonios (daemons), y pueden ser iniciados al arrancar el sistema, al entrar en un runlevel (nivel de ejecución) determinado, o simplemente cuando nosotros los iniciemos.

4.1.- Iniciando y parando servicios

Tradicionalmente init ha sido el programa encargado de controlar la secuencia de arranque de la inmensa mayoría de distribuciones GNU/Linux. Entre los diferentes programas init existentes, mayoritariamente se utilizaba el sistema de arranque de “System-V”, en el cual la ejecución de los distintos procesos estaba gestionada según distintos niveles de ejecución o runlevels.

Desde Ubuntu 6.10, se añade al clásico init de System-V un nuevo sistema llamado Upstart. En Upstart la gestión de arranque está basada en eventos, y no en niveles como en System-V. Los servicios se pueden iniciar o parar en respuesta a ciertos eventos, y además, se puede reiniciar servicios que se cierran inesperadamente. Esto supone una gran mejora sobre el clásico init y sus variantes, aunque se permite también el uso del sistema antiguo.

Por si este cambio fuera poco, en 2011 entra en juego un nuevo sistema de inicio de Linux y de gestión de servicios compatible con sysV llamado systemd, el cual es el que actualmente se está usando por defecto en la mayoría de distribuciones Linux. A partir de Ubuntu 15.04 deja de usarse Upstart en beneficio de systemd por defecto. La principal diferencia respecto a Upstart, es que ya no requiere satisfacer ciertas dependencias al momento de cargar los componentes, Upstart requiere de una causa (evento) para generar el siguiente paso, sin embargo systemd no necesita de tal evento, lo cual lógicamente hace que el proceso sea mas rápido.

Debido a esto, en Ubuntu actualmente tenemos varios modos diferentes de iniciar y parar los servicios, ya que todavía funciona el método utilizado con System-V, aunque parece que poco a poco se irá limitando su uso al igual que el de Upstart en favor de systemd que es el que se está imponiendo.

Método 1: el clásico de System-V

Los programas que ejecutamos como demonios pueden estar ubicados en cualquier lugar del sistema de ficheros, pero todos utilizan un script para manejarlos que se encuentran en el directorio: “/etc/init.d/”.

Por tanto, la sintaxis para iniciar/parar demonios que se encuentran en “/etc/init.d/” es:

$sudo /etc/init.d/<nombre_servicio> start (inicia el servicio)

$sudo /etc/init.d/<nombre_ servicio> stop (detiene el servicio)

Dependiendo del servicio tenemos además:

$sudo /etc/init.d/<nombre_ servicio> restart (detiene e inicia el servicio)

$sudo /etc/init.d/<nombre_ servicio> reload (carga la configuración nuevamente sin detener el servicio)

$sudo /etc/init.d/<nombre_ servicio> status (muestra el estado del servicio)

Método 2: usando el comando service

El comando “service” nos permite ejecutar servicios basados en el servicio de arranque de System-V y de Upstart:

$sudo service <nombre_ servicio> start (inicia el servicio)

$sudo service <nombre_ servicio> stop (detiene el servicio)

$sudo service <nombre_ servicio> restart (detiene e inicia el servicio)

$sudo service <nombre_ servicio> reload (carga la configuración nuevamente sin detener el servicio)

$sudo service <nombre_ servicio> status (muestra el estado del servicio)

$service --status-all (Obtenemos una lista de todos los servicios, tanto si son gestionados por Upstart o System-v, pero no nos vale para saber su estado real.)

Método 3: usando los comandos de upstart (obsoleto) *

Con el nuevo sistema de arranque de Ubuntu (upstart), tenemos una nueva manera de iniciar y parar servicios a través de los siguientes comandos:

$sudo start <nombre_ servicio> (inicia el servicio)

$sudo stop <nombre_ servicio> (detiene el servicio)

$sudo restart <nombre_ servicio> (detiene e inicia el servicio)

$sudo reload <nombre_ servicio> (carga la configuración nueva-mente sin detener el servicio)

$sudo status <nombre_ servicio> (muestra el estado del servicio)

Además, para ver el estado de los servicios controlados por Upstart tenemos el comando:

$initctl list

Método 4: usando los comandos de systemd

Systemd incorpora nuevos comandos para gestionar los servicios o demonios del sistema. Veámoslos:

$**sudo systemctl start <nombre_ servicio>.service** (inicia el servicio)

$**sudo systemctl stop <nombre_ servicio>.service** (detiene el servicio)

$**sudo systemctl restart <nombre_ servicio>.service** (detiene e inicia el servicio)

$**sudo systemctl reload <nombre_ servicio>.service** (carga la configuración nueva-mente sin detener el servicio)

$**sudo systemctl status <nombre_ servicio>.service** (muestra el estado del servicio)

Además, para ver el estado de los servicios controlados por systemd tenemos el comando:

$**systemctl status**

Upstart no utiliza el directorio "**/etc/init.d**", pero Ubuntu si lo incluye debido a que muchos de los scripts de este directorio no están disponibles todavía para el sistema Upstart. Aquellos servicios que si funcionan ya con el sistema Upstart, permiten ser gestionados usando el sistema System-V, pero aparece entonces un mensaje de advertencia recomendándonos que usemos el nuevo sistema de inicio y parada de Upstart (método 3).

4.2.- Servicios basados en runlevels

El sistema System-V sólo es capaz de ejecutar o parar procesos ante la entrada en un runlevel (nivel de ejecución).

El runlevel es cada uno de los estados de ejecución en que se puede encontrar el sistema Linux. Existen 7 niveles de ejecución en total, pero en Debian / Ubuntu no se hace distinción entre los niveles 2 al 5:

Runlevel 0: Apagar la máquina.

Runlevel 1: Arrancar en Modo Monousuario

Runlevel 2 al 5: Arrancar en Modo Multiusuario con entorno gráfico si está instalado.

Runlevel 6: Reiniciar la máquina.

Para indicar los procesos que se han de ejecutar o parar en cada runlevel, existen los directorios “/etc/rcX.d/”, donde X es el número de runlevel.

Dentro de estos directorios, lo que hay son enlaces simbólicos a los scripts existentes en “/etc/init.d/”. Los nombres de estos enlaces empiezan por la letra “S” (start / iniciar) o “K” (kill / parar) seguido de un número y el nombre del servicio al que referencian. El número de dos dígitos (de 00 a 99) indica el orden en que se inicia o para el servicio. A un número menor se iniciará antes que otro con uno mayor.

Para que un servicio no se inicie en un determinado runlevel, solo debemos cambiar el nombre al archivo del servicio deseado del directorio “/etc/rcX.d” correspondiente, cambiando la S inicial por la K.

Seleccionar un runlevel en el arranque de Ubuntu

En el fichero "/etc/init/rc-sysinit.conf" modificamos la siguiente línea:

---------- Fichero /etc/init/rc-sysinit.conf ----------

[…]

env DEFAULT_RUNLEVEL=2

[…]

---------- Fin Fichero /etc/init/rc-sysinit.conf ----------

Con independencia del runlevel seleccionado se ejecutarán todos los servicios de /etc/rcS.d/.

Seleccionar un runlevel una vez iniciado el sistema

- **Ver el runlevel en el que nos encontramos:**

  $runlevel

- **Cambiar el sistema de runlevel:**

  $init <número que indique el runlevel al que queremos ir>

5.- Enlaces en Linux

En Linux, cada archivo en el sistema está representado por un inodo. Un inodo no es más que un bloque que almacena información de los archivos (el índice del fichero), de esta manera a cada inodo podemos asociarle un nombre. A simple vista pareciera que a un mismo archivo no podemos asociarle varios nombres, pero gracias a los enlaces esto es posible.

Un enlace físico no es más que una etiqueta o un nuevo nombre asociado a un archivo. Es una forma de identificar el mismo contenido con diferentes nombres. Éste enlace no es una copia separada del archivo anterior sino un nombre diferente para exactamente el mismo contenido.

Para crear un enlace físico en Linux del archivo archivo.txt a nuevo_nombre.txt, ejecutamos:

$ **ln archivo.txt nuevo_nombre.txt**

El enlace aparecerá como otro archivo más en el directorio y apuntará al mismo contenido de archivo.txt. Cualquier cambio que se haga se reflejará de la misma manera tanto para archivo.txt como para nuevo_nombre.txt, ya que realmente son simplemente dos etiquetas que apuntan a los mismos datos en el disco duro.

Un enlace se puede borrar usando el comando **rm** de la misma manera en que se borra un archivo, sin embargo, el contenido del inodo no se eliminará mientras haya un enlace físico que le haga referencia. Esto puede tener varias ventajas, pero también puede complicar la tarea de seguimiento de los archivos. Un enlace físico tampoco puede usarse para hacer referencia a directorios o a archivos en otros equipos evidentemente.

Un enlace simbólico también puede definirse como una etiqueta o un nuevo nombre asociado a un archivo, pero a diferencia de los enlaces físicos, el enlace simbólico no contiene los datos del archivo, simplemente apunta al archivo donde se encuentran los datos.

Para crear un enlace simbólico del archivo archivo.txt a nuevo_nombre.txt, ejecutamos:

$ **ln -s archivo.txt nuevo_nombre.txt**

Sobre un enlace simbólico también se pueden usar todos los comandos básicos de archivos (rm, mv, cp, etc). Sin embargo, cuando el archivo original es borrado o movido a una ubicación diferente el enlace dejará de funcionar y se dice que el enlace está roto.

Un enlace simbólico permite enlazar directorios, cosa que no podíamos hacer con uno físico.

6. Programación de tareas

Existen una serie de tareas que en determinados momentos nos interesaría ejecutar sin estar delante del ordenador. Por ejemplo, realizar copias de seguridad por la noche, programar el apagado automático a una hora por si se nos olvida apagar el equipo (muy útil en empresas) o saber en cada momento qué usuarios están conectados en una máquina. Ubuntu (y Linux en general) proporcionan una serie de aplicaciones para llevar a cabo esta tarea. Pasemos a estudiarlas.

Antes de programar las tareas hay que comprobar que el servicio **cron** se encuentra en ejecución mediante el comando:

```
#service cron status
```

Para modificar el fichero de configuración de cron, ejecuta el comando:

```
#crontab –e
```

La sintaxis de las tareas programadas es:

```
# .---------------- minuto (0 - 59)
# |   .------------- hora (0 - 23)
# |   |   .---------- día del mes (1 - 31)
# |   |   |   .------- mes (1 - 12) o jan,feb,mar,apr ... (los meses en inglés)
# |   |   |   |   .---- día de la semana (0 - 6) (Domingo = 0 o 7) o sun,mon,tue,wed,thu,fri,sat (los días en inglés)
# |   |   |   |   |
# |   |   |   |   |
  *   *   *   *   *  Comando a ejecutar
```

Además, de este método para programar tareas, crontab también permite al administrador que cree tareas modificando el fichero /etc/crontab, con el siguiente formato casi idéntico a lo visto anteriormente:

Por último, otra forma de poder programar tareas es guardar el script o programa que quiere ejecutar en las siguientes carpetas de configuración de cron:

/etc/cron.hourly # Ejecuta el script cada hora

/etc/cron.daily # Ejecuta el script diariamente

/etc/cron.weekly # Ejecuta el script semanalmente

/etc/cron.monthly # Ejecuta el script mensualmente

Para asegurar el sistema sólo el usuario root puede modificar los scripts que ejecuta crontab.

Ejemplos de crontab

- 0 23 10 * * /home/usuario/copia.sh

  Ejecuta el script “copia.sh” todos los días 10 de cada mes a las 11:00 de la noche.

- 55 23 * * 0 /home/usuario/copia_semanal.sh

  Ejecutará el script “copia_semanal” todos los domingos a las 23:55h.

- 30 17 * * 1,2,3,4,5 /home/usuario/copia.sh

  Ejecuta el script a las 17:30h todos los días de lunes a viernes.

- 00 12 1,15,28 * * /home/usuario/copia.sh

  Ejecuta el script a las 12 a.m. todos los días 1, 15 y 28 de cada mes.

- 0 9,17 * * 1-5 /home/usuario/copia.sh

  Ejecuta el script “copia” de lunes a viernes a las 9:00h y a las 17:00h.

Comando at

El comando o programa at, permite planificar la ejecución de ciertas tareas en un momento dado.

Tanto el administrador del sistema como los usuarios tienen acceso a esta herramienta.

Si no se pudiese ejecutar , habría que instalarlo: apt install at

Creación de tareas.

El comando “at” tiene la siguiente sintaxis:

at [opciones] [hora] [fecha]

Al ejecutar el comando at el sistema se queda esperando a que el usuario introduzca una serie de comandos que quiere ejecutar en el día y hora indicado. Para terminar de introducir comandos, hay que pulsar la combinación de teclas <CTRL+D>.

El comando at acepta horas con formato HH:MM para ejecutar un trabajo a una determinada hora del día (si esa hora ya ha pasado, se asume que es del día siguiente) y fechas con el formato MMDDAA o MM/DD/AA.

Otra opción muy útil es: “now + X unidades”, que cuenta a partir de ahora más un tiempo indicado, siendo las unidades permitidas “minutes (minutos), hours (horas) o days (días)”. Para ejecutar una tarea dentro de 5 minutos:

$at now + 5 minutes

Ejemplo:

$at 21:00 102411

at> cp /etc/network/interfaces /backup

at> <CTRL+D>

job 1 at Mon Oct 24 21:00:00 2011

Lo que hará este ejemplo es copiar el fichero “interfaces” en el directorio /backup el 24 octubre de 2011 a las 21:00 horas.

Ver listado de tareas creadas.

Para ver las tareas que están pendientes de ejecución podemos poner:

```
$atq
```

O también:

```
$at -l
```

Ver contenido de una tarea en concreto.

Para ver el contenido de una tarea en concreto debes poner:

```
$at -c <número de tarea>
```

Eliminar tareas.

Para eliminar tareas creadas hacemos lo siguiente:

```
$atrm <número de tarea>
```

O también:

```
$at –d <número de tarea>
```

El número de tarea lo podemos saber usando atq o at -l.

7. Monitorización del sistema en Linux.

Las herramientas que se utilizan para la monitorización en Linux podemos agruparlas de la siguiente manera:

- **Herramientas integradas**: son herramientas propias del sistema, que existen en la gran mayoría de las distribuciones Linux de forma predeterminada.
- **Monitor del sistema**: es una herramienta que permite monitorizar los procesos que se encuentran en ejecución en el sistema. Realmente se trata de una herramienta integrada pero la diferenciamos por sus características y su importancia.
- **Herramientas Sysstat**: es una colección de herramientas de monitorización que, además de proporcionar datos de rendimiento en tiempo real permite almacenarlos como históricos para futuras referencias.

7.1.- Monitorización a través de herramientas integradas.

La gran variedad de distribuciones Linux hace que el abanico de herramientas integradas disponibles para la monitorización, sea demasiado extenso. De entre todas ellas, por su importancia y presencia en la gran mayoría de las versiones, destacamos las siguientes:

- **uptime**: monitoriza la carga del sistema. Presenta la hora del sistema, cuánto tiempo lleva operativo, el número de usuarios conectados y el valor medio de la carga en el último minuto, los últimos 5 minutos y los últimos 10 minutos.
- **time**: monitoriza el tiempo de ejecución de un programa. Permite conocer cómo se ha distribuido el tiempo de ejecución de su código, en modo usuario y en modo supervisor.
- **top**: monitoriza la actividad de los procesos. Visualiza los procesos que hay en ejecución y cuánta memoria consumen. Por defecto la información se actualiza cada 5 segundos pero puede personalizarse.
- **ps**: monitoriza la actividad de los procesos. Muestra los procesos lanzados en el sistema por el usuario que ha ejecutado la orden.
- **vmstat**: monitoriza la actividad de la memoria. Informa sobre el uso de la memoria física y virtual, de la actividad de intercambio entre la memoria y el disco, las transferencias, las interrupciones, los cambios de contexto y el uso del procesador.
- **free**: monitoriza la actividad de la memoria. Informa sobre el uso de la memoria física y la swap.
- **w**: monitoriza la actividad de los usuarios del sistema. Permite obtener información sobre los usuarios conectados al equipo y ver que están haciendo (indica el proceso que está ejecutando el usuario).

Directorio /proc

El directorio /proc contiene un sistema de archivos imaginario o virtual. Este no existe físicamente en disco, sino que el núcleo lo crea en memoria. Se utiliza para ofrecer información relacionada con el sistema (originalmente acerca de procesos, de aquí su nombre). El núcleo de Linux almacena archivos en este directorio con información relativa a su funcionamiento, de tal forma que, para analizar el comportamiento de un sistema, también se puede recurrir a la consulta de los archivos de este sistema de ficheros. De hecho, prácticamente todas las herramientas analizadas obtienen sus datos de esta fuente.

Dentro del directorio /proc vemos dos tipos de contenido distintos, directorios numerados y ficheros de información del sistema, eso sí hay que destacar que /proc es un sistema de ficheros virtual, si vemos el tamaño que tiene un fichero determinado veremos que ocupa 0 bytes:

```
$ ls -l /proc/cpuinfo
-r--r--r-- 1 root root 0 feb 26 13:58 /proc/cpuinfo
```

Los directorios con nombres y números representan identificadores de procesos, y dentro de uno de ellos está el proceso con ese PID. Normalmente son comandos, variables, descripción de ficheros, limites de procesos y puntos de montaje.

Tenemos una serie de ficheros en /proc sobre información del sistema.

Los más importantes son:

- /proc/cpuinfo – información sobre CPU
- /proc/meminfo – Información sobre Memoria.
- /proc/loadvg – Carga media del sistema. Las primeras tres columnas miden uso de CPU en periodos de 1, 5 y 10 minutos respectivamente. La cuarta columna el número de procesos activos y el número total de procesos. La cuarta el último ID del proceso usado.
- /proc/partitions – Información relativa a particiones.
- /proc/version – Versión Linux

Luego hay algunos comandos en Linux que leen información de este directorio para mostrarnos de una forma cómoda la información.

/proc/cmdline – Este archivo muestra los parámetros pasados al kernel en el momento en que éste inicia.
/proc/devices – Lista de drivers de dispostivios configurados en el kernel actual.
/proc/dma – Canales DMA en uso (Acceso Directo a Memoria)
/proc/fb – Frame Buffer devices.
/proc/filesystems – Sistemas de ficheros soportados por el Kernel.
/proc/interrupts – Número de interrupciones en la arquitectura (IRQ)
/proc/iomem – Mapeo de la memoria para varios dispositivos
/proc/ioports – Lista de puertos de comunicaciones registrados con un dispositivo
/proc/locks – Ficheros bloqueados por el kernel
/proc/misc – Drivers registrados
/proc/modules – Lista de módulos cargados por el kernel
/proc/mounts – Todos los puntos de montaje del sistema.
/proc/pci – Dispositivos PCI en el sistema.
/proc/stat – Mantiene varias estadísticas del sistema desde el arranque.
/proc/swaps – Mide Swap y su uso
/proc/uptime – Información del tiempo del sistema arrancado.
/proc/version – Versión del kernel gcc (compilador) y versión de la distro.

Conviene aclarar que aunque los archivos anteriores tienden a ser archivos de texto fáciles de leer, algunas veces pueden tener un formato que no sea fácil de interpretar. Por ello existen muchos comandos que solamente leen los archivos anteriores y les dan un formato distinto para que la información sea fácil de entender.

Por ejemplo, el comando free, lee el archivo /proc/meminfo y convierte las cantidades dadas en bytes a kilobytes (además de agregar un poco más de información extra).

7.2.- Monitor del sistema.

El Monitor del sistema en Linux es el interfaz gráfico de la orden top vista anteriormente. Este interfaz viene instalado por defecto en la mayoría de las distribuciones y puede encontrarse en la ruta: Sistema→Administración→Monitor del sistema. Si no estuviera, habría que instalarlo a través del paquete gnome-system-monitor, disponible en los repositorios oficiales.

La apariencia de este Monitor es muy similar al Administrador de tareas de Windows. Ofrece las siguientes opciones:

Sistema: proporciona información básica como puede ser la versión del sistema operativo, el procesador integrado o la memoria RAM disponible.

Procesos: contiene un listado con los procesos del sistema, su estado y la carga media de los últimos minutos. También se puede crear un filtro para mostrar sólo determinados procesos, como pueden ser los procesos activos o los procesos del usuario. Sobre los procesos pueden realizarse varias operaciones como, por ejemplo, cambiar su prioridad, ver su mapa de memoria o finalizarlo (también denominado matarlo).

Recursos: informa a través de gráficos dinámicos el uso de los principales recursos del sistema (el procesador, la memoria y la red). Es la forma más directa de monitorización. Esta pestaña permite ser personalizada.

Sistemas de archivos: muestra un listado con los sistemas de archivos montados y sus principales características (espacio libre, punto de montaje, etc…).

A través del **Monitor del sistema** no sólo se puede ejercer una monitorización de los principales elementos del sistema sino que también se puede modificar su comportamiento. Esto se hace aplicando una configuración directamente sobre el **Monitor**. La opción de configurar está disponible en el **Monitor** ejecutando la secuencia **Editar→Preferencias**. Las preferencias hacen alusión principalmente a las pestañas de Procesos y Recursos, que son las que manejan los elementos más dinámicos.

7.3.- Monitorización con Sysstat.

Sysstat es un conjunto de herramientas orientadas a monitorizar el rendimiento de equipos con sistema operativo Linux. Al igual que sucede con las herramientas que hemos estudiado, tienen la capacidad de proporcionar información en tiempo real o de recopilarla para un posterior análisis. El paquete se llama sysstat y está disponible en los repositorios oficiales.

De todas las herramientas de que consta Sysstat, destacamos las siguientes:

mpstat: genera informes del rendimiento de cada procesador del sistema.

iostat: genera informes de la actividad de la CPU y de los dispositivos de E/S.

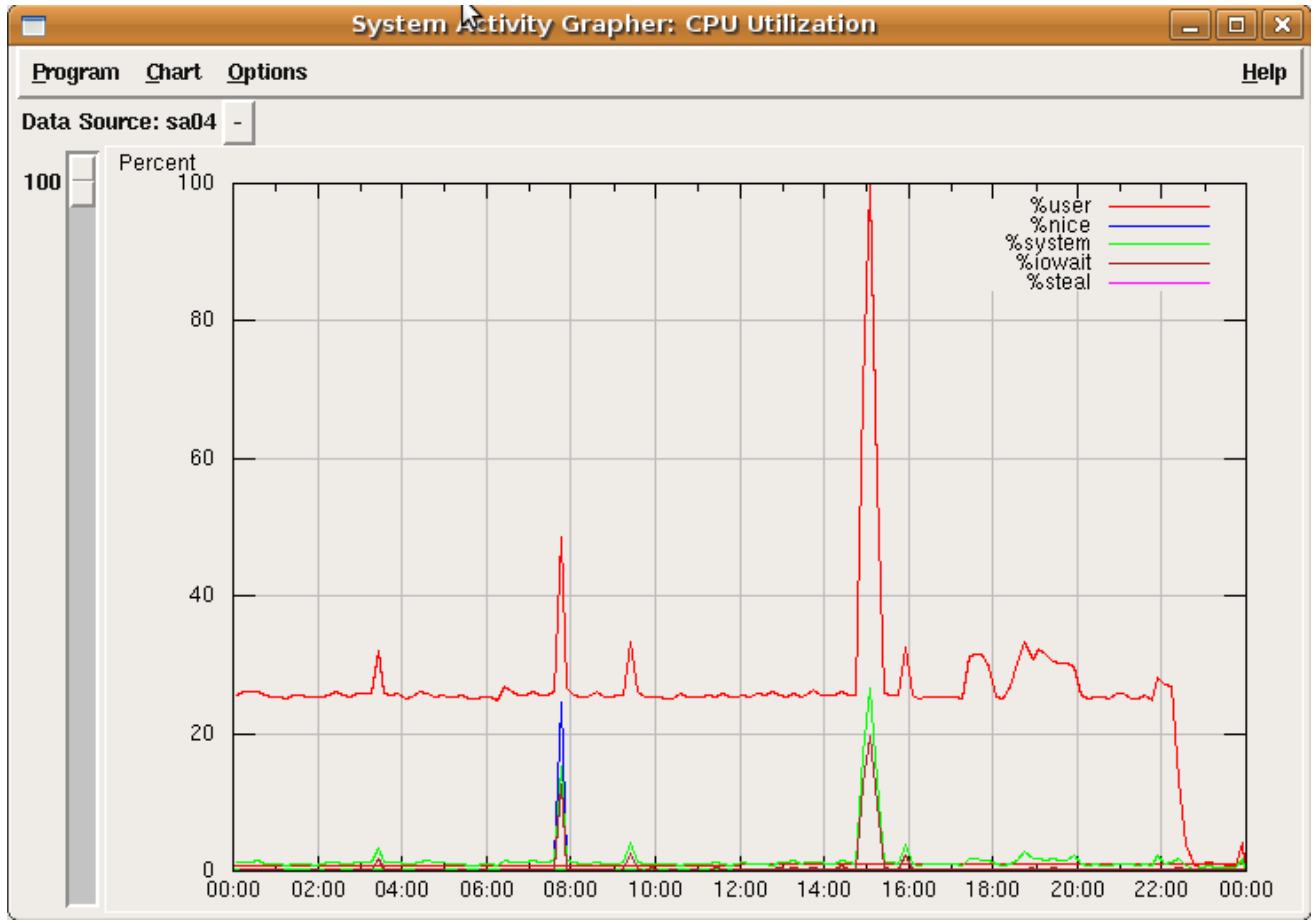
pidstat: genera informes de las tareas gestionadas por el kernel.

sar: recoge la información de rendimiento de todas las áreas del sistema (CPU, memoria, red,…).

sadf: permite exportar los datos recopilados por sar en diferentes formatos (XML, CVS,…).

Para poder recoger datos de forma periódica y guardarlos en ficheros históricos hay que editar el archivo **/etc/default/sysstat** y poner la variable **ENABLED a "true**".

A la hora de interpretar la información podemos hacer uso de **isag**, un programa con interfaz gráfico que se instala independiente de **sysstat** y que nos permite crear gráficas de la información recopilada que tenemos en los ficheros de rendimiento.


System Activity Grapher: CPU Utilization
Program
Chart
Options
Help
Data Source: sa04
-
100
Percent
100
80
60
40
20
0
%user
%nice
%system
%iowait
%steal
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00

7.4.- Registros del sistema: Logs o bitácoras.

Linux tiene una serie de **archivos de registro o logs** (también conocido como bitácoras) que ayudan al administrador en la monitorización y en el mantenimiento de una máquina e incluso de una red. En estos archivos se van almacenando registros de todos los eventos que ocurren en un sistema.

Estos registros ayudan a la localización de errores, ataques maliciosos y hasta podemos saber que hacen nuestros usuarios en el sistema. Como ejemplo podemos saber la fecha y hora en la que acceden los usuarios al sistema, lo que ocurre en el arranque del equipo, etc.

En Ubuntu, al igual que en muchas otras distribuciones Linux, podemos encontrar estos archivos de registro o logs en la carpeta **"/var/log".** En esta carpeta encontraremos casi todos los archivos de registros del sistema, aunque puede haber aplicaciones que creen estos archivos en sus propias carpetas fuera de /var/log.

Para solucionar muchos de los problemas que nos podemos encontrar en Linux, debemos saber que es lo que lo provoca, y eso lo podemos averiguar en la mayoría de los casos gracias a estos logs, por lo que su manejo es fundamental para cualquier usuario de Linux.

Los ficheros de registro más habituales e importantes son (puede ser que no tengas todos estos en tu distribución):

/var/log/messages: Este es el archivo principal de registros donde varias aplicaciones del sistema y demonios almacenan mensajes de interés. Este es el primer archivo que un administrador debería consultar si algo funciona mal.

/var/log/dmesg:En este archivo se almacena la información que genera el kernel durante el arranque del sistema. Podemos ver su contenido con el comando dmesg.

/var/log/syslog: syslog es un log del sistema y del kernel que nos puede dar importante información de eventos que suceden en el sistema y en sus programas.

/var/log/auth.log: Almacena mensajes sobre autenticación de usuarios y permisos.

/var/log/boot.log: Almacena mensajes relacionados con el arranque del sistema.

/var/log/daemon.log: Almacena mensajes sobre demonios o servicios que funcionan en el sistema.

/var/log/dpkg.log: Almacena un registro de los paquetes binarios instalados.

/var/log/kern.log:Almacena los registros del kernel, generados por klogd.

/var/log/Xorg.0.log: Almacena mensajes de Xorg.

/var/log/lpr.log: Almacena mensajes relacionados con la impresora.

/var/log/debug: Almacena mensajes de depuración.

/var/log/mail.*: Formado por varios archivos que almacenan mensajes sobre errores, información, alertas y registro para los correos (requiere tener un servicio de correos funcionando).

/var/log/user.log: Muestra información acerca de los procesos usados por el usuario.

Manejando logs

Visualización básica de logs.

Tenemos diferentes opciones para ver el contenido de estos ficheros logs desde la terminal:

El comando **cat**:

$cat /var/log/syslog

Lo puedo concatenar con “grep” para buscar cadenas determinadas:

$cat /var/log/syslog | grep ntp

El comando **less**:

$less /var/log/syslog

Este comando no carga todo el contenido en memoria por lo que consume pocos recursos. Me puedo mover fácilmente por el fichero usando los cursores. Para salir pulsamos la tecla q.

El comando **tail**:

$tail /var/log/syslog

Tail muestra las últimas líneas de un archivo. Por defecto 10 líneas.

Si queremos ver más o menos líneas ponemos: $tail –n <nº líneas a mostrar> <fichero>

Ejemplo:

$tail –n 20 /var/log/syslog

Monitorización de logs

Una de las tareas más utilizadas por administradores de sistemas Linux es la posibilidad de ver en una o varias terminales los sucesos que ocurren en el sistema en tiempo real.

Imaginemos ver la llegada de un email, ver la entrada de un usuario al sistema o ver como se establece una conexión VPN justo en el momento en que se producen, sin tener que estar buscando los registros una vez haya sucedido los eventos.

Monitorización desde el terminal

Para este propósito usaremos nuevamente el comando tail pero con la opción –f:

```
$tail -f /var/log/syslog
```

Para salir usamos la combinación de teclas <Control+C>.

También podemos hacer esto con el comando less:

```
$less /var/log/syslog
```

Pulsamos las teclas <Shift + f>

Ejemplo:

```
#/etc/init.d/networking restart | tail -f /var/log/syslog
```

Podemos ver como aparecen sucesos relacionados con la red y otros servicios relacionados con ntp para sincronizar la hora entre otros.

Monitorización con aplicaciones gráficas

Además de poder ver los registros del sistema mediante el terminal, también existen aplicaciones gráficas de escritorio para trabajar con los logs del sistema. Entre ellas tenemos:

Visor de sucesos del sistema (GNOME-System-Log, paquete gnome-utils).

Xwatch (paquete xwatch): monitor de logs para las X.

Logs más legibles

La lectura de los logs en el terminal no es nada clara en la mayoría de las ocasiones. Para hacer los logs más legibles vamos a utilizar un aplicación que va a colorear el contenido de los mismos haciendo más fácil su lectura.

Para ello instalamos el paquete **ccze**:

```
#apt-get install ccze
```

Diferentes formas de usar este programa:

Con el comando tail:

```
$tail -f /var/log/syslog | ccze
```

Con el comando less:

```
$ccze -A < /var/log/syslog | less –R
```

Este comando nos permite usar el comando less de manera normal con el texto coloreado.

Exportar los logs a HTML

Además de colorear los logs del sistema, ccze nos permite también crear un archivo HTML con el contenido del log.

Esta funcionalidad es muy interesante si tenemos un servidor web instalado en el equipo, pues podríamos verlo desde cualquier ubicación.

Ejemplos:

```
$ccze -h -o nolookups </var/log/syslog > ~/syslog.html
```

```
$ccze -h -o nolookups </var/log/syslog > /var/www/logs/syslog.html
```

Creación de registros logs propios.

El comando “**logger**” permite enviar mensajes al sistema de registros o logs del sistema desde un terminal o un script propio.

Ejemplo: Escribe en la terminal lo siguiente:

$logger -t mi_programa -f /var/log/syslog "Mensaje de prueba..."

Este comando añadirá al fichero “/var/log/syslog” la línea: Mar 2 12:37:07 Ubuntu-Server mi_programa: Mensaje de prueba…

Rotación de ficheros logs.

Los archivos de registro con el tiempo pueden volverse muy grandes y difíciles de manejar, siendo muy común que estos ficheros terminen llenando el disco duro de nuestro servidor.

El sistema de registros de Linux proporciona un comando que permite archivar los ficheros de registros, creando ficheros nuevos con la información más actual, evitando así que se mezcle con información más antigua. A esto se le conoce como rotación de ficheros logs. El programa encargado de esta tarea es "logrotate".

Logrotate se ejecuta automáticamente cada cierto tiempo (normalmente una vez al día), aunque también podemos ejecutarlo manualmente. Cuando es ejecutado este comando, según este configurado toma la versión actual de los archivos de log y les añade un ".1" al final del nombre de archivo. Luego, cualquier otro archivo rotado previamente será renombrado secuencialmente a ".2", ".3", etc. Cuando más grande es el número del log, más antiguo es.

El archivo de configuración de "logrotate" es "/etc/logrotate.conf", consistente en una serie de especificaciones para los grupos de archivos de log que vamos a administrar.

De esta manera, en este archivo indicamos la configuración de rotación de cada archivo log que me interese, pudiendo indicar parámetros como cada cuanto tiempo se van a rotar (cada mes, cuando se supere un tamaño, ...), cuantos archivos de rotación voy a tener como máximo, si se va a comprimir el fichero de rotación, que tipo de suceso queremos registrar, etc...

Además del fichero anterior, existe también un directorio donde otras aplicaciones que instalemos pueden guardar su configuración para sus propios ficheros log. Este directorio es /etc/logrotate.d.

Por ejemplo, el fichero de configuración de apt se encuentra en este directorio:

```
$cat /etc/logrotate.d/apt
```