Prácticas de Sistemas Operativos

Módulo I. Administración de Linux

Sesión 3. Monitorización del sistema.

1. Introducción a al sesión 3

Esta sesión está pensada, en primer lugar, para que el estudiante se familiarice con las distintas herramientas disponibles para poder monitorizar un sistema y así comprobar el rendimiento del mismo y encontrar los problemas que se puedan estar produciendo. Para ello es muy importante tener información sobre: los procesos que están en ejecución, la cantidad de memoria principal disponible, el espacio disponible en disco, el número de particiones, etc.

Cuando un administrador de un sistema detecta un problema de rendimiento debería seguir la siguiente secuencia de pasos:

- Definir el problema lo más detalladamente posible.
- Determinar la causa o causas del problema, para ello se utilizará alguna de las herramientas que veremos en esta sesión.
- Pensar una solución para mejorar el rendimiento del sistema.
- Llevar a cabo dicha solución, diseñando e implementando las modificaciones al sistema necesarias.
- Monitorizar el sistema para determinar si los cambios realizados han sido efectivos.

Nota: Cuando realices cada una de las actividades propuestas en esta sesión de prácticas, no debes limitarte a ejecutar las órdenes de manera automática y apuntar los resultados. Lo importante es analizar los números que se muestran, y ver cómo cambian cuando modificamos la prioridad de un proceso, o cuando se hace uso de manera intensiva del disco.

2. Objetivos principales

- Conocer y saber usar las órdenes para poder controlar y gestionar el uso de la CPU en un sistema Linux.
- Conocer y saber usar las órdenes para controlar y gestionar la memoria de un sistema.

- Conocer y saber utilizar las órdenes para controlar y gestionar los dispositivos del sistema.
- Saber repartir los recursos del sistema estableciendo límites para éstos.
- Conocer y comprender la diferencia entre los conceptos: ***metadatos del SA*** y ***metadatos de un archivo*** en particular (**inodo**, en inglés ***inode***).
- Conocer y comprender el concepto de ***archivos de tipo enlace***.
- Explorar la potencia de la orden **`ls`** de cara a la consulta de información del SA.
- Profundizar en la compresión de la característica "*interfase abstracta de trabajo con dispositivos*" que proporcionan los SA tipo UNIX. Saber crear ***archivos especiales de dispositivo*** con la orden **`mknod`.**
- Conocer los recursos del SA: bloques e inodos, y saber utilizar las órdenes para consultar su estado: Espacio usado y espacio disponibles en un SA, número de archivos creados y cuántos archivos se podrán crear como máximo.

3. Control y gestión de CPU

En este punto vamos a ver algunas órdenes para comprobar el estado del sistema y su rendimiento. Entre otros aspectos, veremos cómo podemos conocer los recursos que consumen los procesos iniciados, qué usuarios están trabajando en el sistema y qué programas se están ejecutando, el trabajo de un procesador o de todos los que haya en el equipo, etc.

3.1. Orden `uptime`

Muestra una línea con la siguiente información: la hora actual, el tiempo que lleva en marcha el sistema, el número de usuarios conectados, y la carga media del sistema en los últimos 1, 5 y 15 minutos (es la misma información que devuelve la ejecución de la orden **`w`**, la cual muestra los usuarios conectados y lo que están haciendo). La carga del sistema es el número de procesos en la cola de ejecución del núcleo. Un sistema con un comportamiento normal debe mostrar una carga igual o inferior a 1[4], aunque se deben tener en cuenta la configuración y el tipo de programas en ejecución. A continuación se muestra un listado por pantalla de la ejecución de la orden **`uptime`** y de la orden **`w`**:

```
#> uptime
 18:32:07 up 15 min,  3 users,  load average: 0.20, 0.26, 0.16

#> w
 18:33:36 up 17 min,  3 users,  load average: 0,04, 0,19, 0,15
USER     TTY      FROM    LOGIN@   IDLE   JCPU    PCPU  WHAT
patricia tty7     :0      18:16      ?    20.00s  0.22s x-session-manag
patricia pts/0    :0.0    18:19    11:40   0.30s  4.86s gnome-terminal
patricia pts/1    :0.0    18:22    0.00s   0.18s  0.00s w
```

4 En sistemas *multi-core* habría que multiplicar este número por el número de *cores*.

Actividad 3.1

Responde a las siguientes cuestiones y especifica, para cada una, la opción que has utilizado (para ello utiliza **man** y consulta las opciones de las órdenes anteriormente vistas:

a) ¿Cuánto tiempo lleva en marcha el sistema?

b) ¿Cuántos usuarios hay trabajando?

c) ¿Cuál es la carga media del sistema en los últimos 15 minutos?

3.2. Orden `time`

Mide el tiempo de ejecución de un programa y muestra un resumen del uso de los recursos del sistema. Muestra el tiempo total que el programa ha estado ejecutándose (real), el tiempo que se ha ejecutado en modo usuario (user) y el tiempo que se ha ejecutado en modo supervisor (sys), de tal forma que se puede determinar que el tiempo de espera de un proceso es:

T_espera = real – user - sys

En el siguiente ejemplo hemos ejecutado **time** con el programa **ps**, y como podemos observar, primero ejecuta **ps** y después nos muestra los valores de los distintos tiempos: **real**, **user** y **sys**:

```
#> time ps
  PID TTY          TIME CMD
 5836 pts/1    00:00:00 bash
 5896 pts/1    00:00:00 man
 5906 pts/1    00:00:00 tbl
 5907 pts/1    00:00:00 nroff
 5908 pts/1    00:00:00 pager
 5909 pts/1    00:00:00 locale
 5910 pts/1    00:00:02 find
 5928 pts/1    00:00:00 ps

real  0m0.019s
user  0m0.000s
sys   0m0.016s
```

3.3. Órdenes `nice` y `renice`

Linux realiza una planificación por prioridades. Por defecto, un proceso hereda la prioridad de su proceso padre. Se puede establecer el valor de prioridad de un proceso a un valor distinto del que tendría por defecto mediante la orden **nice**. El rango de valores que permite establecer como parámetro la orden **nice** es [-20,19]. Asignar un valor negativo aumenta la posibilidad de ejecución de un proceso y solamente puede hacerlo el usuario **root.** Los usuarios sin privilegios de administración solo pueden utilizar los valores positivos de este rango, desfavoreciendo así la posibilidad de ejecución de un proceso.

Ejemplos:

```
#> nice -5 konqueror # Aumenta en 5 el valor de prioridad de la ejecución de
konqueror
#> nice --10 konqueror # Decrementa en 10 el valor de prioridad (sólo usuario root)
```

La orden **renice** permite alterar el valor de prioridad de uno o más procesos en ejecución.

```
#> renice 14 890       # Aumenta en 14 el valor de prioridad del proceso 890
```

Actividad 3.2

a) Crea un script o guión shell que realice un ciclo de un número variable de iteraciones en el que se hagan dos cosas: una operación aritmética y el incremento de una variable. Cuando terminen las iteraciones escribirá en pantalla un mensaje indicando el valor actual de la variable. Este guión debe tener un argumento que es el número de iteraciones que va a realizar. Por ejemplo, si el script se llama prueba_procesos, ejecutaríamos:

```
# prueba_procesos 1000
el valor de la variable es 1000
```

b) Ejecuta el guión anterior varias veces en *background* (segundo plano) y comprueba su prioridad inicial. Cambia la prioridad de dos de ellos, a uno se la aumentas y a otro se la disminuyes, ¿cómo se comporta el sistema para estos procesos?

c) Obtén los tiempos de finalización de cada uno de los guiones del apartado anterior.

3.4. Orden `pstree`

Visualiza un árbol de procesos en ejecución. Tiene una serie de opciones, algunas de ellas son:

-a	Muestra los argumentos de la línea de órdenes.
-A	Usa caracteres ASCII para dibujar el árbol.
-G	Usa los caracteres de VT100.
-h	Resaltar el proceso actual y sus antepasados.
-H	Igual que **-h**, pero para el proceso que se especifique.
-l	Usa un formato largo, por defecto las líneas se truncan.
-n	Ordena los procesos por el PID de su antecesor en vez de por el nombre (ordenación numérica)
-p	Desactiva el mostrar los PIDs entre paréntesis después del nombre de cada proceso.
-u	Si el uid de un proceso difiere del uid de su padre, el nuevo uid se pone entre paréntesis después del nombre del proceso.
-V	Visualiza información sobre la versión.
-Z	(SELinux) Muestra el contexto de seguridad para cada proceso.

Un ejemplo de la salida mostrada por pantalla tras la ejecución de esta orden es (el árbol de procesos se ha podado para no extender demasiado la ilustración):

```
# pstree
init─┬─NetworkManager
     ├─acpid
     ├─anacron───sh───run-parts───apt───sleep
     ├─atd
     ├─avahi-daemon───avahi-daemon
     ├─bluetoothd
     ├─bonobo-activati───{bonobo-activati}
     ├─console-kit-dae───63*[{console-kit-dae}]
     ├─cron
     ├─cupsd
     ├─2*[dbus-daemon]
     ├─dbus-launch
     ├─dd
     ├─fast-user-switc
     ├─gconfd-2
     ├─gdm───gdm─┬─Xorg
     │           └─x-session-manag─┬─bluetooth-apple
     │                             ├─evolution-alarm───{evolution-alarm}
     │                             ├─gnome-panel
     │                             ├─metacity
```

3.5. Orden `ps`

Esta orden se implementa usando el pseudo-sistema de archivos **`/proc.`** Muestra información sobre los procesos en ejecución:

USER : usuario que lanzó el programa

PID : identificador del proceso

PPID : identificador del proceso padre

%CPU : porcentaje entre el tiempo usado realmente y el que lleva en ejecución

%MEM : fracción de memoria consumida (es una estimación)

VSZ : tamaño virtual del proceso (código+datos+pila) en KB

RSS : memoria real usada en KB

TTY : terminal asociado con el proceso

STAT : estado del proceso que puede ser:

R: en ejecución o listo (Running o Ready)
S: durmiendo (Sleeping)

T: parado (sTopped)
Z: proceso Zombie
D: durmiendo ininterrumpible (normalmente E/S)
N: prioridad baja (> 0)
<: prioridad alta (< 0)
s: líder de sesión
l: tiene multi-thread
+: proceso foreground
L: páginas bloqueadas en memoria

La orden **`ps`** normalmente se ejecuta con las operaciones **`-ef`**, ya que "**`e`**" significa que se seleccione a todo proceso que esté en el sistema y "**`f`**" que se muestre información completa (aunque con la opción "**`l`**" se imprime en pantalla más información). Sin argumentos muestra los procesos lanzados por el usuario que ejecuta esta orden.

Actividad 3.3

a) La orden **`pstree`** muestra el árbol de procesos que hay en ejecución. Comprueba que la jerarquía mostrada es correcta haciendo uso de la orden **`ps`** y de los valores "**`PID`**" y "**`PPID`**" de cada proceso.

b) Ejecuta la orden **`ps`** con la opción **`-A`**, ¿qué significa que un proceso tenga un carácter "**`?`**" en la columna etiquetada como **`TTY`**?

3.6. Orden `top`

Esta orden proporciona una visión continuada de la actividad del procesador en tiempo real, muestra las tareas que más uso hacen de la CPU, y tiene una interfaz interactiva para manipular procesos.

Las cinco primeras líneas muestran información general del sistema:

- las estadísticas de la orden **`uptime`**
    - estadísticas sobre los procesos del sistema (número de procesos, procesos en ejecución, durmiendo, parados o zombies)
    - el estado actual de la CPU (porcentaje en uso por usuarios, por el sistema, por procesos con valor nice positivo, por procesos esperando E/S, CPU desocupada, tratando interrupciones hardware o software, en espera involuntaria por virtualización)
    - la memoria (memoria total disponible, usada, libre, cantidad usada en *buffers* y en memoria caché de página)
    - el espacio de swap (swap total disponible, usada y libre)

  Los datos de la parte inferior son similares a los del **`ps`**, excepto SHR que muestra la cantidad de memoria compartida usada por la tarea.

Ordena los procesos mostrados en orden decreciente en base al uso de la CPU.

La lista se actualiza de forma interactiva, normalmente cada 5 segundos.

La orden **top** permite realizar una serie de acciones sobre los procesos de forma interactiva, como por ejemplo:

14. Cambiar la prioridad de alguno utilizando la opción “r”.
15. Matar o enviar una señal con la opción “k”.
16. Ordenarlos según diferentes criterios (por PID con “N”, uso de CPU con “P”, tiempo con “A”, etc.).
17. Con “n” se cambia el número de procesos que se muestran.
18. Para salir se utiliza la letra “q”.

Un ejemplo de una ejecución de **top**:

```
#>  top

top - 18:25:13 up 8 min,  3 users,  load average: 0.15, 0.22, 0.13
Tasks: 115 total,   2 running, 113 sleeping,   0 stopped,   0 zombie
Cpu(s):  2.6%us,  0.7%sy,  0.0%ni, 96.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   3111428k total,   468220k used,  2643208k free,    19200k buffers
Swap:  1020116k total,        0k used,  1020116k free,   257112k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 4588 root      20   0  110m  20m 7704 S  1.3  0.7   0:08.44 Xorg
 5480 patricia  20   0 34856  15m 9.8m R  1.3  0.5   0:02.04 gnome-terminal
 4616 avahi     20   0  3076 1588 1228 S  0.7  0.1   0:00.16 avahi-daemon
 5852 patricia  20   0  2448 1164  904 R  0.7  0.0   0:00.44 top
    1 root      20   0  3084 1880  556 S  0.0  0.1   0:01.40 init
    2 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      15  -5     0    0    0 S  0.0  0.0   0:00.04 ksoftirqd/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    6 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 events/0
    7 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 khelper
   46 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kintegrityd/0
   48 root      15  -5     0    0    0 S  0.0  0.0   0:00.04 kblockd/0
   50 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
   51 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
  123 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 cqueue
  127 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kseriod
```

3.7. Orden `mpstat`

Muestra estadísticas del procesador (o procesadores) del sistema junto con la media global de todos los datos mostrados (tienes que tener instalado el paquete **sysstat** que se encuentra en el directorio **/fenix/depar/lsi/so/paquetes**, tal y como se vió en la sesión anterior). Permite el uso de parámetros para definir la cantidad de tiempo entre cada toma de datos y el número de informes que se desean (*mpstat time reports*). La información de la cabecera hace referencia a:

5. **CPU** : número del procesador
6. **%user** : porcentaje de uso de la CPU con tareas a nivel de usuario
7. **%nice** : porcentaje de uso de la CPU con tareas a nivel de usuario con prioridad “nice” (> 0)
8. **%sys** : porcentaje de uso de la CPU para tareas del sistema (no incluye el tratamiento de interrupciones) (modo núcleo)
9. **%iowait** : porcentaje de tiempo que la CPU estaba “desocupada” mientras que el sistema tenía pendientes peticiones de E/S
10. **%irq** : porcentaje de tiempo que la CPU gasta con interrupciones hardware
11. **%soft** : porcentaje de tiempo que la CPU gasta con interrupciones software (la mayoría son llamadas al sistema)
12. **%idle** : porcentaje de tiempo que la CPU estaba “desocupada” y el sistema no tiene peticiones de disco pendientes
13. **intr/s** : número de interrupciones por segundo recibidas por el procesador

La sintaxis de la orden es:

```
mpstat [intervalo] [número]
```

donde, intervalo indica cada cuántos segundos debe mostrar los datos, y número, cuántos muestreos se solicitan.

A continuación os mostramos una pantalla con el resultado de la ejecución de mpstat:

```
Linux 2.6.31-15-generic-pae (-desktop)  12/03/2011      _i686_  (4 CPU)
12:55:58 PM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest   %idle
12:55:59 PM    all    1.25      0.00       1.75    0.00      0.00    0.00    0.00      0.00       96.99
12:56:00 PM    all    0.25      0.00       0.25    0.00      0.00    0.00    0.00      0.00       99.51
12:56:01 PM    all    0.50      0.00       0.00    0.00      0.00    0.00    0.00      0.00       99.50
12:56:02 PM    all    0.00      0.00       0.50    0.00      0.00    0.00    0.00      0.00       99.50
12:56:03 PM    all    0.49      0.00       0.49    0.00      0.00    0.00    0.00      0.00       99.02
```

Actividad 3.4

Responde a las siguientes cuestiones y especifica, para cada una, la orden que has utilizado:

a) ¿Qué porcentaje de tiempo de CPU se ha usado para atender interrupciones hardware?

b) ¿Y qué porcentaje en tratar interrupciones software?

c) ¿Cuánto espacio de *swap* está libre y cuánto ocupado?

4. Control y gestión de memoria

4.1. Orden `free`

Aunque algunas de las órdenes anteriores (como **`top`**) muestran información acerca del uso de memoria del sistema, en caso de que queramos centrar la monitorización únicamente en el uso de memoria, sin distraer la atención con otra información adicional, podemos utilizar órdenes específicas para esta labor. Esta subsección describe una orden que es capaz de llevar a cabo esta tarea de forma eficiente.

La orden **`free`** consume menos recursos (CPU y memoria) que por ejemplo la orden **`top`**. Así pues, **`free`** es una orden muy ligera (*lightweight*) que se utiliza para visualizar el uso actual de memoria. Dicha orden informa del consumo de:

- Memoria real o principal (RAM) instalada en la computadora.
- Memoria de espacio de intercambio (*swap*) – esto es, del uso del espacio de intercambio en la partición de almacenamiento en disco correspondiente. Los sistemas operativos utilizan espacio de intercambio cuando necesitan alojar parte del espacio virtual de memoria de un proceso.

Mediante la ejecución de la orden se obtiene una instantánea del uso de memoria tal como se muestra a continuación[5]:

```
#> free
             total       used       free     shared    buffers     cached
Mem:        768408     642280     126128          0      24516     470184
-/+ buffers/cache:     147580     620828
Swap:      1507320          0    1507320
```

Como se puede observar, con respecto a los dos tipos de memoria comentados, se muestra la memoria total, usada y libre. Pero además se muestra la cantidad de memoria correspondiente a los búferes de disco utilizados por el kernel, y la cantidad de memoria que ha sido llevada a caché de disco. La línea **`-/+`** refleja el total de memoria principal dividido en base a la memoria usada por la cada una de estas dos últimas cantidades.

Actividad 4.1

Explora las opciones de las que consta la orden **`free`** prestando especial atención a las diferentes unidades de medida según las que puede informar acerca de memoria. Además, compare los resultados con los obtenidos usando la orden **`watch`.**

4.2. Orden `vmstat`

Otra orden interesante que se puede encontrar en la mayoría de los sistemas operativos UNIX es **`vmstat`**. Sirve para supervisar el sistema mostrando información de memoria pero también acerca de procesos, E/S y CPU. Normalmente su primera salida proporciona una media desde el último arranque del sistema operativo y se pueden obtener informes sobre el uso durante el

5 Los valores de memoria bajo buffers y chached pueden considerarse como memoria libre.

periodo de tiempo actual, indicando el periodo de tiempo en segundos y número de iteraciones deseadas. Por ejemplo, para ejecutar 20 iteraciones de la orden mostrando la información cada 2 segundos, los informes de memoria y procesos son instantáneas en este caso, ejecutaremos la siguiente orden **vmstat** la cual producirá la correspondiente salida:

```
#> vmstat 2 20
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0   1500 193440  41228 355700    0    0   125    36  268  217  5 20 73  2  0
 3  1   1500 193268  41228 355700    0    0     2     0  243  277  2 18 80  0  0
 0  1   1500 193168  41228 355824    0    0    68     0  808  813  9 70  0 21  0
 1  0   1500 186076  41236 358424    0    0  1308    10  931  852 10 79  0 11  0
 1  0   1500 178372  41236 358568    0    0    22     0 1007  573 33 65  0  2  0
 2  0   1500 174116  41244 358624    0    0    14    76  978  921 25 74  0  1  0
 0  0   1500 173496  41244 358608    0    0     2     2  755  697 17 55 29  0  0
 2  0   1500 172876  41284 358604    0    0    16    42  590  533 13 36 51  1  0
 0  0   1500 172000  41284 358604    0    0     0     0  474  385 11 28 61  0  0
 0  0   1500 172000  41292 358608    0    0     0    20  186  185  3 10 87  0  0
 2  0   1500 171868  41376 358828    0    0   158     0  388  404  7 27 65  0  0
 3  0   1500 167140  41380 359264    0    0   214     0 1017  840 25 73  0  2  0
 2  0   1500 165768  41668 359720    0    0   326     0  996  867 15 81  0  5  0
 2  0   1500 164644  41676 360556    0    0   366    40  850  617 14 63 15  9  0
 2  0   1500 161560  41680 361724    0    0   588     0 1083  770 17 83  0  0  0
 3  0   1500 159948  41680 362540    0    0   266     0 1042 1098 19 81  0  0  0
 1  0   1500 159204  41688 362944    0    0    68    20  979 1050 13 83  0  4  0
 1  0   1500 158080  41688 362956    0    0     0     0  897  913 16 72 12  0  0
 1  0   1500 155104  41688 362956    0    0     0    50 1029  663 32 69  0  0  0
 1  0   1500 153972  41912 362936    0    0     0   632  968 1000 15 72  0 13  0
```

Ciertas columnas muestran porcentaje de tiempo de CPU tratando con :

5. Programas o peticiones de usuario (columna **us**).
6. Tareas del sistema (columna **sy**), tal como esperando E/S, actualizando estadísticas del sistema, gestionando prioridades, etc.
7. No haciendo nada en absoluto (columna **id**).

En la columna **us** podríamos ver un alto porcentaje, con la columna **sy** absorbiendo el resto del tiempo excepto aquel que el sistema no está ocupado según indica la columna **id**. Uno podría pensar que el sistema está sobrecargado realizando tareas, pero esencialmente no está haciendo nada más que servir a las tareas de usuario y del sistema relacionadas.

Las E/S de disco suelen ser un cuello de botella para la mayoría de los sistemas, así cualquier aspecto que impacte en el acceso al disco puede provocar importantes diferencias en el sistema. Por ejemplo, una alta demanda de memoria puede provocar en el sistema utilice intensivamente el espacio de intercambio transfiriendo continuamente información de memoria a disco y viceversa. Esta situación podría estar indicando que se están comenzando a ejecutar en un momento dado un número importante de procesos, ello podría quedar reflejado en la salida iterativa de la orden **vmstat** de la siguiente manera:

- La columna **r** mostraría cuántos procesos están actualmente en cola de ejecución.

- La columna **wa** podría indicar que no hay procesos en espacio de intercambio, esto sería un buen indicador en general deseable en cualquier circunstancia.
- La columna **so** indicaría que se se está incrementando el uso del espacio de intercambio, es decir, se lleva información de memoria principal a espacio de intercambio.
- La columna **free** indicaría que la memoria principal libre se puede estar agotando.

Después de que dichos procesos se hayan lanzado el sistema tenderá a estabilizarse liberando algo de memoria principal llevando más información a dispositivo de intercambio. El problema real llegaría a ser más evidente cuando se empieza a tocar techo en la memoria de intercambio y la actividad reflejada con respecto a la transferencia de información entre memoria principal y espacio de intercambio (columnas **si** y **so**) no cesa.

Actividad 4.2

Intente reproducir el escenario justo descrito anteriormente supervisando la actividad del sistema mediante la ejecución periódica de **vmstat** tal cual se ha descrito, y proporcione como muestra la salida almacenada en un archivo de texto.

5. Control y gestión de dispositivos de E/S

En el apartado de introducción a la sesión 2 de prácticas se enumeraron las distintas clases de archivos que soporta un sistema tipo UNIX: archivos regulares, directorios, archivos de tipo enlace, archivos especiales de dispositivo y, archivos para comunicaciones FIFO y Socket.

Obviamente el SO necesita mantener una estructura de datos por cada archivo que contenga la información que le es necesario mantener para poder trabajar con él. A esta información se le conoce comúnmente en el mundo de los SOs como ***metadatos de archivo*** (también como ***atributos de archivo***). Existe un metadato de archivo fundamental que es el ***nombre de archivo***. Este atributo se debe ubicar obligatoriamente en un sitio distinto al del resto de metadatos. ¿Qué motiva que esto deba hacerse así?

Pensad que este atributo lo proporciona el usuario para identificar la información que contiene el archivo (***datos de archivo***), y que el usuario utiliza la estructura jerárquica de directorios para ubicarlo en un directorio concreto. Por tanto, este metadato "especial" se ubica siempre en una ***entrada de directorio***. Los *archivos de tipo directorio* están compuestos básicamente por entradas de directorio, que deben contener obligatoriamente el nombre de archivo – si no, ¡como iba a funcionar el **ls**!

En UNIX se almacena una ***referencia a los metadatos*** en la entrada de directorio y los propios metadatos en un sitio del disco que el SA reserva para este fin: un ***inodo*** (***inode***).

Con respecto a la facilidad para compartir los archivos, los SA tipo UNIX permiten establecer enlaces a archivos. Más adelante en esta sección veremos los tipos de enlaces que soportan y la forma de crearlos mediante la orden **ln**.

También abordaremos en esta sección los tipos de archivos especiales de dispositivo que soportan los SA tipo UNIX, ***archivos especiales para dispositivos de bloques y para dispositivos de caracteres***, los cuales proporcionan al usuario una interfaz abstracta para el

trabajo con dispositivos de E/S y almacenamiento secundario. Para crear este tipo de archivos se utilizará la orden **`mknod`**.

Obviamente, tanto los datos de los archivos como los metadatos de archivo deben almacenarse en los dispositivos de almacenamiento persistente. Para la gestión de este almacenamiento, los sistemas de archivos tipo UNIX mantienen, entre sus metadatos de SA, información relativa a bloques de disco libres/ocupados e inodos libres/asignados. Para acceder a esta información utilizaremos las órdenes **`df`** y **`du`**.

5.1. Consulta de información de archivos.

Como viste en las prácticas de la asignatura Fundamentos del Software, la orden **`ls`** muestra los nombres de los archivos incluidos en un directorio. Por omisión, si no se especifica el nombre de un directorio, la orden **`ls`** actúa sobre el directorio de trabajo (*working directory*). Además, si se utiliza la opción **`-l`**, **`ls`** muestra determinada información relativa a los metadatos de los archivos incluidos en el directorio especificado.

En esta parte de la sesión vamos a ampliar el conocimiento de las opciones que permite esta orden porque, tras haber comprendido el concepto de metadatos de archivo, se le puede sacar más partido desde el punto de vista de un usuario un poco más avanzado y, por supuesto desde el punto de vista de administración de sistemas. A continuación se muestra una tabla con algunas órdenes que permiten conocer información adicional al uso básico de **`ls`**.

Tabla 1. Órdenes útiles para la consulta de metadatos de archivo.

`ls -l`	Print a *long listing format* of file metadata for each file of the specified directory(-ies).
`ls -n`	Print a *long listing format* but list numeric user and group IDs.
`ls -la`	Like **`ls -l`** but do not ignore directory entries starting with "." character (*Hidden entries.*)
`ls -li`	Print a long listing format adding the inode number field.
`ls -lh`	Print a *long listing format* of file metadata but size fields are printed in Kbytes, Mbytes o Gbytes (*human readable format*).

Como puedes observar en la siguiente salida por pantalla, el "*long listing format*" de **`ls`** es muy útil para conocer información de metadatos de archivo, como el tipo de archivo y permisos. Los caracteres asociados al tipo de archivo son:

- **`-`**, archivo regular.
- **`d`**, directorio.
- **`l`**, enlace simbólico. (Ya veremos más adelante que hay dos tipos de enlace en UNIX pero solo un tipo de archivo enlace).
- **`b`**, archivo especial de dispositivo de bloques.
- **`c`**, archivo especial de dispositivo de caracteres.

- **p**, archivo FIFO para comunicaciones entre procesos.

```
~/DIR$> ls -lai
total 20
6163598 drwxr-xr-x 3 aleon aleon 4096 2011-10-23 11:56 .
6293041 drwxr-xr-x 3 aleon aleon 4096 2011-10-23 11:52 ..
6162979 -rw-r--r-- 2 aleon aleon   52 2011-10-23 11:18 archivo.txt
6163010 brw-r--r-- 1 root  root  7, 0 2011-10-23 11:27 blockDeviceSpecialFile
6163027 crw-r--r-- 1 root  root  7, 0 2011-10-23 11:32 characterDeviceSpecialFile
6163009 drwxr-xr-x 2 aleon aleon 4096 2011-10-23 11:19 D1
6163029 prw-rw---- 1 root  root     0 2011-10-23 11:35 FIFOfile
6162979 -rw-r--r-- 2 aleon aleon   52 2011-10-23 11:18 hardLink
6163022 brw-r--r-- 1 root  root  7, 0 2011-10-23 11:30 loop0
6162996 lrwxrwxrwx 1 aleon aleon   11 2011-10-23 11:19 softLink -> archivo.txt
```

Actividad 5.1

Anota al menos dos nombres de archivo de dispositivo de bloques y dos nombres de dispositivo de caracteres de tu sistema UML. Anota los nombres de los archivos ocultos de tu directorio de inicio como usuario root que tienen relación con el intérprete de órdenes que tienes asignado por defecto. Ahora efectúa la misma tarea pero en una consola de terminal del sistema **Ubuntu** que arrancas inicialmente en el laboratorio de prácticas. ¿Qué diferencias encuentras entre los nombres de los archivos?

Ordenar listados de metadatos de archivo (*Sorting listings*)

Aunque podríamos utilizar el conocimiento que poseéis de la asignatura Fundamentos del Software acerca de la orden **`sort`**, para establecer una ordenación de los listados de información que proporciona la orden **`ls`**, ayudándonos de la orden **`cut`**, vamos a aprovechar las capacidades de ordenación por campo que posee intrínsecamente la orden **`ls`**. La siguiente tabla muestra algunas de las opciones de ordenación básicas que proporciona.

Tabla 2. Opciones básicas para ordenación que proporciona ls. Especialmente indicadas para aplicarlas en "long listing format".

`ls -X`	Sort alphabetically by directory entry extension.
`ls -t`	Sort by modification time.
`ls -u`	Sort by access time.
`ls -c`	Sort by **`ctime`** (time of last modification of file status information, e.d. file metadata.)

Actividad 5.2 Ordenar un listado usando los campos de tiempos.

Conocemos la sintaxis de la orden para obtener un listado en formato largo (*"long listing format"*). Manteniendo la opción de listado largo añade las opciones que sean necesarias para obtener un listado largo con las siguientes especificaciones:

- Que contenga el campo "*access time*" de los archivos del directorio especificado y que esté ordenado por dicho campo.
- Que contenga el campo "*ctime*" de los archivos del directorio especificado y que esté ordenado por dicho campo.

Para más información sobre la orden **`ls`** consultar el manual *Texinfo*. Utiliza la orden:

```
$> info coreutils 'ls invocation'
```

5.2 Consulta de metadatos del SA

Para poder asignar espacio en disco, tanto para datos de archivo como para metadatos de archivo, los sistemas de archivos tipo UNIX/Linux mantienen, entre otros metadatos de SA, información relativa a *bloques de disco libres/ocupados* e *inodos libres/asignados*. Esta información es muy relevante para el administrador del sistema ya que la falta de cualquiera de estos dos recursos (*bloques de disco* e *inodos*) puede provocar degradación en la utilización del SA por parte de los usuarios, e incluso, en un caso límite, los usuarios podrían dejar de poder utilizar el SA.

La orden **`df`** permite visualizar, para cada SA montado, información sobre su capacidad de almacenamiento total, el espacio usado para almacenamiento y el espacio libre restante, y el punto de montaje en la jerarquía de directorios para cada SA. La siguiente salida por pantalla muestra esta información para un sistema de ejemplo.

```
~/DIR/D1$> df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda5             50639860   7068448  40999008  15% /
none                   2024264       344   2023920   1% /dev
none                   2028796       728   2028068   1% /dev/shm
none                   2028796        92   2028704   1% /var/run
none                   2028796         0   2028796   0% /var/lock
none                   2028796         0   2028796   0% /lib/init/rw
/dev/sda6            173364612   6419104 158139080   4% /home
```

Utilizando la orden **`df -i`** podemos visualizar la información sobre los inodos de cada SA montado. A continuación se muestra una salida por pantalla para esta orden sobre un sistema de ejemplo.

```
~/DIR/D1$> df -i
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/sda5            3219456  401028 2818428   13% /
none                  506066     860  505206    1% /dev
none                  507199       7  507192    1% /dev/shm
none                  507199      55  507144    1% /var/run
none                  507199       1  507198    1% /var/lock
none                  507199       1  507198    1% /lib/init/rw
/dev/sda6            11010048   21853 10988195    1% /home
```

Para poder ver el espacio en disco que gasta un directorio de la jerarquía de directorios, y todo el subárbol de la jerarquía que comienza en él, se utililiza la orden **du**. Esta orden proporciona el número de bloques de disco asignados a todos los archivos (incluidos directorios) que "cuelgan" del directorio especificado. Hay que tener en cuenta dos observaciones:

- La última línea de la salida por pantalla muestra la cantidad total de bloques de disco utilizados por el subárbol.
- **du** contabiliza el número de bloques de disco asignados estén o no completamente ocupados. Este concepto de espacio libre dentro de un bloque de disco (*fragmentación interna*) se explicará en su momento en teoría.

El siguiente listado por pantalla muestra la salida de la orden **du** para un directorio especificado sobre un espacio de nombres de ejemplo.

```
$> du SOII/
5184  SOII/transparencias/TEMA2
13568 SOII/transparencias
252   SOII/alumnos2009-10
15408 SOII/biblio/Solaris
16    SOII/biblio/Seguridad y Proteccion
92    SOII/biblio/windows/Windows Server 2003
284   SOII/biblio/windows/WindowsNT
612   SOII/biblio/windows
9500  SOII/biblio/OS2Warp_IBM
42676 SOII/biblio
11696 SOII/doc/teoria/transparencias
3608  SOII/doc/teoria/apuntes
23712 SOII/doc/teoria
172   SOII/doc/images
35124 SOII/doc
80296 SOII/
```

Actividad 5.3

Comprueba cuántos bloques de datos está usando la partición raíz del sistema UML del laboratorio. Ahora obtén la misma información pero expresada en "*human readable format*": Megabytes o Gigabytes. Para ello consulta en detalle el manual en línea.

Actividad 5.4

¿Cuántos inodos se están usando en la partición raíz? ¿Cuántos nuevos archivos se podrían crear en esta partición?

Actividad 5.5

¿Cuál es el tamaño del directorio /etc? ¿Y el del directorio /var? Compara estos tamaños con los de los directorios /bin, /usr y /lib. Anota brevemente tus conclusiones.

Actividad 5.6

Obtén el número de bloques de tamaño 4 KB que utiliza la rama de la estructura jerárquica de directorios que comienza en el directorio /etc. En otras palabras, los bloques de tamaño 4 KB del *subárbol* cuya raíz es /etc. ¿Cuál es el tamaño de bloque, por omisión, utilizado en el SA?

5.3. Creación de enlaces a archivos

El objetivo de los enlaces a archivos es disponer de más de un nombre para los archivos en nuestro espacio de nombres de archivo soportado por la estructura jerárquica de directorios. Bajo esta óptica, los enlaces a archivos pueden considerarse como referencias a otros archivos, bien a su nombre, ***enlaces simbólicos***, bien a sus metadatos, ***enlaces duros***[6].

Desde el punto de vista de los enlaces duros, todos los nombres de archivo forman un enlace duro sobre el inodo asociado simplemente por existir en el espacio de nombres. En otras palabras, al crear un archivo, sea del tipo que sea, se establece el primer enlace duro sobre su inodo asociado, cuando el SO crea la entrada en el directorio en donde se va a ubicar.

De hecho, podemos comprobar utilizando la orden **`ls -la`** que todos los directorios tienen dos enlaces duros que crea el SO automáticamente para mantener la jerarquía de directorios mediante el establecimiento de la *relación jerárquica (directorio_padre,directorio_hijo)*. Estos enlaces duros se identifican con las entradas cuyos nombres de archivo son: "**`.`**" para el enlace duro al directorio en el que se encuentra la entrada asociada y, "**`..`**" para el enlace duro al directorio padre del directorio en el que se encuentra la entrada correspondiente. La siguiente salida por pantalla muestra un ejemplo de un directorio `~/DIR` y el contenido mínimo de sus entradas de directorio: **`(inode_number, filename)`**.

```
~/DIR$> ls -ai1
6163598 .
6293041 ..
6162979 archivo.txt
6163010 blockDeviceSpecialFile
```

6 Veremos que un archivo de tipo enlace simbólico (*soft link*) si actúa como referencia a otro nombre de archivo, mientras que un enlace duro (*hard link*) realmente es un nombre distinto para los metadatos (inodo) de un archivo ya existente.

```
6163027 characterDeviceSpecialFile
6163009 D1
6163029 FIFOfile
6162979 hardLink
6163022 loop0
6162996 softLink
```

Para crear enlaces duros o enlaces simbólicos sobre un archivo creado previamente se utiliza la orden **`ln`**. Como argumentos básico debemos proporcionarle el nombre del archivo a enlazar (*link target*) y el nuevo nombre de archivo (*link name*). La siguiente salida por pantalla presenta ejemplos de enlaces duros, tanto el que mantiene la jerarquía de directorios (ver la información del directorio **`D1/`**), como los de los archivos **`archivo.txt`** y **`target_hardLink2.txt`**, **`hardLink`** y **`hardLink2`** respectivamente. Así mismo se muestra el enlace simbólico, **`softLink`**, al archivo **`archivo.txt`**.

Los números que aparecen en la columna inmediatamente anterior al **`username`** propietario del archivo (**`owner`**) son valores del campo ***contador de enlaces***. Este contador mantiene el número de enlaces duros a archivos con el objetivo de poder liberar el inodo cuando todos los nombres de archivo que utilizan dicho inodo hayan sido eliminados de la estructura de directorios y nunca antes, pues si se liberase el inodo con un contador de enlaces mayor que 0, se podría intentar acceder al archivo y se produciría una inconsistencia entre el espacio de nombres y los metadatos de archivo.

```
~/DIR$> ls -lai
total 28
6163598 drwxr-xr-x 3 aleon aleon 4096 2011-10-23 19:05 .
6293041 drwxr-xr-x 3 aleon aleon 4096 2011-10-23 13:20 ..
6162979 -rw-r--r-- 2 aleon aleon   52 2011-10-23 11:18 archivo.txt
6163010 brw-r--r-- 1 root  root  7, 0 2011-10-23 11:27 blockDeviceSpecialFile
6163027 crw-r--r-- 1 root  root  7, 0 2011-10-23 11:32 characterDeviceSpecialFile
6163009 drwxr-xr-x 2 aleon aleon 4096 2011-10-23 11:19 D1
6163029 prw-rw---- 1 root  root     0 2011-10-23 11:35 FIFOfile
6162979 -rw-r--r-- 2 aleon aleon   52 2011-10-23 11:18 hardLink
6163034 -rw-r--r-- 2 aleon aleon   80 2011-10-23 19:04 hardLink2
6163022 brw-r--r-- 1 root  root  7, 0 2011-10-23 11:30 loop0
6162996 lrwxrwxrwx 1 aleon aleon   11 2011-10-23 11:19 softLink -> archivo.txt
6163034 -rw-r--r-- 2 aleon aleon   80 2011-10-23 19:04 target_hardLink2.txt

~/DIR$ cd D1/

~/DIR/D1$ ls -lai
total 8
6163009 drwxr-xr-x 2 aleon aleon 4096 2011-10-23 11:19 .
6163598 drwxr-xr-x 3 aleon aleon 4096 2011-10-23 19:05 ..
```

Actividad 5.7 Creación de enlaces con la orden `ln`

Construye los mismos enlaces, duros y simbólicos, que muestra la salida por pantalla anterior. Para ello crea los archivos **`archivo.txt`** y **`target_hardLink2.txt`** y, utilizando el manual en línea para **`ln`**, construye los enlaces **`softLink`**, **`hardLink`** y **`hardLink2`**. Anota las órdenes que has utilizado.

¿Por qué el contador de enlaces del archivo **`archivo.txt`** vale 2 si sobre el existen un enlace duro **`hardLink`** y un enlace simbólico **`softLink`**?

Actividad 5.8 Trabajo con enlaces

Proporciona las opciones necesarias de la orden **`ls`** para obtener la información de metadatos de los archivos de un directorio concreto en los dos casos siguientes:

- En el caso de que haya archivos de tipo enlace simbólico, la orden debe mostrar la información del archivo al que enlaza cada enlace simbólico y no la del propio archivo de tipo enlace simbólico.
- En el caso de enlaces simbólicos debe mostrar la información del enlace en sí, no del archivo al cual enlaza. En el caso de directorios no debe mostrar su contenido sino los metadatos del directorio.

5.4. Archivos especiales de dispositivo

Los dispositivos de nuestro sistema se representan en un SO tipo UNIX mediante archivos especiales de dispositivo. Existen dos tipos principales de archivos especiales de dispositivo: de bloques y de caracteres. Los archivos especiales de bloque representan a dispositivos de bloques, que normalmente coinciden con los dispositivos de almacenamiento persistente, los *ramdisks* y los dispositivos *loop*. Los archivos especiales de caracteres representan a dispositivos de caracteres del tipo puertos serie, paralelo y USB, consola virtual (**`console`**), **`audio`**, los dispositivos de terminal (**`tty*`**), y muchos más.

La siguiente salida por pantalla muestra un listado (recortado) del directorio **`/dev`** de una distribución común de Linux.

```
$> ls -l /dev
total 0
crw-rw----+ 1 root audio    14,  12 2011-10-23 08:10 adsp
crw-rw----+ 1 root audio    14,   4 2011-10-23 08:10 audio
drwxr-xr-x  2 root root         680 2011-10-23 10:10 block
drwxr-xr-x  2 root root          80 2011-10-23 10:10 bsg
drwxr-xr-x  3 root root          60 2011-10-23 10:10 bus
lrwxrwxrwx  1 root root           3 2011-10-23 08:10 cdrom -> sr0
lrwxrwxrwx  1 root root           3 2011-10-23 08:10 cdrw -> sr0
drwxr-xr-x  2 root root        3500 2011-10-23 08:10 char
crw-------  1 root root      5,   1 2011-10-23 08:10 console
…
```

```
lrwxrwxrwx  1 root root           3 2011-10-23 08:10 dvd -> sr0
lrwxrwxrwx  1 root root           3 2011-10-23 08:10 dvdrw -> sr0
…
brw-rw----  1 root disk      7,   0 2011-10-23 08:10 loop0
brw-rw----  1 root disk      7,   1 2011-10-23 08:10 loop1
…
crw-rw-rw-  1 root root      1,   3 2011-10-23 08:10 null
…
brw-rw----  1 root disk      1,   0 2011-10-23 08:10 ram0
brw-rw----  1 root disk      1,   1 2011-10-23 08:10 ram1
…
brw-rw----  1 root disk      8,   0 2011-10-23 08:10 sda
brw-rw----  1 root disk      8,   1 2011-10-23 08:10 sda1
brw-rw----  1 root disk      8,   2 2011-10-23 08:10 sda2
…
crw-rw----  1 root disk     21,   0 2014-04-25 11:39 sg0
…
crw-rw-rw-  1 root tty       5,   0 2011-10-23 11:27 tty
crw--w----  1 root root      4,   0 2011-10-23 08:10 tty0
crw-------  1 root root      4,   1 2011-10-23 08:10 tty1
…
crw-rw----  1 root root    252,   0 2011-10-23 08:10 usbmon0
crw-rw----  1 root root    252,   1 2011-10-23 08:10 usbmon1
…
crw-rw----+ 1 root video    81,   0 2011-10-23 08:10 video0
crw-rw-rw-  1 root root      1,   5 2011-10-23 08:10 zero
```

Como se puede ver en el listado superior existe un interfaz al disco como dispositivo de caracteres (en negrita) que se suele utilizar como copias "*raw*" del disco. Es decir el disco se lee como un dispositivo secuencial para hacer copias completas de la información almacenada. Esto puede ser útil para duplicar el disco o hacer copias de seguridad.

Podemos crear archivos especiales de dispositivo, tanto de bloques (buffered) como de caracteres (unbuffered) utilizando la orden **`mknod`**. Esta orden permite especificar el nombre del archivo y los números principal (***major***) y secundario (***minor***). Estos números permiten identificar a los dispositivos en el kernel, concretamente en la Tabla de Dispositivos. El número principal determina el controlador al que está conectado el dispositivo y el número secundario al dispositivo en sí. Es necesario consultar el convenio de nombres dado que la asignación de estos números va a depender de la distribución. Todos estos conceptos se explicarán en teoría con detalle. Aquí simplemente se muestra la forma de crearlos como usuario del lenguaje de órdenes, no su implementación en el núcleo del SO.

Actividad 5.9

Consulta el manual en línea para la orden **`mknod`** y crea un dispositivo de bloques y otro de caracteres. Anota las órdenes que has utilizado y la salida que proporciona un **`ls -li`** de los dos archivos de dispositivo recién creados. Puedes utilizar las salidas por pantalla mostradas en esta sección del guión para ver el aspecto que debe presentar la información de un archivo de dispositivo.