David97
www.wuolah.com/student/David97

SOModIsesion3.pdf

Sesión 3 - Módulo I - SO

2º Sistemas Operativos

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

KEEP CALM AND ESTUDIA UN POQUITO

WUOLAH + #QuédateEnCasa

Sesión 3. Monitorización del sistema

Apuntes

1 Introducción

Comprobar el estado del sistema en cuanto a parámetros tales como CPU, memoria, etc. Hay órdenes que muestran más información que otras. Son los procesos los que consumen estos recursos.

Entre el sistema de sectores y archivos el sistema maneja bloques lógicos, que al fin y al cabo es un grupo de sectores físicos.

El sistema sabe que tiene que eliminar un fichero en un SA cuando no haya ninguna entrada que apunte al archivo.

2 Control y gestión de CPU

2.1 Orden `uptime`

Hora actual, tiempo que lleva en marcha el sistema, nº de usuarios conectados, carga media del sistema (últimos 1, 5 y 15 min).

`w` muestra la misma información que `uptime`, además de los usuarios conectados y qué hacen.

Un sistema con un comportamiento normal debe mostrar una carga ≤ (1 × n.º de cores)

2.2 Orden `time`

Mide el tiempo de ejecución de un programa y muestra un resumen del uso de los recursos del sistema. Muestra:

- **real**. Tiempo total que el programa ha estado ejecutándose.
- **user**. Tiempo que se ha ejecutado en modo usuario.
- **sys**. Tiempo que se ha ejecutado en modo supervisor.

El tiempo de espera es: $t_{espera} = real - user - sys$

2.3 Órdenes `nice` *y* `renice`

Un proceso tiene una prioridad que hereda de su proceso padre. La **prioridad** por defecto de este se puede modificar con la orden `nice` **[-20,19]**. Un valor negativo aumenta la posibilidad de ejecución de un proceso y solamente puede hacerlo el usuario `root`. Los valores positivos, que desfavorecen la posibilidad de ejecución, pueden establecerlos cualquiera de los usuarios sin privilegios.

- `nice -5 <proc>` → Aumenta en 5 el valor de prioridad
- `nice --10 <proc>` → Decrementa en 10 el valor de prioridad

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

- `renice <priority> PID`
- `renice [-n] <priority> -g` PID_1, PID_2, … PID_N

2.4 Orden `pstree`

Visualiza un **árbol de procesos** en ejecución.

2.5 Orden `ps`

Muestra información sobre los procesos en ejecución usando el pseudo-sistema de archivos `/proc`

2.6 Orden `top`

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:

- Estadísticas de la orden `uptime`
- Estadísticas sobre los procesos del sistema (nº de procesos, procesos en ejecución, durmiendo, parados o zombies)
- Estado actual de la CPU (% en uso por usuarios, por el sistema, con valor nice positivo, ociosos, esperando E/S, tratando interrupciones hardware o software y en espera involuntaria por virtualización)
- Memoria (total disponible, usada, libre, cantidad usada en buffers y en caché de página)
- Espacio de swap (total disponible, libre, usada y disponible)

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.

2.7 Orden `mpstat`

Muestra estadísticas del procesador/es del sistema junto con la media global de todos los datos mostrados (paquete `sysstat` de /fenix/depar/lsi/so/paquetes). 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*). Se usa del siguiente modo: `mpstat [intervalo (seg)] [n.º reportes]`

3 Control y gestión de memoria

3.1 Orden `free`

Muestra información acerca del uso de memoria del sistema. La orden `free` es una orden muy ligera (*lightweight*) que consume menos recursos que otras órdenes. Informa sobre el consumo de memoria real o principal (RAM) y memoria de espacio de intercambio (swap).

3.2 Orden `vmstat`

Sirve para supervisar el sistema mostrando información de memoria, pero también de procesos, E/S y CPU. Su funcionamiento es similar al de mpstat.

Procs

r: n.º de procesos ejecutables (ejecutándose o esperando el tº de ejecución → cola de ejecución)

b: n.º de procesos en esperar ininterrumpida

Memory

swpd: la cantidad de memoria virtual utilizada

free: la cantidad de memoria libre

buff: la cantidad de memoria usada como búfer

cache: la cantidad de memoria usada como caché

inact: la cantidad de memoria inactiva. (opción -a)

active: la cantidad de memoria activa. (opción -a)

Swap

si: cantidad de memoria intercambiada desde el disco (/s).

so: cantidad de memoria intercambiada hacia el disco (/s).

IO

bi: bloques recibidos de un dispositivo de bloque (blocks/s).

bo: bloques enviados a un dispositivo de bloque (blocks/s).

System

in: n.º de interrupciones por segundo, incluido el reloj

cs: n.º de cambios de contexto por segundo

CPU

Estos son porcentajes del total del tiempo de CPU

us: tº dedicado a ejecutar código que no es del núcleo (*user time*, incluido *nice time*)

sy: tº dedicado a ejecutar código kernel (*system time*)

id: tº ocioso

wa: tº dedicado esperando E/S

st: tº robado de una máquina virtual

4 Control y gestión de dispositivos de E/S

UNIX soporta los siguientes tipos de archivos: archivos regulares, directorios, archivos de tipo enlace, archivos especiales de dispositivo y archivos para comunicaciones FIFO y Socket.

Los **metadatos de archivo** (o **atributos** de archivo) son una estructura de datos que contiene la información necesaria para mantener cada archivo y poder trabajar con él. El metadato de archivo fundamental es el nombre de archivo, el cual se ubica obligatoriamente en un lugar distinto al resto de metadatos ya que es lo que identifica al archivo y lo proporciona el usuario para utilizarlo en la estructura jerárquica de directorios.

4.1 Consulta de información de archivos

Con `ls -l` se pueden ver los nombres de archivos junto con información relativa a sus metadatos. Hay más opciones de ls de utilidad:

- `ls -n` → listado largo con IDs de usuarios y grupos
- `ls -la` → como `ls -l` pero sin ignorar directorios ni archivos ocultos (empiezan por "`.`")
- `ls -li` → listado largo añadiendo un campo con el número de inodo
- `ls -lh` → listado largo con metadatos con el campo de tamaño en *human readable format*

4.2 Consulta de metadatos del SA

UNIX/Linux mantiene, entre otros metadatos de SA, información relativa a bloques de disco libres/ ocupados e inodos libres/asignados.

`df` → visualizar información sobre capacidad de almacenamiento total, espacio usado y espacio libre para cada SA. Con `df -i` se muestra la misma información, peor para i-nodos.

`du` → visualizar espacio en disco que gasta un directorio en la jerarquía de directorios. Muestra el n.º de bloques de disco asignados a todos los archivos (incluidos directorios) que cuelgan de él, estén o no completamente ocupados.

4.3 Creación de enlaces a archivos

Los **enlaces** son archivos de tipo especial. Sirven para compartir un archivos entre diferentes usuarios y/o de diferentes formas. De este modo no se duplica información ni se crean inconsistencias. El objetivo de los enlaces a archivos es disponer de más de un nombre para los archivos en el espacio de nombres de archivos. Los enlaces a archivos pueden considerarse referencias a otros archivos, bien a su **nombre** (**enlaces simbólicos**), bien a sus **metadatos** (**enlaces duros**).

- **Enlaces duros/directos** (*hard links*): "casi" transparente en metadatos. Equivalente a un archivo. Otro acceso para un archivo físico ya existente. Todos los nombres de archivo forman un enlace duro sobre el inodo asociado simplemente por existir.
- **Enlaces blandos/simbólicos** (*soft links*): dirección hacia un archivo de forma indirecta a través de otro archivo. Guarda un camino.

El enlace simbólico es interesante cuando, por su contexto, se tenga que entender en cualquier sistema de archivos. Es un concepto que está por encima de los sistemas de archivos, metadatos, etc. Conveniente cuando se programa y hay que acceder a un archivo.

El enlace directo es mejor porque sólo guarda un metadato para llegar al archivo. Sin embargo sólo sirve dentro de un mismo sistema de archivos (referencia a través de un número de inodo).

Un directorio tiene, como mínimo, dos enlaces duros. Esto se debe a que tiene la entrada desde su propio directorio y la referencia de su padre.

Para crear enlaces se utiliza la orden **`ln`**. Por defecto, sin opciones, se crean enlaces duros; como argumentos básicos hay que proporcionar el nombre del archivo a enlazar (*link target*) y el nuevo nombre de archivo (*link name*). Con el argumento `-s` se crean enlaces simbólicos.

```
$ ln archivo hard_link
$ ln -s archivo soft_link
```

4.4 Archivos especiales de dispositivo

Los dispositivos del sistemas se representan en UNIX mediante archivos de tipo dispositivo. Existen dos tipos principales: de **bloques** y **caracteres**.

- **Archivos especiales de bloque.** Representan dispositivos de bloques, que normalmente coinciden con los dispositivos de almacenamiento persistente, los *ramdisks* y los dispositivos *loop*.
- **Archivos especiales de caracteres.** Representan a dispositivos de caracteres del tipo puertos serie, paralelo y USB, consola virtual (*console*), audio, terminales (*tty**), etc.

Es posible crear archivos especiales de dispositivo utilizando la orden **`mknod`**, permitiendo especificar los números principal (*major* → controlador) y secundario (*minor* → dispositivo), los cuales permiten identificar a los dispositivos en el kernel.

Ejercicios

Actividad 3.1 Consulta de estadísticas del sistema

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:

```
$ w
```

```
17:32:42 up 3 days,  5:18,  1 user,  load average: 1,56, 1,56, 1,78
USUARIO  TTY      DE        LOGIN@   IDLE   JCPU   PCPU    WHAT
david    tty7     :0         vie12   3días  1:22m  2.87s   cinnamon-session
```

a) ¿Cuánto tiempo lleva en marcha el sistema?

3 días, 5 horas y 18 min. También se puede consultar con la siguiente orden:

```
$ uptime -p
up 3 days, 5 hours, 18 minutes
```

b) ¿Cuántos usuarios hay trabajando?

1 usuario

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

1,78

Actividad 3.2 Prioridad de los procesos

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.

He realizado 3 guiones iguales, con la diferencia que cada uno imprimirá 1, 2 ó 3, según cual sea. También otro guion que lanza los tres procesos secuencialmente en segundo plano.

```
#!/bin/bash

var=0

while [[ $var -lt $1 ]]; do
    op=$(( (130*175+30)/71 ))
    let var=var+1
done

echo "Fin 1 con valor $var"
```

```
#!/bin/bash

time ./while_1.sh $1 &
time ./while_2.sh $1 &
time ./while_3.sh $1 &
```

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?

```
# ./lanzar.sh 30000000
```

```
#  renice -20 10173 ; renice 19 10174
```

```
PID USUARIO      PR  NI    VIRT    RES    SHR S  %CPU %MEM     HORA+ ORDEN
10174 root       39  19   15404   1204   1060 R 100,0  0,0   0:33.18 while_3.sh
10171 root       20   0   15404   1252   1108 R  99,7  0,0   0:33.12 while_1.sh
10173 root        0 -20   15404   1240   1100 R  99,3  0,0   0:33.16 while_2.sh
```

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

Fin 1 con valor 30000000	Fin 2 con valor 30000000	Fin 3 con valor 30000000
real 6m35,109s user6m28,562s sys 0m1,027s	real 6m38,891s user6m35,486s sys 0m0,955s	real 6m43,222s user6m34,237s sys 0m0,864s

Actividad 3.3 Jerarquía e información de procesos

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.

```
$ pstree -A
systemd-+-ModemManager---2*[{ModemManager}]
        |               `-2*[{NetworkManager}]
        |-accounts-daemon---2*[{accounts-daemon}]
        |-2*[agetty]
        |-apache2---5*[apache2]
        |-colord---2*[{colord}]
        |-containerd---16*[{containerd}]
        |-csd-printer---2*[{csd-printer}]
        |-cups-browsed---2*[{cups-browsed}]
        |-2*[dbus-daemon]
        |-dockerd---27*[{dockerd}]
        |-firefox-+-2*[Web Content---27*[{Web Content}]]
  .  .  .

$ ps -Al
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY      TIME     CMD
. . .
4 S   101  1096     1  0  80   0 - 17833 -      ?        00:00:16 systemd-resolve
4 S 62583  1101     1  0  80   0 - 36528 -      ?        00:00:00 systemd-timesyn
4 S     0  1168     1  0  80   0 -  9154 -      ?        00:00:00 bluetoothd
4 S     0  1177     1  0  80   0 - 108604 -     ?        00:00:00 ModemManager
4 S     0  1186     1  0  80   0 -  1138 -      ?        00:00:15 acpid
4 S   103  1189     1  0  80   0 - 13084 -      ?        00:00:38 dbus-daemon
4 S     0  1238     1  0  80   0 - 258095 -     ?        00:00:35 NetworkManager
4 S     0  1240     1  0  80   0 - 11457 -      ?        00:00:03 wpa_supplicant
4 S   114  1255     1  0  80   0 - 11816 -      ?        00:00:00 avahi-daemon
. . .
4 S     0  1830     1  0  80   0 - 337808 -     ?        00:00:16 containerd
4 S   115  1836     1  0  80   0 - 80191 -      ?        00:00:00 colord
4 S     0  1891     1  0  80   0 - 91065 -      ?        00:00:00 lightdm
4 S     0  1949  1891  1  80   0 - 167127 -     tty7     01:27:38 Xorg
```

```
4 S     0  1974     1  0  80   0 -  4675 -      tty1     00:00:00 agetty
4 S     0  2192     1  0  80   0 - 558008 -     ?        00:00:55 dockerd
1 S   121  2219     1  0  80   0 - 370680 -     ?        00:01:19 mysqld
1 S   106  2223     1  0  80   0 - 14235 -      ?        00:00:03 kerneloops
1 S   106  2232     1  0  80   0 - 14235 -      ?        00:00:03 kerneloops
5 S     0  2265     1  0  80   0 - 98475 -      ?        00:00:04 apache2
5 S     0  2409     1  0  80   0 - 136884 -     ?        00:00:07 vpnagentd
```

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

Significa que no tiene un terminal asociado. Los procesos sin terminal asociados son demonios (*daemons*), que son procesos no interactivos ejecutados en segundo plano y que no son controlados por el usuario. Esto ocurre porque al cerrar una sesión se terminan todos los procesos relacionados con un TTY, lo cual no es deseable para los procesos de todo el sistema.

Actividad 3.4 Estadísticas de recursos del sistema

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?**
   `mpstat` ← columna `%irq`

b) **¿Y qué porcentaje en tratar interrupciones software?**
   `mpstat` ← columna `%soft`

c) **¿Cuánto espacio de swap está libre y cuánto ocupado?**
   `top` ← apartados `libre` y `usado`

Actividad 3.5 Utilización de las órdenes `free` y `watch`

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`.

```
$ watch -d -n 0.5 free -m
```

`free -m` → muestra el uso de memoria en mebibytes (MiB).

`watch -d -n 0.5` → ejecuta el programa cada medio segundo resaltando las diferencias entre cada salida.

Actividad 3.6 Utilización de `vmstat`

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.

```
$ vmstat 1 5 > vmstat.txt
```

David Carrasco Chicharro

```
procs -----------memoria----------   ---swap-- -----io---- -sistema-- ------cpu-----
 r  b   swpd  libre búfer caché        si   so    bi    bo   in   cs us sy id wa st
 3  0 171916 1115344 3201388 1662868    1    4   188   192   91  423 23  4 71  2  0
 0  0 171916 1104556 3201388 1673536    0    0     0     0  485 1254  3  1 97  0  0
 0  0 171916 1112120 3201388 1665892    0    0     0     0  400  810  1  0 99  0  0
 1  0 171916 1112120 3201388 1665892    0    0     0     0  353  737  0  0 99  0  0
 0  0 171916 1115624 3201388 1662436    0    0     0     0  373  778  1  0 99  0  0
 . . .
```

Actividad 3.7 Consulta de metadatos de archivo

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?

```
# ls -lai /dev
14175 brw-r--r--  1 root root 7, 0 Oct 11 13:09 loop0
   27 crw-rw-rw-  1 root root 5, 0 Nov  3  2010 tty
   67 crw--w----  1 root tty  4, 0 Oct 11 13:12 tty0

# ls -lai ~
   58 -rw-------  1 root root        53 Sep 13  2011 .bash_history
 3958 -rw-r--r--  1 root root        18 Mar 30  2009 .bash_logout
 3959 -rw-r--r--  1 root root       176 Mar 30  2009 .bash_profile
 3960 -rw-r--r--  1 root root       176 Sep 22  2004 .bashrc
 3961 -rw-r--r--  1 root root       100 Sep 22  2004 .cshrc
 3962 -rw-r--r--  1 root root       129 Dec  3  2004 .tcshrc

$ ls -lai /dev
127 brw-rw----   1 root disk        7,   0 oct 11 10:01 loop0
128 brw-rw----   1 root disk        7,   1 oct 11 17:40 loop1
 14 crw-rw-rw-   1 root tty         5,   0 oct 11 10:01 tty
 16 crw--w----   1 root tty         4,   0 oct 11 10:01 tty0
 21 crw--w----   1 root tty         4,   1 oct 11 10:01 tty1

$ ls -lai ~
11797955 -rw-------  1 david david    39412 oct 11 18:52  .bash_history
11800863 -rw-r--r--  1 david david      220 sep 19  2018  .bash_logout
11800862 -rw-r--r--  1 david david     4165 mar 18  2019  .bashrc
11796873 -rw-r--r--  1 david david     3771 mar 18  2019  .bashrc.backup
11800861 -rw-r--r--  1 david david      807  sep 19  2018  .profile
```

No hay diferencias entre los nombres de archivos, a excepción de `.bash_profile` en el sistema UML y `.profile` en el anfitrión.

Actividad 3.8 Listados de metadatos de archivos: ls

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.**
  `$ ls -ltu`
- **Que contenga el campo "ctime" de los archivos del directorio especificado y que esté ordenado por dicho campo.**
  `$ ls -ltc`

Actividad 3.9 Metadatos del sistema de archivos: df y du

Resuelve las siguientes cuestiones relacionadas con la consulta de metadatos del SA:

1. 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.

```
# du /
703788  /

# du -h /
688M    /
```

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

```
# df -i /
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
LABEL=ROOT             65536   14667   50869   23% /
```

Se están usando 14667 inodos.

3. ¿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.

```
# du -h /etc /var
21M     /etc
14M     /var

# du -h /bin /usr /lib
5.3M    /bin
297M    /usr
24M     /lib
```

El directorio que más ocupa es `/usr` ya que almacena los ejecutables, archivos de código fuente, bibliotecas, documentación, etc. Por el contrario `/bin` es el que menos ocupa, ya que únicamente contiene ejecutables.

4. 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, es decir, los bloques de tamaño 4 KB del subárbol cuya raíz es /etc. ¿Cuál es el tamaño de bloque, por omisión, utilizado en el SA?

```
# du --block-size=4k /etc
5264
```

Por defecto el tamaño de bloque es de **1KB**. Se puede comprobar de dos maneras:

1. Consultando la información que proporciona tune2fs.

   ```
   # tune2fs -l /dev/loop0 | grep -i "Block size"
   Block size:               1024
   ```

2. Realizando una conversión:

   ```
   # du /etc
   21056
   # du -h /etc
   21M
   ```

   21M => 21504K / 21056 bloques = 1,021276596 K/bloque → **1KB = 1024B**

Actividad 3.10 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.

```
$ touch archivo.txt target_hardLink2.txt
$ ln archivo.txt hardLink
$ ln target_hardLink2.txt hardLink2
$ ln -s archivo.txt softLink
$ ls -lai
total 16
18360793 drwxrwxr-x 2 david david 4096 oct 11 21:50 .
32642728 drwxrwxr-x 9 david david 4096 oct 11 21:37 ..
18360795 -rw-r--r-- 2 david david   85 oct 11 21:50 archivo.txt
18360795 -rw-r--r-- 2 david david   85 oct 11 21:50 hardLink
18360797 -rw-r--r-- 2 david david    0 oct 11 21:45 hardLink2
18360799 lrwxrwxrwx 1 david david   11 oct 11 21:46 softLink -> archivo.txt
18360797 -rw-r--r-- 2 david david    0 oct 11 21:45 target_hardLink2.txt
```

¿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?

Porque los enlaces simbólicos no cuentan en el contador de enlaces, tan sólo los enlaces duros, que son realmente las referencias reales a los metadatos del archivo.

Actividad 3.11 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.
  `$ ls -Ll`
- 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.
  `$ ls -Llad`

Actividad 3.12 Creación de archivos especiales

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.

```
mknod [OPTION]... NAME TYPE [MAJOR MINOR]
```

El *major number* depende del tipo de dispositivo. Puede consultarse en `/proc/devices`

```
# mknod disp_bloques b 7 1
# mknod disp_caracteres c 7 2
# ls -li
14239 brw-r--r--  1 root root  7, 1 Oct 11 17:46 disp_bloques
14241 crw-r--r--  1 root root  7, 2 Oct 11 17:47 disp_caracteres
```