postdata9
www.wuolah.com/student/postdata9

32575

sesion3.pdf

Módulo I (actualizado)

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

Índice:

1. Control y gestión de CPU

Vamos a ver algunas órdenes para comprobar el estado del sistema y su rendimiento.

1.1 Orden uptime

Sintaxis: **$ uptime**

La información que muestra es, ordenada por columnas:

- **hora actual**: muestra la hora actual del sistema;
- **tiempo**: el tiempo que lleva el sistema en marcha;
- **usuarios**: número de usuarios conectados al sistema;
- **load average**: es la carga media del sistema en los últimos 1, 5 y 10 minutos. La carga del sistema es el número de procesos en la cola de ejecución del núcleo y suele ser menor o igual a 1.

```
[root@localhost ~]# uptime
 06:20:30 up 0 min,  1 user,  load average: 0.00, 0.00, 0.00
```

1.2 Orden w

Sintaxis: **$ w**

La orden **w** muestra en su primera línea de salida lo mismo que uptime, y añade además otras 8 columnas, las cuales son:

- **usuario**: es el usuario actual;
- **tty**: la terminal en la que iniciaron sesión;
- **de**: desde qué máquina remota iniciaron sesión;
- **login@**: a qué hora iniciaron sesión;
- **idle**: a partir de qué hora está inactivo;
- **jcpu**: es el tiempo total de CPU usado por el usuario desde el inicio de sesión;
- **pcpu**: tiempo de CPU del proceso actualmente en ejecución (PCPU siempre es menor que JCPU);
- **what**: da detalles sobre qué comando/aplicación está usando el usuario.

```
[root@localhost ~]# w
 06:21:15 up 1 min,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty0     -                06:20    0.00s  0.00s  0.00s w
```

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:

a) **¿Cuánto tiempo lleva en marcha el sistema?**

$ uptime

```
postdata9@ei143075:~/Escritorio$ uptime
 12:39:21 up 22 min,  1 user,  load average: 0,04, 0,05, 0,12
```

Como vemos con la orden uptime, la segunda columna nos muestra que el sistema lleva en marcha 22 minutos.

b) **¿Cuántos usuarios hay trabajando?**

$ uptime

Con la misma salida de antes, podemos comprobar con la cuarta columna que hay un usuario conectado al sistema.

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

$ uptime

Con la misma salida de antes, la última columna establece la carga media del sistema en los últimos 15 minutos y es de 0,12.

1.3 Orden time

Sintaxis: **$ time <programa>**

Esta orden mide el tiempo de ejecución de un programa y muestra un resumen del uso de los recursos del sistema. Calcula tres tiempos:

- **real**: es el tiempo que el programa ha estado ejecutándose;
- **usr**: el tiempo que se ha estado ejecutando en modo usuario;
- **sys**: el tiempo que se ha estado ejecutando en modo supervisor.

Con esto, podemos calcular el tiempo de espera de un proceso, que vendría dado por: tiempo_espera = real – user – sys

La orden primer ejecuta el programa pasado como argumento y lo muestra por pantalla, para después calcular los valores de los distintos tiempos real, user y sys.

```
[root@localhost ~]# time

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

Sintaxis: **$ nice -<prioridad> <nombre_proceso>**

$ renice <prioridad> <PID>

Linux realiza una planificación por prioridades y, por defecto, un proceso hereda la prioridad de su proceso padre. Con la orden **nice**, podemos modificar el valor de prioridad de un proceso, y su rango de valores está limitado a [-20, 19].

Cuanto menor sea el valor de la prioridad, mayor será la prioridad del proceso. Sólo el usuario **root** puede asignar valores negativos (con más prioridad) a los procesos, mientras que los usuarios normales se limitan al rango de números positivos.

Una orden nice se realiza a un proceso que no se está ejecutando y, una vez realizada la orden nice, se ejecuta el proceso; por tanto, no podemos realizar dos nice de forma seguida a un mismo proceso.

Para esto está la orden renice, que cambia la prioridad del proceso cuando se está ejecutando; en renice, le pasamos como parámetros la prioridad y el PID del proceso.

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.

- Creo el archivo prueba_procesos y lo edito:

  $ vi prueba_procesos

  (para escribir, pulsamos la tecla "A")

```
#!/bin/bash
#prueba_procesos
contador=0
a=0
for i in `seq 1 $1`
do
        a=$[$i * 2]
        sleep 3
        contador=$[$contador + $a]
done
echo "el valor de la variable es " $contador
```

(para guardar y salir → Pulsamos Esc, escribimos :wq y enter).

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?

- Ejecuto el archivo dos veces con dos valores distintos en background:

  $ ./prueba_procesos 174 &

  $ ./prueba_procesos 203 &

```
[root@localhost ~]# ./prueba_procesos 174 &
[3] 1252
[root@localhost ~]# ./prueba_procesos 203 &
[4] 1258
```

- Compruebo la prioridad:

  $ top

```
1252 root      20   0  3120 1120  984 S  0.0  0.1   0:00.00 prueba_procesos
1258 root      20   0  3120 1120  984 S  0.0  0.1   0:00.00 prueba_procesos
1281 root      20   0  2596 1056  852 R  0.0  0.1   0:00.11 top
1332 root      20   0  2012  432  380 S  0.0  0.0   0:00.00 sleep
```

  En la primera columna vemos el PID del proceso, que es el mismo que se nos devolvió cuando ejecutamos el proceso en la terminal en background.

  La tercera columna especifica la prioridad que, como podemos ver, en ambos procesos es 20.

- Cambiamos la prioridad de ambos procesos:

  $ renice -9 1252 (le damos la prioridad -9 al proceso 1252, lo hacemos más prioritario)

  $ renice 13 1258 (le damos la prioridad 13 al proceso 1258, es más prioritario qye antes que tenía 20, pero menos que el proceso 1252)

```
[root@localhost ~]# renice -9 1252
1252: old priority -13, new priority -9
[root@localhost ~]# renice 13 1258
1258: old priority 0, new priority 13
```

- Comprobamos que ha cambiado la prioridad:

  $ top

```
1252 root      11  -9  3120 1120  984 S  0.0  0.1   0:00.00 prueba_procesos
1258 root      33  13  3120 1124  984 S  0.0  0.1   0:00.00 prueba_procesos
1532 root      20   0  2596 1044  852 R  0.0  0.1   0:00.00 top
1541 root      20   0  2012  432  380 S  0.0  0.0   0:00.00 sleep
```

  Vemos que la tercera columna, que es la de prioridad, ha cambiado a -9 y a 13.

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

$ time ./prueba_procesos 10

$ time ./prueba_procesos 15

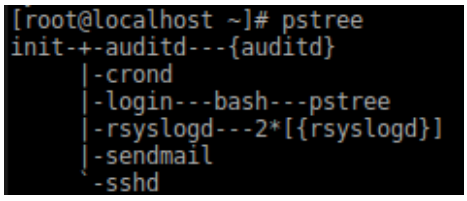
```
[root@localhost ~]# time ./prueba_procesos 10
el valor de la variable es  110

real    0m30.204s
user    0m0.000s
sys     0m0.000s
[root@localhost ~]# time ./prueba_procesos 23
el valor de la variable es  552

real    1m9.424s
user    0m0.000s
sys     0m0.000s
```

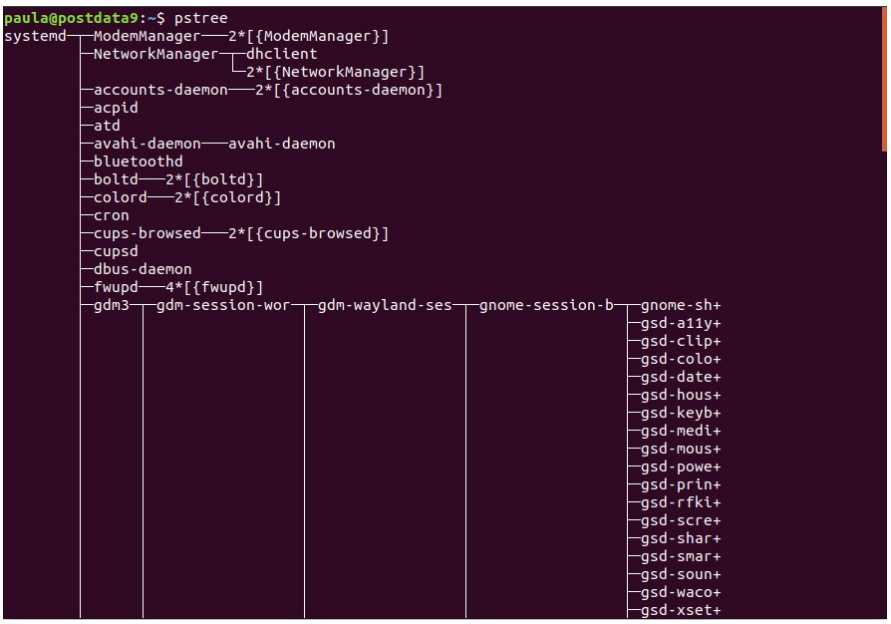
1.5 Orden pstree

Sintaxis: **$ pstree**

Visualiza un árbol de procesos en ejecución del sistema. Las opciones más interesantes son:

- **-a**: muestra los argumentos de la línea de órdenes;
- **-A**: dibuja el árbol con caracteres ASCII;
- **-G**: Usa los caracteres de VT100;
- **-h**: resalta el proceso actual y sus antepasados;
- **-H**: igual que -h, pero especificando el proceso en el argumento;
- **-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;
- **-p**: nos muestra los PIDS entre paréntesis después del nombre de cada proceso;
- **-u**: si el UID es distinto del UID del padre, se escribe el UID entre paréntesis;
- **-V**: visualiza información sobre la versión;
- **-Z**: muestra el contenido de seguridad para cada proceso.





1.6 Orden ps

Sintaxis: `$ ps <opciones>`

Esta orden utiliza el pseudo-sistema de archivos /proc. Si no usamos opciones, muestra información sobre los procesos lanzados por el usuario en cuatro columnas:

- **PID**: identificador de proceso;
- **TTY**: terminal asociado al proceso;
- **TIME**: tiempo total de uso de la CPU;
- **CMD (COMMAND)**: nombre del proceso, incluyendo argumentos si tiene;

Dependiendo de la opción que usemos, nos proporcionará más columnas con más información. Todas las columnas posibles que puede mostrar ps, dependiendo de la opción que usemos, son:

- **F**: las banderas de los procesos;
- **S** ó **STAT**: el código del estado del proceso;
  - **R**: running, en ejecución;
  - **S**: sleeping, durmiendo;
  - **T**: stopped, parado;
  - **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.
- **UID**: identificador del propietario del proceso;
- **PID**: identificador del proceso;
- **PPID**: identificador del proceso padre;
- **C** ó **CP**: uso de la CPU;
- **PRI**: prioridad del proceso;
- **NI**: valor nice;
- **ADDR**: dirección de memoria del proceso;
- **SZ** ó **VSZ**: uso de la memoria virtual;
- **RSS**: uso de la memoria real;
- **WCHAN**: dirección de memoria del evento que el proceso está esperando;
- **STIME** ó **START**: hora en la que empezó el proceso;
- **TT** ó **TTY**: terminal asociado al proceso;
- **TIME**: uso total de la CPU;

- **CMD (COMMAND)**: nombre del proceso, incluyendo lo argumentos si tiene.

Esta orden se ejecuta, normalmente, con las opciones -ef, donde:

- **-e**: selecciona todos los procesos que haya en el sistema;
- **-f**: lista con el formato completo.

```
[root@localhost ~]# ps
  PID TTY          TIME CMD
 1181 tty0     00:00:00 bash
 2160 tty0     00:00:00 ps
[root@localhost ~]# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 10:04 ?        00:00:00 /sbin/init
root         2     0  0 10:04 ?        00:00:00 [kthreadd]
root         3     2  0 10:04 ?        00:00:00 [ksoftirqd/0]
root         4     2  0 10:04 ?        00:00:00 [kworker/0:0]
root         5     2  0 10:04 ?        00:00:00 [kworker/u:0]
root         6     2  0 10:04 ?        00:00:00 [rcu_kthread]
root         7     2  0 10:04 ?        00:00:00 [watchdog/0]
root         8     2  0 10:04 ?        00:00:00 [cpuset]
root         9     2  0 10:04 ?        00:00:00 [khelper]
root        10     2  0 10:04 ?        00:00:00 [kworker/u:1]
root       115     2  0 10:04 ?        00:00:00 [sync_supers]
root       117     2  0 10:04 ?        00:00:00 [bdi-default]
root       118     2  0 10:04 ?        00:00:00 [kblockd]
root       134     2  0 10:04 ?        00:00:00 [rpciod]
root       135     2  0 10:04 ?        00:00:00 [kworker/0:1]
root       143     2  0 10:04 ?        00:00:00 [khungtaskd]
root       144     2  0 10:04 ?        00:00:00 [kswapd0]
root       192     2  0 10:04 ?        00:00:00 [fsnotify_mark]
root       216     2  0 10:04 ?        00:00:00 [ecryptfs-kthrea]
root       218     2  0 10:04 ?        00:00:00 [nfsiod]
root       240     2  0 10:04 ?        00:00:00 [glock_workqueue]
root       241     2  0 10:04 ?        00:00:00 [delete_workqueu]
root       246     2  0 10:04 ?        00:00:00 [gfs_recovery]
root       248     2  0 10:04 ?        00:00:00 [crypto]
root       263     2  0 10:04 ?        00:00:00 [kthrotld]
root       961     2  0 10:04 ?        00:00:00 [jbd2/ubda-8]
root       962     2  0 10:04 ?        00:00:00 [ext4-dio-unwrit]
root      1048     2  0 10:04 ?        00:00:00 [kauditd]
root      1086     1  0 10:04 ?        00:00:00 auditd
root      1104     1  0 10:04 ?        00:00:00 /sbin/rsyslogd -c 4
root      1129     1  0 10:04 ?        00:00:00 /usr/sbin/sshd
smmsp     1155     1  0 10:04 ?        00:00:00 sendmail: Queue runner@01:00:00
root      1166     1  0 10:04 ?        00:00:00 crond
root      1180     1  0 10:05 ?        00:00:00 login -- root
root      1181  1180  0 10:05 tty0     00:00:00 -bash
root      2161  1181  0 10:50 tty0     00:00:00 ps -ef
```

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.

Ejecutamos pstree y vemos los procesos que hay. Con ps -ef, podemos ver el PPID y el PID del proceso, y ver si el PPID de dicho proceso coincide con su padre, según el árbol mostrado en pstree.

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

No tiene asociado ningún terminal.

1.7 Orden top

Sintaxis: **$ top**

Proporciona una visión continuada de la actividad del procesador en tiempo real, mostrando las tareas que más utilizan la CPU y facilita una interfaz interactiva para manipular procesos.

La salida se compone de cinco primeras líneas, seguido de una tabla que se actualiza cada 5 segundos.

Las cinco primeras líneas se corresponden con:

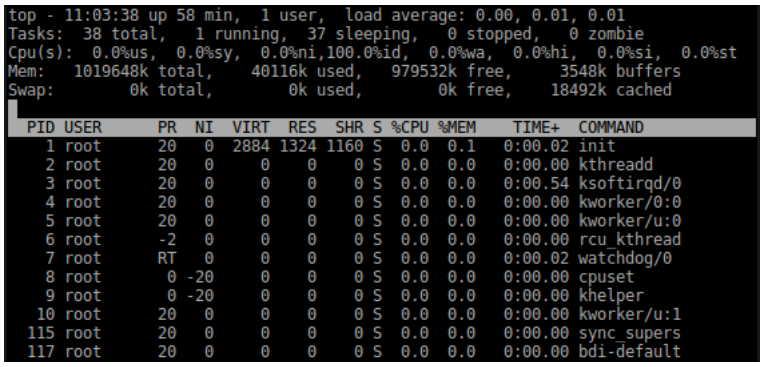
1. Las estadísticas de la orden uptime;
2. información sobre los procesos del sistema, como el número de procesos, los procesos en ejecución, durmiendo, parados o zombies;
3. el estado actual de la CPU, como el porcentaje en uso por usuarios, por el sistema, por procesos con valor nice positivo, por procesos esperando E/S, CPU desocupada por interrupciones hardware o software, espera voluntaria;
4. la memoria, como la memoria total disponible, usada, libre, cantidad usada en buffers y en memoria caché de página;
5. el espacio swap, como el espacio total disponible, usada, libre.

Los datos que se muestran en la tabla interactiva de abajo, es similar a la salida de ps, exceptuando **SHR** que muestra la cantidad de memoria compartida usada por la tarea.

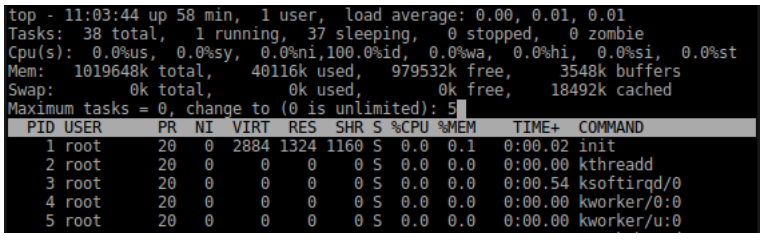
Esta orden permite realizar una serie de acciones sobre los procesos cuando estamos visualizando la tabla interactiva, como:

- cambiar la prioridad de algún proceso con la opción **r**;
- matar o enviar una señal con la opción **k**;
- ordenarlos según el PID con **N**, el uso de la CPU con **P**, tiempo con **A**, etc;
- modificar el número de procesos que se visualizan con **n**;
- para salir, **q**.

La orden top:



La orden top con la opción n:



1.8 Orden mpstat

Sintaxis: **$ mpstat <intervalo> <informes>**

Muestra estadísticas del procesador del sistema junto con la media global de todos los datos mostrados. Para poder utilizar esta orden es necesario tener instalado el paquete sysstat.

Podemos ejecutar la orden sin ninguna opción, lo que nos mostraría las estadísticas, o con un intervalo y un número, que nos permite realizar mpstat varias veces y con un intervalo determinado:

- **intervalo**: es la cantidad de tiempo que transcurre entre cada toma de datos, cada cuántos segundos debe mostrar los datos;
- **informes**: es el número de informes que se desean tomar, el número de muestras que queremos.

La salida que nos muestra es por columnas:

- **hora**: la hora a la que se ejecuta mpstat;
- **CPU**: número del procesador;
- **%usr**: porcentaje de uso de CPU con tareas a nivel de usuario;
- **%nice**: porcentaje de uso de CPU con tareas a nivel de usuario con prioridad nice (> 0);
- **%sys**: porcentaje de uso de CPU con tareas del sistema (no incluye interrupciones) en modo núcleo;
- **%iowait**: porcentaje de tiempo que la CPU estaba esperando por peticiones de E/S;
- **%irq**: porcentaje de tiempo que la CPU realiza interrupciones hardware;
- **%soft**: porcentaje de tiempo que la CPU realiza interrupciones software (la mayoría son sistemas);
- **$steal**: porcentaje de tiempo que la CPU espera involuntariamente mientras el hipervisor estaba prestando servicio a otro proceso virtual;
- **%guest**: porcentaje de tiempo de la CPU para ejecutar un procesador virtual;
- **%gnice**: porcentaje de tiempo de la CPU para ejecutar un nice guest(?;
- **%idle**: porcentaje de tiempo que la CPU está libre y el sistema no tiene peticiones de disco pendientes.

```
paula@postdata9:~$ mpstat
Linux 4.15.0-70-generic (postdata9)     13/12/19        _x86_64_        (4 CPU)

17:11:34     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
17:11:34     all    4,53    0,00    1,21    1,58    0,00    0,14    0,00    0,00    0,00   92,54
```

```
paula@postdata9:~$ mpstat 5 2
Linux 4.15.0-70-generic (postdata9)     13/12/19        _x86_64_        (4 CPU)

17:12:00     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
17:12:05     all   13,51    0,00    3,28    0,05    0,00    0,72    0,00    0,00    0,00   82,45
17:12:10     all    7,70    0,00    1,22    0,00    0,00    0,20    0,00    0,00    0,00   90,89
Media:       all   10,59    0,00    2,24    0,03    0,00    0,46    0,00    0,00    0,00   86,69
```

En la captura de arriba, hemos ejecutado mpstat 2 veces con un espacio de tiempo entre ejecuciones de 5 segundos.

Actividad 3.4. Estadísticas de recursos del sistema

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

```
paula@postdata9:~$ mpstat
Linux 4.15.0-70-generic (postdata9)     13/12/19        _x86_64_        (4 CPU)

17:14:06     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
17:14:06     all    4,65    0,00    1,23    1,54    0,00    0,14    0,00    0,00    0,00   92,43
```

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

El porcentaje de tiempo empleado para interrupciones hardware se encuentra en la columna de %irq que, según la salida anterior, es del 0,00%.

b) **¿Y qué porcentaje en tratar interrupciones software?**

El porcentaje de tiempo empleado para interrupciones software se encuentra en la columna %soft que, según la salida anterior, es del 0,14%.

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

$ top | head -5

```
top - 17:19:05 up  1:34,  1 user,  load average: 0,60, 0,57, 0,57
Tareas: 272 total,   1 ejecutar,  216 hibernar,    0 detener,    1 zombie
%Cpu(s):  4,8 usuario,  1,2 sist,  0,0 adecuado, 92,3 inact,  1,5 en espera,  0,0 hardw int,  0,1 softw int,
KiB Mem :  8031828 total,  5024080 libre,  1196932 usado,  1810816 búfer/caché
KiB Intercambio:  8252412 total,  8252412 libre,        0 usado.  6189960 dispon Mem
```

Vemos las 5 primeras líneas de la orden top, en la que podemos ver la información que se nos pide.

El espacio de swap libre, es el espacio de intercambio. Tiene un total de 8252412KiB, de los cuales no usa ninguno, por lo que tiene todo libre.

2. Control y gestión de memoria

2.1 Orden free

Sintaxis: **$ free**

Se utiliza para visualizar el uso actual de la memoria y consume menos recursos que top. Dicha orden informa del consumo de la memoria real o principal (**RAM**) y del consumo de la memoria de espacio de intercambio (**swap**).

```
[root@localhost ~]# free
             total       used       free     shared    buffers     cached
Mem:       1019648      40100     979548          0       3560      18704
-/+ buffers/cache:      17836    1001812
Swap:            0          0          0
```

2.2 vmstat

Sintaxis: **$ vmstat <intervalo> <informes>**

Esta orden sirve para supervisar el sistema mostrando información de memoria y de procesos, E/S y CPU. Su salida proporciona una media desde el último arranque del sistema y se puede obtener varios informes con un determinado intervalo de tiempo entre ellos como mpstat.

La salida de esta orden se agrupa en:

- **procs**: los valores de este grupo son calculados utilizando /proc:
  - **r**: número de procesos esperando para ejecutarse;
  - **b**: número de procesos durmiendo ininterrumpidamente.
- **memory**:
  - **swpd**: cantidad de memoria virtual usada;
  - **free**: cantidad de memoria sin usar;
  - **buff**: cantidad de memoria usada como buffers;
  - **cache**: cantidad de memoria usada como caché;
  - **inact**: cantidad de memoria inactiva;
  - **active**: cantidad de memoria activa.
- **swap**:
  - **si**: cantidad de memoria intercambiada desde el disco;
  - **so**: cantidad de memoria intercambiada al disco.
- **io**:
  - **bi**: número de bloques recibidos de un dispositivo de bloque;
  - **bo**: número de bloques enviados al dispositivo de bloque.
- **system**:
  - **in**: número de interrupciones por segundo, incluyendo el reloj;
  - **cs**: número de cambios de contexto por segundo.

- **cpu**: porcentajes del tiempo total de la CPU:
  - **us**: tiempo que emplea para procesos de usuario;
  - **sy**: tiempo que emplea para tareas del sistema;
  - **id**: tiempo sin hacer nada;
  - **wa**: tiempo esperando por E/S;
  - **st**: tiempo dedicado a máquina virtual.

```
[root@localhost ~]# vmstat
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 979548   3560  18728    0    0     5     0   98    6  0  0 100  0
0
```

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 5 3 >> ejercicio

```
[root@localhost ~]# vmstat 5 3 >> ejercicio
[root@localhost ~]# cat ejercicio
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      0 979548   3560  18732    0    0     5     0   98    6  0  0 100  0  0
 0  0      0 979532   3560  18732    0    0     0     0   99    3  0  0 100  0  0
 0  0      0 979532   3568  18732    0    0     0     6  100    4  0  0 88 12  0
```

De esta forma, hemos pedido que realice 3 informes con un tiempo entre ellos de 5 segundos de diferencia, almacenando la salida en el fichero *ejercicio*.

---

3. Control y gestión de dispositivos de E/S

El SO necesita mantener una estructura de datos por cada archivo, que contiene información necesaria para poder trabajar con el archivo. A esta información se le conoce como **metadatos de archivo** ó **atributos de archivo**. Un metadato fundamental es el **nombre de archivo**, ya que gracias a él podemos identificarlo.

En Unix, se almacena una **referencia a los metadatos** en la entrada de directorio y un **inodo** que son los propios metadatos del archivo.

También tenemos tipos de archivos especiales para dispositivos de bloque y para dispositivos de caracteres, que proporcionan al usuario una interfaz para trabajar con dispositivos de E/S y almacenamiento secundario.

3.1 Consulta de información de archivos

La orden ls, una vez conocemos el concepto de metadatos de archivo, es muy útil, por lo que vamos a ver las opciones más interesantes y que nos pueden servir para la administración de sistemas:

- **ls -l**: metadatos de cada archivo en formato largo;
- **ls -n**: en la columna de usuario y grupo muestra el UID y PID;
- **ls -la**: como -l, pero tiene en cuenta las entradas ocultas (directorio . y ..);
- **ls -li**: como -l, añadiendo el campo del número de inodo;
- **ls -lh**: como -l, pero el campo del tamaño del archivo está en KB, MB ó GB.

En la salida en formato largo, se nos muestra los metadatos en columnas. Con el formato largo, las columnas serían:

- **tipo y permisos**: del archivo, la estructura sería la siguiente:

  ```
  tipo   u   g   o
  d      rwx rwx rwx
  ```

  el tipo de archivo puede ser:

  - **-**: archivo regular;
  - **d**: directorio;
  - **l**: enlace simbólico;
  - **b**: archivo especial de dispositivo de bloques;
  - **c**: archivo especial de dispositivo de caracteres;
  - **p**: archivo FIFO para comunicaciones entre procesos.

  Los permisos son **u** para el usuario, **g** para grupos y **o** para otros, **r** (read), **w** (write) y **x** (execute). Si no tiene algún permiso, aparece **-**.
- **número de enlaces;**
- **propietario;**
- **grupo propietario;**
- **fecha**: de la última modificación;
- **nombre del archivo.**

Con ls también podemos ordenar la salida:

- **ls -X**: alfabéticamente por entrada de directorio;
- **ls -t**: por fecha de modificación del archivo;
- **ls -u**: según la hora de acceso;
- **ls -c**: ordena según ctime (última modificación de la información del estado del archivo).

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?

Para encontrar un archivo de dispositivo de bloques y de caracteres, cuando realizamos un ls -l, debe tener como primera letra antes de los permisos una b o una c, pero no he encontrado ninguno.

Al realizar los dos ls -la, comprobamos que el único archivo que se encuentra tanto en /root como en nuestro /home es .bashrc.

```
[root@localhost ~]# ls -la /root
total 32
dr-xr-x---  2 root root 4096 Dec 13 11:35 .
dr-xr-xr-x 22 root root 4096 Dec 13 10:04 ..
-rw-------  1 root root   53 Sep 13  2011 .bash_history
-rw-r--r--  1 root root   18 Mar 30  2009 .bash_logout
-rw-r--r--  1 root root  176 Mar 30  2009 .bash_profile
-rw-r--r--  1 root root  176 Sep 22  2004 .bashrc
-rw-r--r--  1 root root  100 Sep 22  2004 .cshrc
-rw-r--r--  1 root root  129 Dec  3  2004 .tcshrc
```

```
postdata9@ei143062:~$ ls -la /home/postdata9/
total 104
drwxr-xr-x 18 postdata9 alumno 4096 dic 13 16:04 .
drwxr-xr-x  4 root      root   4096 dic 13 15:41 ..
-rw-r--r--  1 postdata9 alumno   61 dic 13 15:41 .bashrc
drwx------  8 postdata9 alumno 4096 dic 13 16:12 .cache
drwxr-xr-x 16 postdata9 alumno 4096 dic 13 15:54 .config
drwx------  3 postdata9 alumno 4096 dic 13 15:41 .dbus
drwxr-xr-x  2 postdata9 alumno 4096 dic 13 15:41 Descargas
-rw-r--r--  1 postdata9 alumno   23 dic 13 15:41 .dmrc
drwxr-xr-x  2 postdata9 alumno 4096 dic 13 15:41 Documentos
drwx------  3 postdata9 alumno 4096 dic 13 15:50 .emacs.d
drwxr-xr-x  2 postdata9 alumno 4096 dic 13 17:23 Escritorio
-rwxr-xr-x  1 postdata9 alumno 6024 dic 13 15:41 .face
drwx------  2 postdata9 alumno 4096 dic 13 15:49 .gconf
drwxr-xr-x  2 postdata9 alumno 4096 dic 13 15:41 Imágenes
drwxr-xr-x  3 postdata9 alumno 4096 dic 13 15:41 .local
drwxr-xr-x  2 postdata9 alumno 4096 dic 13 15:41 Música
lrwxrwxrwx  1 postdata9 alumno   26 dic 13 15:41 .netbeans -> /usr/local/plugin
snetbeans
drwxr-xr-x  2 postdata9 alumno 4096 dic 13 15:41 Plantillas
drwxr-xr-x  2 postdata9 alumno 4096 dic 13 15:41 Público
-rw-r--r--  1 postdata9 alumno   38 dic 13 15:41 .session
drwx------  4 postdata9 alumno 4096 dic 13 15:41 .thumbnails
drwxr-xr-x  3 postdata9 alumno 4096 dic 13 16:04 .uml
drwxr-xr-x  2 postdata9 alumno 4096 dic 13 15:41 Vídeos
-rw-------  1 postdata9 alumno   53 dic 13 15:41 .Xauthority
-rw-r--r--  1 postdata9 alumno   44 dic 13 15:41 .xsession
-rw-------  1 postdata9 alumno  336 dic 13 15:41 .xsession-errors
```

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 -lut

```
postdata9@ei143062:~$ ls -lut
total 32
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 17:24 Escritorio
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 16:12 Imágenes
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 15:42 Descargas
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 15:42 Documentos
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 15:42 Música
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 15:42 Público
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 15:42 Vídeos
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 15:41 Plantillas
```

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

  $ ls -lc

```
postdata9@ei143062:~$ ls -lc
total 32
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 15:42 Descargas
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 15:42 Documentos
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 17:23 Escritorio
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 15:42 Imágenes
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 15:42 Música
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 15:42 Plantillas
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 15:42 Público
drwxr-xr-x 2 postdata9 alumno 4096 dic 13 15:42 Vídeos
```

La orden **df** permite visualizar para cada SA montado, información sobre su capacidad de almacenamiento total, espacio usado, espacio libre restante y el punto de montaje en la jerarquía de directorios.

```
[root@localhost ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
LABEL=ROOT             1032088    411180    568480  42% /
tmpfs                  1032088    411180    568480  42% /dev/shm
/tmp                    509824         0    509824   0% /tmp
```

Si lo utilizamos con la opción -i, podemos ver toda la información sobre los inodos de cada SA montado, como se muestra a continuación:

```
[root@localhost ~]# df -i
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
LABEL=ROOT             65536   14665   50871   23% /
tmpfs                  65536   14665   50871   23% /dev/shm
/tmp                  127456       2  127454    1% /tmp
```

La orden **du** muestra el número de bloques de discos asignados (estén o no completamente ocupados) a todos los archivos que cuelgan del directorio especificado. En la salida de la orden tenemos dos columnas, la primera es el número de discos asignados y la segunda columna el path del archivo.

```
[root@localhost ~]# du
28      .
```

Actividad 3.9. Metadatos del sistema de archivos: df y du

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

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 /root

$ du -h /root

```
[root@localhost ~]# du /root
28      /root
[root@localhost ~]# du -h /root
28K     /root
```

La partición raíz tiene 28 KB.

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

$ df -i /

```
[root@localhost ~]# df -i /
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
LABEL=ROOT             65536   14665   50871   23% /
```

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 /etc | tail -1 (para quedarnos con la última entrada, que es en la que aparece el tamaño del directorio)

```
[root@localhost ~]# du /etc | tail -1
21056   /etc
```

El tamaño de /etc es 21056 KBytes.

$ du /var | tail -1

```
[root@localhost ~]# du /var | tail -1
13572   /var
```

El tamaño de /var es de 13572 KBytes.

$ du /bin | tail -1 tiene un tamaño de 5384 KBytes

$ du /usr | tail -1 tiene un tamaño de 303176 KBytes

$ du /lib | tail -1 tiene un tamaño de 24540 KBytes

```
[root@localhost ~]# du /bin | tail -1
5384    /bin
[root@localhost ~]# du /usr | tail -1
303176  /usr
[root@localhost ~]# du /lib | tail -1
24540   /lib
```

El directorio que menos ocupa es /bin, ya que en él se almacena solo ejecutables, mientras el que más ocupa es /usr, debido a que se guardan ahí todos los programas que tenemos instalados en el sistema.

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. 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?

$ du -B 4 /etc | tail -1

```
[root@localhost ~]# du -B 4 /etc | tail -1
5390336 /etc
```

El directorio /etc tiene 5390336 bloques cuyo tamaño es 4 KBytes.

$ tune2fs -l /dev/sda1 | grep -i "Block size"

para saber el tamaño de bloque por omisión utilizado en el SA

3.3 Creación de enlaces a archivos

Los enlaces a archivos nos permiten tener varios nombres para un mismo archivo y son referencias a otros archivos. Tenemos dos tipos de enlaces:

- **enlaces simbólicos (soft link)**: son enlaces a archivos, como un acceso directo, que contienen el nombre del archivo y la ruta en la que se encuentra.
- **enlaces duros (hard link)**: es un enlace a un archivo, que almacena los metadatos del fichero al que apunta; todo archivo contiene un enlace duro.

Con ls -la podemos comprobar que todos los directorios tienen dos enlaces duros que crea el SO de forma automática para mantener la jerarquía de directorios. Estos enlaces duros son:

- **"."** : para el enlace duro al directorio;
- **".."** : para el enlace duro al directorio padre del archivo.

La orden **ln** crea enlaces duros o simbólicos sobre un archivo ya creado. Su sintaxis sería ln <nombre_archivo> <nombre_enlace>; la opción -s para enlaces simbólicos.

```
postdata9@ei141073:~/Escritorio$ touch archivo
postdata9@ei141073:~/Escritorio$ ln archivo p1
postdata9@ei141073:~/Escritorio$ ln -s archivo p1s
```

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. ¿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?

$ touch archivo.txt target_hardLink2.txt

$ ls

```
postdata9@ei141073:~/Escritorio$ touch archivo.txt target_hardLink2.txt
postdata9@ei141073:~/Escritorio$ ls
archivo.txt  Departamentos  Home  target_hardLink2.txt
```

$ ln -s archivo.txt ./softLink

$ ln archivo.txt hardlLink

$ ln target_hardLink2.txt hardLink2

```
postdata9@ei141073:~/Escritorio$ ln archivo.txt hardLink
postdata9@ei141073:~/Escritorio$ ln -s archivo.txt softLink
postdata9@ei141073:~/Escritorio$ ln target_hardLink2.txt hardLink2
```

$ ls -l

```
postdata9@ei141073:~/Escritorio$ ls -l
total 0
-rw-r--r-- 1 postdata9 alumno  0 dic 13 19:00 33.pmg
-rw-r--r-- 2 postdata9 alumno  0 dic 13 19:00 archivo.txt
lrwxrwxrwx 1 postdata9 alumno 12 dic 13 18:56 Departamentos -> /fenix/depar
-rw-r--r-- 2 postdata9 alumno  0 dic 13 19:00 hardLink
-rw-r--r-- 2 postdata9 alumno  0 dic 13 19:00 hardLink2
lrwxrwxrwx 1 postdata9 alumno 24 dic 13 18:56 Home -> /fenix/alum/d3/postdata9
lrwxrwxrwx 1 postdata9 alumno 11 dic 13 19:01 softLink -> archivo.txt
-rw-r--r-- 2 postdata9 alumno  0 dic 13 19:00 target_hardLink2.txt
```

Tiene dos por el enlace duro que tenía cuando se creó el archivo y el que le hemos enlazado nosotros; el enlace simbólico no suma en este contador.

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 -laiL

  Con -l obtenemos el formato largo, con -a incluye los archivos ocultos, con -i añade información de los inodos y con -L muestra la información del archivo al que enlaza el enlace simbólico.

```
postdata9@ei141073:~/Escritorio$ ls -laiL
total 16
     1699 drwxr-xr-x  2 postdata9 alumno 4096 dic 13 19:02 .
     1696 drwxr-xr-x 15 postdata9 alumno 4096 dic 13 18:57 ..
    61601 -rw-r--r--  1 postdata9 alumno    0 dic 13 19:00 33.pmg
    12975 -rw-r--r--  2 postdata9 alumno    0 dic 13 19:00 archivo.txt
163643393 drwxr-xr-x 13 root      root   4096 jul 11  2013 Departamentos
    12975 -rw-r--r--  2 postdata9 alumno    0 dic 13 19:00 hardLink
    12973 -rw-r--r--  2 postdata9 alumno    0 dic 13 19:00 hardLink2
207947054 drwx------ 15 postdata9 alumno 4096 dic 11 08:57 Home
    12975 -rw-r--r--  2 postdata9 alumno    0 dic 13 19:00 softLink
    12973 -rw-r--r--  2 postdata9 alumno    0 dic 13 19:00 target_hardLink2.txt
```

- **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 -laid

  Con -l obtenemos el formato largo, con -a incluye los archivos ocultos, con -i añade información de los inodos y con -d muestra lo que nos pide el enunciado.

```
postdata9@ei141073:~/Escritorio$ ls -laid
1699 drwxr-xr-x 2 postdata9 alumno 4096 dic 13 19:02 .
```

3.4 Archivos especiales de dispositivo

Existen dos tipos de archivos especiales de dispositivo en un SO tipo UNIX:

- **de bloques**: representa a los dispositivos de bloques, que suelen coincidir con los dispositivos de almacenamiento persistente, los ramdisks y los dispositivos loop;
- **de caracteres**: representa a los dispositivos de caracteres del tipo puertos serie, paralelo y USB, consola virtual, audio, dispositivos de terminal.

Estos archivos especiales de dispositivos suelen ubicarse en **/dev**. Estos archivos tienen dos números asociados:

- **principal (major)**: determina el controlador al que está conectado el dispositivo;
- **secundario (minor)**: determina al dispositivo en sí.

Estos número permiten identificar al dispositivo en el kernel, en la tabla de dispositivos.

Podemos crear estos archivos con la orden **mknod**. La sintaxis es:

$ mknod <nombre> <tipo> <major> <minor>

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 bloques b 13 9

$ mknod caracteres c 13 9

```
[root@localhost ~]# mknod bloques b 13 9
[root@localhost ~]# mknod caracteres c 13 9
[root@localhost ~]# ls -li
total 0
14175 brw-r--r-- 1 root root 13, 9 Dec 13 13:13 bloques
14238 crw-r--r-- 1 root root 13, 9 Dec 13 13:13 caracteres
```

4. Preguntas de repaso

1. ¿Cómo podríamos ver únicamente el número de usuarios que hay trabajando en el sistema?

$ uptime | cut -d "," -f2

dividimos la salida de uptime cortando por la "," y nos quedamos con la columna de los usuarios

2. ¿Cómo podemos ver únicamente el tiempo que lleva en marcha el sistema?

$ uptime -p

3. Indique la orden necesaria para listar los archivos del directorio home ordenados según su último acceso.

$ ls -ltu

4. ¿Cómo podríamos saber el tamaño del directorio home en formato entendible?

$ df -h $HOME

$ du -h /home | tail -1

$ ls -hl $HOME | head -1

5. Indique la orden para visualizar el tamaño de bloque del SA de un dispositivo cualquiera (e.g. , /dev/sda1).

$ sudo tune2fs -l /dev/sda1 | grep -i "Block size" (la opción -i es para que no diferencie entre mayúsculas y minúsculas, para que se pueda poner también "block size")

6. Sobre la orden ls -l, ordena por:

Orden alfabético del nombre.

7. ¿Podrías realizar el mismo comando de forma interactiva lanzando top, luego pulsando Shift+tecla? ¿Cómo?

Se puede realizar con una tecla, pero no para ordenar a la vez por memoria, PID y usuario. Por separado sería:

Memoria: SHIFT + M

PID: SHIFT + N

usuario: SHIFT + U

8. Selecciona la respuesta correcta. ¿Qué hace el comando top -o %MEM -o PID -o USUARIO?

Muestra los procesos ordenados, de forma descendente, según el consumo de la memoria, PID, y nombre de usuario, siguiendo ese mismo orden: primero, memoria; después PID; y, por último, el nombre del usuario.

9. Responda Verdadero o Falso. Habiendo ejecutado con éxito la orden ln $k1 $k2:

1. **El contador de enlaces duros de $k1 no ha cambiado como resultado de la ejecución de la orden:**

   Falso

2. **$k1 podría ser de tipo enlace simbólico:**

   Verdadero

3. **$k1 podría ser de tipo directorio:**

   Falso

4. **$k2 podría ser de tipo directorio:**

   Falso

5. **$k2 podría ser de tipo enlace simbólico:**

   Falso

6. **El número de inodo de $k1 es distinto de $k2**

   Falso

10. Elija cuál de las siguientes afirmaciones es correcta:

1. **La ejecución de la orden nice -0 ejecutable:**

   Puede lanzarlo con cualquier usuario.

2. **La ejecución de la orden nice -0 ejecutable se realiza con:**

   Es equivalente a lanzar $ ejecutable.

3. **La ejecución de la orden nice --19 ejecutable:**

   Sólo puede lanzarlo root.

4. **La ejecución de la orden nice -19 ejecutable se realiza con el:**

Mayor valor numérico de prioridad posible.

5. **La ejecución de la orden renice -19 2880:**

   Sólo puede lanzarlo root

6. **La ejecución de la orden renice -19 2880 cambia al:**

   Ninguna

11. Los enlaces duros (hard links):

Aumentan más de 1 el contador de enlaces mostrado en la tercera columna del comando ls -li, apuntan a los metadatos del archivo enlazado y el inodo es el mismo que éste.

12. Los enlaces blandos o soft links:

No aumentan más de 1 el contador de enlaces mostrado en la tercera columna del comando ls -li, pero por el hecho de crearlos tienen asociado un enlace duro.

13. 10. Si ejecutamos el comando ln –s $n1 $n2:

El contador de enlaces duros de $n1 no ha cambiado como resultado de la ejecución de la orden. Y $n1 podría ser de tipo directorio, o también podría ser de tipo enlace simbólico.

14. Desde un usuario sin permisos de superusuario, la ejecución de la orden: $ nice -19 ejecutable.

Lanza la ejecución del ejecutable con el menor peso posible respecto a la asignación de CPU.

15. Si ejecutamos desde un usuario sin permisos de superusuario, la ejecución de la orden: nice -0 ejecutable.

Lanza la ejecución del ejecutable con el mayor peso posible, en nuestro perfil, respecto a la asignación de CPU.

16. La orden top -o %CPU:

Muestra los procesos ordenados, de forma descendente, según el consumo de la CPU.

17. Si la orden uptime devuelve lo siguiente: 20:03 up 8 days, 6:49, 2 users, load averages: 1,70 2,03 2,14

El sistema lleva levantado 8 días, 6 horas y 49 minutos. La carga media de la CPU en el último minuto fue de 1,70. La carga media de la CPU en los últimos 5 minutos fue de 2,03. Y la carga media de la CPU en los últimos 15 minutos fue de 2,14.

18. Sobre la orden ls -l, ordena por:

Orden alfabético del nombre.