Sistemas Operativos

Formulario de auto-evaluación

Sesión 3. Monitorización del sistema.

Nombre y apellidos:

Jose Luis Izquierdo Mañas

a) Cuestionario de actitud frente al trabajo.

El tiempo que he dedicado a la preparación de la sesión antes de asistir al laboratorio ha sido de 0..... minutos.

1. He resuelto todas las dudas que tenía antes de iniciar la sesión de prácticas: ...si... (si/no). En caso de haber contestado “no”, indica los motivos por los que no las has resuelto:

2. Tengo que trabajar algo más los conceptos sobre:

3. Comentarios y sugerencias:

b) Cuestionario de conocimientos adquiridos.

Mi solución a la **actividad 3.1** ha sido:

Tras ejecutar la orden uptime:

hulidex@GE60-2PC:~$ uptime

00:15:56 up 4:10, 1 user, load average: 0,12, 0,21, 0,17

¿Cuanto tiempo lleva en marcha el sistema?

el sistema lleva 4 horas y 10 minutos en marcha

¿Cuantos usuarios hay trabajando?

Hay un sólo usuario

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

La carga media en los últimos 15 minutos es de 0,17

Mi solución a la **actividad 3.2** ha sido:

código del script:

```
inicio=0
while [ $inicio -lt $1 ]
do
        sleep 1
        inicio=`expr $inicio + 1`
done

echo "Valor $inicio"
exit
```

En considerado interesante añadir la orden sleep en cada iteración del bucle para que realmente haya una espera entre iteración e iteración...

Lanzamos 3 procesos:

```
hulidex@GE60-2PC:~$ time ./ejer2_scrip.sh 70000 &
[1] 22274
hulidex@GE60-2PC:~$ time ./ejer2_scrip.sh 70000 &
[2] 24414
hulidex@GE60-2PC:~$ time ./ejer2_scrip.sh 70000 &
[3] 26409
```

Comprobamos su estado:

```
hulidex@GE60-2PC:~$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  2365  4254  0  80   0 - 10487 wait   pts/4    00:00:00 bash
0 R  1000 18005  2365  0  80   0 - 12005 -      pts/4    00:00:00 ps
1 S  1000 22274  2365  0  80   0 - 10487 wait   pts/4    00:00:00 bash
1 S  1000 22276 22274 14  80   0 - 10487 pipe_w pts/4    00:00:01 bash
1 S  1000 24414  2365  0  80   0 - 10487 wait   pts/4    00:00:00 bash
1 S  1000 24415 24414 13  80   0 - 10487 pipe_w pts/4    00:00:00 bash
1 S  1000 26409  2365  0  80   0 - 10487 wait   pts/4    00:00:00 bash
1 S  1000 26413 26409 14  80   0 - 10487 pipe_w pts/4    00:00:00 bash
```

Como podemos contemplar nuestros procesos se han creado todos con la prioridad 80

Modificamos la prioridad de dos de ellos:

```
hulidex@GE60-2PC:~$ sudo renice -20 22274
22274 (process ID) old priority 0, new priority -20
hulidex@GE60-2PC:~$ renice 19 26409
26409 (process ID) old priority 0, new priority 19
```

Volvemos a comprobar el estado de los procesos:

```
hulidex@GE60-2PC:~$ ps -l
```

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  2365  4254  0  80   0 - 10487 wait   pts/4    00:00:00 bash
1 S  1000 22274  2365  0  60 -20 - 10487 wait   pts/4    00:00:00 bash
1 S  1000 22276 22274 15  80   0 - 10487 pipe_w pts/4    00:00:05 bash
1 S  1000 24414  2365  0  80   0 - 10487 wait   pts/4    00:00:00 bash
1 S  1000 24415 24414 14  80   0 - 10487 pipe_w pts/4    00:00:04 bash
0 R  1000 24868  2365  0  80   0 - 12005 -      pts/4    00:00:00 ps
1 R  1000 24873 24415  0  80   0 - 10487 -      pts/4    00:00:00 bash
1 R  1000 24879 22276  0  80   0 - 10487 -      pts/4    00:00:00 bash
1 S  1000 26409  2365  0  99  19 - 10487 wait   pts/4    00:00:00 bash
1 S  1000 26413 26409 15  80   0 - 10487 pipe_w pts/4    00:00:04 bash
```

Ahora podemos apreciar ahora la columna "NI" está modificada con respecto a la gráfica anterior, debido

a los cambios de prioridad realizados, además hay que notar que el proceso con PID 22274 será el que más

prioridad tenga, el proceso con PID 26409 será el que menos prioridad tendrá y los demás tienen la misma prioridad.

Tiempo de ejecución del proceso con PID 22274

```
real    1m14.471s
user    0m2.468s
sys     0m9.044s
```

Tiempo de ejecución del proceso con PID 24414

```
real    1m14.483s
user    0m2.396s
sys     0m8.828s
```

Tiempo de ejecución del proceso con PID 26409

```
real    1m14.571s
user    0m2.616s
sys     0m8.708s
```

Mi solución a la **actividad 3.4** ha sido:

tras ejecutar la orden:

hulidex@GE60-2PC:~$ mpstat 5 36

Linux 4.4.0-45-generic (GE60-2PC) 27/10/16 _x86_64_ (4 CPU)

...

01:58:31 CPU %usr %nice %sys %iowait %irq %soft %steal %guest %gnice %idle

Average: all 10,46 0,00 20,45 1,90 0,00 0,04 0,00 0,00 0,00 67,15

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

0,00%

Orden usada: mpstat

b) ¿Y qué porcentaje en tratar interrupciones software?

0,04%

orden usada: mpstat

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

usando la orden:

hulidex@GE60-2PC:~/Dropbox/UGR/2º CURSO/1º Cuatrimestre/Sistemas Operativos/Prácticas/Práctica 3$ free -m

	total	used	free	shared	buff/cache	available
Mem:	11936	2427	5273	679	4234	8482
Swap:	3905	0	3905			

Espacio usado 0 MB, Espacio Libre 3905MB

Mi solución a la **actividad 3.6** ha sido:

```
#!/bin/bash
vmstat >> monitorizacion.txt
num_procs=`cat "monitorizacion.txt"|tail -1|tr -s " " "|cut -d " " -f2`
num_procs_swap=`cat "monitorizacion.txt"|tail -1|tr -s " " "|cut -d " " -f17`
incremento_swap=`cat "monitorizacion.txt"|tail -1|tr -s " " "|cut -d " " -f9`
mem_libre=`cat "monitorizacion.txt"|tail -1|tr -s " " "|cut -d " " -f5`
echo "Hay " $num_procs " procesos en cola de ejecución"
echo "Hay " $num_procs_swap " procesos ejecutándose en el área de intercambio"
```

```
if [ $incremento_swap -eq 0 ]
then
echo "No hay incremento de procesos pasándose a la memoria de intercambio"
else
echo "Se ha incrementado el numero de procesos que van a la memoria de
intercambio en " $incremento_swap
fi
echo "La memoria libre es de " $mem_libre "KB"
-
 Para realizar una revisión periódica( en este ejemplo que se actualice cada 4 segundos) ,
podemos ejecutar en un terminal el siguiente comando:
# watch –n 4 ./script_4-2.sh
```

Mi solución a la **actividad 3.8** ha sido:

a) Que contenga el campo “access time” de los archivos del directorio especificado y que esté ordenado por dicho campo.

```
hulidex@GE60-2PC:~$ ls -u -lt
total 44
drwxr-xr-x 2 hulidex hulidex 4096 oct 29 20:05 Templates
drwxr-xr-x 2 hulidex hulidex 4096 oct 29 20:05 Music
drwxrwxr-x 3 hulidex hulidex 4096 oct 29 20:05 VirtualBox VMs
drwxr-xr-x 3 hulidex hulidex 4096 oct 29 20:05 Pictures
drwxr-xr-x 2 hulidex hulidex 4096 oct 29 20:05 Documents
drwxrwxr-x 2 hulidex hulidex 4096 oct 29 20:05 NewFolder
drwxr-xr-x 2 hulidex hulidex 4096 oct 29 20:05 Public
drwxr-xr-x 3 hulidex hulidex 4096 oct 29 20:05 Videos
drwx------ 8 hulidex hulidex 4096 oct 29 19:49 Dropbox
drwxr-xr-x 2 hulidex hulidex 4096 oct 29 19:49 Desktop
drwxr-xr-x 6 hulidex hulidex 4096 oct 29 17:37 Downloads
```

b) Que contenga el campo “ctime” de los archivos del directorio especificado y que esté ordenado por dicho campo.

```
hulidex@GE60-2PC:~$ ls -c -lt
total 44
drwx------ 8 hulidex hulidex 4096 oct 29 19:49 Dropbox
drwxr-xr-x 6 hulidex hulidex 4096 oct 27 20:20 Downloads
drwxr-xr-x 3 hulidex hulidex 4096 oct 17 22:48 Videos
drwxr-xr-x 3 hulidex hulidex 4096 oct 17 22:48 Pictures
drwxrwxr-x 2 hulidex hulidex 4096 oct 13 17:55 NewFolder
drwxrwxr-x 3 hulidex hulidex 4096 oct  9 19:15 VirtualBox VMs
drwxr-xr-x 2 hulidex hulidex 4096 oct  9 14:49 Desktop
drwxr-xr-x 2 hulidex hulidex 4096 oct  9 14:34 Documents
drwxr-xr-x 2 hulidex hulidex 4096 oct  9 14:34 Music
drwxr-xr-x 2 hulidex hulidex 4096 oct  9 14:34 Public
drwxr-xr-x 2 hulidex hulidex 4096 oct  9 14:34 Templates
```

Mi solución a la **actividad 3.9** ha sido:

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.

```
hulidex@GE60-2PC:~$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
```

```
udev            6091136       0   6091136   0% /dev
tmpfs           1222260    9680   1212580   1% /run
/dev/sdb3     111073072 10389524  95018320  10% /
tmpfs           6111284   86884   6024400   2% /dev/shm
tmpfs              5120       4      5116   1% /run/lock
tmpfs           6111284       0   6111284   0% /sys/fs/cgroup
/dev/sdb1        238938    3547    235391   2% /boot/efi
/dev/sda1     123723748 21555848  95860060  19% /home
tmpfs           1222260      60   1222200   1% /run/user/1000
/dev/sda2      52428796   99132  52329664   1% /media/hulidex/Files
```

Como se puede ver la partición raíz (expresada por /) en mi computador
usa un numero total de 10389524 bloques.

Para expresar esta capacidad en una unidad de medida entendible por un humano:

```
hulidex@GE60-2PC:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            5,9G     0  5,9G   0% /dev
tmpfs           1,2G  9,5M  1,2G   1% /run
/dev/sdb3       106G   10G   91G  10% /
tmpfs           5,9G   85M  5,8G   2% /dev/shm
tmpfs           5,0M  4,0K  5,0M   1% /run/lock
tmpfs           5,9G     0  5,9G   0% /sys/fs/cgroup
/dev/sdb1       234M  3,5M  230M   2% /boot/efi
/dev/sda1       118G   21G   92G  19% /home
tmpfs           1,2G   60K  1,2G   1% /run/user/1000
/dev/sda2        50G   97M   50G   1% /media/hulidex/Files
```

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

crear en esta partición?

```
hulidex@GE60-2PC:~$ df -i
Filesystem      Inodes  IUsed   IFree IUse% Mounted on
udev           1522784    633  1522151    1% /dev
tmpfs          1527821    865  1526956    1% /run
/dev/sdb3      7061504 332415  6729089    5% /
tmpfs          1527821     37  1527784    1% /dev/shm
tmpfs          1527821      5  1527816    1% /run/lock
tmpfs          1527821     16  1527805    1% /sys/fs/cgroup
/dev/sdb1            0      0        0     - /boot/efi
/dev/sda1      7864320  27889  7836431    1% /home
tmpfs          1527821     30  1527791    1% /run/user/1000
/dev/sda2     52395200     36 52395164    1% /media/hulidex/Files
```

Fijandonos en mi computador, actualmente en la partición raíz se están usando 332415 inodos, se podrían crean tantos archivos como inodos libres es decir 6729089 archivos podrían crearse.

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.

```
hulidex@GE60-2PC:~$ sudo du /etc/ -sh
14M    /etc/
```

En mi caso mi directorio /etc tiene 14 Megabytes

```
hulidex@GE60-2PC:~$ sudo du /var -sh
1,2G    /var
```

El direcotorio /var tiene 1,2 Gygabytes

Comparemos tamaños:

hulidex@GE60-2PC:~$ sudo du /etc /var /bin /usr /lib -sh

14M /etc

1,2G /var

13M /bin

5,5G /usr

879M /lib

tanto /etc como /bin tienen tamaños similares y además los más pequeños esto se puede deber a que los archivos que contienen no son de gran tamaño, como ya se

vio en otras secciones el archivo /etc contiene ficheros de configuranción y el /bin aplicaciones binarias de importancia.

/usr es mucho mas pesada que las demás porque contiene todas las aplicaciones a las que pueden acceder la mayoría de usuarios del sistema

/var contiene archivos que varían, como registros y bases de datos

y /lib contiene todas las librerías del 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?

hulidex@GE60-2PC:~$ sudo du /etc/ --block-size=4k -s

3572 /etc/

Tiene 3572 bloques donde cada uno de esos bloques ocupan 4k.

El tamaño por omisión de un bloque es de 1k.

Mi solución a la **actividad 3.10** ha sido:

Ordenes usadas para reproducir el ejemplo monstrado

```
[root@localhost ~]# touch archivo.txt target_hardLink2.txt
[root@localhost ~]# ln  archivo.txt hardlink
[root@localhost ~]# ln -s archivo.txt  softlink
[root@localhost ~]# ln target_hardLink2.txt hardlink2
[root@localhost ~]# ls
archivo.txt  hardlink  hardlink2  softlink  target_hardLink2.txt
[root@localhost ~]# ls -lai
total 32
  311 dr-xr-x---  2 root root 4096 Oct 29 15:58 .
    2 dr-xr-xr-x 22 root root 4096 Oct 29 14:05 ..
   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
14248 -rw-r--r--  2 root root    0 Oct 29 15:54 archivo.txt
14248 -rw-r--r--  2 root root    0 Oct 29 15:54 hardlink
14249 -rw-r--r--  2 root root    0 Oct 29 15:54 hardlink2
14251 lrwxrwxrwx  1 root root   11 Oct 29 15:58 softlink -> archivo.txt
14249 -rw-r--r--  2 root root    0 Oct 29 15:54 target_hardLink2.txt
```

COmo se puede ver tanto hardlink como archivo.txt comparten el mismo número de inodo 14248. Lo mismo ocurre con target_hardLink2.txt y hardlink2 que comparten el numero de inodo 14249

¿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?

de la misma forma que la mayoria de los archivos tienen como contador de enlaces

el numero 1, porque existe un enlace duro por defecto en todos los archivos que creemos en el SA ese enlace es el enlace duro que tienen sobre el inodo asociado a ellos. Por tanto:

El archivo archivo.txt tiene 2 enlaces duros, el creado por nosotros y el que tiene por defecto con su inodo asociado.

Mi solución a la **actividad 3.12** ha sido:

Ordenes usadas:

```
[root@localhost ~]# mknod Bloques b 10 5
[root@localhost ~]# mknod Caracteres c 10 5
[root@localhost ~]# ls -li
total 0
14253 brw-r--r-- 1 root root 10, 5 Oct 29 16:09 Bloques
14255 crw-r--r-- 1 root root 10, 5 Oct 29 16:10 Caracteres
14248 -rw-r--r-- 2 root root     0 Oct 29 15:54 archivo.txt
14248 -rw-r--r-- 2 root root     0 Oct 29 15:54 hardlink
14249 -rw-r--r-- 2 root root     0 Oct 29 15:54 hardlink2
14251 lrwxrwxrwx 1 root root    11 Oct 29 15:58 softlink -> archivo.txt
14249 -rw-r--r-- 2 root root     0 Oct 29 15:54 target_hardLink2.txt
```