

Ingeniería de Servidores (2014-2015)
GRADO EN INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE GRANADA

Memoria Práctica 3

Jesús Ángel González Novez

2 de diciembre de 2014

Índice

1. ¿Qué archivo le permite ver qué programas se han instalado con el gestor de paquetes? ¿Qué significan las terminaciones .1.gz o .2.gz de los archivos en ese directorio? 3
2. ¿Qué archivo ha de modificar para programar una tarea? Escriba la línea necesaria para ejecutar una vez al día una copia del directorio /codigo a /seguridad/\$fecha donde \$fecha es la fecha actual (puede usar el comando date). 4
3. Comando DMSG. Pruebe a ejecutar el comando, conectar un dispositivo USB y vuelva a ejecutar el comando. Copie y pegue la salida del comando. (considere usar dmesg | tail). Comente qué observa en la información mostrada. 5
4. Ejecute el monitor de "System Performance" y muestre el resultado. Incluya capturas de pantalla comentando la información que aparece. 6
5. Cree un recopilador de datos definido por el usuario (modo avanzado) que incluya tanto el contador de rendimiento como los datos de seguimiento: Todos los referentes al procesador, al proceso y al servicio web. Intervalo de muestra 15 segundos Almacene el resultado en el directorio Escritorio/logs Incluya las capturas de pantalla de cada paso. 7
6. Para Linux:
Para la temperatura del HD tenemos: hddtemp
Proyecto lm-sensors: <http://lm-sensors.org/> que posee una GUI: xsensors
Para Windows:
Open Hardware Monitor: <http://openhardwaremonitor.org/> (también funciona bajo Linux aunque depende de Mono http://www.mono-project.com/Main_Page)
SpeedFan: <http://www.almico.com/speedfan.php>
RealTemp: <http://www.techpowerup.com/realtemp/>
Core Temp: www.alcpu.com/CoreTemp/
CPUID: <http://www.cpuid.com/softwares/hwmonitor.html>
Speccy: <http://www.piriform.com/speccy>
Instale alguno de los monitores comentados arriba en su máquina y pruebe a ejecutarlos (tenga en cuenta que si lo hace en la máquina virtual, los resultados pueden no ser realistas). Alternativamente, busque otros monitores para hardware comerciales o de código abierto para Windows y Linux. 10

7. Visite la web del proyecto y acceda a la demo que proporcionan (http://demo.munin-monitoring.org/) donde se muestra cómo monitorizan un servidor. Monitorice varios parámetros y haga capturas de pantalla de lo que está mostrando comentando qué observa.	14
8. Escriba un breve resumen sobre alguno de los artículos donde se muestra el uso de strace o busque otro y coméntelo.	15
9. Acceda a la consola mysql (o a través de phpMyAdmin) y muestre el resultado de mostrar el "profile" de una consulta (la creación de la BD y la consulta la puede hacer libremente).	16
10. Cuestiones Opcionales	17
10.1. Indique qué comandos ha utilizado para realizarlo así como capturas de pantalla del proceso de reconstrucción del RAID.	17
10.2. Escriba un script en python y analice su comportamiento usando el profiler presentado.	17
10.3. Al igual que ha realizado el "profiling" con MySQL, realice lo mismo con MongoDB y compare los resultados (use la misma información y la misma consulta, hay traductores de consultas SQL a Mongo).	18

Índice de figuras

4.1. Resultado 'System Performance'	6
5.1. Elección de los referentes procesador, proceso y servicio web	7
5.2. Configuración del intervalo de muestra(15s)	8
5.3. Estableciendo el escritorio como destino.	8
5.4. Recopiladores en ejecución.	9
5.5. Resultados finales.	9
6.1. GUI de xsensors	10
6.2. Comando top en ejecución	11
6.3. Comando htop en ejecución	11
6.4. Comando nmon en ejecución	12
6.5. SpeedFan mostrando temperatura del procesador.	13
6.6. RealTemp mostrando temperatura del procesador.	13
7.1. Gráfica sobre usuarios online y logins en un año	14
7.2. Gráfica sobre interacciones con el repositorio GitHub en un año	14
7.3. Gráfica sobre porcentajes de uso del disco en un año	15
8.1. Ejecución del comando 'strace clear'	16

1. ¿Qué archivo le permite ver qué programas se han instalado con el gestor de paquetes? ¿Qué significan las terminaciones .1.gz o .2.gz de los archivos en ese directorio?

Normalmente todos los logs de Linux suelen situarse en `/var/log/`, por tanto era de esperar que todas las acciones realizadas con 'apt-get' son registradas en logs. Estos archivos están ubicados en el directorio `/var/log/apt/`

En este directorio tenemos varios archivos útiles. Para ver el historial más reciente podemos ejecutar:

```
less /var/log/apt/history.log
```

Este registro se va actualizando, los registros antiguos se van comprimiendo en el fichero 'history.log.1.gz', podemos echar un vistazo a su contenido ejecutando:

```
zless /var/log/apt/history.log.1.gz
```

Para ver los logs disponibles podemos usar:

```
ls -la /var/log/apt
```

Por otro lado podemos hacernos un script en bash que nos muestre historiales de instalaciones, actualizaciones y desinstalaciones usando el log de dpkg¹.

```
case "$1" in
    install)
        cat /var/log/dpkg.log | grep 'install '
        ;;
    upgrade|remove)
        cat /var/log/dpkg.log | grep $1
        ;;
    rollback)
        cat /var/log/dpkg.log | grep upgrade | \
            grep "$2" -A1000000 | \
            grep "$3" -B1000000 | \
            awk '{print $4="$5}'
        ;;
    *)
        cat /var/log/dpkg.log
        ;;
esac
```

¹<https://help.ubuntu.com/community/ListInstalledPackagesByDate>

De esta forma podemos ejecutar lo siguiente:

```
bash script.sh install # muestra instalados
o
bash script.sh upgrade # muestra actualizados
o
bash script.sh remove # muestra removidos
o
bash script.sh * # muestra todo el log
```

2. **¿Qué archivo ha de modificar para programar una tarea? Escriba la línea necesaria para ejecutar una vez al día una copia del directorio /codigo a /seguridad/\$fecha donde \$fecha es la fecha actual (puede usar el comando date).**

Creamos este script:

```
copiar.sh
#!/bin/bash
date
var = $?
mkdir ~/seguridad/$var
cp ~/codigo ~/seguridad/$var
```

Le damos permisos adecuados:

```
chmod a+x ~/scripts/copiar.sh
```

Fijaremos la hora de la tarea a las 12:00, creamos un archivo cron² con este contenido:

```
0 12 * * * usuario ~/scripts/copiar.sh
```

Ya solo queda añadir esa tarea al cron:

```
crontab archivo
```

²Visto en asignaturas varias a lo largo de la carrera como son Fundamentos de Software, Sistemas Operativos.

3. **Comando DMESG. Pruebe a ejecutar el comando, conectar un dispositivo USB y vuelva a ejecutar el comando. Copie y pegue la salida del comando. (considere usar `dmesg | tail`). Comente qué observa en la información mostrada.**

Sin dispositivo usb:

```
$dmesg | tail -3
[ 1725.460800] sd 4:0:0:0: [sdb] Test WP failed, assume Write Enabled
[ 1725.462917] sd 4:0:0:0: [sdb] Asking for cache data failed
[ 1725.462922] sd 4:0:0:0: [sdb] Assuming drive cache: write through
```

Conectamos ratón óptico usb:

```
$dmesg | tail -3
[ 1863.924059] usb 6-2: new low-speed USB device number 3 using uhci_hcd
[ 1864.114643] input: USB Optical Mouse as /devices/pci0000:00/0000:00:1d.0/
usb6/6-2/6-2:1.0/input/input11
[ 1864.114897] generic-usb 0003:1BCF:0007.0003: input,hiddev0,hidraw0: USB HID v1.10
Mouse [USB Optical Mouse] on usb-0000:00:1d.0-2/input0
```

Desconectamos ratón óptico usb:

```
$dmesg | tail -3
[ 1932.311208] sd 4:0:0:0: [sdb] Asking for cache data failed
[ 1932.311214] sd 4:0:0:0: [sdb] Assuming drive cache: write through
[ 1948.440076] usb 6-2: USB disconnect, device number 3
```

4. Ejecute el monitor de “System Performance” y muestre el resultado. Incluya capturas de pantalla comentando la información que aparece.

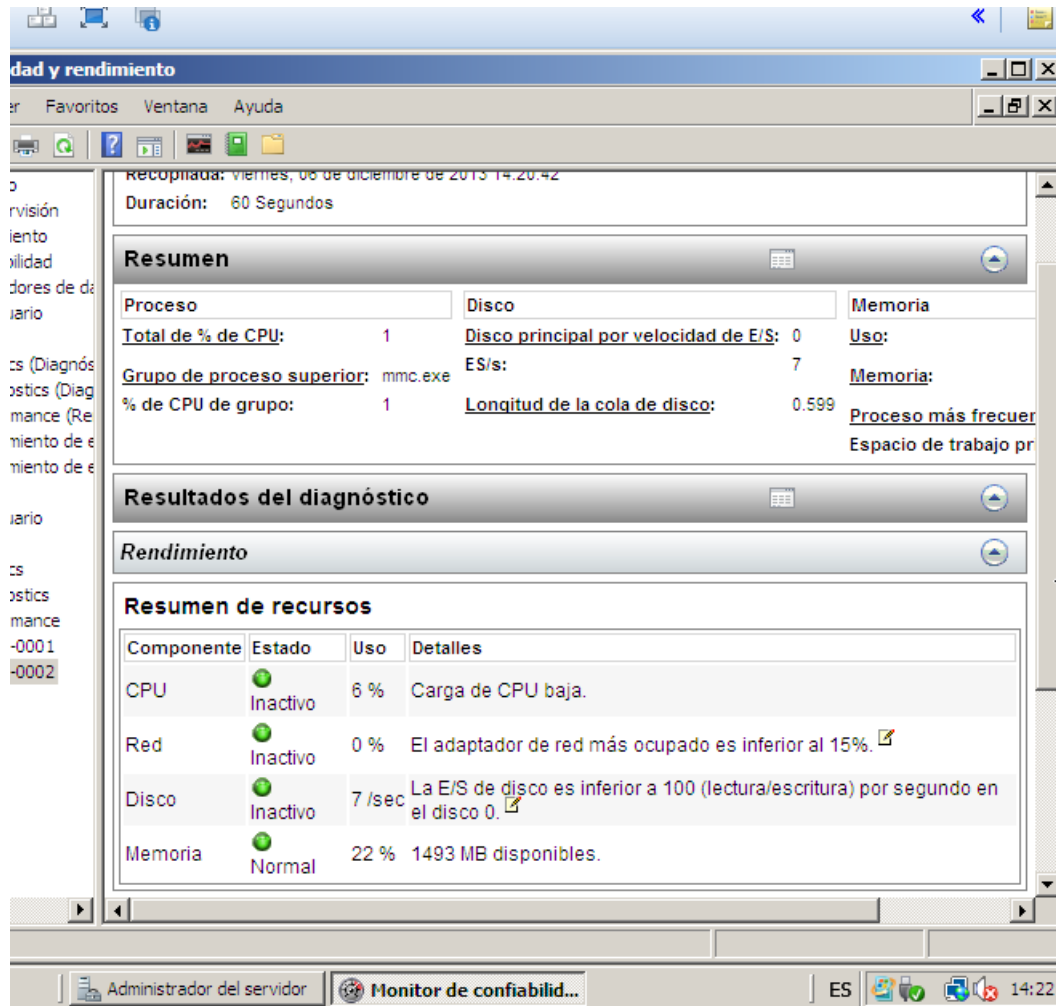


Figura 4.1: Resultado 'System Performance'

Podemos observar una carga de CPU baja(6%), la red está sin uso(no se está usando ningún software que necesite red en ese momento), el disco duro está tranquilo(casi inactivo), y la memoria RAM está usada al 22%, lo cual es la carga normal del propio sistema operativo así como el par de programas en ejecución en ese momento.

5. Cree un recopilador de datos definido por el usuario (modo avanzado) que incluya tanto el contador de rendimiento como los datos de seguimiento: Todos los referentes al procesador, al proceso y al servicio web. Intervalo de muestra 15 segundos Almacene el resultado en el directorio Escritorio/logs Incluya las capturas de pantalla de cada paso.

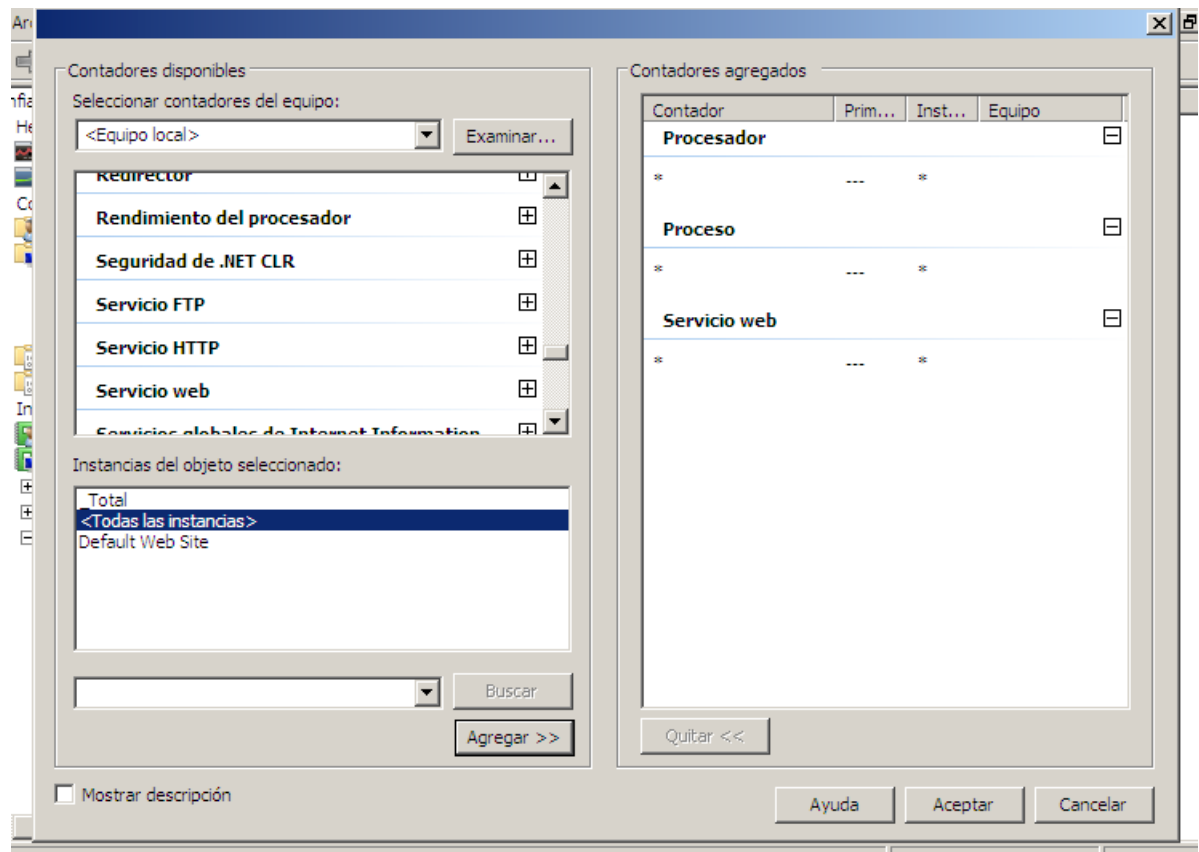


Figura 5.1: Elección de los referentes procesador, proceso y servicio web

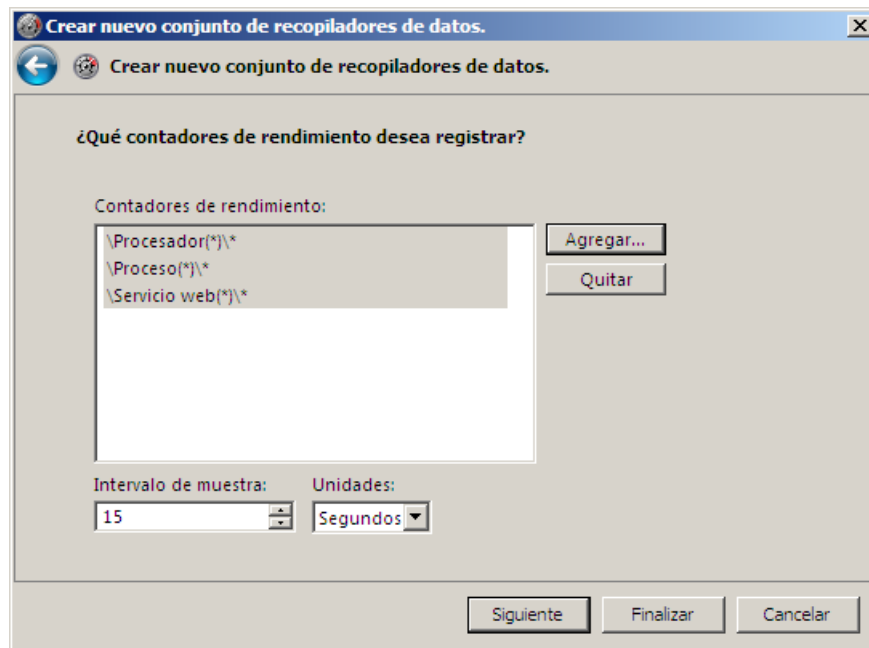


Figura 5.2: Configuración del intervalo de muestra(15s)

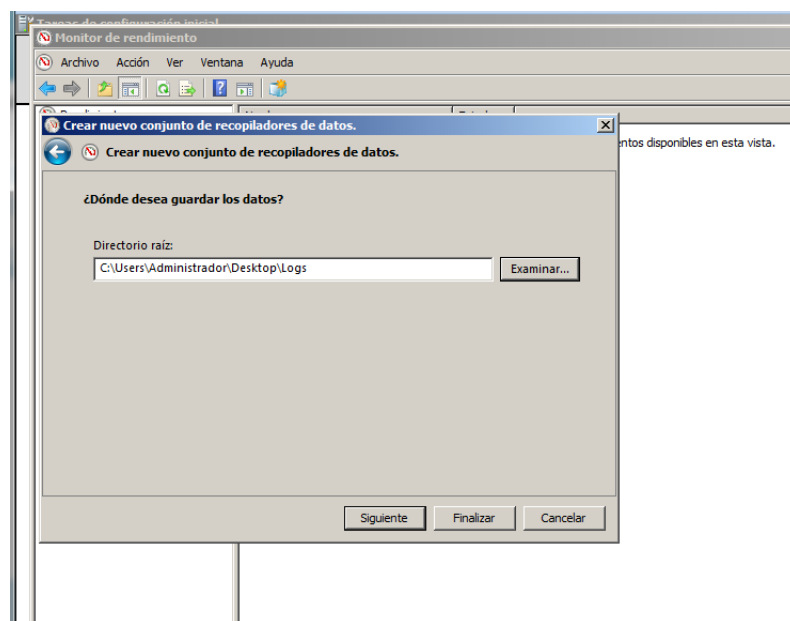


Figura 5.3: Estableciendo el escritorio como destino.

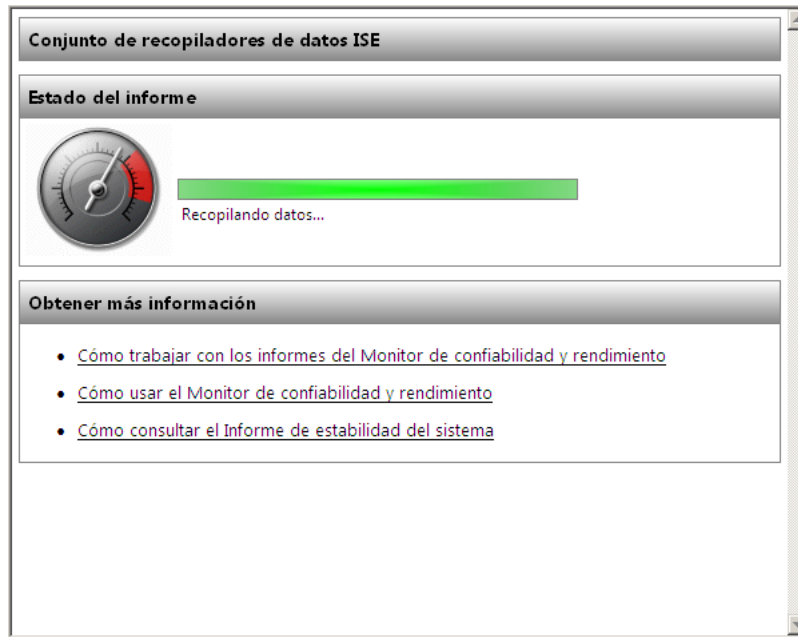


Figura 5.4: Recopiladores en ejecución.

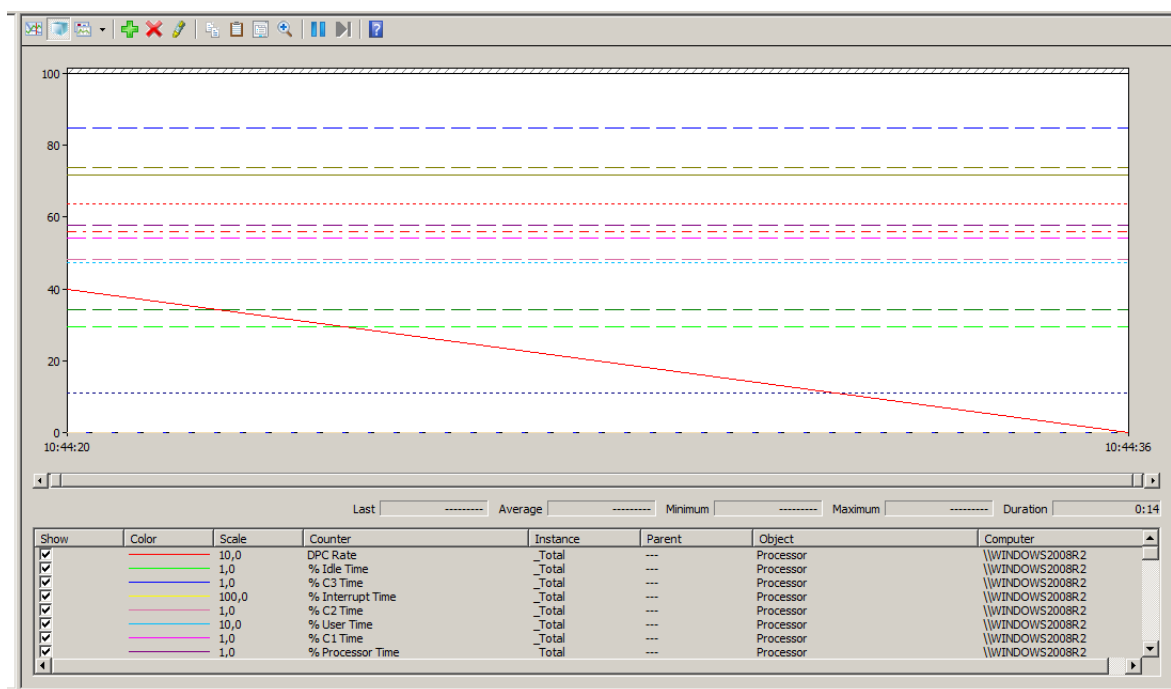


Figura 5.5: Resultados finales.

6. Para Linux:

Para la temperatura del HD tenemos: hddtemp

Proyecto lm-sensors: <http://lm-sensors.org/> que posee una GUI: xsensors

Para Windows:

Open Hardware Monitor: <http://openhardwaremonitor.org/>
(también funciona bajo Linux aunque depende de Mono
http://www.mono-project.com/Main_Page)

SpeedFan: <http://www.almico.com/speedfan.php>

RealTemp: <http://www.techpowerup.com/realtemp/>

Core Temp: www.alcpu.com/CoreTemp/

CPUID: <http://www.cpuid.com/softwares/hwmonitor.html>

Speccy: <http://www.piriform.com/speccy>

Instale alguno de los monitores comentados arriba en su máquina y pruebe a ejecutarlos (tenga en cuenta que si lo hace en la máquina virtual, los resultados pueden no ser realistas). Alternativamente, busque otros monitores para hardware comerciales o de código abierto para Windows y Linux.

Linux

```
jesus@jesuspc:~\$ sudo hddtemp /dev/sda  
/dev/sda: Hitachi HTS543216L9A300: 34°C
```

```
jesus@jesuspc:~\$ xsensors
```

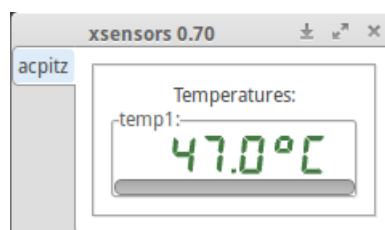


Figura 6.1: GUI de xsensors

Linux extra

A continuación añado algunos monitores útiles para plataformas Linux, que no han sido nombrados en el enunciado de la cuestión pero que a mi parecer se merecen una breve descripción.

Los monitores en cuestión son:

```
jesus@jesuspc:~\$ top
```

```
top - 16:24:56 up 48 min, 4 users, load average: 0.12, 0.20, 0.26
Tasks: 183 total, 1 running, 182 sleeping, 0 stopped, 0 zombie
Cpu(s): 13.0%us, 6.6%sy, 0.2%ni, 80.2%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 3977492k total, 3349968k used, 627524k free, 124772k buffers
Swap: 4502524k total, 0k used, 4502524k free, 2113308k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1067	root	20	0	169m	27m	9796	S	11	0.7	1:43.85	Xorg
10998	jesus	20	0	509m	20m	10m	S	7	0.5	0:00.92	gnome-screensho
4564	jesus	20	0	800m	69m	17m	S	7	1.8	0:35.30	gala
8257	jesus	20	0	1332m	52m	29m	S	5	1.4	1:05.71	vlc
4800	jesus	9	-11	416m	7528	4864	S	3	0.2	0:43.80	pulseaudio
4539	jesus	20	0	807m	23m	13m	S	1	0.6	0:02.40	gnome-settings-
1009	mongodb	20	0	623m	15m	4908	S	1	0.4	0:16.94	mongod
10996	jesus	20	0	17336	1372	968	R	1	0.0	0:00.04	top
10	root	20	0	0	0	0	S	0	0.0	0:00.36	ksoftirqd/1
1631	root	20	0	71984	4908	3940	S	0	0.1	0:02.41	teamviewerd
2571	root	35	15	22004	6264	960	S	0	0.2	0:02.78	preload
4693	jesus	20	0	20184	940	768	S	0	0.0	0:00.80	syndaemon
4988	jesus	20	0	1184m	17m	5616	S	0	0.4	0:14.03	python2
1	root	20	0	24572	2484	1376	S	0	0.1	0:01.04	init

Figura 6.2: Comando top en ejecución

```
jesus@jesuspc:~\$ htop
```

```
1  [|||||] 6.0% Tasks: 131, 274 thr; 1 running
2  [|||||] 7.3% Load average: 0.18 0.21 0.26
Mem [|||||] 1086/3884MB Uptime: 00:49:45
Swp [|||||] 0/4396MB
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
8257	jesus	20	0	1332M	53796	30656	S	4.0	1.4	1:09.26	/usr/bin/vlc /ho
10928	jesus	20	0	1332M	53796	30656	S	4.0	1.4	0:03.69	/usr/bin/vlc /ho
11035	jesus	20	0	26980	2080	1424	R	3.0	0.1	0:00.39	htop
1067	root	20	0	169M	28636	9796	S	1.0	0.7	1:46.71	/usr/bin/X :0 -a
4564	jesus	20	0	800M	71088	17952	S	1.0	1.8	0:36.64	gala
4800	jesus	9	-11	416M	7792	5128	S	1.0	0.2	0:45.52	/usr/bin/pulseau
4807	jesus	-6	-11	416M	7792	5128	S	1.0	0.2	0:29.65	/usr/bin/pulseau
11037	jesus	20	0	509M	20632	11000	S	0.0	0.5	0:00.88	gnome-screenshot
8274	jesus	20	0	1332M	53796	30656	S	0.0	1.4	0:06.35	/usr/bin/vlc /ho
4539	jesus	20	0	807M	24464	13928	S	0.0	0.6	0:02.49	/usr/lib/gnome-s
4834	jesus	20	0	602M	22028	11752	S	0.0	0.6	0:08.90	plank
5396	jesus	20	0	594M	21300	11784	S	0.0	0.5	0:05.35	pantheon-termina
4534	jesus	20	0	25400	2404	612	S	0.0	0.1	0:02.46	//bin/dbus-daemo

```
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice +F9Kill F10Quit
```

Figura 6.3: Comando htop en ejecución

```
jesus@jesuspc:~\$ nmon
```

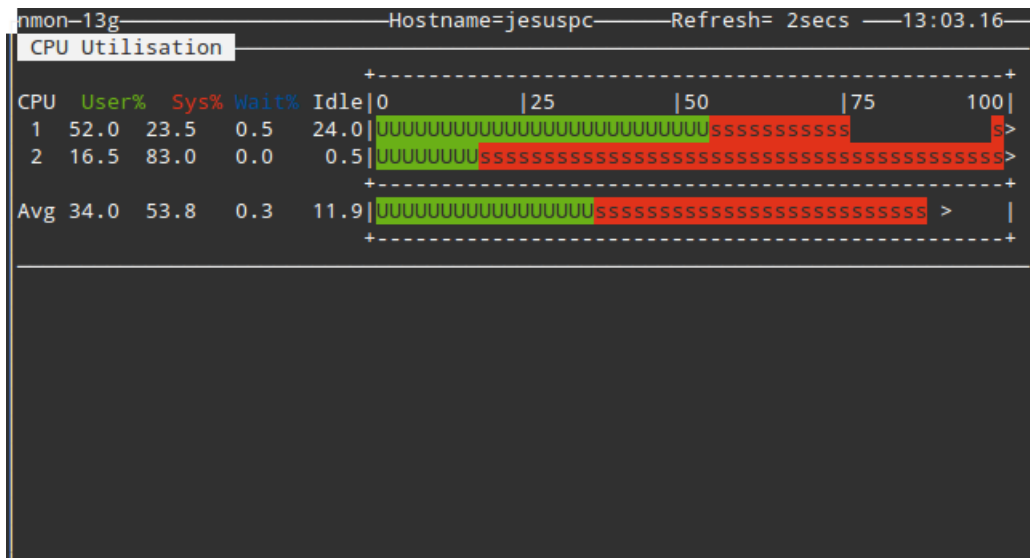


Figura 6.4: Comando nmon en ejecución

Windows

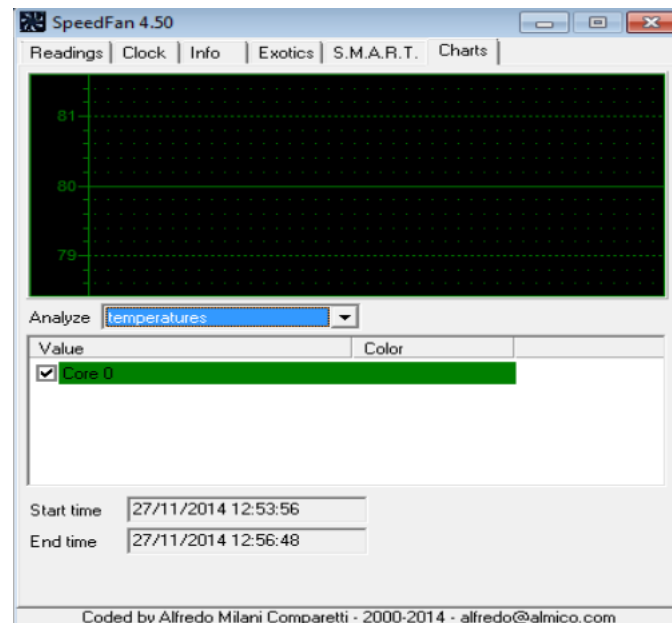


Figura 6.5: SpeedFan mostrando temperatura del procesador.

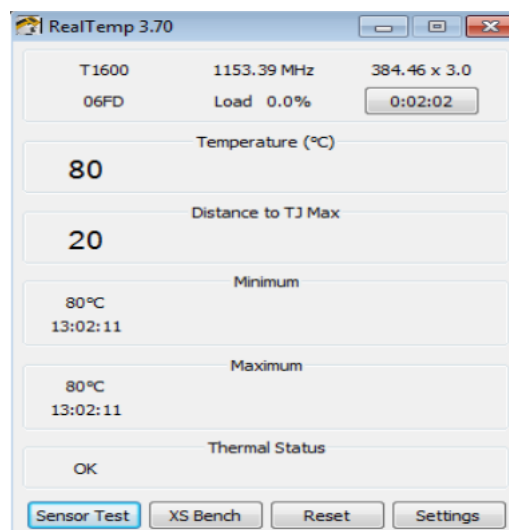


Figura 6.6: RealTemp mostrando temperatura del procesador.

7. Visite la web del proyecto y acceda a la demo que proporcionan (<http://demo.munin-monitoring.org/>) donde se muestra cómo monitorizan un servidor. Monitorice varios parámetros y haga capturas de pantalla de lo que está mostrando comentando qué observa.

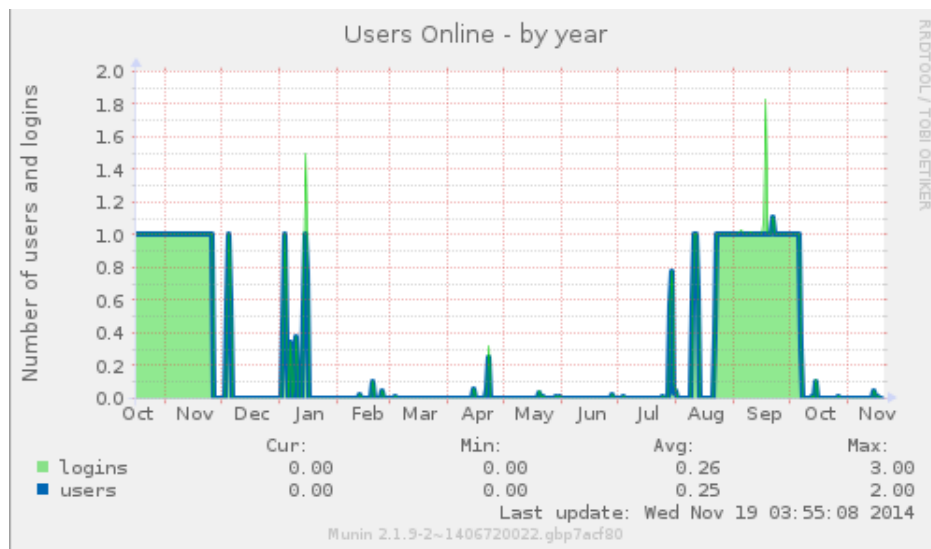


Figura 7.1: Gráfica sobre usuarios online y logins en un año

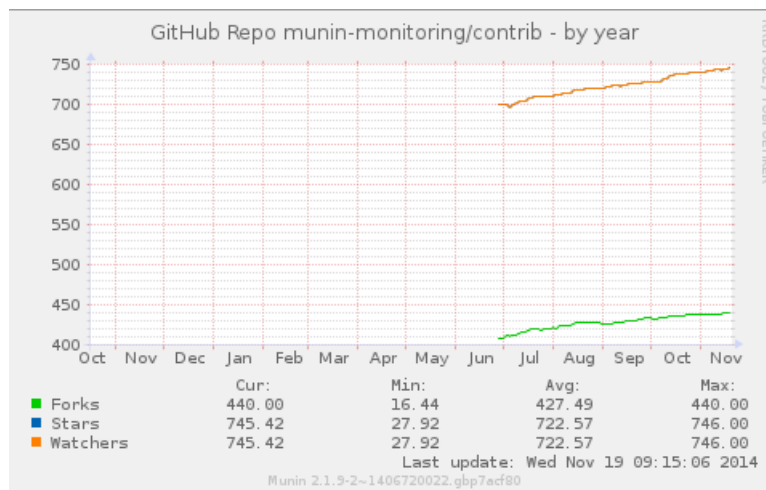


Figura 7.2: Gráfica sobre interacciones con el repositorio GitHub en un año

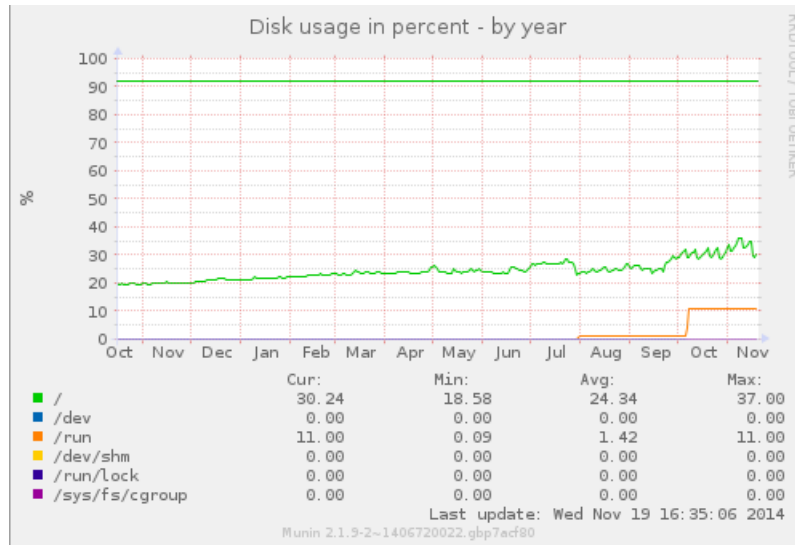


Figura 7.3: Gráfica sobre porcentajes de uso del disco en un año

8. Escriba un breve resumen sobre alguno de los artículos donde se muestra el uso de strace o busque otro y coméntelo.

Lo realizaré sobre el segundo³, me pareció más claro, aunque he leído ambos: strace es una herramienta que nos permite ver qué ocurre en situaciones que no encontramos errores de depuración aparentes, o no tenemos ficheros de log adecuados. Como el propio autor del artículo dice "no es un debugger ni una herramienta de programadores, es una herramienta de administradores de sistemas". Básicamente strace nos permitirá seguir la pista de las llamadas al sistema que realiza nuestro programa detalladamente y tal vez por ahí podamos ver que está haciendo nuestro programa realmente y contrastarlo con lo que pensábamos o deseábamos que hiciese.

Su funcionamiento es bien sencillo, delante de la orden que queramos ejecutar añadimos strace.

```
strace cat fichero.txt
strace clear
strace ./mi_programa
...
```

³<http://goo.gl/A6SQNe>

A continuación voy a mostrar un ejemplo sencillo usando el comando clear:

```
access("/lib/terminfo/x/xterm", R_OK) = 0
open("/lib/terminfo/x/xterm", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=3213, ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f
9f11f92000
read(3, "\32\1\34\0&\0\17\0\235\1Z\5xterm|X11 terminal e"... , 4096) = 3213
read(3, "", 4096) = 0
close(3) = 0
munmap(0x7f9f11f92000, 4096) = 0
ioctl(1, SNDCTL_TMR_TIMEBASE or TCGETS, {B38400 opost isig icanon echo ...}) =
0
ioctl(1, SNDCTL_TMR_TIMEBASE or TCGETS, {B38400 opost isig icanon echo ...}) =
0
ioctl(1, TIOCGWINSZ, {ws_row=21, ws_col=79, ws_xpixel=0, ws_ypixel=0}) = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 3), ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f
9f11f92000
write(1, "\33[H\33[2J", 7
) = 7
exit_group(0) = ?
```

Figura 8.1: Ejecución del comando 'strace clear'

La figura anterior sería más extensa pero solo he puesto la última parte, para el caso nos basta, a excepción de algunas llamadas como close() o similares, el resto, al menos personalmente están fuera de mis dominios.

9. Acceda a la consola mysql (o a través de phpMyAdmin) y muestre el resultado de mostrar el "profile" de una consulta (la creación de la BD y la consulta la puede hacer libremente).

<http://dev.mysql.com/doc/refman/5.0/en/show-profile.html> Primero accederemos a la base de datos 'datos_usuario'.

```
mysql -u root -p
use datos_usuario;
```

Ahora pondremos el 'profiler' en activo.

```
set profiling = 1;
```

Ahora realizaremos una inserción de una tupla en la tabla 'usuarios' y luego mostraremos todas las tuplas insertadas hasta ahora.

```
insert into usuarios values('id','Juan','juanlopez@gmail.com');
Query OK, 1 row affected, 1 warning (0.05 sec)
```

```

select * from usuarios;
+-----+-----+-----+
| id_usuario | name | email |
+-----+-----+-----+
|          1 | pepe | pepe@gmail.com |
|          2 | pepe2 | pepe2@gmail.com |
|          3 | juan | juan@gmail.com |
|          4 | juan2 | juan@gmail.com |
|          5 | Jaime | jaimelopez@gmail.com |
|          6 | Juan | jl@gmail.com |
+-----+-----+-----+
6 rows in set (0.00 sec)

```

Finalmente consultamos el "profiler" para ver cuanto tardó cada operación.

```

show profiles;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
|          1 | 0.04748150 | insert into usuarios values('id','Juan','jl@g.com') |
|          2 | 0.00048375 | select * from usuarios |
+-----+-----+-----+

```

En este caso fueron operaciones muy veloces dada la magnitud de la base de datos. Como vemos "mysql" de por sí nos da un tiempo de hasta dos decimales tras cada operación, pero vemos que el insert nos dijo 0.05s (lo redondeó) y vemos que para el select nos dijo que tardó 0.00s cuando realmente fue algo más.

10. Cuestiones Opcionales

10.1. Indique qué comandos ha utilizado para realizarlo así como capturas de pantalla del proceso de reconstrucción del RAID.

Dado que en la práctica 2 realicé el mismo proceso, el propio profesor (Alberto G.) me dijo que simplemente lo dijese y que se daba por respondida esta cuestión.

10.2. Escriba un script en python y analice su comportamiento usando el profiler presentado.

El script diseñado es bastante sencillo, consiste en un bucle que incrementa la variable j de uno en uno durante 999999 iteraciones. Para hacer el profile de ese for usaremos cProfile como se muestra en el mismo script.

```

import cProfile
class prueba:

```

```

def bucle(self):
    j = 1
    print("Valor inicial de j:"+str(j))
    for i in range(999999):
        j = j + 1
    print("Valor final de j:"+str(j))
p = prueba()
cProfile.run('p.bucle()')

```

El resultado que nos da por consola, aparte de la salida de los print, es la siguiente:

```

Valor inicial de j:1
Valor final de j:1000000
      4 function calls in 0.158 seconds

```

Ordered by: standard name

ncalls	totttime	percall	cumtime	percall	filename:lineno(function)
1	0.000	0.000	0.158	0.158	<string>:1(<module>)
1	0.112	0.112	0.158	0.158	script1.py:3(bucle)
1	0.000	0.000	0.000	0.000	{method 'disable' of '_lsprof.Profiler' objects}
1	0.046	0.046	0.046	0.046	{range}

10.3. Al igual que ha realizado el “profiling” con MySQL, realice lo mismo con MongoDB y compare los resultados (use la misma información y la misma consulta, hay traductores de consultas SQL a Mongo).

Bueno primero de todo crearemos una base datos llamada 'datos_usuario' y una colección llamada 'usuarios'.

```

>use datos_usuario
switched to db datos_usuario
>db.usuarios.save({name:'Juan',mail:'jl@gmail.com'})

```

Visitando <http://www.querymongo.com> traducimos una sentencia 'select * from usuarios'.

```

>db.usuarios.find();
{ "_id" : ObjectId("5475f06151ce882e732e6d4f"), "name" : "Juan",
  "mail" : "jl@gmail.com" }

```

En mongodb debemos indicar el nivel de profiling con valor 1 para consultas lentas y valor 2 para todas, como la nuestra será una consulta rápida usaremos el 2.

```
>db.setProfilingLevel(2)
```

Ahora realizaremos la consulta usando profiling.

```
>db.system.profile.find()
```

Todo lo referente a mongodb se puede encontrar en su documentación oficial⁴ aunque para desarrollar este ejercicio me he bastado de lo aprendido en una asignatura de 4º curso llamada DAI.

⁴<http://www.mongodb.org/>