

**INGENIERÍA DE SERVIDORES (2016-2017)**  
DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS  
UNIVERSIDAD DE GRANADA

---

## Memoria Práctica 3

---

Miguel Ángel Torres López

9 de mayo de 2017

## Índice

<b>1</b>	<b>El subsistema de archivos</b>	<b>4</b>
1.1	¿Qué archivo le permite ver qué programas se han instalado con el gestor de paquetes? . . . . .	4
1.2	¿Qué significan las terminaciones .1.gz o .2.gz de los archivos en ese directorio?	4
<b>2</b>	<b>Pruebe a ejecutar el comando dmesg, conectar un dispositivo USB y vuelva a ejecutar el comando. Copie y pegue la salida del comando. Comente qué observe en la información mostrada.</b>	<b>5</b>
<b>3</b>	<b>Ejecute el monitor de "<i>System Performance</i>" y muestre el resultado. Incluya capturas de pantalla comentando la información que aparece.</b>	<b>8</b>
<b>4</b>	<b>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 procesado y al servicio web, intervalo de muestra 5 segundos. Almacene el resultado en el directorio Escritorio/logs.</b>	<b>10</b>
<b>5</b>	<b>Visite la web del proyecto y acceda a la demo que proporcionan donde se muestra cómo monitorizan un servidor. Monitorice varios parámetros y haga capturas de pantalla de lo que está mostrando comentando qué observa.</b>	<b>13</b>
<b>6</b>	<b>Escribe un breve resumen sobre alguno de los artículos donde se muestra el uso de strace o busque otro y coméntelo.</b>	<b>15</b>
<b>7</b>	<b>Escriba un script en Python o PHP y analice su comportamiento usando el profiler presentado.</b>	<b>16</b>
<b>8</b>	<b>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).</b>	<b>17</b>
<b>9</b>	<b>(Cuestión opcional 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.</b>	<b>18</b>

## Índice de figuras

1.1. Archivos log de dpkg. . . . .	4
1.2. Archivos log de apt. . . . .	4
2.1. Filtrado de la información dada por dmesg. . . . .	5
3.1. Página inicial de perfmon. . . . .	8
3.2. Gráfica de monitorización de características. . . . .	9
3.3. Monitorizando varias características en la misma gráfica. . . . .	9
4.1. Creando un recopilador. Parte 1. . . . .	10
4.2. Creando un recopilador. Parte 2. . . . .	10
4.3. Creando un recopilador. Parte 3. . . . .	11
4.4. Creando un recopilador. Parte 4. . . . .	11
4.5. Creando un recopilador. Parte 5. . . . .	12
4.6. Creando un recopilador. Parte 6. . . . .	12
4.7. Creando un recopilador. Parte 7. . . . .	13
5.1. Threads ejecutándose en la máquina . . . . .	14
5.2. Procesos ejecutándose en la máquina . . . . .	15
7.1. Script Python de prueba. . . . .	16
7.2. Profile de un script en Python . . . . .	16
7.3. Salida del <i>profile</i> . . . . .	17
8.1. Consulta sobre una tabla. . . . .	17
8.2. Tabla del <i>profile</i> . . . . .	18
8.3. Diagrama circular del <i>profile</i> . . . . .	18
9.1. Línea del archivo crontab. . . . .	19
9.2. Script ejecutado por el cron. . . . .	19

## Índice de tablas

## 1. El subsistema de archivos

### 1.1. ¿Qué archivo le permite ver qué programas se han instalado con el gestor de paquetes?

Como se menciona la documentación de Debian[2] en el directorio `/var/log` se encuentran los archivos de tipo `log`, es decir, los archivos que guardan el historial de algunos datos del sistema y otros programas importantes. En nuestro caso nos interesa mirar dos archivos, ambos pueden verse en las Figuras 1.1 y 1.2. El archivo `/var/log/apt/history.log` guarda la información referente a antiguas ejecuciones de `apt` ordenadas por fecha. El otro archivo, `/var/log/dpkg.log` contiene información referente al comando `dpkg`, la herramienta básica para instalar paquetes.

```
miguelangel@torres@buntuserver:~$ cat /var/log/dpkg.log | tail -n10
2017-04-17 19:48:24 status unpacked phpmyadmin:all 4:4.5.4.1-2ubuntu2
2017-04-17 19:48:24 status half-configured phpmyadmin:all 4:4.5.4.1-2ubuntu2
2017-04-17 19:59:14 status installed phpmyadmin:all 4:4.5.4.1-2ubuntu2
2017-04-17 19:59:14 trigproc libc-bin:amd64 2.23-0ubuntu5 <ninguna>
2017-04-17 19:59:14 status half-configured libc-bin:amd64 2.23-0ubuntu5
2017-04-17 19:59:14 status installed libc-bin:amd64 2.23-0ubuntu5
2017-04-17 19:59:14 trigproc libapache2-mod-php7.0:amd64 7.0.15-0ubuntu0.16.04.4 <ninguna>
2017-04-17 19:59:14 status half-configured libapache2-mod-php7.0:amd64 7.0.15-0ubuntu0.16.04.4
2017-04-17 19:59:14 status installed libapache2-mod-php7.0:amd64 7.0.15-0ubuntu0.16.04.4
2017-04-17 19:59:14 startup packages configure
```

Figura 1.1: Archivos log de dpkg.

```
migtorlog@don abr 30 - 13:52:21:~$ cat /var/log/apt/history.log | tail
Start-Date: 2017-04-30 13:50:50
Commandline: /usr/bin/unattended-upgrade
Upgrade: libnss3-nssdb:amd64 (2:3.26.2-0ubuntu0.16.04.2, 2:3.28.4-0ubuntu0.16.04.1), libnss3:amd64 (2:3.26.2-0ubuntu0.16.04.2, 2:3.28.4-0ubuntu0.16.04.1), libnsspr4:amd64 (2:4.12-0ubuntu0.16.04.1, 2:4.13.1-0ubuntu0.16.04.1)
End-Date: 2017-04-30 13:50:51

Start-Date: 2017-04-30 13:52:18
Commandline: apt install tree
Requested-By: migtorlog (1000)
Install: tree:amd64 (1.7.0-3)
End-Date: 2017-04-30 13:52:19
```

Figura 1.2: Archivos log de apt.

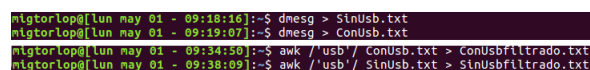
### 1.2. ¿Qué significan las terminaciones `.1.gz` o `.2.gz` de los archivos en ese directorio?

En ocasiones, los archivos `log` o archivos de historial pueden ocupar bastante espacio, más aun si estamos hablando de un servicio usado por terceros. Por este motivo, los archivos `log` suelen comprimirse y así ahorrar algo de espacio en disco. La terminación `.gz` hace referencia a la compresión con `gzip` y la numeración indica el orden de antigüedad. Como ejemplo de esta práctica, podemos ver el comando `logrotate`[6], útil cuando se quiere administrar de forma rápida y automática la compresión de archivos de historial.

## 2. Pruebe a ejecutar el comando `dmesg`, conectar un dispositivo USB y vuelva a ejecutar el comando. Copie y pegue la salida del comando. Comente qué observe en la información mostrada.

El comando `dmesg`[4] sin argumentos puede servirnos para ver información de un USB y de otras muchas características. Con el motivo de evitar grandes cantidades de texto ilegibles, tras ejecutar el comando `dmesg` con y sin USB conectado, he realizado un filtrado con el comando `awk` para que solo puedan verse las líneas en las que aparece la palabra `usb`, como puede verse en la Figura 2.1. Los dos ficheros de texto pueden verse a continuación de ésta. Podemos ver distintos dispositivos USB pero nos interesan principalmente las últimas líneas del archivo. Podemos ver que hay un dispositivo definido como `device 5` que corresponde con un móvil de Samsung. En la última línea se especifica que el dispositivo 5 está desconectado. De la misma forma, en el segundo archivo, realizado con el móvil conectado por USB, podemos ver los mismos datos del dispositivo 5. No obstante, esta vez en la última línea ya no está el comentario que se hacía antes por la desconexión del USB.

Cabe destacar que en el archivo en el que el móvil está desconectado el sistema operativo ya conoce el dispositivo móvil porque éste había sido conectado con anterioridad. En caso de no conocerlo, no se tendría registrada la información que aparece de éste.



```
migtorlop@lun may 01 - 09:18:16]:~$ dmesg > SinUsb.txt
migtorlop@lun may 01 - 09:19:07]:~$ dmesg > ConUsb.txt
migtorlop@lun may 01 - 09:34:50]:~$ awk '/usb/' ConUsb.txt > ConUsbFiltrado.txt
migtorlop@lun may 01 - 09:38:09]:~$ awk '/usb/' SinUsb.txt > SinUsbFiltrado.txt
```

Figura 2.1: Filtrado de la información dada por `dmesg`.

### SinUsbFiltrado.txt

```
[ 0.223866] usbcore: registered new interface driver usbfs
[ 0.223874] usbcore: registered new interface driver hub
[ 0.223889] usbcore: registered new device driver usb
[ 0.836243] usb usb1: New USB device found, idVendor=1d6b, idProduct=0002
[ 0.836246] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[ 0.836248] usb usb1: Product: EHCI Host Controller
[ 0.836259] usb usb1: Manufacturer: Linux 4.4.0-75-generic ehci_hcd
[ 0.836260] usb usb1: SerialNumber: 0000:00:1d.0
[ 0.838022] usb usb2: New USB device found, idVendor=1d6b, idProduct=0002
[ 0.838025] usb usb2: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[ 0.838027] usb usb2: Product: xHCI Host Controller
[ 0.838028] usb usb2: Manufacturer: Linux 4.4.0-75-generic xhci-hcd
[ 0.838030] usb usb2: SerialNumber: 0000:00:14.0
[ 0.839765] usb usb3: New USB device found, idVendor=1d6b, idProduct=0003
[ 0.839766] usb usb3: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[ 0.839767] usb usb3: Product: xHCI Host Controller
```

```

[ 0.839769] usb usb3: Manufacturer: Linux 4.4.0-75-generic xhci-hcd
[ 0.839770] usb usb3: SerialNumber: 0000:00:14.0
[ 0.840303] usb: port power management may be unreliable
[ 1.147926] usb 2-3: new low-speed USB device number 2 using xhci_hcd
[ 1.147928] usb 1-1: new high-speed USB device number 2 using ehci-pci
[ 1.279619] usb 2-3: New USB device found, idVendor=093a, idProduct=2510
[ 1.279622] usb 2-3: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 1.279624] usb 2-3: Product: USB Optical Mouse
[ 1.279625] usb 2-3: Manufacturer: PixArt
[ 1.279795] usb 2-3: ep 0x81 - rounding interval to 64 microframes
[ 1.280270] usb 1-1: New USB device found, idVendor=8087, idProduct=8001
[ 1.280272] usb 1-1: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[ 1.395975] usb 2-5: new full-speed USB device number 3 using xhci_hcd
[ 1.580909] usb 2-5: New USB device found, idVendor=8087, idProduct=07dc
[ 1.580912] usb 2-5: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[ 1.747927] usb 2-6: new high-speed USB device number 4 using xhci_hcd
[ 1.977798] usb 2-6: New USB device found, idVendor=13d3, idProduct=5652
[ 1.977801] usb 2-6: New USB device strings: Mfr=3, Product=1, SerialNumber=2
[ 1.977803] usb 2-6: Product: TOSHIBA Web Camera - HD
[ 1.977805] usb 2-6: Manufacturer: Azurewave
[ 1.977806] usb 2-6: SerialNumber: NULL
[ 1.985006] usbcore: registered new interface driver usbhid
[ 1.985008] usbhid: USB HID core driver
[ 1.986362] input: PixArt USB Mouse as
/devices/pci0000:00/0000:00:14.0/usb2/2-3/2-3:1.0/0003:093A:2510.0001/input/input11
[ 1.986497] hid-generic 0003:093A:2510.0001: input,hidraw0:
USB HID v1.11 Mouse [PixArt USB Mouse] on usb-0000:00:14.0-3/input0
[ 3.085368] toshiba_acpi: Supported laptop features: hotkeys touchpad usb-sleep-charge
[ 3.204720] usbcore: registered new interface driver btusb
[ 3.208107] input: TOSHIBA Web Camera - HD as
/devices/pci0000:00/0000:00:14.0/usb2/2-6/2-6:1.0/input/input17
[ 3.208173] usbcore: registered new interface driver uvcvideo
[ 3.318285] usb 2-5: USB disconnect, device number 3
[ 990.006691] usb 2-1: new high-speed USB device number 5 using xhci_hcd
[ 990.136512] usb 2-1: New USB device found, idVendor=04e8, idProduct=6860
[ 990.136515] usb 2-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 990.136517] usb 2-1: Product: SAMSUNG_Android
[ 990.136518] usb 2-1: Manufacturer: SAMSUNG
[ 990.136520] usb 2-1: SerialNumber: 3444c25d
[ 1163.377557] usb 2-1: USB disconnect, device number 5

```

### ConUsbFiltrado.txt

```

[ 0.223866] usbcore: registered new interface driver usbfs

```

```
[ 0.223874] usbcore: registered new interface driver hub
[ 0.223889] usbcore: registered new device driver usb
[ 0.836243] usb usb1: New USB device found, idVendor=1d6b, idProduct=0002
[ 0.836246] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[ 0.836248] usb usb1: Product: EHCI Host Controller
[ 0.836259] usb usb1: Manufacturer: Linux 4.4.0-75-generic ehci_hcd
[ 0.836260] usb usb1: SerialNumber: 0000:00:1d.0
[ 0.838022] usb usb2: New USB device found, idVendor=1d6b, idProduct=0002
[ 0.838025] usb usb2: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[ 0.838027] usb usb2: Product: xHCI Host Controller
[ 0.838028] usb usb2: Manufacturer: Linux 4.4.0-75-generic xhci-hcd
[ 0.838030] usb usb2: SerialNumber: 0000:00:14.0
[ 0.839765] usb usb3: New USB device found, idVendor=1d6b, idProduct=0003
[ 0.839766] usb usb3: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[ 0.839767] usb usb3: Product: xHCI Host Controller
[ 0.839769] usb usb3: Manufacturer: Linux 4.4.0-75-generic xhci-hcd
[ 0.839770] usb usb3: SerialNumber: 0000:00:14.0
[ 0.840303] usb: port power management may be unreliable
[ 1.147926] usb 2-3: new low-speed USB device number 2 using xhci_hcd
[ 1.147928] usb 1-1: new high-speed USB device number 2 using ehci-pci
[ 1.279619] usb 2-3: New USB device found, idVendor=093a, idProduct=2510
[ 1.279622] usb 2-3: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 1.279624] usb 2-3: Product: USB Optical Mouse
[ 1.279625] usb 2-3: Manufacturer: PixArt
[ 1.279795] usb 2-3: ep 0x81 - rounding interval to 64 microframes
[ 1.280270] usb 1-1: New USB device found, idVendor=8087, idProduct=8001
[ 1.280272] usb 1-1: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[ 1.395975] usb 2-5: new full-speed USB device number 3 using xhci_hcd
[ 1.580909] usb 2-5: New USB device found, idVendor=8087, idProduct=07dc
[ 1.580912] usb 2-5: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[ 1.747927] usb 2-6: new high-speed USB device number 4 using xhci_hcd
[ 1.977798] usb 2-6: New USB device found, idVendor=13d3, idProduct=5652
[ 1.977801] usb 2-6: New USB device strings: Mfr=3, Product=1, SerialNumber=2
[ 1.977803] usb 2-6: Product: TOSHIBA Web Camera - HD
[ 1.977805] usb 2-6: Manufacturer: Azurewave
[ 1.977806] usb 2-6: SerialNumber: NULL
[ 1.985006] usbcore: registered new interface driver usbhid
[ 1.985008] usbhid: USB HID core driver
[ 1.986362] input: PixArt USB Mouse as
/devices/pci0000:00/0000:00:14.0/usb2/2-3/2-3:1.0/0003:093A:2510.0001/input/input11
[ 1.986497] hid-generic 0003:093A:2510.0001: input,hidraw0:
USB HID v1.11 Mouse [PixArt USB Mouse] on usb-0000:00:14.0-3/input0
[ 3.085368] toshiba_acpi: Supported laptop features: hotkeys touchpad usb-sleep-charge
[ 3.204720] usbcore: registered new interface driver btusb
```

```
[ 3.208107] input: TOSHIBA Web Camera - HD as
/devices/pci0000:00/0000:00:14.0/usb2/2-6/2-6:1.0/input/input17
[ 3.208173] usbcore: registered new interface driver uvcvideo
[ 3.318285] usb 2-5: USB disconnect, device number 3
[ 990.006691] usb 2-1: new high-speed USB device number 5 using xhci_hcd
[ 990.136512] usb 2-1: New USB device found, idVendor=04e8, idProduct=6860
[ 990.136515] usb 2-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 990.136517] usb 2-1: Product: SAMSUNG_Android
[ 990.136518] usb 2-1: Manufacturer: SAMSUNG
[ 990.136520] usb 2-1: SerialNumber: 3444c25d
```

### 3. Ejecute el monitor de "System Perfomance" y muestre el resultado. Incluya capturas de pantalla comentando la información que aparece.

Abriendo la consola de PowerShell y ejecutando el comando *perfmon* se abre el monitor de "System Perfomance". Se mostrará una página de inicio con información básica y varios enlaces a documentación y ayuda, como puede verse en la Figura 3.1.

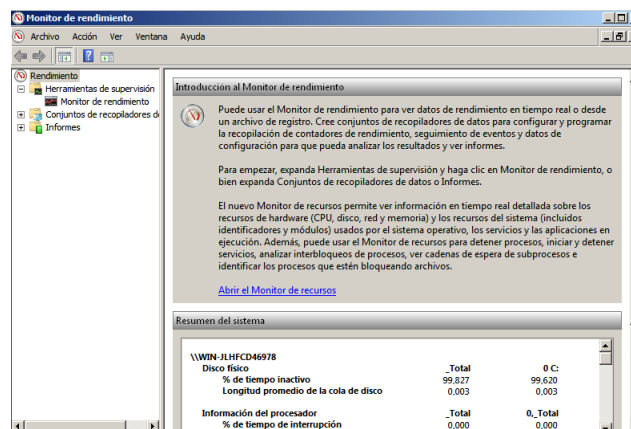


Figura 3.1: Página inicial de perfmon.

Se puede navegar un poco por la interfaz gráfica del monitor y, en la pestaña de monitor de rendimiento, observamos una gráfica en la que se puede hacer seguimiento en el tiempo de distintas características del sistema. Por defecto tenemos un histograma que mide el porcentaje de CPU consumida, además tenemos datos del promedio en los últimos minutos de esta característica. Se puede ver esta ventana en la Figura 3.2. Observamos que en mitad de la gráfica hay un pico de subida de CPU usada. Este pico corresponde a la ejecución del buscador de internet, podemos ver cómo al cerrar el buscador, la gráfica baja otra vez a los datos anteriores.



Para mostrar un ejemplo de su uso, añadiremos una característica más para inspeccionar. Al pulsar el icono de *más*, se nos abre una ventana en la que podremos elegir distintas características para añadir a inspección. Elegiremos, por ejemplo, añadir a la gráfica el porcentaje de bytes usados en memoria y, para poder compararlo bien con la característica anterior, activaremos los dos seguimientos en el desplegable de abajo y cambiaremos el modo gráfico a gráfica de barras. Los colores de cada característica se puede cambiar también. Tras hacer estos cambios, observamos que en nuestro monitor aparece algo parecido a la Figura 3.3

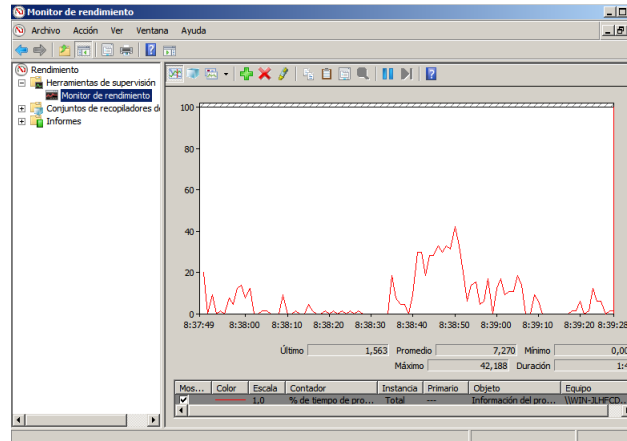


Figura 3.2: Gráfica de monitorización de características.

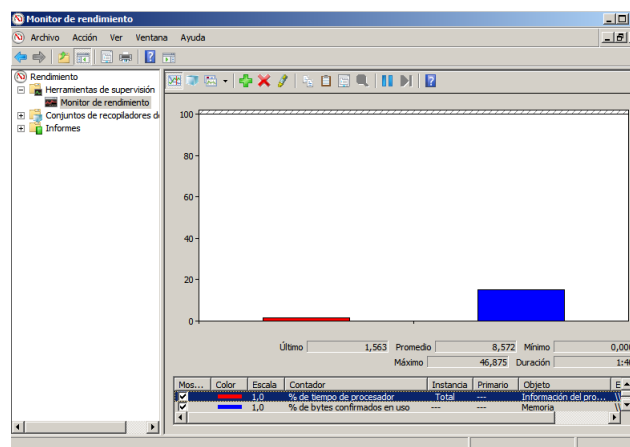


Figura 3.3: Monitorizando varias características en la misma gráfica.

4. 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 procesado y al servicio web, intervalo de muestra 5 segundos. Almacene el resultado en el directorio Escritorio/logs.

En primer lugar nos vamos a la pestaña de recopiladores de datos de usuario que, en principio estará vacía. Hacemos click secundario y creamos un recopilador como se muestra en la Figura 4.1.

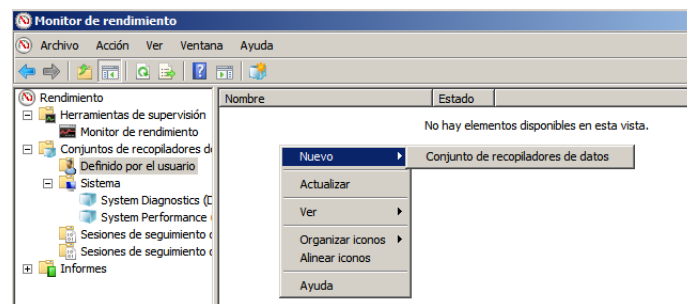


Figura 4.1: Creando un recopilador. Parte 1.

Nos aparecerán ahora una serie de ventanas que nos permitirán personalizar nuestro recopilador. Seleccionamos datos de seguimiento y contador de rendimiento en la primera ventana, como se muestra en la Figura 4.2

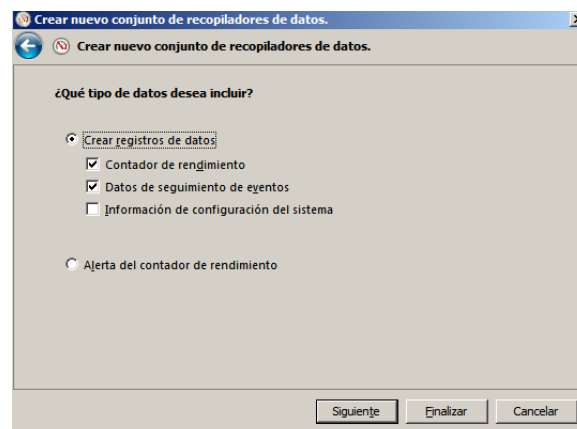


Figura 4.2: Creando un recopilador. Parte 2.

Ahora elegimos los datos a recopilar, en nuestro caso todos los referentes al procesado y al

servicio web.

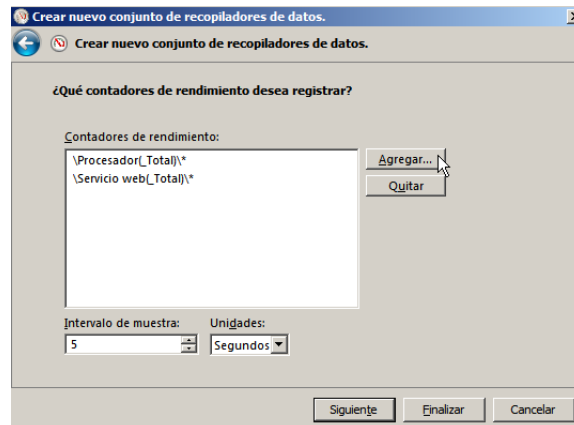


Figura 4.3: Creando un recopilador. Parte 3.

Por último, elegimos el lugar donde queremos guardar todos los datos. Como se especifica en el ejercicio, los guardaremos en el Escritorio en un carpeta logs.

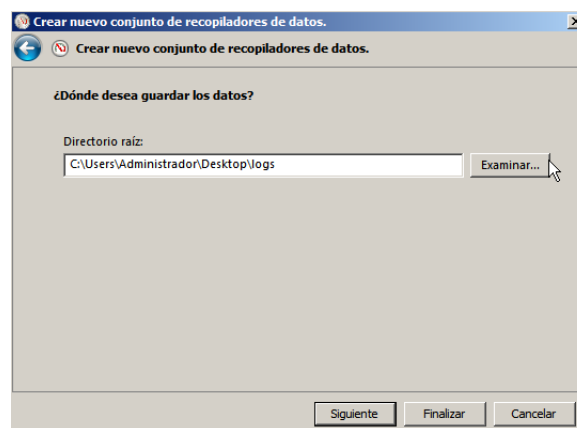


Figura 4.4: Creando un recopilador. Parte 4.

Como se puede ver en la Figura 4.5, el recopilador empieza por defecto desactivado. Para activarlo y que empiece a recoger información es necesario hacer click secundario sobre él e iniciarlo. En cualquier momento podemos cambiar sus preferencias en el mismo menú.

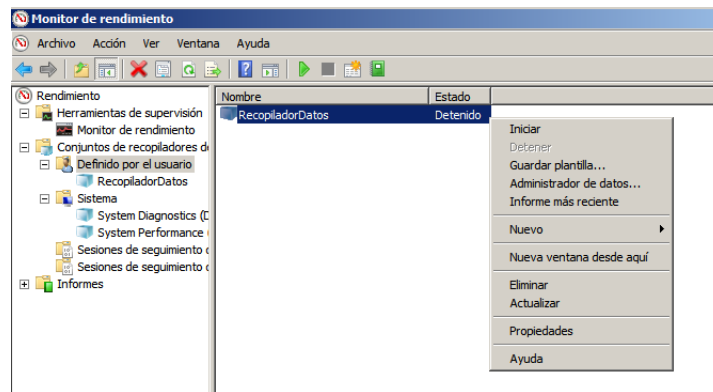


Figura 4.5: Creando un recopilador. Parte 5.

Una vez activado esperamos unos minutos para que recoja datos de nuestra máquina. Al parar el recolector de datos podemos ver el resultado de la ejecución. Vemos ahora en la Figura 4.6 la gráfica con todos los datos estudiados. Pero como trabajar con esta gráfica es algo incómodo debido a la cantidad de gráficas superpuestas, vamos a desactivar todas las gráficas menos dos. El resultado puede verse en la Figura 4.7. He elegido esas dos características porque son complementarias. Una de ellas es el porcentaje de tiempo que la máquina está inactiva, la otra marca el porcentaje de tiempo que está activa. Puede verse que ambas gráficas son complementarias.

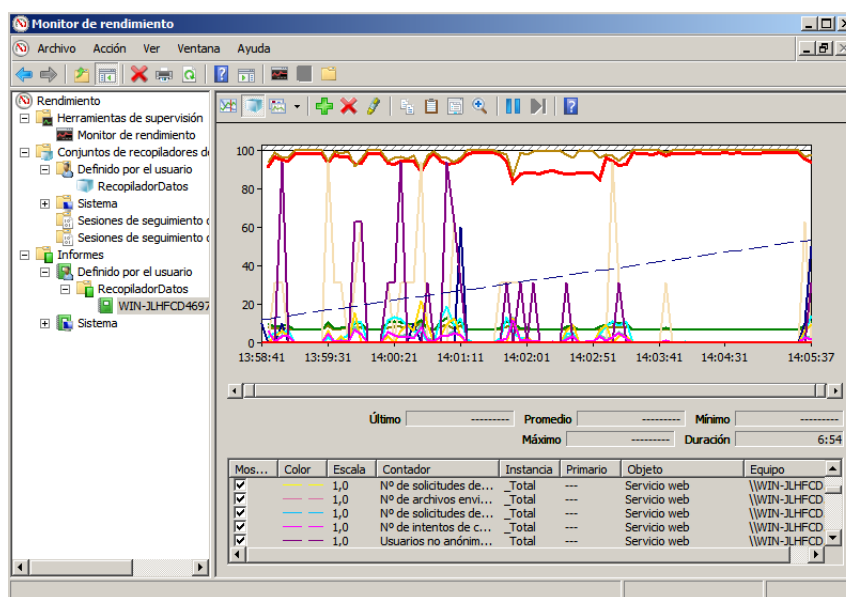


Figura 4.6: Creando un recopilador. Parte 6.

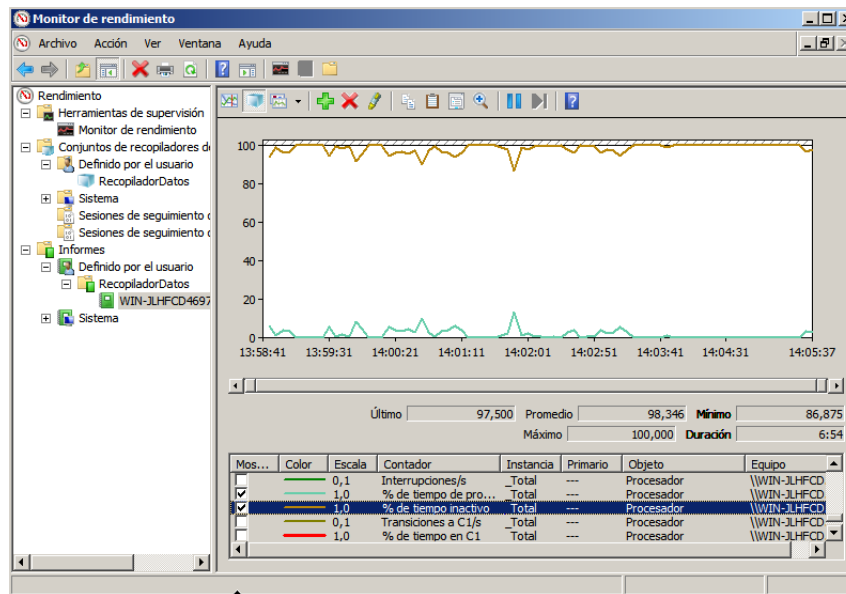


Figura 4.7: Creando un recopilador. Parte 7.

- Visite la web del proyecto y acceda a la demo que proporcionan donde se muestra cómo monitorizan un servidor. Monitorice varios parámetros y haga capturas de pantalla de lo que está mostrando comentando qué observa.

La web de Munin[1] ofrece una demo que muestra distintas mediciones de parámetros a lo largo de un día, una semana, un mes o un año. Para mostrar un ejemplo, observaremos la información referente a los procesos.

En primer lugar vemos que se nos aporta una gráfica5.1 con el número de threads que tiene la máquina ejecutándose. Salvo variaciones de unos 10 procesos como máximo, vemos que el número se mantiene estable. Además los threads aumentan o disminuyen en paquetes, esto puede ser el resultado de ejecutar programas que dependen o dividen el trabajo en varios threads. Podemos ver, en la gráfica semanal, que el jueves hay durante unos minutos un pico notablemente más alto, puede que fuera una sobrecarga del sistema. Algo curioso, que no soy capaz de explicar, es la desaparición de los datos durante un par de horas. Los threads no parecen haber caído a 0, si no que siguen en la misma línea que los días anteriores, pero la gráfica sufre un corte alrededor de las 20:00 del viernes.

buildd.munin-monitoring.org

all disk other processes system

Number of threads

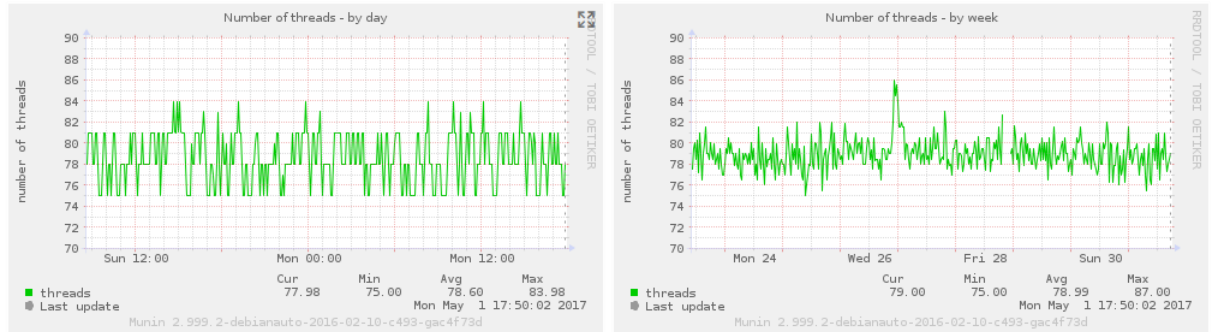


Figura 5.1: Threads ejecutándose en la máquina

Así mismo, disponemos de las gráficas correspondientes al mismo periodo de tiempo de los procesos de la máquina. Podemos ver, en la Figura 5.2 la misma idea que en la gráfica anterior. Se mantienen alrededor de los 75 procesos con variaciones que no exceden los 10 procesos. Observamos como el pico del jueves también aparece, lo que nos da a entender que en ese momento fue de gran actividad. De forma adicional al análisis anterior, ahora podemos detectar que la mayoría de los procesos en ejecución están en estado inactivo, tan solo unos cuantos permanecen en cada instante con la posibilidad de ser ejecutados. Esto puede ser un problema, pues quizás la máquina no esté aprovechándose al máximo. Para asegurarse, habría que analizar los datos de la CPU. Por último, me gustaría destacar que, otra vez, el viernes hay una pérdida de datos y la gráfica se corta. No parece que sea una caída del sistema, pues a continuación la gráfica tomaría un tiempo de reactivación de los procesos y durante unos minutos la actividad sería menor. Más bien parece una interrupción de la monitorización.

## Processes

processes • processes

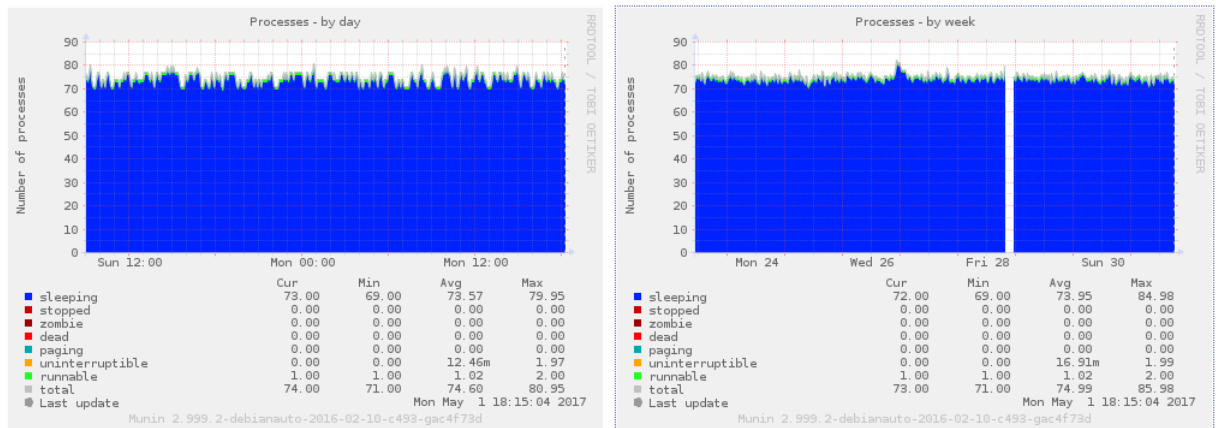


Figura 5.2: Procesos ejecutándose en la máquina

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

Mostraremos un breve resumen sobre el segundo artículo<sup>1</sup> mencionado en el guión prácticas. El artículo primero explica la naturaleza del uso de strace. Afirma que no es una herramienta para programadores sino para administradores de sistema, y que no hace falta tener grandes nociones de programación, tan solo unos comandos básicos de Linux. Strace no es un *'debugger'*, solo es una herramienta para monitorizar las llamadas al sistema realizadas por el proceso. Para ejecutar strace sobre un proceso al que se quiere monitorizar, basta con *adjuntar* la herramienta al proceso, en el artículo se detalla un ejemplo con el comando *strace cat test.txt*, donde test.txt es un archivo con un par de frases escritas. Al ejecutarlo, se puede ver que la ejecución ha causado múltiples llamadas al sistema, algunas de ellas reconocibles de inmediato. Aparece la llamada *open*, para abrir un archivo, la llamada *nmap* para traer a memoria principal datos de disco, y una llamada *close*, para cerrar el archivo. Apoyándose en esta ejecución, se explican los elementos básicos del resultado de strace sobre un proceso.

Para mostrar un ejemplo de la utilidad de strace, se plantea la siguiente cuestión, ¿cómo podemos ubicar los archivos de tipo log de un servicio en ejecución? Supongamos que Apache está escribiendo los archivos log en algún lugar que no esperábamos y no encontramos la forma de averiguar dónde. Podemos ejecutar *strace -Ff -o output.txt -e open /etc/init.d/httpd*

<sup>1</sup>[http://blog.softlayer.com/2013/sysadmin-tips-and-tricks-using-strace-to-monitor-system-calls#utm\\_source=twitter&utm\\_medium=social&utm\\_content=beyond-the-command-line-with-strace&utm\\_campaign=blog\\_development-tips-and-tricks](http://blog.softlayer.com/2013/sysadmin-tips-and-tricks-using-strace-to-monitor-system-calls#utm_source=twitter&utm_medium=social&utm_content=beyond-the-command-line-with-strace&utm_campaign=blog_development-tips-and-tricks)

*restart* para seguir las llamadas al sistema de tipo open en todas las hebras y guardarlas en el archivo output.txt. Ahora podemos filtrar por la palabra log y encontraremos qué archivos está abriendo.

Otra gran ventaja de strace es que está disponible para todas las distribuciones mayoritarias de Linux y es de sencilla instalación a partir del gestor de paquetes de la distribución en uso.

## 7. Escriba un script en Python o PHP y analice su comportamiento usando el profiler presentado.

He creado un script básico con dos funciones. Una de ellas la ejecuto muchas veces, la otra solo una vez cuando el programa termina. Puede verse el código fuente en la Figura 7.1.

```
1 def miFuncionTonta( x ):
2     print "Hasta el infinito y más allá! (%d)" % (x)
3
4 def miFuncionInteligente():
5     print "Adiós"
6
7 x = 1
8 while x<1000000:
9     miFuncionTonta(x)
10    x += 1
11    x += x/x - 1
12 miFuncionInteligente()
~^
```

Figura 7.1: Script Python de prueba.

Para ejecutar el programa de *profiling* Cprofile[5] basta con el comando mostrado en la Figura 7.2. Hay muchas otras formas de usar esta herramienta, aunque por motivos de simplicidad usaré esta para poder hacer el *profile* a un script.

```
migtorlop@mar may 02 - 12:20:30]:~$ python -m cProfile test.py
```

Figura 7.2: Profile de un script en Python

Podemos ver el resultado de la ejecución cuando termine el programa en la propia terminal. Observamos en la Figura 7.3 que nos salen datos de cada función ejecutada en el script, obtenemos el tiempo consumido en cada función, el número de ejecuciones y el tiempo medio. En nuestro caso, vemos que hay una función que se ejecuta muchas veces y por tanto su tiempo consumido es casi el total del script. La otra función sólo tiene una ejecución y el tiempo consumido es tan pequeño que con la pérdida por aproximación nos da 0. Existe la posibilidad de alterar Cprofile para que nos de otra información. Su uso es bastante sencillo, lo que lo convierte en una herramienta práctica y rápida para optimizar cuellos de botella en programas pesados.



```
Hasta el infinito y más allá! (999996)
Hasta el infinito y más allá! (999997)
Hasta el infinito y más allá! (999998)
Hasta el infinito y más allá! (999999)
Adiós
1000002 function calls in 8.055 seconds

Ordered by: standard name

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
1      0.733    0.733    8.055    8.055 test.py:1(<module>)
999999  7.321    0.000    7.321    0.000 test.py:1(miFuncionTonta)
1      0.000    0.000    0.000    0.000 test.py:4(miFuncionInteligente)
1      0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Figura 7.3: Salida del *profile*.

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

Desde phpMyAdmin he creado una tabla vacía con dos columnas. No importa el tipo de las columnas. Sobre esa tabla he realizado una consulta 8.1 desde la consola mysql que nos facilita phpMyAdmin. Si marcamos la opción de perfilar, podremos hacer un *profile* de la consulta.

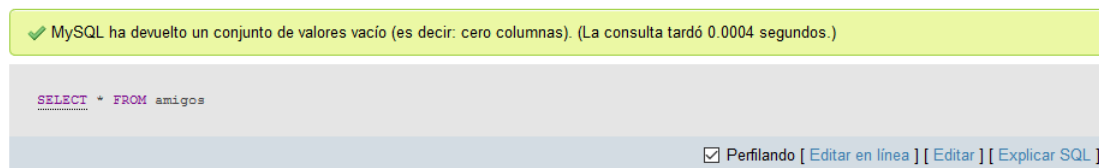
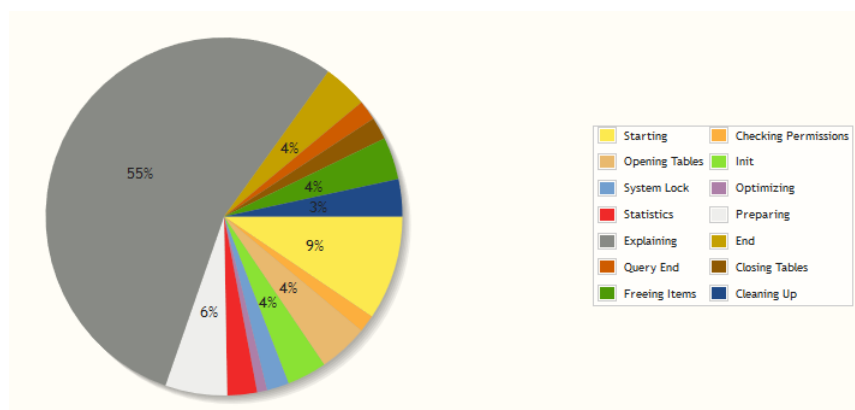


Figura 8.1: Consulta sobre una tabla.

Una vez hecha la consulta, podemos ver una detallada tabla en la que se separa el tiempo consumido en cada tarea, el porcentaje el mismo y el orden de ejecución de las distintas tareas. PhpMyAdmin nos da dos interfaces para verlo, una tabla como la que puede verse en la Figura 8.2 o un diagrama circular como en la Figura 8.3. En ambos se ve claramente que la tarea que más tiempo ha consumido es la de realizar el profile y plasmar los datos en las gráficas. Esto se debe a que la tabla creada está vacía y por lo tanto la consulta es muy rápida. En consultas más complejas y con tablas de mayor peso, el acceso a memoria debería ser la tarea más costosa.

Perfilando							
Perfilación detallada			Resumen por estado				
Orden ▾	Estado ⚙	Tiempo	Estado ⚙	Tiempo Total ▾	% de Tiempo	Llamadas	σ de Tiempo
1	Starting	42 μs	Explaining	243 μs	54.61%	1	243 μs
2	Checking Permissions	7 μs	Starting	42 μs	9.44%	1	42 μs
3	Opening Tables	20 μs	Preparing	25 μs	5.62%	1	25 μs
4	Init	16 μs	Opening Tables	20 μs	4.49%	1	20 μs
5	System Lock	9 μs	End	18 μs	4.04%	1	18 μs
6	Optimizing	4 μs	Freeing Items	17 μs	3.82%	1	17 μs
7	Statistics	12 μs	Init	16 μs	3.60%	1	16 μs
8	Preparing	25 μs	Cleaning Up	15 μs	3.37%	1	15 μs
9	Explaining	243 μs	Statistics	12 μs	2.70%	1	12 μs
10	End	18 μs	System Lock	9 μs	2.02%	1	9 μs
11	Query End	8 μs	Closing Tables	9 μs	2.02%	1	9 μs
12	Closing Tables	9 μs	Query End	8 μs	1.80%	1	8 μs
13	Freeing Items	17 μs	Checking Permissions	7 μs	1.57%	1	7 μs
14	Cleaning Up	15 μs	Optimizing	4 μs	0.90%	1	4 μs

Figura 8.2: Tabla del *profile*.Figura 8.3: Diagrama circular del *profile*.

9. (Cuestión opcional 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.

A pesar de que no se recomienda hacerlo en el manual[3] el archivo a modificar para programar una tarea estaría situado en /var/spool/cron/crontabs. No obstante la documentación recomienda usar el comando crontab, diseñado específicamente para programar tareas

llevadas a cabo por el *daemon* cron.

Un ejemplo de su uso sería añadir la línea que aparece en la Figura 9.2 al archivo de crontab. Esto nos ejecutaría el script `sscodigo` cada día a las 2 de la madrugada, normalmente las copias de seguridad se hacen en horas de bajo uso. El script puede contener cualquier tipo de trabajo pero para este ejercicio podría ser el que aparece en la Figura 9.1.

```
#!/bin/bash
cp ~/codigo ~/seguridad/$(date)
```

Figura 9.1: Línea del archivo crontab.

```
00 02 * * * /home/miguelangeltorres/bin/sscodigo
```

Figura 9.2: Script ejecutado por el cron.

## Referencias

- [1] <http://demo.munin-monitoring.org/>, Consultado el 1 de Mayo de 2017.
- [2] Debian. Debian package management: <https://www.debian.org/doc/manuals/debian-reference/ch02.en.html>. 2013.
- [3] Linux die. Man crontab: <https://linux.die.net/man/1/crontab>.
- [4] Python.org. <http://man7.org/linux/man-pages/man1/dmesg.1.html>.
- [5] Python.org. <https://docs.python.org/2/library/profile.html>.
- [6] Erik Troan. Linux man: [http://www.linuxcommand.org/man\\_pages/logrotate8.html](http://www.linuxcommand.org/man_pages/logrotate8.html).