

INGENIERÍA DE SERVIDORES (2016-2017)
GRADO EN INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE GRANADA

Memoria Práctica 3

David Criado Ramón

8 de diciembre de 2016

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?	5
2	¿Qué archivo ha de modificar para programar una tarea? Escriba la línea necesaria para ejecutar una vez al día una copia del directorio <code>~/codigo a ~/seguridad/\$fecha</code> donde <code>\$fecha</code> es la fecha actual (puede usar el comando <code>date</code>).	8
2.1	Ubuntu Server 14.04	8
2.2	CentOS	9
3	Pruebe a ejecutar el comando, conectar un dispositivo USB y vuelva a ejecutar el comando. Copie y pegue la salida del comando. (considere usar <code>dmesg tail</code>). Comente qué observa en la información mostrada.	10
4	Ejecute el monitor de “System Performance” y muestre el resultado. Incluya capturas de pantalla comentando la información que aparece.	10
5	Cree un recopilador de datos definido por el usuario (modo avanzado) que incluya tanto el contador de rendimiento como los datos de seguimiento. Almacene el resultado en el directorio <code>Escritorio/logs</code> . Incluya las capturas de pantalla de cada paso.	12
6	Visite la web del proyecto y acceda a la demo que proporcionan (http://demo.munin-monitoring.org/) donde se muestra cómo monitorizan un servidor. Monitoree varios parámetros y haga capturas de pantalla de lo que está mostrando comentando qué observa.	20
7	Escriba un breve resumen sobre alguno de los artículos donde se muestra el uso de <code>strace</code> o busque otro y coméntelo.	23
8	Escriba un script en Python o PHP y analice su comportamiento usando el profiler presentado.	24
9	Acceda a la consola <code>mysql</code> (o a través de <code>phpMyAdmin</code>) y muestre el resultado de mostrar el “profile” de una consulta (la creación de la BD y la consulta la puede hacer libremente).	26

Índice de figuras

1.1.	Miramos la localización de log de <code>dpkg</code> en Ubuntu Server y miramos las últimas 25 líneas que contienen <i>installed</i>	5
------	---	---

1.2.	Miramos la localización de log de yum en CentOS y miramos las líneas que contienen <i>Installed</i>	6
1.3.	Miramos las fechas de los archivos que empiezan con dmesg en /var/log y las primeras 20 líneas del último dmesg.	7
1.4.	Comprobamos el tipo de archivo con <i>file[1]</i> y utilizamos <i>gunzip -c[2]</i> para mostrar su contenido.	7
1.5.	Comprobamos ahora las últimas líneas del archivo.	8
2.1.	Comprobamos con cat que hemos modificado el archivo indicado en la documentación.	9
2.2.	Comprobamos que efectivamente se ha ejecutado la tarea de cron.	9
3.1.	A la izquierda <i>dmesg</i> sin introducir el dispositivo USB, a la derecha <i>dmesg</i> nada más conectarlo.	10
4.1.	Ventana en la que seleccionamos los elementos que vamos a monitorizar. .	11
4.2.	Gráfico de la monitorización en Windows Server 2012 R2 de los parámetros propuestos.	12
5.1.	Le damos un nombre al recopilador de datos (en mi caso, ISE-ejercicio5) y marcamos <i>Crear manualmente (avanzado)</i> antes de pulsar <i>Siguiente</i> . . .	13
5.2.	Dejamos <i>Crear registro de datos</i> , marcamos <i>Contador de rendimiento y Datos de seguimiento de eventos</i> y pulsamos en <i>Siguiente</i>	13
5.3.	Dejamos el intervalo de muestra en 15 segundos y pulsamos en <i>Agregar...</i>	14
5.4.	Seleccionamos todos los campos relativos a las secciones <i>Procesador</i> , <i>Proceso</i> y <i>Servicio web</i>	14
5.5.	Revisamos que efectivamente se han añadido todos los campos de las tres áreas mencionadas (indicando con un * para cada área) y pulsamos en <i>Siguiente</i>	15
5.6.	Volvemos a pulsar en <i>Siguiente</i>	15
5.7.	Pulsamos en Examinar y seleccionamos la ruta (en mi caso, C:\Users\Administrador\Desktop\logs) donde deseamos que se guarden los <i>logs</i> del recopilador de datos.	16
5.8.	Revisamos que la ruta sea correcta y pulsamos en <i>Siguiente</i>	16
5.9.	Marcamos <i>Iniciar ahora este conjunto de recopiladores de datos</i> (puesto que voy a realizar la monitorización de inmediato) y pulsamos <i>Finalizar</i> . .	17
5.10.	Paramos el recopilador de datos una vez que han pasado 5 minutos haciendo click derecho en el menú lateral al recopilador de datos que tiene el nombre que hemos puesto y pulsamos <i>Detener</i>	17
5.11.	Gráfica del recopilador de datos mostrando todos los parámetros.	18
5.12.	Gráfica del recopilador de datos con una selección de parámetros.	19
6.1.	Página de inicio de la demo de munin.	20
6.2.	Monitorización de procesos en Munin.	21
6.3.	Monitorización de disco en Munin.	22
6.4.	Monitorización de memoria en Munin.	23
8.1.	Resultados del profiler cProfile de Python tras ejecutar el script.	25
9.1.	En <i>Bases de datos</i> creamos una nueva base de datos a la que llamamos ISE.	26
9.2.	Pulsamos en la lista de bases de datos <i>ISE</i>	27

9.3.	Creamos una nueva tabla de 4 columnas llamada Test y pulsamos en Enviar.	27
9.4.	Creamos una nueva tabla con llave primaria un dni de 9 caracteres y el resto varchar de 50 caracteres (apellidos, nombre, grupo.	28
9.5.	Insertamos una tupla válida en la tabla pulsando en el menú superior en <i>Insertar</i> y rellenando los campos.	28
9.6.	Pulsamos en la parte superior en SQL y realizamos la consulta. En mi caso he mostrado todos los datos de la tabla de aquellas personas que tuviesen el DNI de la tupla insertada.	29
9.7.	Podemos observar que el único resultado que obtenemos es la tupla que insertamos.	29
9.8.	Tras pulsar en <i>Perfilando</i> aparece esta gráfica.	30

Índice de tablas

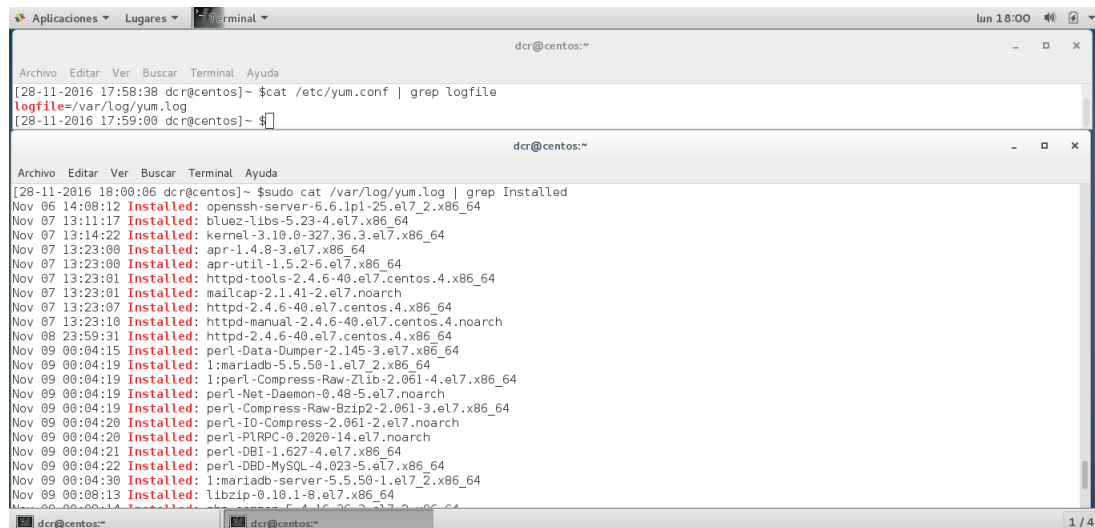
1. ¿Qué archivo le permite ver qué programas se han instalado con el gestor de paquetes? ¿Qué significan las terminaciones .1.gz o .2.gz de los archivos en ese directorio?

El archivo que contiene qué programas se ha instalado es el *log* del gestor de paquetes. En Ubuntu Server, todas las instalaciones podemos encontrarlas en el archivo de configuración de *dpkg* (*Gestor de paquetes de Debian*): `/var/log/dpkg.log` [3]. Esto se debe a que el gestor de descargas *apt*, que a su vez trabaja sobre *apt-get* se encarga de añadir/utilizar a la base de datos de *dpkg* todas las acciones realizadas con *apt* (excepto cuando se descargan los fuentes). [4].

```
(28-11-2016 18:17:04 dcr@ubuntu1) $cat /etc/dpkg/dpkg.cfg | grep log
log /var/log/dpkg.log
(28-11-2016 18:17:10 dcr@ubuntu1) $cat /var/log/dpkg.log | grep installed | tail -n 25
2016-11-10 13:11:35 status half-installed openssh-server:amd64 1:6.6p1-2ubuntu2.8
2016-11-10 13:11:35 status half-installed openssh-server:amd64 1:6.6p1-2ubuntu2.8
2016-11-10 13:11:35 status half-installed openssh-server:amd64 1:6.6p1-2ubuntu2.8
2016-11-10 13:11:35 status installed ufw:all 0.34~rc-0ubuntu2
2016-11-10 13:11:36 status installed ureadahead:amd64 0.100.0-16
2016-11-10 13:11:36 status installed man-db:amd64 2.6.7.1-1ubuntu1
2016-11-10 13:11:41 status installed ufw:all 0.34~rc-0ubuntu2
2016-11-10 13:11:41 status installed openssh-server:amd64 1:6.6p1-2ubuntu2.8
2016-11-10 13:11:41 status installed ureadahead:amd64 0.100.0-16
2016-11-11 12:25:31 status half-installed ucbmin:all 1.820
2016-11-11 12:25:47 status half-installed ucbmin:all 1.820
2016-11-11 12:26:14 status installed ureadahead:amd64 0.100.0-16
2016-11-11 12:26:56 status half-installed libnet-ssleay-perl:amd64 1.58-1
2016-11-11 12:26:56 status half-installed libauthn-pam-perl:amd64 0.16-2build4
2016-11-11 12:26:56 status half-installed libio-pty-perl:amd64 1:1.08-1build4
2016-11-11 12:26:56 status half-installed libapt-pkg-perl:amd64 0.1.29build1
2016-11-11 12:26:56 status half-installed apt-show-versions:all 0.22.3
2016-11-11 12:26:58 status installed man-db:amd64 2.6.7.1-1ubuntu1
2016-11-11 12:26:58 status installed libnet-ssleay-perl:amd64 1.58-1
2016-11-11 12:26:58 status installed libauthn-pam-perl:amd64 0.16-2build4
2016-11-11 12:26:58 status installed libio-pty-perl:amd64 1:1.08-1build4
2016-11-11 12:26:58 status installed libapt-pkg-perl:amd64 0.1.29build1
2016-11-11 12:27:10 status installed apt-show-versions:all 0.22.3
2016-11-11 12:28:33 status installed ucbmin:all 1.820
2016-11-11 12:28:33 status installed ureadahead:amd64 0.100.0-16
(28-11-2016 18:17:13 dcr@ubuntu1) $_
```

Figura 1.1: Miramos la localización de log de dpkg en Ubuntu Server y miramos las últimas 25 líneas que contienen *installed*.

En CentOS utilizamos *yum* (*Yellowdog Updater Modified*) por lo que hemos de mirar en el parámetro logfile de su archivo de configuración [5] para saber su ubicación, que resulta ser `/var/log/yum.log` como podemos apreciar en la figura.



```
dcr@centos:~  
[28-11-2016 17:58:38 dcr@centos]~ $cat /etc/yum.conf | grep logfile  
logfile=/var/log/yum.log  
[28-11-2016 17:59:00 dcr@centos]~ $  
  
dcr@centos:~  
[28-11-2016 18:00:06 dcr@centos]~ $sudo cat /var/log/yum.log | grep Installed  
Nov 06 14:08:12 Installed: openssh-server-6.6.1p1-25.el7_2.x86_64  
Nov 07 13:11:17 Installed: bluez-libs-5.23-4.el7.x86_64  
Nov 07 13:14:22 Installed: kernel-3.10.0-327.36.3.el7.x86_64  
Nov 07 13:23:00 Installed: apr-1.4.8-3.el7.x86_64  
Nov 07 13:23:00 Installed: apr-util-1.5.2-6.el7.x86_64  
Nov 07 13:23:01 Installed: httpd-tools-2.4.6-48.el7.centos.4.x86_64  
Nov 07 13:23:01 Installed: mailcap-2.1.41-2.el7.noarch  
Nov 07 13:23:07 Installed: httpd-2.4.6-48.el7.centos.4.x86_64  
Nov 07 13:23:10 Installed: httpd-manual-2.4.6-48.el7.centos.4.noarch  
Nov 08 23:59:31 Installed: httpd-2.4.6-48.el7.centos.4.x86_64  
Nov 09 00:04:15 Installed: perl-Data-Dumper-2.145-3.el7.x86_64  
Nov 09 00:04:19 Installed: mariadb-5.5.50-1.el7_2.x86_64  
Nov 09 00:04:19 Installed: perl-Compress-Raw-Zlib-2.061-4.el7.x86_64  
Nov 09 00:04:19 Installed: perl-Net-Daemon-0.48-5.el7.noarch  
Nov 09 00:04:19 Installed: perl-Compress-Raw-Bzip2-2.061-3.el7.x86_64  
Nov 09 00:04:20 Installed: perl-IO-Compress-2.061-2.el7.noarch  
Nov 09 00:04:20 Installed: perl-PLRPC-0.2020-14.el7.noarch  
Nov 09 00:04:21 Installed: perl-DBI-1.627-4.el7.x86_64  
Nov 09 00:04:22 Installed: perl-DBD-MySQL-4.023-5.el7.x86_64  
Nov 09 00:04:30 Installed: mariadb-server-5.5.50-1.el7_2.x86_64  
Nov 09 00:08:13 Installed: libzip-0.18.1-8.el7.x86_64  
Nov 09 00:08:14 Installed: libzip-devel-0.18.1-8.el7.x86_64
```

Figura 1.2: Miramos la localización de log de yum en CentOS y miramos las líneas que contienen *Installed*.

Nota: Tanto para yum como para dpkg, aunque estemos viendo una lista y seleccionando los instalados, que un programa se encuentre en dicha lista no implica que en el momento actual se encuentren instalados en el sistema, sino que en algún momento el gestor de paquetes realizó su instalación.

Las terminaciones *.gz en el directorio /var/log/ indican que son antiguas versiones del log comprimidas con gzip debido a la utilidad **logrotate** [6] normalmente. Dicha utilidad nos permite periódicamente (según queramos configurarlo) rotar, comprimir y eliminar tras un cierto número de rotaciones un log antiguo. En Ubuntu Server 14.04 podemos observar el log *dmesg*.

```

[2B-11-2016 18:47:02 dcr@ubuntu1log $ls -l dmesg*
-rw-r----- 1 root adm 36602 nov 28 18:12 dmesg
-rw-r----- 1 root adm 36622 nov 28 17:17 dmesg.0
-rw-r----- 1 root adm 12057 nov 24 13:55 dmesg.1.gz
-rw-r----- 1 root adm 11929 nov 24 13:46 dmesg.2.gz
-rw-r----- 1 root adm 11971 nov 19 17:01 dmesg.3.gz
-rw-r----- 1 root adm 0 nov 19 14:42 dmesg.4.gz
[2B-11-2016 18:47:07 dcr@ubuntu1log $cat dmesg | head -n 20
[ 0.000000] Initializing cgroup subsys cpuset
[ 0.000000] Initializing cgroup subsys cpu
[ 0.000000] Initializing cgroup subsys cpacct
[ 0.000000] Linux version 4.4.0-31-generic (buildde1gw01-43) (gcc version 4.8.4 (Ubuntu 4.8.4-2ub
untu14.04.3) ) #50~14.04.1-Ubuntu SMP Wed Jul 13 01:07:32 UTC 2016 (Ubuntu 4.4.0-31.50~14.04.1-gen
eric 4.4.13)
[ 0.000000] Command line: BOOT_IMAGE=/vmlinuz-4.4.0-31-generic root=/dev/mapper/ABD-sra1z_crypt ro
nomdmonddf nomdmonisw
[ 0.000000] KERNEL supported cpus:
[ 0.000000] Intel GenuineIntel
[ 0.000000] AMD AuthenticAMD
[ 0.000000] Centaur CentaurHauls
[ 0.000000] x86/fpu: xstate.offset[2]: 576, xstate.sizes[2]: 256
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x01: 'x87 floating point registers'
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x02: 'SSE registers'
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x04: 'AVX registers'
[ 0.000000] x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using 'standard' for
mat.
[ 0.000000] x86/fpu: Using 'lazy' FPU context switches.
[ 0.000000] e820: BIOS-provided physical RAM map:
[ 0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbfff] usable
[ 0.000000] BIOS-e820: [mem 0x000000000009fc00-0x000000000000ffff] reserved
[ 0.000000] BIOS-e820: [mem 0x0000000000010000-0x000000000000ffff] reserved
[ 0.000000] BIOS-e820: [mem 0x0000000000010000-0x00000000003ffff] usable
[2B-11-2016 18:47:08 dcr@ubuntu1log $

```

Figura 1.3: Miramos las fechas de los archivos que empiezan con dmesg en /var/log y las primeras 20 líneas del último dmesg.

Podemos observar que las rotaciones más antiguas son las que van adquiriendo un número mayor y que sólo se guardan rotaciones de la 0 a la 4 así como la actual. Probemos que efectivamente son archivos comprimidos con gzip y comparémoslo con el log actual.

```

[2B-11-2016 18:55:27 dcr@ubuntu1log $file dmesg.1.gz
dmesg.1.gz: gzip compressed data, was 'dmesg.0', from Unix, last modified: Thu Nov 24 13:55:02 2016,
max compression
[2B-11-2016 18:55:28 dcr@ubuntu1log $gunzip -c dmesg.1.gz | head -n 20
[ 0.000000] Initializing cgroup subsys cpuset
[ 0.000000] Initializing cgroup subsys cpu
[ 0.000000] Initializing cgroup subsys cpacct
[ 0.000000] Linux version 4.4.0-31-generic (buildde1gw01-43) (gcc version 4.8.4 (Ubuntu 4.8.4-2ub
untu14.04.3) ) #50~14.04.1-Ubuntu SMP Wed Jul 13 01:07:32 UTC 2016 (Ubuntu 4.4.0-31.50~14.04.1-gen
eric 4.4.13)
[ 0.000000] Command line: BOOT_IMAGE=/vmlinuz-4.4.0-31-generic root=/dev/mapper/ABD-sra1z_crypt ro
nomdmonddf nomdmonisw
[ 0.000000] KERNEL supported cpus:
[ 0.000000] Intel GenuineIntel
[ 0.000000] AMD AuthenticAMD
[ 0.000000] Centaur CentaurHauls
[ 0.000000] x86/fpu: xstate.offset[2]: 576, xstate.sizes[2]: 256
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x01: 'x87 floating point registers'
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x02: 'SSE registers'
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x04: 'AVX registers'
[ 0.000000] x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using 'standard' for
mat.
[ 0.000000] x86/fpu: Using 'lazy' FPU context switches.
[ 0.000000] e820: BIOS-provided physical RAM map:
[ 0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbfff] usable
[ 0.000000] BIOS-e820: [mem 0x000000000009fc00-0x000000000000ffff] reserved
[ 0.000000] BIOS-e820: [mem 0x0000000000010000-0x000000000000ffff] reserved
[ 0.000000] BIOS-e820: [mem 0x0000000000010000-0x00000000003ffff] usable
[2B-11-2016 18:55:41 dcr@ubuntu1log $

```

Figura 1.4: Comprobamos el tipo de archivo con *file[1]* y utilizamos *gunzip -c[2]* para mostrar su contenido.

Podemos observar que el contenido de ambas para las primeras líneas es el mismo ya que corresponde a contenido escrito por el kernel al arrancar, probemos ahora con las últimas líneas.

```

(28-11-2016 19:03:49 dcr@ubuntu1log $ cat dnscg | tail -n 5
[ 55.701770] audit: type=1400 audit(1480353126.140:24): apparmor="STATUS" operation="profile_repla
ce" profile="unconfined" name="/usr/lib/connman/scripts/dhclient-script" pid=1623 comm="apparmor_par
ser"
[ 55.702056] audit: type=1400 audit(1480353126.140:25): apparmor="STATUS" operation="profile_repla
ce" profile="unconfined" name="/usr/lib/connman/scripts/dhclient-script" pid=1623 comm="apparmor_par
ser"
[ 55.705027] audit: type=1400 audit(1480353126.140:26): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/sbin/mysqld" pid=1624 comm="apparmor_parser"
[ 55.713930] audit: type=1400 audit(1480353126.156:27): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/sbin/tcpdump" pid=1626 comm="apparmor_parser"
[ 56.054421] audit: type=1400 audit(1480353126.496:28): apparmor="STATUS" operation="profile_repla
ce" profile="unconfined" name="/usr/sbin/mysqld" pid=1725 comm="apparmor_parser"
(28-11-2016 19:03:49 dcr@ubuntu1log $ gunzip -c dnscg.1.gz | tail -n 5
[ 140.326630] audit: type=1400 audit(1479992101.602:23): apparmor="STATUS" operation="profile_repla
ce" profile="unconfined" name="/usr/lib/NetworkManager/nm-dhcp-client.action" pid=1538 comm="apparmo
r_parser"
[ 140.326635] audit: type=1400 audit(1479992101.602:24): apparmor="STATUS" operation="profile_repla
ce" profile="unconfined" name="/usr/lib/connman/scripts/dhclient-script" pid=1538 comm="apparmor_par
ser"
[ 140.326912] audit: type=1400 audit(1479992101.602:25): apparmor="STATUS" operation="profile_repla
ce" profile="unconfined" name="/usr/lib/connman/scripts/dhclient-script" pid=1538 comm="apparmor_par
ser"
[ 140.329940] audit: type=1400 audit(1479992101.602:26): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/sbin/mysqld" pid=1539 comm="apparmor_parser"
[ 140.337980] audit: type=1400 audit(1479992101.606:27): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/sbin/tcpdump" pid=1541 comm="apparmor_parser"
(28-11-2016 19:03:54 dcr@ubuntu1log $

```

Figura 1.5: Comprobamos ahora las últimas líneas del archivo.

Aunque los mensajes son bastante parecidos podemos observar que hay diferencias en la marca de tiempo puesta desde que el sistema había arrancado en cada uno de los casos.

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`).

2.1. Ubuntu Server 14.04

En Ubuntu el archivo a modificar para programar una tarea es `/var/spool/cron/crontabs/$user` (donde `$user` es el nombre del usuario que quiere añadir el trabajo de cron) [7].

Para añadir la tarea diaria editamos dicho archivo. Para ello podemos utilizar `crontab -e` [8] al editarlo añadimos la siguiente línea:

```
0 13 * * * mkdir ~/seguridad/$(date +%d-%m-%Y) && cp -R ~/codigo
~/seguridad/$(date +%d-%m-%Y)"
```

Nota: He puesto 13 como hora para poder comprobar rápidamente que funciona correctamente. Lo más habitual es ponerlo a las 12 de la noche (0) o a alguna otra hora con poca carga de trabajo (relativa) para el servidor.

Antes de comprobar el funcionamiento correcto de `crontab` vamos a comprobar que efectivamente con `crontab -e` hemos modificado el archivo que se nos ha indicado realizado un `cat` sobre el mismo, tal y como podemos apreciar en la siguiente captura de pantalla.


```

06-12-2016 12:50:59 dcr@ubuntu1 $ sudo cat /var/spool/cron/crontabs/dcr
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.Q8UqEe/crontab installed on Tue Dec  6 12:47:23 2016)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
0 13 * * * mkdir ~/seguridad/$(date +%d-%m-%Y)" && cp -R ~/codigo ~/seguridad/$(date +%d-%m-%Y)"
06-12-2016 12:51:01 dcr@ubuntu1 $

```

Figura 2.1: Comprobamos con cat que hemos modificado el archivo indicado en la documentación.

Al llegar la hora vamos a comprobar que efectivamente se ha ejecutado la tarea. Primero miramos el log del sistema para ver si efectivamente se ha ejecutado la orden indicada y luego revisaremos que efectivamente ha aparecido una nueva carpeta con la fecha del día y se ha creado a las 13:00.

```

06-12-2016 13:07:11 dcr@ubuntu1 $ cat /var/log/syslog | grep CRON | tail -1
Dec  6 13:00:01 ubuntu CRON[2337]: (dcr) CMD (mkdir ~/seguridad/$(date +%d-%m-%Y)" && cp -R ~/codigo ~/seguridad/$(date +%d-%m-%Y)" )
06-12-2016 13:07:14 dcr@ubuntu1 $ ls -l seguridad
total 4
drwxrwxr-x 3 dcr dcr 4096 dic  6 13:00 06-12-2016
06-12-2016 13:07:17 dcr@ubuntu1 $ _

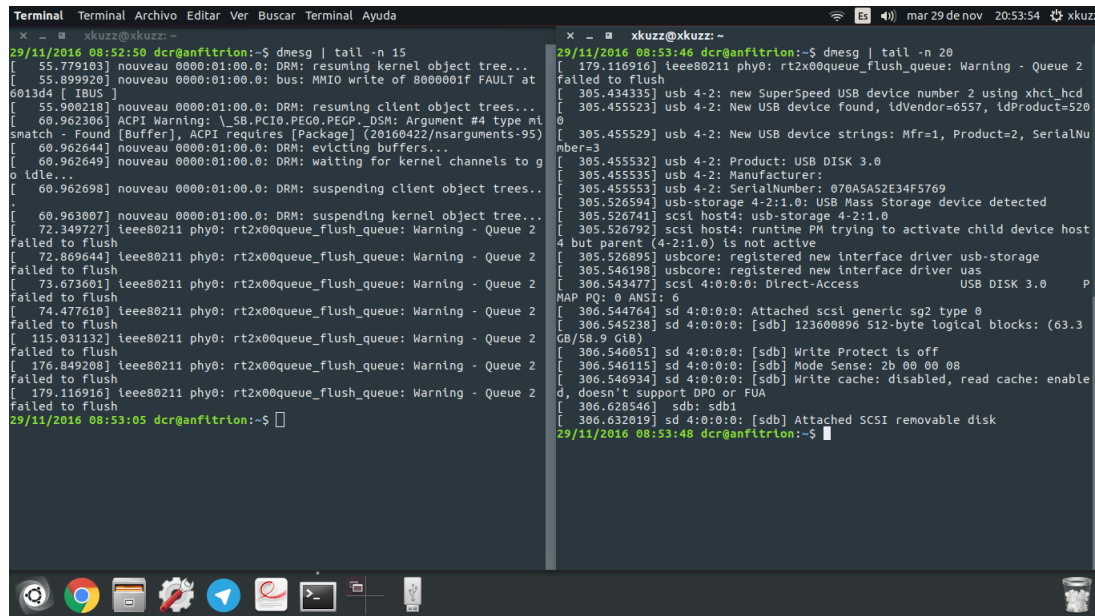
```

Figura 2.2: Comprobamos que efectivamente se ha ejecutado la tarea de cron.

2.2. CentOS

En CentOS el archivo a modificar para programar una tarea es `/var/spool/cron/$user` (donde \$user es el nombre del usuario que quiere añadir el trabajo de cron) [9].

3. 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.



```
Terminal Terminal Archivo Editor Ver Buscar Terminal Ayuda
xkuzz@xkuzz:~$ dmesg | tail -n 15
[ 55.779103] nouveau 0000:01:00.0: DRM: resuming kernel object tree...
[ 55.899920] nouveau 0000:01:00.0: bus: MMIO write of 8000001f FAULT at 0013d4 [ 1B05 ]
[ 55.900218] nouveau 0000:01:00.0: DRM: resuming client object trees...
[ 60.962386] ACPI Warning: \_SB.PCI0.PEG0.PEGP.DSM: Argument #4 type mismatch - Found [Buffer], ACPI requires [Package] (20100422/nsarguments-95)
[ 60.962644] nouveau 0000:01:00.0: DRM: evicting buffers...
[ 60.962649] nouveau 0000:01:00.0: DRM: waiting for kernel channels to go idle...
[ 60.962698] nouveau 0000:01:00.0: DRM: suspending client object trees...
[ 72.349727] ieee80211 phy0: rt2x00queue_flush_queue: Warning - Queue 2 failed to flush
[ 72.869644] ieee80211 phy0: rt2x00queue_flush_queue: Warning - Queue 2 failed to flush
[ 73.673601] ieee80211 phy0: rt2x00queue_flush_queue: Warning - Queue 2 failed to flush
[ 74.477610] ieee80211 phy0: rt2x00queue_flush_queue: Warning - Queue 2 failed to flush
[ 115.031132] ieee80211 phy0: rt2x00queue_flush_queue: Warning - Queue 2 failed to flush
[ 176.849208] ieee80211 phy0: rt2x00queue_flush_queue: Warning - Queue 2 failed to flush
[ 179.116916] ieee80211 phy0: rt2x00queue_flush_queue: Warning - Queue 2 failed to flush
29/11/2016 08:53:05 dcr@anfitrión:~$

xkuzz@xkuzz:~$ dmesg | tail -n 20
[ 179.116916] ieee80211 phy0: rt2x00queue_flush_queue: Warning - Queue 2 failed to flush
[ 305.434335] usb 4-2: new SuperSpeed USB device number 2 using xhci_hcd
[ 305.455523] usb 4-2: New USB device found, idVendor=6557, idProduct=5200
[ 305.455529] usb 4-2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 305.455532] usb 4-2: Product: USB DISK 3.0
[ 305.455535] usb 4-2: Manufacturer:
[ 305.455553] usb 4-2: SerialNumber: 070A5A5E34F5769
[ 305.526594] usb-storage 4-2:1.0: USB Mass Storage device detected
[ 305.526741] scsi host4: usb-storage 4-2:1.0
[ 305.526792] scsi host4: runtime PM trying to activate child device host4 but parent (4-2:1.0) is not active
[ 305.526895] usbcore: registered new interface driver usb-storage
[ 305.546198] usbcore: registered new interface driver usb_lpm
[ 306.543477] scsi 4:0:0:0: Direct-Access USB DISK 3.0 P
MAP PQ: 0 ANSI: 6
[ 306.544764] sd 4:0:0:0: Attached scsi generic sg2 type 0
[ 306.545238] sd 4:0:0:0: [sdb] 123600896 512-byte logical blocks: (63.3 GB/58.9 GiB)
[ 306.546051] sd 4:0:0:0: [sdb] Write Protect is off
[ 306.546115] sd 4:0:0:0: [sdb] Mode Sense: 2b 00 00 08
[ 306.546934] sd 4:0:0:0: [sdb] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[ 306.628546] sdb: sdb1
[ 306.632019] sd 4:0:0:0: [sdb] Attached SCSI removable disk
29/11/2016 08:53:48 dcr@anfitrión:~$
```

Figura 3.1: A la izquierda `dmesg` sin introducir el dispositivo USB, a la derecha `dmesg` nada más conectarlo.

Podemos ver que aparece que se ha conectado un nuevo dispositivo USB, al que se le asocia un número de dispositivo y se nos muestra una serie de detalles como su identificador de producto, identificador de vendedor o el número de serie [10]. El sistema determina que es una unidad de almacenamiento (*storage*) y la monta en `sdb` mostrando información como el tamaño de bloque lógico (512 bytes) y la capacidad de almacenamiento del dispositivo (58.9 GiB) o si está activado el modo de protección contra escritura.

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

Tras iniciar el monitor utilizando el comando `perfmon` en una ventana de consola de PowerShell hacemos click en Herramientas de supervisión > Monitor de Rendimiento y pulsamos en el botón + para añadir los parámetros que deseamos observar. En esa ventana agregamos los que nos interese monitorizar y al pulsar sobre ellos si tenemos marcada la

casilla “Mostrar descripción” podemos información un poco más detallada sobre lo que monitorizan. [11]

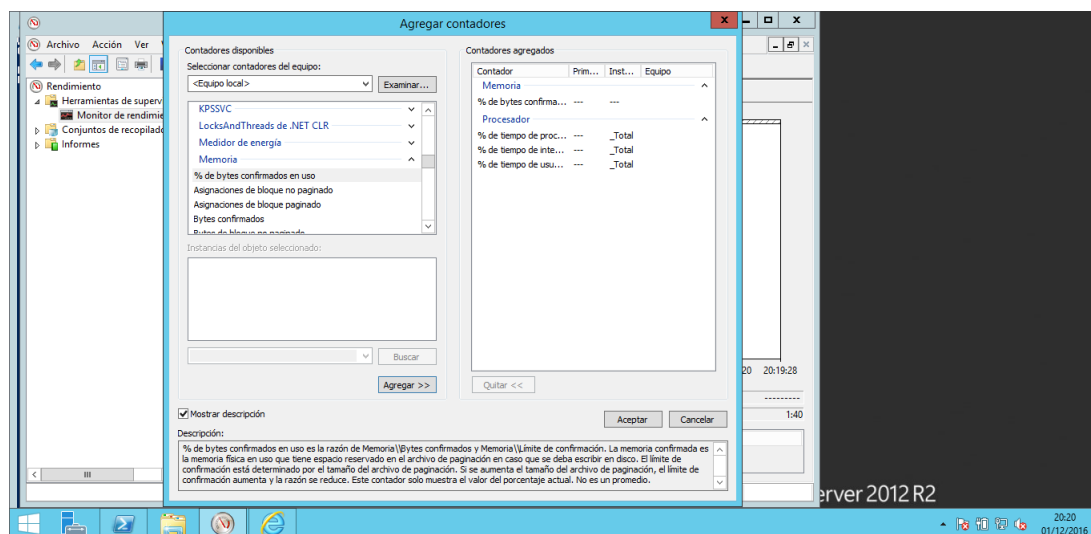


Figura 4.1: Ventana en la que seleccionamos los elementos que vamos a monitorizar.

Vamos a analizar el comportamiento del sistema mientras usamos la máquina para realizar las siguientes cosas: iniciar la descarga de un archivo grande para que no termine la descarga durante el tiempo de monitorización, en plena descarga cargamos una página con gran cantidad de contenido multimedia (YouTube) posteriormente cerramos la página con contenido multimedia y por último cancelamos la descarga.

- **% tiempo de procesador** (*Línea verde*) - Porcentaje de tiempo que el procesador se encuentra atendiendo algún proceso activo (Existe un proceso inactivo que consume ciclos cuando no hay procesos activos listos para usar el procesador).
- **% tiempo de usuario** (*Línea amarilla*) - Porcentaje de tiempo que el procesador se encuentra trabajando en modo usuario. (Existe también el modo privilegiado que es utilizado exclusivamente por el sistema operativo y las llamadas que se hacen al mismo).
- **% tiempo de interrupciones** (*Línea azul*) - Porcentaje de tiempo que el procesador se encuentra atendiendo interrupciones hardware.
- **% de bytes confirmados en memoria** (*Línea roja*) - Memoria física en uso que tiene espacio reservado en el archivo de paginación si hubiese que guardar memoria en disco.

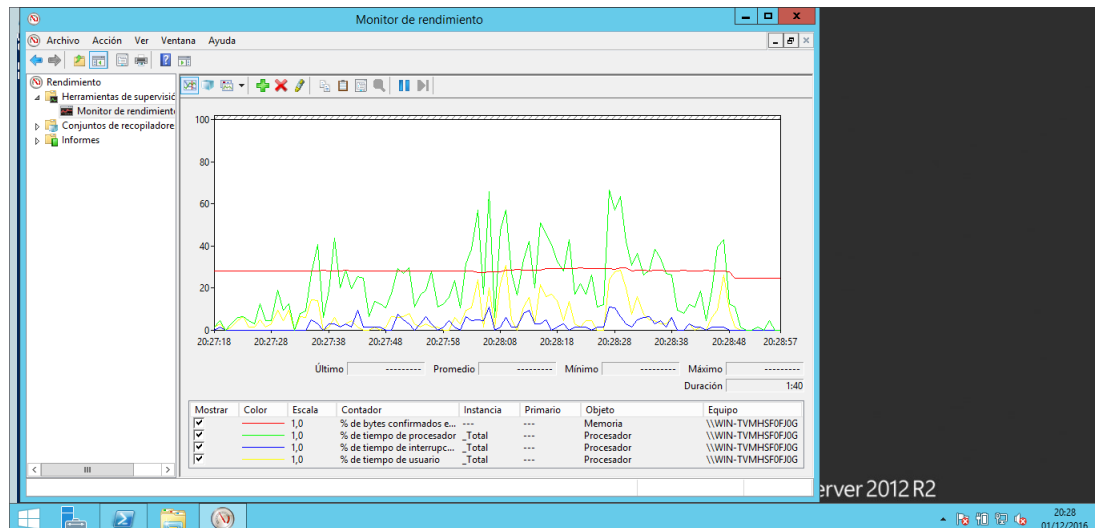


Figura 4.2: Gráfico de la monitorización en Windows Server 2012 R2 de los parámetros propuestos.

Al inicio nos encontramos navegando para buscar el archivo a descargar, por lo que vemos que del porcentaje de uso del procesador la mayor parte es en modo usuario. Una vez se inicia la descarga y hasta que se cancela podemos observar pequeños picos en el porcentaje de tiempo que se encuentra atendiendo interrupciones hardware. Estas deben ser interrupciones de disco provocadas por la descarga. Justo en el centro de la gráfica vemos los mayores picos en el uso de la CPU, en ese momento nos encontramos abriendo un vídeo en YouTube mientras sigue la descarga y dentro de esos picos vemos que hay gran parte de llamadas al sistema por la diferencia que existe entre el porcentaje de tiempo de procesador y el porcentaje de tiempo de procesador en modo usuario. Conforme se van cancelando y cerrando las pestañas del navegador podemos apreciar como rápidamente los porcentajes de procesador, procesador atendiendo interrupciones y procesador en modo usuario bajan a valores muy cercanos al 0 %. Por último, nos queda hablar del porcentaje de bytes confirmados en memoria, puesto que las operaciones que han realizado no han necesitado de grandes cantidades de memoria sólo vemos ligeros cambios en el mismo.

5. Cree un recopilador de datos definido por el usuario (modo avanzado) que incluya tanto el contador de rendimiento como los datos de seguimiento. Almacene el resultado en el directorio *Escritorio\logs*. Incluya las capturas de pantalla de cada paso.

Para empezar tal y como hicimos en la cuestión anterior abrimos una consola de PowerShell y utilizamos el comando `perfmon`. Tras abrirse la ventana del monitor del rendimiento

en el menú lateral abrimos la pestaña *Conjunto de recopiladores de datos* y haciendo click derecho en *Definido por el usuario* pinchamos en *Nuevo >Conjunto de recopiladores de datos* y seguimos los pasos indicados en las capturas de pantalla siguientes para configurarlo. [12]

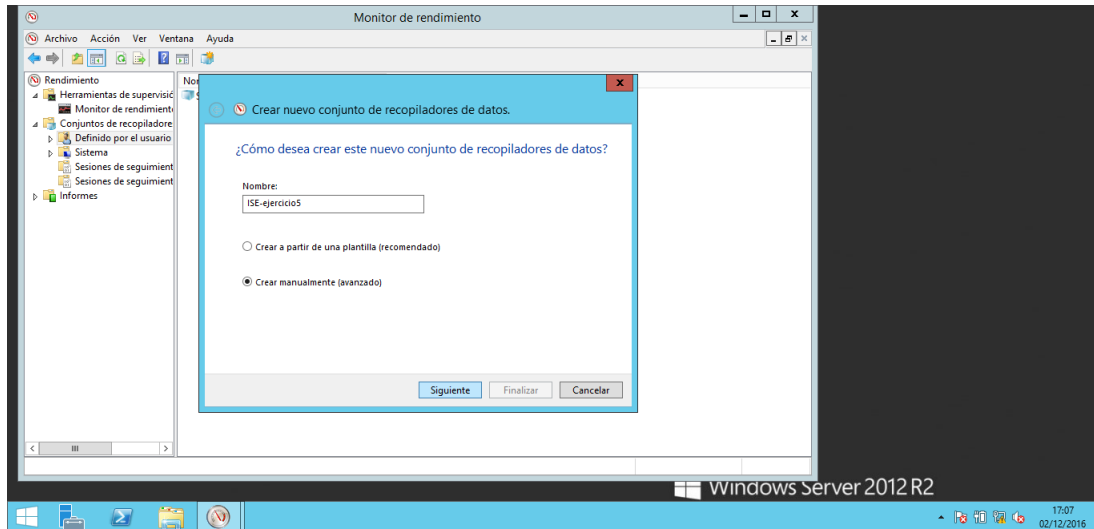


Figura 5.1: Le damos un nombre al recopilador de datos (en mi caso, ISE-ejercicio5) y marcamos *Crear manualmente (avanzado)* antes de pulsar *Siguiente*.

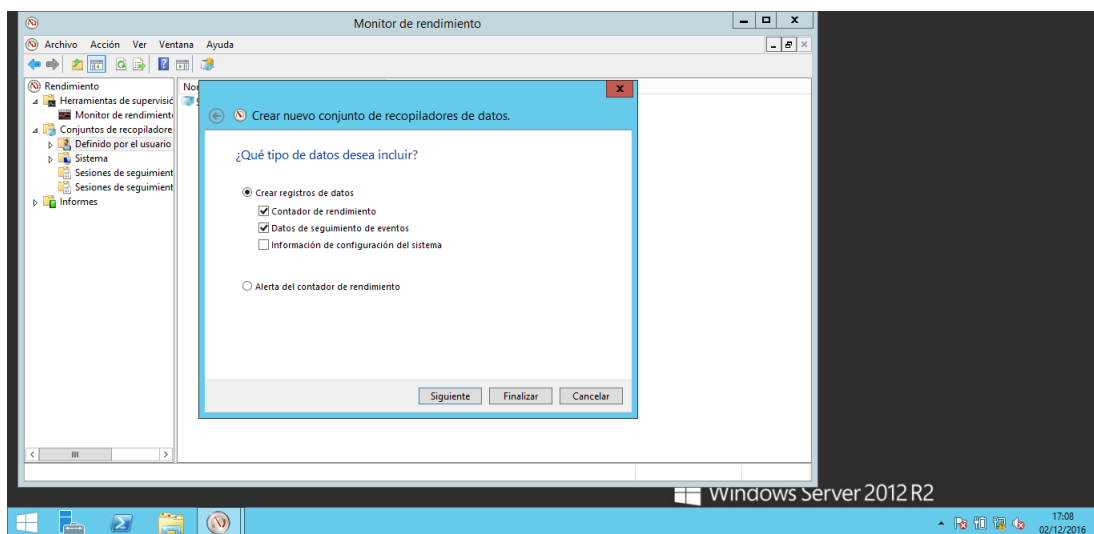


Figura 5.2: Dejamos *Crear registro de datos*, marcamos *Contador de rendimiento* y *Datos de seguimiento de eventos* y pulsamos en *Siguiente*.

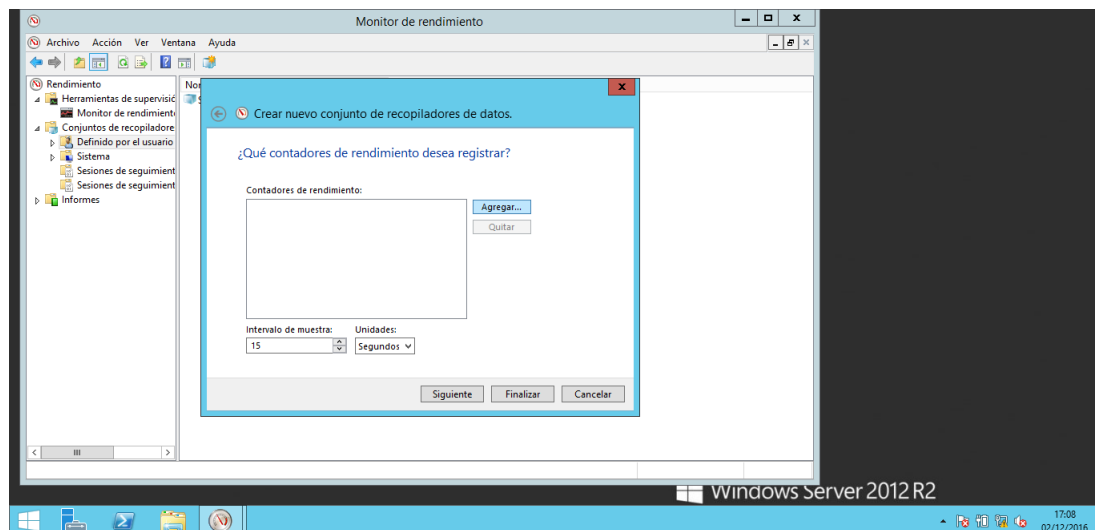


Figura 5.3: Dejamos el intervalo de muestra en 15 segundos y pulsamos en *Agregar...*

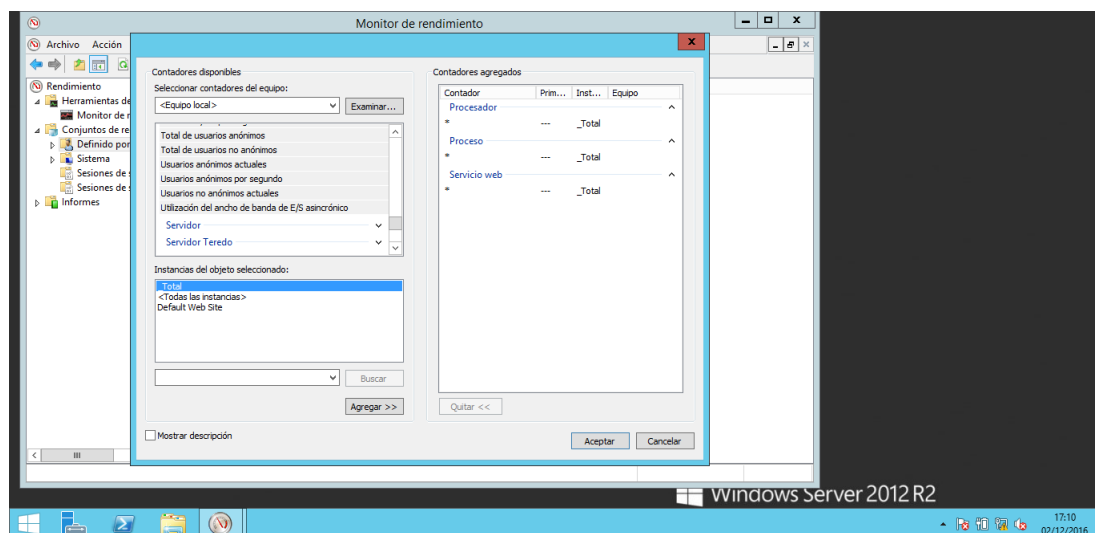


Figura 5.4: Seleccionamos todos los campos relativos a las secciones *Procesador*, *Proceso* y *Servicio web*.

Nota: Podemos añadir todos los campos de una sección pulsando el primero y mientras pulsamos la tecla “Mayus” pulsamos el último, seleccionando todo el rango de valores y añadiéndolos pulsando en Agregar ».

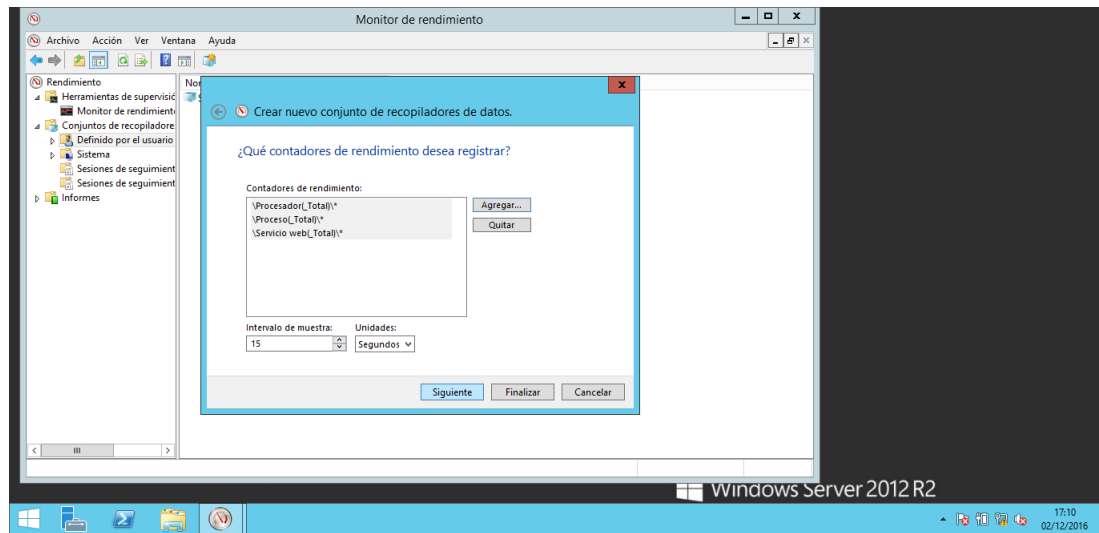


Figura 5.5: Revisamos que efectivamente se han añadido todos los campos de las tres áreas mencionadas (indicando con un * para cada área) y pulsamos en *Siguiente*.

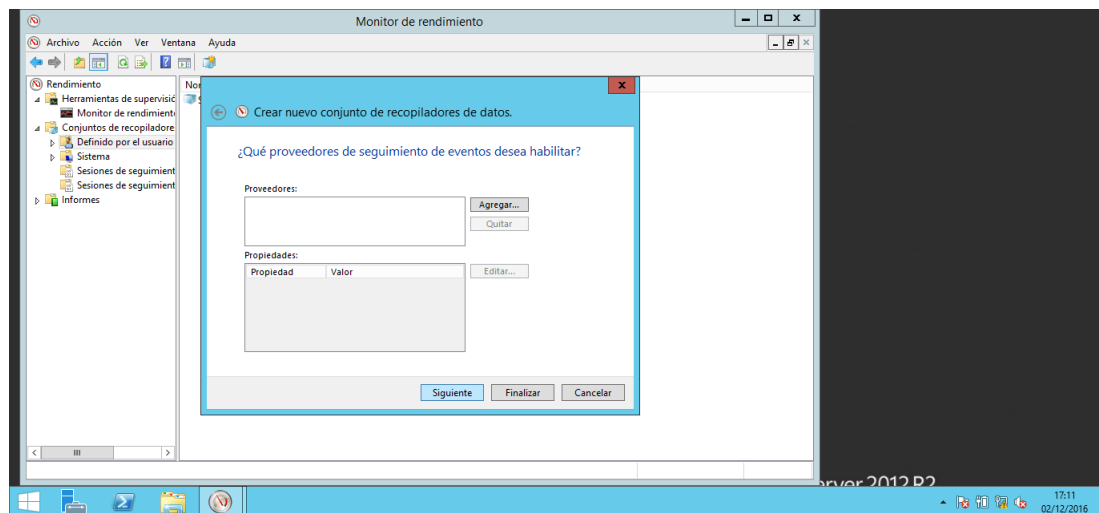


Figura 5.6: Volvemos a pulsar en *Siguiente*.

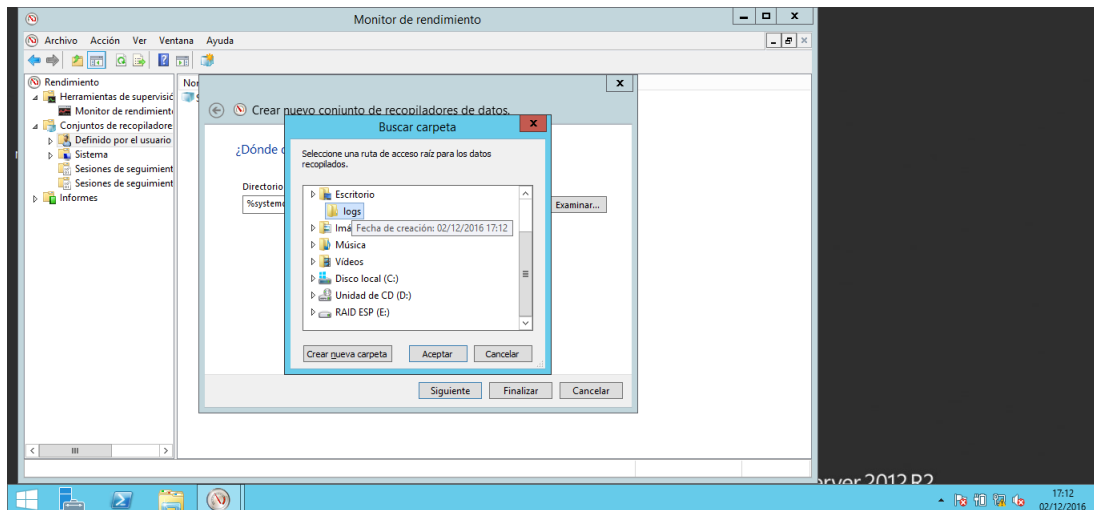


Figura 5.7: Pulsamos en Examinar y seleccionamos la ruta (en mi caso, C:\Users\Administrador\Desktop\logs) donde deseamos que se guarden los *logs* del recopilador de datos.

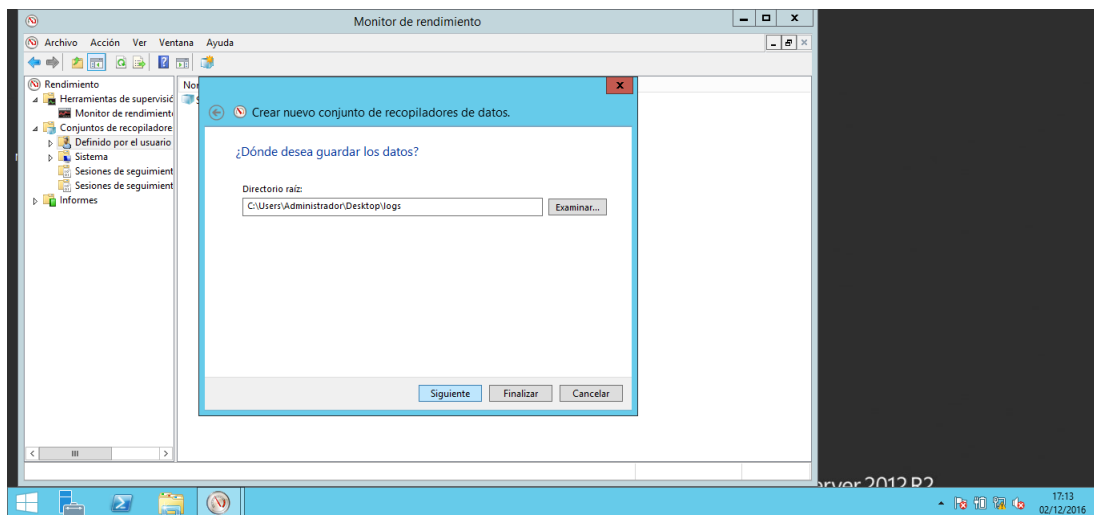


Figura 5.8: Revisamos que la ruta sea correcta y pulsamos en *Siguiente*.

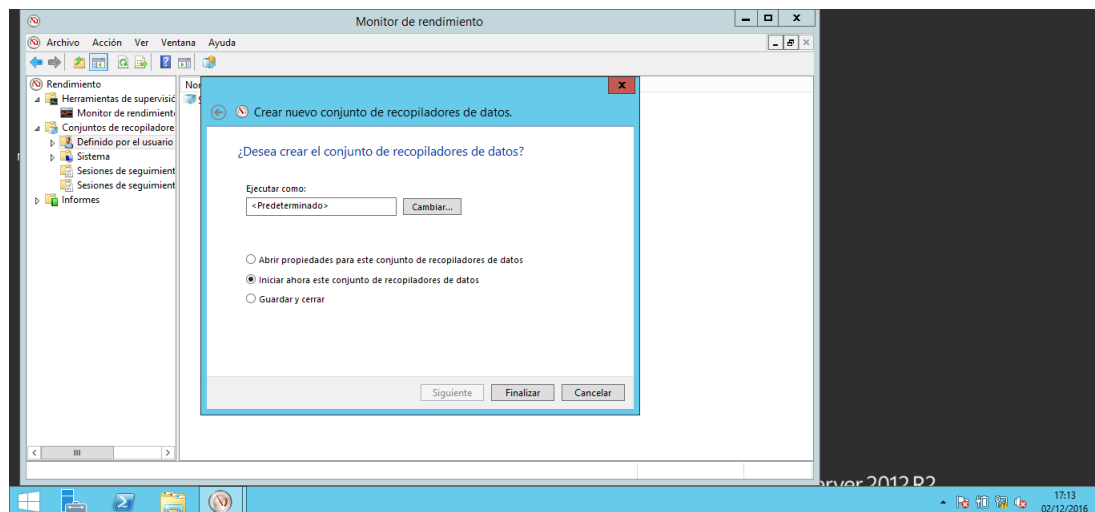


Figura 5.9: Marcamos *Iniciar ahora este conjunto de recopiladores de datos* (puesto que voy a realizar la monitorización de inmediato) y pulsamos *Finalizar*.

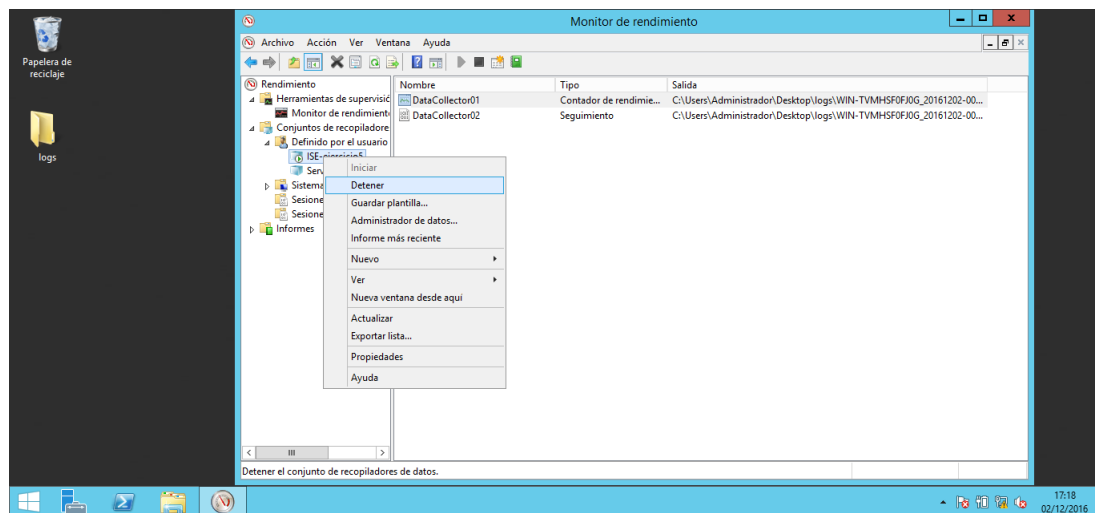


Figura 5.10: Paramos el recopilador de datos una vez que han pasado 5 minutos haciendo click derecho en el menú lateral al recopilador de datos que tiene el nombre que hemos puesto y pulsamos *Detener*.

Tras detenerlo cerramos la ventana y en la ruta en la que hemos almacenado los *logs* encontramos un archivo del monitor del rendimiento. Haciendo doble click en él (en mi caso, nombrado *DataCollector01*) se nos habrá una nueva ventana del monitor de rendimiento con la gráfica de lo que ha ocurrido durante el tiempo de monitorización con los parámetros que hemos ido recopilando, tal y como podemos apreciar en la siguiente figura.

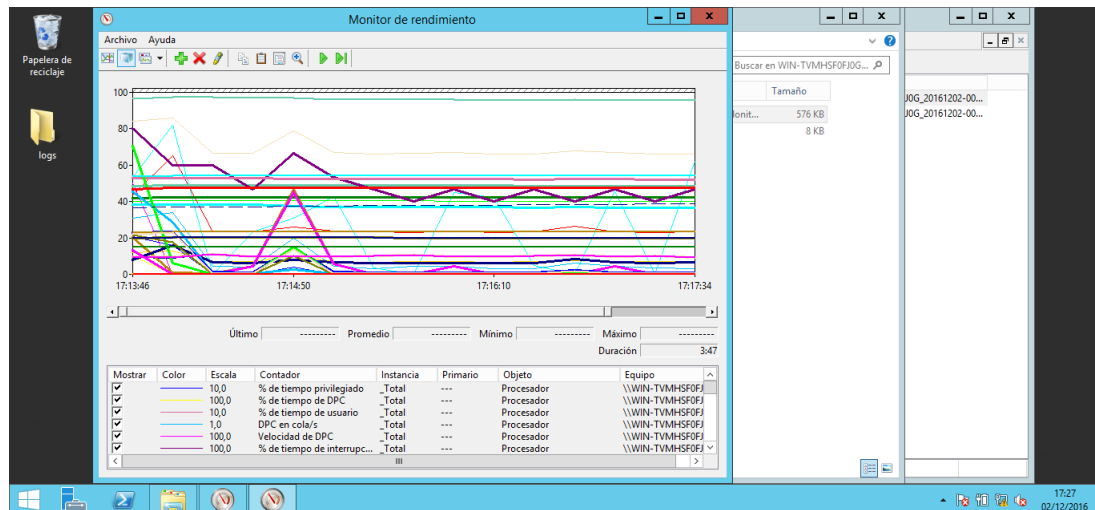


Figura 5.11: Gráfica del recopilador de datos mostrando todos los parámetros.

Para poder interpretar que es lo que ha ocurrido vamos a seleccionar algunos parámetros de los almacenados, en concreto, vamos a fijarnos en los siguientes:

- Operaciones ES (Entrada/Salida) de escritura (*Línea morada gruesa*).
- Bytes de escritura de ES (*Línea rosa*).
- Interrupciones (*Línea amarilla*).
- Porcentaje de tiempo que el procesador se encuentra activo (*Línea azul*).
- Porcentaje de tiempo que el procesador está atendiendo interrupciones hardware (*Línea morada fina*).
- Porcentaje de tiempo del procesador en modo usuario (*Línea verde*).

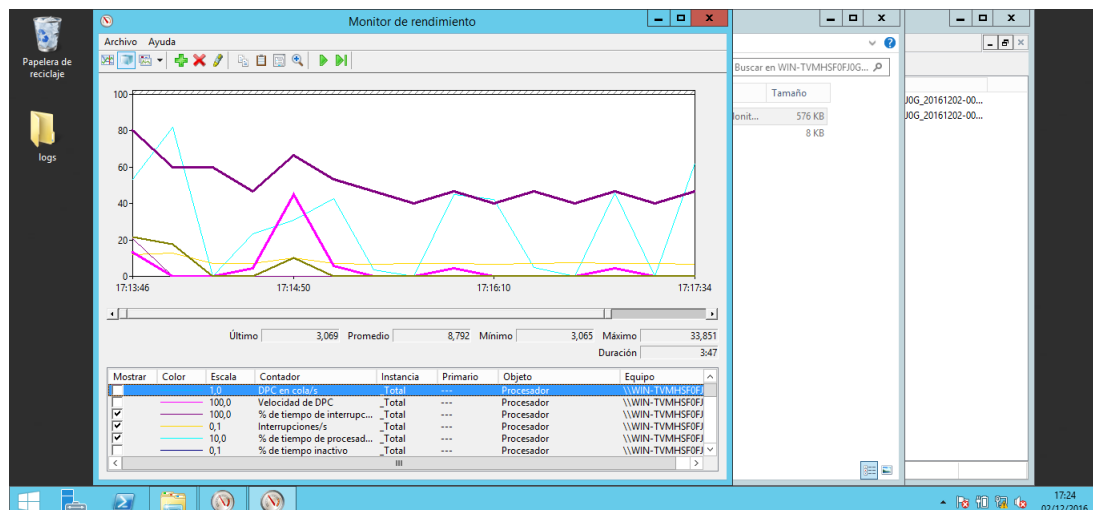


Figura 5.12: Gráfica del recopilador de datos con una selección de parámetros.

Durante los minutos que el recopilador estuvo tomando datos, una vez que revisé que la carpeta de logs se encontraba en su sitio y se estaba guardando algún dato, me puse a utilizar la máquina anfitriona para terminar de explicar la cuestión anterior.

En la gráfica podemos apreciar los primeros 3 minutos y 48 segundos. Podemos observar que hay un uso bajo del procesador en modo usuario (probablemente mientras estoy comprobando que el directorio existe y que se ha escrito algo) y luego no es usado (mientras me encuentro en la máquina anfitriona redactando la cuestión anterior). Las operaciones de escritura son altas cuando se inicia el monitor pero vemos que al final se va estabilizando probablemente y se deben a la escritura del log del monitor. También llama la atención el hecho de que a las 17:14:50 se encuentra el máximo absoluto en la gráfica con respecto a ambas medidas de escritura y luego ambas se acaban estabilizando, es posible que se estuviese creando por primera vez el archivo de log y se tuviese que guardar más información de la habitual. Podemos ver que el número de interrupciones hardware es bastante estable y excepto al principio el porcentaje de tiempo que el procesador dedica a dichas interrupciones es prácticamente inapreciable. Por último, observamos la línea de tiempo que el procesador está activo, puesto que no estamos usando la máquina virtual hemos de suponer o que los picos en éste se deben al *overhead* que produce la recopilación de todos estos datos o al uso de la máquina anfitriona.

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

Tras entrar en la página web encontramos los siguiente:

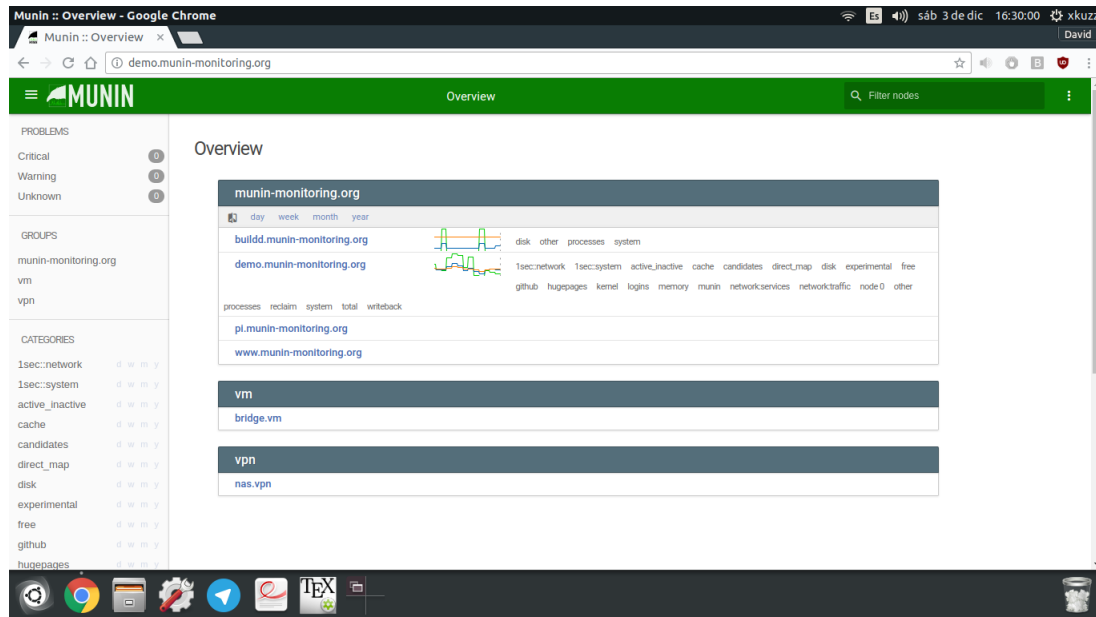


Figura 6.1: Página de inicio de la demo de munin.

A la izquierda podemos un panel en el que se nos informan de posibles problemas, disponemos de un filtro para los nodos en el menú superior y en el centro de la imagen tenemos una serie de grupos, de los cuales sólo el primero de ellos contiene algo de información. Primero vamos a visitar la información del sitio `builddd.munin-monitoring.org` pulsando en el mismo. Si pulsamos en la parte superior en la pestaña *processes* podemos observar información relativa a los procesos relacionados con ese sitio.



Figura 6.2: Monitorización de procesos en Munin.

Podemos observar como mientras que existen mayores diferencias entre número de hebras que las ligeras diferencias que hay en número de procesos. Y que hay una diferencia considerable de carga del sitio durante el fin de semana anterior con respecto a los días laborales.

Ahora vamos a pasar a ver un parámetro del otro sitio, para cambiar de sitio pulsamos en el menú que podemos encontrar en la parte superior con el nombre de la url y escogemos el sitio `demo.munin-monitoring.org`. En este vamos a seleccionar el disco como parámetro a analizar. Dentro de esta zona encontramos información sobre la latencia de disco, el tiempo de servicio de entrada/salida, el uso del sistema de archivos o el número de operaciones de entrada/salida por segundo, en el que nos vamos a centrar.

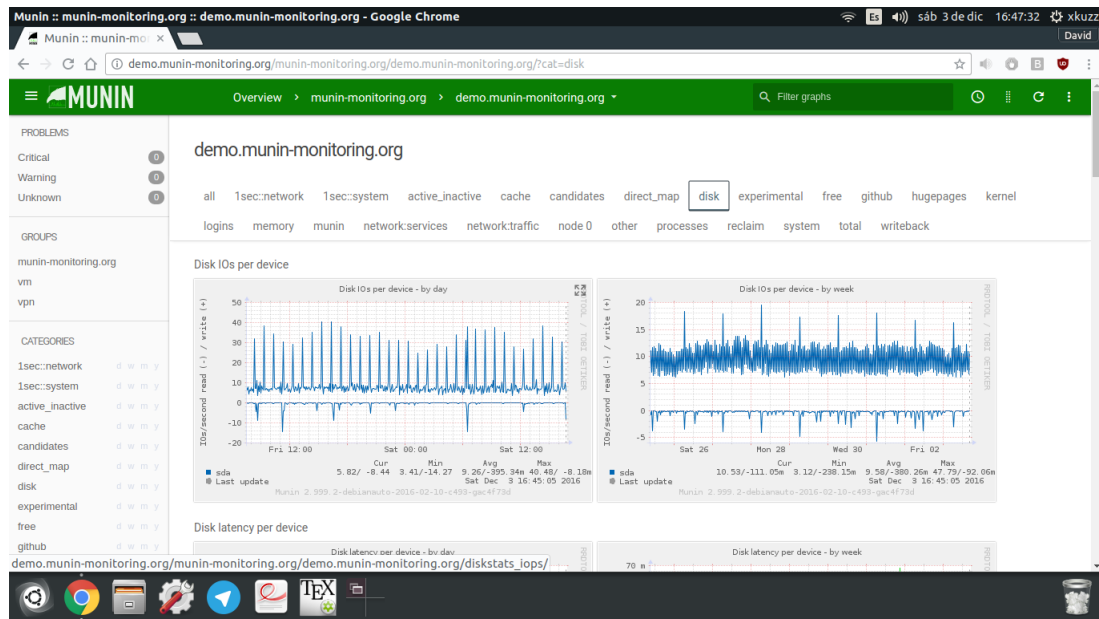


Figura 6.3: Monitorización de disco en Munin.

Podemos observar una serie de periódicas y de similares dimensiones en las escrituras en disco a lo largo de un día (escrituras de log, tareas temporales de 1 o 2 horas) y que son consecuentes también en la semana, aunque parece ser que al inicio de cada día se ejecutan más escrituras (tareas diarias). Las lecturas son más irregulares y las diferencias probablemente se deban al uso humano del sistema en vez de cualquier procedimiento automático.

Para acabar vamos a mirar otro parámetro más de este mismo sitio así que en la parte superior vamos a seleccionar la pestaña memory. Dentro esta zona podemos gráficas sobre el uso de swap, la fragmentación externa, el uso de memoria virtual y el uso de memoria física en el que nos vamos a centrar.

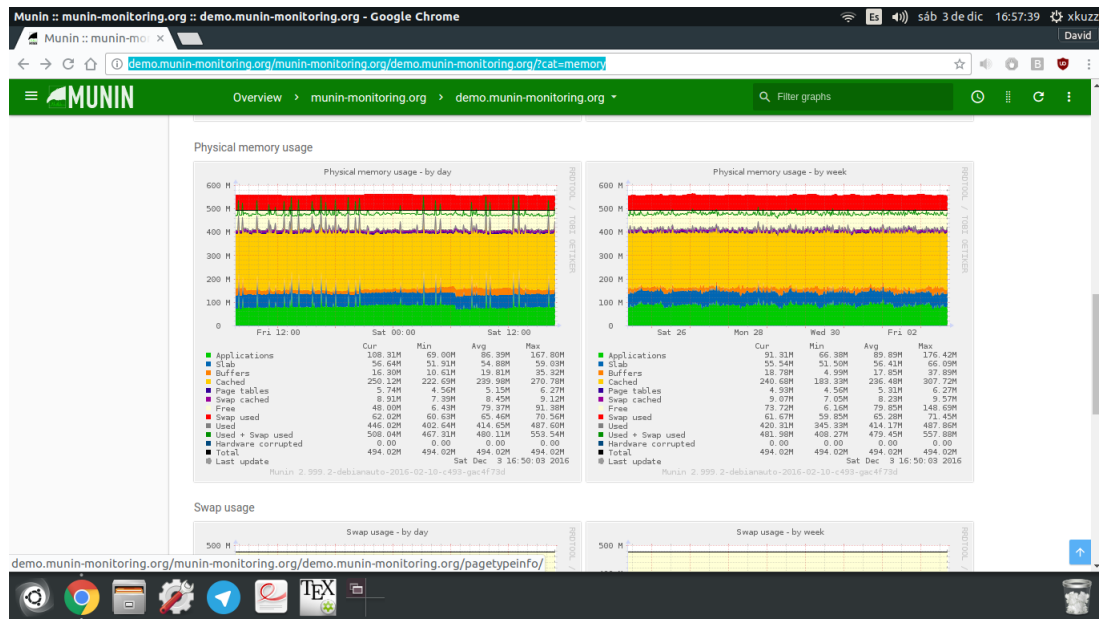


Figura 6.4: Monitorización de memoria en Munin.

Podemos observar que *Munin* nos ofrece una gráfica muy detallada sobre el uso de memoria, podemos observar cuanto memoria se dedica a cosas como aplicaciones, memoria que se encuentra en caché, memoria libre, etc. Vemos que durante el día existe un uso más o menos regular de memoria con algunos picos y que en las horas cercanas a las 0:00 del sábado observamos un intervalo de un menor uso relativo respecto a el resto del intervalo del día monitorizado. En el gráfico semanal podemos observar que apenas hay variaciones significativas en el uso. Es interesante que el uso de memoria en el servidor está limitada a 500 M y normalmente está memoria es utilizada prácticamente por completo todos los días y a diario, indicando que quizás se ha ajustado el funcionamiento del servidor y puesto sólo la cantidad de recursos de memoria justo y necesario para que junto con el swap el sistema funcione correctamente.

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

He escogido el segundo artículo propuesto.[13]

Tras indicar el autor que *strace* nos herramienta para depuración ni para programadores sino para administradores de sistemas nos presenta tres casos diferentes en los que podemos usar dicha herramienta. El primero de ellos es un simple ejemplo introductorio en el que se hace *cat* sobre un archivo, al ejecutar *strace* sobre ese ejemplo obtenemos todas las

llamadas del sistema asociadas. El autor destaca dos líneas, la línea de *open* que nos indica qué fichero se ha abierto y cómo se ha abierto (lo cual puede resultar más útil para llamadas de las que desconocemos que archivos se van a abrir) y la línea de *read* en la que sólo se muestran los primeros 32 caracteres que se van a leer pero indicando cuantos caracteres se leen en total.

El segundo ejemplo nos ayuda a encontrar archivos de “logs” cuando no sabemos dónde se encuentran, para ello reinicia un servicio y hace *strace* sobre todas las bifurcaciones posibles que se creen y guardando la información en un archivo de salida. Tras buscar “log” con la herramienta *grep* encuentra las llamadas *open* con las que se habían abierto dichos archivos con sus localizaciones.

Por último, nos enseña un ejemplo en el que detecta un error al iniciar apache tras buscar el código de error que devuelven las llamadas del sistema (-1) con *grep* para delimitar las causas de que apache no arranque. En ese ejemplo, el log de apache estaba puesto como inmutable y por tanto al realizar *open* con intención de escribir se le denegaba el acceso.

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

Vamos a realizar un script en Python utilizando el módulo *cProfile* [14]. El script recibirá un número como parámetro y ejecutará dos funciones. Una calculará la sumatoria de todos los múltiplos de 3 menos la sumatoria de los que no sean múltiplos de 3. La segunda función calculará el productorio de todos los números no múltiplos de 3 entre el productorio de múltiplos de 3. En ambos casos se empieza en el 1 y el último número a considerar será el pasado como parámetro. El script en Python sería el siguiente:

```
import cProfile, pstats, StringIO, sys
# Sumatoria de multiplos de 3 menos sumatoria de no multiplos de 3 hasta el rango dado
def sumatoria(sup):
    sumatoria = 1
    for num in range(2, sup):
        if (num % 3 == 0):
            sumatoria += num
        else:
            sumatoria -= num
    return sumatoria

# Productorio de no multiplos de 3 dividido entre productorio multiplos de 3
def productorio(sup):
    producto = 1.0
    for num in range(2, sup):
        if (num % 3 == 0):
```



```

        producto /= num
    else:
        producto *= num
    return producto

def main(argv):
    superior = int(argv[1])
    pr = cProfile.Profile()
    pr.enable()
    print sumatoria(superior)
    print productorio(superior)
    pr.disable()
    s = StringIO.StringIO()
    sortby = 'cumulative'
    ps = pstats.Stats(pr, stream=s).sort_stats(sortby)
    ps.print_stats()
    print s.getvalue()

if __name__ == "__main__":
    main(sys.argv)

```

```

toProfile.py
import cProfile, pstats, StringIO, sys
# Sumatoria de multiples de 3 menos sumatoria de no multiples de 3 hasta
el rango dando
def sumatoria(sup):
    sumatoria = 1
    for num in range(2, sup):
        if (num % 3 == 0):
            sumatoria += num
        else:
            sumatoria -= num
    return sumatoria

# Productorio de no multiples de 3 dividido entre productorio multiples
de 3
def productorio(sup):
    producto = 1.0
    for num in range(2, sup):
        if (num % 3 == 0):
            producto /= num
        else:
            producto *= num
    return producto

def main(argv):
    superior = int(argv[1])
    pr = cProfile.Profile()
    pr.enable()
    print sumatoria(superior)
    print productorio(superior)

Python  Anchura del tabulador: 8  Ln 14, Col 17  INS

```

```

dcr@centos:~
[07-12-2016 19:42:55 dcr@centos]~ $python toProfile.py 10000
-16658332
inf
5 function calls in 0.006 seconds

Ordered by: cumulative time

ncalls  tottime  percall  cumtime  percall  filename:lineno(function)
1      0.003    0.003    0.003    0.003  toProfile.py:13(productorio)
1      0.002    0.002    0.003    0.003  toProfile.py:3(sumatoria)
2      0.000    0.000    0.000    0.000  {range}
1      0.000    0.000    0.000    0.000  {method 'disable' of '_lspro
f.Profiler' objects}

[07-12-2016 19:42:56 dcr@centos]~ $

```

Figura 8.1: Resultados del profiler cProfile de Python tras ejecutar el script.

Podemos observar como el profiler nos devuelve una tabla ordenada por tiempo acumulado de todas las funciones, vemos que la función que más ha tardado ha sido el productorio, seguido inmediatamente de la sumatoria. En la parte superior (antes de la tabla), además de los resultados obtenemos el tiempo total de ejecución y vemos que son 0.006 segundos. En la tabla se nos informa también del número de veces que se llama a una función, el

tiempo total y el tiempo acumulado y sus respectivas versiones por llamada a la función. Aunque no hay una diferencia muy grande entre el tiempo de ejecución de la sumatoria y el productorio de los resultados podemos deducir que el cuello de botella de nuestro programa tal y como está es el productorio y si queremos una mejor eficiencia esa es la función que debemos optimizar.

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).

Vamos a hacerlo a través de phpMyAdmin, primero vamos a crear una base de datos y una tabla e insertar una tupla en la misma tal y como podemos apreciar en las siguientes capturas de pantalla. [15]

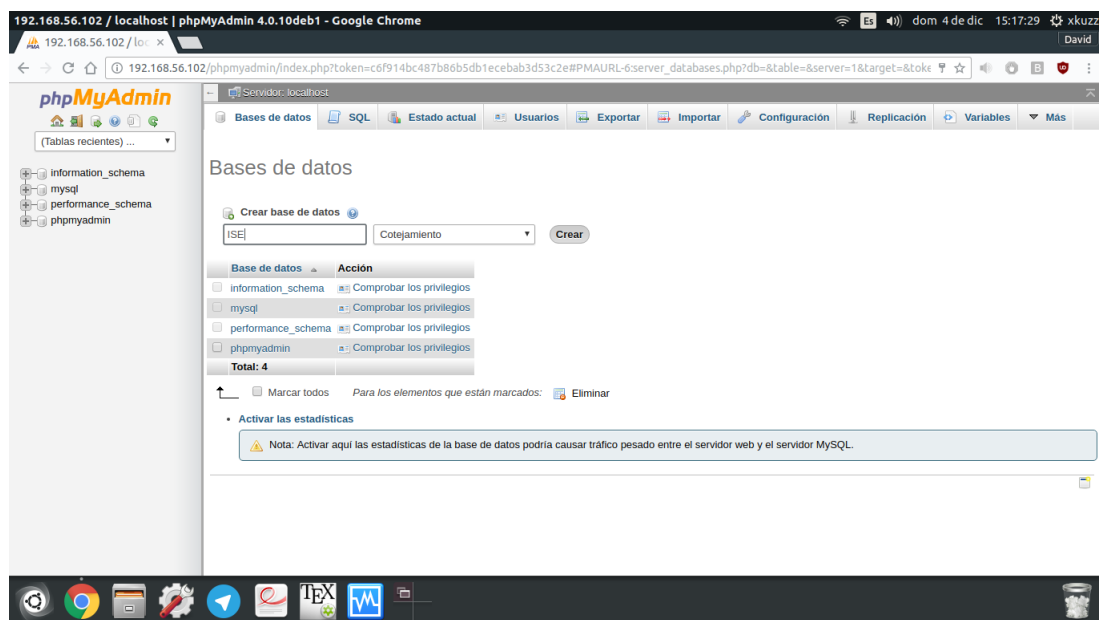


Figura 9.1: En *Bases de datos* creamos una nueva base de datos a la que llamamos ISE.

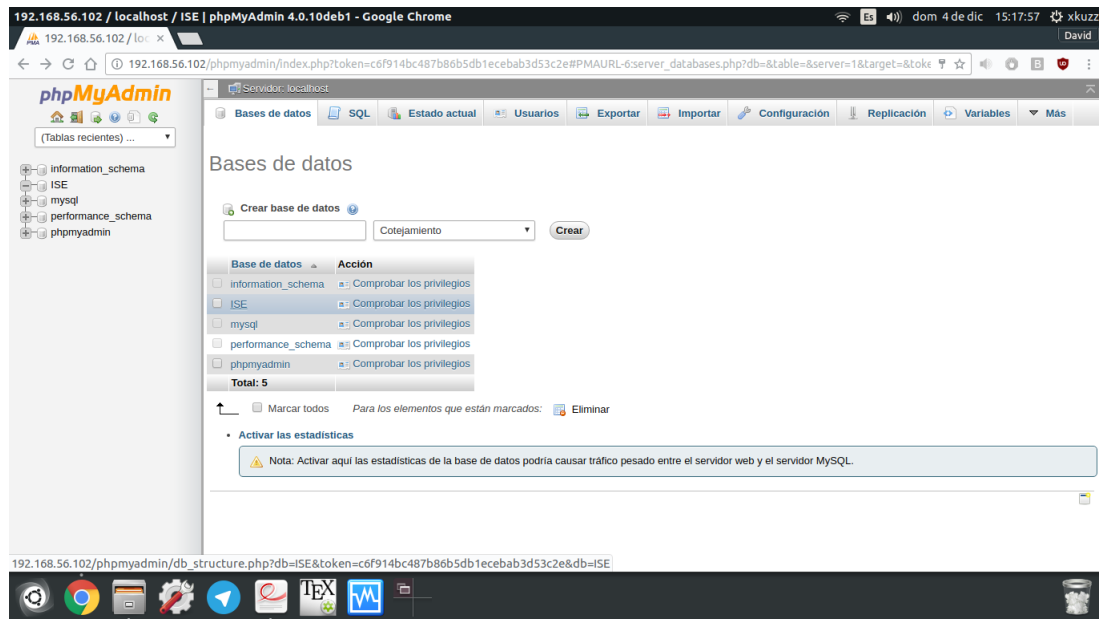


Figura 9.2: Pulsamos en la lista de bases de datos *ISE*.

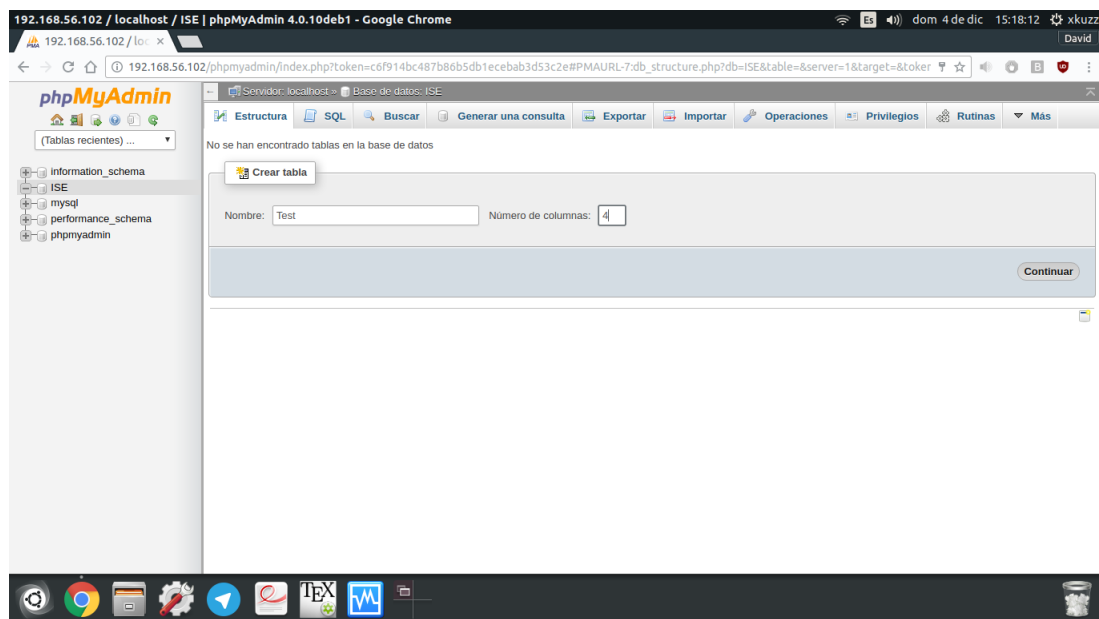


Figura 9.3: Creamos una nueva tabla de 4 columnas llamada Test y pulsamos en Enviar.

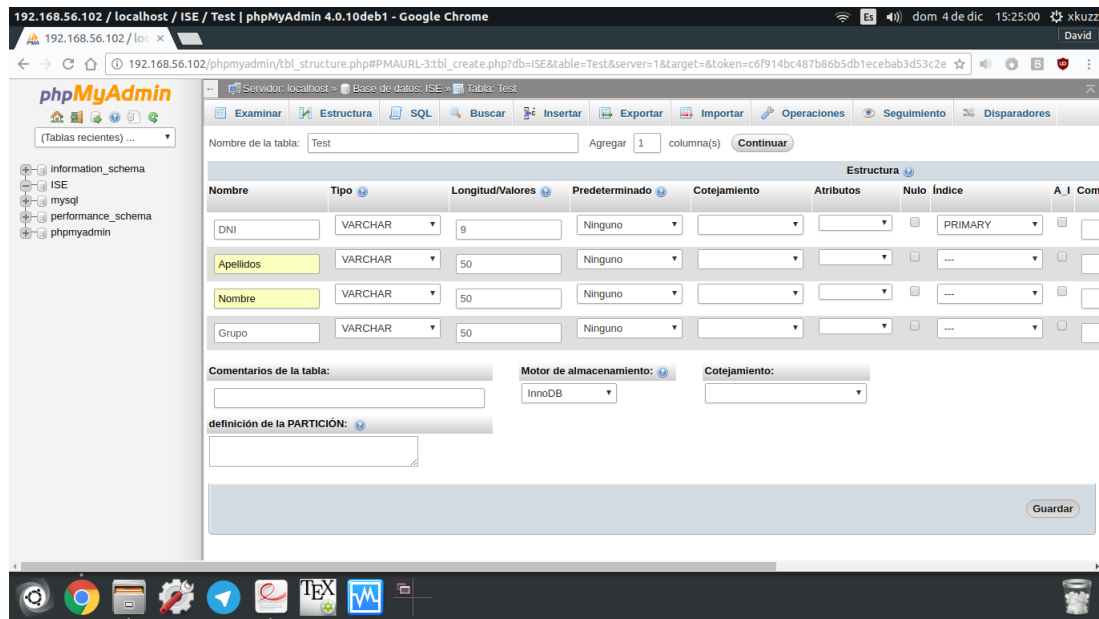


Figura 9.4: Creamos una nueva tabla con llave primaria un dni de 9 caracteres y el resto varchar de 50 caracteres (apellidos, nombre, grupo).

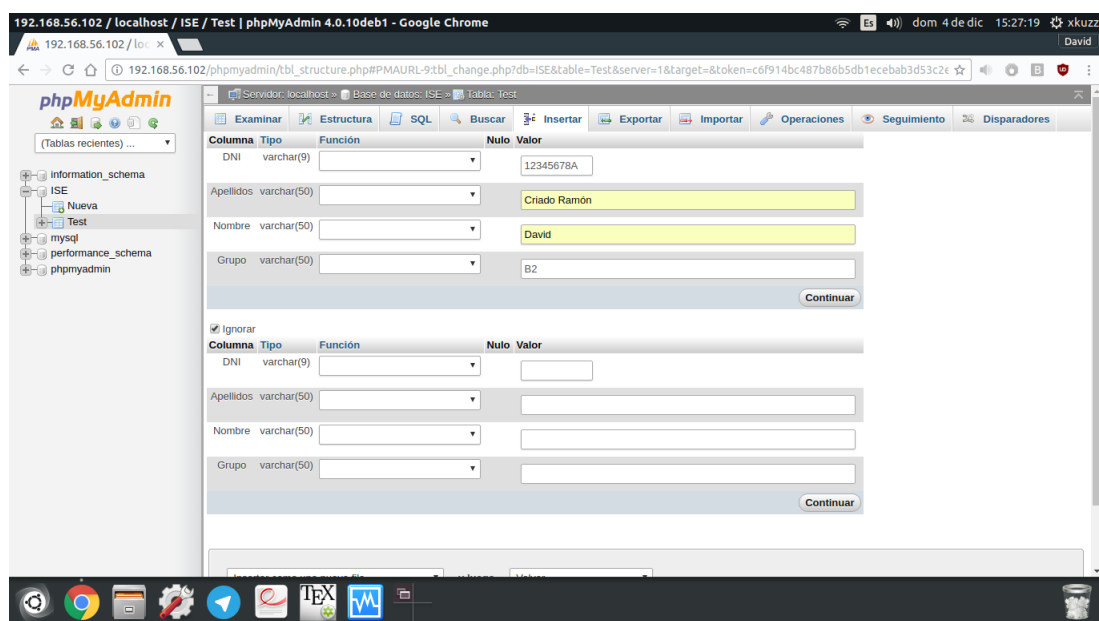


Figura 9.5: Insertamos una tupla válida en la tabla pulsando en el menú superior en *Insertar* y rellenando los campos.

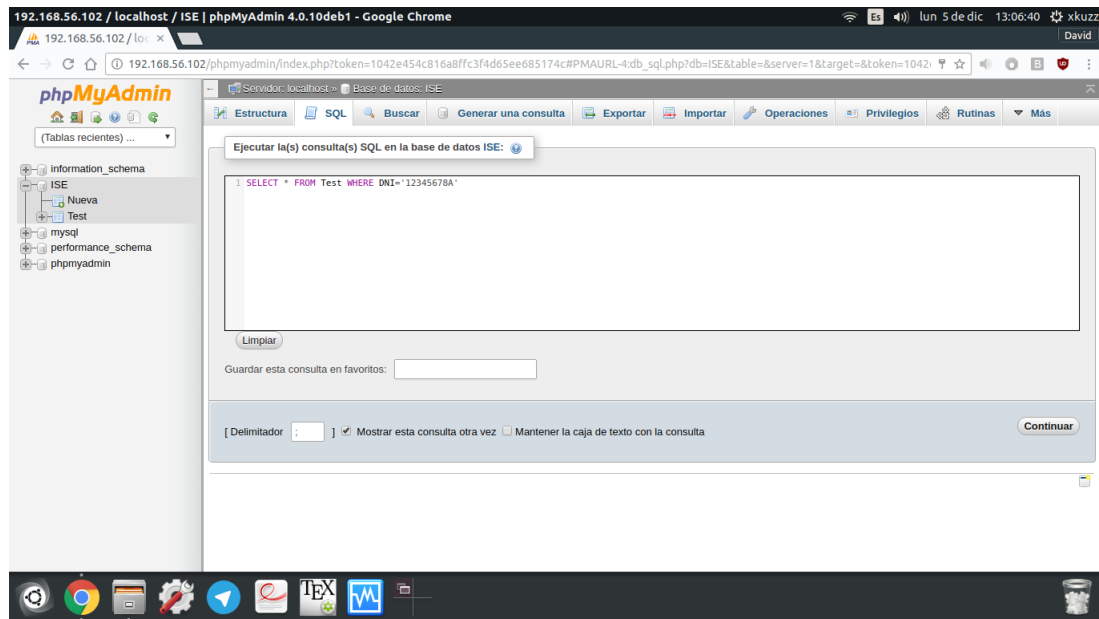


Figura 9.6: Pulsamos en la parte superior en SQL y realizamos la consulta. En mi caso he mostrado todos los datos de la tabla de aquellas personas que tuviesen el DNI de la tupla insertada.

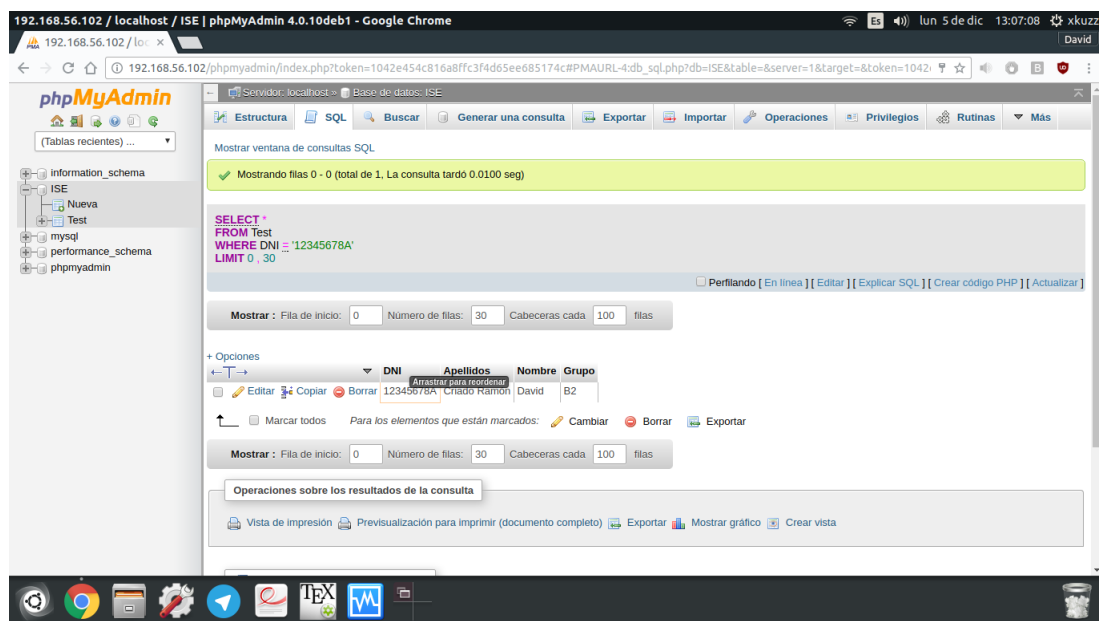


Figura 9.7: Podemos observar que el único resultado que obtenemos es la tupla que insertamos.

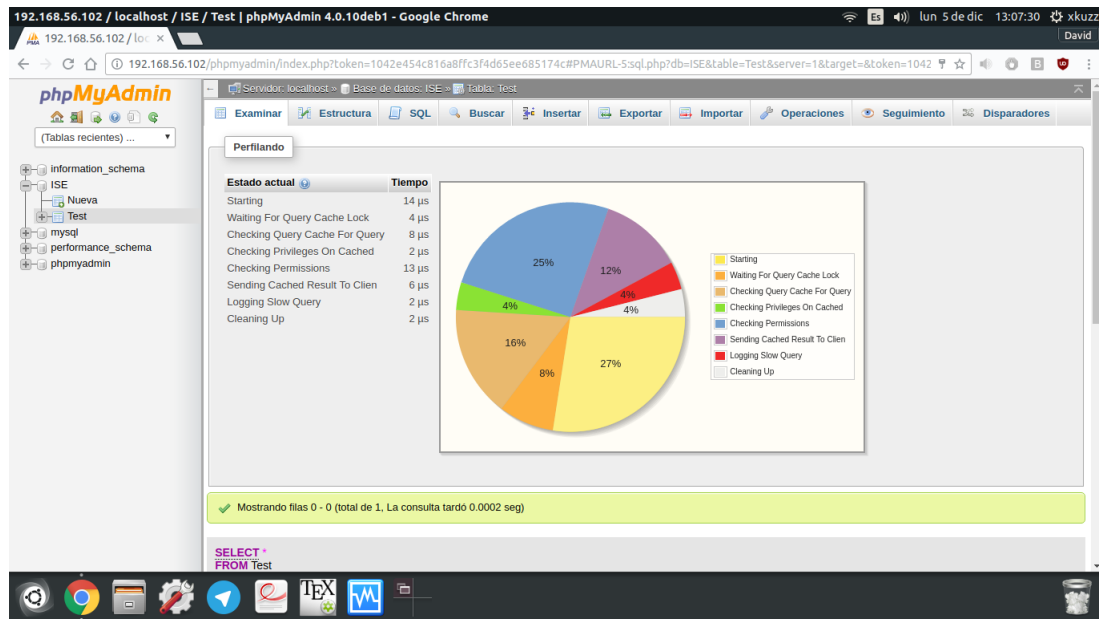


Figura 9.8: Tras pulsar en *Perfilando* aparece esta gráfica.

A la izquierda podemos observar una tabla en la que viene cada uno de los campos que analiza el *profiler* junto a su tiempo, los cuales se encuentran el rango de los microsegundos. A la derecha tenemos un diagrama de sectores asociados a dichos valores. Podemos observar que debido a la gran sencillez de la consulta la mayor parte del tiempo ha sido dedicada a iniciar y comprobar los permisos. Ambos juntos superan la mitad de los tiempos de la consulta. El siguiente trozo relativamente más grande de tiempo se ha dedicado a comprobar si la consulta estaba en caché y que comprobar los permisos que tenemos en los elementos en caché, registrar la consulta y limpiar han sido las operaciones más rápidas.

Referencias

- [1] “Página de manual para el comando “file” en Ubuntu Server 14.04.”
- [2] “Página de manual para el comando “gzip” en Ubuntu Server 14.04.”
- [3] “Página de manual para el archivo “dpkg.log” en Ubuntu Server 14.04.”
- [4] “Página de manual para el comando “apt-get” en Ubuntu Server 14.04.”
- [5] “Página de manual para el archivo “yum.conf” en CentOS 7.”
- [6] “Página de manual para el comando “logrotate” en Ubuntu Server 14.04.”
- [7] “Página de manual para el comando “cron” en Ubuntu Server 14.04.”

- [8] “Página de manual para el comando “crontab” en Ubuntu Server 14.04.”
- [9] “Página de manual para el comando “cron” en Centos 7.”
- [10] SystemSoft and Intel, “USB Common Class Specification.” http://www.usb.org/developers/docs/devclass_docs/usbccs10.pdf, 1997.
- [11] Microsoft, “Configurando la salida por pantalla del monitor de rendimiento (en Inglés).” [https://technet.microsoft.com/en-us/library/cc722300\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/cc722300(v=ws.11).aspx), consultado el 30 de noviembre de 2016.
- [12] Microsoft, “Creando un recopilador de datos manualmente (en Inglés).” [https://technet.microsoft.com/en-us/library/cc766404\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/cc766404(v=ws.11).aspx), consultado el 1 de diciembre de 2016.
- [13] L. Latham, “Sysadmin Tips and Tricks - Using strace to Monitor System Calls.” <http://blog.softlayer.com/2013/sysadmin-tips-and-tricks-using-strace-to-monitor-system-calls>, consultado el 6 de diciembre de 2016.
- [14] Python, “Documentación oficial de Python: cProfile (en Inglés).” <https://docs.python.org/2/library/profile.html#module-cProfile>, consultado el 7 de diciembre de 2016.
- [15] M. Delisle, “Mastering phpMyAdmin 3.4 for Effective MySQL Management (Capítulo 4 1ª Edición),” 2012.