

# **Memoria Practica 1**

## Administración de Servidores

Almudena García Jurado-Centurión

# Índice

1. Crear máquina virtual
  1. Datos de la instalación
    1. Configuración básica
    2. Memoria y BIOS
    3. Procesador
    4. Almacenamiento
  2. Instalando CentOS 6.9
  3. Iniciando CentOS y primeras configuraciones
2. Activando el reenvío de puertos al sistema anfitrión
3. Probando vmstat
4. Aumentando la RAM
5. Analizando el efecto de la entrada/salida de un programa en ejecución
6. Analizando el efecto en la creación de procesos de otro programa en ejecución
7. Analizando impacto en memoria de la ejecución de un proceso
8. Filtrando los datos del comando sar
9. Conclusiones

## 1. Crear maquina Virtual

Creamos una maquina virtual de CentOS 6.9 i386 con VirtualBox, usando los siguientes parámetros:

Memoria RAM = 1000 MB

Número de CPUs = 1

Tamaño de disco = 20GB

### - Datos de la instalación

- **Configuración básica**

La maquina virtual esta configurada como un sistema Linux, Red Hat. Por motivos de compatibilidad, vamos a usar la versión 32 bits del sistema.

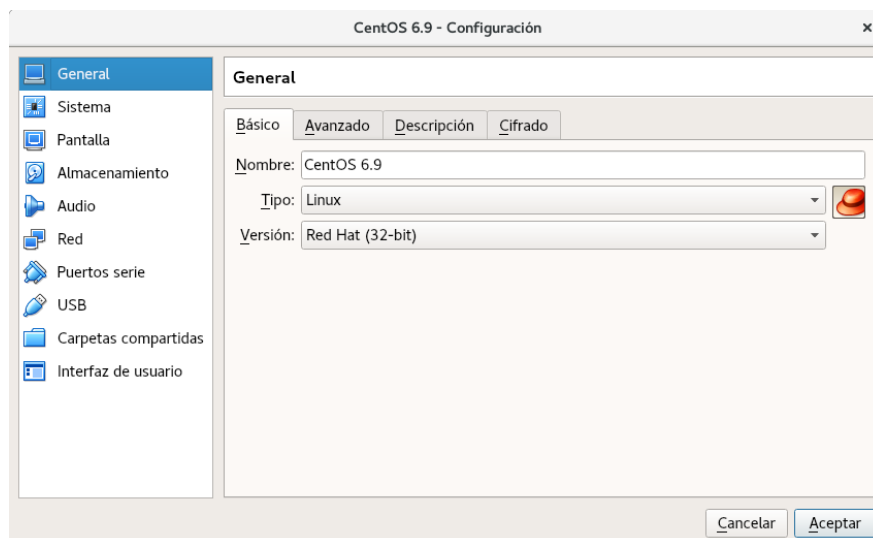


Figura 1: Configuración básica de la máquina virtual

- **Memoria**

y

**BIOS**

Dado que la maquina virtual es de 32 bits, no podemos asignar mas de 4 GB de memoria RAM.

En este caso, puesto que el consumo del sistema es bastante bajo, hemos optado por asignar únicamente 1GB de RAM.

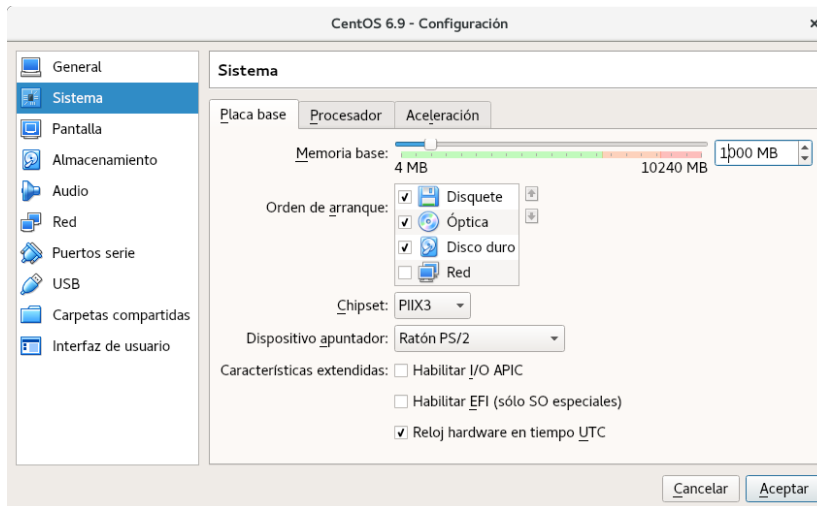


Figura 2: Configuración de memoria y BIOS

- **Procesador**

En cuanto al procesador, le hemos asignado una sola CPU sin límite de ejecución

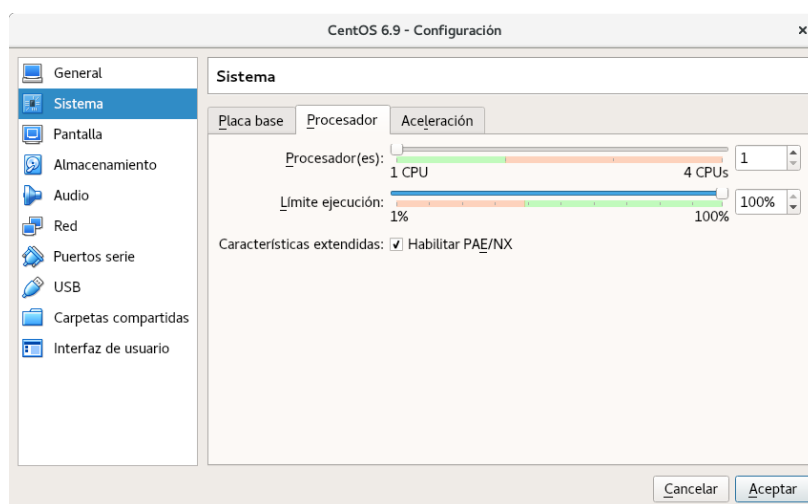


Figura 3: Configuración de CPU

- **Almacenamiento**

Hemos creado un solo disco duro SATA de 20 GB.

El disco duro esta configurado como almacenamiento dinámico, de forma que su tamaño inicial sea menor del asignado inicialmente, el cual se irá incrementando conforme se vaya necesitando mas capacidad.

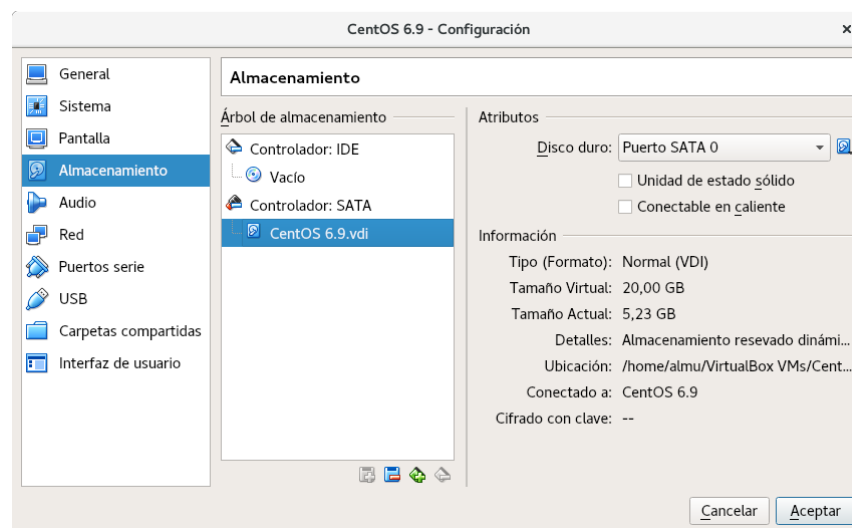


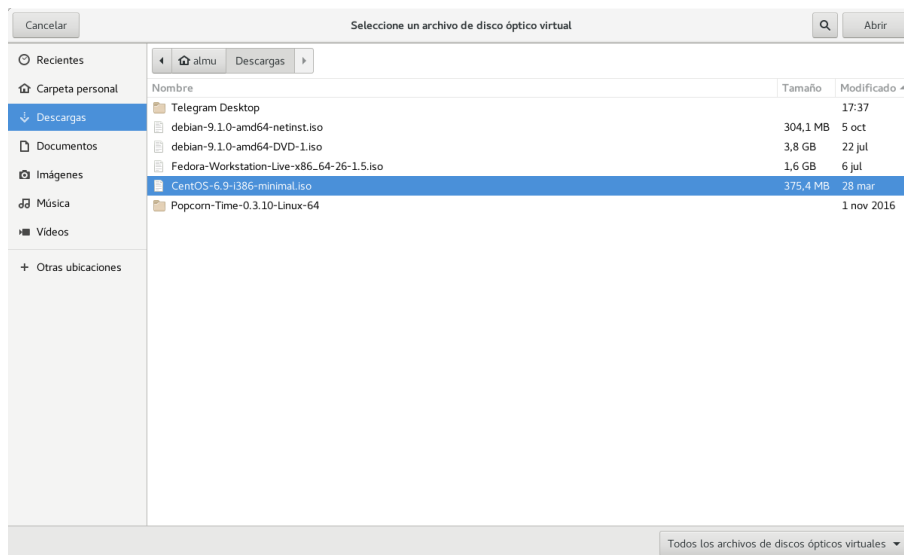
Figura 4: Configuración de disco duro y almacenamiento

## - Instalando CentOS 6.9

Arrancamos la maquina virtual con la imagen iso de CentOS.

Para ello, con la máquina iniciada, pulsamos en el menú Dispositivos → Unidades Ópticas → Seleccionar imagen de disco. Nos mostrará un explorador donde podremos buscar la imagen iso descargada, y seleccionarla.

Buscamos el archivo con nombre “CentOS-6.0-i386-minimal.iso” y pulsamos en Abrir.



Reiniciamos la maquina e iniciamos el proceso de instalación. Para ello, seleccionamos la primera opción y pulsamos Enter.

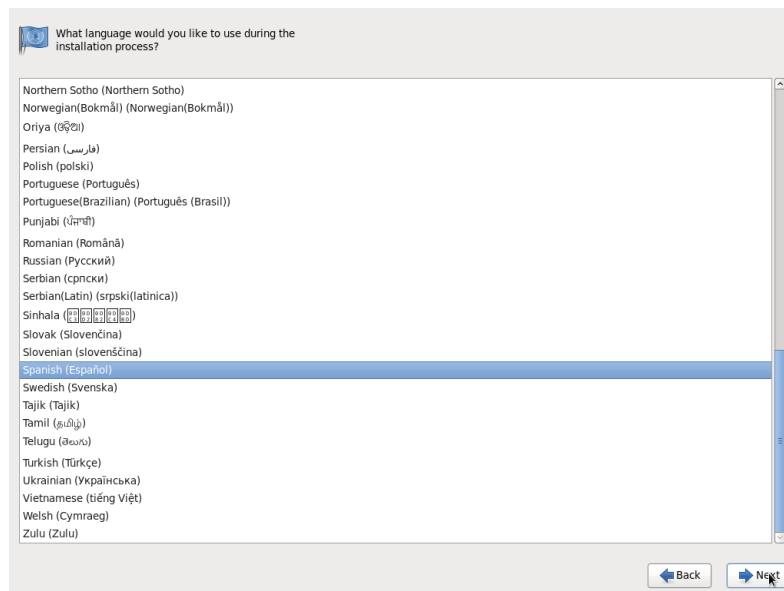


El proceso de instalación es sencillo, solo hay que seguir los pasos que se van indicando en el asistente. El usar una máquina virtual nos facilita el proceso, puesto que no tenemos que preocuparnos por el soporte de hardware.

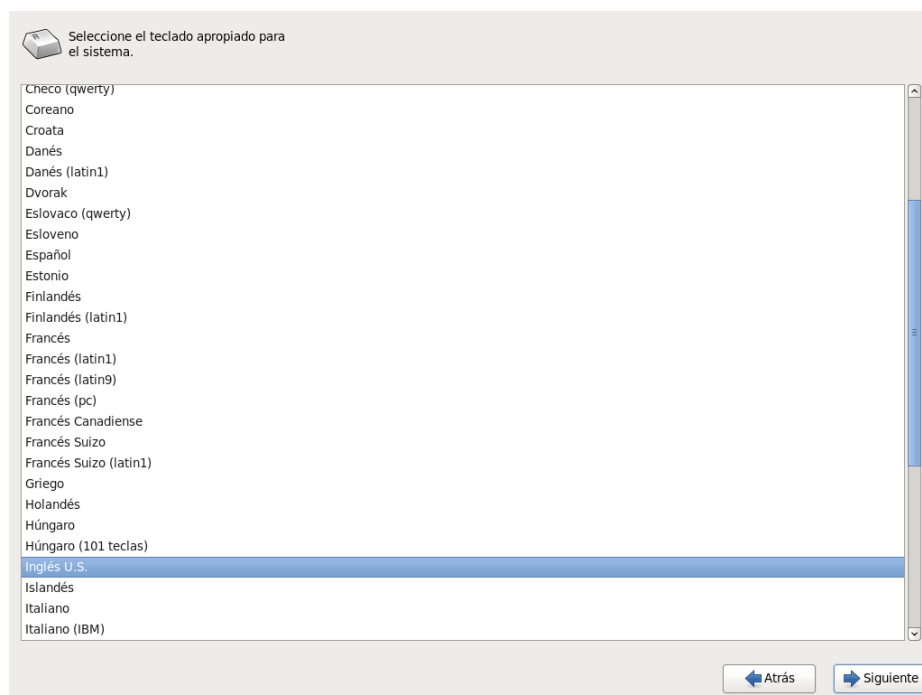
Tras pulsar en la primera opción, y tras hacer el test de discos, se nos mostrará esta ventana. Pulsamos en Next



Nos preguntará por el idioma del instalador. En este caso, hemos seleccionado “Spanish (Español)”

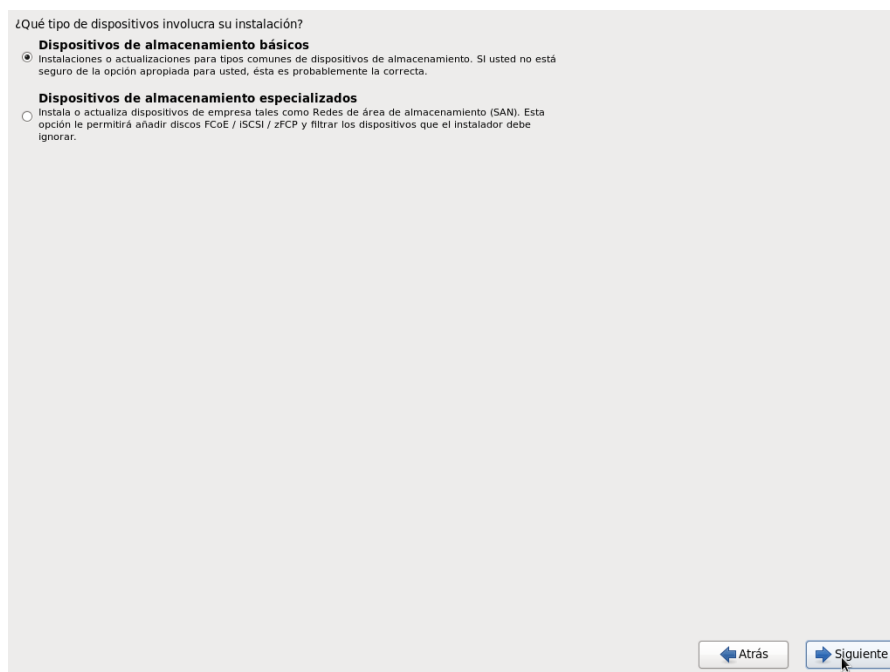


Seleccionamos el mapa de teclado. Dado que mi portátil usa teclado inglés, he seleccionado “Inglés US”



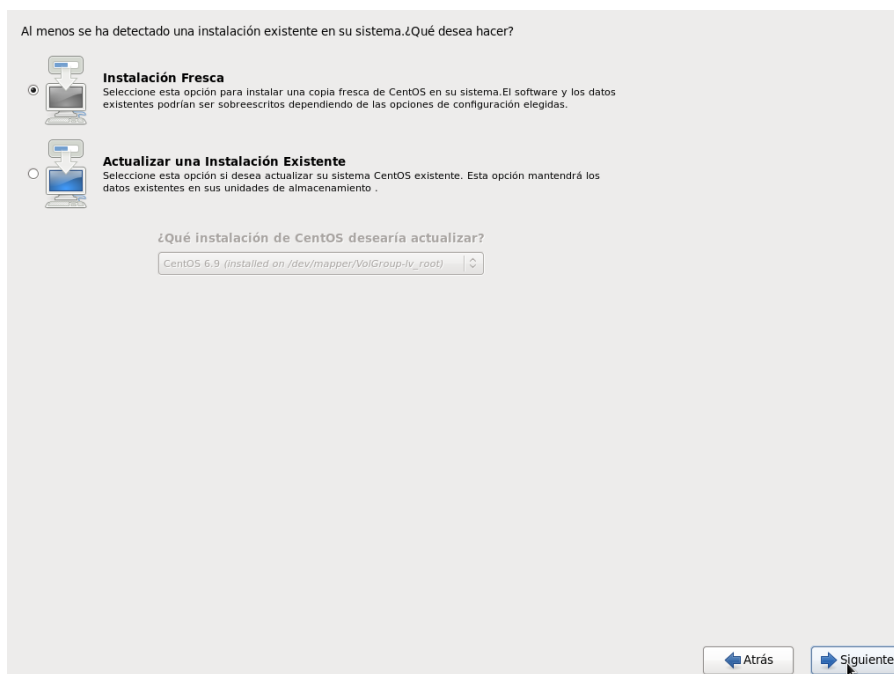


Seguimos la instalación, y pasamos al particionado. Como nuestro disco duro es SATA, se considera que es un dispositivo de almacenamiento básico, así que seleccionamos la primera opción.

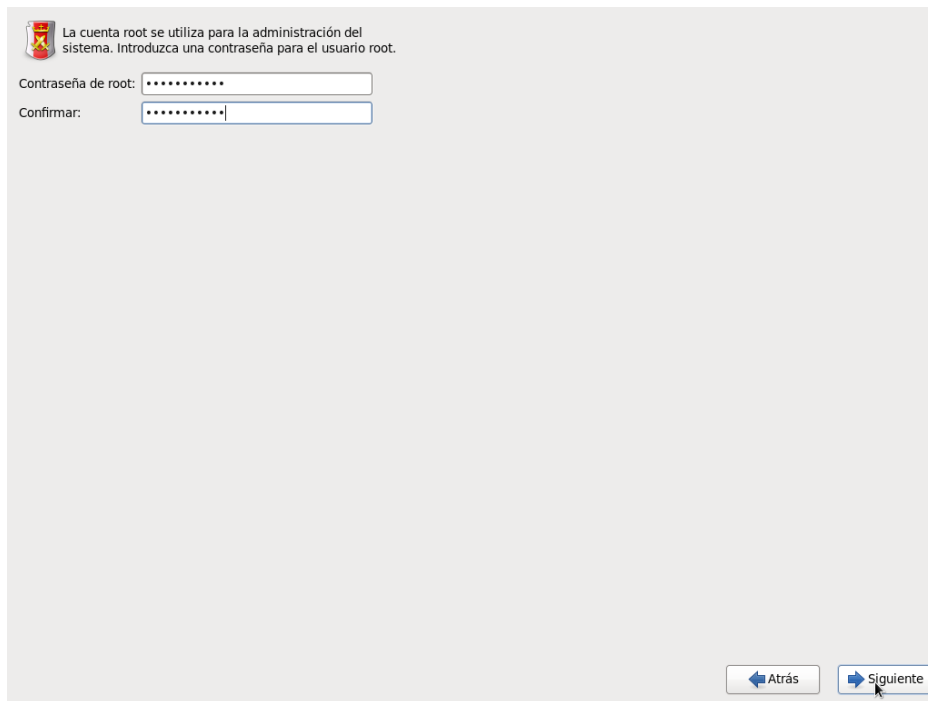


Seguidamente, el instalador nos preguntará el tipo de instalación que queremos realizar. En mi caso, como estoy reinstalando el sistema, me da la opción de actualizar la instalación ya existente.

Como yo quiero reinstalar el sistema completo, ignoro esa opción y pulso en “Instalación Fresca”



En el siguiente paso, nos pedirá que asignemos una contraseña al usuario root. Escribimos la clave, la confirmamos escribiéndola de nuevo y pulsamos en Siguiente




La cuenta root se utiliza para la administración del sistema. Introduzca una contraseña para el usuario root.

Contraseña de root:

Confirmar:

[< Atrás](#) [Siguiente >](#)

Se iniciará el proceso de formateo y particionamiento

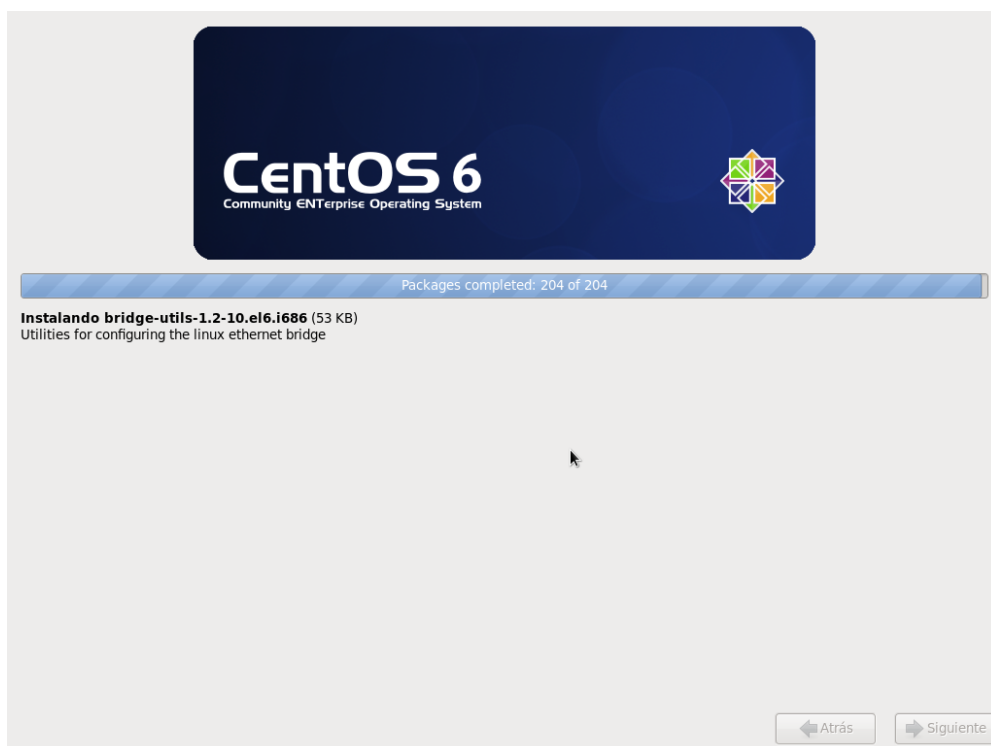
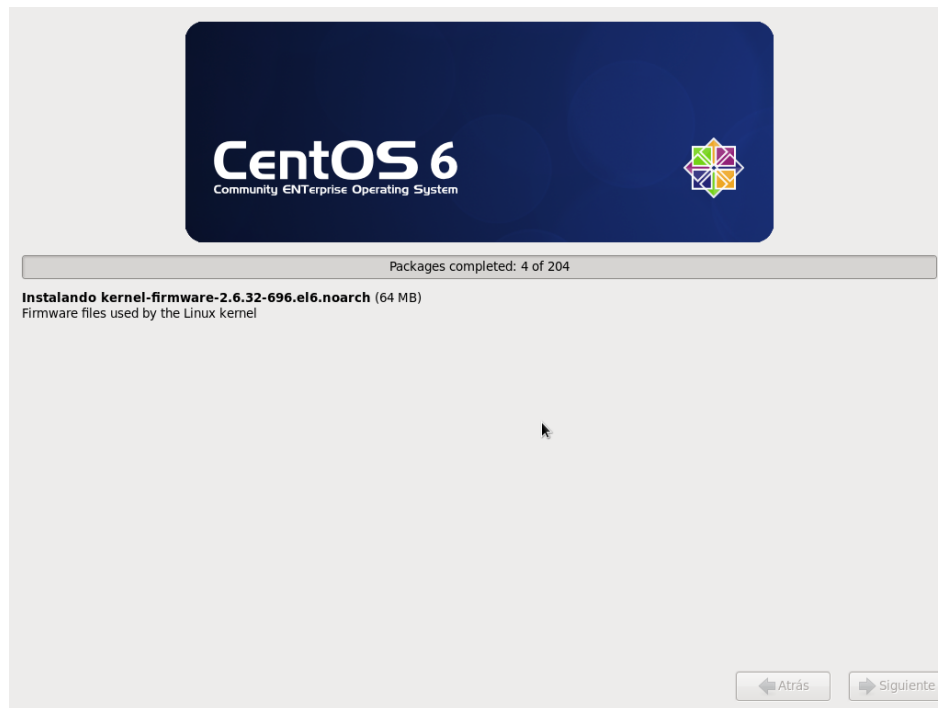


**Formateo**

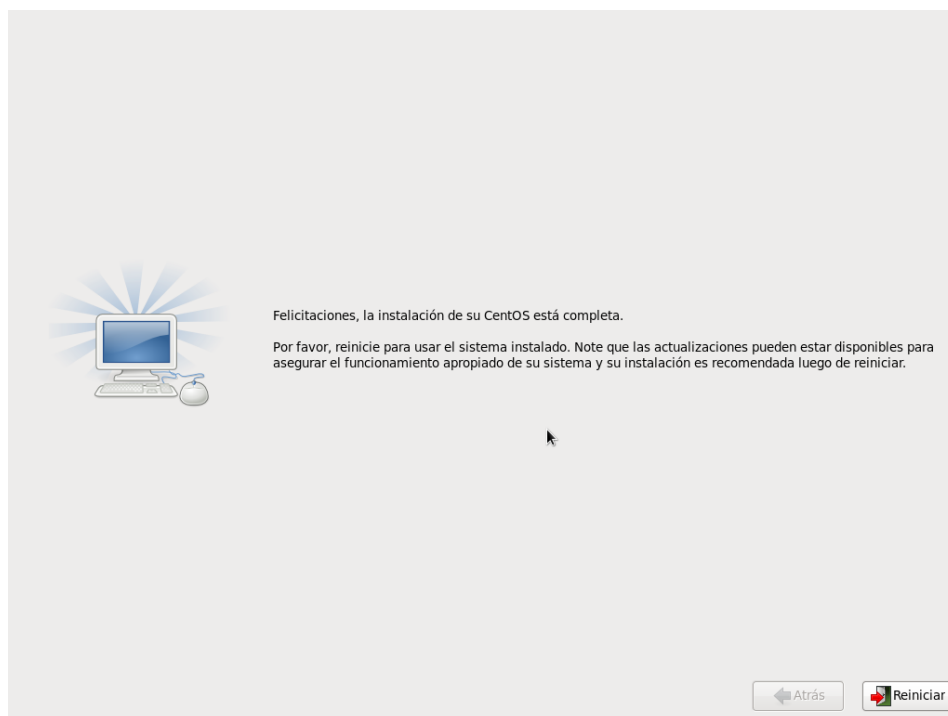
Creando sistema de archivos ext4 en /dev/mapper/VolGroup-lv\_root

[< Atrás](#) [Siguiente >](#)

Una vez formateado, comenzará la instalación del sistema en el disco duro

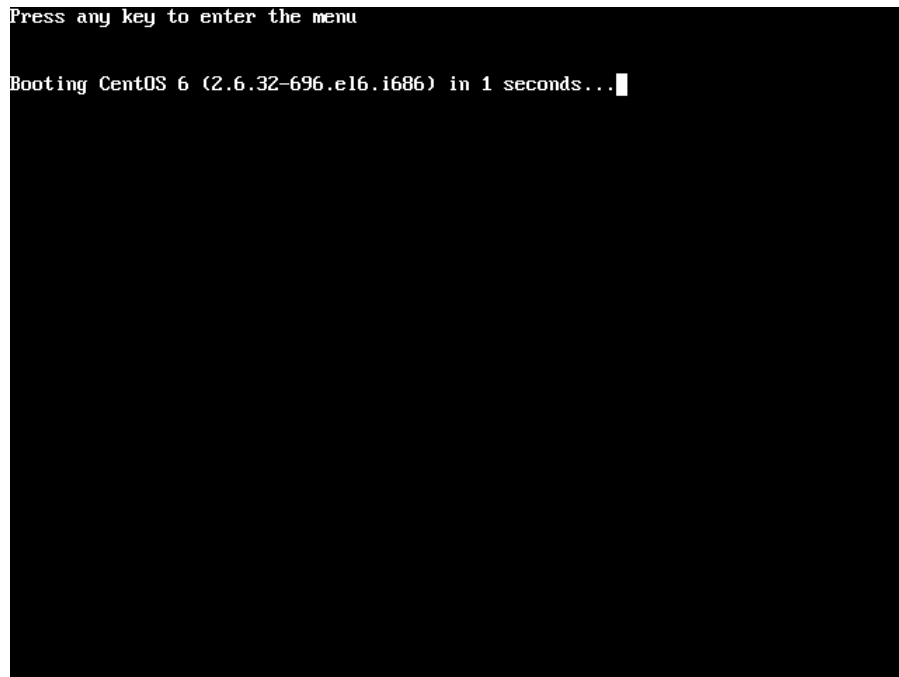


Tras este último paso, CentOS estará instalado en el disco duro, y podremos iniciar el sistema pulsando en “Reiniciar”



## - Iniciando CentOS y primeras configuraciones

Tras reiniciar la maquina, se iniciará el sistema CentOS recién instalado.



El sistema nos mostrará la tty con la pantalla de login. Dado que no hemos creado ningún usuario adicional, hemos de iniciar como root.

En *login* escribimos **root**, y pulsamos enter. Nos pedirá la password, así que escribimos la clave que definimos durante la instalación.

Si todo ha ido bien, se nos abrirá el interprete de comandos de root (identificado por su característico prompt #), tras lo cual podremos empezar a trabajar con el sistema

```
CentOS release 6.9 (Final)
Kernel 2.6.32-696.el6.i686 on an i686

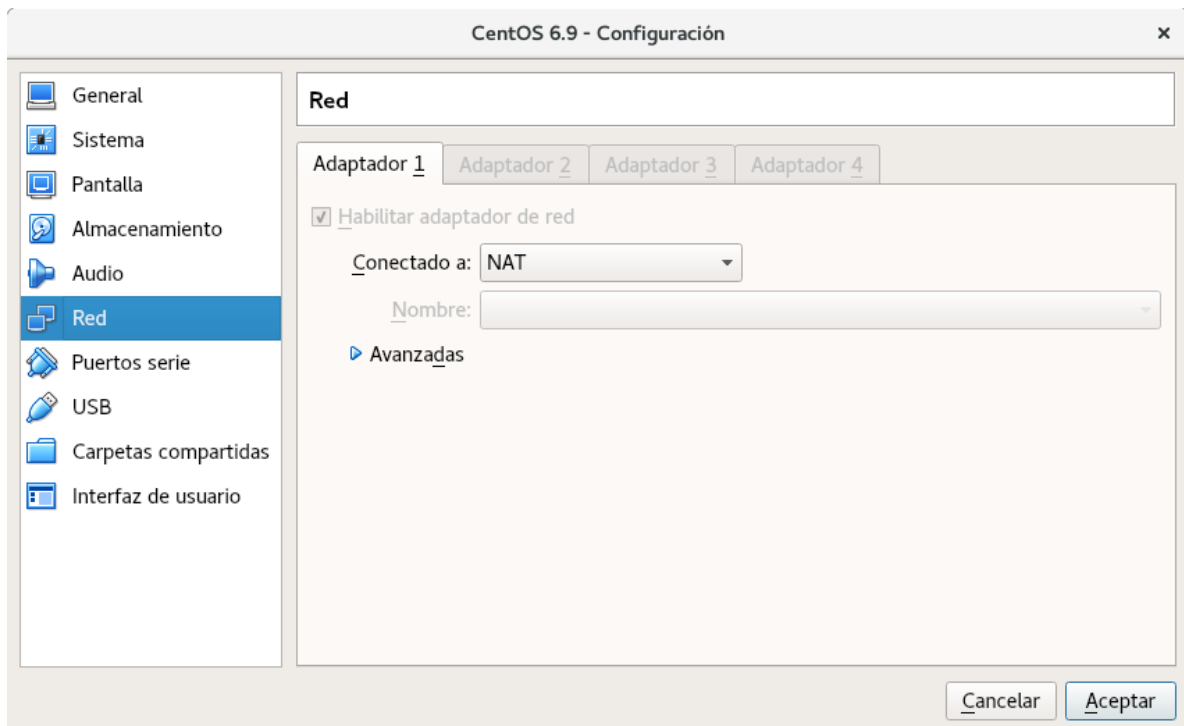
localhost login: root
Password:
[root@localhost ~]# _
```

Una vez iniciada la sesión, probamos la conexión de red lanzando un ping a una URL conocida.

```
CentOS release 6.9 (Final)
Kernel 2.6.32-696.el6.i686 on an i686

localhost login: root
Password:
[root@localhost ~]#
[root@localhost ~]# ping google.es
ping: unknown host google.es
[root@localhost ~]# _
```

Comprobamos que no tenemos conexión de red, así que revisamos la configuración de red de la maquina virtual para ver que no haya ningún problema.



Tras revisar la configuración de la maquina virtual, comprobamos que el cable esta conectado correctamente en modo NAT, así que el problema debe provenir del propio sistema.

Empezamos revisando el estado de las interfaces con *ifconfig*.

```
CentOS release 6.9 (Final)
Kernel 2.6.32-696.el6.i686 on an i686

localhost login: root
Password:
[root@localhost ~]#
[root@localhost ~]# ping google.es
ping: unknown host google.es
[root@localhost ~]# ifconfig
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:8 errors:0 dropped:0 overruns:0 frame:0
            TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:600 (600.0 b)  TX bytes:600 (600.0 b)

[root@localhost ~]# _
```

Vemos que la única interfaz activa es *lo* , correspondiente al bucle local, y que la interfaz ethernet no esta activa.

Para levantar la interfaz, usamos el comando *ifup*. Dado que no existen mas tarjetas de red, la interfaz ethernet se llamará *eth0*.

```
CentOS release 6.9 (Final)
Kernel 2.6.32-696.el6.i686 on an i686

localhost login: root
Password:
[root@localhost ~]#
[root@localhost ~]# ping google.es
ping: unknown host google.es
[root@localhost ~]# ifconfig
lo                Link encap:Local Loopback
                  inet addr:127.0.0.1  Mask:255.0.0.0
                  inet6 addr: ::1/128 Scope:Host
                  UP LOOPBACK RUNNING MTU:65536 Metric:1
                  RX packets:8 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:0
                  RX bytes:600 (600.0 b)  TX bytes:600 (600.0 b)

[root@localhost ~]# ifup eth0

Determinando la información IP para eth0... hecho.
[root@localhost ~]# _
```

Comprobamos que se ha establecido la conexión de red lanzando un ping a una URL conocida.

```
lo                Link encap:Local Loopback
                  inet addr:127.0.0.1  Mask:255.0.0.0
                  inet6 addr: ::1/128 Scope:Host
                  UP LOOPBACK RUNNING MTU:65536 Metric:1
                  RX packets:8 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:0
                  RX bytes:600 (600.0 b)  TX bytes:600 (600.0 b)

[root@localhost ~]# ifup eth0

Determinando la información IP para eth0... hecho.
[root@localhost ~]# ping google.es
PING google.es (216.58.210.131) 56(84) bytes of data.
64 bytes from mad06s09-in-f131.1e100.net (216.58.210.131): icmp_seq=1 ttl=63 time=36.9 ms
64 bytes from mad06s09-in-f131.1e100.net (216.58.210.131): icmp_seq=2 ttl=63 time=47.2 ms
64 bytes from mad06s09-in-f131.1e100.net (216.58.210.131): icmp_seq=3 ttl=63 time=45.8 ms
^C
--- google.es ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2559ms
rtt min/avg/max/mdev = 36.934/43.339/47.255/4.569 ms
[root@localhost ~]# _
```

El ping es exitoso, con lo cual comprobamos que la conexión de red se ha establecido correctamente.



Al levantar la interfaz, hemos conseguido establecer la conexión. Pero, al reiniciar, la interfaz volverá a estar inactiva, con lo cual tendremos que repetir este proceso para establecer la conexión.

Para evitar eso, debemos configurar el sistema para que levante esa interfaz durante el arranque. Para conseguirlo, debemos editar el fichero */etc/sysconfig/network-scripts/ifcfg-eth0*

Abrimos el fichero con vi

```
DEVICE=eth0
HWADDR=08:00:27:A5:E2:C1
TYPE=Ethernet
UUID=d0be6f43-88eb-40b9-a88e-6a2e884117c4
ONBOOT=no
NM_CONTROLLED=yes
BOOTPROTO=dhcp

"/etc/sysconfig/network-scripts/ifcfg-eth0" 7L, 136C
```

Vemos que el parámetro ONBOOT está con el valor “no”, así que lo cambiamos a “yes”

Pulsamos Esc + a para activar el modo de edición, y cambiamos ese valor.

```
DEVICE=eth0
HWADDR=08:00:27:A5:E2:C1
TYPE=Ethernet
UUID=d0be6f43-88eb-40b9-a88e-6a2e884117c4
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=dhcp

"/etc/sysconfig/network-scripts/ifcfg-eth0" 7L, 137C written
```

Guardamos pulsando Esc, y escribiendo :w y salimos del editor con :x

Reiniciamos para comprobar que se han aplicado los cambios y la interfaz se levanta correctamente.

```
DEVICE=eth0
HWADDR=08:00:27:A5:E2:C1
TYPE=Ethernet
UUID=d0be6f43-88eb-40b9-a88e-6a2e884117c4
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=dhcp

[root@localhost ~]# reboot_
```

Tras reiniciar, comprobamos que la conexión se establece correctamente desde el inicio.

```
CentOS release 6.9 (Final)
Kernel 2.6.32-696.el6.i686 on an i686

localhost login: root
Password:
Last login: Thu Oct 12 19:22:02 on tty1
[root@localhost ~]# ping google.es

PING google.es (216.58.210.131) 56(84) bytes of data:
64 bytes from mad06s09-in-f131.1e100.net (216.58.210.131): icmp_seq=1 ttl=63 time=37.0 ms
64 bytes from mad06s09-in-f131.1e100.net (216.58.210.131): icmp_seq=2 ttl=63 time=47.3 ms
64 bytes from mad06s09-in-f131.1e100.net (216.58.210.131): icmp_seq=3 ttl=63 time=60.5 ms
^C
--- google.es ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2726ms
rtt min/avg/max/mdev = 37.034/48.283/60.502/9.608 ms
[root@localhost ~]# _
```

## 2. Activando el reenvío de puertos al sistema anfitrión

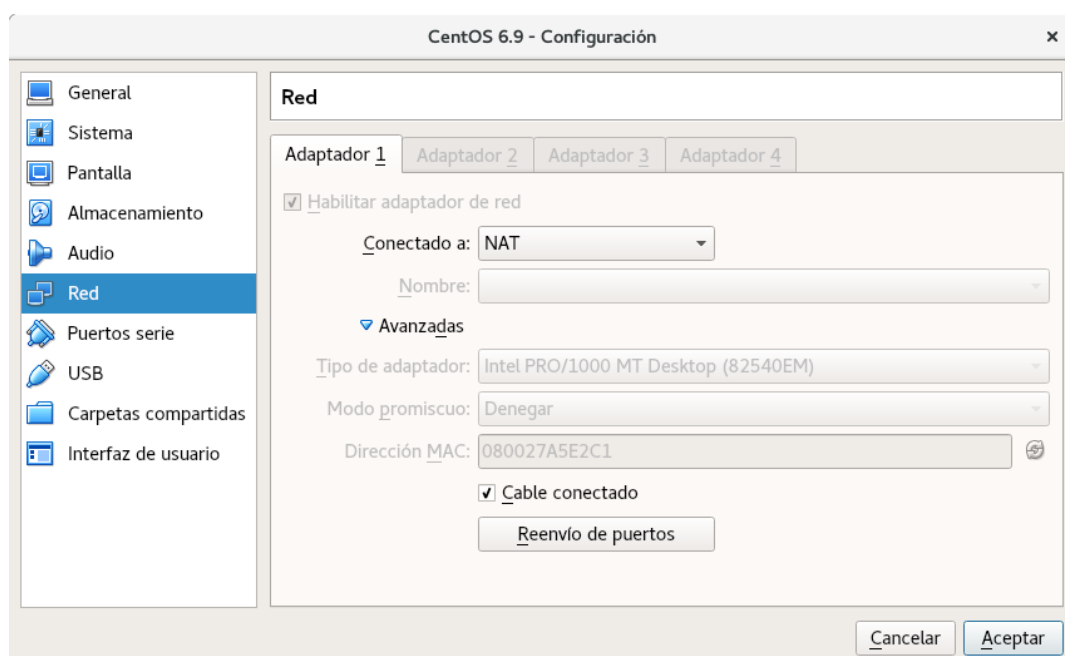
Dado que la máquina virtual es muy incómoda a la hora de introducir comandos (dado que no permite copiar y pegar), nos interesa activar un acceso remoto mediante ssh.

En un paso anterior, hemos comprobado que la máquina virtual tiene conexión tipo NAT, lo cual quiere decir que el sistema anfitrión no tiene acceso a los puertos de la máquina virtual. En esta situación, el acceso mediante ssh es imposible, dado que la máquina virtual esta aislada del sistema.

Para resolver esto, vamos a establecer un reenvío de puertos desde la máquina virtual al sistema anfitrión.

El protocolo ssh se comunica a través del puerto 22, así que el puerto que debemos redireccionar es ese. Para no chocar con el propio puerto 22 del anfitrión, este puerto va a ser redireccionado a otro puerto sin uso, en este caso el 2222.

Para redireccionar los puertos, nos vamos a la configuración de red de la máquina virtual, y pulsamos en “Avanzadas”

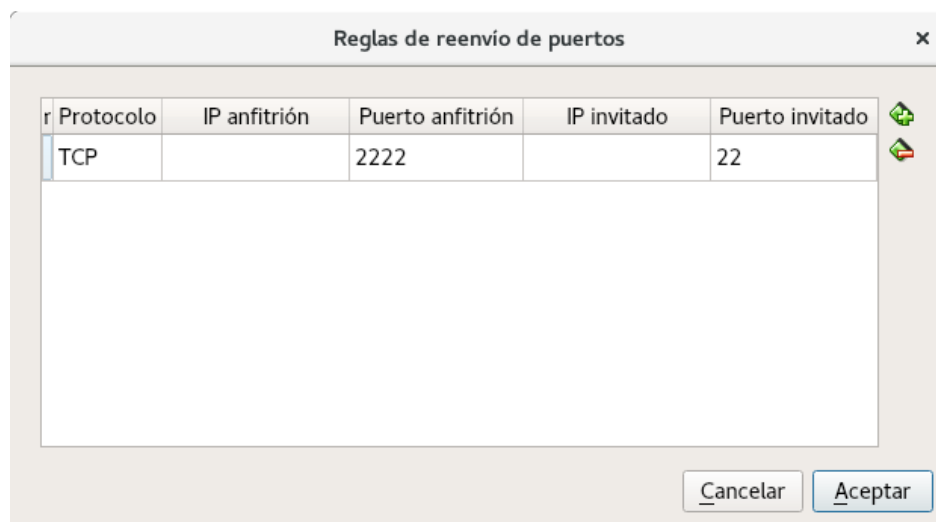


Pulsamos en “Reenvío de Puertos”, tras lo cual se nos abrirá esta ventana.

En “Puerto Anfitrión” ponemos el puerto del sistema anfitrión al que queremos redireccionar (en este caso el 2222), y en “Puerto Invitado” el puerto de la máquina virtual que queremos redireccionar (en este caso el 22).

Dado que solo vamos a redireccionar los puertos, y no queremos asignar IPs , los otros dos campos los dejamos en blanco.

Pulsamos en Aceptar para aplicar los cambios



Probamos el redireccionamiento invocando ssh desde la terminal. Para ello necesitamos el nombre de usuario, el nombre de host y el puerto.

Como no tenemos mas usuarios, vamos a acceder directamente como root, por lo que el nombre de usuario es *root*. El nombre de host es el que trae el sistema por defecto, en este caso *localhost.localdomain*.

Acerca del puerto, ssh suele usar el puerto 22, pero como hemos redireccionado al 2222, hemos de indicarlo mediante el argumento *-p*

Ejecutamos el comando desde la terminal

```
almu@debian:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[almu@debian ~]$ ssh root@localhost.localdomain -p 2222  
^C  
[almu@debian ~]$ ssh root@localhost.localdomain -p 2222  
^C  
[almu@debian ~]$
```

Vemos que no funciona, con lo cual revisamos la configuración de la máquina virtual. Intentamos instalarlo con *yum install openssh*.

```
--> Ejecutando prueba de transacción
--> Package openssh-clients.i686 0:5.3p1-122.el6 will be actualizado
--> Package openssh-clients.i686 0:5.3p1-123.el6_9 will be an update
--> Package openssh-server.i686 0:5.3p1-122.el6 will be actualizado
--> Package openssh-server.i686 0:5.3p1-123.el6_9 will be an update
--> Resolución de dependencias finalizada

Dependencias resueltas

=====
Paquete                Arquitectura      Versión           Repositorio      Tamaño
=====
Actualizando:
openssh                 i686             5.3p1-123.el6_9   updates          280 k
Actualizando para las dependencias:
openssh-clients         i686             5.3p1-123.el6_9   updates          450 k
openssh-server          i686             5.3p1-123.el6_9   updates          328 k

Resumen de la transacción
=====
Actualizar             3 Paquete(s)

Tamaño total de la descarga: 1.0 M
¿Está de acuerdo [s/N]:_
```

Yum nos indica que está instalado, y nos ofrece la posibilidad de actualizarlo, así que aceptamos.

```
Ejecutando el rpm_check_debug
Ejecutando prueba de transacción
La prueba de transacción ha sido exitosa
Ejecutando transacción
  Actualizando   : openssh-5.3p1-123.el6_9.i686                1/6
  Actualizando   : openssh-clients-5.3p1-123.el6_9.i686       2/6
  Actualizando   : openssh-server-5.3p1-123.el6_9.i686        3/6
  Limpieza       : openssh-server-5.3p1-122.el6.i686          4/6
  Limpieza       : openssh-clients-5.3p1-122.el6.i686          5/6
  Limpieza       : openssh-5.3p1-122.el6.i686                  6/6
  Verifying      : openssh-clients-5.3p1-123.el6_9.i686        1/6
  Verifying      : openssh-server-5.3p1-123.el6_9.i686        2/6
  Verifying      : openssh-5.3p1-123.el6_9.i686                3/6
  Verifying      : openssh-5.3p1-122.el6.i686                  4/6
  Verifying      : openssh-server-5.3p1-122.el6.i686          5/6
  Verifying      : openssh-clients-5.3p1-122.el6.i686          6/6

Actualizado:
openssh.i686 0:5.3p1-123.el6_9

Dependencia(s) actualizada(s):
openssh-clients.i686 0:5.3p1-123.el6_9 openssh-server.i686 0:5.3p1-123.el6_9

¡Listo!
[root@localhost ~]# _
```

Después de esto, sigue sin funcionar, así que revisamos la configuración de ssh de la maquina virtual. Dicha configuración esta en el fichero `/etc/ssh/ssh_config`. Abrimos el fichero con `vi` y lo revisamos.

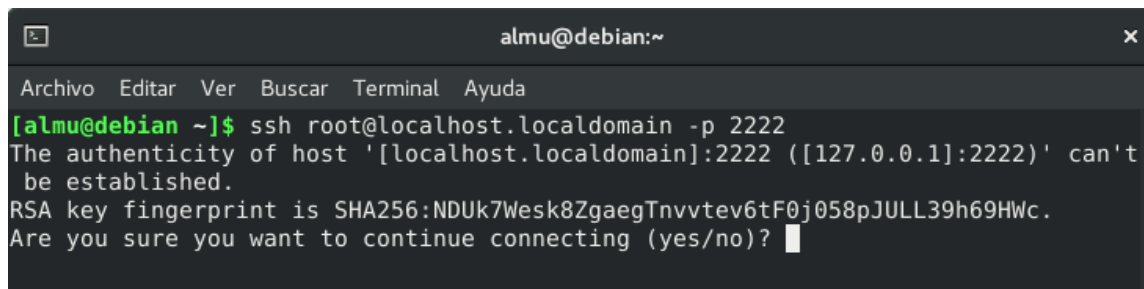
Vemos que la opción `Port 22` está comentada, así que lo descomentamos y guardamos los cambios

```
# GSSAPITrustDNS no
# BatchMode no
# CheckHostIP yes
# AddressFamily any
# ConnectTimeout 0
# StrictHostKeyChecking ask
# IdentityFile ~/.ssh/identity
# IdentityFile ~/.ssh/id_rsa
# IdentityFile ~/.ssh/id_dsa
# Port 22
# Protocol 2,1
# Cipher 3des
# Ciphers aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,3des-cbc
# MACs hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160
# EscapeChar ~
# Tunnel no
# TunnelDevice any:any
# PermitLocalCommand no
# VisualHostKey no
Host *
    GSSAPIAuthentication yes
# If this option is set to yes then remote X11 clients will have full access
# to the original X11 display. As virtually no X11 client supports the untrusted
-- INSERT --
```

Reiniciamos el servicio ssh y probamos de nuevo

```
# AddressFamily any
# ConnectTimeout 0
# StrictHostKeyChecking ask
# IdentityFile ~/.ssh/identity
# IdentityFile ~/.ssh/id_rsa
# IdentityFile ~/.ssh/id_dsa
# Port 22
# Protocol 2,1
# Cipher 3des
# Ciphers aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,3des-cbc
# MACs hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160
# EscapeChar ~
[root@localhost ~]# /etc/init.d/
auditd          iscsid          netfs           saslauthd
blk-availability killall         network        single
crond           lvm2-lvmetad   postfix        sshd
halt            lvm2-monitor   rdisc          udev-post
ip6tables      mdmonitor      restorecond
iptables       multipathd     rsyslog
iscsi          netconsole     sandbox
[root@localhost ~]# /etc/init.d/ssh restart
Parando sshd: [ OK ]
Iniciando sshd: [ OK ]
[root@localhost ~]# _
```

Vemos que esta vez la conexión ha sido exitosa. Escribimos “yes” para iniciar la conexión.

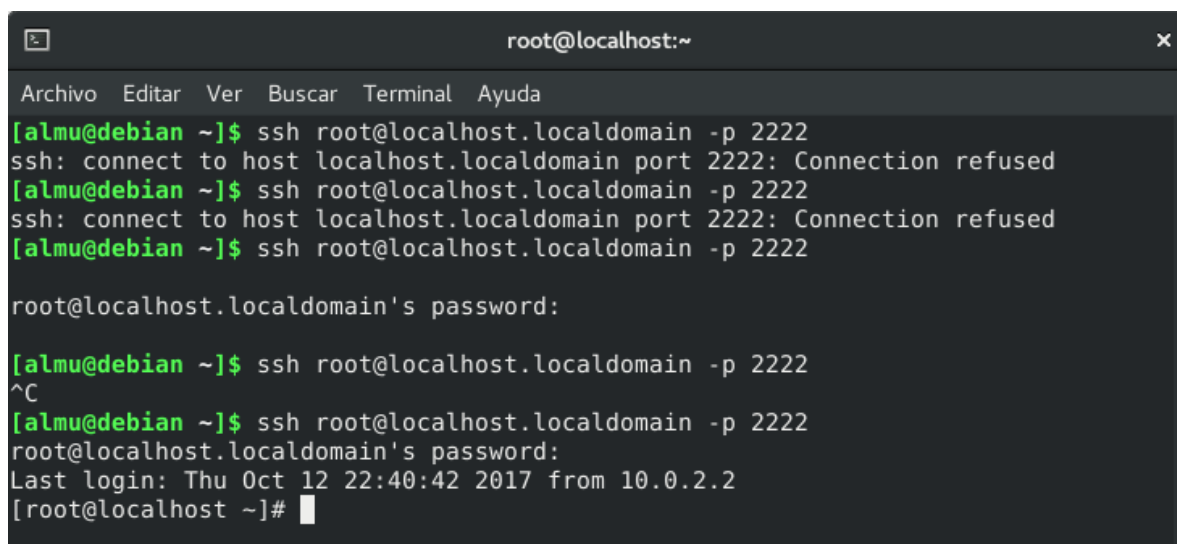
A terminal window titled 'almu@debian:~' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The prompt is '[almu@debian ~]\$'. The command 'ssh root@localhost.localdomain -p 2222' has been entered. The output shows a warning about host authenticity: 'The authenticity of host '[localhost.localdomain]:2222 ([127.0.0.1]:2222)' can't be established. RSA key fingerprint is SHA256:NDUk7Wesk8ZgaegTnvvtev6tF0j058pJULL39h69HWc. Are you sure you want to continue connecting (yes/no)?'. A cursor is visible at the end of the line.

```
almu@debian:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[almu@debian ~]$ ssh root@localhost.localdomain -p 2222  
The authenticity of host '[localhost.localdomain]:2222 ([127.0.0.1]:2222)' can't  
be established.  
RSA key fingerprint is SHA256:NDUk7Wesk8ZgaegTnvvtev6tF0j058pJULL39h69HWc.  
Are you sure you want to continue connecting (yes/no)?
```

Nos pregunta si queremos añadir la huella digital del servidor, a lo que respondemos que sí. Tras la respuesta, se cierra la conexión.

Finalmente, comprobamos que es un conflicto con la red WiFi del móvil al que estamos conectados.

Nos desconectamos de la red y, tras varios intentos, logramos establecer la conexión ssh.

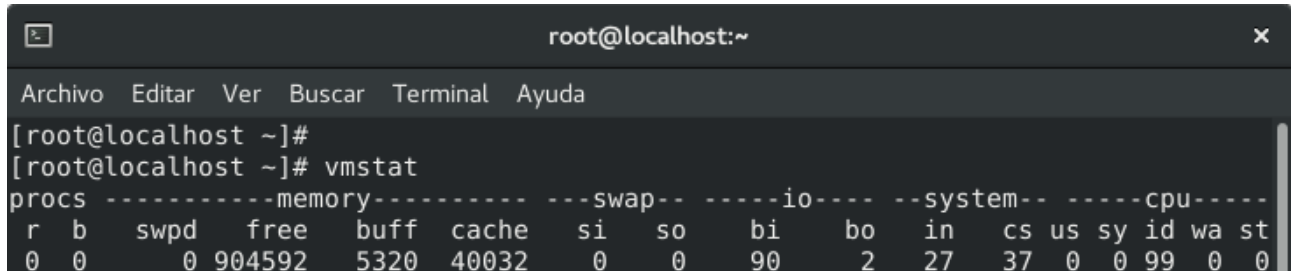
A terminal window titled 'root@localhost:~' with a menu bar (Archivo, Editar, Ver, Buscar, Terminal, Ayuda). The prompt is '[almu@debian ~]\$'. Three 'ssh root@localhost.localdomain -p 2222' commands are entered, each resulting in 'ssh: connect to host localhost.localdomain port 2222: Connection refused'. Then, the prompt 'root@localhost.localdomain's password:' is shown. After pressing Ctrl-C (^C), the prompt is shown again. Finally, the 'ssh root@localhost.localdomain -p 2222' command is entered, followed by the password prompt. The output shows 'Last login: Thu Oct 12 22:40:42 2017 from 10.0.2.2' and the root prompt '[root@localhost ~]#'.

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[almu@debian ~]$ ssh root@localhost.localdomain -p 2222  
ssh: connect to host localhost.localdomain port 2222: Connection refused  
[almu@debian ~]$ ssh root@localhost.localdomain -p 2222  
ssh: connect to host localhost.localdomain port 2222: Connection refused  
[almu@debian ~]$ ssh root@localhost.localdomain -p 2222  
root@localhost.localdomain's password:  
^C  
[almu@debian ~]$ ssh root@localhost.localdomain -p 2222  
root@localhost.localdomain's password:  
Last login: Thu Oct 12 22:40:42 2017 from 10.0.2.2  
[root@localhost ~]#
```

### 3. Probando vmstat

Probamos la herramienta vmstat, que permite recoger información sobre el estado del sistema; indicando información sobre CPU, memoria y disco, entre otros.

Ejecutamos el comando sin ningún argumento:



```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[root@localhost ~]#  
[root@localhost ~]# vmstat  
procs -----memory----- --swap-- -----io----- --system-- -----cpu-----  
r  b   swpd   free   buff   cache   si   so    bi    bo    in    cs  us  sy  id  wa  st  
0  0     0 904592   5320  40032    0    0    90    2   27   37  0  0 99  0  0
```

En esta salida vemos ya distintos datos sobre el estado del sistema:

Sobre los procesos (bloque procs), vemos:

**r**: numero de procesos preparados

**b**: numero de procesos bloqueados

En este caso vemos que no hay procesos preparados ni bloqueados

En la sección de memoria (bloque memory) tenemos:

**swpd**: memoria guardada en disco (swap)

**free**: espacio libre en memoria

**buff**: memoria destinada a buffer

**cache**: memoria destinada a cache

En este caso, vemos que no se está usando la swap, que la memoria libre es 904592 KB (prácticamente toda la memoria), hay 5320 KB usados en buffer y 40032 KB en cache

En la sección de swap vemos dos columnas:

**si**: trafico disco → memoria / segundo. Trafico de entrada a la memoria procedente del disco, expresado en KB/s

**so**: tráfico memoria → disco *segundo*. Trafico de salida de memoria con destino a disco, expresado en KB/s.



En este caso vemos que no hay trafico entre la memoria y la swap del disco.

La sección de io muestra el numero de bloques/segundo leídos o escritos en los dispositivos de bloques

**bi:** trafico de entrada, bloques leídos

**bo:** trafico de salida, bloques escritos

En este caso, vemos que el trafico de lectura es de 90 bloques/s, y el de escritura de 2 bloques/s.

En la sección de system vemos dos columnas:

**in:** número de interrupciones por segundo

**cs:** cambios de contexto por segundo

En este caso vemos que se producen 27 interrupciones/s y 32 cambios de contexto/s.

En la sección de CPU tenemos los porcentajes de uso del procesador:

**us:** user, tiempo dedicado a tareas de usuario

**sy:** system, tiempo dedicado a tareas del sistema

**id:** idle, tiempo de inactividad

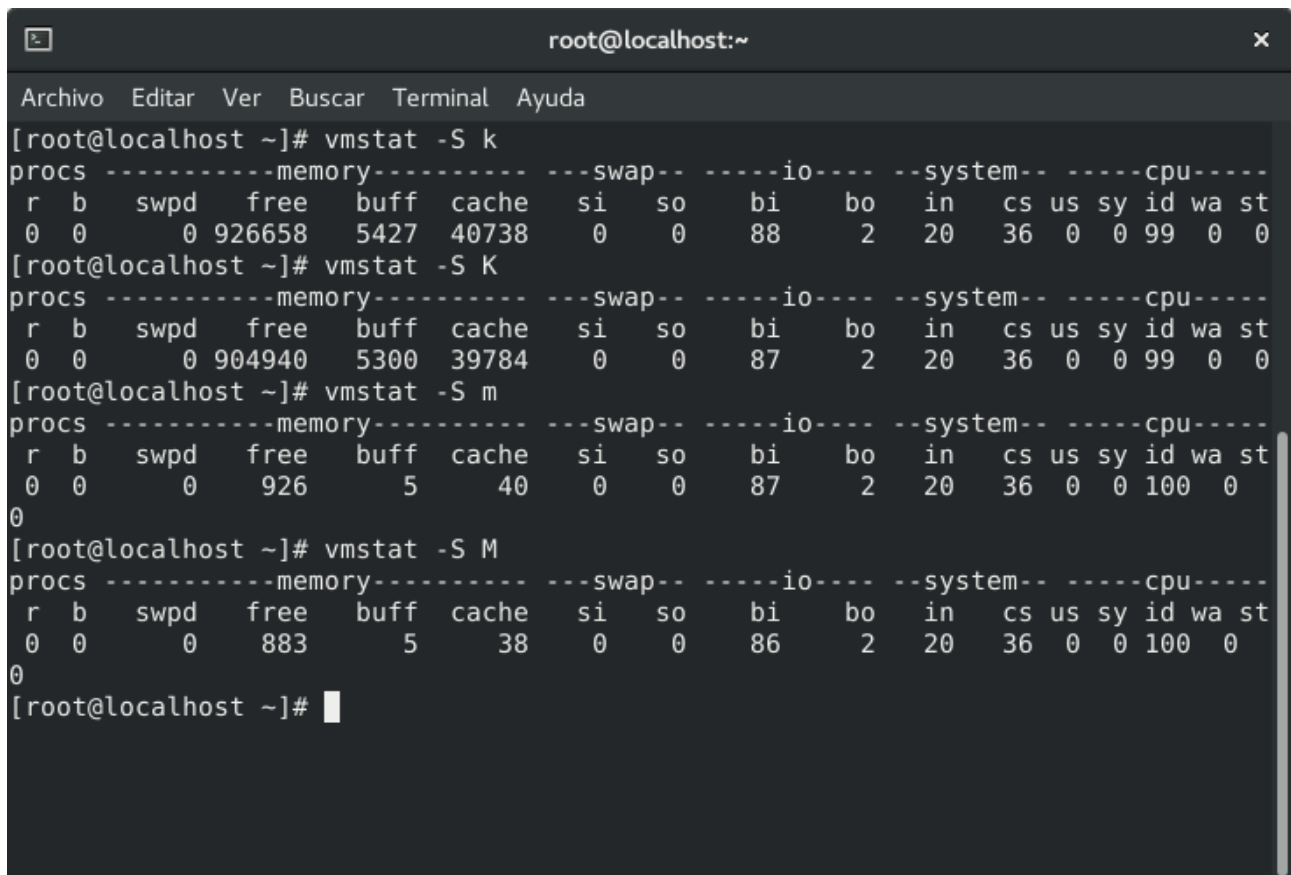
**wa:** wait, tiempo de bloqueo

**st:** service time, tiempo de ocupación de la CPU con procesos distintos a la máquina virtual

En este caso vemos que prácticamente no se ha dedicado tiempo a tareas tanto de usuario como de sistema (0%) y que el procesador ha estado prácticamente todo el tiempo inactivo (99%). También vemos que los tiempos de bloqueo y servicio son 0%.

Esto es debido a que la máquina virtual esta recién iniciada y el usuario aún no ha realizado prácticamente ninguna acción sobre el sistema.

En la siguiente prueba, vamos a probar con diferentes unidades de medida



```
root@localhost:~
Archivo Editar Ver Buscar Terminal Ayuda
[root@localhost ~]# vmstat -S k
procs -----memory----- ---swap-- -----io----- --system-- -----cpu-----
r b  swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id  wa  st
0 0    0 926658  5427 40738   0   0  88   2  20  36  0  0 99  0  0
[root@localhost ~]# vmstat -S K
procs -----memory----- ---swap-- -----io----- --system-- -----cpu-----
r b  swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id  wa  st
0 0    0 904940  5300 39784   0   0  87   2  20  36  0  0 99  0  0
[root@localhost ~]# vmstat -S m
procs -----memory----- ---swap-- -----io----- --system-- -----cpu-----
r b  swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id  wa  st
0 0    0    926    5    40   0   0  87   2  20  36  0  0 100  0
0
[root@localhost ~]# vmstat -S M
procs -----memory----- ---swap-- -----io----- --system-- -----cpu-----
r b  swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id  wa  st
0 0    0    883    5    38   0   0  86   2  20  36  0  0 100  0
0
[root@localhost ~]#
```

El argumento -S nos permite especificar la unidad de medida en que queremos mostrar los resultados

Las opciones son:

- k** = KB, siendo K = 1000
- K** = KiB, siendo Ki = 1024
- m** = MB, siendo M = 1.000.000
- M** = MiB, siendo Mi = 1024<sup>2</sup>

Como vemos en la imagen, al expresar los valores en KiB y MiB, los valores se reducen ligeramente respecto a sus equivalentes en KB y MB, debido a que la unidad de medida es superior.

También vemos que, al expresar los valores en MB y MiB, estos reducen su precisión respecto a sus valores en KB y KiB, al no incluir las cifras decimales de los mismos.

Vemos que podemos crear una monitorización periódica indicando un numero después del comando. Este número se corresponderá a la cantidad de segundos de espera entre una llamada y otra del comando.

```

root@localhost:~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
[root@localhost ~]# vmstat 5
procs  -----memory-----  ---swap--  -----io-----  --system--  -----cpu-----
r  b    swpd    free    buff    cache    si    so    bi    bo    in    cs  us  sy  id  wa  st
1  0      0 904348    5460   39932     0     0    31     1    10    16  0  0 100  0
0
1  0      0 904340    5460   39932     0     0     0     0     9     7  0  0 100  0
0
0  0      0 904340    5468   39932     0     0     0     2     7     8  0  0 100  0
0
0  0      0 904340    5468   39932     0     0     0     0     5     6  0  0 100  0
0
0  0      0 904340    5468   39932     0     0     0     0     6     6  0  0 100  0
0
0  0      0 904340    5468   39932     0     0     0     0     6     7  0  0 100  0
0
0  0      0 904340    5468   39932     0     0     0     0     6     6  0  0 100  0
0
0  0      0 904340    5468   39932     0     0     0     0     6     6  0  0 100  0
0
0  0      0 904340    5468   39932     0     0     0     0     6     6  0  0 100  0
0
0  0      0 904340    5468   39932     0     0     0     1     6     7  0  0 100  0
0
0  0      0 904340    5468   39932     0     0     0     0     6     6  0  0 100  0

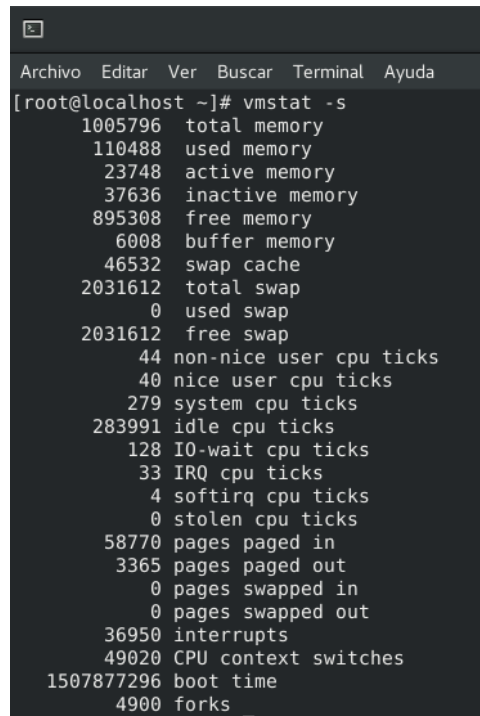
```

En este caso, hemos puesto un tiempo de espera de 5 segundos. Vemos que los consumos de CPU y memoria prácticamente se mantienen constantes (salvo una pequeña variación en el buffer en la tercera llamada).

En la primera llamada vemos un porcentaje de llamadas a disco algo superior a las siguientes, probablemente debido a la búsqueda del programa previa a la ejecución. Esto se replica en las interrupciones y cambios de contexto del sistema.

En las siguientes llamadas se aprecian ligeros cambios en el numero de interrupciones y cambios de contexto, pero el resto de valores se mantienen constantes.

Podemos mostrar las estadísticas del sistema con `vmstat -s`



```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
[root@localhost ~]# vmstat -s
1005796 total memory
110488 used memory
23748 active memory
37636 inactive memory
895308 free memory
6008 buffer memory
46532 swap cache
2031612 total swap
0 used swap
2031612 free swap
44 non-nice user cpu ticks
40 nice user cpu ticks
279 system cpu ticks
283991 idle cpu ticks
128 IO-wait cpu ticks
33 IRQ cpu ticks
4 softirq cpu ticks
0 stolen cpu ticks
58770 pages paged in
3365 pages paged out
0 pages swapped in
0 pages swapped out
36950 interrupts
49020 CPU context switches
1507877296 boot time
4900 forks
```

Aquí vemos de forma algo mas detallada la información dada anteriormente.

Como añadido a la información anterior, vemos más información sobre los ticks de CPU, las interrupciones y la paginación.

En tareas del usuario, la CPU lleva gastados 44 ticks “non-nice” y 40 ticks “nice”. También vemos que la CPU ha gastado 279 ticks en tareas del sistema, 283991 ticks en estado inactivo, y 128 ticks en espera a entrada/salida.

Se han producido 49020 cambios de contexto y no se ha producido ningún robo de ciclo.

En cuanto a interrupciones, la CPU ha gastado 33 ticks en interrupciones hardware y 4 en interrupciones software. En total, se han producido 36950 interrupciones.

Sobre los procesos, vemos que hay 58770 lecturas a paginas y 3365 escrituras a paginas, y ninguna se ha movido a la swap. Se han producido 4900 forks de procesos.

#### 4. Aumentando la RAM

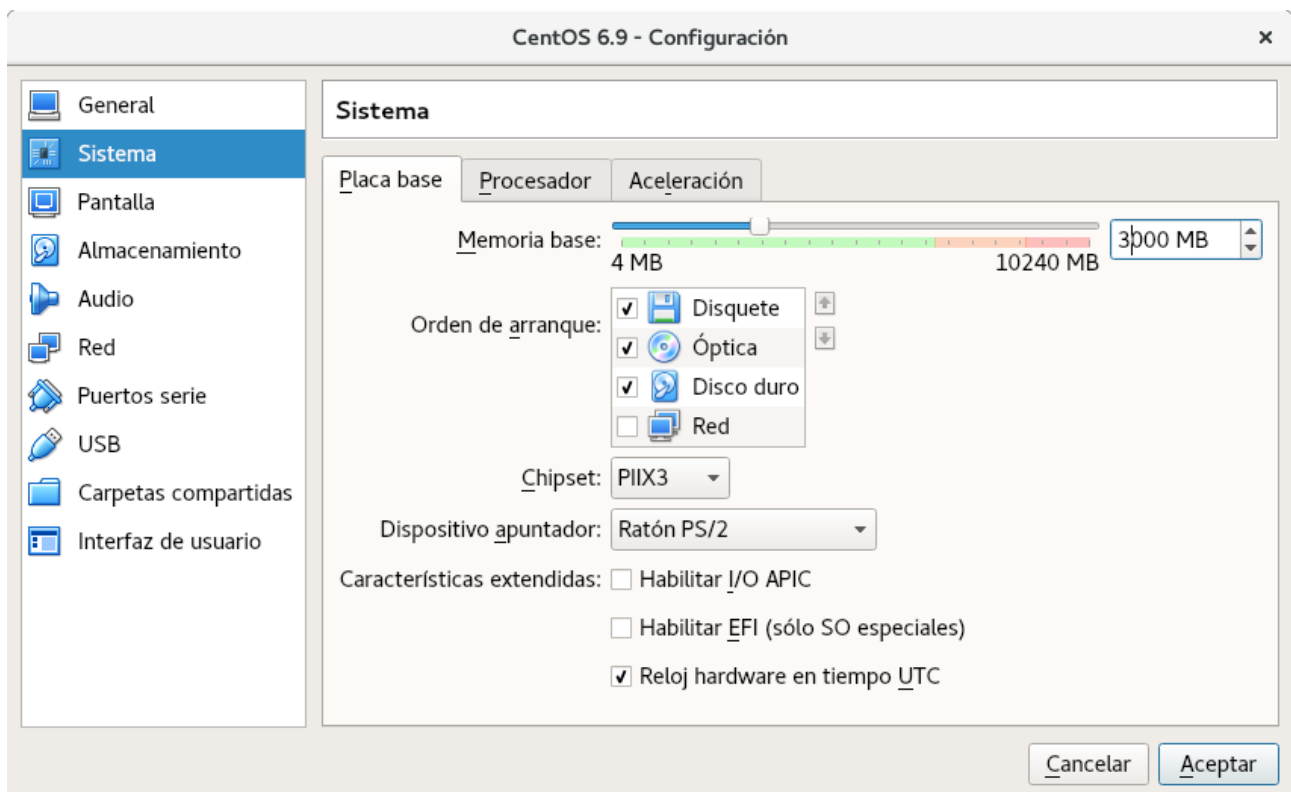
Vamos a aumentar la RAM de la máquina virtual, y comprobar que el cambio se vea reflejado en el sistema. Para ello, vamos a usar los comandos *free* y *vmstat*

Comprobamos el estado inicial de la memoria, usando MiB como medida:

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[root@localhost ~]# free -h  
Mem:      total      used      free      shared    buffers     cached  
-/+ buffers/cache:      56M      925M  
Swap:      1.9G       0B       1.9G  
[root@localhost ~]# vmstat -S M  
procs -----memory----- ---swap-- -----io----- --system-- -----cpu-----  
-  
r  b  swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id  wa  s  
t  
0  0      0   872    6   47   0   0  12   1  10  12  0  0 100  0  
0
```

Vemos que la memoria total es 982 MiB, lo cual se corresponde a los 1000 MB reservados para la máquina; estando libre 872 MiB, con 47 MiB de cache, y 6.2 MiB en buffer. Los datos coinciden en ambos comandos, mostrando mayor precisión en el comando *free*.

Apagamos la máquina virtual y ampliamos la memoria a 3000 MB.



Volvemos a comprobar el estado de la memoria

```

root@localhost:~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
[root@localhost ~]# free -h
              total        used        free      shared    buffers     cached
Mem:           2.8G         102M         2.7G          200K          5.1M          38M
-/+ buffers/cache:          58M         2.7G
Swap:          1.9G           0B          1.9G
[root@localhost ~]# vmstat -S M
procs -----memory----- --swap--  -----io----- --system--  -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo    in   cs us sy id wa s
t
0  0     0   2734     5    38    0    0  2108   15  362  740  2  7 89  3
0

```

Vemos que, efectivamente, el sistema refleja el aumento de memoria realizado en la máquina, y que ambos comandos siguen mostrando la misma información.

## 5. Analizando el efecto de la entrada/salida de un programa en ejecución

En este paso tenemos que monitorizar el efecto de la ejecución de un programa dentro del sistema, usando para ello *vmstat* e *iostat*.

El programa a ejecutar es este:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char str[] = "This is tutorialspoint.com";
    FILE *fd = fopen("prueba.txt", "w");
    int i, j;

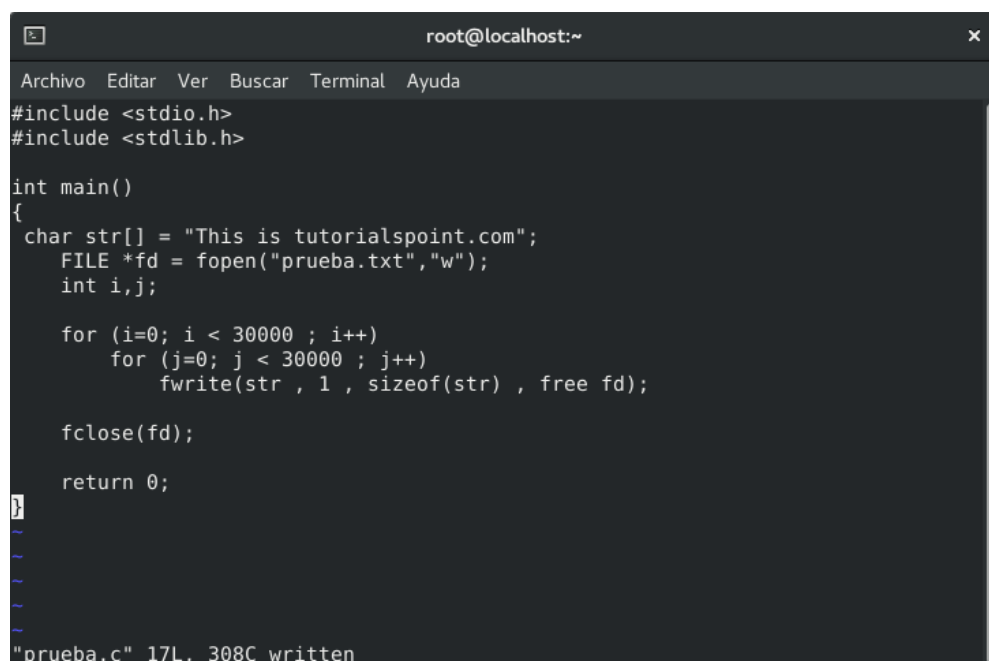
    for (i=0; i < 30000 ; i++)
        for (j=0; j < 30000 ; j++)
            fwrite(str , 1 , sizeof(str) , free fd);

    fclose(fd);

    return 0;
}
```

Este programa irá realizando escribiendo una cadena de caracteres en un fichero alojado en disco. El programa realizará un total de  $30000^2$  escrituras en disco. Para analizar el efecto que producen esas llamadas, usaremos *vmstat* e *iostat*.

Copiamos el código en *vi*, con el nombre *prueba.c*, y lo guardamos.



```
root@localhost:~
Archivo Editar Ver Buscar Terminal Ayuda
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char str[] = "This is tutorialspoint.com";
    FILE *fd = fopen("prueba.txt", "w");
    int i, j;

    for (i=0; i < 30000 ; i++)
        for (j=0; j < 30000 ; j++)
            fwrite(str , 1 , sizeof(str) , free fd);

    fclose(fd);

    return 0;
}

"prueba.c" 17L, 308C written
```

GCC no está instalado, así que lo instalamos usando yum.

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[root@localhost ~]# yum install gcc  
Complementos cargados:fastestmirror  
Configurando el proceso de instalación  
Loading mirror speeds from cached hostfile  
+ base: mirror.airenetworks.es  
+ extras: mirror.airenetworks.es  
+ updates: mirror.airenetworks.es  
base | 3.7 kB | 00:00  
extras | 3.3 kB | 00:00  
updates | 3.4 kB | 00:00  
Resolviendo dependencias  
--> Ejecutando prueba de transacción  
--> Package gcc.i686 0:4.4.7-18.el6 will be instalado  
--> Procesando dependencias: libgomp = 4.4.7-18.el6 para el paquete: gcc-4.4.7-18.el6.i686  
--> Procesando dependencias: cpp = 4.4.7-18.el6 para el paquete: gcc-4.4.7-18.el6.i686  
--> Procesando dependencias: glibc-devel >= 2.2.90-12 para el paquete: gcc-4.4.7-18.el6.i686  
--> Procesando dependencias: cloog-ppl >= 0.15 para el paquete: gcc-4.4.7-18.el6.i686  
--> Procesando dependencias: libgomp.so.1 para el paquete: gcc-4.4.7-18.el6.i686  
--> Ejecutando prueba de transacción  
--> Package cloog-ppl.i686 0:0.15.7-1.2.el6 will be instalado  
--> Procesando dependencias: libppl.c.so.2 para el paquete: cloog-ppl-0.15.7-1.2.el6.i686  
--> Procesando dependencias: libppl.so.7 para el paquete: cloog-ppl-0.15.7-1.2.el6.i686  
--> Package cpp.i686 0:4.4.7-18.el6 will be instalado  
--> Procesando dependencias: libmpfr.so.1 para el paquete: cpp-4.4.7-18.el6.i686  
--> Package glibc-devel.i686 0:2.12-1.209.el6_9.2 will be instalado  
--> Procesando dependencias: glibc-headers = 2.12-1.209.el6_9.2 para el paquete: glibc-devel-2.12-1.209.el6_9.2.i686  
--> Procesando dependencias: glibc-headers para el paquete: glibc-devel-2.12-1.209.el6_9.2.i686  
--> Package libgomp.i686 0:4.4.7-18.el6 will be instalado  
--> Ejecutando prueba de transacción  
--> Package glibc.i686 0:2.12-1.209.el6 will be actualizado  
--> Procesando dependencias: glibc = 2.12-1.209.el6 para el paquete: glibc-common-2.12-1.209.el6.i686  
--> Package glibc-common.i686 0:2.12-1.209.el6_9.2 will be an update  
--> Package glibc-headers.i686 0:2.12-1.209.el6_9.2 will be instalado  
--> Procesando dependencias: kernel-headers >= 2.2.1 para el paquete: glibc-headers-2.12-1.209.el6_9.2.i686  
--> Procesando dependencias: kernel-headers para el paquete: glibc-headers-2.12-1.209.el6_9.2.i686  
--> Package mpfr.i686 0:2.4.1-6.el6 will be instalado
```

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
--> Procesando dependencias: kernel-headers para el paquete: glibc-headers-2.12-1.209.el6_9.2.i686  
--> Package mpfr.i686 0:2.4.1-6.el6 will be instalado  
--> Package ppl.i686 0:0.10.2-11.el6 will be instalado  
--> Ejecutando prueba de transacción  
--> Package glibc-common.i686 0:2.12-1.209.el6 will be actualizado  
--> Package glibc-common.i686 0:2.12-1.209.el6_9.2 will be an update  
--> Package kernel-headers.i686 0:2.6.32-696.13.2.el6 will be instalado  
--> Resolución de dependencias finalizada  
Dependencias resueltas  
=====
```

Paquete	Arquitectura	Versión	Repositorio	Tamaño
Instalando:				
gcc	i686	4.4.7-18.el6	base	8.2 M
Instalando para las dependencias:				
cloog-ppl	i686	0.15.7-1.2.el6	base	93 k
cpp	i686	4.4.7-18.el6	base	3.4 M
glibc-devel	i686	2.12-1.209.el6_9.2	updates	991 k
glibc-headers	i686	2.12-1.209.el6_9.2	updates	628 k
kernel-headers	i686	2.6.32-696.13.2.el6	updates	4.5 M
libgomp	i686	4.4.7-18.el6	base	136 k
mpfr	i686	2.4.1-6.el6	base	153 k
ppl	i686	0.10.2-11.el6	base	1.3 M
Actualizando para las dependencias:				
glibc	i686	2.12-1.209.el6_9.2	updates	4.4 M
glibc-common	i686	2.12-1.209.el6_9.2	updates	14 M

```
Resumen de la transacción  
=====
```

Instalar	9 Paquete(s)
Actualizar	2 Paquete(s)

```
Tamaño total de la descarga: 38 M  
Está de acuerdo [s/N]:
```

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
Actualizando : glibc-common-2.12-1.209.el6_9.2.i686 3/13  
Instalando : glibc-headers-2.12-1.209.el6_9.2.i686 4/13  
Instalando : glibc-devel-2.12-1.209.el6_9.2.i686 5/13  
Instalando : libgomp-4.4.7-18.el6.i686 6/13  
Instalando : mpfr-2.4.1-6.el6.i686 7/13  
Instalando : cpp-4.4.7-18.el6.i686 8/13  
Instalando : ppl-0.10.2-11.el6.i686 9/13  
Instalando : cloog-ppl-0.15.7-1.2.el6.i686 10/13  
Instalando : gcc-4.4.7-18.el6.i686 11/13  
Limpieza : glibc-2.12-1.209.el6.i686 12/13  
Limpieza : glibc-common-2.12-1.209.el6.i686 13/13  
Verifying : libgomp-4.4.7-18.el6.i686 1/13  
Verifying : gcc-4.4.7-18.el6.i686 2/13  
Verifying : glibc-common-2.12-1.209.el6_9.2.i686 3/13  
Verifying : glibc-2.12-1.209.el6_9.2.i686 4/13  
Verifying : kernel-headers-2.6.32-696.13.2.el6.i686 5/13  
Verifying : glibc-headers-2.12-1.209.el6_9.2.i686 6/13  
Verifying : mpfr-2.4.1-6.el6.i686 7/13  
Verifying : ppl-0.10.2-11.el6.i686 8/13  
Verifying : cpp-4.4.7-18.el6.i686 9/13  
Verifying : cloog-ppl-0.15.7-1.2.el6.i686 10/13  
Verifying : glibc-devel-2.12-1.209.el6_9.2.i686 11/13  
Verifying : glibc-common-2.12-1.209.el6.i686 12/13  
Verifying : glibc-2.12-1.209.el6.i686 13/13  
Instalado:  
gcc.i686 0:4.4.7-18.el6  
Dependencia(s) instalada(s):  
cloog-ppl.i686 0:0.15.7-1.2.el6 cpp.i686 0:4.4.7-18.el6 glibc-devel.i686 0:2.12-1.209.el6_9.2 glibc-headers.i686 0:2.12-1.209.el6_9.2  
kernel-headers.i686 0:2.6.32-696.13.2.el6 libgomp.i686 0:4.4.7-18.el6 mpfr.i686 0:2.4.1-6.el6 ppl.i686 0:0.10.2-11.el6  
Dependencia(s) actualizada(s):  
glibc.i686 0:2.12-1.209.el6_9.2 glibc-common.i686 0:2.12-1.209.el6_9.2  
¡Listo!  
[root@localhost ~]#
```



```

root@localhost:~
Archivo Editar Ver Buscar Terminal Ayuda
[root@localhost ~]# gcc prueba.c -o prueba
prueba.c: En la función 'main':
prueba.c:12: error: expected ')' before 'fd'
prueba.c:12: aviso: se pasa el argumento 4 de 'fwrite' desde un tipo de puntero incompatible
/usr/include/stdio.h:710: nota: se esperaba 'struct FILE * __restrict__' pero el argumento es de tipo 'void (*)(void *)'

```

```

root@localhost:~
Archivo  Editor  Ver  Buscar  Terminal  Ayuda

#include <stdio.h>
#include <stdlib.h>

int main()
{
    char str[] = "This is tutorialspoint.com";
    FILE *fd = fopen("prueba.txt","w");
    int i,j;

    for (i=0; i < 30000 ; i++)
        for (j=0; j < 30000 ; j++)
            fwrite(str , 1 , sizeof(str) , free fd);

    fclose(fd);

    return 0;
}

"prueba.c" 17L, 308C

```

33

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
#include <stdio.h>  
#include <stdlib.h>  
  
int main()  
{  
    char str[] = "This is tutorialspoint.com";  
    FILE *fd = fopen("prueba.txt","w");  
    int i,j;  
  
    for (i=0; i < 30000 ; i++)  
        for (j=0; j < 30000 ; j++)  
            fwrite(str , 1 , sizeof(str) , fd);  
  
    fclose(fd);  
  
    return 0;  
}
```

Tras este cambio, comprobamos que el programa ya compila correctamente

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[root@localhost ~]# vi prueba.c  
[root@localhost ~]# gcc prueba.c -o prueba  
[root@localhost ~]#
```

Vemos que *iostat* no está instalado, así que lo buscamos en yum y lo instalamos. Para buscar el paquete usamos el comando *yum search iostat*, que nos muestra que *iostat* pertenece al paquete *sysstat*, así que lo instalamos con *yum install sysstat*

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[root@localhost ~]# vi prueba.c  
[root@localhost ~]# gcc prueba.c -o prueba  
[root@localhost ~]# iostat  
-bash: iostat: no se encontró la orden  
[root@localhost ~]# yum search iostat  
Complementos cargados:fastestmirror  
Loading mirror speeds from cached hostfile  
* base: mirror.airenetworks.es  
* extras: mirror.airenetworks.es  
* updates: mirror.airenetworks.es  
base | 3.7 kB | 00:00  
extras | 3.3 kB | 00:00  
updates | 3.4 kB | 00:00  
===== N/S Matched: iostat =====  
pcp-import-iostat2pcp.i686 : Performance Co-Pilot tools for importing iostat  
: data into PCP archive logs  
sysstat.i686 : The sar and iostat system monitoring commands  
Name and summary matches only, use "search all" for everything.  
[root@localhost ~]# yum install sysstat
```

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
Loading mirror speeds from cached hostfile  
* base: mirror.airenetworks.es  
* extras: mirror.airenetworks.es  
* updates: mirror.airenetworks.es  
Resolviendo dependencias  
--> Ejecutando prueba de transacción  
--> Package sysstat.i686 0:9.0.4-33.el6 will be instalado  
--> Resolución de dependencias finalizada  
  
Dependencias resueltas  
  
=====
```

Paquete	Arquitectura	Versión	Repositorio	Tamaño
Instalando: sysstat	i686	9.0.4-33.el6	base	228 k

```
=====
```

Resumen de la transacción

Instalar 1 Paquete(s)

Tamaño total de la descarga: 228 k  
Tamaño instalado: 804 k  
Está de acuerdo [s/N]:

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
Instalando:  
sysstat i686 9.0.4-33.el6 base 228 k  
  
Resumen de la transacción  
=====
```

Instalar 1 Paquete(s)

Tamaño total de la descarga: 228 k  
Tamaño instalado: 804 k  
Está de acuerdo [s/N]:s

Descargando paquetes:  
sysstat-9.0.4-33.el6.i686.rpm | 228 kB 00:00  
Ejecutando el rpm\_check\_debug  
Ejecutando prueba de transacción  
La prueba de transacción ha sido exitosa  
Ejecutando transacción

Instalando	: sysstat-9.0.4-33.el6.i686	1/1
Verifying	: sysstat-9.0.4-33.el6.i686	1/1

```
Instalado:  
sysstat.i686 0:9.0.4-33.el6  
  
¡Listo!  
[root@localhost ~]#
```

Una vez instalado, probamos a ejecutar *iostat* para comprobar que funciona bien

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[root@localhost ~]# iostat  
Linux 2.6.32-696.el6.i686 (localhost.localdomain) 13/10/17 _i686_ (1 CPU)  
  
avg-cpu:  %user   %nice %system %iowait  %steal   %idle  
           0,03    0,00    0,10    0,05    0,00   99,81  
  
Device:            tps    Blk_read/s    Blk_wrtn/s    Blk_read    Blk_wrtn  
sda                 1,84        100,98         7,68      279374      21258  
dm-0                 2,52         97,61         7,68      270066      21240  
dm-1                 0,11         0,87         0,00        2400         0
```

Iniciamos la ejecución del programa, usando *vmstat* e *iostat* en segundo plano para recopilar datos de monitorización.

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[root@localhost ~]# vmstat 5 >> vmstat &  
[1] 1358  
[root@localhost ~]# iostat 5 >> iostat &  
[2] 1359  
[root@localhost ~]# ./prueba && killall vmstat && killall iostat  
[1]- Terminado          vmstat 5 >> vmstat  
[2]+ Terminado          iostat 5 >> iostat
```

Una vez terminada la ejecución, comprobamos los resultados

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[root@localhost ~]# vmstat 5 >> vmstat &  
[1] 1358  
[root@localhost ~]# iostat 5 >> iostat &  
[2] 1359  
[root@localhost ~]# ./prueba && killall vmstat && killall iostat  
[1]- Terminado          vmstat 5 >> vmstat  
[2]+ Terminado          iostat 5 >> iostat  
[root@localhost ~]# more vmstat  
procs -----memory----- --swap-- -----io----- --system-- -----cpu-----  
 r b swpd free buff cache si so bi bo in cs us sy id wa st  
 0 0 0 1350772 14160 1456256 0 0 0 19 520 11 12 0 0 99 0 0  
 0 0 0 1350764 14172 1456264 0 0 0 0 26 22 24 0 0 100 0 0  
 0 1 0 2379704 14188 440240 0 0 0 50814 239 90 2 6 60 32 0  
 0 2 0 2101448 14204 712368 0 0 0 53190 245 115 1 5 0 94 0  
 2 1 0 1775080 14224 1036000 0 0 0 63169 375 148 3 7 0 89 0  
 0 1 0 1460616 14244 1347304 0 0 0 61991 320 159 2 6 0 92 0  
 0 1 0 1102504 14268 1699560 0 0 2 74843 351 159 3 7 0 90 0  
 0 1 0 797960 14284 1998568 0 0 0 59709 315 141 2 6 0 92 0  
 1 0 0 565832 14304 2228452 0 0 0 55814 378 67 10 12 0 78 0  
 1 0 0 565980 14304 2228452 0 0 0 0 118 5 95 5 0 0 0  
 1 0 0 566004 14312 2228452 0 0 0 4 136 8 97 3 0 0 0  
 1 0 0 566028 14320 2228456 0 0 0 4 140 7 93 7 0 0 0  
 1 0 0 566052 14320 2228456 0 0 0 0 111 5 93 7 0 0 0  
[root@localhost ~]# more iostat  
Linux 2.6.32-696.el6.i686 (localhost.localdomain) 13/10/17 _i686_ (1 CPU)  
  
avg-cpu:  %user   %nice %system %iowait  %steal   %idle  
           0,06    0,00    0,11    0,46    0,00   99,36  
  
Device:            tps    Blk_read/s    Blk_wrtn/s    Blk_read    Blk_wrtn  
sda                 2,27         36,76      1039,98      280014      7922962  
dm-0                130,57        35,53      1039,98      270706      7922944  
dm-1                 0,04         0,32         0,00        2400         0  
  
avg-cpu:  %user   %nice %system %iowait  %steal   %idle  
           1,64    0,00    6,09   17,56    0,00   74,71
```

La salida de vmstat durante este tiempo es:

procs		-----memory-----				---swap--		-----io----		--system--		-----cpu-----				
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
0	0	0	1350772	14160	1456256	0	0	19	520	11	12	0	0	99	0	0
0	0	0	1350764	14172	1456264	0	0	0	26	22	24	0	0	100	0	0
0	1	0	2379704	14188	440240	0	0	0	50814	239	90	2	6	60	32	0
0	2	0	2101448	14204	712368	0	0	0	53190	245	115	1	5	0	94	0
2	1	0	1775080	14224	1036000	0	0	0	63169	375	148	3	7	0	89	0
0	1	0	1460616	14244	1347304	0	0	0	61991	320	159	2	6	0	92	0
0	1	0	1102504	14268	1699560	0	0	2	74843	351	159	3	7	0	90	0
0	1	0	797960	14284	1998568	0	0	0	59709	315	141	2	6	0	92	0
1	0	0	565832	14304	2228452	0	0	0	55814	378	67	10	12	0	78	0
1	0	0	565980	14304	2228452	0	0	0	0	118	5	95	5	0	0	0
1	0	0	566004	14312	2228452	0	0	0	4	136	8	97	3	0	0	0
1	0	0	566028	14320	2228456	0	0	0	4	140	7	93	7	0	0	0
1	0	0	566052	14320	2228456	0	0	0	0	111	5	93	7	0	0	0

Tras revisar la salida de vmstat, vemos varios datos interesantes:

- El consumo de memoria se dispara, llegando a ocupar casi la mitad de la capacidad total de la misma
- Vemos que en las colas de preparados y bloqueados se van alternando 3 procesos, lo cual corresponde a los 3 procesos que hemos iniciado (vmstat, iostat y el programa). Estos 3 procesos se van alternando entre ambas colas, terminando finalmente solo uno de ellos.
- En la sección de entrada/salida, vemos un incremento muy destacable del trafico de escritura a disco en la parte media de la tabla. Eso probablemente se deban a las peticiones de escritura a disco tanto del propio programa con fwrite(), como los redireccionamientos a ficheros de vmstat e iostat.
- En el apartado de sistema, vemos gran cantidad de interrupciones y cambios de contexto por la parte media de la tabla, justo coincidiendo con el aumento del trafico de entrada/salida.  
El incremento es especialmente notable en el caso de las interrupciones, lo cual corresponde con lo esperado en una secuencia de operaciones de entrada/salida con 3 procesos efectuando peticiones de escritura a disco.
- En el apartado de CPU, vemos un aumento importante de los tiempos de CPU bloqueada en la parte media de la tabla, los cuales se corresponden con las peticiones de entrada/salida citadas antes. Tras finalizar los bloqueos, se ve un incremento notable del tiempo dedicado a tareas de usuarios.

La salida de iostat es la siguiente:

```
avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           0,06   0,00   0,11    0,46   0,00  99,36
```

```
Device:            tps Blk_read/s  Blk_wrtn/s  Blk_read  Blk_wrtn
sda                2,27    36,76    1039,98    280014    7922962
dm-0               130,57    35,53    1039,98    270706    7922944
dm-1                0,04     0,32     0,00     2400         0
```

```
avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           1,64   0,00   6,09    17,56   0,00  74,71
```

```
Device:            tps Blk_read/s  Blk_wrtn/s  Blk_read  Blk_wrtn
sda             101,87     0,00   81839,81         0    349456
dm-0          14636,77     0,00  117094,15         0   499992
dm-1              0,00     0,00     0,00         0         0
```

```
avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           1,44   0,00   4,09   94,47   0,00   0,00
```

```
Device:            tps Blk_read/s  Blk_wrtn/s  Blk_read  Blk_wrtn
sda             127,16     0,00  126323,08         0   525504
dm-0          16098,08     0,00  128784,62         0   535744
dm-1              0,00     0,00     0,00         0         0
```

```
avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           3,38   0,00   7,00   89,61   0,00   0,00
```

```
Device:            tps Blk_read/s  Blk_wrtn/s  Blk_read  Blk_wrtn
sda             154,11     0,00  155586,47         0   644128
dm-0          19175,36     0,00  153402,90         0   635088
dm-1              0,00     0,00     0,00         0         0
```

```
avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           1,96   0,00   6,85   91,20   0,00   0,00
```

```
Device:            tps Blk_read/s  Blk_wrtn/s  Blk_read  Blk_wrtn
sda             147,43     0,00  149085,57         0   609760
dm-0          18317,36     0,00  146538,88         0   599344
dm-1              0,00     0,00     0,00         0         0
```

```
avg-cpu:  %user  %nice %system %iowait  %steal   %idle
           2,72   0,00   7,18   90,10   0,00   0,00
```

Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn
sda	175,99	3,96	178118,81	16	719600
dm-0	22487,13	3,96	179893,07	16	726768
dm-1	0,00	0,00	0,00	0	0

avg-cpu: %user %nice %system %iowait %steal %idle  
 1,69 0,00 5,07 93,24 0,00 0,00

Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn
sda	130,19	0,00	129277,29	0	535208
dm-0	15912,32	0,00	127298,55	0	527016
dm-1	0,00	0,00	0,00	0	0

avg-cpu: %user %nice %system %iowait %steal %idle  
 2,48 0,00 6,68 90,84 0,00 0,00

Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn
sda	151,24	0,00	150829,70	0	609352
dm-0	19265,59	0,00	154124,75	0	622664
dm-1	0,00	0,00	0,00	0	0

avg-cpu: %user %nice %system %iowait %steal %idle  
 70,59 0,00 29,41 0,00 0,00 0,00

Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn
sda	297,06	0,00	297482,35	0	202288
dm-0	8948,53	0,00	71588,24	0	48680
dm-1	0,00	0,00	0,00	0	0

avg-cpu: %user %nice %system %iowait %steal %idle  
 95,24 0,00 4,76 0,00 0,00 0,00

Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn
sda	7,94	0,00	63,49	0	40
dm-0	7,94	0,00	63,49	0	40
dm-1	0,00	0,00	0,00	0	0

avg-cpu: %user %nice %system %iowait %steal %idle  
 95,45 0,00 4,55 0,00 0,00 0,00

Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn
sda	0,00	0,00	0,00	0	0
dm-0	0,00	0,00	0,00	0	0

dm-1	0,00	0,00	0,00	0	0
------	------	------	------	---	---

avg-cpu:	%user	%nice	%system	%iowait	%steal	%idle
	93,33	0,00	6,67	0,00	0,00	0,00

Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn
sda	6,67	0,00	66,67	0	40
dm-0	8,33	0,00	66,67	0	40
dm-1	0,00	0,00	0,00	0	0

avg-cpu:	%user	%nice	%system	%iowait	%steal	%idle
	94,12	0,00	5,88	0,00	0,00	0,00

Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn
sda	7,84	0,00	62,75	0	32
dm-0	7,84	0,00	62,75	0	32
dm-1	0,00	0,00	0,00	0	0

Vemos que las medias de uso de la CPU coinciden aproximadamente con las dadas por *vmstat*.

En las medias de uso de disco, coinciden los valores de bloques leídos/segundo, que estan en 0 durante la mayor parte del tiempo; pero los valores bloques escritos/segundo difieren bastante de los dados por *vmstat*, de los que *iostat* da unas cifras mucho mas altas que el anterior.



## 6. Analizando el efecto en la creación de procesos de otro programa en ejecución

En esta ocasión, tenemos que monitorizar el efecto de un programa en ejecución en cuanto a la creación de procesos.

El código del programa es este:

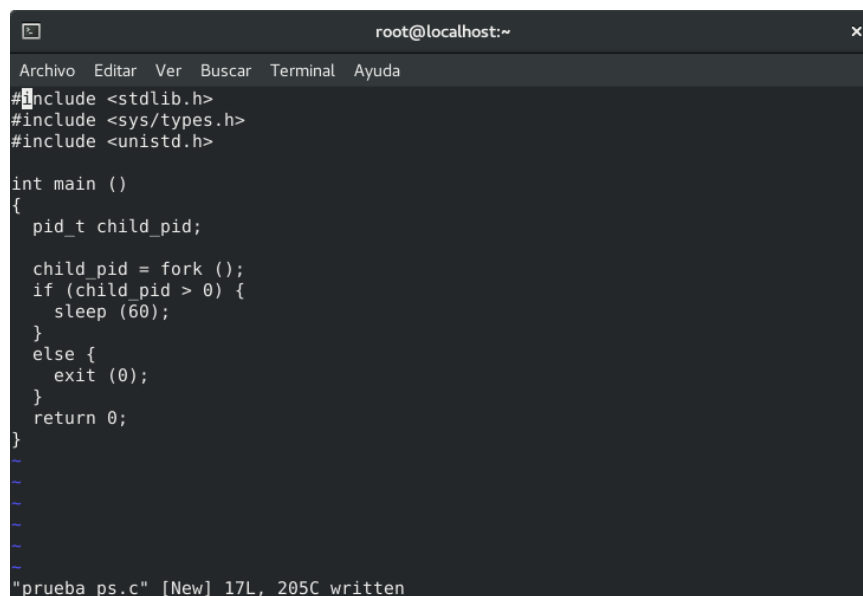
```
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main ()
{
    pid_t child_pid;

    child_pid = fork ();
    if (child_pid > 0) {
        sleep (60);
    }
    else {
        exit (0);
    }
    return 0;
}
```

Este programa creará un proceso hijo, y se mantendrá en ejecución durante 60 segundos. El proceso hijo morirá tras ser creado.

Copiamos el código usando vi, y lo guardamos con el nombre “prueba\_ps.c”

A screenshot of a terminal window titled "root@localhost:~". The window contains the C code for "prueba\_ps.c". The code is as follows: 

```
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main ()
{
    pid_t child_pid;

    child_pid = fork ();
    if (child_pid > 0) {
        sleep (60);
    }
    else {
        exit (0);
    }
    return 0;
}
```

 At the bottom of the terminal, it says: 

```
"prueba_ps.c" [New] 17L, 205C written
```

Compilamos el código y lo ejecutamos. Para poder tener la terminal ssh libre, ejecutamos el programa desde la terminal de la propia máquina virtual.



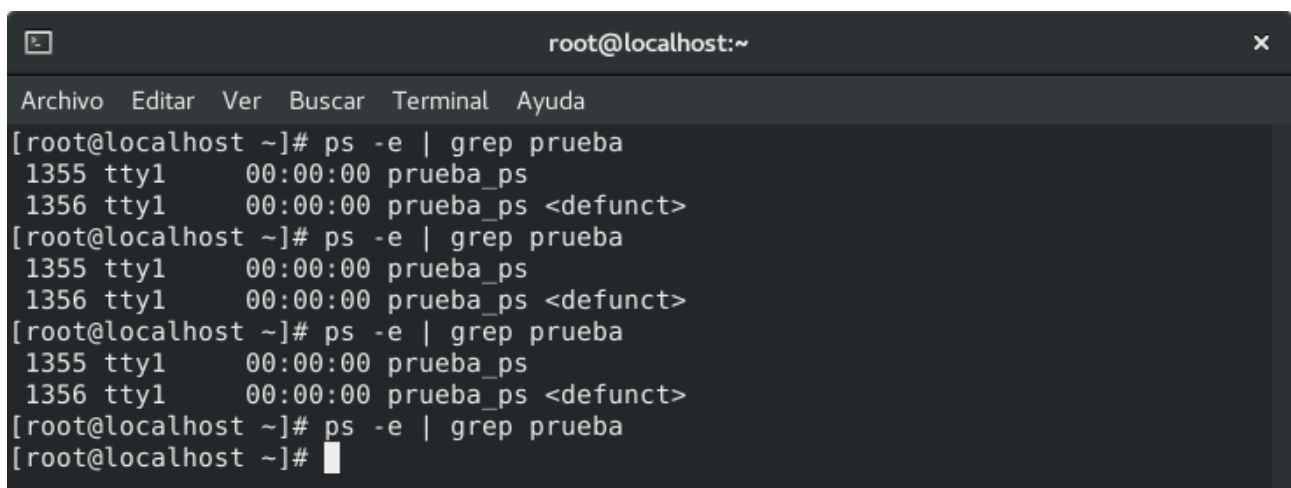
```
CentOS 6.9 [Corriendo] - Oracle VM VirtualBox x
Archivo Máquina Ver Entrada Dispositivos Ayuda
CentOS release 6.9 (Final)
Kernel 2.6.32-696.el6.i686 on an i686

localhost login:

CentOS release 6.9 (Final)
Kernel 2.6.32-696.el6.i686 on an i686

localhost login: root
Password:
Last login: Sat Oct 14 15:18:28 from 10.0.2.2
[root@localhost ~]# gcc prueba_ps.c -o prueba_ps
[root@localhost ~]# ./prueba_ps
—
```

Mientras el programa se mantiene en ejecución, vemos su efecto usando `ps`



```
root@localhost:~ x
Archivo Editar Ver Buscar Terminal Ayuda
[root@localhost ~]# ps -e | grep prueba
1355 tty1 00:00:00 prueba_ps
1356 tty1 00:00:00 prueba_ps <defunct>
[root@localhost ~]# ps -e | grep prueba
1355 tty1 00:00:00 prueba_ps
1356 tty1 00:00:00 prueba_ps <defunct>
[root@localhost ~]# ps -e | grep prueba
1355 tty1 00:00:00 prueba_ps
1356 tty1 00:00:00 prueba_ps <defunct>
[root@localhost ~]# ps -e | grep prueba
[root@localhost ~]#
```

Vemos que el proceso ha creado un proceso hijo, con PID consecutivo al del proceso padre. Y vemos que, tal y como estaba previsto, el proceso hijo muere; pero se mantiene en la tabla de procesos hasta que finaliza el proceso padre.

Para corroborar los datos, repetimos el proceso ejecutando el programa, y usando el comando `ps -efjH`, el cual muestra las relaciones padre/hijo, y el tiempo de uso de la CPU

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[root@localhost ~]# ps -efjH | grep prueba  
root      1403  1237  1402  1237  0 16:28 pts/0      00:00:00      grep prueba  
root      1400  1337  1400  1337  0 16:28 tty1      00:00:00      ./prueba_ps  
root      1401  1400  1400  1337  0 16:28 tty1      00:00:00      [prueba_ps]  
<defunct>  
[root@localhost ~]# ps -efjH | grep prueba  
root      1400  1337  1400  1337  0 16:28 tty1      00:00:00      ./prueba_ps  
root      1401  1400  1400  1337  0 16:28 tty1      00:00:00      [prueba_ps]  
<defunct>  
[root@localhost ~]# ps -efjH | grep prueba  
root      1407  1237  1406  1237  0 16:28 pts/0      00:00:00      grep prueba  
root      1400  1337  1400  1337  0 16:28 tty1      00:00:00      ./prueba_ps  
root      1401  1400  1400  1337  0 16:28 tty1      00:00:00      [prueba_ps]  
<defunct>  
[root@localhost ~]# ps -efjH | grep prueba  
root      1409  1237  1408  1237  0 16:28 pts/0      00:00:00      grep prueba  
root      1400  1337  1400  1337  0 16:28 tty1      00:00:00      ./prueba_ps  
root      1401  1400  1400  1337  0 16:28 tty1      00:00:00      [prueba_ps]  
<defunct>  
[root@localhost ~]# ps -efjH | grep prueba  
[root@localhost ~]#
```

Vemos que, efectivamente, el proceso difunto es el proceso hijo, el cual se mantiene en la tabla de procesos marcado como “defunct”; y el que el proceso padre se mantiene en ejecución durante esos 60 segundos. Una vez muere el proceso padre, el proceso hijo también desaparece.

## 7. Analizando impacto en memoria de la ejecución de un proceso

Volvemos a analizar el impacto de otro programa en ejecución, esta vez, sobre la memoria.

El código a ejecutar es el siguiente:

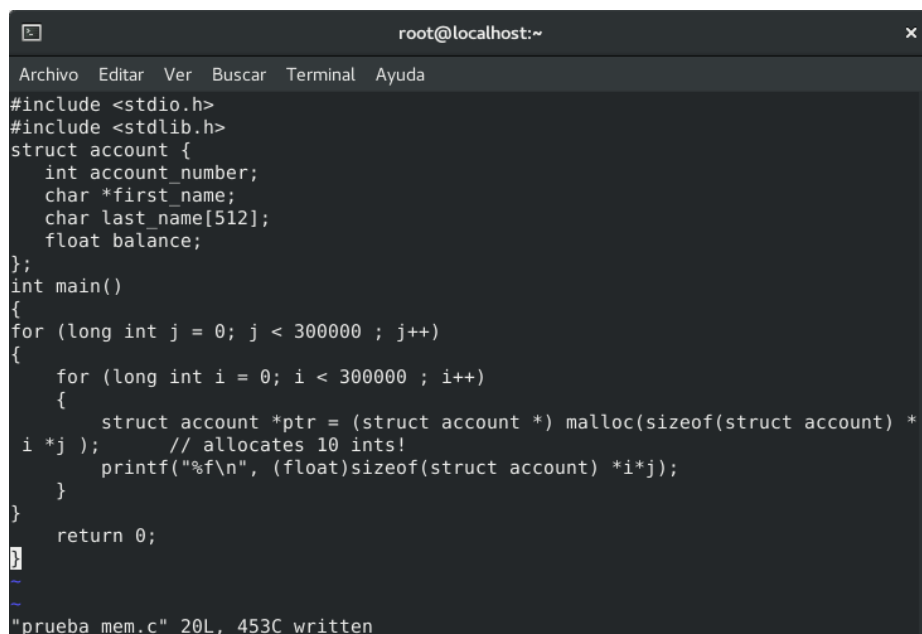
```
#include <stdio.h>
#include <stdlib.h>
struct account {
    int account_number;
    char *first_name;
    char last_name[512];
    float balance;
};
int main()
{
    for (long int j = 0; j < 300000 ; j++)
    {
        for (long int i = 0; i < 300000 ; i++)
        {
            struct account *ptr = (struct account *) malloc(sizeof(struct account) * i * j );    // allocates 10 in
            printf("%f\n", (float)sizeof(struct account) *i*j);
        }
    }
    return 0;
}
```

Este código usa un struct, el cual contiene un int, un float y dos cadenas de char.

Durante su ejecución, va generando punteros a estructuras basadas en el struct, reservando espacios de memoria cada vez mayores, y mostrando por pantalla el tamaño del espacio reservado en memoria.

Para analizar su efecto usaremos los comandos *vmstat* y *free*.

Comenzamos copiando el código, con el nombre de prueba\_mem.c, y compilándolo



The screenshot shows a terminal window titled "root@localhost:~". The menu bar includes "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The code being executed is the same as shown in the previous block. The output of the program is visible at the bottom of the terminal, showing the memory size calculation for the first iteration: "prueba\_mem.c" 20L, 453C written.

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[root@localhost ~]# gcc prueba_mem.c -o prueba_mem  
prueba_mem.c: En la función 'main':  
prueba_mem.c:11: error: sólo se permiten las declaraciones iniciales del ciclo '  
for' en modo C99  
prueba_mem.c:11: nota: use la opción -std=c99 o -std=gnu99 para compilar su código  
prueba_mem.c:13: error: sólo se permiten las declaraciones iniciales del ciclo '  
for' en modo C99  
[root@localhost ~]#
```

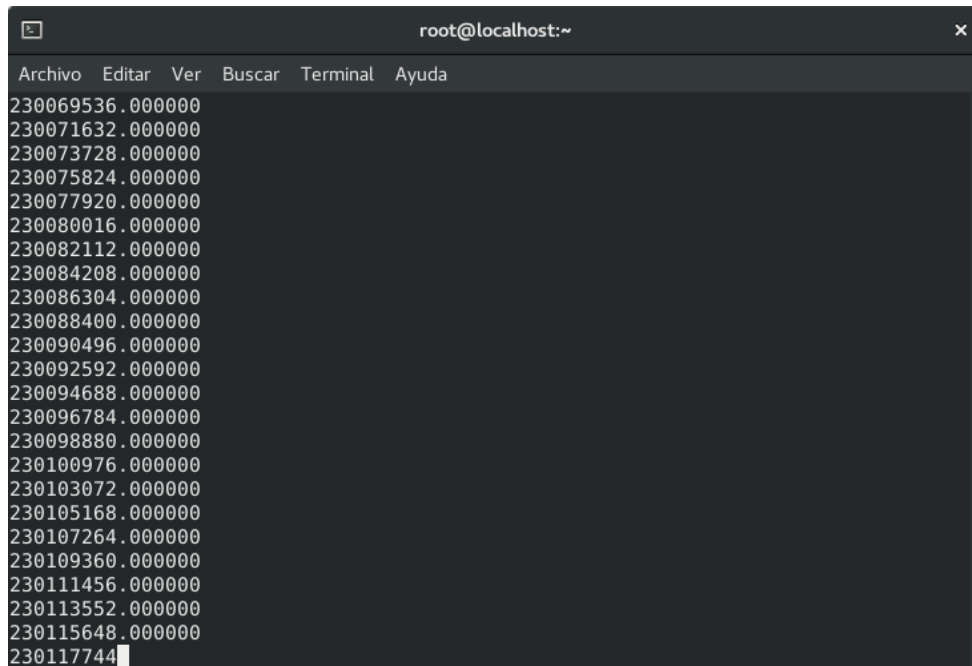
Al compilar nos sale un pequeño error. Recompilamos con el argumento indicado en el aviso, y lo solucionamos.

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[root@localhost ~]# gcc prueba_mem.c -o prueba_mem  
prueba_mem.c: En la función 'main':  
prueba_mem.c:11: error: sólo se permiten las declaraciones iniciales del ciclo '  
for' en modo C99  
prueba_mem.c:11: nota: use la opción -std=c99 o -std=gnu99 para compilar su código  
prueba_mem.c:13: error: sólo se permiten las declaraciones iniciales del ciclo '  
for' en modo C99  
[root@localhost ~]# gcc prueba_mem.c -o prueba_mem -std=gnu99  
[root@localhost ~]#
```

Una vez compilado, iniciamos el programa

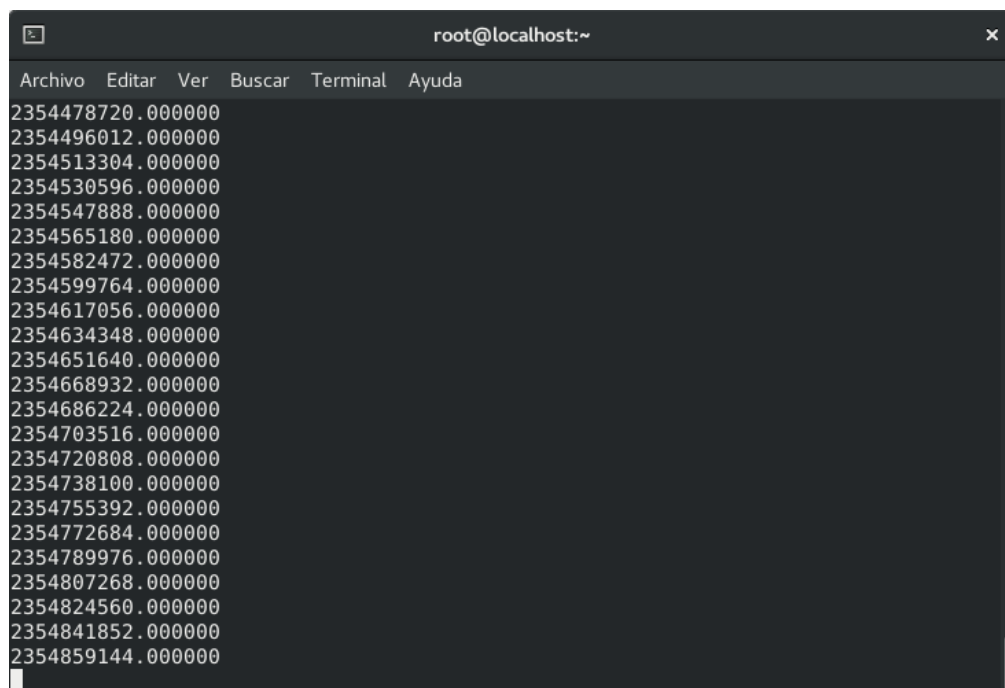
```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
[root@localhost ~]# ./prueba_mem
```

Una vez iniciado el programa, comenzamos a ver los valores de salida del mismo, correspondientes al espacio de memoria ocupado



```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
230069536.000000  
230071632.000000  
230073728.000000  
230075824.000000  
230077920.000000  
230080016.000000  
230082112.000000  
230084208.000000  
230086304.000000  
230088400.000000  
230090496.000000  
230092592.000000  
230094688.000000  
230096784.000000  
230098880.000000  
230100976.000000  
230103072.000000  
230105168.000000  
230107264.000000  
230109360.000000  
230111456.000000  
230113552.000000  
230115648.000000  
230117744.
```

Vemos como, progresivamente, los valores devueltos van siendo cada vez mayores, lo cual indica que la memoria reservada es cada vez mayor.



```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
2354478720.000000  
2354496012.000000  
2354513304.000000  
2354530596.000000  
2354547888.000000  
2354565180.000000  
2354582472.000000  
2354599764.000000  
2354617056.000000  
2354634348.000000  
2354651640.000000  
2354668932.000000  
2354686224.000000  
2354703516.000000  
2354720808.000000  
2354738100.000000  
2354755392.000000  
2354772684.000000  
2354789976.000000  
2354807268.000000  
2354824560.000000  
2354841852.000000  
2354859144.000000  
.
```

Mostramos el consumo de memoria con *free*, indicando como unidad de medida MiB, con un intervalo de repetición de 20 segundos.

```
CentOS 6.9 [Corriendo] - Oracle VM VirtualBox x
[root@localhost ~]# free -m -s 20
      total        used        free      shared    buffers     cached
Mem:      2837         128       2708          0          5         39
-/+ buffers/cache:      84       2752
Swap:      1983           0       1983
```

Vemos como el consumo de memoria, poco a poco, se va incrementando.

```
CentOS 6.9 [Corriendo] - Oracle VM VirtualBox x
Mem:      2837         128       2708          0          5         39
-/+ buffers/cache:      84       2752
Swap:      1983           0       1983

      total        used        free      shared    buffers     cached
Mem:      2837         128       2708          0          5         39
-/+ buffers/cache:      84       2752
Swap:      1983           0       1983

      total        used        free      shared    buffers     cached
Mem:      2837         129       2707          0          5         39
-/+ buffers/cache:      84       2752
Swap:      1983           0       1983

      total        used        free      shared    buffers     cached
Mem:      2837         129       2707          0          5         39
-/+ buffers/cache:      84       2752
Swap:      1983           0       1983
```

Cambiamos la unidad de medida a KiB para ver mejor los cambios

```
CentOS 6.9 [Corriendo] - Oracle VM VirtualBox x
[root@localhost ~]# free -k -s 20
      total        used        free      shared    buffers     cached
Mem:    2905316    132768    2772548         200        5604        40088
-/+ buffers/cache:    87076    2818240
Swap:    2031612         0    2031612
```

```
CentOS 6.9 [Corriendo] - Oracle VM VirtualBox x
[root@localhost ~]# free -k -s 20
      total        used        free      shared    buffers     cached
Mem:    2905316    132768    2772548         200        5604        40088
-/+ buffers/cache:    87076    2818240
Swap:    2031612         0    2031612

      total        used        free      shared    buffers     cached
Mem:    2905316    132776    2772540         200        5612        40088
-/+ buffers/cache:    87076    2818240
Swap:    2031612         0    2031612

      total        used        free      shared    buffers     cached
Mem:    2905316    132776    2772540         200        5612        40088
-/+ buffers/cache:    87076    2818240
Swap:    2031612         0    2031612

      total        used        free      shared    buffers     cached
Mem:    2905316    132776    2772540         200        5612        40088
-/+ buffers/cache:    87076    2818240
Swap:    2031612         0    2031612
```



Vemos como el consumo de memoria va aumentando lentamente.

CentOS 6.9 [Corriendo] - Oracle VM VirtualBox						
Mem:	2905316	132776	2772540	200	5620	40088
-/+ buffers/cache:		87068	2818248			
Swap:	2031612	0	2031612			
	total	used	free	shared	buffers	cached
Mem:	2905316	132776	2772540	200	5620	40088
-/+ buffers/cache:		87068	2818248			
Swap:	2031612	0	2031612			
	total	used	free	shared	buffers	cached
Mem:	2905316	132900	2772416	200	5644	40088
-/+ buffers/cache:		87168	2818148			
Swap:	2031612	0	2031612			
	total	used	free	shared	buffers	cached
Mem:	2905316	132900	2772416	200	5644	40088
-/+ buffers/cache:		87168	2818148			
Swap:	2031612	0	2031612			
	total	used	free	shared	buffers	cached
Mem:	2905316	132900	2772416	200	5644	40088
-/+ buffers/cache:		87168	2818148			
Swap:	2031612	0	2031612			
	total	used	free	shared	buffers	cached
Mem:	2905316	132900	2772416	200	5644	40088
-/+ buffers/cache:		87168	2818148			
Swap:	2031612	0	2031612			
	total	used	free	shared	buffers	cached
Mem:	2905316	132900	2772416	200	5644	40088
-/+ buffers/cache:		87168	2818148			
Swap:	2031612	0	2031612			
	total	used	free	shared	buffers	cached
Mem:	2905316	132900	2772416	200	5644	40088
-/+ buffers/cache:		87168	2818148			
Swap:	2031612	0	2031612			

Contrastamos los valores usando *vmstat*

```
CentOS 6.9 [Corriendo] - Oracle VM VirtualBox
```

```
[root@localhost ~]# vmstat -S K
```

procs		-----memory-----				---swap--		-----io----		--system--		-----cpu-----				
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
1	0	0	2772548	5676	40112	0	0	46	1	4401	6727	33	45	22	0	0

```
[root@localhost ~]# free -k
```

	total	used	free	shared	buffers	cached
Mem:	2905316	132768	2772548	200	5684	40112
-/+ buffers/cache:		86972	2818344			
Swap:	2031612	0	2031612			

```
[root@localhost ~]# _
```

Pasamos a hacer un nuevo seguimiento usando *vmstat*

```
CentOS 6.9 [Corriendo] - Oracle VM VirtualBox x
[root@localhost ~]# vmstat 20
procs -----memory----- --swap-- -----io----- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo    in   cs us sy id wa st
 1  0       0 2772424   5700  40112    0    0    42     1 4486 6867 34 46 21  0
0
_
```

Corroboramos los mismos datos obtenidos con *free*. Se percibe un importante numero de interrupciones, pero sin ningún patrón destacable.

```
CentOS 6.9 [Corriendo] - Oracle VM VirtualBox x
1 0 0 2772416 5700 40112 0 0 0 0 4210 6014 41 57 3 0
0
1 0 0 2772416 5700 40112 0 0 0 0 4126 6171 42 56 2 0
0
1 0 0 2772416 5700 40112 0 0 0 1 4123 6539 43 55 2 0
0
1 0 0 2772416 5700 40112 0 0 0 0 4160 6520 43 55 2 0
0
2 0 0 2772416 5700 40112 0 0 0 0 4154 6717 44 54 2 0
0
1 0 0 2772416 5716 40112 0 0 0 1 4036 6975 45 53 2 0
0
2 0 0 2772416 5716 40112 0 0 0 0 4091 6757 44 54 2 0
0
1 0 0 2772416 5716 40112 0 0 0 0 4157 6003 40 57 3 0
0
2 0 0 2772292 5748 40116 0 0 0 4 4270 6032 40 57 3 0
0
1 0 0 2772292 5748 40116 0 0 0 0 4224 6564 43 55 2 0
0
1 0 0 2772292 5748 40116 0 0 0 1 4147 6450 43 55 3 0
0
1 0 0 2772292 5756 40116 0 0 0 1 4075 6337 42 55 3 0
0
_
```

Una vez detenido el programa, observamos como se libera la memoria usada por el programa

```
root@localhost:~  
Archivo Editar Ver Buscar Terminal Ayuda  
187231014828.000000  
187231689216.000000  
187232363604.000000  
187233037992.000000  
187233712380.000000  
187234386768.000000  
187235061156.000000  
187235735544.000000  
^C  
[root@localhost ~]# free -k  
              total        used        free      shared    buffers     cached  
Mem:          2905316      107720      2797596          200         5780         40116  
-/+ buffers/cache:          61824      2843492  
Swap:          2031612           0         2031612  
[root@localhost ~]# free -k  
              total        used        free      shared    buffers     cached  
Mem:          2905316      107596      2797720          200         5780         40116  
-/+ buffers/cache:          61700      2843616  
Swap:          2031612           0         2031612
```

## 8. Filtrando los datos del comando sar

El comando *sar* nos permite obtener estadísticas de uso del sistema, mostrando información de diferentes parámetros.

Nosotros lo vamos a usar para mostrar información de las diferentes núcleos de la CPU.

Vamos a crear un script que, para cada uno de sus núcleos, muestre sus porcentajes de uso en formato XML.. El script invocará a *sar -P* con el numero de cada procesador, y filtrará los datos obtenidos para mostrar únicamente los necesarios.

*sar* genera sus estadísticas cada 10 minutos (salvo que lo ejecutemos en tiempo real indicando manualmente la espera), por lo cual, la salida del comando puede incluir muchas filas, correspondientes a cada estadística generada.

Nuestro script solo tomará el último valor, mostrado en la penúltima fila.

El código es el siguiente:

```
#!/bin/bash

#Obtenemos el numero de nucleos de la CPU
NUM_CPU=$(nproc --all)

#Asociamos cada posición del vector a la descripción de uno de los campos
field_name=("usuario", "buen uso", "sistema", "espera a entrada/salida", "robo de ciclo", "ociosidad")

#Por cada nucleo, mostramos sus estadísticas
for i in $(seq 0 1 $(($NUM_CPU-1))); do

    #Guardamos la última medida de sar en una variable
    last_time=$(sar -P $i | tail -n 2 | head -n 1 | tr -s " " | cut -d " " -f 3-)

    #Establecemos el índice para el vector
    index=0

    #Recorremos todos los campos y mostramos sus valores
    for j in $last_time; do

        #Mostramos el valor en XML usando Heredoc
        cat - <<EOF

        <module>
        <name><![CDATA[SAR: CPU$i $j]]></name>
        <description>Muestra el % de tiempo de ${field_name[$index]} de la cpu $i</description>
        <type><![CDATA[generic_data]]></type>
        <![CDATA[15,1]]>
        </module>

        EOF

        index=$((index+1))

    done;
done;
```

En este código, usamos dos bucles, uno para mostrar los datos de cada procesador, y otro para imprimir cada uno de los campos que devuelve `sar`.

El código usa las siguientes variables:

**NUM\_CPU:** Número de procesadores

**i :** Variable índice para seleccionar procesador

**last\_time:** lista con las últimas medidas obtenidas por `sar`, a la que se le han extraído los dos primeros campos, referentes a la hora y al número de CPU.

**field\_name:** array que almacena los nombres de los diferentes campos

**index:** variable índice, usada para obtener los datos de `field_name`.

El script ejecuta el comando `sar -P` con el número de procesador dado por la variable `i`. La salida del comando la filtra, para quedarse únicamente con los valores de la última medición, y para eliminar los campos referentes a la hora y al número de procesador (campos 1 y 2). Esta salida filtrada se guarda en la variable `last_time`.

Seguidamente, inicializamos la variable `index`, que será usada posteriormente para obtener la descripción correspondiente a nuestro campo.

Una vez inicializado el índice, pasamos a ejecutar un bucle `for`, que va recorriendo los valores de nuestra lista (almacenada en `last_time`), e imprimiendo sus valores y sus descripciones en formato XML.

Para imprimir las descripciones, obtenemos el valor almacenado en la posición de `field_name` indicada por la variable `index`.

Este proceso se repetirá para cada uno de los campos obtenidos para cada procesador.

Tras ejecutarlo, la salida es similar a esta:

```
<module>
<name><![CDATA[SAR: CPU0 0,00]]></name>
<description>Muestra el % de tiempo de usuario, de la cpu 0</description>
<type><![CDATA[generic_data]]></type>
<![CDATA[15,1]]>
</module>

<module>
<name><![CDATA[SAR: CPU0 0,00]]></name>
<description>Muestra el % de tiempo de buen uso, de la cpu 0</description>
<type><![CDATA[generic_data]]></type>
<![CDATA[15,1]]>
</module>

<module>
<name><![CDATA[SAR: CPU0 0,01]]></name>
<description>Muestra el % de tiempo de sistema, de la cpu 0</description>
<type><![CDATA[generic_data]]></type>
<![CDATA[15,1]]>
</module>

<module>
<name><![CDATA[SAR: CPU0 0,00]]></name>
<description>Muestra el % de tiempo de espera a entrada/salida, de la cpu 0</description>
<type><![CDATA[generic_data]]></type>
<![CDATA[15,1]]>
</module>

<module>
<name><![CDATA[SAR: CPU0 0,00]]></name>
<description>Muestra el % de tiempo de robo de ciclo, de la cpu 0</description>
<type><![CDATA[generic_data]]></type>
<![CDATA[15,1]]>
</module>

<module>
<name><![CDATA[SAR: CPU0 99,99]]></name>
<description>Muestra el % de tiempo de ociosidad de la cpu 0</description>
<type><![CDATA[generic_data]]></type>
<![CDATA[15,1]]>
</module>

<module>
<name><![CDATA[SAR: CPU1 0,00]]></name>
<description>Muestra el % de tiempo de usuario, de la cpu 1</description>
<type><![CDATA[generic_data]]></type>
<![CDATA[15,1]]>
</module>

<module>
```

```

<name><![CDATA[SAR: CPU1 0,00]]></name>
<description>Muestra el % de tiempo de buen uso, de la cpu 1</description>
<type><![CDATA[generic_data]]></type>
<![CDATA[15,1]]>
</module>

<module>
<name><![CDATA[SAR: CPU1 0,00]]></name>
<description>Muestra el % de tiempo de sistema, de la cpu 1</description>
<type><![CDATA[generic_data]]></type>
<![CDATA[15,1]]>
</module>

<module>
<name><![CDATA[SAR: CPU1 0,00]]></name>
<description>Muestra el % de tiempo de espera a entrada/salida, de la cpu 1</description>
<type><![CDATA[generic_data]]></type>
<![CDATA[15,1]]>
</module>

<module>
<name><![CDATA[SAR: CPU1 0,00]]></name>
<description>Muestra el % de tiempo de robo de ciclo, de la cpu 1</description>
<type><![CDATA[generic_data]]></type>
<![CDATA[15,1]]>
</module>

<module>
<name><![CDATA[SAR: CPU1 99,99]]></name>
<description>Muestra el % de tiempo de ociosidad de la cpu 1</description>
<type><![CDATA[generic_data]]></type>
<![CDATA[15,1]]>
</module>

```

Para comprobar que el bucle principal funciona, le hemos añadido una núcleo más a la CPU de la máquina virtual, la cual ahora tiene 2 núcleos.

Vemos que, por cada CPU, se muestran 6 estructuras XML, una por cada dato que queremos mostrar. En la línea *name* se muestra el procesador al que corresponde el dato, seguido por el dato en sí. En *description* se describe la medida a la que se corresponde ese dato, y en que procesador se ha tomado.

## 9. Conclusiones

La monitorización del sistema es una de las tareas mas importantes que debe saber hacer un administrador de sistemas. La interpretación de los datos obtenidos con cada una de las herramientas nos permite descubrir un posible problema, y el origen del mismo, lo cual facilita el poder solucionarlo correctamente.

Los comandos *vmstat*, *iostat* y *free* permiten obtener información muy detallada sobre el estado de los recursos hardware del sistema.

El comando *ps* nos permite observar los procesos que están en ejecución dentro del sistema, y las interrelaciones entre ellos.

El comando *sar* nos permite obtener estadísticas en tiempo real de los recursos, y generar informes sobre ellos.

Todos estos comandos, junto a algunos mas, nos permiten saber el estado actual del sistema, y detectar posibles problemas incluso antes de que sucedan.