

SISTEMAS OPERATIVOS EN RED

UT5 - Monitorización del sistema Linux y gestión remota.

Mario García Alcázar

Esta obra esta sujeta a la Licencia Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-nc-nd/3.0/es/> o envíe una carta Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.



Última revisión Julio de 2017.

Índice de contenido

Sumario

1. Introducción.....	4
2. Monitorización del sistema.....	5
2.1. Analizar el estado general del ordenador.....	5
2.2. Configuración del arranque del sistema.....	8
2.2.1 Configuración de servicios.....	11
2.3. Análisis de los sucesos del sistema.....	14
2.4. Desinstalación de servicios no necesarios.....	15
2.5. Eliminación de paquetes no utilizados.....	15
2.6. Programa localepurge.....	15
2.7. Programa preload.....	16
2.8. Programa Bleachbit.....	17
3. Administración remota de sistemas Linux.....	19
3.1. Servicio SSH.....	20
3.2. Servicio VNC.....	23

1. Introducción.

En los temas anteriores hemos tratado como instalar un sistema Linux y configurar algunos servicios importantes, como SAMBA, LDAP o CUPS. No obstante una de las principales tareas de un administrador informático es gestionar el correcto funcionamiento del ordenador.

En este tema, estudiaremos de qué forma podemos analizar el rendimiento de un sistema Linux para mantener este en unos niveles de rendimiento óptimos.

Por otra parte, no es algo raro, que en cualquier empresa con servidores Linux, el o los administradores accedan a ellos remotamente desde otros equipos, ubicados en la empresa o fuera de ella. A lo largo de este documento estudiaremos los principales protocolos de acceso remoto a equipos.

2. Monitorización del sistema.

Como se introdujo en el punto anterior, una de las principales funciones de un administrador informático es la de asegurar el correcto funcionamiento del sistema, esto puede englobar entre otras las siguientes tareas:

- Analizar el estado de la memoria RAM, CPU y discos duros.
- Configurar el arranque del ordenador.
- Gestionar los servicios.
- Analizar los sucesos del sistema.
- Optimización del funcionamiento.

En este apartado enumeramos algunas tareas necesarias para revisar y mejorar el funcionamiento del sistema.

2.1. Analizar el estado general del ordenador.

Es vital para el correcto funcionamiento del equipo, que en condiciones normales no esté saturado ninguno de sus recursos hardware básicos, estos son: la **CPU**, la **memoria RAM** y los **discos duros**. En caso contrario, probablemente se producirán graves aumentos en los tiempos de respuesta de las aplicaciones y del propio sistema operativo.

Seguidamente se aporta una tabla con algunos de los comandos que podemos usar para revisar el estado los elementos antes mencionados.

Comando	Descripción
free	Muestra el estado de la memoria RAM del equipo
top	Muestra un monitor en tiempo real con los procesos del sistema y su consumo de RAM y CPU.
df	Permite ver el estado de los discos duros del ordenador
lshw	Permite ver la lista completa de elementos hardware del ordenador. Muestra mas información si se ejecuta como superusuario.

Sistemas Operativos en Red

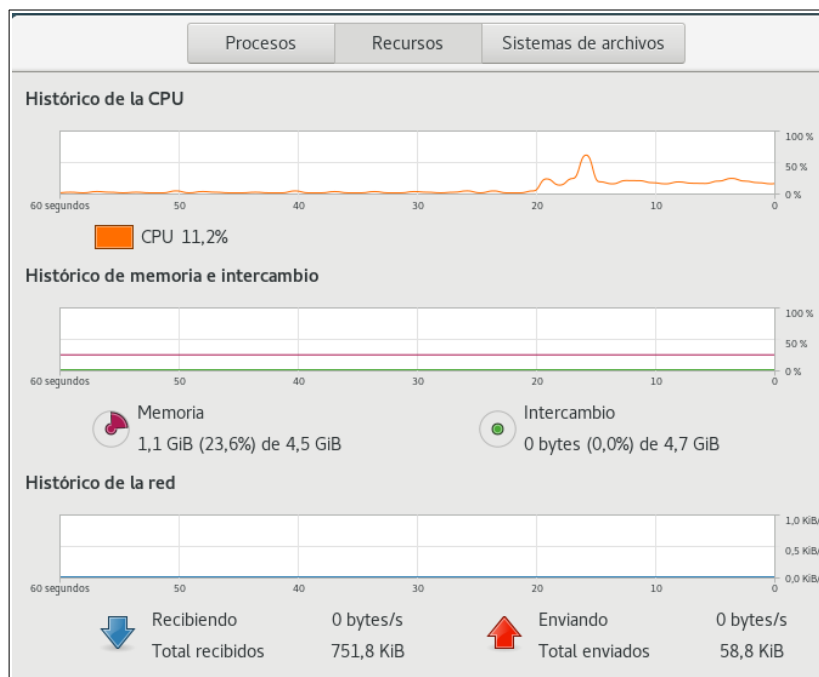
UT5 - Monitorización del sistema Linux y gestión remota

Por otra parte, si estamos trabajando en modo gráfico, siempre podremos utilizar alguna herramienta de tipo **monitor de sistema**, que en el caso de Debian 9 nos muestra las siguientes pantallas de datos:

- **Lista de procesos en tiempo real:**

Procesos Recursos Sistemas de archivos						
Nombre del proceso	Usuario	% CPU	ID	Memoria	Prioridad	
at-spi2-registr...	mario	0	1194	748,0 KiB	Normal	
at-spi-bus-launcher	mario	0	1186	652,0 KiB	Normal	
dbus-daemon	mario	0	1093	1,4 MiB	Normal	
dbus-daemon	mario	0	1191	472,0 KiB	Normal	
dconf-service	mario	0	1277	564,0 KiB	Normal	
deja-dup-monitor	mario	0	1370	1,0 MiB	Normal	
evolution-addressbook-factory	mario	0	1489	3,1 MiB	Normal	
evolution-addressbook-factory-	mario	0	1514	3,2 MiB	Normal	
evolution-alarm-notify	mario	0	1399	10,0 MiB	Normal	
evolution-calendar-factory	mario	0	1356	6,3 MiB	Normal	
evolution-calendar-factory-subp	mario	0	1474	4,9 MiB	Normal	

- **Estado de la CPU, RAM y Red.**



- **Estado de los discos y particiones.**

Procesos Recursos Sistemas de archivos x						
Dispositivo	Carpeta ▼	Tipo	Total	Disponible	Usado	
 /dev/sda1	/	ext4	9,6 GB	4,2 GB	4,9 GB	<div><div></div></div> 53%
 /dev/sda6	/home	ext4	17,1 GB	16,1 GB	178,8 MB	<div><div></div></div> 1%
 /dev/sr0	/media/cdromC	iso9660	58,5 MB	0 bytes	58,5 MB	<div><div></div></div> 100%

2.2. Configuración del arranque del sistema.

En todos los sistemas Linux, **init es el primer proceso que se lanza cuando encendemos el ordenador y se encarga principalmente de gestionar el arranque del sistema operativo**. Este proceso funciona como un demonio y suele tener el valor 1 como PID.

Por todo lo anterior, podemos decir que init es el único proceso que arranca el kernel del sistema operativo y es también el encargado de lanzar al resto de programas/servicios del sistema. En Linux, todo proceso es hijo de init.

Por otra parte, **Init distingue múltiples niveles de ejecución (runlevels)**, cada uno de los cuales puede iniciar su propio conjunto de procesos y servicios. Por ejemplo, podemos tener runlevels donde no se inicie el entorno gráfico o donde no tengamos soporte para la red, etc.

Los runlevels o niveles de ejecución válidos van de 0 a 6. Concretamente:

Nivel	Descripción	¿Configurable?
0	Modo de parada o apagado del equipo	no
1	Modo monousuario sin red. Suele usarse para tareas de mantenimiento.	no
2	Modo gráfico multiusuario. Runlevel por defecto en Debian y Ubuntu	si
3	Modo gráfico multiusuario.	si
4	Modo gráfico multiusuario.	si
5	Modo gráfico multiusuario.	si
6	Modo de reinicio	no

Si queremos **conocer en qué runlevel nos encontramos** solo necesitamos ejecutar el comando **runlevel**


```
mario@c
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
root@debian:/etc/init# runlevel
N 5
root@debian:/etc/init# █
```

Y en el caso de que queramos cambiar de un nivel a otro basta con tener permisos de administrador y usar el comando **init**. Por ejemplo init 6 reiniciaría el sistema.

En este punto, hay que diferenciar entre los tres tipos de sistemas de arranque existentes en Linux. Estos son **systemV**, **systemd** y **upstart**, aunque realmente systemV está muy desfasado y ya no se utiliza, por tanto en este documento solo trataremos los sistemas **systemd** y **upstart**.

- **SystemV.**

En un sistema Linux con arranque SystemV, la configuración del arranque del sistema en los diferentes niveles de ejecución se especificaba en el fichero `/etc/inittab`.

Este fichero consta de una serie de líneas que se ejecutan secuencialmente, e indican un proceso que debe ejecutarse o no, según el nivel de ejecución en el que esté arrancando el sistema.

Actualmente este sistema de arranque ha caído en desuso.

- **Systemd.**

Systemd es un "sistema de inicio" relativamente reciente utilizado por Debian a partir de su versión 8. Las versiones anteriores usaban el sistema de inicio "System V".

Básicamente Systemd es un sistema y administrador de servicios para Linux, compatible con scripts de inicio SysV. Entre las mejoras que ofrece Systemd tenemos:

- **Capacidades de paralelización agresiva usando socket:** systemd crea de una misma vez todos los *sockets* para todos los demonios acelerando así el arranque completo e iniciar más procesos en paralelo. En un segundo paso *systemd* ejecutará a la vez todos los demonios.
- **Activación D-Bus para iniciar servicios:** Un servicio puede ser iniciado la primera vez que es accedido.

- **Seguimiento de procesos utilizando Linux cgroups:** cgroup también llamados grupos de control, es una característica del kernel para crear límites, políticas e incluso explicar el uso de los recursos de ciertos grupos de procesos. cgroup asocia un conjunto de tareas con un conjunto de parámetros, para uno o más subsistemas, proporcionando un control de servicios y tareas, así como todos sus futuros 'hijos' en grupos jerárquico. Un subsistema es un módulo resultado de la agrupación de diversas tareas con el fin de mantener un mejor control sobre estas de forma particular.
- **Mantiene puntos de montaje y automontaje:** Puede utilizarse para montar o desmontar los puntos de montaje, quedando `/etc/fstab` como una fuente de configuración adicional.

En definitiva, **systemd** ha sido creado para ofrecer un inicio mas rápido y flexible que SysV, permitiendo el arranque paralelo de servicios y su inicio basado en la detección de conexión de nueva unidad externa.

Nota: Hasta ahora el **PID1** era para el programa **init**, cosa que ha cambiado en systemd a favor de **/usr/lib/systemd/systemd** y además **systemd** al igual que **Upstart** deja de utilizar el archivo `/etc/inittab`

Otro concepto interesante es que **systemd** utiliza **target** en vez de **runlevels** (0123456) que reciben un nombre (en vez de un número) para identificar un propósito específico, con la posibilidad de realizar más de una acción al mismo tiempo.

Algunos targets pueden heredar todos los servicios de otro target e implementarse con servicios adicionales. La siguiente tabla muestra la similitud entre algunos de los target con los runlevels:

Nivel de ejecución sysvinit	Target systemd	Notas
0	runlevel0.target, poweroff.target	Detener el sistema.
1, s, single	runlevel1.target, rescue.target	Modo monousuario.
3	runlevel3.target, multi-user.target	Multiusuario, no gráfico. Los usuarios pueden usualmente hacer login por medio de múltiples consolas o desde la red.
5	runlevel5.target, graphical.target	Multiusuario, gráfico. Usualmente todos los servicios del nivel de ejecución 3 sumando el login gráfico.
6	runlevel6.target, reboot.target	Reiniciar el sistema.
emergency	emergency.target	Shell de emergencia.

Podemos cambiar de *target* (o modo de ejecución) actual con el comando:

systemctl isolate graphical.target

Esto podría ser equivalente a la orden **# *init* 5** de SysV

En el caso que quieras cambiar el target de arranque por defecto.

El *target* **/etc/systemd/system/default.target** es el *target* predeterminado de arranque, es un enlace simbólico que por defecto apunta a **/lib/systemd/system/graphical.target** por lo que para cambiar de *target* de arranque, bastará con eliminar dicho link y crear uno nuevo apuntando al nuevo *target*.

- **Upstart** es un demonio init creado por el **Ubuntu-team** que mejora en gran medida el funcionamiento del init de System V, aportando entre otras cosas: Arranque de servicios en paralelo, guiado por eventos, servicios que se rearrancan automáticamente si mueren, etc.

Upstart se utiliza en las distribuciones Linux de Ubuntu, Kubuntu y Linux Mint. A fecha de este documento todavía no se ha incorporado a Debian.

Como en el caso de systemd, no trataremos el funcionamiento de upstart, pero se incluyen algunos de sus ficheros de configuración más importantes:

- **Ficheros /etc/init/ttyX.conf**: que indican en qué nivel de ejecución ha de arrancarse cada consola virtual en modo texto.
- **Fichero /etc/init/rc-sysinit.conf**: En este fichero se establece el runlevel por defecto del sistema. Concretamente se define en la variable **DEFAULT_RUNLEVEL** de este fichero.
- **Fichero /etc/init/mdm.conf**: indica en qué runlevels ha de iniciarse el entorno gráfico.
- **Fichero /etc/init/networking.conf**: define en qué runlevels ha de estar presente el entorno de red.

2.2.1 Configuración de servicios.

Otro concepto fundamental y muy relacionado con el arranque del sistema, son los **procesos lanzados como servicio**.

Un **servicio** puede definirse como un proceso que se ejecuta en segundo plano y gestiona una determinada funcionalidad del sistema. Normalmente el proceso init lanza los servicios durante el arranque del sistema. Ejemplos de servicios son Samba, NFS, LDAP, el entorno gráfico, etc.

Es importante indicar que, tanto en el sistema **systemd** como en **upstart**, para cada nivel de ejecución hay un directorio llamado `/etc/rcX.d` donde tenemos enlaces simbólicos a todos los **servicios que han de iniciarse o detenerse en dicho nivel de ejecución**. Por ejemplo para el runlevel 2 tendremos el directorio `/etc/rc2.d`

Si el nombre del enlace comienza por una '**S**' es que hay que iniciar ese servicio, y si comienza por una '**K**' es que hay que detenerlo. Además cada enlace tiene en su nombre un número que indica el orden de ejecución de los programas del directorio.

Por tanto, si queremos que en un determinado nivel de ejecución se añada un servicio, solo hemos de crear un enlace soft al ejecutable de este fichero, en el directorio del runlevel. Por ejemplo, si queremos que el programa `miservicio` se ejecute al arrancar en el runlevel 2, podríamos ejecutar el siguiente comando:

```
# ln -s /etc/init.d/miservicio.sh /etc/rc2.d/S99miservicio
```

Hay que destacar que Debian obliga a que los servicios estén ubicados en el directorio `/etc/init.d`

Por otra parte, si queremos **arrancar o detener de forma manual un servicio** los comandos a ejecutar son los siguientes:

```
# /etc/init.d/miservicio start → Este comando arranca el servicio indicado.  
# /etc/init.d/miservicio stop → Este comando para el servicio indicado.  
# /etc/init.d/miservicio restart → Esto reinicia un servicio ya arrancado.
```

También podemos usar el comando **service**, con idéntico resultado:

```
# service miservicio start  
# service miservicio stop  
# service miservicio restart
```

Sistemas Operativos en Red

UT5 - Monitorización del sistema Linux y gestión remota

Y si queremos **ver los servicios parados y arrancados** podemos ejecutar el comando:

service --status-all

```
root@debian:/etc/init.d# service --status-all
[ - ] alsa-utils
[ + ] anacron
[ + ] apache-htcacheclean
[ + ] apache2
[ + ] avahi-daemon
[ - ] bluetooth
[ - ] console-setup.sh
[ + ] cron
[ + ] cups
[ + ] cups-browsed
[ + ] dbus
[ + ] exim4
[ + ] gdm3
[ - ] hwclock.sh
[ - ] keyboard-setup.sh
[ + ] kmod
[ + ] minissdpd
[ + ] network-manager
[ - ] networking
[ - ] nfs-common
```

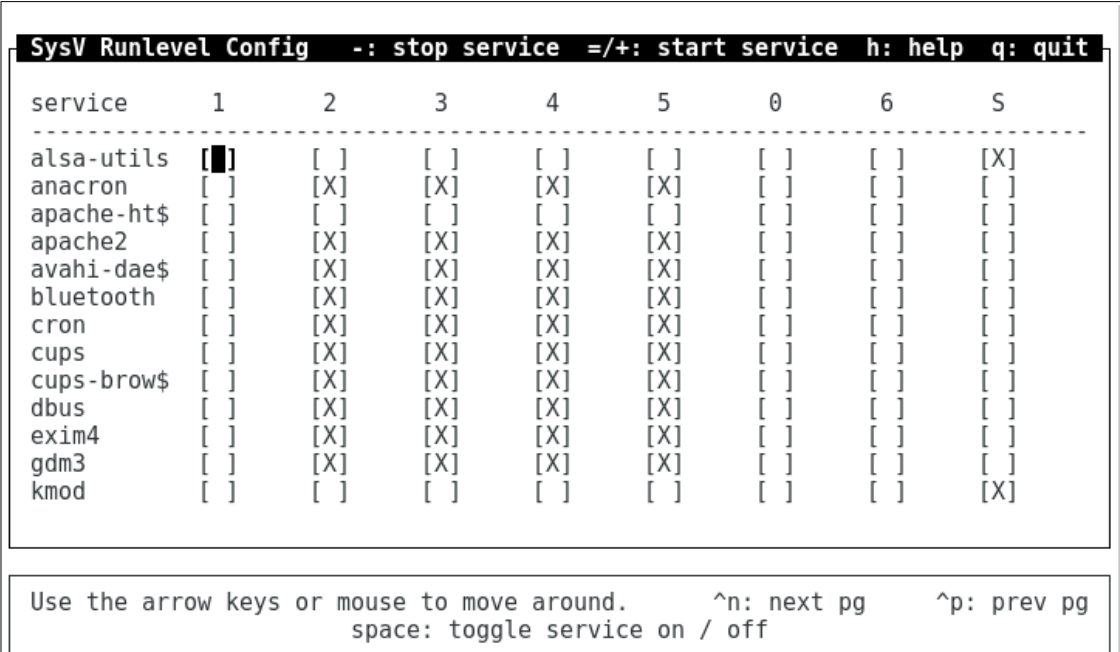
Por último añadir que para **facilitar la configuración de los servicios de arranque**, existen varias aplicaciones como **sysv-rc-conf**.

Para instalar sysv-rc-conf, podemos utilizar el comando:

apt-get install sysv-rc-conf

Una vez instalado el paquete lo lanzamos con el comando:

sysv-rc-conf



SysV Runlevel Config --: stop service =/+: start service h: help q: quit

service	1	2	3	4	5	0	6	S
alsa-utils	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
anacron	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
apache-ht\$	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
apache2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
avahi-dae\$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
bluetooth	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
cron	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
cups	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
cups-brow\$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
dbus	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
exim4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
gdm3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
kmod	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Use the arrow keys or mouse to move around. ^n: next pg ^p: prev pg
space: toggle service on / off

Como podemos ver, esta aplicación nos permite ver y modificar en qué runlevels se inicia cada uno de los principales servicios del sistema.

2.3. Análisis de los sucesos del sistema.

Siempre que se produce algún error o mal funcionamiento en el sistema, es conveniente analizar que situaciones se han producido en este. Para ello, Linux registra (por medio del demonio **rsyslogd**) los principales eventos que ocurren en el sistema, dentro de una serie de ficheros del directorio **/var/log**.

Algunos de los principales ficheros son los siguientes:

- **auth.log** → para eventos de autenticación de usuarios.
- **cron** → con mensajes acerca de los servicios de programación de tareas, cron y atd.
- **daemon.log** → Mensajes sobre procesos sin clasificación especial (DNS, NTP, etc.).
- **kern.log** → Guarda los eventos generados en la carga del kernel o núcleo del sistema operativo. Un ejemplo sería un mensaje que registre la carga de un controlador.
- **syslog y messages.log** → Almacenan eventos generales del sistema.
- **user.log** → Mensajes genéricos de usuario.
- **Xorg.log** → Con mensajes sobre el entorno gráfico.

Además, Cada mensaje tiene asociado también un nivel de prioridad. Aquí está la lista en orden decreciente:

Nivel	Descripción
EMERG	Emergencia. Fallo grave de integridad del sistema
ALERT	Fallo peligroso. Debe solucionarlo de inmediato.
CRIT	Error crítico del sistema
ERR	Error
WARN	Advertencia
NOTICE	Las condiciones son normales pero el mensaje es importante
INFO	Mensaje informativo
DEBUG	Mensaje de depuración.

Sistemas Operativos en Red
UT5 - Monitorización del sistema Linux y gestión remota

En el caso tener un sistema operativo Debian 9 en modo gráfico, podemos ver los registros del sistema desde la aplicación **Registros**.

2.4. Desinstalación de servicios no necesarios.

Como hemos visto anteriormente, según el runlevel con el que estemos trabajando, se inician determinados servicios.

Si queremos desactivar algún servicio solo hemos de borrar el enlace existente en el directorio del runlevel (/etc/rc2.d si estamos en el runlevel 2). También podemos modificarlo cambiando la S inicial de su nombre por una K.

Por otra parte, también podemos usar el comando **sysv-rc-conf** como se explicó en el apartado 2.2.1 de este documento.

2.5. Eliminación de paquetes no utilizados.

Cuando instalamos paquetes, es posible que con el tiempo algunos de ellos se vuelvan obsoletos y dejen de utilizarse.

Si queremos limpiar esos paquetes del sistema, hemos de ejecutar el comando:

```
# apt-get autoremove
```

2.6. Programa localepurge.

Localepurge es una herramienta que permite eliminar archivos de idiomas, optimizando de esta manera el espacio ocupado por nuestro sistema.

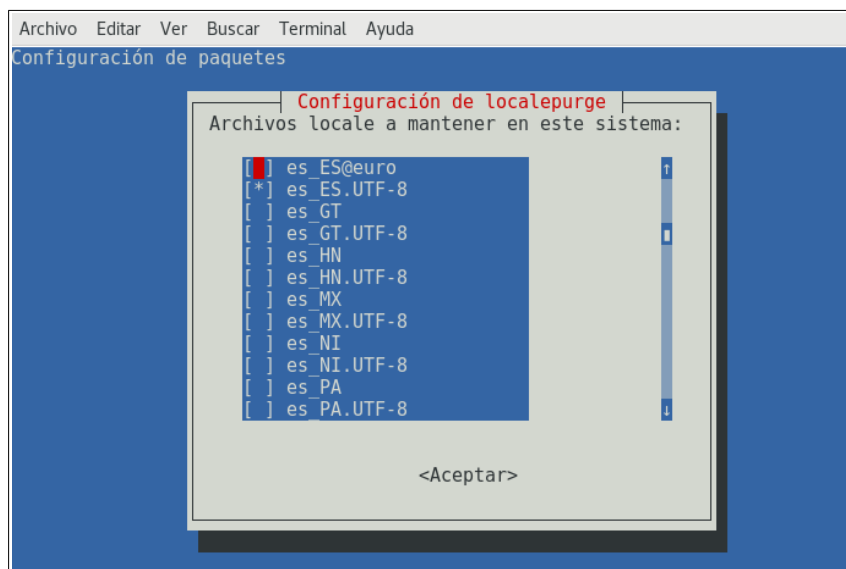
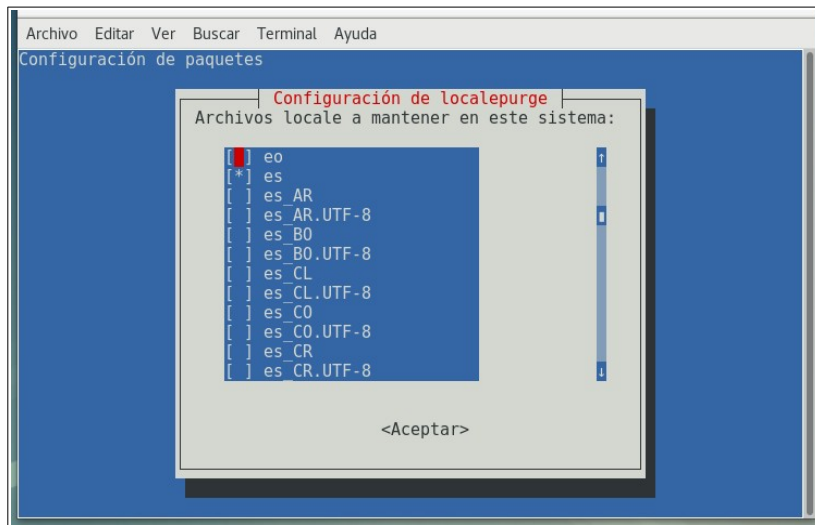
Para instalar ejecutaremos el comando:

```
# apt-get install localepurge
```

Una vez instalado, si lo ejecutamos, aparecerá una ventana con todos los posibles idiomas instalados. Aquí **hemos de seleccionar SOLO EL IDIOMA QUE NO QUERAMOS BORRAR.**

Sistemas Operativos en Red

UT5 - Monitorización del sistema Linux y gestión remota



2.7. Programa preload.

Preload es un demonio que se encarga de analizar el uso de los programas de usuario y cargar en memoria partes de las aplicaciones más utilizadas, mejorando por tanto, la carga de dichos procesos.

Para instalarlo simplemente debemos ejecutar:

sudo apt-get install preload

Esto descarga al programa y lo instala como servicio del sistema, de forma que siempre

que arranquemos el equipo estará activo.

2.8. Programa Bleachbit.

Bleachbit es una aplicación gráfica que permite eliminar de forma segura ficheros no innecesarios del sistema, liberando de esta forma espacio en los discos duros.

Entre la información que es capaz de limpiar, tenemos:

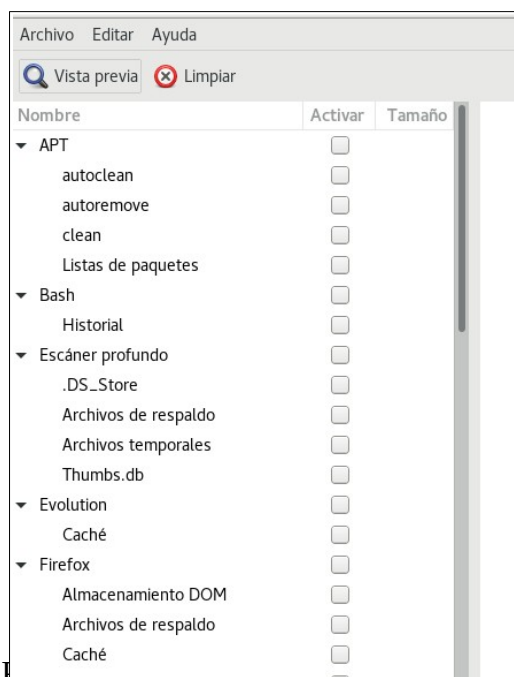
- Paquetes obsoletos.
- Ficheros de log antiguos.
- Cache, cookies e historial de navegadores
- Papelera de usuarios.
- Información desfasada de aplicaciones (VLC, thuderbird,etc)

Para instalar este programa ejecutaremos el comando:

```
# apt-get install bleachbit
```

Hay que indicar que podemos lanzar bleachbit en modo usuario común o administrador. Lógicamente la diferencia está en que en modo administrador tendremos acceso a todos los directorios del sistema.

Cuando ejecutamos el programa se muestr la siguiente pantalla:



Su uso es muy sencillo, solo hemos de seleccionar los elementos a chequear y pulsar el botón "**Limpiar**".

3. Administración remota de sistemas Linux.

En la mayoría de las empresas u organizaciones, el sistema informático está constituido por uno o varios servidores, los cuales pueden estar en la misma ubicación física o no. Por ejemplo, podríamos tener un servidor por planta y uno general, o puede darse el caso de que los equipos de los usuarios se conecten a un servidor que se encuentre en otra planta o edificio.

En estos casos, sería una pérdida de tiempo que el administrador del sistema tuviera que desplazarse entre las distintas ubicaciones para gestionar el funcionamiento de los servidores. Para evitar esto (o al menos minimizarlo), existen diversos protocolos que nos permiten abrir sesiones en un equipo desde otro conectado a la red.

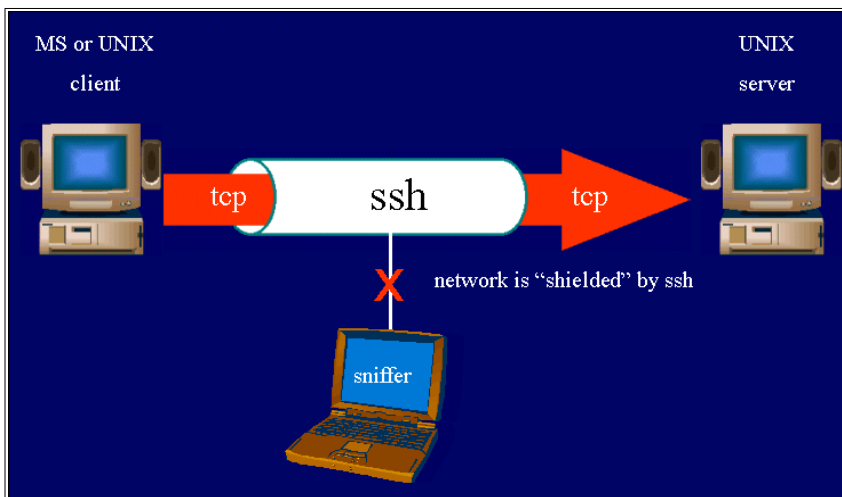
En este punto trataremos algunos de los principales sistemas para administrar un servidor Linux de forma remota.

3.1. Servicio SSH.

SSH (Secure Shell, en español: intérprete de órdenes seguro) es el nombre de un protocolo que sirve para acceder a servidores privados a través de una puerta trasera (también llamada *backend*).

SSH es un protocolo que puede iniciar sesiones en modo texto, dentro de servidores Linux, de forma remota.

Por otra parte, **SSH usa técnicas de cifrado** que hacen que la información que viaja por el medio de comunicación vaya de manera no legible, evitando que terceras personas puedan descubrir el usuario y contraseña de la conexión ni lo que se escribe durante toda la sesión.



Para instalar SSH **en el servidor** hemos de ejecutar el comando:

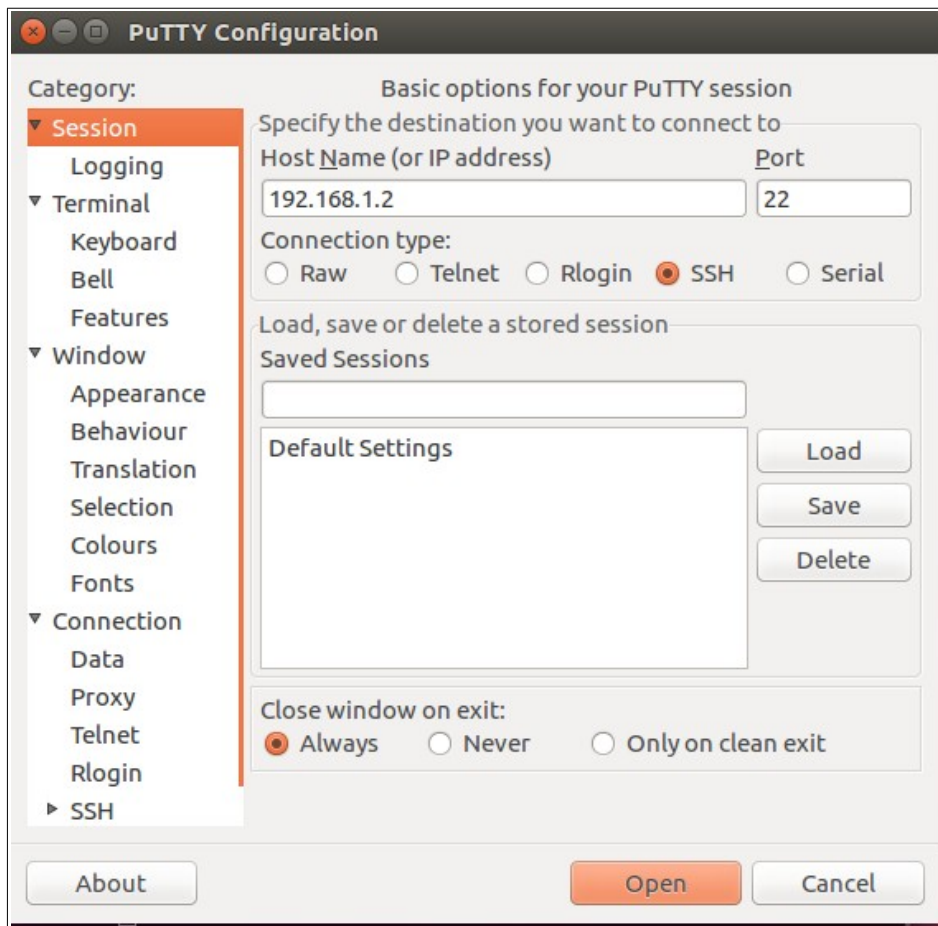
```
# apt-get install ssh
```

Seguidamente, instalaremos **en el cliente** un programa que permita acceder al servidor con SSH. Se pueden usar varios, uno de ellos es el **PuTTY**.

- Si estamos en un **cliente Windows**, descargaremos PuTTY desde Internet (<http://www.putty.org/>).
- Si estamos en un **cliente Linux**, ejecutaremos como administrador:

```
# apt-get install putty
```

Una vez hecho esto sólo hay que ejecutar en el cliente el programa y veremos la siguiente pantalla.



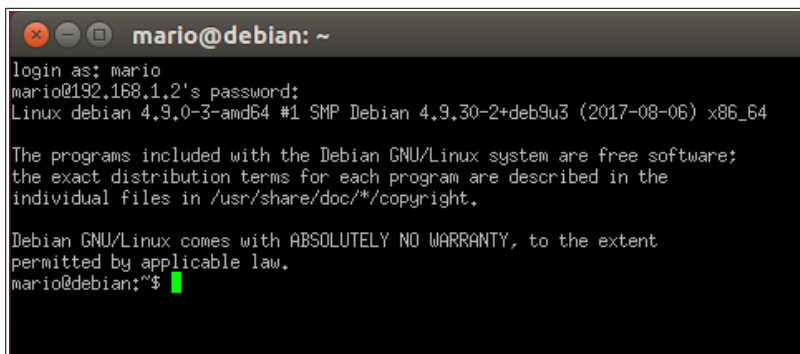
Como podemos observar PUTTY nos permite conectarnos con un equipo por medio de diferentes protocolos (Telnet, Rlogin, SSH, etc).

En nuestro caso introduciremos la dirección IP o el hostname del servidor y seleccionaremos SSH.

Sistemas Operativos en Red

UT5 - Monitorización del sistema Linux y gestión remota

Seguidamente se abrirá una ventana modo texto que nos pedirá que nos validemos con un usuario Linux existente en el servidor y tendremos acceso al servidor remoto.

A terminal window titled 'mario@debian: ~' with standard window controls. The text inside shows a login process: 'login as: mario', 'mario@192.168.1.2's password:', and system information 'Linux debian 4.9.0-3-amd64 #1 SMP Debian 4.9.30-2+deb9u3 (2017-08-06) x86_64'. It also displays the Debian GNU/Linux license notice. The prompt 'mario@debian:~\$' is followed by a green cursor.

```
mario@debian: ~
login as: mario
mario@192.168.1.2's password:
Linux debian 4.9.0-3-amd64 #1 SMP Debian 4.9.30-2+deb9u3 (2017-08-06) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
mario@debian:~$
```

3.2. Servicio VNC.

El servicio VNC permite el acceso a un equipo existente en la misma red local, ya sea Windows o Linux por medio de un terminal en modo gráfico.

Existen distintas implementaciones de VNC (x11vnc, vnc4server, tightvncserver, etc). En este capítulo explicaremos el uso de **x11vnc**, debido a que hoy por hoy es el sistema que ofrece un mayor rendimiento y seguridad al proporcionar una conexión gráfica sobre un canal cifrado SSH.



Entrando ya en el proceso, para instalar y configurar VNC seguiremos los pasos siguientes:

- 1. Instalación del servicio VNC en nuestro servidor.**

Para ello instalamos los paquetes necesarios con el comando:

```
# apt-get install x11vnc
```

- 2. Seguidamente hemos de definir la contraseña de acceso desde los equipos cliente.**

```
# x11vnc -storepasswd
```

- 3. Preparar un script para arrancar el servicio.**

Primero de todo crearemos un script con la orden:

```
nano /etc/init.d/vnc.sh
```

```
#!/bin/bash  
x11vnc -forever -shared -ncache -o $HOME/x11vnc.log &
```

Seguidamente le damos permisos de ejecución al fichero.

```
chmod a+x /etc/init.d/vnc.sh
```


Por último, configuraremos el sistema para que nuestro script arranque como servicio:

In -s /etc/init.d/vnc.sh /etc/rc5.d/S99vnc

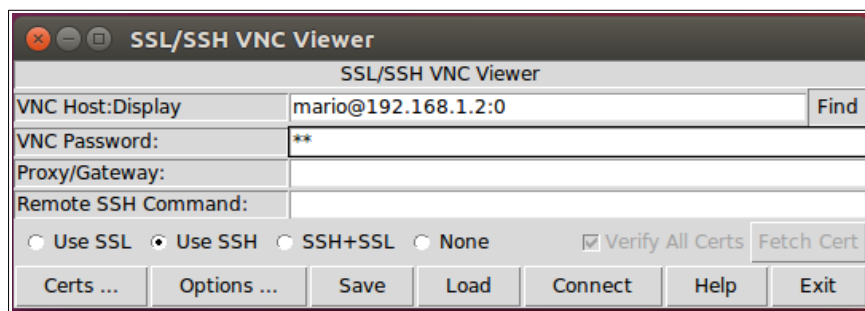
4. Por último **para acceder desde el equipo cliente**, podemos utilizar alguna aplicación específica, por ejemplo en Linux tenemos el programa **ssvnc**.

Para instalarlo ejecutaremos el comando:

apt-get install ssvnc

Su uso, es muy sencillo, simplemente introducimos:

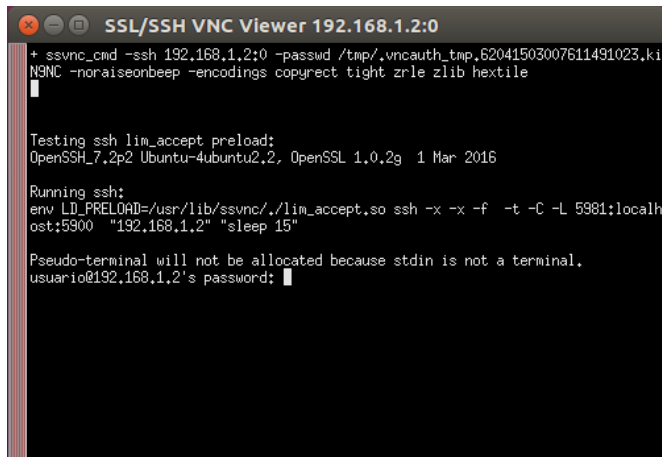
- **La IP del servidor y el número de display** que usa x11vnc (normalmente usaremos el 0), por ejemplo 192.168.1.2:0.
- **La contraseña para acceder a VNC** que introducimos en el paso 2.
- Indicamos también que usaremos una **conexión de tipo SSH**.
- Pulsamos al botón **“Connect”**.



El sistema nos pedirá ahora la clave de un usuario local del servidor (en este caso el usuario “usuario”)

Sistemas Operativos en Red

UT5 - Monitorización del sistema Linux y gestión remota



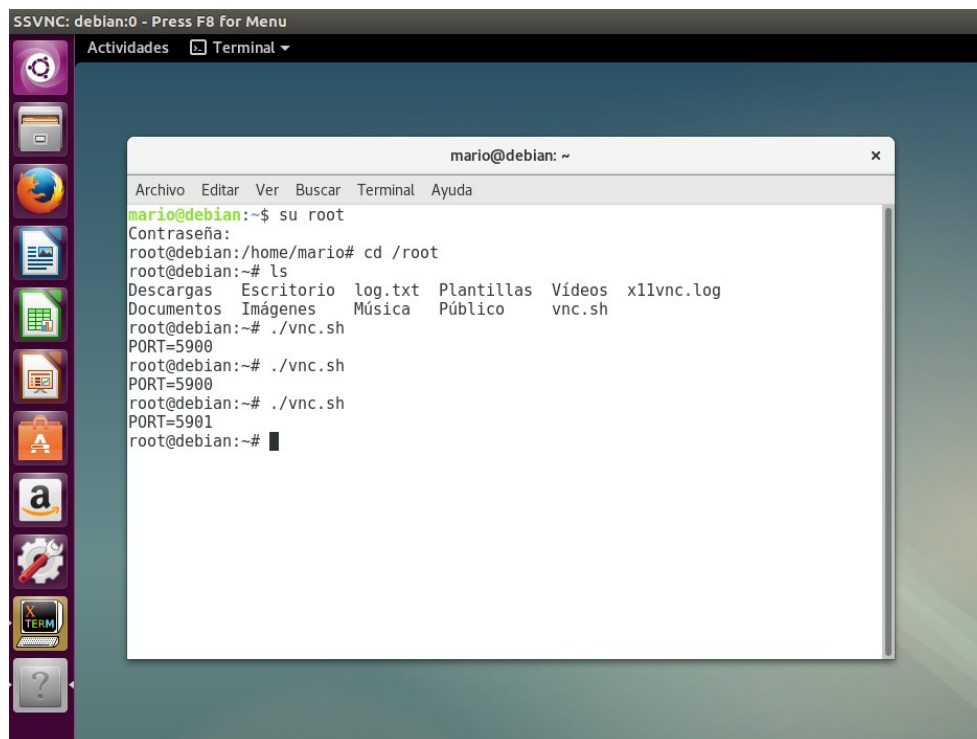
```
SSL/SSH VNC Viewer 192.168.1.2:0
+ ssvnc_cmd -ssh 192.168.1.2:0 -passwd /tmp/.vncauth_tmp.62041503007611491023.ki
N9NC -noraiseonbeep -encodings copyrect tight zrle zlib hextile

Testing ssh lim_accept preload:
OpenSSH_7.2p2 Ubuntu-4ubuntu2.2, OpenSSL 1.0.2g 1 Mar 2016

Running ssh:
env LD_PRELOAD=/usr/lib/ssh/lib/lim_accept.so ssh -x -x -f -t -C -L 5981:localh
ost:5900 "192.168.1.2" "sleep 15"

Pseudo-terminal will not be allocated because stdin is not a terminal.
usuario@192.168.1.2's password:
```

Con esto ya tendremos acceso al equipo servidor.



```
SSVNC: debian:0 - Press F8 for Menu
Actividades Terminal
mario@debian: ~
Archivo Editar Ver Buscar Terminal Ayuda
mario@debian:~$ su root
Contraseña:
root@debian:/home/mario# cd /root
root@debian:~# ls
Descargas Escritorio log.txt Plantillas Videos x11vnc.log
Documentos Imágenes Música Público vnc.sh
root@debian:~# ./vnc.sh
PORT=5900
root@debian:~# ./vnc.sh
PORT=5900
root@debian:~# ./vnc.sh
PORT=5901
root@debian:~#
```

Por último, solo decir que **en el caso de Windows** hay también múltiples aplicaciones como **PUTTY** o **UltraVNC**, que funcionan de manera parecida.