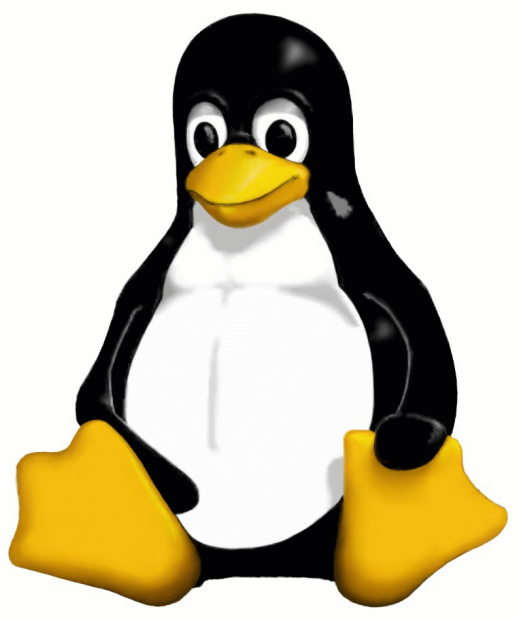
CURSO LINUX: Administración de Sistema y Servicios



Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Antes de nada...

GNU

GPL

Software Libre

Linux

Kernel

GNU/Linux

Debian

Distribución Linux

Paquete

KDE

(K)Ubuntu

Gnome

Emule

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Nuestra distro

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

¿Qué es Debian?

- Sistema operativo libre desarrollado por la comunidad.
- Conjunto de programas básicos y utilidades.
- Utiliza el núcleo Linux (el corazón del sistema operativo).
- Herramientas básicas Proyecto GNU -> **GNU/Linux**.

Debian Sarge

- Versión estable de Debian.
- Probada y testeada a conciencia.
- Óptima para un servidor en producción.

Debian Testing e Inestable

- Versiones de desarrollo y experimentación.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Contrato Social de Debian:

- «Contrato social» con la comunidad de software libre:
  1) Debian permanecerá 100% libre.
  2) Contribuiremos a la comunidad de software libre.
  3) No ocultaremos los problemas.
  4) Nuestra prioridad son nuestros usuarios y el software libre.
  5) Rama de paquetes “contrib” y “non-free”.
- http://www.debian.org/social_contract

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Debian

Instalación para todas las arquitecturas:

- Debian soporta múltiples arquitecturas.
- http://www.us.debian.org/releases/stable/installmanual

Instalación para i386:

- http://www.us.debian.org/releases/stable/i386

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Instalación Debian Sarge

- Desde CD **Net-Install**:
  - http://www.us.debian.org/CD/netinst/
- Durante la instalación:
  - Kernel 2.6 (opción linux26).
  - Red por DHCP.
  - Repositorios externos por http.
  - Particionamiento (dependiendo del disco duro):
    - / -> Ext3 4GB
    - /var -> Ext3 6GB
    - /home -> Ext3 40GB
    - swap -> 1GB

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Introducción a GNU/Linux

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

GNU/Linux

- Herramientas GNU + Linux (núcleo).
- Compatible con UNIX (GNU's Not UNIX).
- Sistema multiusuario y multitarea.

Bash

- Herramienta GNU.
- Shell, intérprete de comandos.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Shells

- Existen muchas shells: `sh,csh,ksh,bash...`
- Entorno de trabajo:
  - Case sensitive: sensible a mayúsculas (ECHO != echo).
  - Sintáxis: comando arg1 arg2... argn
  - Si un programa no está en el PATH: ./programa
  - Prompt:
    - $: usuario normal
    - #: usuario administrador (root)

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Comandos

- Manejo del sistema de ficheros: `ls, cd, cp, mv, rm, mkdir, rmdir...`
- Información sobre ficheros: `cat, more, less, file...`
- Busquedas: `find, whereis, locate...`
- Filtros: `grep, sed, cut, tr...`
- Usuarios y grupos: `id, whoami, su, sudo...`
- Permisos: `chmod, chown, chgrp...`
- Otros: `date, tar, gzip, echo...`

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Tuberías

- Un proceso en un sistema UNIX-like tiene inicialmente abiertos 3 canales:
  - 0: STDIN o entrada estándar
  - 1: STDOUT o salida estándar
  - 2: STDERR o salida de error
- Imaginémonos una refinería:
  - Metes crudo por 0 (STDIN), consigues gasolina por 1 (STDOUT) y bastantes residuos por el "desagüe" 2 (STDERR).

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Tuberías

- Redirigiendo la salida de un comando:
  - > : redirigir STDOUT a un fichero:

```
ls > listado.txt
```

  - >>: redirigir STDOUT al final de un fichero (añadir):

```
ls >> listados.txt
```

  - 2>: redirigir STDERR a un fichero:

```
ls 2> errores.txt
```

  - 2>>: redirigir STDERR al final de un fichero:

```
ls 2>> errores.txt
```

  - 2>&1: redirigir STDOUT y STDERR a un fichero:

```
ls > salida 2>&1
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Tuberías

- Redirigiendo la entrada de un comando:
  - <: redirigir el contenido de un fichero a STDIN:

```
tr a A < fichero.txt
```

- | : es posible recoger la salida de un desagüe y conducirlo a la entrada de otro comando.

```
cat fichero.txt | tr a A
echo "Solo la X de este texto" | cut -d" " -f3
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Programación Básica en Shell (Bash)

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Shell Scripts

- Script = Guión
- Tareas repetitivas se pueden agrupar en un guión y ejecutarse automáticamente (Batch Processing).
  - Es sencillo ejecutar 4 comandos para crear un buzón de correo.
  - No lo es tanto para crear 20.000 buzones.
  - Es sencillo hacer un bucle que se repita 20.000 veces ;-)

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Nuestro primer shell script

- Usamos un editor y creamos el fichero hola.sh:

```
#!/bin/sh
echo hola
```

- Con #! en la primera línea indicamos quién debería interpretar el resto de comandos (/bin/sh).
- Posteriormente escribimos los comandos separados por saltos de línea.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Variables

- Una variable tiene un nombre y un valor, y sirve para dotar de dinamismo a nuestros scripts:

```
FECHA="15/07/2004"
echo "Hoy es $FECHA"
```

  - `FECHA` es el nombre de la variable.
  - `$FECHA` es su valor.
- Para asignar un valor, se utiliza "=". ¡¡¡SIN ESPACIOS!!!

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Variables de entorno

- Al arrancar una shell, ya hay muchas variables definidas, son las variables de entorno.
  - Podemos ver su valor con el comando “env”.
- Ámbito de una variable:
  - Si se define una variable en una shell, sólo tiene valor en esa shell, a no ser que se exporte a los programas “hijo”.
  - `export USUARIO="joaquin"`
  - Si desde esa shell lanzamos un script u otro programa, la variable `USUARIO` contendrá “joaquin”.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Variables: interactividad

- Es posible leer del usuario el valor de una variable, dotando a nuestros scripts de interactividad.
- `cat hola.sh`

```
#!/bin/sh
echo "Dime tu nombre:"
read NOMBRE
echo "Hola $NOMBRE, encantado de conocerte"
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Variables: argumentos

- Es posible pasar los parámetros o argumentos que queramos y utilizarlos dentro del script.
- `cat nombre.sh`

```
#!/bin/sh
echo "Nombre: $1"
echo "Primer Apellido: $2"
echo "Segundo Apellido: $3"
```

- `./nombre.sh Juan López Martínez`
- `./nombre.sh "Maria Dolores" Pradera Sánchez`

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Variables: argumentos

- `$1`, `$2`, `$3`... `${10}`, `${11}`: argumentos
- `$0` es el propio script.
  - `basename $0`: nombre del script.
  - `dirname $0`: ruta al nombre del script.
- `shift`: rota los argumentos hacia la izquierda
  - `$1` ahora vale lo que valía `$2`, `$2` lo que valía `$3`, etc.
  - `$0` no cambia.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Variables: argumentos especiales

- `$#`: número de argumentos que nos han pasado.
- `$*`: todos los argumentos. "$*" = "$1 $2 $3..."
- `$@`: todos los argumentos. "$@" = "$1" "$2" "$3"...
- `$_`: comando anteriormente ejecutado.
- `$$`: PID del propio proceso shell.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Variables: sustitución de comandos

- Es posible almacenar en una variable el resultado de la ejecución de un comando.
- Dos sintaxis:
  - Acentos graves: compatibilidad

```
LISTADO=`ls`
```

  - Con $(): anidable

```
LISTADO=$(ls)
LISTADO=$(ls $(cat directorios.txt))
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

`expr`: Permite realizar operaciones aritméticas.

- Sintaxis: expr ARG1 OP ARG2

```
$ SUMA=`expr 7 + 5`          (ojo espacios!)
$ echo $SUMA
12

$ expr 7 \> 5                    (ojo escapar operadores!)
$ expr \( 7 + 5 \) \* 2
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- Condiciones: `test` ó `[ ]`
  - `test "$NOMBRE" == "Juan"` (==, !=, >, <, >=, <=)
  - `test $DINERO -eq 1000` (-eq, -ne, -gt, -lt, -ge, -le)
  - `test -f /etc/passwd` (-f, -d, -l, -r, -w, -x)
- Modifican el valor de $?
  - cero = verdadero
  - no cero = falso (¡¡AL REVÉS QUE EN C!!)

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- `if`: alternativa simple. Sintaxis:

```
if condición_1
then
  comandos
elif condición_2
then
  comandos
else
  comandos
fi
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- `if`. Ejemplo:

```
if test "$NOMBRE" == "Juan"
then
  echo "Hola Juanin, ¿qué tal?"
elif test "$NOMBRE" == "Pedro"
then
  echo "Pedreteee, ¡cuánto tiempo!"
else
  echo "No te conozco"
fi
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- `case`: cómodo para evitar alternativas anidadas. Sintaxis:

```
case $VARIABLE in
  "VALOR1") comandos
    ;;
  "VALOR2") comandos
    ;;
  *) comandos;
esac
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- `case`. Ejemplo:

```
case $NOMBRE in
  “Juan”) echo “Hola Juanin, ¿qué tal?”
    ;;
  “Pedro”) “Pedreteee, ¡cuánto tiempo!”
    ;;
  *) echo “no te conozco”;
esac
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- `while`. Ejecución de 0 a N veces. Sintaxis:

```
while condición
do
  comandos
done
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- `while`. Ejemplo:

```
N=1
while [ $N -lt 100 ]
do
  echo “Repito esta frase, ya voy $N veces”
  N=$(expr $N + 1)
  sleep 1 # Esperamos 1 segundo
done
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- `until`. Ejecución de 0 a N veces. Idéntico a `while` con la condición negada. Sintaxis:

```
until comando
do
  comandos
done
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- `until`. Ejecución de 0 a N veces. Idéntico a while con la condición negada. Sintaxis:

```
N=1
until [ $N -ge 100 ]
do
  echo “Repito esta frase, ya voy $N veces”
  N=$(expr $N + 1)
done
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- `for`: ejecución repetitiva asignando a una variable de control valores de una lista. Sintaxis:

```
for VARIABLE in LISTA
do
  comandos
done
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- `for`. Ejemplo:

```
for N in “Sopa” “Carne” “Pan de ajo”
do
  echo “Hoy comemos $N”
done
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- `for`: la LISTA define la separación de cada elemento por el valor de la variable `IFS` (que por defecto vale " `\t\n`"). Ejemplo:

```
IFS=":"
echo "Directorios en el PATH..."
for DIR in $PATH
do
  echo $DIR
done
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- `for`. Ejemplos numéricos:

```
for N in 1 2 3 4 5 6 7 8 9 10
do
  echo "N ahora vale $N"
done
for N in $(seq 10)
do
  echo "N ahora vale $N"
done
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- `select`: muestra las opciones especificadas en LISTA y asigna a VARIABLE la opción escogida. Sintaxis:

```
select VARIABLE in LISTA
do
  comandos
done
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Control del flujo de ejecución

- `select`: Ejemplo:

```
select OPCION in "Doner Kebab" "Pizza"
do
  case $OPCION in
    "Doner Kebab") echo "Mmmm..."
      break;;
    "Pizza") echo "Slurppp!"
      break;;
    *) echo "No sé qué es eso"
  esac
done
```

IMPORTANTE: sin el break el select seguiría ejecutándose indefinidamente.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

`function`

- Podemos modularizar los scripts agrupando tareas en funciones.
- Es necesario que una función esté definida ANTES de que sea llamada.
- Dentro de una función, $1, $2, $3, etc. serán los parámetros pasados a la función, no al script en sí.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

`function`. Ejemplo:

```
#!/bin/sh
function suma
{
  echo $(expr $1 + $2)
}
suma 4 6
suma 3 234
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

source,.

- Con source o con "." podemos incluir el código de otro script en el nuestro:

```
#!/bin/sh
source funciones.sh # ahí se define suma
suma 1 3
suma 12 12312
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Shell Scripts

- Es posible mejorar el aspecto de nuestros menús y opciones más allá de “read” y “select” usando herramientas como “dialog”, “whiptail”, “Xdialog”, “gdialog” o “kdialog”.
- Son bastante similares y sencillas de utilizar.
- Son capaces de generar cajas de texto, diálogos de petición de texto o contraseñas, menús, barras de progreso, etc.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

dialog

- dialogos desde shell scripts en ncurses.
- MsgBox:

```
dialog --title "Alerta"  --backtitle  "Cursillo de Bash
Shell" --msgbox "Este es un mensaje con dialog" 8 50
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

dialog

- YesNo:

```
dialog --title "Pregunta" --backtitle "Curso de Bash
Shell" --yesno "\n¿Estas aprendiendo algo?" 7 60

RESPUESTA=$?
case $RESPUESTA in
  0) echo "Bien!";;
  1) echo "Mal!";;
  255) echo "Salir [ESC]";;
esac
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

dialog

- InputBox y PasswordBox:

```
dialog --title "Creacion de usuarios" --inputbox "Nombre
de usuario" 0 0 2> /tmp/dialog.$$
USUARIO=$(cat /tmp/dialog.$$)

dialog -title "Creacion de usuarios" --passwordbox
"Clave" 0 0 2> /tmp/dialog.$$
CONTRASENYA=$(cat /tmp/dialog.$$)

rm /tmp/dialog.$$
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

dialog

- Menu:

```
dialog --title "Creacion de usuarios" --menu "Grupos" 0 0
3 0 root 100 users 5 audio 2>
  /tmp/dialog.$$

GRUPOS=$(cat /tmp/dialog.$$)

rm /tmp/dialog.$$
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

dialog

- Barras de progreso (gauge):

```
{
  echo 10; sleep 1
  echo 40; sleep 2
  echo 95; sleep 1
  echo 100
} | dialog --title "Creacion de usuarios" --gauge
“Creando...” 0 0 0
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Shell Scripts

Todo lo anterior se puede hacer con el resto, con cambios mínimos en la sintaxis:

- whiptail: ncurses, aspecto mejorado
- Xdialog: widgets X simples
- gdialog: GNOME
- kdialog: KDE

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Tareas periódicas

cron

- El demonio `cron` permite la ejecución programada de procesos, scripts o simples comandos.
- `Cron` mira cada minuto si tiene que lanzar algún proceso y en caso afirmativo lo lanza. No se puede planificar una ejecución de forma más precisa.
- Cada usuario tiene su tabla de planificación (`crontab`) aunque también existen una tabla de planificación global del sistema (`/etc/crontab`) y una serie de directorios especiales.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Tareas periódicas

cron

- Ficheros de información de `cron`. Los ficheros de `cron` tienen una sintaxis especial y son los siguientes:
  - `/etc/crontab`: Fichero de cron del sistema.
  - `/etc/cron.d/`: Directorio para que paquetes puedan meter ficheros de cron.
  - fichero de cron de cada usuario:
    - No se ven (`/var/spool/cron...`).
    - No es necesario indicar el usuario que sobre el que se ejecutará, ya que pertenecen a un usuario concreto.
    - Se edita con `crontab -e` y se muestra con `crontab -l`.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Tareas periódicas

cron

- La sintaxis típica de los ficheros de cron consiste en 6 campos:

```
* * * * * [user] comando parametros

minuto: 0-59. Número, rango, intervalo o lista (separada
por comas).
hora: 0-23.
día del mes: 1-31.
mes: 1-12 o nombres.
día de la semana:  0-7 o nombres.
usuario: sólo en ficheros comentados anteriormente.
comando: el resto de la línea es el comando a ejecutar.
```

- Nota: También puede haber líneas al principio con declaración de variables y comentarios (#).
- Más info (`man 5 crontab`).

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Tareas periódicas

cron

- Ejemplo de cron de un usuario (root):

```
PATH=$PATH:/usr/bin:/usr/local/bin

# los viernes cada 2 horas...
* */2 * * 5 cd /root/scripts; ./generaEstadisticas.sh

# cada día por la noche
20 3   * * * /root/scripts/backupDiario.sh

# rotación de backups cada mes
0 1 1 * * cd /root/scripts; ./rotaBackups.sh
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Tareas periódicas

cron

- Creación o edición del fichero de cron del usuario actual:

```
$ EDITOR=vi
$ export EDITOR
$ crontab -e
(se edita...)
:wq (se guarda y se sale)
```

- Mostrar el contenido del cron de un usuario:

```
$ crontab [-u user] -l
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Tareas periódicas

cron

- Directorios especiales:

```
/etc/cron.daily/
/etc/cron.hourly/
/etc/cron.monthly/
/etc/cron.weekly/
```

- Los ficheros ejecutables que metamos en esos directorios se ejecutarán en su momento por el usuario root. NO DEBEN SER FICHEROS DE CRON, sino scripts o programas.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Servicios y Niveles de Ejecución

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Proceso de arranque

- La BIOS busca un dispositivo de inicio (disco duro, CD-ROM,..) y pasa el control al MBR (512 bytes).
- Se carga el gestor de arranque (instalado en MBR).
- Se carga el kernel.
- Se monta el sistema de ficheros raiz (`/`).
- Se inicia el `init` (el abuelo de todos los procesos).
- Se lee el archivo `/etc/inittab`.
- Se ejecutan los scripts indicados por el nivel de ejecución de arranque.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Gestores de arranque

LiLo

- LILO (Linux Loader) es un gestor de arranque capaz de arrancar diferentes sistemas operativos en diferentes particiones y discos duros.
- Normalmente se instala en el MBR (master boot record) del disco duro principal.
- El archivo de configuración en Debian GNU/Linux es: `/etc/lilo.conf`
- Siempre que se realice un cambio en la configuración hay que ejecutar '`lilo`' para que se escriba el sector de arranque de nuevo.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

LiLo

- Datos de interes del archivo `lilo.conf`:

```
boot=/dev/hda       # Donde se instala el lilo
install=menu        # Tipo de instalación (interfaz)
delay=20            # Tiempo de espera
default=Linux-2.6.8 # Label de arranque por defecto

image=/boot/bzImage-2.6.8.1   # Imagen del kernel
     label=Linux-2.6.8     # Etiqueta
     read-only
     #vga=0x317            # (1024x768)

     append="video=i810fb:xres:1024,yres:768,bpp:8,
      hsync1:30,hsync2:55,vsync1:50,vsync2:85"

other=/dev/hda2               # Otros S.O.
     label=Windows
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Gestores de arranque

LiLo

- LILO con menú gráfico:

```
install=/boot/boot-bmp.b
bitmap=/boot/debian-bootscreen-woody.bmp # debe existir
bmp-colors=1,,0,2,,0
bmp-table=120p,173p,1,15,17
bmp-timer=254p,432p,1,0,0
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Grub

- Grub (Grand Unified Bootloader) es un gestor de arranque capaz de arrancar diferentes sistemas operativos en diferentes particiones y discos duros.
- Normalmente se instala en el MBR (master boot record) del disco duro principal.
- El archivo de configuración en Debian GNU/Linux es: `/boot/grub/menu.lst`
- No es necesario ejecutar ningún comando para que se tengan en cuenta los cambios realizados.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Gestores de arranque

Grub

- Grub no es capaz de distinguir entre dispositivos IDE, SCSI u otros.
- Sintaxis:
  - Dispositivos
    - `(dispositivo[particion][,letra_particion])`

```
(hd0)              # disco maestro IDE primario
(fd0)              # disquetera
(hd0,1)            # partición 2 del maestro IDE 0
```

  - Ficheros
    - Es necesario indicar la ruta completa

```
(hd0,0)/boot/grub/menu.lst
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Grub

- Datos de interes del archivo `menu.lst`:

```
default          0
timeout          3

title            Ubuntu, kernel 2.6.12-10-386
root             (hd0,1)
kernel           /boot/vmlinuz-2.6.12-10-386
root=/dev/hda2 ro quiet splash
initrd           /boot/initrd.img-2.6.12-10-386
savedefault
boot

title            Microsoft Windows
root             (hd0,2)
savedefault
makeactive
chainloader      +1
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Niveles de Ejecución

En los sistemas Unix hay 7 (0...6) niveles de ejecución

- nivel 0: estado de parada (`halt`)
- nivel 1: monousuario
- nivel 2, 3 y 5: multiusuario
- nivel 4: no tiene un uso específico
- nivel 6: estado de reinicio (`reboot`)

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Niveles de Ejecución

Init

- Es el primer proceso que se crea (PID=1).
- Se configura mediante `/etc/inittab`.
  - Sintaxis

```
id:nivel:accion:proceso
```

    - `id`: nombre de la línea
    - `nivel`: nivel o niveles en los que la línea debe procesarse
    - `accion`

```
wait (espera), once (solo 1 vez), respawn (rearranca
una vez finalizado), off (ignora la línea)
```

    - `proceso`: path del proceso a ejecutar

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Init

- Valores típicos del archivo `/etc/inittab`:

```
id:2:initdefault

l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3

1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2

z6:6:respawn:/sbin/sulogin

# CTRL-ALT-DEL!
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Arranque del sistema

- Cuando la máquina se inicia en un nivel de ejecución N se ejecutan todos los scripts que empiecen por 'S' o 'K' contenidos en `/etc/rcN.d/` con las siguientes particularidades.
  - Si el nombre del script comienza por S se le pasa como parámetro '`start`'.
  - Si el nombre del script comienza por K se le pasa como parámetro '`stop`'.
- Normalmente todos estos ficheros de `/etc/rcN.d` son enlaces a scripts localizados en `/etc/init.d/`

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Configurando servicios en arranque y parada

update-rc.d

- Herramienta para configurar automágicamente los enlaces a los scripts de init tipo System V que están en `/etc/rcN.d/[S|K]NNnombre` y que apuntan a los scripts `/etc/init.d/nombre`.
- Ejemplos

```
# update-rc.d 3ware defaults
# update-rc.d script start 90 1 2 3 4 5 . stop 20 0 6 .
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Gestión de Paquetes

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Paquetes .deb

DPKG, Sistema de gestión de paquetes de Debian

- Permite la instalación, borrado y mantenimiento de .debs
  - `dpkg -i paquete.deb`: instala un paquete.
  - `dpkg -r [--purge] paquete`: elimina un paquete
  - `dpkg -L paquete`: muestra el contenido completo de un paquete.
  - `dpkg -S file`: busca paquetes que contengan el fichero.
  - `dpkg -l`: muestra la lista completa de paquetes instalados en el sistema.
  - `dpkg -s paquete`: muestra información del estado de un paquete.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

APT, front-end avanzado para DPKG

- APT resuelve dependencias. Al instalar un paquete puede que éste dependa de otro u otros para su funcionamiento. APT detecta esta dependencia e instala los paquetes necesarios.
- Funcionamiento:
  - Se crea una base de datos local con la información de los paquetes instalables. Para crear esta base de datos hace falta un fichero con las fuentes (`/etc/apt/sources.list`) de donde bajarse la información.
  - Esta base de datos local hay que actualizarla periódicamente (`apt-get update`)

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

APT, front-end avanzado para DPKG

- Funcionamiento (cont.):
  - Cuando se solicita la instalación de un paquete, APT comprueba primero en el sistema que el paquete no esté ya instalado y posteriormente comprueba en la base de datos local si el paquete está disponible.
  - Si el paquete está disponible entonces se conecta a la fuente en cuestión para bajarse el paquete (archivo .deb). Estos paquetes bajados se guardan en `/var/cache/apt/archives`
  - Posteriormente se procede a la instalación y configuración automática del paquete. Si el paquete necesita datos de configuración nos los pedirá el APT.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Paquetes .deb

APT, front-end avanzado para DPKG

- Definición de fuentes para APT:
  - Se definen en el fichero `/etc/apt/sources.list`
  - Se pueden configurar con un asistente ejecutando `apt-setup`.
  - Para leer CDs con fuentes podemos utilizar también `apt-cdrom`.
- Ejemplo de sources.list básico:

```
deb http://ftp.fi.debian.org/debian stable main contrib non-free
deb http://ftp.se.debian.org/debian-non-US stable/non-US main contrib non-free
deb http://security.debian.org/ stable/updates main contrib non-free
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Paquetes .deb

`apt-cache`: realiza búsquedas sobre la base de datos local de paquetes.

- Útil para la obtención de información sobre software disponible.
  - `apt-cache search patron`: busca paquetes que cumplan un patrón.
  - `apt-cache show paquete`: muestra la información de un paquete
  - `apt-cache depends paquete`: muestra las dependencias del paquete.
  - `apt-cache rdepends paquete`: muestra las dependencias inversas del paquete.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Paquetes .deb

`apt-get`: interfaz para instalar y desinstalar paquetes así como para la generación y actualización de la base de datos local de paquetes.

- `apt-get update`: Actualiza la base de datos local de paquetes.
- `apt-get upgrade`: Actualiza todos los paquetes instalados que pueda (solo si hay versiones nuevas disponibles)
- `apt-get dist-upgrade`: Actualiza todos los paquetes que pueda incluso cuando la actualización implique la instalación de paquetes nuevos.
- `apt-get install paquete1 paquete2...`: Instala paquetes

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Paquetes .deb

apt-get

- `apt-get remove [--purge] paquete1 paquete2 ...` : Desinstala paquetes.
- `apt-get clean`: elimina archivos descargados en /var/cache/apt/archives (para liberar espacios, NO desinstala los paquetes).

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Networking

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Configuración de la red

- Las interfaces de red se configuran en el fichero '/etc/network/interfaces' (man interfaces).

```
aktor@irontec:~$ cat /etc/network/interfaces

auto eth0 eth1
iface eth0 inet static
        address 192.168.0.2
        netmask 255.255.255.0
        broadcast 192.168.0.255
        gateway 192.168.0.1
iface eth1 inet dhcp
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Configuración manual de la red

- `ifconfig`: configura interfaces de red. Asigna IP, máscara, gateway, etc.

```
$ ifconfig eth0 192.168.0.12 netmask 255.255.255.0 up
```

- `route`: añade rutas estáticas. Ejemplo típico:

```
$ route add default gw 192.168.0.1
$ route add -net 10.10.0.0 netmask 255.255.255.0 gw
192.168.0.100
$ route -n (muestra tabla de rutas)
```

- `/etc/resolv.conf`: indica los servidores DNS

```
$ cat /etc/resolv.conf

nameserver 195.235.113.3
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Configuración de la red

- IP alias: crear un interfaz de red sobre otro.
  - ➔ Una misma red física puede albergar distintas redes con distinto direccionamiento y rango.

```
$ ifconfig eth0:1 192.168.1.123 netmask 255.255.255.0
```

```
$ ifconfig eth0:1
eth0:1
      Link encap:Ethernet  HWaddr 00:14:85:E8:D3:AF
      inet addr:192.168.1.123  Bcast:192.168.1.255
      Mask:255.255.255.
      UP BROADCAST RUNNING MULTICAST  MTU:1500 Metric:1
      Interrupt:217 Base address:0xe800
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Configuración manual de la red

- `/etc/hosts`: realiza una resolución directa de nombre a IP sin realizar consulta DNS. Normalmente este fichero se mira antes de preguntar al servidor DNS (sólo para consulta de tipo A).
- Para asegurarnos que el sistema pregunte al DNS o al fichero '/etc/hosts' para resolver nombres de host hay que mirar el fichero '/etc/nsswitch.conf'

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Configuración de la red

Configuración automática de la red (DHCP)

- Configuración para el arranque: en /etc/network/interfaces definimos la interfaz como “inet dhcp”
- Configuración manual:
  - ➔ se utiliza un cliente dhcp como `dhclient` o `pump`

```
# dhclient eth0
# pump -i eth0
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Comandos básicos de administración de red

`ping`: manda un mensaje ICMP – echo request.

- Se utiliza normalmente para comprobar si un interfaz de red remoto está levantado.

```
$ ping 212.55.8.132
PING 212.55.8.132 (212.55.8.132) 56(84) bytes of data.
bytes from 212.55.8.132: icmp_seq=1 ttl=242 time=166 ms

¡hay conectividad!
```

- Nota: Si no hay ping puede que el tráfico ICMP esté filtrado por algún firewall.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Comandos básicos de administración de red

`telnet`: protocolo de terminal remoto

- Se utiliza para conectarse a una máquina remota.

```
$ telnet IP puerto (default 23)
```

- NO es un protocolo seguro (SSH sí).
- Se utiliza también para comprobar estado de servicios remotos. Ejemplo:

```
$ telnet www.euskalnet.net 80
Connected to eui3h.euskaltel.es.
Escape character is '^]'.

¡el servidor web funciona!
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Comandos básicos de administración de red

`netstat`: muestra conexiones de red

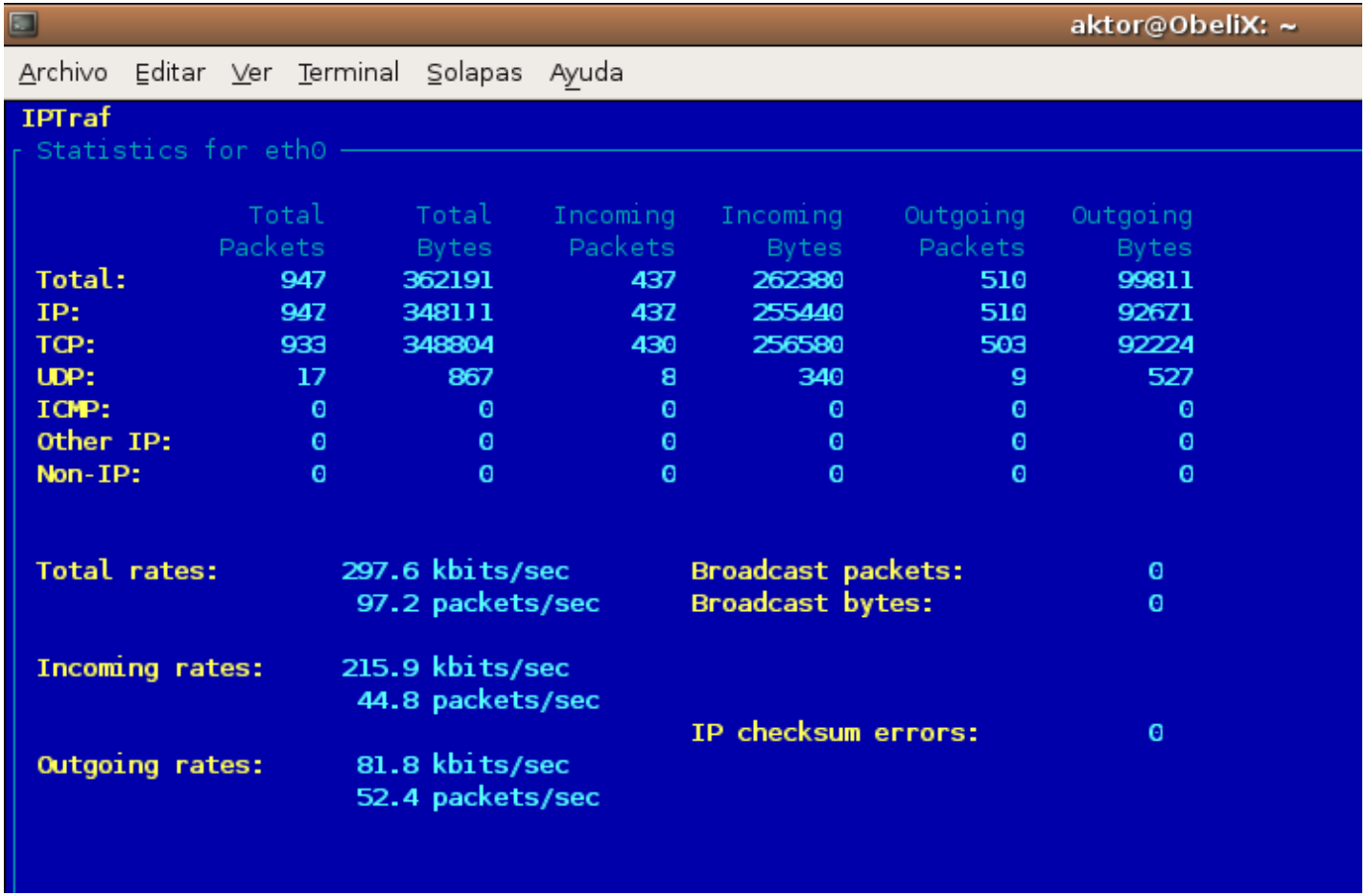
- Comando potente que permite mostrar casi toda la información de la configuración TCP/IP de la máquina (man netstat).
- Ejemplos:

```
netstat -atup # muestra tambien procesos
netstat -a # muestra todas las conex.
netstat -a | grep LISTEN
netstat -a | grep ESTABLISHED
netstat -nrv # muestra tabla de rutas
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Comandos avanzados administración de red

iptraf:



Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Comandos avanzados administración de red

`netcat (nc)`: "navaja suiza" de red

Capaz de funcionar como cliente, como servidor, etc.

```
telnet 127.0.0.1 80
     Trying 127.0.0.1...
     Connected to 127.0.0.1.
     Escape character is '^]'.
GET /
     <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2
     Final//EN">
     <html>
      <head>
            ...
```

- Lo mismo con `netcat` (nc):

```
echo "GET /" | nc 127.0.0.1 80
```

- Cliente/Servidor:

```
nc -l -p 5000
cat fichero | nc 192.168.1.1 5000
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Gestión de Incidencias

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Herramientas de monitorización

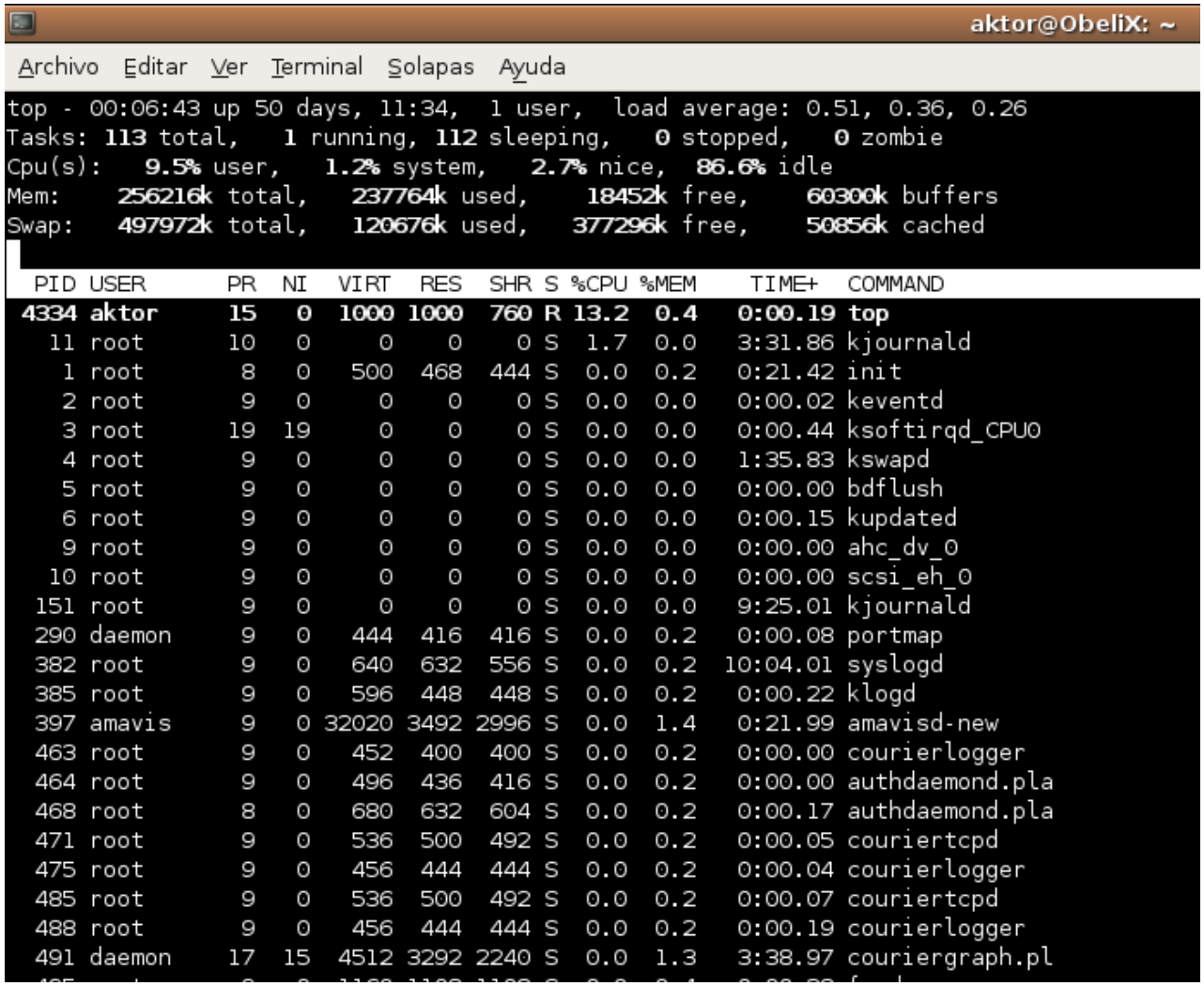
`top`

- Herramienta presente en todos los Unix que muestra información de forma continua (cada 3s.) sobre el estado del sistema, incluyendo la lista de procesos que más CPU están usando.
- Los procesos pueden ser ordenados en base a:
  - Recursos: utilización de CPU, memoria...
  - Usuarios: uid, gid...
  - Procesos: prioridad, estado...

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Herramientas de monitorización

top



Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Herramientas de monitorización

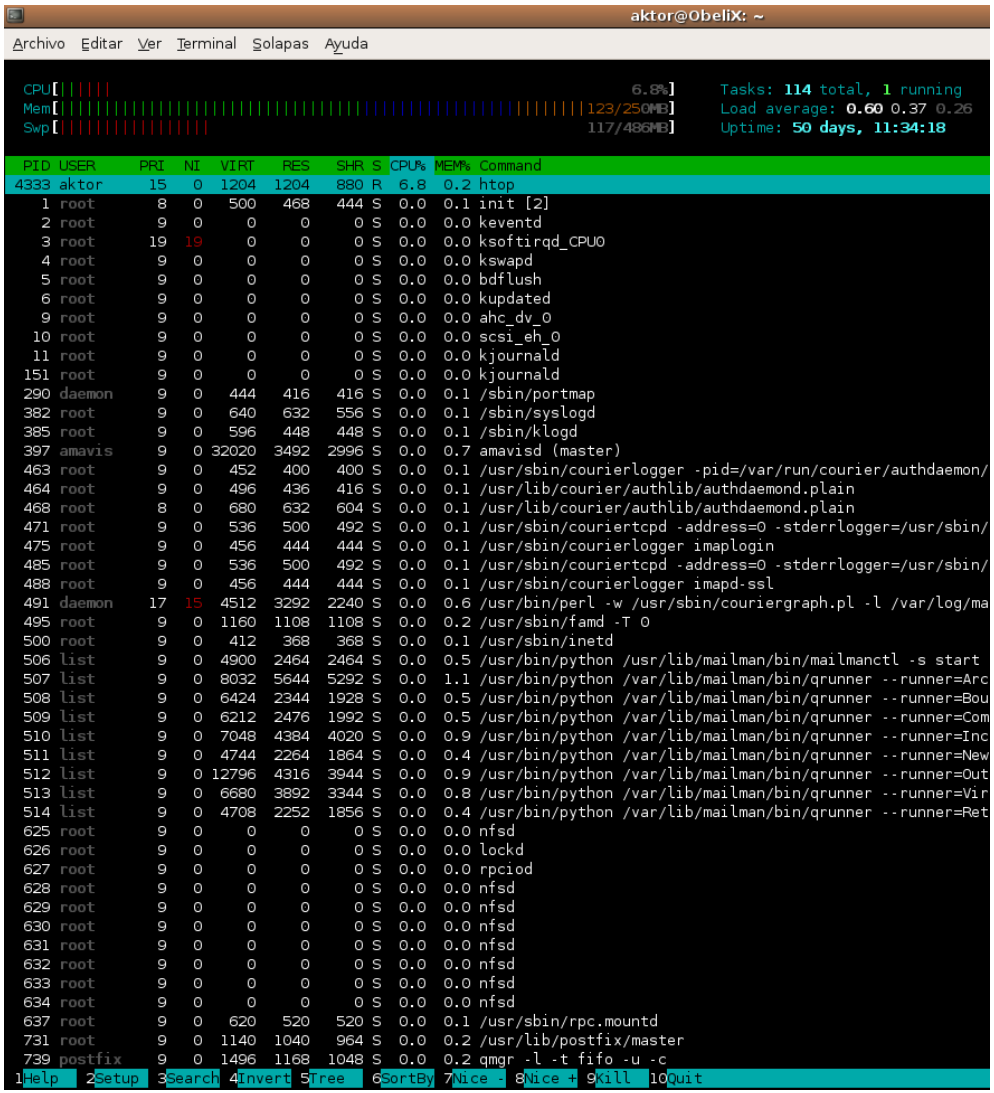
`htop`

- Herramienta similar a top pero con soporte para desplazamiento vertical entre los procesos.
- Permite gestionar los procesos (reiniciarlos, matarlos..) sin tener que conocer su PID.

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Herramientas de monitorización

htop



Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Herramientas de monitorización

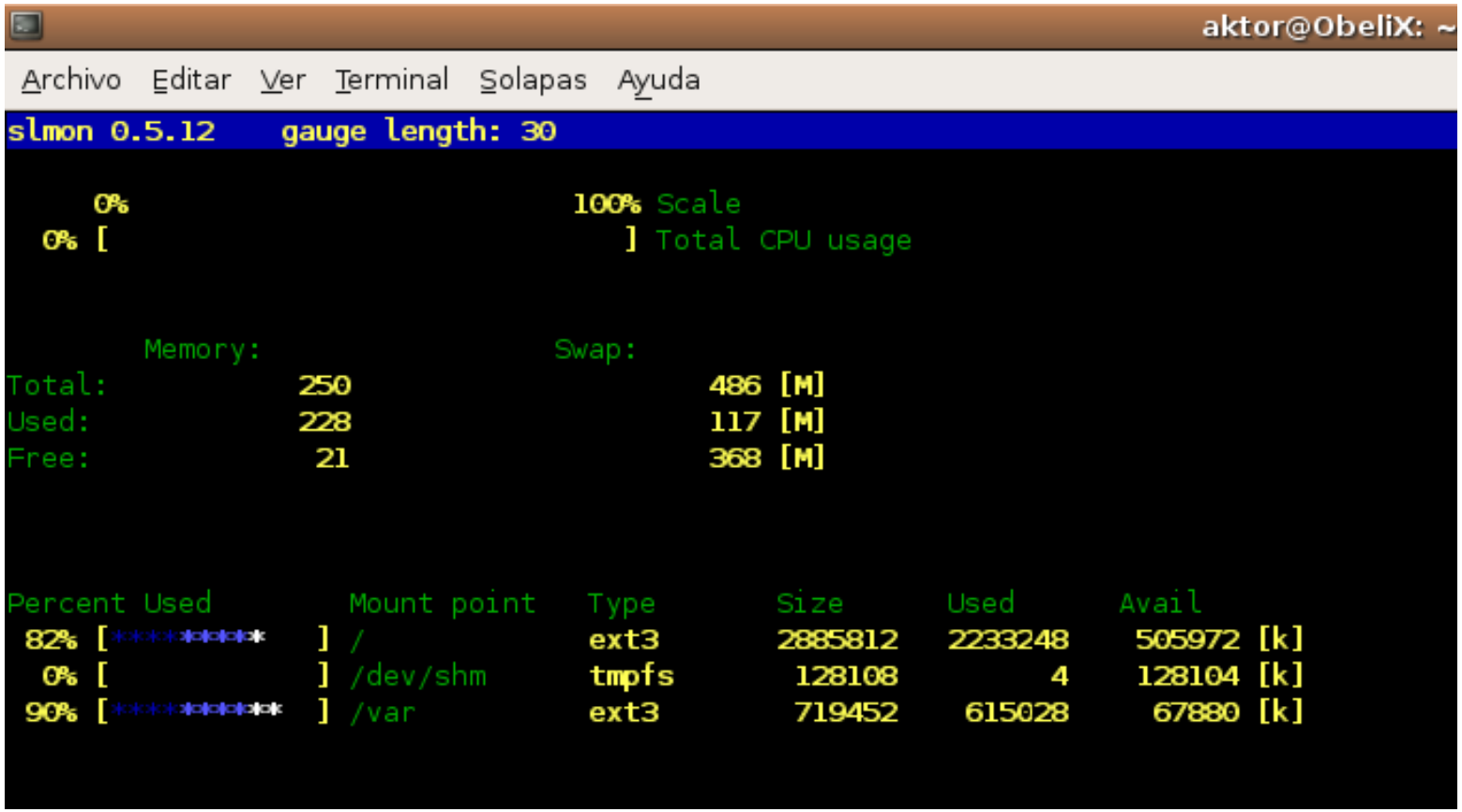
`slmon`

- Herramienta que permite monitorizar el rendimiento de un sistema en tiempo real.
- Los recursos que monitoriza son:
  - Carga de CPU
  - Memoria
  - Interfaces de red
  - Nº de usuarios logueados
  - Sistemas de ficheros
  - Procesos

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Herramientas de monitorización

slmon



Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Herramientas de monitorización

iostat

- Estadísticas de CPU y acceso a disco (`man iostat`)

```
Linux 2.6.15-26-686 (mihost)   22/11/06

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          51,29    0,01    0,42    0,13    0,00   48,15

Device:  tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
hda      2,55    14,45        59,21      13825459   56666320
hdb      0,09    18,78         7,29      17967684    6978592
```

  - `%user:` Uso de CPU en nivel de usuario.
  - `%nice:` Uso de en nivel de usuario con prioridad.
  - `%system:` Uso de CPU a nivel de sistema (kernel).
  - `%iowait:` Porcentaje de tiempo esperando peticiones I/O de disco.
  - `%steal:` Tiempo de espera involuntario por uso de varios CPU's (virtuales).
  - `%idle:` Tiempo de espera sin recibir petición I/O de disco.

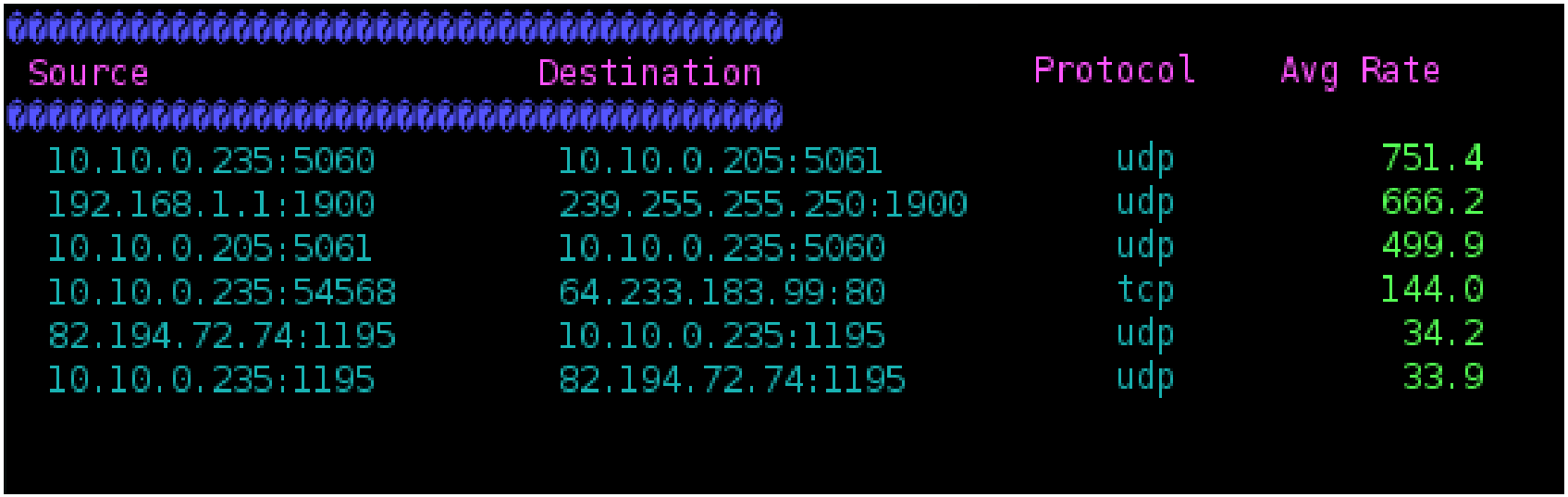
Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Herramientas de monitoriazión

potion

- Consumo de ancho de banda por interfaz y conexión.

```
potion eth0
```



Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Gestores de Arranque: LiLo

- En caso de que algo haya machacado nuestro MBR...
  - El objetivo es ejecutar `/sbin/lilo` de nuestro sistema para volver a escribir el MBR de la máquina y que lilo vuelva a funcionar.
  - Para ello podemos arrancar de diferentes formas:
    - Diskette de arranque preparado de antemano + /sbin/lilo
    - CD 1 de Debian GNU/Linux con opcion rescue, rescbf24 o similar:
    - Live CD (Knoppix o similar) + montar disco + chroot + /sbin/lilo (más complicada)

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Gestores de Arranque: Grub

- En caso de que algo haya machacado nuestro MBR...
  - ➔ Podemos recuperarlo de varios modos
    - Arrancamos con live-CD, montamos la partición, chroot y ejecutamos `grub-install /disco/duro`
    - Arrancamos con live-CD, consola, shell de grub (`grub`) y ejecutamos:

```
grub> root(hd0,0)
grub> setup(hd0)
grub> quit
```


Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Referencias

- Bash Scripting:
  - http://www.tldp.org/LDP/abs/html/
- Gestores de arranque:
  - http://www.gnu.org/software/grub/manual/html_node/
  - http://www.tldp.org/HOWTO/LILO.html
- Gestión de paquetes:
  - http://www.debian.org/doc/manuals/apt-howto/
- Administración de redes:
  - http://www.faqs.org/docs/linux_network/

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>

Licencia Copyleft

- Para la elaboración de este documento se han utilizado imágenes y documentos de otras personas como Eduardo González de la Herran y Pablo Garaizar Sagarminaga entre otros.
- Este documento está protegido bajo la licencia Reconocimiento-Compartir-Igual 2.5 España de Creative Common. http://creativecommons.org/licenses/by-sa/2.5/es/

Iker Sagasti Markina <iker@irontec.com> | Iñaki Baz Castillo <ibc@irontec.com>