8. Gestión de Recursos

8.1. Gestión del Procesador

Gestión del Procesador

Aunque la forma en que podemos ejecutar nuestros programas, incluidos los del administrador, no entra de lleno en lo que es la tarea de administración, siempre es conveniente saberla, incluida alguna que si es propia de él, como la ejecución periódica de ciertas tareas a través del demonio ***crond***.

Podemos esquematizar la ejecución de programas en dos tipos: ejecución presencial (la que menos interesa al administrador) y ejecución no presencial (realizada por el programador para grandes tareas o por el administrador para servicios en baja carga).

Ejecución presencial: background y manejo de "jobs"

Dentro de la ejecución presencial tenemos a su vez dos tipos: la ejecución en *foreground* y la ejecución en *background*. La primera es la normal, la segunda es la que se realiza cuando queremos que el control del terminal vuelve a la *shell* y no al programa ejecutado. Hay que tener en cuenta que las salidas del programa se podrán mezclar en el terminal con el *prompt* y los comandos que introduzcamos. Para realizar la ejecución de esta manera basta con poner un "&" detrás del comando:

```
$ programa > salida &
```

Cuando se ejecutan varios procesos en *background* se puede ver si realmente se están ejecutando con el comando ***ps*** (status de proceso) que no tiene argumentos y en el cual las opciones nos dicen que procesos representar (todos los del sistema, los de un usuario, etc.) y su formato. La forma más normal es:

```
$ ps
PID   TT    TIME  COMMAND
3243  30    0:11  ksh
3254  30    0:01  ps
```

la información que aparece se refiere a la identificación del proceso (PID), al número de terminal, al tiempo empleado en la ejecución y al nombre del proceso.

Además

/proc - pseudo-sistema de ficheros asociado a los procesos. Es empleado como interfase a las estructuras de datos del kernel asociado a los procesos. Permite visualiar el estado de cada proceso (statm),

```
[vpuente@bonito /proc]$ ls /proc/
1 18570 20219 23902 24908 29068 348 4464 827 9609 bus ioports mounts swaps
10735 18572 20224 23905 24922 29069 349 4467 828 973 cmdline kcore mtrr sys
10736 18917 21097 24002 24945 29070 359 476 8381 974 cpuinfo kmsg net tty
13352 18919 21368 24004 24988 29071 4 5 8391 975 devices ksyms partitions uptime
13648 18920 21369 24006 27536 29072 427 507 902 976 dma loadavg pci version
14128 18923 22844 24131 28453 29073 429 527 9077 977 fb locks rtc
14244 19566 23353 24147 28980 29074 430 566 9078 9987 filesystems mdstat scsi
17049 19691 23754 24809 29046 3 431 6 9079 9989 fs meminfo self
17055 19983 23899 24817 29056 307 432 615 955 9990 ide misc slabinfo
17899 2 23901 24907 29067 317 446 819 956 9991 interrupts modules stat
[vpuente@bonito /proc]$ ls 24817
cmdline cpu cwd environ exe fd maps mem root stat statm status
```

fd Ficheros Abiertos por el proceso
maps Rangos de memoria física asociada al proceso
stat estado actual del proceso: PID, PPID, utime, etc...
ps p. ej. accede a este sistema de ficheros para informarnos el estado de los procesos.
pstree

Conociendo el PID (número) de un proceso siempre lo podemos abortar con el comando ***kill***, al cual daremos como argumento el número del proceso.

Los procesos son generales al sistema, pero la shell permite tratar a los procesos creados en una sesión de una forma más sencilla, son los ***jobs*** (no hay que pensar que los jobs sean diferentes a los procesos, simplemente es una forma más sencilla de numerarlos, pertenecen a una sesión de trabajo y empiezan por cero). De esta manera, nosotros podemos suspender un proceso que se está ejecutando (*foreground*) con la pulsación de `ctrl-z` (no confundir con ejecución en *background*, en ésta, el proceso se está realmente ejecutando pero en segundo plano) y mandar un nuevo proceso a ejecutar. Para ver la lista de procesos suspendidos, tenemos en comando ***jobs***, que nos dará una lista de los mismos empezando con el identificador 0 entre corchetes. Para ejecutarles sólo tenemos que utilizar los comandos ***fg*** y ***bg***, que ponen respectivamente al *job* en ejecución *foreground* o *background*.

Ejecución no presencial

Como su nombre indica, la ejecución no presencial se realiza cuando queremos ejecutar un programa y no estar presentes, para ello nos deberemos asegurar que el programa no va a pedir datos interactivos, o si los va a pedir, hacer que estos procedan de un fichero y que la salida del programa no se realice en pantalla, si no a algún dispositivo o fichero.

Dentro de la ejecución no presencial podemos distinguir dos tipos: por un lado la ejecución futura que realizarán los usuarios del sistema cuando quieran que los programas se ejecuten a una determinada hora, o la ejecución a cargas bajas, cuando un programa por políticas de administración de la CPU no pueda ejecutarse en *foreground* por consumir muchos recursos del sistema y afectar a otros usuarios, y por otro lado la ejecución periódica, típicamente administrativa, realizada por *root* a través del demonio **crond**.

La ejecución futura y a cargas bajas se puede realizar respectivamente con los comandos ***at*** y ***batch***, internamente ejecutados con el demonio ***atd***. Este demonio se controla con el comando ***atd***, que en otros sistemas se llama ***atrun***. Los comandos relacionados con la ejecución futura serán:

- ***Atd***. Ya mencionado. Se utiliza para controlar al demonio de ejecución en dos aspectos: con –l la carga media de trabajo por debajo de la cual se ejecutarán los trabajos en batch, por defecto está al 80% y con –b el intervalo en segundos entre dos ejecuciones en batch, por defecto 60 seg.

- ***Batch***. Sirve para ejecutar programas en *batch* o *background* se activará cuando la carga del sistema lo permita, o dicho de otra manera, cuando el trabajo interactivo normal esté bajo (controlable con ***atd***). A este comando con –q se le puede cambiar la cola de ejecución por defecto (existen varias colas de ejecución nombradas desde la “a” a la “z”, la cola “a” con más prioridad [menor número nice] será la por defecto para ejecuciones con at, la cola “b” será para ejecuciones en batch), con –f indicaremos el fichero (normalmente en forma de una macro que se encargará de ejecutar lo que queremos) de ejecución, si no se pone nada se pedirá de forma interactiva, con –m indicaremos por *mail* al usuario la terminación del programa, y por último opcionalmente se puede poner el tiempo futuro en el que queremos que se ejecute (ver siguiente apartado).

- ***At***. Para ejecutar programas en un futuro. Tiene las mismas opciones que ***batch***, pero será obligatorio poner un tiempo que estará en forma de POSIX.2 extendido. Así, se puede poner en formato de hora como HH:MM; a unas horas predeterminadas: “midnight”, “noon” y “teatime”; con fecha como MMDDYY o MM/DD/YY o MM.DD.YY (mes, día año, siendo el año opcional); o de forma relativa como “tomorrow” (existen combinaciones de las anteriores y otros formatos como nombrados de mes, incluso con formato de 12 o 24 horas, pero lo más sencillo es como se ha indicado). De está manera si quisiéramos ejecutar un programa (dándosele de forma interactiva) a las 23 horas y 59 minutos del 14 de julio de 2001 deberíamos ejecutar entre otras posibilidades:

```
at 23:59 07.14.01
```

  si no hemos usado –f (modo interactivo) nos preguntará lo que queremos ejecutar, para salir utilizaremos “^d”. Como nota final del comando, deberemos asegurarnos que la fecha y la hora están bien con el comando ***date***.

- ***Atq***. Con este comando veremos una cola (–q) de trabajos a ejecutar, por defecto la “a”. Si somos el administrador veremos todos los trabajos, si somos un usuario sólo veremos los nuestros. Cada trabajo vendrá identificado con un número, estando almacenados internamente en el directorio de spool: `/var/spool/at`.

- ***Atrm***. Aceptará (si somos los propietarios) una identificación de trabajo *at* visto con ***atq*** y lo quitará de la cola.

No todos los usuarios pueden ejecutar el comando ***at***, esto dependerá de la política de administración del sistema. Para ello existen dos ficheros de control: `/etc/at.allow` y `/etc/at.deny`. Si los ficheros no existen sólo el administrador podrá usar ***at***. Si el primero existe, sólo los usuarios que figuren en él podrán usar ***at***. Si no existe, pero existe el segundo, todos los usuarios no mencionados en él podrán usar ***at*** (si el fichero existe pero no contiene nada, situación por defecto, todos los usuarios podrán usarlo).

Ejecución no presencial periódica

El otro tipo de ejecución no presencial es la realización de tareas periódicas, típicamente administrativas. Estas tareas serán ejecutadas por el demonio **crond** (*cron* procede de cronógrafo), que leerá el fichero de `cron` (normalmente `/etc/crontab`) donde estarán las instrucciones a ejecutar y sus periodos. En muchos sistemas este demonio se despertará cada minuto y revisará el fichero de órdenes (*crontab*) cada minuto, en otros sistemas más modernos, el analizar el fichero de órdenes hace que el demonio se despierte sólo a la siguiente hora en que le toque ejecutar algo.

Como casi siempre, existen dos grandes versiones de **cron**, una de BSD, donde el fichero de **cron** reside en `/usr/lib` y se distinguen dos clases de fichero: el **cron** local "`crontab.local`" y el **cron** de red "`crontab`", siendo estos ficheros sólo modificados por *root* (aunque se pueden ejecutar los comandos indicados en modo usuario). La edición del `crontab` se realiza a mano y se advierte del hecho, mandando al demonio ***crond*** una señal *hangup*. La otra versión es la de ATT escrita por Paul Vixie de DEC, es más abierta y la que se suele utilizar siempre. Existe un directorio de *cron* donde están los ficheros de *cron* (de cada usuario) que son remitidos a ese directorio a través de un comando "`crontab`", que es el que se encarga de avisar a ***crond*** para que se actualice. Además en ATT se puede explicitar que usuarios pueden ejecutar ***crontab*** siguiendo una política idéntica al comando ***at***, pero en este caso con los ficheros: `/etc/cron.allow` y `/etc/cron.deny`.

Independientemente de la versión (actualmente casi todas siguen POSIX), el formato clásico del fichero `crontab` consta de líneas de siete campos: minuto (de 0 a 59), hora (de 0 a 23), día (de 1 a 31), mes (de 1 a 12), día de la semana (de 1 a 7, si es 0 es domingo también), usuario (si es el crontab del sistema, sino ya está diferenciado por ficheros) y comando (las líneas de comentarios empiezan por "#"). Los campos estarán separados por espacios, "," significarán varios valores para un campo, "/" pasos y "-" un rango, un asterisco en un campo significa que éste no importa. Así, "0,30 * 13 * 2" significa cada media hora del martes y cada media hora del día 13 del mes, no sólo del martes 13; y "*/2 * * * *" significa cada dos minutos.

Además, en los ficheros modernos, se pueden definir y redefinir variables de entorno. Por ejemplo ***cron*** coge *sh* como SHELL y LOGNAME (USER en BSD) y HOME de `/etc/passwd` si tiene su usuario (cron) definido en `/etc/passwd`. Además, la variable MAILTO nos indicará a quien se manda un *mail* cuando se ejecuta un comando, si está vacío ("") no será enviado *mail* y si no está, será enviado al usuario propietario del **crontab** (hay que tener en cuenta que los comandos que pongamos sean los adecuados; no es adecuado poner un comando que tiene la salida por pantalla, porque entre otras cosas, si tenemos puesta la variable MAILTO, la salida irá al *mail* y no a la pantalla, esto es importante para las pruebas de funcionamiento que hagamos, de todos modos, la filosofía de *cron* no es hacer ejecutar programas interactivos, sino servicios que se necesita que se ejecuten cada cierto tiempo sin intervención humana).

A continuación aparece el fichero crontab general de mi sistema residente en `/etc`, donde se definen las variables que hemos comentado antes de aparecer la parte ejecutable:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly

# sysstat
0 * * * 0,6 /usr/lib/sa/sa1 600 6 &
5 19 * * * /usr/lib/sa/sa2 -A &
```

Después, como hemos dicho, cada usuario autorizado puede tener el suyo en `/var/spool/cron`. En este directorio (en algunos sistemas en vez de `/var` será `/usr`) aparecerán los nombres de los usuarios que hayan realizado el ***crontab***, estos ficheros no se editan, sino que son generados por el sistema al ejecutar ***crontab***. El proceso sería el siguiente:

1. Crear un fichero que tenga una o varias líneas con el formato de **crontab** (5 campos de tiempo, uno de usuario [opcional] y otro con el comando ejecutable por *sh*).

2. Ejecutar el comando ***crontab*** con el nombre del fichero.

3. Comprobar con la opción `-l` que está.

con lo cual aparecerá algo como esto:

```
# crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (trabajo.cron installed on Fri Jan 12 14:47:40 2001)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
*/2  * * * * cp /root/pepe.c /root/jose.c
*/2  * * * * echo "por fin ha funcionado"
```

si queremos editarlo después, usaremos `crontab -e` (aparecerán solo las dos últimas líneas en el ejemplo) y si queremos borrarlo `crontab -r`.

El sistema guarda en el directorio de registro de sucesos: `/var/log`, uno o varios ficheros referidos a ***cron*** (llamados precisamente así), donde podremos encontrar lo que ha estado ejecutando el demonio, así como en `messages` los cambios hechos por ***crontab***. Esto puede ser útil para ver si realmente se está haciendo lo que hemos indicado.

Existe un comando parecido al entorno ***cron*** que es el ***anacron***, que está indicado para sistemas que no están permanentemente encendidos. El periodo se indica en días y el fichero de órdenes será `/etc/anacrontab`. La dinámica es parecida a cron (se puede ver información como siempre con `man anacron`). Conviene recordar que *man* tiene una posible opción que es una cifra indicando la sección del manual donde queremos buscar la información pedida. Así, `man crontab` nos buscará información sobre el comando, pero `man 5 crontab` nos mostrará información sobre el fichero de configuración y su formato.

8.2. Monitorización

Monitorización del Sistema

Existen varios comandos para monitorizar el funcionamiento del sistema *on-line*, los más comunes son:

- ***top***. Es un comando interactivo (acepta subcomandos) mucho más potente que ***ps***, que nos dará una lista de los procesos más activos del sistema, actualizándose cada cierto tiempo (puede ser configurado con la opción `-d` o el interactivo "s"). En la cabecera de información muestra la línea de "*uptime*" (se puede conseguir también con el comando ***uptime***); la línea de las estadísticas de procesos: el número total de ellos y el número de cada estado (ejecutándose, dormido, parado, terminado, etc.); el estado de la CPU, dando porcentajes de uso de la misma en los diferentes estados: modo usuario, modo sistema, modo nice (tareas que tienen el número nice negativo), y de inactividad (estas dos últimas líneas pueden ser ocultadas/mostradas con el subcomando "t"); y dos líneas con el estado de la memoria (total, libre, usada y compartida), tanto real (física) como virtual (swap) que pueden ser ocultadas/mostradas con "m".

  Después vendrá información de los procesos, donde podremos obtener información de: el PID, pid; PPID, pid del padre; UID, identificación del usuario; USER, nombre del usuario; PRI, prioridad de la tarea; NI, número nice; SIZE, tamaño del código y datos del programa en Kbytes; TSIZE, tamaño del código (no sirve para procesos del sistema o con formato ELF); DSIZE, tamaño de datos más el stack asignado; SWAP, tamaño que reside en swap del proceso; RSS, tamaño de la memoria física usada, incluyendo librerías; SHARE, tamaño de memoria compartida usada; STAT, estado del proceso (S, durmiendo, R, ejecutándose, Z, zombie, T, parado, N, con nice positivo y W para "swapeado"); TIME, tiempo total de CPU desde que arrancó; %CPU y %MEM, porcentaje de CPU y memoria física desde la última actualización; y por último COMMAND que es el nombre del proceso.

  Como hemos dicho, ***top*** es un comando interactivo que aceptará órdenes, de las que destacamos: espacio o `^l` para actualizar, h para ayuda, k para matar procesos, n para cambiar el número de procesos a enseñar, r para cambiar la prioridad (nice), y f para añadir campos. Además podemos cambiar entre el tipo de información con: c, t, m, l. Y ordenar los procesos por criterios con: N, A, P, M, y T. ***top*** puede salvar la configuración actual con W en un fichero de configuración `/etc/toprc`.

- ***free***. Enseña el estado de la memoria, tanto física como secundaria (*swap*). El aspecto y la información presentada es:

```
# free
             total       used       free
shared    buffers     cached
Mem:         62872      58580       4292
40948       6936      27984
-/+ buffers/cache:      23660      39212
Swap:       136512       5460     131052
```

- ***uptime***. Nos dará la información que aparece en la primera línea de ***top*** (tiempo que lleva el sistema encendido, usuarios conectados y media de carga en el últimos 1, 5 y 15 minutos), con un aspecto como:

```
#uptime
  8:22pm  up 1 day,  3:04,  3 users,  load average:
0.40, 0.13, 0.06
```

- ***renice***. Cambia la prioridad o bien a un proceso, o bien a un usuario o bien a un grupo. Para ello habrá que dar el número nice (ver curso de programación) y respectivamente las siguientes opciones con sus argumentos: `-p`, `-u` y `-g`.

En lo que concierne a la monitorización del sistema, éste, en sus diferentes partes, guarda información sobre los sucesos que ocurren en el mismo. Este almacenamiento se puede hacer de varias maneras dependiendo de la capacidad del disco:

- Nulo. Se tiran los mensajes de sucesos, esto se hace desviándolos a `/dev/null`.
- Presentación. Se ponen en pantalla haciendo un desvío a `/dev/console`.
- Rotatorio. Se guardan los mensajes de hoy, de ayer, de dos días, etc. en ficheros que tienen la extensión nombre, nombre.1, nombre.2, etc., de esta manera, cada día que pasa, se van renombrando con un límite determinado que suele estar entre 4 y 8 días, es decir, entre 4 y 8 ficheros. Este método es el que sigue LINUX.
- Permanentes. En algunos casos, por seguridad, interesa tener estos ficheros guardados, esto se hace de forma rotatoria, pero el último fichero no desaparece sino que es almacenado en cinta o el sistema de *backup*, normalmente de forma comprimida.

Los mensajes del sistema son configurables en `/etc/syslog.conf`, donde indicamos a donde tienen que ir los mensajes. El aspecto de este fichero puede ser:

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none                          /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                          /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  /var/log/maillog

# Log cron stuff
cron.*                                                  /var/log/cron

# Everybody gets emergency messages, plus log them on another
# machine.
*.emerg                                                 *

# Save mail and news errors of level err and higher in a
# special file.
uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
```

```
local7.*                                    /var/log/boot.log
```

Los ficheros de registro o de *syslog* normalmente residen en el directorio `/var/log` y podremos encontrar ficheros generados por el *kernel*, por ***cron***, por el correo, por la red, etc. Además podremos ejecutar el comando ***lastlog*** en `/usr/bin` que interpreta el fichero `lastlog` en `/var/log` y que nos indicará para todos los usuarios, la última entrada al sistema.

8.3. Dispositivos

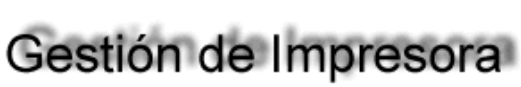
Como se vio en el apartado de creación de sistemas de ficheros y en el de el árbol de directorios, existe un directorio donde están todos los dispositivos del sistema llamado `/dev`. Básicamente existen dos tipos de dispositivos, los de bloque o acceso aleatorio y los de carácter o acceso secuencial, la gran diferencia entre ellos es que el de bloque almacena los datos y estos pueden ser accedidos en cualquier orden (de forma aleatoria) y el de carácter no, simplemente nos dará la siguiente ristra de bytes. Al hacer un ls –l /dev veremos en la primera columna de datos una b o una c que nos indicará el tipo de dispositivo, además, antes de la fecha (en vez del tamaño), veremos dos cifras que son el número mayor y el menor (major y minor), el mayor indica al kernel cual de los controladores (hardware y sus parte software el driver) corresponde al dispositivo y el menor el propio dispositivo (este número suele ser parte del nombre del fichero). La construcción de este nombre suele ser un poco críptica - por ejemplo hda2 o sda2 - y en algunos casos se suele hacer un enlace software (`ln -s`) con otro nombre más claro - por ejemplo lo que se hace con cdrom - .

Cuando se adquiere un nuevo dispositivo hay que crear un controlador (como se comentó la creación se suele hacer con el comando `mknod`), la parte software del controlador (driver) que se encarga de realizar las operaciones asociados al dispositivo - lecturas escrituras y control - debe suministrarse con el dispositivo. Esto en algunas ocasiones (muy reciente o muy viejo) es difícil de conseguir en Linux.

Una vez que tengamos el driver, deberemos incorporarlo al sistema (ver capítulo del kernel). Antiguamente estos drivers formaban parte del núcleo y por tanto este debía compilarse de nuevo. Actualmente existen los módulos que son parte del software del sistema pero que no están incluidos en el kernel y además así se consigue aligerar al núcleo. La filosofía es que los drivers muy utilizados estén en el núcleo y los poco utilizados en los módulos, con la ventaja añadida de que las pruebas se pueden hacer fuera del núcleo sin necesidad de compilarlo de nuevo.

Para visualizar los módulos que existen en el sistema se puede utilizar el comando **`/sbin/lsmod`** o también ver el directorio /proc/modules. Para cargar los controladores externos en primer lugar se incluirán en el directorio `/lib/modules` y se cambiarán los ficheros `/etc/modules` y `/etc/modules.conf` o `/etc/conf.modules`. Después se usarán los comandos **`/sbin/insmod`** y **`/sbin/modprobe`** para realizar la carga y la prueba (en algunos sistemas esto se hace de forma automática por el demonio **`kerneld`**).

8.4. Impresora



Antes de empezar a trabajar con la impresora y configurarla, conviene tener en cuenta varios aspectos importantes.

1. La impresora es un recurso compartido, esto quiere decir que puede ser usado por varios usuarios a la vez, con lo cual tiene que haber un arbitraje del recurso para evitar conflictos. Esto se consigue a través de un mecanismo de *spooling*, es decir, los trabajos de impresión no se mandan directamente al dispositivo, sino que se almacenan en una cola, y es un programa el que prioriza estos trabajos y los manda a la impresora secuencialmente.

2. Las impresoras modernas no son de texto puro, normalmente trabajan con un mapa de puntos, de hecho, la calidad de una impresora se define en dpi (puntos por pulgada) que normalmente pueden ir de 300 a 1200. Además, aceptan un lenguaje conocido como de descripción de página (PDL) para colocar las imágenes, pudiendo tener un módulo de RIP, que no es más que un sistema que convierte el PDF a bitmap. Entre los PDL más conocidos están el PostScript de Adobe o el PCL (Lenguaje de comandos de impresora) de Hewlett-Packard.

3. No se usa el mismo esquema de impresión en todas las máquinas UNIX, como casi siempre, existen tres familias: las que siguen el esquema BSD (vanilla), las de ATT o una combinación de ambos. En este caso, casi nadie sigue el esquema de POSIX. La primera usa comandos del tipo: lpd, lpc, lpr, etc., la segunda lpsched, lpadmin, lp, etc. La forma más segura de distinguir cual es nuestro sistema es mirar que demonio está ejecutándose, si es lpd será de tipo BSD y si es lpsched de tipo ATT, ya que muchas veces los comandos están combinados o se aceptan todos. En el caso de Red Hat se utiliza el esquema BSD, que actualmente es el más seguido y al que nos referiremos a partir de ahora.

Siguiendo el mecanismo indicado de *spooling*, los elementos encargados de realizar este trabajo en el sistema UNIX son:

- ***lpr***. Es el comando que utilizará el usuario para mandar trabajos de impresión. Cuando mandamos un trabajo de impresión, este comando crea dos ficheros en el directorio de *spool* (cola de impresión), uno de control, usualmente llamado cf o tf seguido de la identificación del trabajo (un número), y otro de datos, df más el número comentado. Cuando están creados los ficheros, ***lpr*** manda una notificación a ***lpd***. ***lpr*** se puede usar con varias opciones: –C para cambiar de prioridad al trabajo (cada prioridad viene marcada desde la "A" [por defecto] hasta la "Z" [mayor prioridad]), –h para quitar el encabezado, –k para el número de copias, –m para enviar un correo si hay problemas de impresión, –P para escoger la impresora, –T pone titulo al trabajo (se usará el comando ***pr***), –U especifica el usuario, y –w para dar la anchura del trabajo.

- ***lpq***. Servirá para ver como está la cola de impresión. Con –P podremos escoger la impresora y con –a nos dará información de todas.

- ***lprm***. Borrará un trabajo de la cola de impresión. Este trabajo se puede indicar de varias maneras: "**nada**", borrará el último trabajo que el usuario haya enviado, "**usuario**", borrará todos los trabajos de un usuario, "**all**", borrará todos los trabajos de esa cola. Se puede escoger la cola (impresora) con –P y el usuario con –U.

- `/var/spool/lpd`. Será la cola de impresión. Usualmente habrá un directorio por cada impresora conectada al sistema, de hecho, este directorio tendrá el nombre de la impresora. Por ejemplo, si sólo tenemos una impresora, ésta se llamará **lp** y el directorio colgado de `/var/spool/lpd` también. Las siguiente impresoras, a no ser que indiquemos lo contrario de llamarán **lp1**, **lp2**, etc.

- `/etc/printcap`. Será el fichero de configuración de las impresoras. En el caso de Red Hat este fichero será modificado con la utilidad gráfica ***printtool***, que nos dejará añadir impresoras locales, remotas en máquinas LINUX, Windows, Samba o NetWare, y con su propia dirección. Funciona como una base de datos y debe ser creado antes de poder usar la impresora. Este fichero está compuesto por definiciones de impresoras (los comentarios empiezan con #), que empiezan por su nombre (nombres supletorios pueden ser dados detrás separador por "|"). Por cada impresora existen una serie de definiciones identadas que empiezan y terminan por ":" (si forman parte de la misma definición de impresora y están en distintas líneas, éstas acabarán con una "\"), los tipos de definiciones (o variables de printcap) las podemos agrupar es:

  - Especificaciones de ficheros y directorios: Las principales variables son: sd, directorio de spool; lf, fichero de errores; lp, nombre de dispositivo (si la impresora es en red será nulo); af, fichero de accounting.

  - Información de acceso remoto: rm, máquina remota donde está conectada la impresora; rp, nombre de la impresora remota.

  - Filtros de impresión. of, fichero de salida, if, fichero de entrada.

  - Parámetros. mx, tamaño máximo de trabajo, br, ratio de baudios (sólo para impresoras conectadas por línea serie), existen otras variables para cambiar ciertos bits de

comunicación.

- Información de página. pw y pl para anchura y longitud de página y px y py para la resolución horizontal y vertical.

A continuación aparece un ejemplo de un fichero ***printcap*** con una impresora conectada directamente a la red con su propio nombre y dirección, con una impresora local y con una conectada remotamente en un sistema UNIX:

```
##PRINTTOOL3## DIRECT
lp|cachon:\
:sd=/var/spool/lpd/lp:\
    :mx#0:\
    :sh:\
    :af=/var/spool/lpd/lp/acct:\
    :lp=/dev/null:\
    :if=/usr/lib/rhs/rhs-printfilters//directprint:
##PRINTTOOL3## LOCAL
lp0:\
    :sd=/var/spool/lpd/lp0:\
    :mx#0:\
    :sh:\
    :lp=/dev/lp0:
##PRINTTOOL3## REMOTE
lp1:\
    :sd=/var/spool/lpd/lp1:\
    :mx#0:\
    :sh:\
    :rm=bonito.atc.unican.es:\
    :rp=/var/spool/lpd/lp:\
    :lpd_bounce=true:
```

- ***lpd***. Es el proceso de sistema o demonio encargado de leer la cola de impresión y mandar los trabajos almacenados al dispositivo impresora. Cuando es despertado por ***lpr*** , coge el primer trabajo de la cola (política FIFO que se puede cambiar con ***lpc***) y verifica a través de `/etc/printcap` si el trabajo es para una impresora local o remota, en el primer caso chequea que haya un demonio creado para la cola de impresión y si no lo hay crea uno que es una copia de si mismo (esquema típico cliente/servidor), en el segundo establece una conexión con el ***lpd*** de la otra máquina y le manda los dos ficheros, el de control y el de datos.

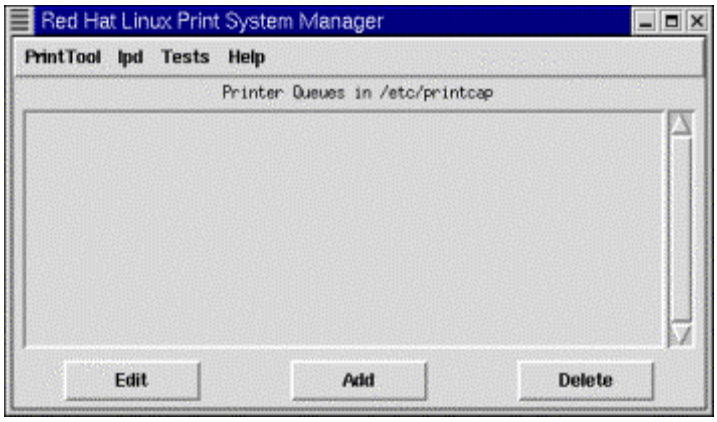
- Existirá un fichero de filtro de entrada opcional, indicado en `/etc/printcap`, donde se podrá hacer alguna transformación del formato de los datos a enviar a la impresora. Estos ficheros de filtro cada día son menos importantes, de hecho en la mayoría de las impresoras, si se necesitan, vienen con el software de la misma.

- ***lpc***. Es el comando de control de la impresora. Servirá para: habilitar o deshabilitar una impresora (o todas), así como una cola de impresión (o todas); mover trabajos al principio de la cola (política de planificación distinta FIFO); ver el estado de impresoras, demonios de impresión y sus colas asociadas; arrancar o parar servicios de impresión; bloquear o liberar trabajos de impresión; redirigir trabajos de una impresora a otra; restablecer trabajos de impresión después de que haya habido problemas y reimprimir trabajos.

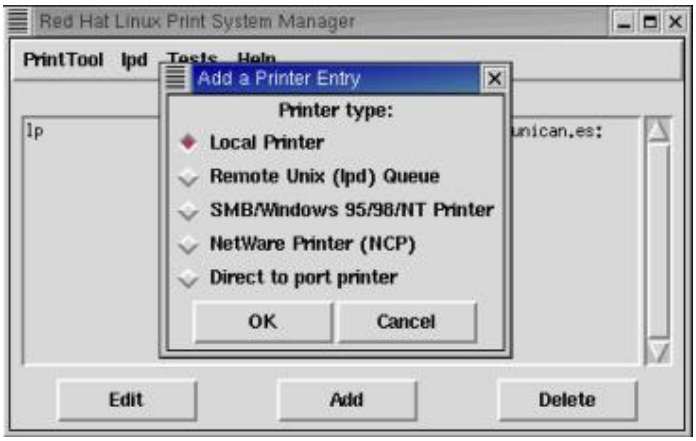
Printtool

En Red Hat es muy sencillo instalar una impresora, ya que disponemos de la herramienta gráfica ***printtool***, usable sólo por el administrador. Antes de instalarla, lo primero que deberemos de garantizar es que la impresora está soportada por el sistema para, para ello podemos visitar: http://hardware.redhat.com/redhatready/cgi-bin/us/db-hcl.cgi para cualquier tipo de hardware y específicamente para impresoras: http://www.linuxprinting.org/printer_list.cgi.

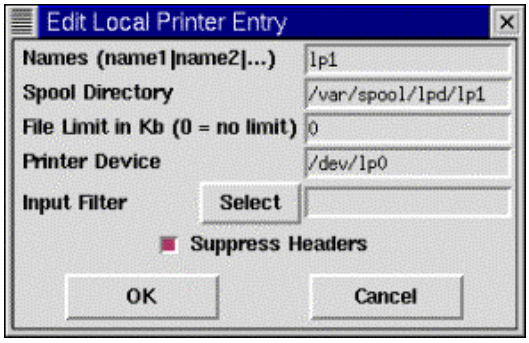
Una vez comprobado, arrancaremos la aplicación, que tiene un aspecto como la siguiente figura:



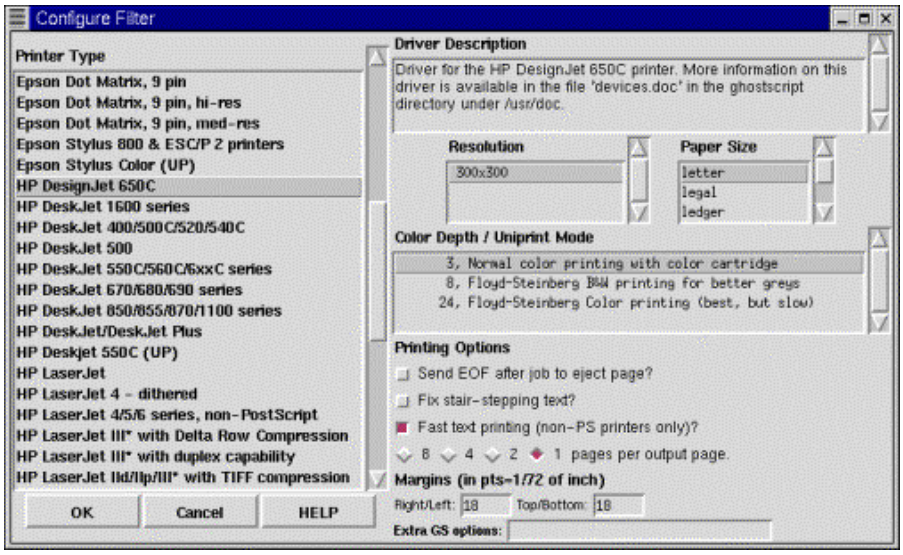
En ella podremos editar, añadir o borrar impresoras. Estas impresoras pueden estar conectadas de muchas maneras, en Red Hat 7.0 de cinco maneras: local, remota en UNIX, remota en Windows, remota en Net Ware y con su propia dirección. Una ventana como la siguiente nos aparecerá en pantalla:



Dependiendo del tipo de impresora que escojamos, habrá que dar una serie de parámetros que rellenará de forma automática el fichero `/etc/printcap` sin más molestias. En la siguiente figura aparece la ventana de datos de una impresora local.



Cuando seleccionemos el filtro podemos tener problemas si nuestra impresora no aparece en la lista suministrada. La solución más sencilla es encontrar uno parecido (del mismo fabricante, pero un modelo diferente). Después se pueden seleccionar otras opciones, como el tamaño del papel y otro tipo de resoluciones (ver siguiente figura). Una vez configurada la impresora iremos a la ventana principal de la aplicación e intentaremos imprimir una página de prueba, si no funciona deberemos cambiar el filtro. Existen otra forma más sofisticada que es utilizando el "***lpdomatic***" de la forma que se menciona en http://www.linuxprinting.org/lpd-doc.html.



Además, los escritorios Gnome y KDE nos proporcionan a través del menú, dos aplicaciones más de impresión, una para controlar las impresoras de tipo HP Laser Jet y otra para gestionar las colas de impresión de forma gráfica (lo que se podía hacer con el comando ***lpc***).

Otros sistemas

Existen otras técnicas para manejar impresoras como las CUPS (common unix printing system) de parecidas características. Son más complejas de configurar pero tienen la ventaja de ser compatibles con los servidores de web (apache).

8.5. P. Serie

Gestión de Líneas Serie

*D*entro de este apartado entran dispositivos como los terminales de texto (básicamente la configuración del teclado), modems o incluso ratones. Veremos someramente como se tratan y nos referiremos exclusivamente a sistemas de tipo Linux sobre PC basados en Intel.

Cada dispositivo tiene asignado un fichero en el directorio `/dev`. En este directorio (en mi caso hay 2034 dispositivos tty) podremos encontrar los que están asignados a los cuatro puertos serie de un PC, que además estarán diferencias por entrada o salida:

```
/dev/cua0, /dev/ttyS0 (COM1) dirección 0x3f8 IRQ 4
/dev/cua1, /dev/ttyS1 (COM2) dirección 0x2f8 IRQ 3
/dev/cua2, /dev/ttyS2 (COM3) dirección 0x3e8 IRQ 4
/dev/cua3, /dev/ttyS3 (COM4) dirección 0x2e8 IRQ 3
```

Los *ttySx* serán los dispositivos serie de entrada (teclado - las consolas virtuales tienen los nombres `/dev/tty1` a `tty6 –`) y los *cuax* serán los de salida (orientados a modem). Cada puerto serie tiene su propia dirección y tiene asignada una interrupción, esto se puede configurar con el comando ***setserial*** (ver `man setserial`). Además de estos cuatro dispositivos, en ese directorio puede haber enlaces para el ratón o el modem (`mouse` y `modem`) que apuntarán a los dispositivos básicos correspondientes, por ejemplo en mi sistema existe un ratón serie que tiene asignado un dispositivo ***mouse*** que es un enlace a *ttyS0*, ya que está conectado a COM1. Otro ejemplo de enlace simbólico sería el terminal principal `/etc/console`.

Como sabemos, casi todos los dispositivos vienen creados ya en el sistema y vienen asignados en la tabla de

dispositivos por un número mayor y uno menor, que es el que realmente determina su comportamiento, independientemente del nombre. En el caso de los puertos serie, estos números serán:

```
/dev/ttyS0 mayor 4, menor 64    /dev/cua0 mayor 5, menor 64
/dev/ttyS1 mayor 4, menor 65    /dev/cua1 mayor 5, menor 65
/dev/ttyS2 mayor 4, menor 66    /dev/cua2 mayor 5, menor 66
/dev/ttyS3 mayor 4, menor 67    /dev/cua3 mayor 5, menor 67
```

Si por cualquier causa no existieran, siempre les podremos crear con la orden ***mknod***:

```
$ mknod -m 666 /dev/cua0 c 5 64
$ mknod -m 666 /dev/ttyS0 c 4 64
```

Creados los dispositivos, por cada línea serie se establece en el proceso de arranque (a través de `inittab` o de un fichero asociado como `rc.local` o `rc.serial`), el proceso ***getty***, que coloca las principales características del terminal o modem (existe una versión mínima de ***getty*** que es ***mingetty*** especialmente diseñado para las consolas virtuales, que tienen asignadas los dispositivos *ttyx*) y después ejecuta un proceso de *login*, dando valor a la variable de la shell TERM.

El proceso ***getty*** toma el dispositivo terminal de la línea de comando de `inittab`, la asociación entre el dispositivo terminal y el tipo de terminal está en un fichero en `/etc` que depende del tipo de sistema que usemos: *ttys*, *ttytab* o en Linux *ttytype* acompañado del fichero `/etc/securetty` que nos dirá desde que terminales podemos entrar como *root*.

Al igual que ocurre con las impresoras, pero de forma más compleja, existe una base de datos que nos indica los posibles tipos de terminales que podemos conectar y que reside en el fichero `/etc/termcap`, donde también se gestionan la asignación de las teclas del terminal (teclado). En algunas distribuciones existe el script `/etc/sysconfig/keyboard` para trabajar con ese fichero y la asignación de teclas. Obviamente los tipos de terminal que podamos poner en *ttytype* tendrán una entrada en esta base de datos.

Comandos relacionados con la disposición de terminales serán:

- ***tty***. Pinta el dispositivo terminal que estamos usando.

- ***tset***. Inicializa el terminal, para ello lee el tipo de terminal de la propia línea de comandos, de la variable TERM, o lo pondrá a desconocido.

- ***stty***. Cambia parámetros del terminal, tomando como argumento opcional un tipo de dispositivo. Esto nos permitirá cambiar algunos parámetros del terminal o caracteres especiales, como por ejemplo como se realiza el borrado. Si no ponemos nada, el comando presentará la velocidad del terminal y la definición de caracteres especiales.