Introducción a GNU/Linux

Explicación de práctica 2

Introducción a los Sistemas Operativos

Facultad de Informática
Universidad Nacional de La Plata

2015

- Configuración de discos IDE (*Integrated Device Electronics*):
  - Master o Slave
  - Primer y Segundo bus IDE
- Denominación de los discos basada en los buses:
  - **/dev/hda**: configurado como Master en el 1º bus IDE
  - **/dev/hdb**: configurado como Slave en el 1º bus IDE
  - **/dev/hdc**: configurado como Master en el 2º bus IDE
  - **/dev/hdd**: configurado como Slave en el 2º bus IDE
- Particiones primarias $\rightarrow$ 1 a 4
- Particiones logicas $\rightarrow$ desde 5 en adelante

Características - Configuración de discos (cont.)

- Configuración de discos SCSI: se basa en *LUN*
- Denominación de los discos basada en la identificación de los buses:
  - **/dev/sda**
  - **/dev/sdb**
  - **/dev/sdc**
  - **/dev/sdd**
  - ...
- La nomenclatura para los discos SATA es la misma
- Particiones primarias:
  - Se numeran de la 1 a la 4 (solo estas se pueden marcar como activas $\rightarrow$ booteables)
- Particiones extendidas:
  - Sus unidades o particiones lógicas se numeran a partir de la 5

Características - Configuración de discos (cont.)

- Nueva nomenclatura utilizada:
  - Con la evolución de las distribuciones GNU/Linux, se comenzó a utilizar **"udev"** (*.rules*) como gestor de dispositivos:
    - Su función es controlar dinámicamente los archivos del /dev *SOLO* en base al hardware detectado
    - Soporta **Persistent Device Naming**
    - Motiva su uso, el no poder garantizar que tras distintos arranques del SO, los dispositivos se sigan llamando de la misma manera. (Suponga disco 1 y 2, que disco 1 se quita y contorladoras SCSI/SATA mixtas)
    - Reemplaza a *devfs* y *hotplug*
    - Se desentiende del **Major** y **Minor Number**
    - Se basa en eventos y permite que nuevos dispositivos sean agregados posteriormente al arranque

Características - Configuración de discos (cont.)

- A futuro, todos los dispositivos llamados hdX serán denominados sdX ← Introducido en Debian/Squeeze
- Por estas y otras razones se adoptan 4 mecanismos nuevos para nomenclar[1]:
  - Nombres persistentes por **UUID** (Universal Unique Identifier):

```
$ ls -l /dev/disk/by-uuid/
2d781b26-0285-421a-b9d0-d4a0d3b55680 -> ../../
    sda1
31f8eb0d-612b-4805-835e-0e6d8b8c5591 -> ../../
    sda7
```

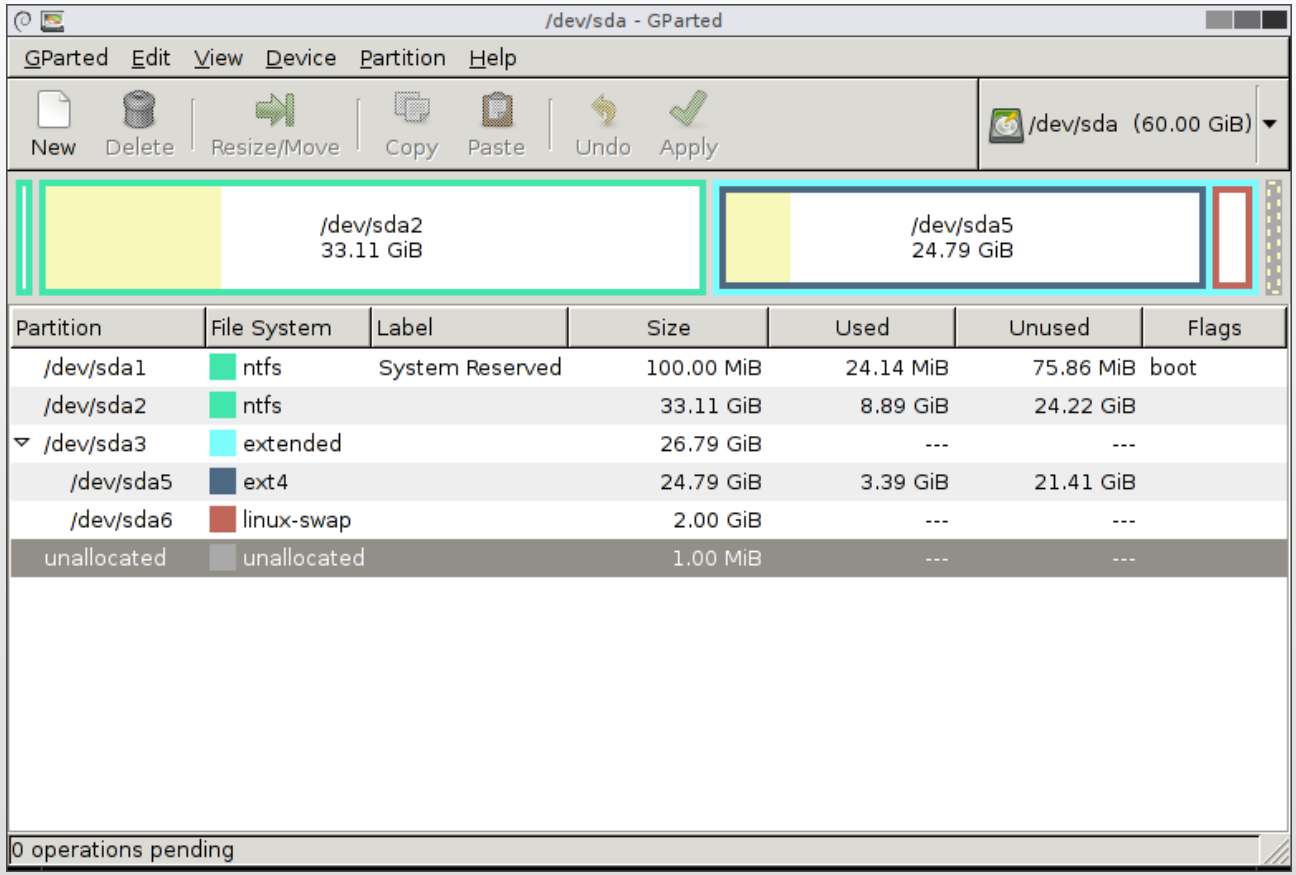
  - Utilizando **labels**

```
$ ls -l /dev/disk/by-label
data -> ../../sdb2
data2 -> ../../sda2
```

[1] http://wiki.debian.org/Part-UUID

- Existen diversos modos de instalar GNU/Linux:
  - Debemos tener en cuenta la arquitectura de hardware
    - amd64: Arquitectura de 64 bits
    - arm ó armel: Advanced Risc Machine
    - i386: Arquitectura de 32 bits
    - ia64: intelItanium o Intel Architecture-64
    - Otras
  - Podemos instalarlo desde un CD descargao de la web
  - Podemos instalarlo desde un USB: Unetbootin
    - Permite crear instaladores o LiveCD utilizando USBs

- El particionado de un disco se lo puede realizar mediante:
  - Software destructivo: *fdisk*
  - Software no destructivo: *fips*, *gparted*



- No existe el concepto de *extensión* en el nombre de un archivo
- Los subdirectorios no se separan con el carácter '\'
- Es case sensitive
- Entre un comando y sus parámetros debemos dejar obligatoriamente un espacio en blanco
- Separación de entorno gráfico y texto

- Presente en cualquier distribución de GNU/Linux
- Posee 3 modos de ejecución:
  - Modo Insert (**Ins** o **i**)
  - Modo Visual (**v**)
  - Modo de Órdenes o Normal (**Esc**)
- Se le puede enviar una serie de comandos útiles
  - **w**: escribir cambios
  - **q** ó **q!**: salir del editor
  - **dd**: cortar
  - **y**: copiar al portapapeles
  - **p**: pegar desde el portapapeles
  - **u**: deshacer
  - /frase: busca “frase” dentro del archivo

- Todo usuario debe poseer credenciales para acceder al sistema
  - root: es el administrador del sistema (superusuario)
  - otros: usuarios estándar del sistema (/etc/sudoers)
- Archivos de configuración:
  - **/etc/passwd**

```
$ cat /etc/passwd
ndelrio:x:2375:500:Nico del Rio,,,,Usuarios:/
    home/admins/ndelrio:/bin/bash
```

  - **/etc/group**

```
$ cat /etc/group
infraestructura:x:500:
```

  - **/etc/shadow**

```
$ cat /etc/shadow
ndelrio:$1$HamkgCYM$TtgfLJLpIltxutaiqh/u9
    /:13273:0:99999:7:::
```

- Comando para el manejo de usuarios:
  - **useradd** <nombreUsuario>:
    - Agrega el usuario
    - Modifica los archivos /etc/passwd
    - Alternativa → **adduser**
  - **passwd** <nombreUsuario>:
    - Asigna o cambia la contraseña del usuario
    - Modifica el archivo /etc/shadow
  - **usermod** <nombreUsuario>:
    - **-g**: modifica grupo de login (Modifica /etc/passwd)
    - **-G**: modifica grupos adicionales (Modifica /etc/group)
    - **-d**: modifica el directorio *home* (Modifica /etc/passwd)
  - **userdel** <nombreUsuario>: elimina el usuario
  - **groupdel** <nombreGrupo>: elimina el grupo

- Se aplican a directorios y archivos
- Existen 3 tipos de permisos y se basan en una notación octal:

Permiso	Valor	Octal
Lectura	R	4
Escritura	W	2
Ejecución	X	1

- Se aplican sobre los usuarios:
  - Usuario: permisos del dueño → **U**
  - Usuario: permisos del grupo → **G**
  - Usuario: permisos de otros usuario → **O**
- Se utiliza el comando **chmod**:

```
$ chmod 755 /tmp/script
```

- Algunos comandos útiles:
  - **ls**
  - **cd**
  - **mkdir**
  - **rmdir**
  - **rm**
  - **mv**
  - **cp**
  - **man**
  - **info**

- El bootloader o cargador de arranque es un programa que permite cargar el Sistema Operativo. Puede llegar a cargar un entorno previo a la carga del sistema
- Generalmente se utilizan los cargadores multietapas, en los que varios programas pequeños se van invocando hasta lograr la carga del SO
- En cierto sentido, el código del *BIOS* forma parte del bootloader, pero el concepto está más orientado al código que reside en el *Master Boot Record* (512b)
- El MBR está formado por el *MBC* (446b) y la *Tabla de Particiones* (64b)
- Sólo el MBC del Primary Master Disk es tenido en cuenta
- El MBR existe en todos los discos, ya que contiene la tabla de particiones

Proceso de arranque **System V**

1. Se empieza a ejecutar el código de la BIOS
2. La BIOS ejecute el POST
3. La BIOS lee el sector de arranque (MBR)
4. Se carga el gestor de arranque (MBC)
5. El bootloader carga el *kernel* y el *initrd*
6. Se monta el initrd como sistema de archivos raíz y se inicializan componentes esenciales (e.j.: scheduler)
7. Se ejecuta el proceso *init* y se desmonta en initrd
8. Se lee el /etc/inittab
9. Se ejecutan los scripts apuntados por el *runlevel 1*
10. El final del runlevel 1 le indica que vaya al runlevel por defecto
11. Se ejecutan los scripts apuntados por el runlevel por defecto
12. El sistema está listo para usarse

Proceso de arranque (cont.)

1. Su función es cargar todos los subprocesos necesarios para el correcto funcionamiento del SO
2. Posee el PID 1 y se encuentra en /sbin/init
3. En SysV se lo configura a través del archivo /etc/inittab
4. No tiene padre y es el padre de todos los procesos (*pstree*)
5. Es el encargado de montar los filesystems y de hacer disponible los demás dispositivos

- Es el modo en que arranca Linux (3 en *Redhat*, 2 en *Debian*)
- El proceso de arranque lo dividimos en niveles
- Cada uno es responsable de levantar o bajar una serie de servicios

- Se encuentran definidos en /etc/inittab
  id:nivelesEjecución:acción:proceso
  - **Id**: identifica la entrada en inittab (1 a 4 carácteres)
  - **NivelesEjecución**: el/los niveles de ejecución en los que se realiza la acción
  - **Acción**: describe la acción a realizar
    - **wait**: inicia cuando entra al runlevel e init espera a que termine
    - **initdefault**
    - **ctrlaltdel**: se ejecutará cuando init reciba la señal *SIGINT*
    - **off**, **respawn**, **once**, **sysinit**, **boot**, **bootwait**, **powerwait**, etc.
  - **Proceso**: el proceso exacto que será ejecutado

```
$ cat /etc/inittab
id:2:initdefault:
si::sysinit:/etc/init.d/rcS
ca::ctrlaltdel:/sbin/shutdown -t3 -r
```

Runlevels (cont.)

- Existen 7, y permiten iniciar un conjunto de procesos al arranque o apagado del sistema
- Según el estándar:
  - **0**: halt (parada)
  - **1**: single user mode (monousuario)
  - **2**: multiuser, without *NFS* (modo multiusuario sin soperte de red)
  - **3**: full multiuser mode console (modo multiusuario completo por consola)
  - **4**: no se utiliza
  - **5**: X11 (modo multiusuario completo con login gráfico basado en X)
  - **6**: reboot

- Los scripts que se ejecutan están en /etc/init.d
- En /etc/rcX.d (donde X = 0..6) hay links a los archivos del /etc/init.d
- Formato de los links:
  [S|K]<orden><nombreScript>

```
$ ls -1 /etc/rcS.d/
S55urandom
S70x11-common
```

- **S**: lanza el script con el argument start
- **K**: lanza el script con el argument stop

insserv

- Se utiliza para administrar el orden de los enlaces simbólicos del /etc/rcX.d, resolviendo las dependencias de forma automática
- Utiliza cabeceras en los scripts del /etc/init.d que permiten especificar la relacion con otros scripts rc $\rightarrow$ LSBInit (Linux Standard Based Init)
- Es utilizado por *update-rc.d* para instalar/remover los links simbólicos

insserv *(cont.)*

- Las dependencias se especifican mediante *facilities* → **Provides** keyword
- Las facilities que comienzan con $ se reservan para el sistema (*$syslog*)
- Los scripts deben cumplir *LSB init script*:
  - Proveer al menos 'start, stop, restart, force-reload and status'
  - Retornar un código apropiado
  - Declarar las dependencias

- LSB init script headers:

```
### BEGIN INIT INFO
# Provides:          my_daemon
# Required-Start:    $syslog $remote_fs
# Required-Stop:     $syslog $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: This is a test daemon
# Description:       This is a test daemon
#                    This provides example about
    how to
#                    write a Init script.
### END INIT INFO
```

- Upstart es el reemplazo de SystemV (*Ubuntu*, *Fedora*, etc.)
- Permite la ejecución de trabajos en forma asincrónica a través de eventos (*event-based*) como principal diferencia con sysVinit que es estrictamente sincrónico (*dependency-based*)
- Estos trabajos se denominan **jobs**
- El principal objetivo de un job es definir servicios o tareas a ser ejecutadas por init
- Son scripts de texto plano que definene las acciones/tareas (unidad de trabajo) a ejecutar ante determinados eventos
- Cada job es definido en el /etc/init (**.conf**)
- Suelen ser de dos tipos:
  - **Task**: ejecución finita (*task*) $\rightarrow$ not respawning $\rightarrow$ exit 0 o uso de *stop*
  - **Service**: ejecución indeterminada $\rightarrow$ respawning

- Los jobs son ejecutados ante eventos (arranque del equipo, inserción de un dispositivo USB, etc.):
  - Es posible crear eventos pero existen algunos de manera estándar[2]
  - Definido por **start on** y **stop on**
- Es compatible con SystemV → /etc/init/rc-sysinit.conf, runlevels, scripts en /etc/init.d, objetivo start y stop
- Cada job posee un objetivo (*goal start/stop*) y un estado (*state*)
  - En base a ellos se ejecuta un proceso específico
  - Al inicio, init emite el evento *startup*

[2] `http://people.canonical.com/~jhunt/upstart/upstart-states.png`

Proceso de arranque **Upstart** *(cont.)*

- Un job puede tener uno o varias tareas ejecutables como parte de su ciclo de vida y siempre debe existir la tarea principal
- Las tareas de un job se definen mediante *exec* o *script ... end script*
- A través de *initctl* podemos administrar los jobs del demonio de Upstart:
  - start <job>: cambia el objetivo a start del job especificado
  - stop <job>: cambia el objetivo a stop del job especificado
  - emit <event>: event es emitido causando que otros jobs cambien a objetivo start o stop
- No más /etc/inittab

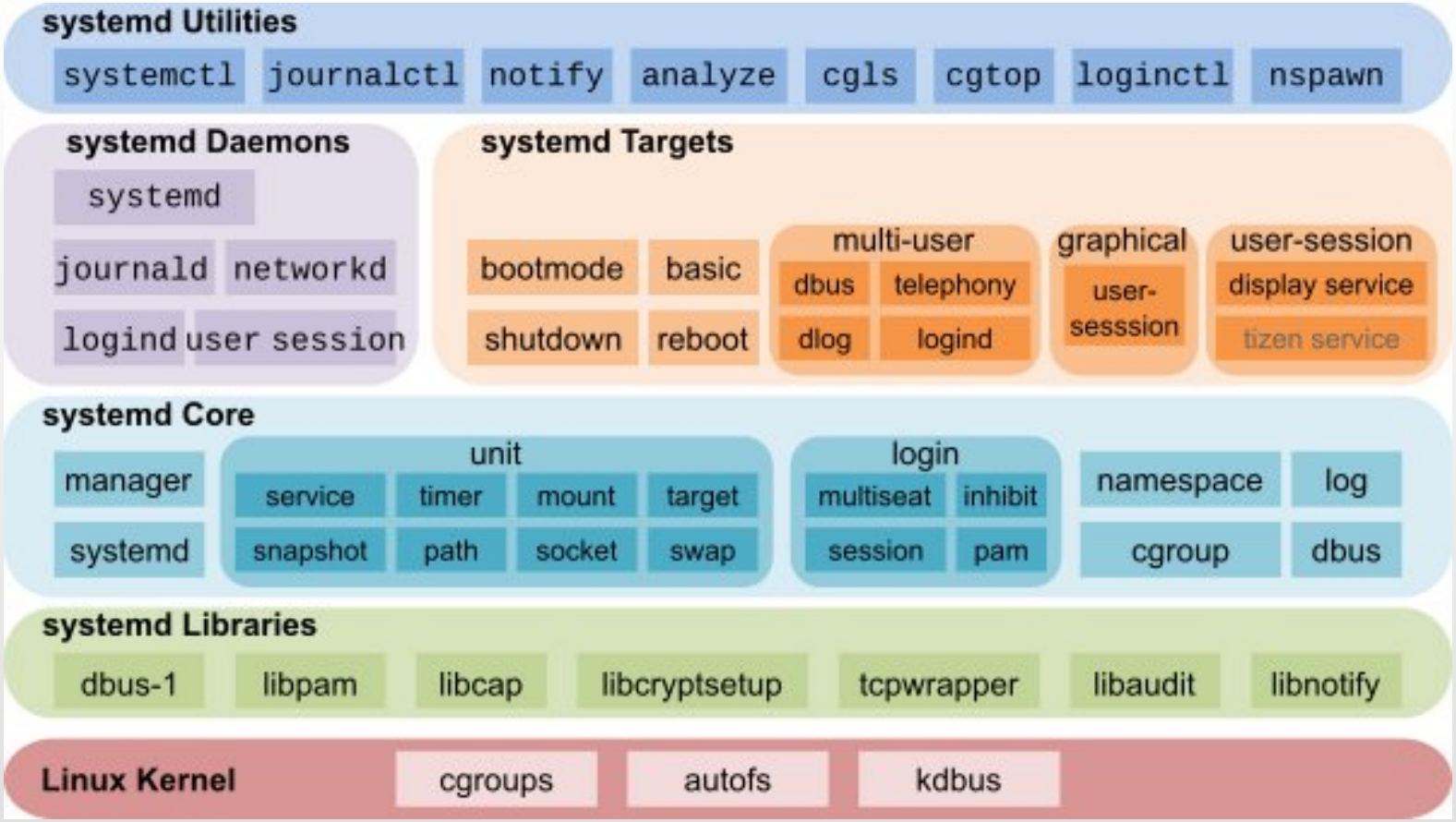
Systemd

- Es un sistema que centraliza la administración de demonios y librerias del sistema
- Mejora el paralelismo de booteo
- Puede ser controlado por **systemctl**
- Compatible con SysV $\rightarrow$ si es llamado como *init*
- El demonio *systemd* reemplaza al proceso init $\rightarrow$ este pasa a terner *PID 1*
- Los runlevels son reemplazados por **targets**
- Al igual que con Upstart el archivo /etc/inittab no existe más

Systemd *(cont.)*

- Las unidades de trabajo son denominadas **units** de tipo:
  - **Service**: controla un servicio particular (*.service*)
  - **Socket**: encapsula IPC, un sockect del sistema o file system *FIFO* (*.socket*) $\rightarrow$ sockect-based activation
  - **Target**: agrupa units o establece puntos de sincronización durante el booteo (*.target*) $\rightarrow$ dependencia de unidades
  - **Snapshop**: almacena el estado de un conjunto de unidades que puede ser restablecido más tarde (*.snapshot*)
  - etc.
- Las units pueden tener dos estados $\rightarrow$ **active** o **inactive**

Systemd *(cont.)*



- No todos los servicios que se inician en el booteo se utilizan:
  - impresora
  - servidor en el puerto 80
  - etc.
- Es un mecanismo de iniciación bajo demanda $\rightarrow$ podemos ofrecer una variedad de servicios sin que realmente esten iniciados
- Cuando el sockect recibe una conxión spawnea el servicio y le pasa el socket
- No hay necesidad de definir dependencias entre servicios $\rightarrow$ se inician todos los sockects en primer medida

Systemd - cgroups

- Permite organizar un grupo de procesos en forma jerárquica
- Agrupa conjunto de procesos relacionados (servidor apache con sus dependientes)
- Tareas que realiza:
  - Tracking grupal $\rightarrow$ no se utiliza el PID $\rightarrow$ doble *fork* no funciona para escapar de systemd
  - Limitar el uso de recursos
  - etc.

fstab

- Define qué particiones se montan al arranque
- Suconfiguración se encuentra en /etc/fstab:

```
$ cat /etc/fstab
# <file system> <mount point> <type> <options> <
    dump> <pass>
/dev/sda1 / ext4 errors=remount-ro 0 1

UUID=3FDE00F9523092AE /home/iso/datos ntfs user,
    auto,rw,exec,uid=1000,gid=1000,umask=000 0 2

/dev/sda2 none swap sw 0 0
```

- Opciones:
  - **user**: cualquier usuario puede montar la partición
  - **auto**: monta la partición al inicio
  - **ro**: read only, **rw**: read and write
  - etc.

- Al utilizar redirecciones mediante $>$ (destructiva):
  - Si el archivo de destino no existe, se lo crea
  - Si el archivo existe, se lo trunca y se escribe el nuevo contenido
- Al utilizar redirecciones mediante $>>$ (no destructiva):
  - Si el archivo de destino no existe, se lo crea
  - Si el archivo existe, se agrega la información al final

- El "|" nos permite comunicar dos procesos por medio de unpipe o tubería desde la *shell*
- El pipe conecta *stdout* (salida estándar) del primer comando con la *stdin* (entrada estándar) del segundo.
- Por ejemplo:

```
$ ls | more
```

  - Se ejecuta el comando *ls* y la salida del mismo, es enviada como entrada del comanda *more*

- Se pueden anidar tantos pipes como se deseen
- ¿Cómo haríamos si quisiéramos contar la cantidad de usuarios del sistema que en su nombre de usuario aparece una letra "a"?

```
$ cat /etc/passwd | cut -d: -f1 | grep a | wc -l
```

- El "|" nos permite comunicar dos procesos por medio de unpipe o tubería desde la *shell*
- El pipe conecta *stdout* (salida estándar) del primer comando con la *stdin* (entrada estándar) del segundo.
- Por ejemplo:

```
$ ls | more
```

  - Se ejecuta el comando *ls* y la salida del mismo, es enviada como entrada del comanda *more*

- Se pueden anidar tantos pipes como se deseen
- ¿Cómo haríamos si quisiéramos contar la cantidad de usuarios del sistema que en su nombre de usuario aparece una letra "a"?

```
$ cat /etc/passwd | cut -d: -f1 | grep a | wc -l
```

¿Preguntas?