espol Facultad de Ingeniería en Electricidad y Computación

Programación de Sistemas CCPG1051

Federico Domínguez, PhD.

Unidad 2 – Sesión 4: Comandos y procesos en UNIX/LINUX

Agenda

1. Comandos
2. Procesos
3. Redirección
4. Pipelines
5. Permisos
6. Scripts

Un **comando** es una directiva a un programa para que ejecute una tarea específica.

Tipos de comandos

- Un programa ejecutable
  - C, C++, script de Shell, Perl, etc.
- Un comando integrado al Shell
  - Lo provee el Shell que se este usando, en nuestro caso, bash
  - `shell builtins`
- Función del Shell: Integrados en el ambiente
- Alias
  - Comandos definidos por el usuario

command [-option(s)] [argument(s)]

Comandos

- `type` – Indicate how a command name is interpreted
- `which` – Display which executable program will be executed
- `help` – Get help for shell builtins
- `man` – Display a command's manual page
- `apropos` – Display a list of appropriate commands
- `info` – Display a command's info entry
- `whatis` – Display a very brief description of a command
- `alias` – Create an alias for a command

Comandos

Ejemplos comandos ejecutables:

- *grep*: busca patrones en un archivo
- *tail* y *head*: muestran las primeras y últimas líneas de un archivo
- *lsof*: lista archivos abiertos
- *top*: lista consumo de recursos (memoria, procesos) en tiempo real

Ejemplos de Shell builtins:

- *pwd*
- *cd*
- *type*
- *alias*

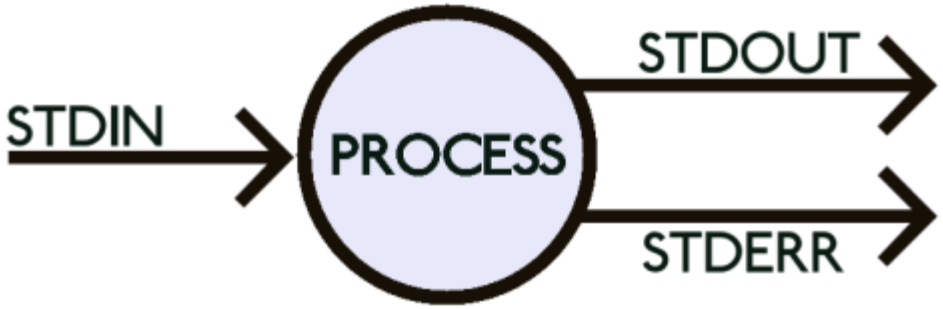
Comandos

Alias

- Creación rápida de comandos de usuario
- *alias name = 'string'*
- `alias foo='cd /usr; ls; cd'`
- El alias creado solo es valido durante la sesión, puede ser eliminado con `unalias`.

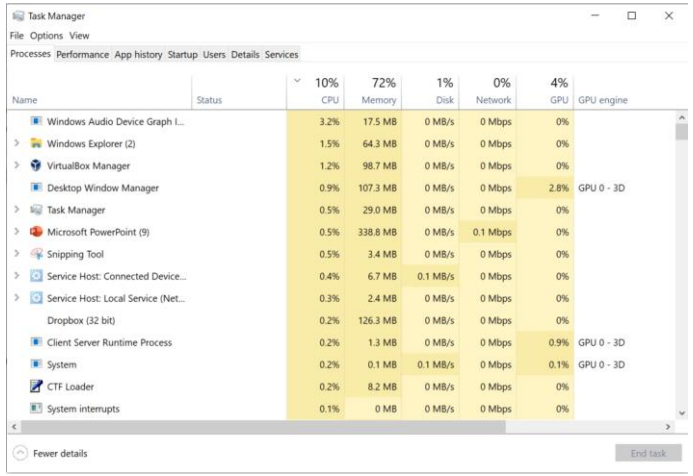
Un **proceso** es un programa en ejecución.

En sistemas UNIX/Linux un proceso tiene tres standard *streams*: **stdin, stdout, stderr**

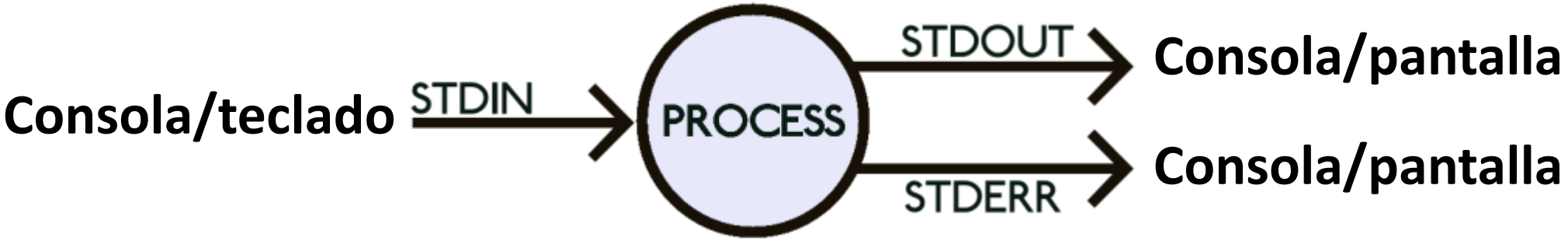



```
top - 16:16:17 up  5:26,  1 user,  load average: 0.06, 0.05, 0.00
Tasks: 198 total,   1 running, 197 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.8 us,  0.3 sy,  0.0 ni, 98.5 id,  0.3 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3936.2 total,   2052.9 free,    883.4 used,    999.8 buff/cache
MiB Swap:    448.5 total,    448.5 free,      0.0 used.   2786.5 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1393 federico  20   0 4282568 472404 135204 S   1.0  11.7   1:25.73 gnome-shell
     11 root      20   0       0      0      0 I   0.3   0.0   0:01.23 rcu_sched
   1216 federico  20   0  100012   2144   1764 S   0.3   0.1   0:39.69 VBoxClient
   5179 federico  20   0   20608   4016   3416 R   0.3   0.1   0:00.03 top
      1 root      20   0  102272  11760   8520 S   0.0   0.3   0:02.36 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-kblockd
      9 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     10 root      20   0       0      0      0 S   0.0   0.0   0:00.10 ksoftirqd/0
     12 root      rt   0       0      0      0 S   0.0   0.0   0:00.13 migration/0
     13 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/0
     14 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
```



Por defecto, las *streams* standard de un proceso están ligados al proceso que los inició: la **consola**.




```
chris@ubuntu: ~
chris@ubuntu:~$ bash --version
GNU bash, version 4.3.46(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
chris@ubuntu:~$
```

STDIN: Stream de **entrada** del proceso.
STDOUT: Stream de **salida** del proceso.
STDERR: Stream de mensajes de **error** del proceso.

Redirección

Redireccionar *stdout* a un archivo: >

Redireccionar *stdin* a un archivo: <

Redireccionar *stderr* a un archivo: 2>

Usar >, < y 2> crean un archivo o sobrescriben un archivo existente.

Usar >>, << y 2>> para agregar información al final del archivo.

Usar &> y &>> para redireccionar *stdout* y *stderr* a un mismo archivo.

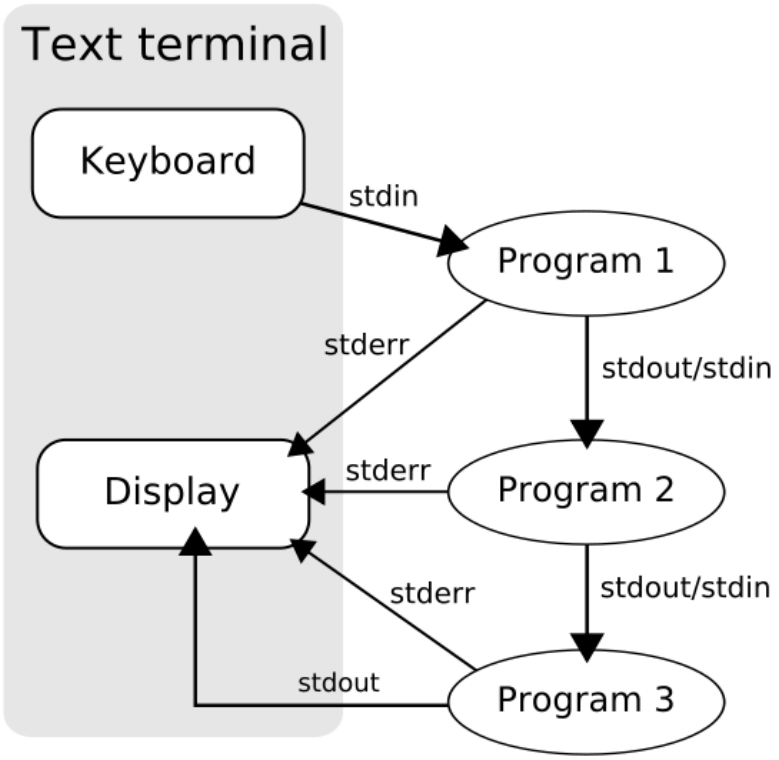
Para descartar la información generada por un comando, redireccionar al archivo */dev/null*

Pipelines

Una secuencia de procesos conectados por sus *standard streams*

Usa el carácter “|”

Ejemplo: `ls -l | grep log | less`



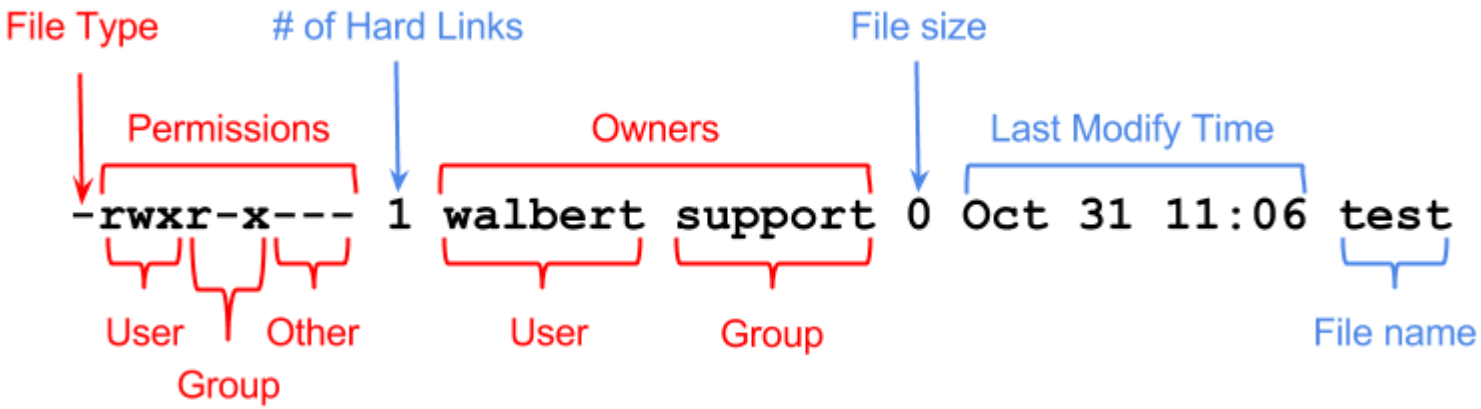
Demostración

Permisos

LINUX es multiusuario, por lo tanto es necesario que el sistema defina permisos de acceso.

- `id` – Display user identity
- `chmod` – Change a file's mode
- `umask` – Set the default file permissions
- `su` – Run a shell as another user
- `sudo` – Execute a command as another user
- `chown` – Change a file's owner
- `chgrp` – Change a file's group ownership
- `passwd` – Change a user's password

Permisos


File Type
# of Hard Links
File size
Permissions
Owners
Last Modify Time
-rwxr-x--- 1 walbert support 0 Oct 31 11:06 test
User
Other
Group
User
Group
File name

Permisos

Attribute	Files	Directories
r	Allows a file to be opened and read.	Allows a directory's contents to be listed if the execute attribute is also set.
w	Allows a file to be written to or truncated, however this attribute does not allow files to be renamed or deleted. The ability to delete or rename files is determined by directory attributes.	Allows files within a directory to be created, deleted, and renamed if the execute attribute is also set.
x	Allows a file to be treated as a program and executed. Program files written in scripting languages must also be set as readable to be executed.	Allows a directory to be entered, e.g., cd *directory*.

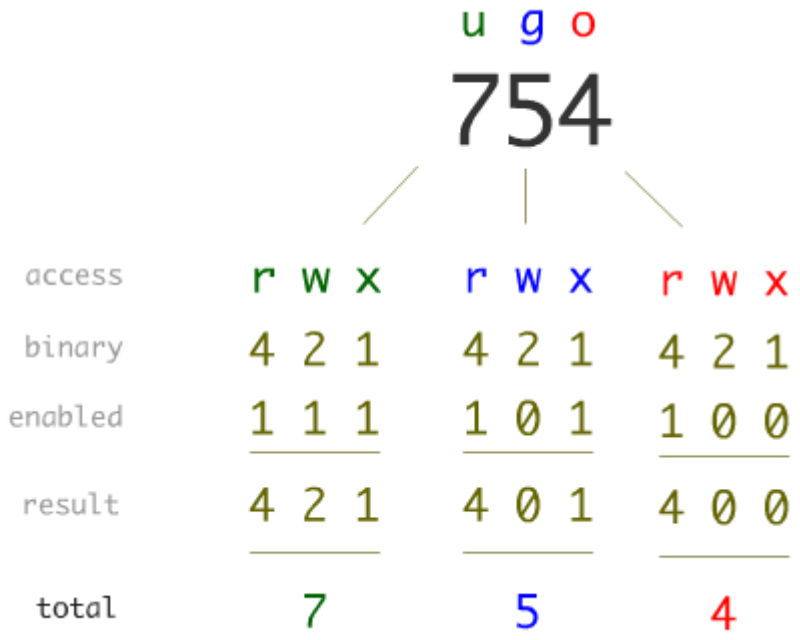
Owner	Group	World
rwx	rwx	rwx

Permisos

chmod: cambia los permisos en un archivo

Usa notación octal para especificar los permisos

Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx



Permisos

su: Permite ejecutar comandos como otro usuario

sudo: Permite ejecutar comandos como el usuario *root*

Permisos

chown: cambia el dueño de un archivo

chgrp: cambia el grupo dueño de un archivo

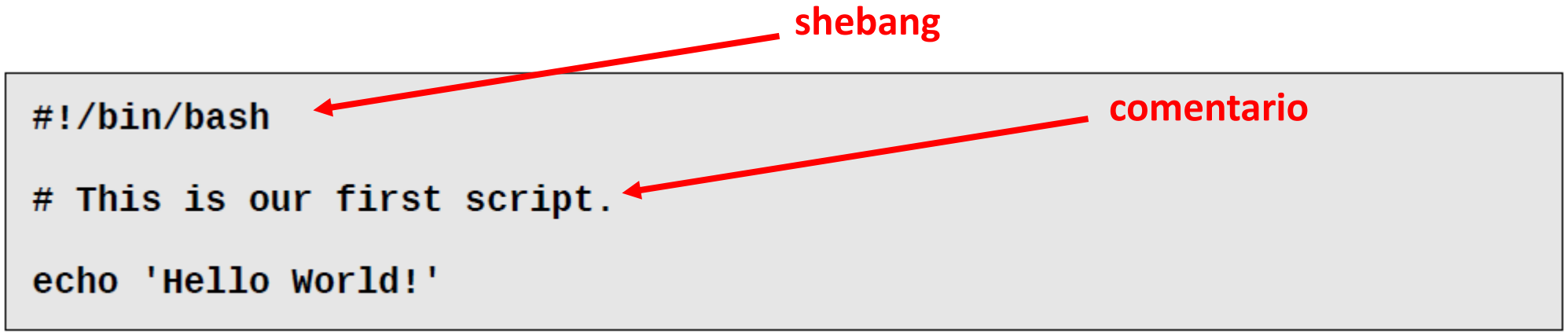
```
chown [owner][:[group]] file...
```

Argument	Results
`bob`	Changes the ownership of the file from its current owner to user `bob`.
`bob:users`	Changes the ownership of the file from its current owner to user `bob` and changes the file group owner to group `users`.
`:admins`	Changes the group owner to the group `admins`. The file owner is unchanged.
`bob:`	Change the file owner from the current owner to user `bob` and changes the group owner to the login group of user `bob`.

Scripts

Un archivo de texto el cuál contiene una lista de comandos a ejecutarse…

Debe tener permisos de ejecución.



Scripts

Debe de estar en el PATH para ser ejecutado directamente:

```
[me@linuxbox ~]$ hello_world
bash: hello_world: command not found
```

```
[me@linuxbox ~]$ ./hello_world
Hello World!
```

Demostración

Referencias

Comandos (Capítulo 5 TLCL)

Redireccionamiento y piplelines (Capítulo 6 TLCL)

Permisos (Capítulo 9 TLCL)

Scripts(Capítulo 24 TLCL)