



# Programación de Sistemas

## CCPG1008

---

Federico Domínguez, PhD.

Unidad 1 – Sesión 3: Redireccionamiento, Pipelines y Scripts

# Contenidos

---

1. Comandos (Capítulo 5 TLCL)
2. Redireccionamiento y pipelines (Capítulo 6 TLCL)
3. Permisos (Capítulo 9 TLCL)
4. Vi (Capítulo 12 TLCL)
5. Scripts(Capítulo 24 TLCL)
6. Práctica: Mi primer script

# Comandos

---

**Standard streams: stdin, stdout, stderr**



# Comandos

---

## 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

# 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

---

*grep*: busca patrones en un archivo

*tail* y *head*: muestran las primeras y últimas líneas de un archivo

# Comandos

---

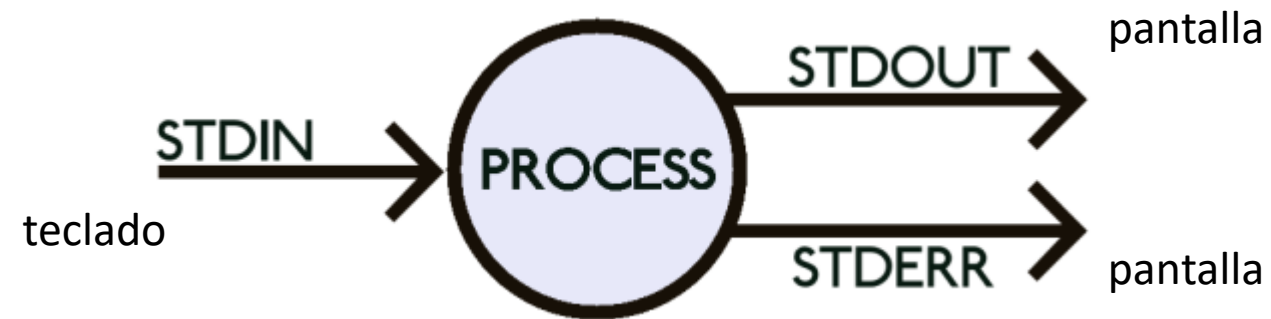
## Alias

- Creación rápida de comandos de usuario
- *alias name = 'string'*
- `alias foo='cd /usr; ls; cd'`

# Redirección

---

## Standard streams: stdin, stdout, stderr





# 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.

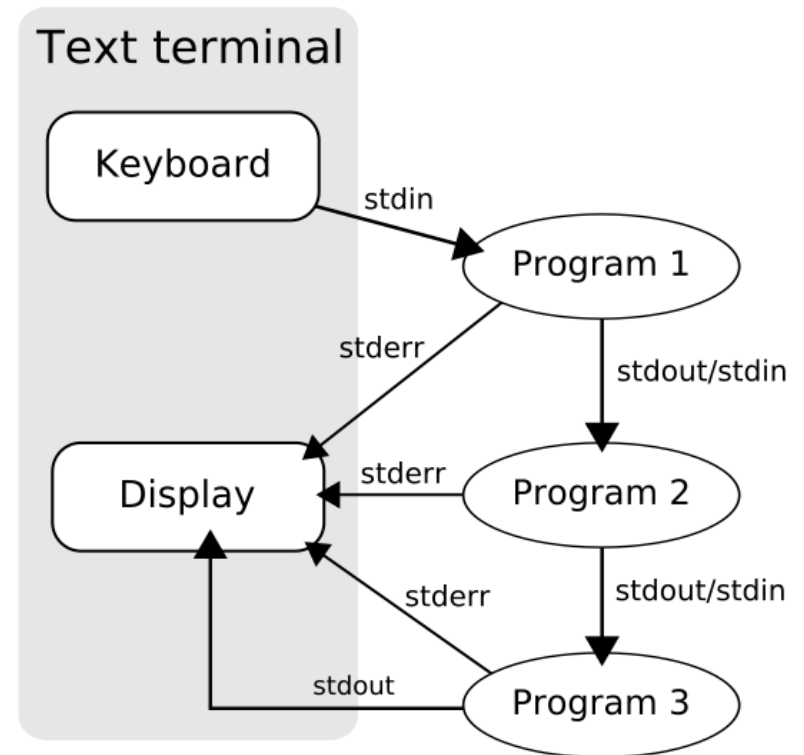
Para descartar la información generada por un comando, redireccionar al archivo */dev/null*

# Pipelines

Una secuencia de procesos conectados por sus *standard streams*

Usa el carácter “|”

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



# 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      Group      User      Group      File name

# Permisos

Attribute	Files	Directories	Owner	Group	World
r	Allows a file to be opened and read.	Allows a directory's contents to be listed if the execute attribute is also set.	rwX	rwX	rwX
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., <code>cd directory</code> .			

# 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	- w x
4	100	r - -
5	101	r - x
6	110	r w -
7	111	r w x

	u	g	o
	754		
access	r w x	r w x	r w x
binary	4 2 1	4 2 1	4 2 1
enabled	1 1 1	1 0 1	1 0 0
result	4 2 1	4 0 1	4 0 0
total	7	5	4

# 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.



# vi: Editor POSIX de sistemas UNIX/LINUX

---

vi es un editor de texto incluido por defecto en casi todas las distribuciones de Linux.

- Siempre disponible
- Rápido y de bajos recursos
- Muy popular en la comunidad código abierto

Otra opción: nano

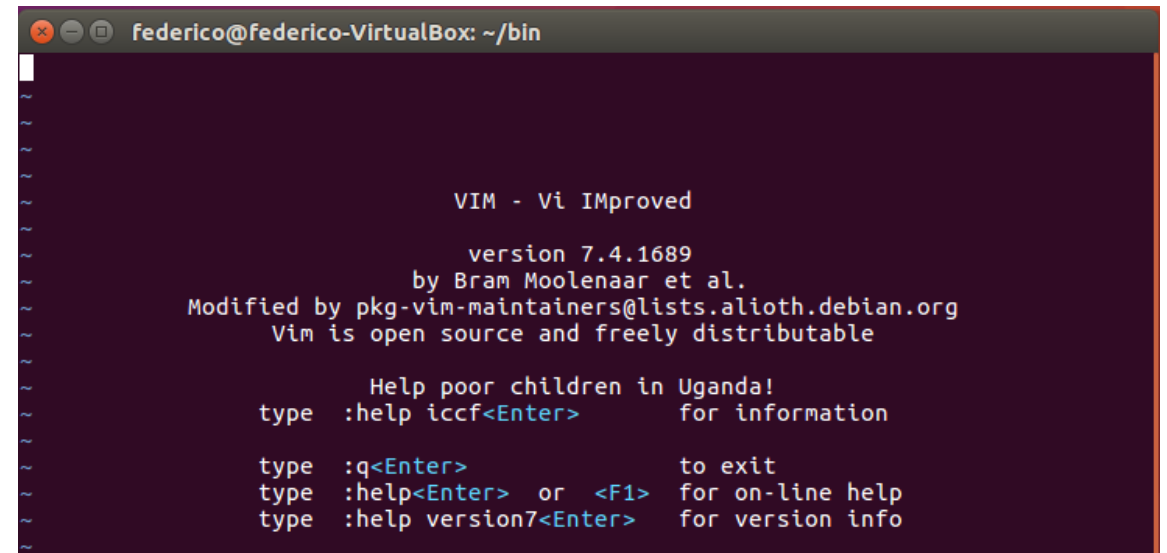
# vim: versión mejorada de vi

---

Incluida en la mayoría de distribuciones

Usualmente, vi es un alias a una versión de vim con capacidades mínimas, recomendado instalar vim:

- Ubuntu: `sudo apt-get install vim`
- Centos/Fedora: `yum install vim`

A screenshot of a terminal window titled "federico@federico-VirtualBox: ~/bin". The terminal displays the Vim startup screen with the following text: "VIM - Vi IMproved", "version 7.4.1689", "by Bram Moolenaar et al.", "Modified by pkg-vim-maintainers@lists.alioth.debian.org", "Vim is open source and freely distributable", "Help poor children in Uganda!", "type :help iccf<Enter> for information", "type :q<Enter> to exit", "type :help<Enter> or <F1> for on-line help", "type :help version7<Enter> for version info". The terminal has a dark purple background and a light blue cursor on the first line.

```
federico@federico-VirtualBox: ~/bin
VIM - Vi IMproved
      version 7.4.1689
      by Bram Moolenaar et al.
Modified by pkg-vim-maintainers@lists.alioth.debian.org
Vim is open source and freely distributable

  Help poor children in Uganda!
type  :help iccf<Enter>      for information

type  :q<Enter>              to exit
type  :help<Enter> or <F1>   for on-line help
type  :help version7<Enter> for version info
```

# vi tiene varios modos de trabajo

Empieza en modo de comando

Modo de inserción (a,A,i,o,O)

Modo de ex command ( : )

a	Enter insert mode. Characters typed are inserted into the file just after the current character.
A	Enter insert mode. Characters typed are inserted into the file at the end of the current line.
dd	Delete the current line.
h	Same as left arrow key.
i	Enter insert mode. Characters typed are inserted into the file just before the current character.
j	Same as down arrow key.
k	Same as up arrow key.
l	Same as right arrow key.
o	Open a new line beneath the current line, and enter insert mode.
O	Open a new line above the current line, and enter insert mode.
p	Put back most recently deleted text.
r	Replace the current character with the next character typed.
u	Undo previous change.
x	Delete the current character.
ctl-B	Backward screen.
ctl-D	Forward half-screen.
ctl-F	Forward screen.
ctl-U	Backward half-screen.
:q!	Exit without writing the file.
:wq	Exit, writing the file.
:w	Write the file, but don't exit.
.	Repeat previous command.
/	Search for a string. Type the string to search for after typing /, followed by carriage return. / immediately followed by return repeats the previous search.
?	Same as /, but search backward.
csc	The escape character is used to exit insert mode. Be sure to exit insert mode before trying to issue commands.

# Scripts

---

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

Debe de tener permisos de ejecución

```
#!/bin/bash  
  
# This is our first script.  
  
echo 'Hello World!'
```

# 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!
```

# Para la próxima clase...

---

Introducción a C ...