# Utilizando la Termial

*Arturo Rivas Rojas*

*17/12/2018*

# Contents

# Chapter 1

# Manual

- **man** [comando] : muestra el manual del comando.
    - **Enter** : avanza una linea
    - **space** : avanza una pagina
    - **b** : retorcede una pagina

## 1.1 Herramientas básicas

- **open_command** [archivo] : abrir archivo con el programa por defecto.
- **more** [archivo] : nos permite paginar un archivo, nos desplazamos con space una pagina y con enter una linea.
- **cat** [archivo] : concatena e imprime todo el archivo
- [comando|programa] **>** [archivo] : guarda la salida del comando o programa en el archivo dado sobreescribiendolo o creandolo.
- [comando|programa] » [archivo] : guarda la salida del comando o programa en el archivo dado, agregandolo a lo que ya contenga el archivo.
- [comando|programa] **<** [archivo] : el contenido del archivo es la entrada del comando o programa.
- **Ctrl+c** : termina el proceso.
- **tail** [archivo] : muestra las ultimas lineas del archivo.
    - **-#** : define el numero de lineas que enseñara.
    - **-f** : la salida se recresca si el archivo tiene cambios.
- **wc** : muestra la cuenta de los saltos de linea, palabras y bytes.
- **-l** : cuenta lineas.

# Chapter 2

# Arbol de directorios

## 2.1 Navegación

- `.` : simboliza al directorio **actual**.
- `..` : simboliza al directorio **padre**.
- `/` : simboliza al directorio **raiz**.
- `~` : simboliza al directorio **HOME**.
- `*` : comodin, lo que sea.
- **ls** : listar archivos y directorios
    - **-l** : listar hacia abajo y ademas mostrar información extra de los archivos, permisos, owner, tamaño, ultima modificación y nombre.
    - **-h** : mostrar la información de una manera mas legible
    - **-a** : mostrar archivos ocultos.

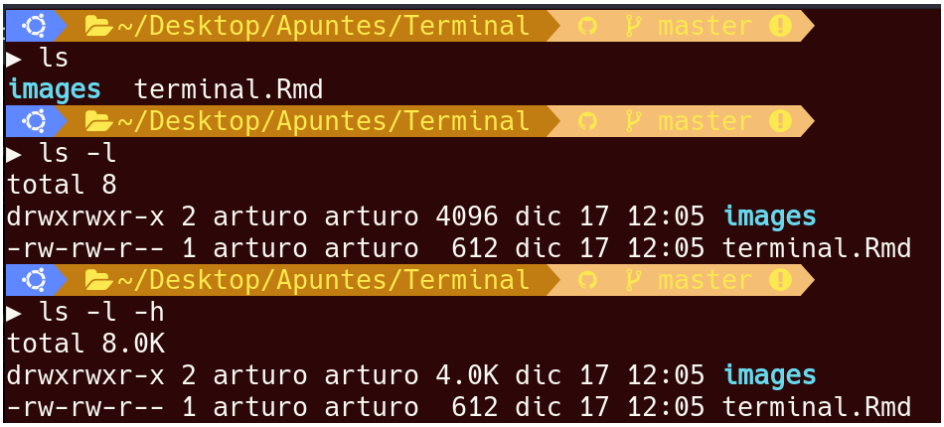


Figure 2.1: comando ls y sus banderas

- **pwd** : nos muestra el "Working directory" el directorio en el que estamos actualmente.
- **cd** [ruta]: change directory nos permite navegar entre los directorios. Acepta tanto rutas absolutas como relativas. Si no se le pasa ninguna ruta lleva a **HOME**
- **clear** : limpia la terminal, se puede ejecutar con **Ctrl+l**.

## 2.2 Manipular

- **mkdir** [nombre] : crea el directorio.
- **touch** [nombre] : crea un archivo y si ya existe modifica su fecha de creación.
- **mv** [archvivo|directorio] [ruta/nombre_nuevo] : mueve el archivo a la ruta dada y lo renombra. Si solo se da el nombre nuevo sirve para renombrar el archivo.
- **rm** [archvivo|directorio] : elimina el archivo.

    - **-r** : borrar de manera recursiva, es decir directorios.
    - **-f** : fuerza la operación.

- **cp** [archivo|directorio] [ruta] : copia el archivo a la ruta dada.

### 2.2.1 Pila de directorios

- **pushd** [directorio] : ingresar el directorio actual a la pila y moverse al directorio dado.
- **popd** [directorio] : sacar un directorio de la pila y moverse al directorio.
- **dirs** : vizualizar la pila de directorios.

    - **-v** : vizualizar con una numeración.
    - **-c** : borrar la pila de directorios.

Ejemplo: supongamos que tenemos que estar constantemente moviendonos entre directorios que estan en rutas “distantes” podemos hacer lo siguiente.

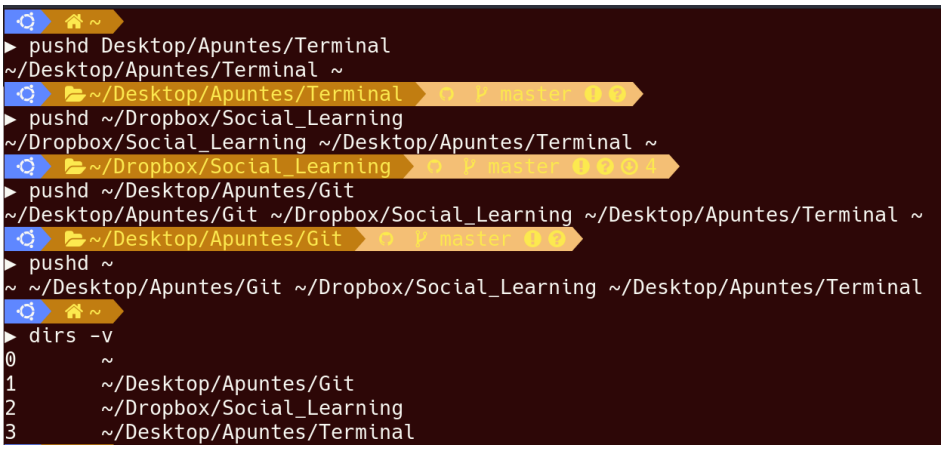


Figure 2.2: agregando los directorios a la pila

Una vez que los directorios estan agregados en la pila podemos navegar a ellos utilizando los numeros como alias.

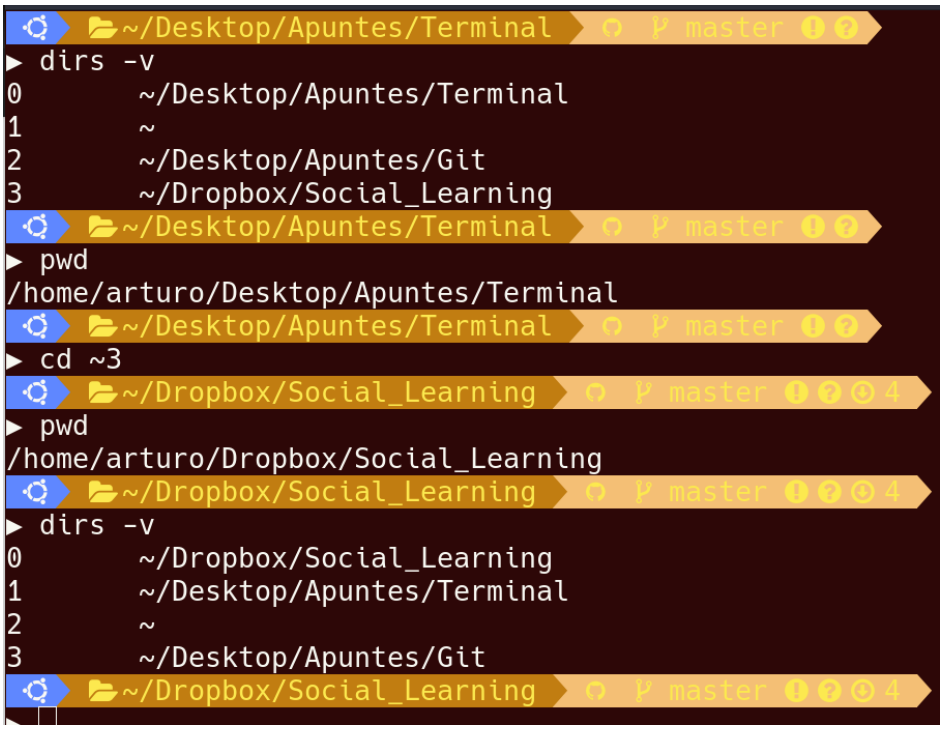


Figure 2.3: navegando con alias

Cabe recalcar que la pila y por lo mismo los alias se van modificando según como naveguemos. Siempre siendo el 0 el directorio actual.

### 2.2.2 Links Simbolicos

- **ln -s** [archivo objetivo] [link] : genera el archivo link que es una link simbolico que apunta al archivo objetivo, es decir que si lo eliminamos no se eliminará el archivo al que apunta.

  - para borrar basta con rm

# Chapter 3

# Variables y Entorno

- **which** [comando] : nos dice en donde esta el ejecutable del comando.
- **echo** [$variable] : imprime el contenido de la variable.

Para correr un ejecutable debe estar en la variable PATH.

- **alias** [trigger] [‘comandos’] : encapsula el comando o comandos con sus banderas en un mismo disparador. Digamos que crea un nuevo comando.

# Chapter 4

# Streams

Todo programa hace uso de streams, tienen flujos de entrada y de salida. Los 3 streams mas importantes son los standard input, standard output y el standard error.

- **standard input** : ingreso de datos.
- **standard output** : salida de datos, todo lo que si deberia de pasar.
- **standard error** : notificación de errores, lo que no deberia de pasar.

En la terminal la salida es la variable 1 y el error es la varibale 2 podemos mandar estas variables a otros lugares ademas de la pantalla. Para enviarlos al mismo lugar seria usando apuntadores 2>&1 y asi decimos que mande la variable 2 al mismo lugar que manda la variable 1.

# Chapter 5

# Procesos

Todo lo que se ejecute en la computadora es un proceso.

- **top** : lista todos los procesos que estan corriendo en ese momento. Nos da información como el identificador del proceso, el nombre, la memoria que esta gastando, el porcentaje del CPU que esta ocupando, entre otras cosas.
- **kill** [PID] : mata la proceso.
- **-9** : mata sin preguntar y sin importar lo que este haciendo el proceso.
- [comando|programa] **&** : corre el proceso en el background. y nos regresa el PID del proceso para tener cierto control.
- [comando|programa 1] **;** [comando|programa 2] : primero se ejecuta el programa 1 y hasta que este termine incia el 2. Se pueden secuenciar n procesos todos con un **;** intermedio.
- **ps -wS** : lista todos los procesos que se estan ejecutando en el momento.

# Chapter 6

# Pipe (|)

Concatena comandos, es decir el standard output del primer comando lo manda como standard input del segundo.

- [comando|programa 1] **|** [comando|programa 2] : concatena los comandos.

# Chapter 7

# Power Tools

## 7.1 Busqueda

- **grep** [archivo]: busca en el archivo las ocurrencias de una regex.
    - **-e** [regex] : definir la regex.
    - **-r** [directorio] : buscar en los archivos dentro de los directorios.
    - **-n** : nos permite ver el numero de linea donde encontro la regex.
    - **-v** [regex] : no imprime esa regex.
    - **-i** [regex] : busqueda no sensible a mayusculas.
- **find** [directorio de inicio] : busca en la metadata de los archivos, no dentro de ellos a diferencia de **grep**.
    - **-name** [“cadena”] : busca en el nombre de archivo la cadena.
    - **-type** [tipo] : f es archivos, d directorio, entre otras opciones.
- **du** : disk usage nos dice cuanta memoria ocupa cada nodo hoja directorio actual
    - **-h** : human readable.
    - **-d** [#]: niveles de recursividad.

## 7.2 Variadas

- **curl** [URL]: comando que emula un browser y nos regresa la respuesta de la URL.
    - **-o** [archivo] : crea o sobreescribe el archivo con lo que se obtuvo de la URL.
- **zip** [nombre del archivo].zip [archivos] : comprime los archivos dados en un archivo.zip.
- **unzip** [archivo.zip] : descomprime el archivo .zip
    - **-vl** : enlista el contenido del zip con información extra.
- **tar cfz** [nombre del archivo].tar.gz [archivos]: comprime los archivos a un tar.gz utiliza zip como algoritmo de compresión.
- **tar xfz** [archivo.tar.gz]: descomprime el archivo tar.gz.
- **awk** ‘{codigo a ejecutar}’: aplicar codigo en C al flujo de entrada.
    - **-F**[“cadena”] : split por la cadena.

## 7.3 Crontab (Automatizando tareas)

- crontab **-l** : lista las tareas que tenemos programadas/instaladas.

Table 7.1: Formato de salida de crontab -l

| Minuto -> 0-59 | Hora -> 0-23 | Dia-Mes -> 1-31 | Mes -> 1-12 | Dia-Semana -> 0-7 | Ejecutable -> ruta |
|---|---|---|---|---|---|

- crontab **-l** : lista las tareas que tenemos programadas/instaladas.
- crontab **-e** : edita o crea el archivo de crontab que basta con llenar 6 columans para automatizar una tarea.

El formato de los valores de las primeras 5 columanas no aceptan espacios y son los siguientes:

Table 7.2: Formatos de entra validos

| Nombre | Formato | Significado |
|---|---|---|
| Unico | 1 | Solo se ejecutara cuando el valor sea exactamente 1. |
| Multiples | 1,13,34 | Se ejecutará cunado el valor sea 1, 13 o 34. |
| Multiplos | ‘*‘/5 | Se ejecutara en las horas multiplos de 5, cada 5 horas. |
| Rango | 1-7 | Se ejecutara de la hora 1 a la 7 |
| Comodin | ‘*‘ | Se ejecutara en cada valor |

# Chapter 8

# Usuarios y Permisos

- **whoami** : regresa el usaurio con el qeu esta logeado.

Cuando se hace un ls -lh se imprime algo como lo siguiente:

Table 8.1: Encabezados del ls -lh

| Permisos | Usuario | Grupo | Información del archivo |
|---|---|---|---|

```
drwxrwxr-x 2 arturo arturo 4.0K dic 17 13:17 images
-rw-rw-r-- 1 arturo arturo 384K dic 17 20:16 terminal.pdf
-rw-rw-r-- 1 arturo arturo 9.1K dic 17 22:29 terminal.Rmd
```

Figure 8.1: Retorno del comando ls -lh

Concentrandonos en lso permisos. Por ejemplo:

```
-rw-rw-r--
```

Figure 8.2: ejemplo de permisos

El primer caracter de los permisos nos dice:

Table 8.2: Tipos de ficheros

| Caracter | significado |
|---|---|
| - | archivo |
| d | directorio |
| l | link |

Los siguietes caracteres los agruparesmo de 3 en 3. Cada grupo nos dira los permisos del: 1. dueño. 2. grupo. 3. quien sea.

Los permisos se leén de la siguiente mandera:

Table 8.3: Nomenclatura de los permisos

| Numero | Binario | Caracteres | Permisos |
|---|---|---|---|
| 0 | 000 | — | ninguno |
| 1 | 001 | -x | Ejecutar |
| 2 | 010 | -w- | Escribir |
| 3 | 011 | -wx | Ejecutar y Escribir |
| 4 | 100 | r- | Leer |
| 5 | 101 | r-x | Ejecutar y Leer |
| 6 | 110 | rw- | Leer y Escribir |
| 7 | 111 | rwx | Ejecutar, Leer y Escribir |

## 8.1 Cambiar permisos

- **chmod** [dueño][grupo][quien sea] : chmod recibe un numero de 3 cifras cada cifra es el permiso correspondiente al dueño, grupo o quien sea segun la tabla de nomenclatura.

# Chapter 9

# Scripts

Cuando se agrega un #! en la primera linea esto no es un comentario sino que son instrucciones para la ejecución de ese archivo.

```
#!/bin/zsh

Rscript -e "rmarkdown::render(\"$1\", clean=TRUE)"

```

Figure 9.1: Ejemplo de script