Sistemas Operativos
Tarea #1
Shell Linux

Escuela de Ingeniería Civil Informática
Universidad de Valparaíso

Nombre: Pedro Pablo López Mena

Asignatura: Teoría de Sistemas Operativos.

Profesor: Gabriel Astudillo Muñoz.

1 Objetivo

En esta tarea se verá la programación a nivel de comandos en un ambiente Unix. El objetivo es familiarizarse con los comandos más usados y conocer algunas aspectos de la programación shell.

2 Instrucciones

Su trabajo deberá ser entregado en el servidor en su cuenta en GitHub.com. El nombre del repositorio debe ser SSOO-tarea01

```
+ /SSOO-tarea01
      - README.md
      - Makefile
      - src/
```

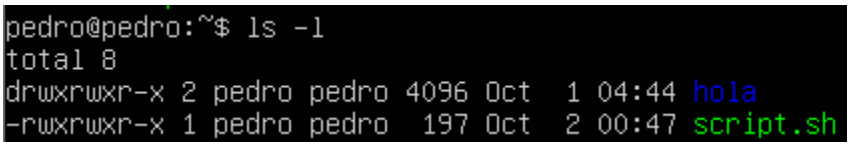
El archivo **`README.md`** es un archivo en markdown, donde se debe explicar el diseño de su solución, funciones utilizadas, etc. Además, debe especificar el nombre del autor y su correo institucional. En el directorio **`src/`** están los códigos fuentes de su proyecto. Además, debe de entregar un documento llamado "ApellidoPaterno_Nombre.pdf" evidenciando el trabajo realizado, de **NO** cumplir con el método de entrega solicitado será calificado con nota mínima.

3 Trabajo a realizar

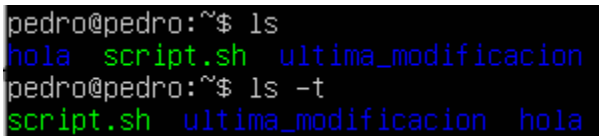
a) Estudie y explique para qué sirven los comandos **ls**, **cat**, **chmod**, **echo**, **grep**, **cp**, **mv**, **rm** y **wc**. Dé ejemplos de uso. Para el comando **ls** averigüe para qué sirven las opciones -l, -t y -a. De ejemplo de uso para cada uno de esas opciones y la combinación de ellas. Ayuda: use el comando man, ej. **man ls**.

- ls: el comando ls sirve para mostrar los directorios y archivos de una ruta determinada.

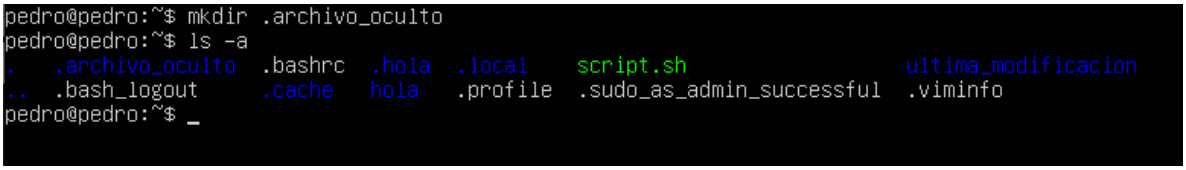
- ls -l: muestra un listado en el formato largo, con información de permisos, número de enlaces asociados al archivo, usuario, grupo, tamaño y fecha de última modificación además del nombre.



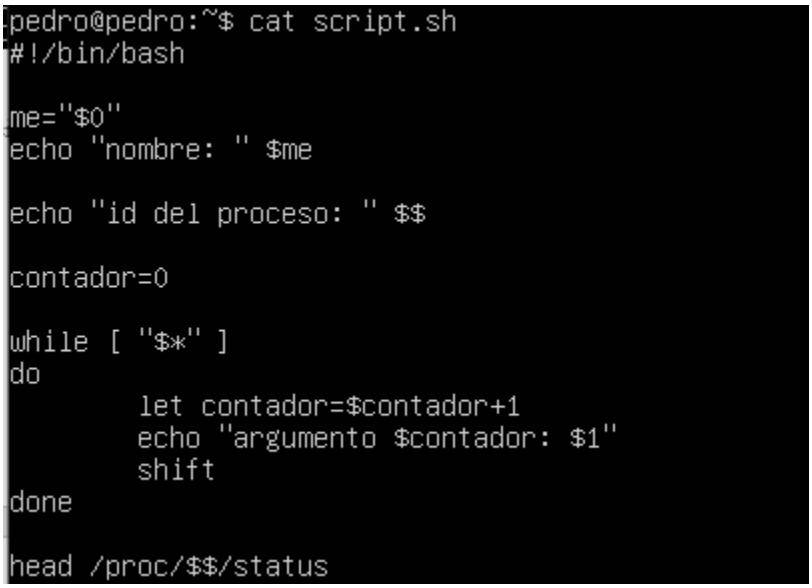
- ls -t: muestra ordenada por la fecha de última modificación.



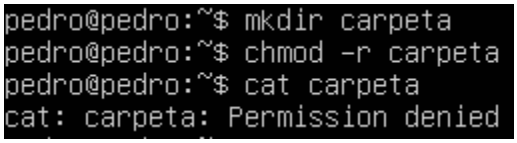
- ls -a: muestra los archivos ocultos.



- cat: el comando cat se utiliza para mostrar el contenido de un archivo de texto.



- chmod: el comando se utiliza para cambiar los permisos de un archivo o un determinado directorio, puede variar entre escritura, lectura, ambos u otros.



- echo: permite imprimir información en el Shell.

```
pedro@pedro:~$ echo HOLA MUNDO
HOLA MUNDO
```

- grep: permite filtrar por determinados parámetros información.

```
pedro@pedro:~$ ls
carpeta  hola  script.sh  ultima_modificacion
pedro@pedro:~$ ls | grep script.sh
script.sh
```

- cp: Este comando sirve para copiar archivos y directorios dentro del sistema de archivos

```
pedro@pedro:~$ cp script.sh copia
pedro@pedro:~$ ls
carpeta  copia  hola  script.sh  ultima_modificacion
```

- mv: permite mover archivos o directorios a otras rutas.

```
pedro@pedro:~$ ls
carpeta  copia  hola  script.sh  ultima_modificacion
pedro@pedro:~$ cd hola
pedro@pedro:~/hola$ ls
pedro@pedro:~/hola$ cd ..
pedro@pedro:~$ mv script.sh /home/pedro/hola
pedro@pedro:~$ ls
carpeta  copia  hola  ultima_modificacion
pedro@pedro:~$ cd hola
pedro@pedro:~/hola$ ls
script.sh
```

- rm: este comando se utiliza para eliminar archivos o directorios.

```
pedro@pedro:~$ mkdir archivo_eliminable
pedro@pedro:~$ ls
archivo_eliminable  carpeta  copia  hola  script.sh  ultima_modificacion
pedro@pedro:~$ rm -d archivo_eliminable/
pedro@pedro:~$ ls
carpeta  copia  hola  script.sh  ultima_modificacion
```

- wc: este comando permite realizar diferentes conteos desde la entrada estándar, ya sea de palabras, caracteres o saltos de líneas. Por ejemplo, para ver el número de líneas hacemos:

```
pedro@pedro:~$ wc -l archivo.txt
6 archivo.txt
```

Ejemplos de uso combinado:

- filtrar la búsqueda de archivos o directorios por letra y nombre:

```
pedro@pedro:~$ ls
archivo_prueba  archivo.txt  carpeta  copia  directorio  hola  script.sh  ultima_modificacion
pedro@pedro:~$  ls | grep a
archivo_prueba
archivo.txt
carpeta
copia
hola
ultima_modificacion
pedro@pedro:~$ ls | grep archivo
archivo_prueba
archivo.txt
```

- mostrar todas las líneas con la letra “a” dentro de un archivo de texto.

```
pedro@pedro:~$ cat script.sh  | grep a
#!/bin/bash
contador=0
        let contador=$contador+1
        echo "argumento $contador: $1"
head /proc/$$/status
pedro@pedro:~$
```

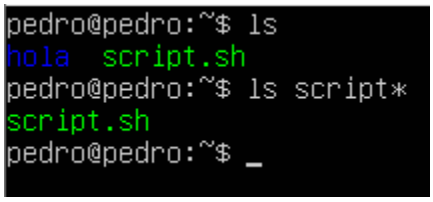
b) Explique qué son los metacaracteres y dé ejemplos de uso de ellos.

Los metacaracteres son caracteres no alfabéticos que poseen un significado especial en las expresiones regulares, en GNU+Linux son utilizados para describir otros caracteres y nos permiten representar conjuntos de caracteres convenientemente para buscarlos y transformarlos. Algunos ejemplos serían:

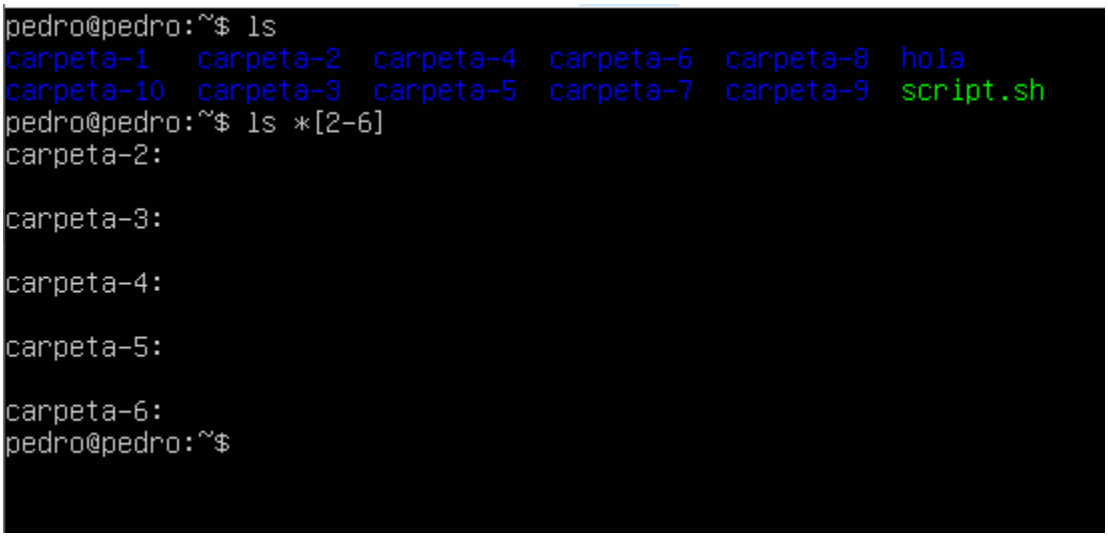
- “*z” cualquier conjunto que acabe en z.

```
pedro@pedro:~$ ls
hola  script.sh
pedro@pedro:~$ ls *sh
script.sh
pedro@pedro:~$ _
```

- “z*” cualquier conjunto que comience con z.



- []: Se utiliza para especificar una lista de caracteres o un rango. Si se quiere especificar un rango se debe ubicar el carácter “-“ entre el primer y el último carácter del rango.



c) Explique en que conciste la expanción por paréntesis de conjunto (Brace Expansion). Con esta herramienta, resuelve el siguiente problema:

En un directorio, se quieren crear subdirectorios para que almacenen respaldos diarios de todo un año. Debe tener la siguiente estructura :

```
directorio_actual/
      + 2021-01-01/
      + 2021-01-02/
      + 2021-01-03/
      . . .
      + 2021-09-18/
      . . .
      + 2021-10-18/
      . . .
      + 2021-12-31/
```

Suponga que todos los meses tienen 31 días. Debe ejecutar UN sólo comando para crear la estructura de directorios solicitada.

El brace expansion consiste en un mecanismo a través del cual se crean listas de cadenas para ser utilizados en scripts o en comandos en GNU+Linux.

El comando utilizado para crear la estructura de carpetas solicitada es el siguiente:

```
1   mkdir 2021-{1..12}-{1..31}
```

d) La interconexión de comandos a través de *pipes* permite construir, de forma muy simple, nuevas herramientas. Como ejemplo considere los comandos ls y wc, que interconectados permiten contar archivos del directorio actual: ls | wc -l. Mediante el uso de pipes resuelva:

1. Mediante grep, encontrar archivos cuyo nombre contenga el carácter i en el directorio /bin.

```
pedro@pedro:~$ ls /bin |grep i _
```

2. Contar los archivos con una secuencia de permisos r-x en los directorios /bin y /usr/bin.

```
pedro@pedro:~$ ls -l /usr/bin | grep r-x | wc -l
805
pedro@pedro:~$ ls -l /usr/ | grep r-x | wc -l
13
```

e) Las variables de ambiente definen aspectos del entorno de programación, y los comandos set y echo (mediante el metacaracter $) permiten ver su contenido.

1. Investigue el uso de las variables HOME, SHELL, PATH, y PWD. ¿Cómo se puede visualizar su contenido?

   El uso de las variables pedidas es el siguiente:

   1) HOME: la variable de entorno HOME identifica el directorio principal del usuario actual.
   2) SHELL: describe el Shell que ejecutará cualquier comando que sea ingresado. En la mayoría de los casos será bash de forma predeterminada, pero se pueden establecer otros valores si así se desea.
   3) PATH: establece una lista de directorios del sistema que se comprobarán cuando el usuario utilice comandos.
   4) PWD: establece el directorio de trabajo actual.

Para poder visualizar su contenido podemos usar el comando “printenv” acompañado del nombre de la variable de entorno.

```
pedro@pedro:~$ printenv HOME
/home/pedro
pedro@pedro:~$
```

2. Investigue cómo se puede modificar el valor de una variable de ambiente (ayuda: investigue el operador ’=’ y export).

El valor de una variable de ambiente se puede modificar a través del comando “export NOMBRE-VARIABLE” el cual exporta la variable. Para modificarla se utiliza la sintaxis “export NOMBRE-VARIABLE=valor”. El comando export le indica al sistema que tal variable se está utilizando como variable de ambiente, de no

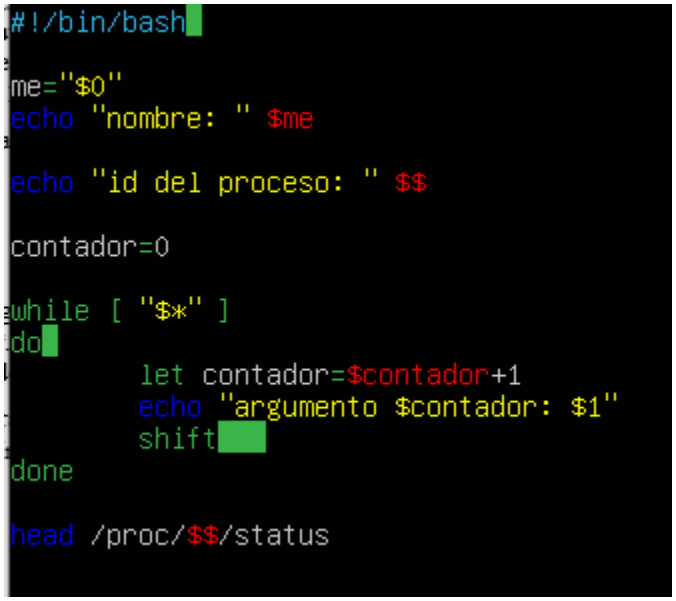
utilizarse para cambiar un valor se entendería que la variable no es de ambiente. El operador "=" se utiliza para asignar un valor a la variable de entorno, como en casi cualquier lenguaje de programación.

3. Ejecutando el comando *bash* dentro de la línea de comandos se crea un sub-shell, ¿Cómo afecta esto las variables de ambiente? ¿Cuál es el efecto de export? Explique y dé ejemplos.

El utilizar el comando bash crea una subshell, está subshell es una Shell hija de la Shell original en la que fue llamado el comando, por tal está hereda todas las variables de entorno de la Shell padre. Dentro de está subshell se puede variar las variables de ambiente sin afectar las del padre, editándolas. Sin embargo si se utiliza el comando export para editar las variables de entorno, entonces se editarán tanto en la Shell padre como en las hijas.

f) El intérprete de comandos bash es también un lenguaje de programación, con estamentos de control de flujo como for, while, etc. El código fuente a menudo se llama *script*. Si el archivo que contiene el script se llama **ejemplo.sh**, el comando **chmod +x ejemplo.sh**, le da permisos de ejecución al archivo **ejemplo.sh**.

Implementar un script BASH que liste cada argumento de entrada por separado, incluyendo el nombre del script. Además, debe mostrar su PID y mostrar las 10 primeras líneas del archivo /proc/PID/status.



```
#!/bin/bash

me="$0"
echo "nombre: " $me

echo "id del proceso: " $$

contador=0

while [ "$*" ]
do
        let contador=$contador+1
        echo "argumento $contador: $1"
        shift
done

head /proc/$$/status
```