Alumno: Bastián Fernando Contreras Wasilkowski ***Run:*** 20483921-2

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.

Respuesta:

Comandos:

- ls [-a, -t y -l]: El comando 'ls' enseña una lista del contenido del directorio actual, al agregar el argumento '-a' o '--all' se mostrarán además todos los archivos ocultos, incluyendo aquellos que comienzan con un punto. Con el argumento '-t' ordena la lista por fecha de última modificación, siendo el primero el más reciente. Por último, al utilizar '-l' la lista se mostrará en formato largo, es decir, la lista, además del nombre de cada archivo, mostrará el tipo de fichero (directorio/archivo/socket/etc..), el usuario propietario, el grupo propietario, los permisos, el tamaño, los enlaces duros (cantidad de archivos o entradas apuntando a ese elemento), y el tiempo y fecha de la última modificación.
  Si se utiliza el comando de la forma 'ls -a -t -l' o de la forma 'ls -atl' todas las definiciones anteriores se combinarán, es decir, se mostrará un listado de todos los archivos o directorios; incluyendo ocultos; ordenados por fecha de última modificación y con todo el detalle de estos.

- cat: El comando 'cat' lo que hace es mostrar todo el contenido de un fichero o ficheros, de tal manera que "concatena" todas las líneas del texto y las muestra en pantalla. El comando funciona solo con caracteres ASCII, de forma que si se intenta usar el comando con algún archivo que no sea de texto plano no lo mostrará correctamente. Se utiliza de la forma 'cat nombre_fichero', o también se puede usar después del comando, por ejemplo: 'man ls | cat' así el comando 'man ls' que muestra el manual de instrucciones de ls por página, mostrará todas las páginas de una sola vez.

- chmod: chmod es un comando utilizado para cambiar los permisos de un archivo, utilizado de la forma 'chmod opciones permisos nombre_archivo', por ejemplo, si poseo el archivo 'test.txt' y deseo modificar el permiso del usuario para que solo lo pueda leer escribo lo siguiente: 'chmod u=r test.txt', sea u=user y r=read'.

- echo: El comando echo simplemente toma el texto ingresado y lo reproduce por el terminal. Ejemplo: 'echo hola' mostraría 'hola' por el terminal. También se puede utilizar para escribir en un archivo de texto el texto ingresado de la forma 'echo test del comando echo > test.txt', este comando guardaría la frase 'test del comando echo' en un archivo llamado 'test.txt'.

- grep: El comando 'grep' filtra el texto dado como parámetro y entrega las filas que lo contienen en el texto señalado, por ejemplo, si escribo 'ifconfig | grep netmask' mostrará solo las líneas que contienen la palabra 'netmask' en la ejecución del comando ifconfig', el cual permite ver la configuración actual de las redes del sistema.

- cp: El comando cp copia los archivos y/o directorios de un lugar a otro, por ejemplo: 'cp test.txt .local/', al ejecutar este comando se copiará el archivo de texto test.txt en el directorio '.local/', aunque sin los permisos. Si se requiere copiar el archivo con los permisos incluidos se necesita agregar el argumento '-p' al final del comando.

- mv: El comando 'mv' puede ser utilizado tanto para mover o renombrar archivos, de la forma 'mv test.txt directorio/' o 'mv test.txt test2.txt' respectivamente.

- rm: El comando 'rm' se utiliza para remover archivos o directorios, simplemente colocando 'rm nombre' para remover un archivo, y 'rm directorio -r' para remover un directorio y todo su contenido.

- wc: Al utilizar el comando se muestran la cantidad de líneas, palabras, y caracteres del documento. Ejemplo: Simulando que 'test.txt' posea la línea "This is a test." al utilizar el comando de la siguiente manera: 'wc test.txt' la salida seria '1 4 15 test.txt', indicando una línea, cuatro palabras y 16 caracteres.

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

Respuesta:

Los metacaracteres son todos aquellos caracteres no alfabéticos que poseen un significado especial en el mundo de la informática, permitiendo representar otro conjunto de caracteres o ciertas operaciones específicas. En el caso del terminal de Linux existen:

- * : El asterisco se utiliza para reemplazar cero o más caracteres dentro de una sentencia de búsqueda, la cual entrega todas las coincidencias.
- ^ : El símbolo de suma se utiliza para hacer coincidir el texto dado con el inicio de una línea, en una búsqueda.
- [] : Los corchetes son usados para hacer coincidir en búsquedas un conjunto de caracteres.
- $: El símbolo de peso se utiliza para hacer coincidir el texto dado con el final de una línea, en una búsqueda. Este carácter también se puede utilizar para acceder a una variable en bash.
- ? : El interrogante cumple la misma función que el '*', sin embargo, solo funciona para exactamente un carácter.
- \ : El 'slash' invertido se utiliza para indicar que el siguiente metacaracter se debe ignorar y utilizar como carácter normal.
- ' ': En caso de querer anular el uso de metacaracteres se puede colocar el texto entre comillas simples. Estas comillas también se pueden utilizar para hacer uso de comandos en líneas de texto.
- | : Este símbolo (pipe) permite alterar la resolución normal de un comando, pasando el output de un comando como input de otro.
- > : Indica que lo anterior se transforma en un output.

- < : Se le asigna un input a lo escrito.
- || : Se comporta como un OR condicional.
- && : Se comporta como un AND condicional.
- & : Este comando permite hacer funcionar un comando en el terminal mientras otro se ejecuta.

Algunos ejemplos de uso son:

- ls /home/user/[¡ex]* → Buscará todo directorio o archivo en 'user' que no comience por la letra 'e' o 'x', tenga cualquier cantidad de caracteres luego de eso.
- ls /home/user/echo???????? → Buscará todo archivo o directorio en 'user' que comience por echo y tenga exactamente ocho caracteres más.
- echo \$USER → Normalmente el comando 'echo $USER' señalaría por terminal el nombre del usuario, sin embargo, dado el slash oblicuo la salida será "$USER", ignorando el simbolo de peso que toma a 'USER' como una variable.
- echo hola | grep h → Al output del comando echo, es decir, "hola" se buscará la línea que contenga una 'h, ósea, el output será la línea de 'hola'.
- echo hola > hola.txt → Se ejecutará el comando 'echo hola' y el output se almacenará en el archivo 'hola.txt'
- ls /home/user/[qxz]* && echo true → Ejecutará el comando 'echo true' siempre y cuando se encuentre un directorio en 'user' que comience por q, x o z.

c) Explique en que consiste la expansión 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.

Respuesta:

Brace Expansion o Expansión por Paréntesis de Conjunto es un método o mecanismo para generar arreglos de caracteres de forma arbitraria, es decir, crea una lista de elementos por la lista de elementos señalada, donde cada elemento está separado por comas, o para el rango dado, donde entre el carácter de partida y el de término existen dos puntos simples '..' explicitando que ese será un rango.

De forma que, la aplicación del comando con, por ejemplo, un echo es la siguiente:

- echo {0,1,2,3,4,5} → Output = 0 1 2 3 4 5
- echo {0..5} → Output = 0 1 2 3 4 5

Esto puede ser utilizado entre otros caracteres, tal y como se utilizará para resolver el problema dado, de la forma:

- mkdir 2021-{01..12}-{01..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.***
2. ***Contar los archivos con una secuencia de permisos r-x en los directorios /bin y /usr/bin.***

Respuesta:

1.- La ejecución de 'ls' y 'grep' interconectados se pueden mostrar todos aquellos archivos del directorio señalado que posean una 'i' entre sus caracteres, de la forma:

- ls /bin | grep i

2.- Para contar los archivos con una secuencia de permisos r-x en los directorios '/bin' y '/usr/bin' se hace uso del comando 'ls' para mostrar los archivos con el argumento '-la' para mostrar en lista todo el detalle de los archivos, el comando grep para; de todos esos archivos; filtrar los que contengan el arreglo 'r-x' y el comando 'wc -l' para mostrar la cantidad de líneas resultantes del comando 'grep', es decir, la cantidad de archivos, dado que cada línea es un archivo. Por lo tanto, el conjunto de comandos utilizados queda de la forma:

- ls /bin /usr/bin | grep r-x | wc -l

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

Respuesta:

1.- Las variables de entorno son aquellas variables del sistema que afectan procesos del mismo, y al ser variables, nos permiten obtener información del sistema. En el caso de Linux, las variables de entorno normalmente son marcadores de posición para la información almacenada en el sistema.

Existen varias variables de ambiente en Linux, por ejemplo:

- USER = Contiene el usuario actual que inició sesión en el sistema.
- HOME = Posee el directorio o ruta principal del usuario actual.
- SHELL = Posee el camino del Shell o interprete de comandos del usuario actual.
- PATH = 'PATH' es una lista de directorios separados por ':', en los cuales el sistema revisará la existencia de comandos utilizados.
- PWD = Directorio actual.
- SHLVL = Posee el nivel del Shell actual.

Utilizando el comando 'echo' y el metacaracter '$' se puede acceder a dicha información de cada variable, de la forma:

- echo $variable_ambiente

2.- Para modificar una variable simplemente se puede crear una variable del mismo nombre la cual la reemplazará. Para crear una variable se ejecuta una línea de comandos de la forma:

- VAR="valor"

Sin embargo, se debe aplicar el comando 'export' previo a lo anterior para convertir la variable de local a global, afectando así a todo el sistema, quedando de siguiente manera:

- export VAR="valor"

3.- Al crear un sub-Shell, las variables locales no funcionarán en ese terminal, dado que están en otro Shell, a pesar de que si las variables globales las cuales serán visibles desde todas los sub-Shell que se creen. HOME, por ejemplo, al ser variable global funcionará en todos los sub-Shell que se creen dando el mismo valor de resultado.

El 'export', como se dijo anteriormente, exporta las variables a otros Shell, de forma que la variable local creada pasará a ser global.

Un ejemplo de todo lo anterior seria:

TEST=”test”
echo $TEST → Output: test
bash
echo $TEST → Output (no hay).
exit
export TEST
bash
echo $TEST → Output: test

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.

Respuesta:

El script se encuentra adjunto y subido al GitHub.

A continuación, se explican algunas variables especiales de bash utilizadas:

- ‘$#’ = Muestra la cantidad de argumentos que se le pasaron al script en cuestión.
- ‘$*’ = Muestra la lista completa de argumentos pasados al script.
- ‘$$’ = Muestra el PID del script en el que se ejecuta.
- ‘$0’ = Muestra el nombre del script actual.
- Con el carácter ‘#’ se pueden realizar comentarios en Bash.

Todos los ejemplos y el código utilizado en esta pregunta se encuentran subidos en el Github:

https://github.com/Wasil-Was/SSOO-tarea01