Práctica 1

Introducción a Linux, Shell y Shell Scripting

Datos útiles

Para realizar esta práctica puede utilizar la máquina virtual provista por la cátedra (se recomienda hacer un snapshot de la VM antes de empezar a usarla, para poder volver atrás en caso de que algo deje de funcionar).
El link de descarga se encuentra en el moodle junto a las credenciales de acceso. Además, la VM se encuentra instalada en las PCs de la Sala A del Lab. Bacarla donde se realizan las prácticas de la materia.

1)
   a) Abra una consola de comandos y estudie el funcionamiento de los comandos: `help`, `ls`, `sort`, `cat`, `more`, `less`, `cp`, `mv`, `rm`, `mkdir` y `echo`.
      i) Determine cuales son comandos externos y cuales comandos internos. (Los internos pueden averiguarse mediante `man bash-builtins`)
      ii) Investigue las distintas opciones de estos comandos.
   b) Estudie la utilidad de los operadores `> >> | < <<`
   c) Ubíquese en el directorio `/usr/bin/` . Usando los operadores, genere en su directorio `home` un archivo de texto llamado "listado" con la salida del comando `ls -l`. Ordene el archivo obtenido utilizando el comando `sort`, de manera que quede ordenado por orden alfabético inverso y se guarde en otro archivo llamado "listado.r"
   d) Cómo utilizaría los comandos `cat`, `sort` y `more` para visualizar el archivo obtenido en c) por páginas y ordenado alfabéticamente en forma ascendente?

2) Uso de comodines. Se utilizan como comodines o caracteres de sustitución a los caracteres * y `?`

- `*` se utiliza para sustituir uno o más caracteres por ejemplo ***x*** significa cualquier nombre de archivo que incluya una letra x y ***term*** significa todos los archivos cuyo nombre incluya term .
- `?` se utiliza para sustituir un carácter que se encuentre en esa posición específica por ejemplo **x?123** va a corresponder a xa123, y a xb123 pero no a xab123.

Sitúese en la carpeta `/usr/bin/` y liste:

a) Todos los archivos que comiencen con **r**.

b) Todos los archivos que tengan en su nombre la partícula **min**.

c) Todos los archivos cuya tercera letra sea **x**.

3)

a) Explore la utilidad de los comandos `info` y `man`. Puede hacerlo mediante `info info` y `man man`.

b) Lea `info coreutils`. Se describen distintos comandos útiles para distintas tareas.

c) Explore los comandos `chown` y `chmod`. Explique cómo los usaría para convertir un archivo en ejecutable, para el dueño e impedir que se pueda leer, escribir y ejecutar al resto del mundo.

d) Explore el comando `id`. Averigüe a qué grupos pertenece y cuáles son sus valores numéricos.

4) Escriba un archivo de comandos (script) que realice las siguientes tareas:

1. Se sitúe en el directorio **/usr/bin**

2. Liste todos los archivos usando **ls -l** y copiando el listado en un archivo ubicado en el directorio **/home/soyredes/Documentos** llamado **listado.txt.**

3. Genere dos archivos llamados **listador.txt** y **listadon.txt** que serán una versión del archivo anterior pero ordenados alfabéticamente en orden decreciente y por tamaño de menor a mayor respectivamente.

```
4. Muestre en pantalla, de a una página por vez, los tres
archivos generados.
```

5) Explore la utilidad de las variables de ambiente, en particular `HOME`, `PATH`, `USER`. También explore la utilización de los comandos `wc`, `tail` y `head`.
   a) Explore la utilidad de las variables de ambiente de un script.
   b) Explore el uso de argumentos o parámetros en la línea de comandos del script.
   c) Escriba un script que acepte como argumentos de entrada el nombre de un directorio y un número **N** comprendido entre 1 y 100. Cuando se ejecute el script, se deben mostrar las primeras **N** líneas del listado de archivos correspondientes al directorio ingresado.
   d) Verifique que efectivamente se muestran esa cantidad de líneas, mediante el comando `wc`.

6) Explore la utilidad de los siguientes comandos: `pwd`, `df`, `shutdown`, `reboot`, `halt`, `find`, `locate`, `uname`, `dmesg`, `who`, `lspci`, `at`, `touch`, `tail`, `head`, `mount`, `umount`. ¿Cuáles de ellos modifican el funcionamiento del sistema? ¿Alguno de ellos necesita permisos especiales?

7) Explore los comandos para programar scripts en bash, en particular `for`, `while`, `until` e `if` (puede leer man bash, o en castellano un recurso útil es https://bioinf.comav.upv.es/courses/unix/scripts_bash.html) o también https://www.iqanansoft.es/categoria/tutoriales-programacion/tutoriales-scripting-bash/page/2/

8) Abra una consola en Windows, (utilizando cmd.exe). ¿Cuáles son las principales similitudes y diferencias entre esta CLI y la CLI de Linux?. Encuentre las equivalencias a los comandos `ls`, `cp`, `mv`, `rm`, `cd`, `mkdir`, `more`, `cat` y `echo` en la CLI de Windows.

9) Analice el funcionamiento del comando `grep`. Explore las opciones para mostrar líneas que cumplan cierto patrón o para mostrar líneas que no lo cumplan (opción `-v`).

Utilícelo para mostrar todas las líneas que contienen un directorio cuando ejecuto el comando `ls -l`.

10) Escriba un script que al pasarle un directorio en la línea de comando realice las siguientes tareas:

- Verificar que el argumento pasado en la línea de comandos es efectivamente un directorio.
- Revisar el contenido del directorio e imprimir:
  - cantidad de directorios que contiene.
  - cantidad de archivos regulares que contiene.
  - cantidad de archivos que no son regulares ni son directorios que contiene

11) Escribe un script de shell que ordene un arreglo de números enteros utilizando el método de la **burbuja** o el método de **selección**. El script debe pedir al usuario que introduzca los números del arreglo y luego mostrar el arreglo ordenado.

12) Implemente un script que agregue a un arreglo todos los archivos del directorio `/home/soyr/` del usuario cuya terminación sea `.txt`. Adicionalmente, implemente las siguientes funciones que le permitan acceder a la estructura creada:

- verArchivo <**nombre_de_archivo**>: Imprime el archivo en pantalla si el mismo se encuentra en el arreglo. Caso contrario imprime el mensaje de error "Archivo no encontrado" y devuelve como valor de retorno **5**
- cantidadArchivos: Imprime la cantidad de archivos del **/home** con terminación **.doc**
- borrarArchivo <nombre_de_archivo>: Consulta al usuario si quiere eliminar el archivo lógicamente. Si el usuario responde **Si**, elimina el elemento solo del arreglo. Si el usuario responde **No**, elimina el archivo del arreglo y también del FileSystem. Debe validar que el archivo exista en el arreglo. En caso de no existir, imprime el mensaje de error “Archivo no encontrado” y devuelve como valor de retorno **10**

13) **(Ejercicio entregable)**

Escribe un script de shell que permita gestionar un inventario de productos. El script debe ofrecer un menú con opciones para agregar un producto, listar los productos, buscar un producto por nombre, y ordenar los productos por precio. Los productos se almacenan en un archivo de texto, y cada línea del archivo contiene la información de un producto en el formato: nombre,precio,cantidad.

Requisitos:

1. **Agregar Producto:**
   - Solicitar al usuario el nombre, precio y cantidad del producto.
   - Añadir el producto al archivo de inventario.
2. **Listar Productos:**
   - Mostrar todos los productos en el archivo de inventario.
3. **Buscar Producto:**
   - Solicitar al usuario un nombre de producto y mostrar los detalles del producto si existe.
4. **Ordenar Productos por Precio:**
   - Ordenar los productos en el archivo de inventario por precio en orden ascendente y mostrar el listado ordenado.

Instrucciones:

- El archivo de inventario se llamará **inventario.txt**.
- El script debe validar las entradas del usuario.
- Utiliza estructuras de control como **if**, **while**, y **for**.
- Maneja adecuadamente la entrada y salida estándar, así como los errores.
- El script debe brindar un menú de ayuda al pasar cómo parámetro la opción **--help** ó **-h**

Pautas de entrega

La resolución debe hacerse en base a las pautas explicadas en clase, es decir:

- Seleccione una plataforma de control de versiones (ej: GitLab, GitHub, Bitbucket, …)
- Dentro de la misma, debe crear un grupo:
    - Nombre: **`SOyR 2024 Grupo X`** (**`X`** representa el número de grupo).
    - Descripción: apellido, nombre y número de alumno de cada integrante del grupo.
    - Debe agregar al docente asignado con el rol de `Owner`.
- Dentro del grupo, debe crear un repositorio:
    - Nombre: **`Entregable Práctica X`** (`X` representa el número de práctica)
    - El repositorio debe contener lo siguiente:
        - Archivo de programa **`eX.Y`** (**`X`** representa el número de ejercicio. **`Y`** representa la extensión del archivo, ejemplo: **`c`**, **`sh`**) correctamente estructurado y con los comentarios pertinentes.
        - Archivo **`README.md`**. Usando sintaxis **`markdown`** incorporar lo siguiente:
            - Copia del enunciado
            - Interpretación del problema. (qué debo hacer, qué datos tengo y cómo se organizan, cuál es la interfaz con el usuario)
            - Propuesta de solución (Cómo resuelvo lo pedido) describo los modelos utilizados: algoritmos, matemáticos o conceptuales. Aproximar la solución con un diagrama o pseudocódigo, modularizar.
            - Instrucciones de uso
        - Directorio **`imagenes`** ó **`figuras`** (opcional: en caso de que se requiera adjuntar imágenes en el **`README.md`**) con las figuras correspondientes.
        - **`.gitignore`** (opcional: en caso de que haya trabajado con más archivos en local)