0. Guía Complementaria de Ejercicios - Comandos de Shell (DOS & UNIX)

Nota: Para la resolución de los siguientes ejercicios, asegúrese de disponer de los privilegios (permisos) requeridos sobre su directorio de trabajo.

0.1. Comandos básicos de DOS y Unix

1) [**DOS**] Considere el comando `dir` de DOS. Analice sus distintos argumentos. Explique el significado de: `dir /o /p`.

2) [**UNIX**] Considere el comando `ls` de UNIX. Analice sus argumentos.

Explique el significado de las siguientes invocaciones, y analice sus resultados:

```
ls
ls -la
ls -ltr
```

3) [**DOS**] Sarta de comandos de DOS

a) Cree un subdirectorio denominado `version01` en la ruta: `c:\algoritmos1\pruebas_shell\tp1`, y posiciónese en ese directorio.

b) Copie el archivo `c:\windows\system32\calc.exe` al directorio actual, y ejecútelo localmente.

c) Cree un subdirectorio denominado `trash` dentro de la estructura: `c:\algoritmos1\pruebas_shell`.

d) Muestre por pantalla el contenido del archivo anterior, como para analizarlo línea por línea.

e) Cree la siguiente estructura de directorios dentro del directorio `trash`:

```
temp
  + clases
      + repaso
          + shell
```

f) Copie todos los archivos de extensión `dll` ubicados en el path `c:\windows\system32` al directorio `repaso` (usar el metacaracter "`*`")

g) Posiciónese en el directorio `shell`.

h) Genere cuatro archivos de texto conteniendo fechas, utilizando el comando `date`, con redirección a un archivo de texto:

```
date /t > fecha1.txt
```

Idem para los archivos `fecha2.txt`, `fecha3.txt` y `fecha4.txt`.

i) Muestre por pantalla el contenido de los cuatro archivos. Si lo desea puede editarlos con el editor de textos de DOS, Edit.

j) Copie estos cuatro archivos al directorio `repaso` ubicado un nivel más arriba en la estructura de directorios.

k) Elimine todos los archivos del directorio `repaso`, de forma tal que sólo persistan los copiados en el punto anterior. Verifique el resultado de la operación por medio del comando `dir`.

l) Copie los archivos `fecha1.txt` y `fecha3.txt` bajo los nombres `fecha11.txt` y `fecha33.txt`, respectivamente.

m) Elimine del directorio `repaso` los archivos cuyos nombres sean de la forma: `fecha%.txt`, en donde el símbolo "`%`" indica la presencia de un único carácter en dicha posición. Después de la eliminación, sólo deben quedar en el directorio los archivos `fecha11.txt` y `fecha33.txt` (utilizar el metacaracter "`?`").

n) Verifique los resultados de la eliminación con el comando `dir`.

ñ) Mueva todos los archivos del directorio `repaso` al directorio `trash`, verificando que no existan más en el directorio anterior.

o) Renombre el directorio `temp` con el nombre `a_borrar`.

p) Copie el directorio `trash` y todo su contenido al directorio `a_borrar` (copiado recursivo).

q) Elimine el directorio `a_borrar` pendiente de `trash`, con todo su contenido.

4) [UNIX]

a) Cree un directorio denominado `release` de acuerdo a la siguiente jerarquía, pendiendo de su directorio *home*:

`algoritmos/pruebas_shell/tp1/release`

y posiciónese en ese directorio.
(Para posicionarse en su directorio `home`, puede utilizar: `cd`)

b) Copie el archivo `/usr/include/math.h` al directorio actual (`.`).

c) Renombre el archivo copiado como `math.h` a `borrar.h`.

d) Muestre en pantalla el contenido del archivo. ¿Qué mecanismo puede utilizar para mostrar un archivo tan largo en pantalla? (R: *piping*, con `cat borrar.h | more`)

e) Cree la siguiente estructura de directorios:

```
temp
  + clases
      + repaso
          + shell
```

pendiendo del path: `${HOME}/algoritmos`.

f) Posiciónese en el directorio `shell`.

g) Copie todos los archivos de "extensión" `.h` al subdirectorio `shell` que ya creó en el punto $4e$).

h) Elimine el directorio `shell` y todo su contenido.

i) Renombre el directorio `repaso` por `a_borrar`.

j) Copie el directorio `clases` y todo su contenido (copiado recursivo) al path:

`${HOME}/algoritmos/pruebas_shell`

k) Mueva el directorio `a_borrar` al directorio `temp`.

l) Obtenga la ruta hacia el punto del sistema de archivos, donde usted se encuentra actualmente (comando `pwd`).

0.2. *Scripts*, permisos y *piping*

5) [UNIX]

a) Genere un archivo de texto, denominado `myscript.sh`, en su directorio *home*, con el siguiente contenido:

myscript.sh

```
1 clear
2 echo "Compilando modulo 1"
3 echo "Compilando modulo 2"
4 echo "Tarea finalizada: ‘date‘"
```

(Notar que ‘ es un acento grave, el que se utiliza por ejemplo en "à".)

b) Asigne al script `myscript.sh` los permisos de ejecución que considere convenientes.

c) Ejecute el script `myscript.sh`.

6) [UNIX/DOS]

a) Genere un archivo de texto, denominado `dir`, con el contenido del directorio actual, utilizando redireccionamiento:

```
dir > dir        [DOS]
ls > dir        [UNIX]
```

b) ¿Puede ser ejecutado el archivo recién creado? ¿Por qué?

c) Si le concediera al archivo anterior permisos de ejecución, ¿podría ejecutarlo?

d) ¿Es necesario que este archivo contenga extensión?

7) [UNIX/DOS]

a) Practique la concatenación de datos sobre archivos, utilizando el operador `>>`.
Ejemplo:

```
ls -la >> datos.log         [UNIX]
echo "hola" >> datos.log
ps -ef >> datos.log         (¿Qué hace este comando?)

dir *.* >> datos.log        [DOS]
time /t >> datos.log
echo "OK" >> datos.log
```

Revise luego el contenido del archivo `datos.log`.

8) [UNIX]

a) Cree en su directorio `${HOME}` el archivo `ejecutar` (sin extensión), con el contenido: `ls -la`, esto es:

```
ls -la > ejecutar
```

b) Examine el contenido del archivo (`cat`) y el del directorio (`ls -la`).

c) Cambie los permisos de ejecución del archivo `ejecutar`, para que lo pueda ejecutar cualquier usuario (*owner + group + other*), mediante el comando chmod (`chmod +x ejecutar`).

d) Ejecute el archivo `ejecutar` (ahora ejecutable). Si es posible, conéctese nuevamente al login con otro usuario (o utilice el comando `su`, switch user), y verifique que puede ejecutar el archivo `ejecutar` con un usuario distinto.

e) ¿Qué sucedería si un extraño con los permisos requeridos cambiara el contenido del archivo `ejecutar` por:

```
rm -rf
```

y luego lo ejecutara, gracias a los permisos concedidos? **NOTA: ¡NI LO INTENTE!**

9) [UNIX]

a) ¿Qué riesgos de seguridad presenta un directorio con permisos `777`?

b) ¿Qué riesgos de seguridad presenta un archivo con permisos `777`, logrado con `chmod 777 *`?

c) ¿Qué riesgos de seguridad presenta un directorio con permisos `751`?

10) [UNIX] En referencia al script del punto 8), ejecute el comando:

```
rm Ejecutar
```

¿Por qué genera el shell un mensaje de error?

11) [DOS]

a) Cree un archivo de texto denominado `ejecutar.bat`, con el siguiente contenido:

ejecutar.bat

```
1 dir *.*
2 time /t
3 echo "Proceso finalizado"
```

(para la edición puede utilizar la facilidad Edit de DOS).

b) Ejecute el archivo `ejecutar.bat`, y observe los resultados. ¿Por qué no hace falta asignarle permisos de ejecución al archivo de texto, de extensión `.bat`?

12) [**UNIX**]

a) Genere un archivo de texto denominado `test` (sin extensión) con el contenido:

test

```
1 ls -la
```

b) Asígnele permisos de ejecución convenientes.

c) Intente ejecutar el archivo `test`. ¿Por qué el shell muestra un mensaje de error? (Consulte las *man pages* en relación al comando nativo `test`, i.e. `man test`).

0.3. Procesos (optativo)

13) [**UNIX**] Analice el comando `kill`, ventajas y desventajas, y en qué casos particulares se recomienda su uso.