Práctica 3. Protección y Manipulación de Archivos. Variables del Shell.

1. Objetivos.

- Establecer y modificar los permisos de acceso al sistema de archivos Linux.
- Optimizar el acceso al sistema de archivos mediante el uso de caracteres especiales.
- Utilizar variables del *shell* y realizar operaciones aritméticas con ellas.

2. Protección de archivos.

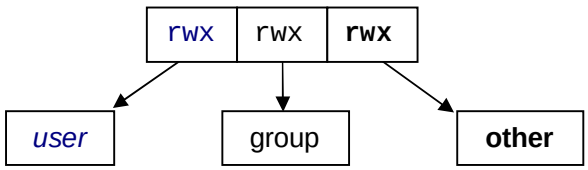
El sistema operativo Linux permite que diferentes usuarios compartan una misma computadora, por lo que es importante poder restringir el acceso a los diferentes archivos. Linux consigue esta protección de la siguiente forma:

1. Un usuario solamente puede realizar tres operaciones sobre un archivo: lectura, escritura (incluye las posibilidades tanto de modificación de su contenido como de eliminación del archivo) y ejecución.
2. El propietario del archivo es el responsable de especificar cuáles de estas operaciones puede realizar cada tipo de usuario.

Existen tres *categorías de usuario* diferentes:

1. User. Es el propietario del archivo (corresponde con el usuario que lo ha creado).
2. Group. Conjunto de usuarios que pertenecen al mismo grupo que el propietario del archivo.
3. Other. Cualquier otro usuario.

Cuando se observan las propiedades de un archivo (orden **`ls -l`**), los permisos aparecen representados en la primera columna mediante los caracteres *r* (lectura), *w* (escritura), *x* (ejecución) y – (no hay ningún permiso) de la siguiente forma:



Precediendo a estos nueve caracteres aparece otro que indica el tipo de archivo del que se trata. En el caso de archivos normales aparece el carácter – (guión), en el caso de subdirectorios aparece el carácter d. En el caso de subdirectorios, el permiso de ejecución se refiere a la posibilidad de visualizar o no los nombres de los archivos del mismo (por ejemplo, con la orden ls).

Ejemplos:

```
--wxr-x-w- 1 zzz00    alumnos    670 Mar  5  2000 semaforo.c
```

Se permite el acceso al archivo *semaforo.c* por parte de su propietario (usuario *zzz00*) tanto para escritura como para ejecución (-wx). Los usuarios del mismo grupo que el propietario (grupo *alumnos*) pueden leerlo y ejecutarlo (r-x). El resto de usuarios pueden escribir en él, pero no pueden leerlo ni ejecutarlo (-w-).

```
drwx-w---- 3 root     alumnos    64  Mar  6 19:43 temp
```

temp es un subdirectorio cuyo propietario es el usuario *root* que tiene todo tipo de permisos sobre él. Los usuarios del grupo *alumnos* pueden escribir en el subdirectorio (pegar archivos o crear archivos nuevos) pero no pueden leer los archivos que haya almacenados en el mismo (permiso de lectura denegado) ni siquiera obtener una lista de cuáles son (permiso de ejecución denegado). El resto de usuarios no pueden acceder de ninguna manera al contenido del subdirectorio *temp*.

Cambio de los permisos de acceso a un archivo (chmod).

La orden chmod permite cambiar los permisos de acceso a un archivo:

Sintaxis: `chmod Tipo_usuarioOperaciónPermisos archivo`

Tipo_usuario	**u** user **g** group **o** other Cualquier combinación (ug,uo,go,ugo).
Operación	**+** Añadir permiso **-** Quitar permiso **=** Asignar permiso
Permisos	**r** lectura **w** escritura **x** ejecución Cualquier combinación (rw,rx,wx,rwx).

Existe una alternativa para asignar permisos utilizando números en lugar de símbolos. Las correspondencias son las siguientes:

	User	Group	Other
r	400	40	4
w	200	20	2
x	100	10	1

Para obtener el valor numérico del permiso deseado basta con sumar los valores correspondientes.

Ejemplos:

Asignación de permisos de lectura y ejecución sobre el subdirectorio *temp* para los usuarios del grupo *alumnos*:

```
chmod g+rx temp
```

Modificación de los permisos del archivo *semaforo.c* para que el usuario *zzz00* solamente pueda leer su contenido. No se modifican los permisos de los demás usuarios.

```
chmod u=r semaforo.c
```

Asignación de permisos de lectura sobre el archivo *semaforo.c* para todos los usuarios, de escritura solamente para el propietario (*zzz00)* y de ejecución solamente para los usuarios pertenecientes al grupo *alumnos*.

Consultando en la tabla, los valores numéricos solicitados son:

r_user+r_group+r_other+w_user+x_group=400+40+4+200+10=654;

Entonces, la orden necesaria es:

```
chmod 654 semaforo.c
```

Permisos por defecto (umask).

Existe la posibilidad de establecer los permisos que se desea que por defecto no tengan los archivos que un usuario crea. La instrucción umask acepta como parámetro el valor numérico (máscara) de los permisos que se quieren quitar a todos los archivos que se creen posteriormente.

Sintaxis: `umask [permisos_numéricos]`

Cuando se utiliza sin parámetros, devuelve el valor actual de la máscara.

Ejemplo:

```
umask  444
```

A partir de la ejecución de la instrucción anterior, los archivos que se creen tendrán denegado el acceso para lectura a todos los usuarios.

3. Manipulación de Archivos.

Linux, a diferencia de otros sistemas operativos, no utiliza extensiones para diferenciar unos tipos de archivos de otros.

Además, dispone de una serie de caracteres especiales que simplifican ciertas tareas de manipulación de archivos (ls, cp, rm, ...):

Carácter	Equivale a
*****	0 ó más caracteres cualesquiera.
?	1 carácter cualquiera.
[ ]	Uno cualquiera de los caracteres que figuran dentro de los corchetes.

Se puede eliminar el significado de los caracteres especiales:

Carácter	Acción
' (Comilla simple)	Todos los caracteres especiales contenidos entre comillas simples son ignorados.
" (Comillas dobles)	Se ignoran todos los caracteres especiales excepto **$ ' ** cuando aparecen entre comillas dobles.
**** (backslash)	Se ignora cualquier carácter especial que vaya inmediatamente detrás.

Ejemplos:

`ls sin*.c`	Visualiza los nombres de todos los archivos que comiencen por *sin* y terminen con *.c*. Por ejemplo, listaría *sin.c* y *sincroniza.c*.
`ls diario?`	Lista los nombres de todos los archivos que comiencen con la cadena *diario* y tengan otro carácter adicional. Por ejemplo, visualizaría: *diario0*, *diario1*, *diarioA*, y *diario_*; pero no listaría *diario*, *diario23* ni *diario_1*.
`cp [Hh]ola.c ./`	Copia los archivos *Hola.c* y *hola.c*, si existen, al directorio actual.
`touch 'hola[amigos]'`	Crea un archivo de nombre *hola[amigos]* en el directorio actual.
`touch pasa_Pepe\?`	Crea un archivo de nombre *pasa_Pepe?* en el directorio actual.

4. Variables del Shell.

El *shell* de Linux no es otra cosa que una orden que se ejecuta al acceder a la computadora y que proporciona una interfase con el usuario para que éste pueda ejecutar tareas.

El shell proporciona un mecanismo para el empleo de variables que puedan almacenar información con un doble objetivo:

Control de entorno.

Programación *shell*.

Instrucciones relacionadas con variables

set: Asigna o borra el valor de una variable.

Sintaxis: `set nombre_variable[=valor_variable]`

El shell de Linux no admite tipos numéricos. Entonces, el valor asignado a una variable se considera siempre como una cadena de caracteres.

El efecto de omitir el valor que se asigna a la variable es el de borrarla.

Ejemplo: `set mivariable=mi_primera_variable`

Es importante observar que no hay espacios en blanco antes ni después del carácter **=**, pues los blancos son caracteres especiales.

Para referirse al valor de una variable se utiliza su nombre precedido del carácter **$**.

Dependiendo del shell utilizado, puede ocurrir que para asignar valores a variables no se utilice la orden *set*. En estos casos, la sintaxis utilizada para la asignación queda como se indica:

Sintaxis: `nombre_variable=valor_variable`

echo: Visualiza el valor de una variable o muestra por pantalla el resultado de evaluar una expresión.

Sintaxis: `echo $nombre_variable`

`echo cadena_de_caracteres`

Ejemplos:

```
echo $mivariable
```

Como resultado, se muestra por pantalla la cadena: *mi_primera_variable*.

```
echo "Me gusta jugar al Mus"
```

Como resultado, se muestra por pantalla la cadena: *me gusta jugar al mus*.

unset: Elimina variables (diferente de borrar su valor).

Sintaxis: `unset nombre_variable`

Evaluación de expresiones de tipo entero (expr).

Como se ha dicho, el *shell* de Linux no admite tipos numéricos, solamente cadenas de caracteres. De todas formas, existe la posibilidad de evaluar expresiones enteras utilizando la orden expr.

Sintaxis: `expr entero operador entero <operador entero>`

Observa que siempre hay un espacio en blanco como separador entre *operador* y *entero*. Además, pueden utilizarse variables para los operandos.

Ejemplo:

```
set a=5
expr $a + 3 – 1
```

Como resultado se visualiza la cadena de caracteres “7” (5+3-1=7).

Si se quiere asignar el resultado a una variable es necesario utilizar la orden *expr* entre caracteres ` (tilde invertida):

```
set b=`expr $a + 2`
```

En la variable *b* queda almacenada la cadena de caracteres “7” (5+2=7).

Variables de entorno.

Linux utiliza ciertas variables durante su ejecución para conocer, entre otras cosas, las preferencias del usuario y las características del sistema de computador utilizado. Algunas de ellas son las siguientes:

Variable	Función
HOME	Directorio propiedad del usuario.
SHELL	Nombre del *shell* que se está utilizando.
USER	Nombre de usuario.
TERM	Tipo de terminal que se usa.
DISPLAY	Pantalla para X-Windows.
PATH	Rutas de los archivos que se pueden ejecutar sin más que utilizar su nombre. Los directorios se separan entre sí con el carácter **:** (dos puntos).
prompt	*Prompt* de la línea de órdenes.

Como es lógico, el usuario del sistema puede comprobar los valores de estas variables y también modificarlos, aunque no suele ser recomendable.

5. Enunciado de la práctica.

Protección de archivos.

1. ¿Qué tipos de accesos permite el archivo */etc/passwd*? ¿Y */etc/shadow*? ¿Quién es el propietario de estos archivos?
2. Copia el archivo */etc/passwd* a tu directorio. Asigna a tu copia los siguientes derechos de acceso:

   El propietario puede leer, escribir y ejecutar el archivo.
   Todos los usuarios del grupo pueden leerlo y ejecutarlo, pero no escribir en él.
   El resto de usuarios solamente pueden ejecutarlo.

   Comprueba que el archivo tiene los permisos de acceso deseados.
3. Convierte los siguientes permisos a sus valores numéricos:

   ```
   rwxrwxrwx
   ---r--r--
   ```
4. Asigna a tu copia de *passwd* los permisos siguientes utilizando el valor numérico correspondiente:

   ```
   rw-r-xr--
   ```
5. Utiliza *umask* para que ningún otro usuario distinto del propietario pueda acceder a los archivos que se creen a partir de este momento. Comprueba que la máscara ha quedado actualizada.

Manipulación de archivos.

Entra dentro del subdirectorio temporal de tu grupo (`../temp`) y realiza las siguientes operaciones con una sola instrucción:

1. Muestra por pantalla el contenido de todos los archivos.
2. Visualiza los nombres y los permisos de los archivos que comienzan por la cadena ene.
3. Visualiza los nombres de los archivos que acaban con un carácter numérico y que tienen 5 y solamente 5 caracteres.

4. Vuelve a tu directorio $HOME y crea archivos con los siguientes nombres:

```
Archivo*
este es mi archivo
"otrofichero"
```

5. Comprueba con *ls* que realmente están creados y luego bórralos.

Variables del shell.

1. Prueba las siguientes instrucciones:

```
expr 5 + 6
expr 5 * 6
expr 5 – 10  /  5
```

¿Por qué se produce un error en la segunda instrucción? Corrígela para que funcione correctamente.

2. Crea una variable de nombre *mivar* y asígnale el valor 5. Posteriormente ejecuta la instrucción:

```
set mivar2=$mivar+1
```

   Comprueba el valor del resultado almacenado en *mivar2*.

3. Almacena en *mivar2* el resultado de la operación *mivar*5/3* y visualízalo.

4. Borra las variables *mivar* y *mivar2*.

Ejercicio

1. Crea un archivo de texto de nombre *script1* y teclea en él, en líneas distintas, las órdenes necesarias para:

   Mostrar por pantalla todos los archivos del directorio temporal que comiencen por *ene* y guardar el resultado en un archivo de nombre *listado*.

   Contar el número de líneas del archivo *listado*.

   Modificar los permisos de *listado* para que sean los siguientes:

```
rwx------
```

   Crear una variable de *shell* de nombre *valor1* y asignarle el valor 10.

   Crear otra variable de nombre *valor2* y asignarle el valor 30.

   Multiplicar ambas variables y dejar el resultado en otra variable de nombre *resultado*.

Mostrar por pantalla un mensaje con el valor de la variable *resultado*:

```
El valor de resultado es 'valor'
```

(Es imprescindible que aparezcan comillas simples a ambos lados del valor de la variable).

2. Inserta al principio del archivo *script1* la siguiente línea:

```
#!/bin/csh
```

3. Si es necesario, modifica las propiedades del archivo *script1* para que su propietario pueda ejecutarlo.

4. Ejecuta *script1*. Observa los resultados y comprueba cómo se ejecutan las órdenes del archivo de texto de igual forma que si se hubieran tecleado de una en una en la línea de órdenes.