Práctica 2. Introducción al sistema operativo Unix.

1.- Objetivos.

- Modificar la fuente de los datos de entrada o el destino de los resultados de las órdenes UNIX.
- Filtrar el contenido de archivos de texto para seleccionar solamente parte la la información que contienen.
- Establecer y modificar permisos de acceso al sistema de archivos UNIX.
- Optimizar el acceso al sistema de archivos mediante el uso de caracteres especiales.
- Utilizar variables de *shell* y realizar operaciones aritméticas con ellas.

2.- Redireccionamiento de entradas y salidas.

Todas las funciones de Entrada/salida (E/S) se realizan en UNIX utilizando archivos, incluidas las salidas por pantalla y las entradas desde teclado.

Por defecto, todas las entradas se hacen desde el teclado (archivo *stdin*) y las salidas se envían hacia la pantalla (archivo *stdout*). También se envían hacia la pantalla los mensajes de error (archivo *stderr*).

Los operandos `>, >>, <, >>, |` permiten obtener datos necesarios para una orden desde un archivo diferente del teclado, así como guardar los resultados en ficheros en lugar de mostrarlos por pantalla.

Operando	Resultado
`orden < archivo`	`orden` utiliza como datos el contenido de `archivo`.
`orden > archivo`	Guarda en `archivo` los resultados de `orden`. Si archivo ya existe, lo sobreescribe.
`orden << etiqueta`	`orden` utiliza como datos los que se escriban en las líneas siguientes hasta encontrar una línea que contenga `etiqueta`.
`orden >> archivo`	Añade los resultados de `orden` al final de `archivo`. Si no existe, lo crea.
`orden1 | orden2`	Envía el resultado de `orden1` a como dato para `orden2` utilizando una tubería o pipe (mediante el carácter [|]).

Ejemplos:

- `grep 'Pedro' lista_Amigos > lista_Pedros`
  Guarda en el archivo `lista_Pedros` las líneas del archivo `lista_Amigos` que contienen la palabra `Pedro`. Si `lista_Pedros` ya existía desaparece su contenido anterior.
- `grep 'Juan' lista_Amigos | wc -l`
  Busca las líneas del archivo `lista_Amigos` que contienen la palabra `Juan` y se cuenta el número de líneas encontradas. Podría servir para contar cuantos amigos de nombre `Juan` contiene el fichero `lista_Amigos`.

3.- Filtros.

Los filtros son órdenes o conjuntos de órdenes que permiten encontrar información concreta dentro de un archivo o cambiar su formato.

Expresiones regulares.

Hay órdenes que permiten buscar patrones de caracteres dentro de archivos de datos. Para indicarles qué patrones deben buscar se utilizan las llamadas *expresiones regulares*.

En las expresiones regulares pueden aparecer una serie de caracteres especiales cuyas funciones son las siguientes:

Carácter	Patrón de caracteres que representa
`c`	Si `c` es cualquier carácter distinto de `\[.*^]$` entonces representa una única aparición de ese carácter.
`\`	Elimina el significado especial de cualquier carácter.
`.`	Cualquier carácter.
`^`	Comienzo de una línea
`$`	Fin de una línea.
`[caracteres]`	Uno cualquiera de los caracteres contenidos en la lista que aparece entre corchetes. Son patrones especiales [A-Z], [a-z] y [0-9] que representan, respectivamente, a todas las letras mayúsculas, a todas las minúsculas y a todos los caracteres numéricos. También se pueden utilizar combinaciones de estos patrones especiales.
`[^caracteres]`	Un carácter cualquiera que no pertenezca a la lista que aparece entre corchetes.

Ejemplos:

- `jpg$`
  Línea que finaliza con los caracteres jpg.
- `^[A-Za-z][0-9]...ZZZ`
  Línea que comienza por una letra seguida de un número, tres caracteres cualesquiera y tres zetas mayúsculas.
- `[iI]magen\.doc`
  Línea que contiene la cadena de caracteres *imagen.doc* o bien *Imagen.doc*. Observe el significado especial del carácter (.) se elimina con la barra invertida (\ o *backslash*).

Búsqueda de patrones en archivos.

Para buscar un patrón concreto de caracteres dentro de un archivo se utiliza la instrucción *grep* (Global Regular Expression Pattern match).

Sintaxis: `grep expresion_regular archivo`

Por ejemplo, sea el contenido de un archivo de nombre *amigos* el siguiente:

```
juan%Burgos%18%juan@ubu.es
Luisa%Avila%19%luisa@estudiantes.unileon.es
Ernesto%Leon%18%ernjuan@estudiantes.unileon.es
Luis%Burgos%20%Leon@ubu.es
Marta%Leon%19%marLuis@estudiantes.unileon.es
Juan%Leon%19%juan@unilein.es
```

Observe que es una pequeña base de datos formada por un solo archivo en la que aparece un registro diferente en cada línea y en la que los campos de cada registro (nombre, localidad, edad y

dirección e correo electrónico) están separados por el carácter *%*.

Si se desea ver la información de todos los amigos que se llaman *Juan*, podría utilizarse la siguiente orden: `grep '[Jj]uan' amigos`

Los caracteres entre corchetes indican que se acepta la inicial tanto en mayúsculas como en minúsculas.

El resultado que se obtendría es el siguiente:

```
juan%Burgos%18%juan@ubu.es
Ernesto%Leon%18%ernjuan@estudiantes.unileon.es
Juan%Leon%19%juan@unilein.es
```

No solamente aparecen los amigos de nombre *Juan*, sino también *Ernesto*, que tiene los caracteres de *juan* en su dirección de correo electrónico. La solución correcta pasa por indicar a la orden *grep* que el patrón buscado debe estar al comienzo de la línea (que es donde se encuentra el campo nombre):

```
grep '^[Jj]uan' amigos
```

El resultado sería entonces:

```
juan%Burgos%18%juan@ubu.es
Juan%Leon%19%juan@unilein.es
```

Selección de campos dentro de un archivo.

La orden cut permite visualizar caracteres o campos concretos de todas las líneas de un fichero.

Sintaxis:
```
cut -cRANGO archivo
cut [-dcaracter] -fRANGO archivo
```

Modificadores:
- `-cRango`: Selecciona los caracteres dentro del *RANGO*.
- `-dcaracter`: Especifica el separador de campos
- `-fRANGO`: Selecciona los campos entro del *RANGO*.

RANGO puede tener los siguientes formatos:

`número-`	Todos los caracteres o campos desde `número` hasta el último.
`número1-número2`	Todos los caracteres o campos entre `número1` y `número2`.
`número1,número2,...`	Solamente los campos `número1`, `número2`, ...

Para una aplicación que envíe correos electrónicos a toda la lista de amigos, podría ser interesante obtener las direcciones de todos ellos a partir del archivo *amigos:*

```
cut -d% -f4 amigos
```

El resultado es:

```
juan@ubu.es
luisa@estudiantes.unileon.es
ernjuan@estudiantes.unileon.es
Leon@ubu.es
marLuis@estudiantes.unileon.es
juan@unilein.es
```

Si, además, se deseara tener el nombre de cada amigo, la orden adecuada sería:

```
grep -d% -f1,4 amigos
```

Cuyo resultado es:

```
juan%juan@ubu.es
Luisa%luisa@estudiantes.unileon.es
Ernesto%ernjuan@estudiantes.unileon.es
Luis%Leon@ubu.es
Marta%marLuis@estudiantes.unileon.es
Juan%juan@unilein.es
```

Ordenamiento del contenido de un archivo.

Se pueden ordenar alfabéticamente las líneas de un archivo utilizando la orden *sort*.

Sintaxis: `sort [-ru] archivo`

Modificadores:
`-r`: Orden descendente.
`-u`: Elimina filas duplicadas.
`-ru`: Orden descendente eliminando filas duplicadas.

El ordenamiento se realiza en base a los valores numéricos de los caracteres ASCII, por ello, el carácter *Z* (mayúscula) irá antes del carácter *a* (minúscula) en el archivo ordenado de forma ascendente.

Traducción de caracteres.

La orden *tr* facilita la tarea de reemplazar unos caracteres por otros dentro de un fichero.

Sintaxis:

```
tr 'conjunto inicial' 'conjunto final' < fichero
tr -d 'conjunto inicial' < fichero
```

Cuando se utiliza sin ningún modificador, *tr* lee el archivo `fichero` y reemplaza los caracteres contenidos en el conjunto inicial por los caracteres del conjunto final. Se visualiza el resultado pero no se modifica el archivo inicial.

Con el modificador `-d`, lee el archivo `fichero` y elimina los caracteres contenidos en el conjunto inicial. Se visualiza el resultado, pero no se modifica el archivo original.

Ejemplos:

- `tr '%' '&' < amigos > amigos_2`
  Cambia el carácter *%* (separador de campos) del archivo `amigos` por el carácter *&* y guarda el resultado en el archivo `amigos_2`. El contenido de `amigos` no varía, mientras que en `amigos_2` se puede visualiza con la orden *cat* y es el siguiente:

  ```
  juan&Burgos&18&juan@ubu.es
  Luisa&Avila&19&luisa@estudiantes.unileon.es
  Ernesto&Leon&18&ernjuan@estudiantes.unileon.es
  Luis&Burgos&20&Leon@ubu.es
  Marta&Leon&19&marLuis@estudiantes.unileon.es
  Juan&Leon&19&juan@unilein.es
  ```

- `tr '[A-Z]' '[a-z]' < amigos`

Cambia todos los caracteres en mayúsculas por caracteres en minúsculas, visualiza el resultado, pero no guarda los cambios

- `tr -d '@' < amigos >> amigos`
  Elimina todos los caracteres @ del fichero amigos y guarda el resultado añadiéndolo al final del mismo. No se visualiza el resultado por pantalla porque se redirecciona la salida (>>).

4.- Protección de archivos.

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

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

rwx	rwx	**rwx**
user	group	**other**

Precediendo de estos nueve caracteres aparece otro que indica el tipo de archivo del que se trata. En el caso de los 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 (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 o copiar archivos nuevos) pero 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 permisos de un archivo.

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

Sintaxis: `chmod Tipo_usuario*Operación*Permisos 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+rw 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*.

  Consultado 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.

Cuando creamos un archivo en nuestro sistema por defecto se le asignan permisos de lectura y escritura tanto para el propietario como para todos los demás, los permisos pueden ser modificados para que se asignen otros permisos distintos a los que vienen por defecto en la distribución, ésto lo logramos mediante el comando umask.

Sintaxis: `umask [permisos_numéricos]`

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

Ejemplo:

- `umask 066`
  A partir de la ejecución del comando anterior, los archivos que se creen tendrán denegado el acceso para lectura, escritura y ejecución a todos los usuarios que no sean propietarios del archivo.

¿Cómo se determina la máscara a aplicar? Debe tenerse en cuenta que existen permisos base para la creación de archivos y la de directorios, los permisos base de los archivos es 666 y los de los directorios es 777 partiendo de ésta base de permisos la operación para determinar los permisos de creación de archivos y de directorios consiste en restar el valor que se le da umask a los permisos base, si creamos un archivo se lo restamos a los permisos base de los archivos que son 666 y en caso de ser un directorio lo que se hace es restárselo a los permisos base de los directorios que son 777, a continuación veremos un ejemplo para que sea más claro.

Por ejemplo si la máscara de permisos es 022 supone que los directorios se van a crear con la máscara de permisos resultante de la resta 777-022=755 y en el caso de los archivos de 666-022=644.

5.- Manipulación de archivos.

UNIX, 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, etc.):

Carácter	Equivale a
`*`	0 o 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 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 ficheros que comiencen por *sin* y terminen con *.c*. Por ejemplo, listaría *sin.c* y *sincroniza.c*.
- `ls dirario?`
  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.

6.- Variables de Shell.

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

El *shell* proporciona un mecanismo para el empleo de variables que puedan almacenar información con un doble objetivo:
- Control del entorno.
- Programación *shell*.

Instrucciones relacionadas con variables.

Existen diferentes funciones para poder asignar el valor de una variable por ejemplo:

set: Asigna o borra el valor de una variable.

Sintaxis: `set nombre_variable[=valor_variable]`

Sin embargo este tipo de asignación de valor no funciona en las *shell* de tipo *bash* como la de los ordenadores del aula. En este caso la asignación de un valor a una variable se haría del siguiente modo:

Sintaxis: `nombre_variable[=valor_variable]`

El shell de UNIX 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.

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 *$*.

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

Para visualizar los valores de las variables de la shell actual pueden emplearse el comando **set** sin parámetros adicionales

Sintaxis:
```
echo $nombre_variable
echo cadena_de_caracteres
```

Ejemplos:
- echo $mi_variable

Como resultado. se muestra por pantalla la cadena 'mi_primera_variable'.

- echo “Me gusta jugar al mus”
  Muestra por pantalla la cadena 'Me gusta jugar al mus'.

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

Sintaxis: `unset nombre_variable`

En ocasiones es interesante el uso de una variable por los procesos hijos de una shell y en ese caso se utilizaría **export**

Sintaxis: export `nombre_variable[=valor_variable]`

Evaluación de expresiones de tipo entero.

Como se ha dicho, el *shell* de UNIX 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.

Ejemplos:

- `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).

Variables de entorno.

UNIX 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:

Variable	Función
`HOME`	Directorio propiedad del usuario.
`SHELL`	Nombre del *shell* que se está usando.
`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.

7.- Desarrollo de la práctica.

Obtención del número de caracteres, palabras y líneas de un archivo.

1. Utiliza el comando *man* para obtener los archivos de ayuda que contienen la palabra file (Sintaxis: `man -k file`).
2. En lugar de ver la salida en pantalla, envíala a un archivo que se llame *lista.file*.
3. Obtén el número de archivos de ayuda que hacen referencia a la palabra *file*. Para ello cuenta el número de líneas del archivo *lista.file*. Utiliza la orden *wc*, cuya sintaxis es la siguiente:

   `wc [-lwm] archivo`
   Modificadores:
   -l: Devuelve el número de líneas de archivo.
   -w: Devuelve el número de líneas de archivo.
   -m: Devuelve el caracteres de líneas de archivo.
4. Utiliza una tubería (carácter [|]) para ver el mismo resultado sin necesidad de usar el archivo intermedio *lista.file*. El resultado de `man -k file` se debe enviar a la orden `wc`.

Busqueda de patrones.

1. Visualiza las líneas de *lista.file* que contienen la palabra *password*.
2. Visualiza las líneas de que comienzan con la letra *o*.
3. Visualiza las líneas que terminan con letra mayúscula.
4. Cuenta las líneas de l*ista.file* que contienen la palabra *password*. Obtén el resultado empleando una sola línea de órdenes y sin utilizar el archivo *lista.file*.

Selección de campos y ordenamiento

1. Visualiza los caracteres de las columnas 1 a la 2, 27, 44 y de la 54 a la 55 del archivo *cortaletras* que se encuentra en la carpeta temporal y que contiene el siguiente texto:

```
PARA RAQUEL ESTABA MUY CLARO EL HECHO DE QUE AUGUSTO CESAR QUERIA CASARSE CON
SU MADRE, LUISA MARY.
ESPABILATE, LE AZUZABA SU TIA LINA. SI VAS A MURCIA ANTES QUE LUISA MARY,
HALLARAS UN BIZCONDE QUE TE AYUDARA A ENCONTRAR SOLUCION A TU MAL.
>;---------------------PEDRO DE LA BARCA
```

2. Visualiza los nombres de las órdenes referenciadas en el archivo *lista.file*. Observa que el carácter separador de campos es – (guion) y que las órdenes aparecen en el primer campo de cada línea.
3. Repite el ejercicio anterior pero de manera que en lugar de visualizar el resultado, éste se almacene en el archivo *lista.ordenes*. Si ese archivo no existe, debe crearlo. Si ya existe, debe sobreescribirlo.
4. Escribe la línea de órdenes necesaria para cortar el resultado de `man -k file` el nombre de las instrucciones y presentarlas ordenadas alfabéticamente.

Traducción de caracteres.

1. Utiliza la orden *apropos* para ver los ficheros de ayuda que hay con la palabra *password* (sintaxis: `apropos password`). Guarda el resultado en un archivo de nombre *lista*.
2. Añade al final de *lista* el resultado de la misma operación con la palabra *passwd*.
3. Ordena el archivo *lista* y elimina las filas duplicadas. Guarda el resultado en un archivo de nombre *lista.sinduplicar*.
4. Cambia todos los caracteres – (guion) que aparezcan en *lista.sinduplicar* por caracteres @ y guarda el resultado en *lista2*.

5. Cambia las letras minúsculas de *lista2* por mayúsculas y deja el resultado en *lista3*.
6. Borra los espacios en blanco de *lista3*. Deja el resultado en *lista2*.
7. Vuelca por pantalla el resultado de *lista2*.

Protección de archivos.

Ss

1. ¿Qué tipos de accesos permite el fichero /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.
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.

Variables de 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*.