# GUIÓN Nº 2 DE PRÁCTICAS DE UNIX

**En la respuesta a cada pregunta tiene que aparecer literalmente cada una de las líneas de comando ejecutadas, teniendo en cuenta que hay que saber razonar esa respuesta.**

1. (Este ejercicio es el 11 del guión nº1). Establece un enlace entre el fichero de texto **ejemplo.txt** creado en el apartado 9, con otro que llamaremos **ejemplo_enlace.txt**, y que ubicaremos dentro del directorio **dir1** que cuelga del HOME.

De igual forma, crea en tu directorio HOME un fichero de texto llamado **ejemplo2.txt**. Ahora, en vez de enlazarlo, como antes, vamos a copiarlo, para lo cual, copia ese fichero, ubicando el fichero destino en **dir1**, con nombre **ejemplo2_copia.txt**.

Mediante el editor *vi* modifica (por ej., simplemente borrando un caracter) el contenido de los ficheros **ejemp1o_enlace.txt** y **ejemplo2_copia.txt**. Comparar el contenido de éstos con el de **ejemplo.txt** y **ejemplo2.txt,** respectivamente. Razonar el resultado de la comparación propuesta: cuál de estos últimos también ha variado su contenido, por qué, … Borrar tanto **ejemplo_enlace.txt** como **ejemplo2_copia.txt**.

2.Mediante el comando **cat** crea un fichero de texto de nombre **texto1** situado en tu **HOME**.
Concatenar los ficheros iso y ascii que cuelgan de /usr/pub en otro llamado **todo** situado en tu **HOME**.

3.Ordenar el fichero **texto1** del apartado anterior, y el resultado guardarlo en un fichero que llamaremos **texto1.ord**. Realizar esta operación de dos formas distintas:

- pasando al comando *sort* el fichero **texto1** del apartado anterior como argumento,
- y usando redirección de entrada.

4.Crear un fichero de texto que contenga bastantes líneas (el contenido es irrelevante). A partir de éste:

- Crear un nuevo fichero que contenga todas las líneas que empiezan[1] por caracteres numéricos (0, 1, 2, …, 9)[2].
- Ordenar su contenido de forma que las líneas aparezcan en pantalla de forma descendente (primero las que empiezan por z y terminando por las que empiezan por A).
- Buscar las apariciones de la cadena de caracteres **os** en el fichero, y que todas las líneas que la contengan sean ordenadas de forma alfabética, eliminando todas las posibles líneas repetidas.

5.Mediante tuberías, crear una línea de comandos cuyo resultado sea el número de ficheros y directorios de un determinado directorio.

---

[1] Cuando se quiere hacer referencia al principio de una línea de texto, se usa el carácter ^ precediendo a el/los carácter/es que queremos encontrar en el principio de línea.
De forma análoga, cuando se quiere hacer referencia al final de línea se utiliza el símbolo $

[2] Cuando se quiere expresar en el nombre de un fichero, directorio o en una expresión (cadena de caracteres que se le pasa como argumento a un comando) que un determinado intervalo de caracteres está permitido en determinada posición, se utiliza la notación que se ve en los siguientes ejemplos:

- [A-D] en el nombre del fichero o directorio indica que en esa posición puede estar cualquier carácter ASCII comprendido entre la A y la D (en este caso A, B, C y D). **Por ejemplo**: MARC[A-C]S agruparía las siguientes cadenas de caracteres: MARCAS, MARCBS, MARCCS.
- [A-z] en el nombre del fichero o directorio indica que en esa posición puede estar cualquier carácter ASCII comprendido entre la A y la z (lo que incluye a todas las letras mayúsculas y minúsculas)

Vamos a afinar más: crear dos líneas de comandos diferentes, de forma que una dé como resultado el número de ficheros, SÓLO FICHEROS, de un determinado directorio, y la otra dé como resultado el número de directorios, SÓLO DIRECTORIOS, de un determinado directorio.

Repetir lo mismo pero ahora enviando el resultado de las diferentes líneas de comandos a un mismo fichero, de forma que el contenido final de éste ha de ser (las cadenas de caracteres en cursiva, deben aparecer literalmente en el fichero):

*Numero total de ficheros y directorios:*
xx
*Numero de ficheros:*
ff
*Numero de directorios:*
dd

Utilizar las líneas de comandos necesarias.

6.Interpretar los resultados que se obtendrían al ejecutar las siguientes líneas de comandos, indicando cuál o cuáles de ellas son equivalentes entre sí, considerando para ello solamente sus informaciones de entrada y salida:

cat *[ruta]fichero* | grep *cadena_caracteres* | wc -l
cat | grep *cadena_caracteres* | wc -l
cat < *[ruta]fichero* | grep *cadena_caracteres* | wc -l
cat *[ruta]fichero* | grep *cadena_caracteres* | sort | uniq
sort *[ruta]fichero* | uniq | grep *cadena_caracteres*

De las anteriores líneas de comandos: proponer una equivalente a la primera pero con tan solo dos comandos en la tubería, o sea, “comando 1 | comando 2”.

Si redireccionáramos la salida de cualquiera de las líneas de comandos anteriores, cuáles de las siguientes tendrían el mismo resultado, suponiendo que el directorio actual es uno llamado **tmp**, que es hijo de nuestro HOME:

*línea de comandos* > $HOME/tmp/resultado.txt
*línea de comandos* > resultado.txt
*línea de comandos* > /resultado.txt
*línea de comandos* 2> ../resultado.txt
*línea de comandos* 2> $HOME/resultado.txt
*línea de comandos* 2> resultado.txt

Comprueba las respuestas con un caso práctico.

7.Interpretar la siguiente línea de comandos, primero sin ejecutarla, y luego verificarla ejecutándola:

sort | uniq | grep a > a.out

8.Indicar el error en cada una de las siguientes líneas (¡OJO!: puede no dar error de ejecución pero ser semánticamente incorrecta):

cat | grep *cadena_caracteres [ruta]fichero* | wc -l
sort | uniq < *[ruta]fichero* > *[ruta]fichero*
cp *[ruta]fichero/s_origen* > *[ruta]fichero/s_destino*
cat *[ruta]fichero/s* > *[ruta]fichero* | grep *cadena_caracteres*
ll -d /usr/p* > grep ^d
ll *.txt > cp textos
cadena_caracteres > *[ruta]* fichero

Corregir aquéllas que se pueda.

9.Haz que el directorio raíz sea tu directorio actual, y **sin moverte de él:**

Determinar cuál es el valor de la variable de entorno **HOME**, y **usando dicha variable**, resolver los siguientes puntos:

- crear en tu directorio **HOME** un subdirectorio llamado **temp**
- crear en el directorio **temp** del apartado anterior un fichero llamado **basura** que contenga la lista de todos los ficheros (**sólo los ficheros**) del directorio **/usr/lib** que contengan en su nombre la cadena **ld**
- mostrar por pantalla el contenido del fichero **basura** ordenado alfabéticamente, y por último
- borrar el directorio **temp**.

10.El siguiente comando UNIX produce un error al ser ejecutado (el comando **gcc** es un compilador de C):

**gcc $TERM**

Mediante el procedimiento adecuado, hacer que el mensaje de error que aparece se añada a los ya existentes en un fichero situado en el directorio **HOME** llamado **basura.msg**. Justificar la aparición del error.

11.Mostrar todos tus procesos, sólo **TUS** procesos (mediante la opción adecuada del comando *ps*), identificando para cada uno: su nombre, su PID, su proceso padre, la hora de comienzo y el tiempo acumulado

Cada sesión de trabajo lleva asociada la ejecución de un "shell", mediante el cual nos comunicamos con el KERNEL del Sistema Operativo. En nuestro caso el nombre de este proceso es **bash (GNU Bourne-Again SHell)**.

- Obtener el PID del proceso shell de la sesión actual.
- Comprobar que la hora de comienzo de su ejecución coincide con la de inicio de la sesión.
- Identificar el propietario del proceso padre del shell actual.
- Cerrar la sesión actual eliminando mediante el comando *kill* el proceso shell asociado a ésta.

12.Abrir una nueva sesión de trabajo.

a) Vamos a cambiar de intérprete de comandos, o sea, de shell. Los principales existentes son los siguientes:
   - **bash.**
   - **sh.** Bourne shell. Puede ser considerado como el shell estándar.
   - **csh.** C-shell. Se caracteriza por la similitud con el lenguaje de programación C en alguno de sus comandos, fundamentalmente los asociados con la creación de programas shell.
   - **ksh.** Korn shell.

   Para cambiar de shell basta con ejecutar el programa asociado al deseado. El nombre de este programa (del fichero ejecutable) se corresponde con la abreviatura indicada al principio de cada línea de la lista anterior
   A partir de lo expuesto:
   1.- Cambiar de shell (a cualquiera de los anteriores). Ver cómo cambia el prompt del sistema.
   2.- Mostrar todos tus procesos, identificando el padre del nuevo shell que acabas de abrir.
   3.- Eliminar el shell que se abrió con la sesión de trabajo (el bash).
   4.- ¿Por qué se ha cerrado también el último shell abierto?

b) Abrir una nueva sesión. Desde ésta vamos a abrir otra nueva ejecutando el siguiente comando:

**ssh tu_nombre_usuario@duero**

1.-Con qué máquina nos hemos conectado.
2.-Mira tus procesos. ¿Cuántas sesiones tienes abiertas?
3.-Mediante kill cierra la última sesión abierta.

13.La variable PATH contiene el grupo de directorios donde el sistema operativo busca los ficheros ejecutables. Para entender este concepto realizar el siguiente ejercicio:

a) Examinar el valor de la variable **PATH** con el comando adecuado.
b) Vaciar la variable **PATH** de su contenido escribiendo en la línea de comandos **PATH=**
c) Intentar hacer un listado del directorio actual y ver qué pasa.
d) Asignar a la variable **PATH** la ruta del directorio donde está el comando **ls**, es decir, escribir en la línea de comandos **PATH=/usr/bin**
e) Hacer un listado del directorio actual y ver qué ocurre ahora y por qué
f) Restaurar el valor inicial de la variable **PATH** cerrando la sesión y volviendo a entrar, o ejecutando la siguiente línea de comandos (incluyendo el "." del principio):

**. /etc/profiles/setpaths**

g) Crear en el directorio **HOME** un subdirectorio llamado **bin**.
h) Copiar el comando **ls** con el nombre de **mils** en el directorio **bin** que acabas de crear.
i) Hacer que el directorio actual sea el raíz y desde allí ejecutar el comando **mils**.
j) Modificar el valor de la variable **PATH** de forma que contenga la vía de acceso absoluta al directorio **bin** que creaste, además de lo que ya contenía.
k) Ejecutar el comando **mils** desde el directorio raíz. Ver que se puede ejecutar ahora sin escribir su vía de acceso.

14.Abrir dos sesiones distintas en dos ventanas distintas. Llamaremos a una sesión A y a la otra sesión B.

a) Desde la sesión A ejecutar el comando: **find / -name ascii 2> /dev/null**. Como puedes comprobar tarda en finalizar su ejecución, dejando el terminal bloqueado mientras tanto. Eliminar este proceso desde la sesión B.
b) Desde la sesión A ejecutar el mismo comando, pero ahora sin que el terminal quede bloqueado (en background). Comprobar que efectivamente se pueden seguir ejecutando otros comandos mientras el anterior sigue su ejecución. Desde la sesión B mostrar todos tus procesos activos, entre ellos debe estar el find. Cerrar la sesión A, y desde la B volver a mostrar todos tus procesos activos, comprueba que ha desaparecido el find: ¿por qué?
c) Abrir una nueva sesión A y repetir el apartado anterior, pero ahora haciendo que la ejecución de find no finalice al cerrar la sesión A. Una vez cerrada ésta, comprobar que esto es cierto, o sea, que tras cerrar la sesión el comando find sigue activo.