GUIÓN DE PRÁCTICAS DE UNIX
PROGRAMACIÓN DEL SHELL

1. Analizar el comportamiento del siguiente programa shell al ser ejecutado mediante las tres formas que conoce

```
echo dentro
exit
echo fuera
```

2. Cree en su directorio HOME un subdirectorio que se llame `bin`. Modificando adecuadamente el contenido del fichero `.profile`, añadir a la variable `PATH` el camino a este nuevo directorio[1]. La idea es dejar en este directorio todos aquellos comandos o aplicaciones que vayamos desarrollando, y así, al añadir su camino al `PATH`, podremos ejecutarlos desde cualquier punto de la estructura de directorios en el que nos encontremos.

3. Interpretar cuál es la función de la siguiente línea de comandos:

```
for a in `cat  /usr/pub/ascii`
do
   echo “$a”
   echo “$a” | wc
   echo
done
```

4. Crear un procedimiento shell que borre todos los ficheros que se encuentran especificados en el fichero `$HOME/bin/purgar`. Por ejemplo, si ese fichero contiene las siguientes líneas:

```
a.out
*.o
core
```

el proceso habrá de borrar todos los ficheros `a.out`, `*.o` y `core` que se encuentren en el directorio de trabajo actual.

[1] Antes de realizar esta modificación asegurarse de hacer una copia de seguridad del fichero original.

5. Crear los siguientes programas shell:

- Un programa llamado `inicializa` que verifique si existe el directorio `$HOME/bin`. Si no existe deberá crearlo.
- Un programa llamado `agrega` que verifique que recibe un único argumento. El siguiente paso es comprobar si dicho argumento se encuentra incluido en un fichero situado en `$HOME/bin/claves.tex`. Si no es así, deberá añadirse a él.
- Un programa llamado `borra` que en primer lugar verifique que recibe al menos un argumento. También debe comprobar que existe el fichero `$HOME/bin/claves.tex` y que no está vacío. En caso de que lo esté o no exista, el proceso termina generando un mensaje por pantalla. En el caso afirmativo, ha de eliminar del fichero `$HOME/bin/claves.tex` las apariciones de los argumentos que se le han pasado por la línea de comandos.
- Un programa llamado `ver` que muestre el contenido del fichero `$HOME/bin/claves.tex` por pantallas completas. Si dicho fichero no existe, el proceso genera un mensaje de error y termina.
- Un programa llamado `busca` que liste nombres de los ficheros del directorio de trabajo actual que contengan alguna de las palabras (cadenas de caracteres) contenidas en `$HOME/bin/claves.tex`.
- Un programa llamado `consistencia` que elimine del fichero `$HOME/bin/claves.tex` todas las líneas repetidas.

6. Construir un programa shell que presente por pantalla un menú en el que aparezcan como opciones los programas desarrollados para el problema anterior, y que quede a la espera de una elección. El programa elegido será ejecutado.

7. Vamos a crear un programa shell que automatice el proceso de compilar y ejecutar seguidamente un programa de Pascal. Al proceso se le pasa, al menos, un argumento: el nombre, sin la extensión, del programa Pascal. El proceso debe hacer las siguientes operaciones:

- Comprobar si el fichero en fuente Pascal pasado como primer argumento existe.
- Compilar el programa, enviando los mensajes de error, si los hay, a un fichero.
- Si existe algún error en la compilación, se indica esta eventualidad, y se finaliza la ejecución del proceso.
- Si no existe error en la compilación, primero se pide la conformidad del usuario para su ejecución, y si se la da, se ejecuta el programa Pascal.

8. Crear un proceso capaz de limpiar un(os) determinado(s) directorio(s) de sus subdirectorios hijos. A este proceso se le pasa como argumento(s) el(los) directorio(s) cuyos subdirectorios hijos quieren ser borrados. El proceso antes de borrar cada subdirectorio hijo ha de pedir confirmación de la operación al usuario. Si la respuesta es negativa ese subdirectorio no es borrado y se va a por el siguiente.
NOTA: El comando rm con la opción adecuada permite borrar un subdirectorio y toda la estructura de árbol que cuelga de él sin necesidad de tener que borrar previamente su contenido.

9. Realizar un comando de visualización del contenido de un directorio distinto de los proporcionados por UNIX. La sintaxis de invocación al proceso por lotes será:

Nombre_del_proceso –d/-f *[directorio]*

El primero argumento puede tener, como se ve, dos valores: -d ó –f, si se especifica un valor distinto de estos dos, se indica el error de sintaxis y se abandona la ejecución. Si vale –d se muestra por pantalla el nombre de los subdirectorios (sólo subdirectorios) existentes en *directorio*. Al final de la lista se añade la frase:

“ `número de directorios: n`”

donde n ha de ser el número de directorios listados. Si vale –f, se muestra por pantalla el nombre de los ficheros (sólo ficheros) existentes en *directorio*. Añadiendo al final:

“ `número de ficheros: n`”

donde n ha de ser el número de ficheros.

En cuanto al segundo argumento puede aparecer o no. Si aparece, seré el nombre y ruta del directorio cuyo contenido se va a mostrar (en este caso, antes de hacer nada se habrá de examinar si existe o no el directorio especificado). Si no aparece se supone que el directorio a mostrar será el de trabajo.

Ejecutar el comando anterior, haciendo que la información de salida (lista de directorios o ficheros) aparezca en pantalla ordenada alfabéticamente en sentido inverso. ¡¡OJO!!, no se pide modificar el proceso por lotes creado, solo tratar sus datos de salida.

10. (Problema de examen) Es el último día para entregar las prácticas de Pascal y todos los puestos del laboratorio están ocupados. Al final hemos convencido a un amigo nuestro que tiene reservada la hora y está trabajando en un terminal, para que nos permita usarlo, “pero sólo el mínimo tiempo posible”. Nosotros le hemos prometido que sólo vamos a modificar los programas pero que no los vamos a compilar porque ya no nos daría tiempo a hacer más correcciones.

Tenemos un comando llamado `modifica`, situado en el directorio `bin` (no accesible por la variable PATH) que cuelga de nuestro directorio HOME. Este comando recibe siempre como argumento el fichero fuente Pascal (**no admite metacaracteres como parte del nombre**). Al ser ejecutado produce ciertos mensajes por la salida estándar, lee por la entrada estándar las modificaciones a realizar sobre el fichero fuente especificado y efectúa estas modificaciones, dejando el resultado (el programa fuente modificado) en un fichero con el mismo nombre que el original. Por desgracia el tiempo que tarda en efectuarse la modificación es muy elevado, incluso para un único fichero.

Después de haber transferido los ficheros que contienen la información de las modificaciones al directorio `ftp`, hijo de nuestro HOME, entramos a nuestra cuenta, nos situamos en `practicas`, hijo también de HOME, y en ese momento nuestro amigo nos dice que “necesita ya el terminal”. ¿Que haríamos para modificar los programas fuentes Pascal situados en `practicas`? También sería interesante no perder nunca los mensajes de error que se produzcan en el proceso de modificación.

NOTA: los ficheros fuentes Pascal tienen nombres que terminan por .p. Cada fichero fuente Pascal tiene un único fichero con sus modificaciones que tiene el mismo nombre que el fichero fuente, pero termina por .p.modif. Un ejemplo: prog1.p, prog1.p.modif.

11. (Problema de examen) Nos encontramos en un lugar desconocido del sistema de ficheros de un sistema UNIX y queremos instalar una nueva aplicación en el sistema. Dicha aplicación se compone de varios ficheros que se nos han proporcionado empaquetados en dos ficheros independientes `bin.xtr` y `lib.xtr` situados en `/usr/pkgtool.`

Se pide crear un procedimiento Shell que permita la instalación adecuada de dicho programa, realizando las siguientes operaciones:

En primer lugar ha de verificar se el programa en cuestión ya está instalado, para ello se consulta la variable PRGCONFIG; si su valor es distinto de la cadena vacía, significa que dicho programa ya ha sido instalado con anterioridad, y por lo tanto, no es necesario continuar con el programa Shell.

En el caso de que sea necesaria la instalación, lo primero que hay que hacer es extraer los distintos ficheros almacenados en `bin.xtr` y `lib.xtr` con la herramienta `extraer` que se encuentra en un lugar accesible por la variable PATH. Al ser llamada pasándole como argumento un fichero de extensión `.xtr`, extraer los contenidos de dicho fichero, dejando los componentes obtenidos en el directorio actual, desde donde se ejecuta el comando. Ha de tenerse en cuenta que los ficheros empaquetados en `bin.xtr` deben dejarse en `/usr/local/lib`, y los de `lib.xtr` en `/usr/local/lib`.

Una vez que se hayan desempaquetado los ficheros indicados, debe crearse en el directorio HOME del usuario que realiza la instalación un directorio llamado `prog.config`, y copiar en él todos lo ficheros situados en `/usr/local/lib` que contengan la subcadena `cfg` en cualquier posición de su nombre. Además, deben añadirse al fichero `.profile` las líneas necesarias que declaren la variable PRCONFIG, con valor la vía de acceso al directorio que se acaba de crear.

12. (Problema de examen) **Nos encontramos en un directorio desconocido del sistema de ficheros de una máquina UNIX y no nos movemos de ahí, es decir, no se permite utilizar el comando cd.**

Crear un shell script UNIX que reciba al menos un argumento.

Si recibe sólo uno comprobar:
- si es un fichero ordinario pedir al usuario cadenas de caracteres y añadirlas al fichero pasado como argumento. El usuario indicará que no quiere introducir más pulsando Enter.
- Si es un directorio borrarlo (tener en cuenta que puede no estar vacío).
- En caso de que no sea ni un fichero ni un directorio, es decir, sea una cadena de caracteres, se deberá hacer un listado con los nombres de ficheros y directorios del HOME que la contienen en su nombre.

Si recibe más de un argumento, crear los directorios cuyos nombres se pasan como argumentos teniendo en cuenta que el primer directorio debe colgar del padre del actual, y los demás, cada uno del anterior.

13. (Problema de examen) **Nos encontramos en un directorio desconocido del sistema de ficheros de una máquina UNIX, y no nos movemos de ahí.**

Crear un shell script UNIX que reciba al menos un argumento. En caso de no recibir argumentos el programa termina presentando un mensaje por la salida estándar.

Se deberá comprobar que el o los argumentos pasados corresponden a directorios existentes. Si alguno de ellos no existe el programa termina presentando un mensaje de error por la salida estándar.

Concatenar los ficheros que se encuentren dentro de el/los directorios pasados como argumento en un fichero llamado *todo.tmp* que se debe situar en el padre del actual. (Sólo se deben concatenar los ficheros, no los subdirectorios que cuelguen de los directorios pasados).

Pedir al usuario una cadena de caracteres y decir si se encuentra o no en el fichero *todo.tmp*. Si no se encuentra, añadirla a un fichero llamado *cadenas.txt* situado en el HOME.

14. (Problema de examen) **Nos encontramos en un directorio desconocido del sistema de ficheros de un máquina UNIX y no nos movemos de ahí.**

Crear un shell script UNIX que, dependiendo del número de argumentos que reciba, realice unas operaciones u otras.

- En el caso de no recibir argumentos, se enviará por correo electrónico al usuario root de la máquina, el fichero *login.txt* que se encuentra en el directorio padre del actual.

- Si se recibe un argumento, éste será el nombre de un directorio que se halla en el directorio HOME. Se deberá de mostrar un mensaje que indique si ese directorio existe y no está vacío.

- En caso de que haya más de un argumento, se trata de crear en el fichero *extension.txt*, que se debe situar en el directorio raíz, una lista con los nombres de aquellos ficheros cuya extensión viene dada por el primer argumento y que se encuentren en los directorios indicados por el segundo y siguientes argumentos. (El fichero *extensión.txt* debe contener **sólo** nombres de ficheros, tener en cuenta que en los directorios indicados por el 2º y siguientes argumentos, puede haber subdirectorios cuya extensión sea la buscada y sus nombres no deben ser incluidos en el fichero *extensión.txt).*