Ejercicio 11:

Escribir shellscript que, a partir del fichero / etc / passwd, diga cuál es el intérprete de comandos más utilizado por los usuarios del sistema (es decir, aquel que utilizan más usuarios). este shellscript se puede resolver con una única (y larga) comanda y sin utilizar ningún tipo de bucle.

```
#!/bin/bash
echo "$(more /etc/passwd | cut -d: -f7 | grep ^/ | uniq -c | sort -nr | head -1 | tr -s " " " " | cut -d" " -f3)";
```

```
alumne@alumne-VirtualBox:~/Escritorio/shells$ ./ex11.sh
/bin/false
alumne@alumne-VirtualBox:~/Escritorio/shells$
```

Ejercicio 12:

Elabore un shellscript que reciba como parámetros un número indeterminado de palabras en minúscula. Lo que deberá hacer el shellscript será añadir cada una de las palabras a ficheros que llamarán como su inicial.
Por ejemplo, si la ejecutamos con los siguientes parámetros:
./exercici ratón periférico teclado pantalla
En el fichero "r" se añadirá la palabra "ratón", el archivo "p" se añadirá la palabra "periférico", al "t" teclado y al "p", nuevamente, "pantalla".

```
#!/bin/bash
# comprobar el parametro
for i in $*
do
    #muestra por pantalla el parametro
    echo $i
    # se guarda el parametro en un fichero con el mismo nombre
    echo $i > $i
done;

exit 0;
```

Ejercicio 13:

Indicar justificadamente cuál es la función del siguiente shellscript, indicando cuál es el significado más lógico de los parámetros.

```
#!/bin/bash
for i in `sort $1`
do
if grep $2 $i > /dev/null
then
echo A
cp $i /tmp
exit
fi
done
echo
```

Ejercicio 14:

Escribir shellscript que, a partir de la ruta de un directorio determinado, si existe, acceda a este directorio.

● Si no existe, deberá crear este directorio y acceder.

● En caso de no poder crear el directorio, mostrará por pantalla el mensaje "No se pudo crear el directorio "y quedarse en la ruta original.

Resuelva este ejercicio sin utilizar los operadores && y ||, gestionando la casuística con estructuras alternativas.

Para comprobar que su shellscript se comporta de la forma esperada, muestre por pantalla el directorio de trabajo actual. Dónde queda tu terminal? Donde había hecho la ejecución o al nuevo directorio?

```
1   #!/bin/bash
2   echo "Escribe un nombre para el directorio"
3   read dir
4
5   if [[ ! -d "$dir" ]]
6   then
7       if [ -L $dir ]
8       then
9       # cambiar directorio
10      cd "${dir}"
11      else
12      echo "Directorio no existe. Creando ahora"
13      mkdir -p -- "$dir"
14      echo "Directorio creado"
15      fi
16  else
17      echo "Se ha producido un error"
18  fi
19
20  exit 0;
```

```
alumne@alumne-VirtualBox: ~/Escritorio/shells
alumne@alumne-VirtualBox:~/Escritorio/shells$ rm -r /home/alumne/harman
alumne@alumne-VirtualBox:~/Escritorio/shells$ ./ex14.sh
Escribe un nombre para el directorio
/home/alumne/harman
Directorio no existe. Creando ahora
Directorio creado
alumne@alumne-VirtualBox:~/Escritorio/shells$
```

Ejercicio 15:

**Escribir shellscript que compruebe con qué usuario se ha ejecutado. Si este se ha ejecutado como superusuario, aparecerá el mensaje "Soy el dueño del mundo". Si no se ha ejecutado inicialmente como superusuario, se volverá a ejecutar a sí mismo como superusuario.
En ningún punto del shellscript puede aparecer el nombre del propio shellscript, será necesario que utilice algún otro tipo de mecanismo o parámetro para poder ejecutarlo como superusuario. Tenga en cuenta que si se ha ejecutado algún sudo desde el terminal recientemente el resultado puede resultar confuso: cierre y abra el terminal para asegurarse del resultado.**

Ejercicio 16:

Escribir shellscript que se mate a sí mismo (como proceso). Para comprobar que funciona correctamente, coloque alguna instrucción debajo de la finalización del proceso y ver que no se ejecuta.

```
#!/bin/bash
PID=`ps -eaf | grep ex16.sh | grep -v grep | awk '{print $2}'`
if [[ "" !=  "$PID" ]]; then
  echo "killing $PID"
  kill -9 $PID
fi
```

```
alumne@alumne-VirtualBox:~/Escritorio/shells$ ./ex16.sh
killing 3975
3976
Terminado (killed)
alumne@alumne-VirtualBox:~/Escritorio/shells$
```

Ejercicio 17:

Escribir shellscript que a partir de un nombre de archivo ($ 1) y un número de partes ($ 2), rompa el archivo $ 1 en $ 2 partes. Para hacerlo será necesario que utilice el comando que rompe ficheros y alguna operación matemática. Pruebe a romper los archivos en varios números de partes y controle que siempre se genera el número de archivos que se ha indicado, exactamente.
→ Después e

Ejercicio 18:

Escribir shellscript que, dado un archivo (con su ruta absoluta) y el nombre de un paquete, nos indique si el archivo pertenece al paquete o no (es decir, si fue instalado o requerido por el paquete).
→ Después e

Ejercicio 19:

Tiene un fichero de texto donde, cada línea, hay dos nombres de archivo (separados por guión "-"). Escribir shellscript que procese todas las líneas de este fichero efectuando la siguiente operación: si el primer archivo existe, lo copiará sobre el segundo; si no existe, copiará el contenido del fichero / etc / group sobre el segundo.
→ Después e