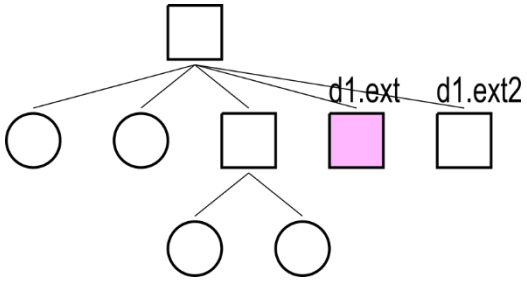
Examen de shell script. Diciembre 2022



Escribir un guión shell que no utilice *sed* ni *awk* denominado `vacios`. `vacios` toma como primer argumento de línea de mandato la ruta de un directorio y como segundo una extensión. Se invocaría, por ejemplo, así:

```
$ vacios /etc ext
```

(**I**) Escriba una función denominada `ddot` que: (**4 puntos**)

1) Toma como primer parámetro la ruta de un fichero de texto que ha de contener una palabra por línea, como por ejemplo el fichero que sigue `period`:

   ```
   binfmt.d
   cron.hourly
   libpaper.d
   opt
   sensors.d
   skel
   tmpfiles.d
   update-notifier
   usb_modeswitch.d
   ```

   Toma como segundo parámetro una extensión, como por ejemplo `d`. Compruebe que recibe dos parámetros. En otro caso la función emite un mensaje en error estándar y devuelve 2.

2) Devuelve 0 si el fichero primer parámetro existe y 1 si no existe.

3) Filtra a salida estándar las líneas del fichero primer parámetro que terminan en la extensión segundo parámetro. Por ejemplo, la invocación

   ```
   ddot period d
   ```

   emite a salida estándar

   ```
   binfmt.d
   libpaper.d
   sensors.d
   tmpfiles.d
   usb_modeswitch.d
   ```

4) Registra en la variable `dot_ext` el número de estas líneas. Por ejemplo, 5 en el caso anterior.

(**II**) El guión: (**4 puntos**)

1) Comprueba que toma dos parámetros y que el primero es un directorio.

2) Produce un archivo denominado *emptyDirs* que contiene los nombres de todos los subdirectorios vacíos del directorio parámetro, uno por línea.

3) Invoca la función `ddot` sobre el fichero *emptyDirs* con segundo parámetro dado por el segundo parámetro del guión.

4) Si el retorno de `ddot` es 1 o 2 avisa en error estándar y termina con error 1. En otro caso escribe la variable `dot_ext` en salida estándar.

(**III**) Ejecución correcta: (**2 puntos**)

Tiempo: 1 hora y 20 minutos

Solución

vacios

```
#! /bin/bash

# (I). Define una función denominada ddot
#
function ddot ()
{
  # (I).1. Compruebe que recibe dos parámetros.
  #         En otro caso la función emite un mensaje en error estándar y devuelve 2.
  if [ $# != 2 ]; then
    echo Usage: ddot file extension
    return 2
  fi
  # (I).2. Devuelve 0 si el fichero primer parámetro existe y 1 si no existe.
  #
  if [ ! -f $1 ]; then
    ret=1
    return $ret
  fi
  # (I).3. Filtra a salida estándar las líneas del fichero primer parámetro
  #         que terminan en la extensión segundo parámetro.
  #
  grep "\.$2$" $1
  # (I).4. Registra en la variable dot_ext el número de estas líneas
  #
  dot_ext=`grep "\.$2$" $1 | wc -w`
  return 0
}

# (II).1. Comprueba que toma dos parámetros y que el primero es un directorio.
#
if [ $# -ne 2 -o  ! -d $1 ]; then
  echo Usage: vacios directory extension 1>&2
  exit 1
fi

# (II).2. Produce un archivo denominado emptyDirs que contiene los nombres
#         de todos los subdirectorios vacíos del directorio parámetro, uno por línea
echo -n > emptyDirs
ls $1 |
while read folder
do
  if [ -d $1/$folder ]
  then
    files=`ls $1/$folder | wc -l`
    # echo $files
    if [ $files -eq 0 ]
    then
      echo $folder >> emptyDirs
    fi
  fi
done

# (II).3. Invoca la función ddot sobre el fichero emptyDirs
#         con segundo parámetro dado por el segundo parámetro del guión
ddot emptyDirs $2

# (II).4. Si el retorno de ddot es 1 o 2 avisa en error estándar y termina con error 1.
#         En otro caso escribe la variable dot_ext en salida estándar
if [ $? -eq 1 ]; then
  echo ddot: The file emptyDirs does not exists 1>&2
  exit 1
elif [ $? -eq 2 ]; then
  echo Usage: %1: ddot file extension 1>&2
  exit 1
fi
echo $dot_ext
```