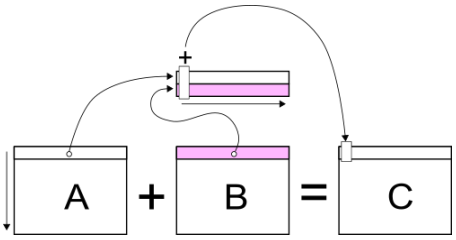
Examen de shell script. Enero 2022



(**I**) Escribir un guión shell denominado *matrixGen* que genere en salida estándar una matriz de enteros aleatorios entre 0 y 9 de tamaño *N*x*M* donde *N* y *M* son los parámetros del guión. Los componentes de cada fila estarán separados por un espacio en blanco. Tenga en cuenta que la expresión `$(($RANDOM % 10))` genera un número aleatorio entre 0 y 9. Se invocaría, por ejemplo, así (**1 punto**)

```
$ ./matrixGen 2 3
8 1 6
6 1 4
```

(**II**) Escribir una función denominada `vectorSum_f` que genere la suma de dos vectores *V1* y *V2* de dimensión *N*. Ambos son ficheros de una única línea de elementos enteros separados por un espacio en blanco. `vectorSum_f` toma *V1* como primer parámetro y *V2* como segundo parámetro y genera el vector suma en el fichero dado como tercer parámetro. Se invocaría, por ejemplo, como (**2 puntos**)

```
vectorSum_f v1 v2 summa
```

Escribir un guión shell denominado *vectorSum* que tome como parámetros dos vectores *V1* y *V2* para generar en salida estándar el vector suma de ambos. Haga uso de la función `vectorSum_f`. Se invocaría, por ejemplo, así (**1 punto**)

```
$ ./matrixGen 1 5 > v1
$ ./matrixGen 1 5 > v2
$ cat v1 v2
7 0 4 3 2
0 4 4 9 4
$ ./vectorSum v1 v2
7 4 8 12 6
```

(**III**) Escribir una función denominada `rowGet_f` que extraiga la fila *r* de una matriz *R* de *M*x*N* enteros separados por un espacio en blanco. `rowGet_f` toma *r* como primer parámetro y toma *R* como segundo parámetro. Extrae la fila en el fichero dado como tercer parámetro. Se invocaría, por ejemplo, como (**2 puntos**)

```
rowGet_f 8 m1 r
```

Escribir un guión shell denominado *rowGet* que use la función `rowGet_f` para extraer una fila de una matriz. *rowGet* toma el número de fila *r* (de 1 en adelante) como primer parámetro y toma la matriz *R* como segundo parámetro. La fila extraída se muestra en salida estándar. Se invocaría, por ejemplo, como (**1 puntos**)

```
$ ./matrixGen 3 5 > m
$ cat m
7 6 6 1 7
4 3 4 7 6
7 3 2 7 9
$ ./rowGet 2 m
4 3 4 7 6
```

(**IV**) Escribir un guión shell denominado *matrixSum* que tome como parámetros dos matrices *M1* y *M2* de *M*x*N* enteros separados por un espacio en blanco para generar en salida estándar la matriz suma de ambas. Haga uso de las funciones `rowGet_f` y `vectorSum_f`. Se invocaría, por ejemplo, así: (**2 puntos**)

```
$ ./matrixGen 3 5 > m1
$ ./matrixGen 3 5 > m2
$ ./matrixSum m1 m2
7 6 6 1 7
4 3 4 7 6
7 3 2 7 9
```

V) Ejecución correcta (**1 punto**)

Entrega: Subir a la tarea del campus virtual los guiones shell *matrixGen*, *vectorSum*, *rowGet* y *matrixSum*.
Tiempo: 1 hora y 20 minutos.

Solución

matrixGen

```
#! /bin/bash
if [ $# -ne 2 ]; then
  echo Usage: matrixGen N M 1>&2
  exit 1
fi

for (( i = 0; i < $1; i++ )); do
  for (( j = 0; j < $2; j++ )); do
    echo -n "$(($RANDOM % 10)) "
  done
  echo
done
```

vectorSum

```
#! /bin/bash

function vectorSum_f ()
{
  cat $1 $2 > pair

  read fila < pair                    # Determine the number of columns
  nCol=0
  for i in $fila ; do
    nCol=`expr $nCol + 1`
  done

  echo -n > $3
  i=1
  while [ $i -le $nCol ]; do
    cut -d" " -f$i pair > columnFile  # Extract the column
    totalColumn=0                     # Run through the column
    while read number; do
      totalColumn=`expr $totalColumn + $number`
    done < columnFile
    echo -n "$totalColumn " >> $3
    i=`expr $i + 1`
  done
  echo >> $3
  return
}

if [ $# -ne 2 ]; then
  echo Usage: vectorSum v1 v2 1>&2
  exit 1
fi

vectorSum_f $1 $2 summa.$$
cat summa.$$
rm summa.$$
```

rowGet

```
#! /bin/bash

function rowGet_f ()
{
  rowNr=$1
  matrix=$2
  row=$3
  tail -n +$rowNr < $matrix | head -n1 > $row
  return
}

if [ $# -ne 2 ]; then
  echo Usage: rowGet row matrix 1>&2
  exit 1
fi

rowGet_f $1 $2 row.$$
cat row.$$
rm row.$$
```

matrixSum

```
#! /bin/bash

function rowGet_f ()
{
```

```
  ...
}

# Generates the sum of vectors V1 and V2 of dimension N
#
function vectorSum_f ()
{
  ...
}

if [ $# -ne 2 ]; then
  echo Usage: matrixSum Matrix_1 Matrix_2 1>&2
  exit 1
fi
matrix_1=$1
matrix_2=$2

#
# Determine the number of rows of the matrices in M
M=0
while read mRow
do
  M=`expr $M + 1`
done < $matrix_1

#
# Run through both matrices
for(( ii=1; ii <= $M; ii++ ))
do
  rowGet_f $ii $matrix_1 row_1.$$
  rowGet_f $ii $matrix_2 row_2.$$
  vectorSum_f row_1.$$ row_2.$$ vSum
  cat vSum
  rm row_1.$$ row_2.$$ vSum
done
rowGet_f $1 $2 row.$$
cat row.$$
rm row.$$
```