Shell

Exam 2020/06/16 - Ex 4 (3.0 points)

Italiano

Dato un file di nome "file.txt" contenente delle parole (cioè delle sequenze di caratteri alfabetici separati da una spazio, anche su più righe) con un contenuto simile al seguente:

```
one two three
four five six
seven
```

Realizzare uno script bash che stampi la parola con la lunghezza maggiore.

Dato l'esempio precedente, lo script dovrà produrre in output la parola "three" o la parola "seven" perché entrambe hanno 5 caratteri.

English

Given a file named "file.txt" containing some words (i.e., sequences of alphabetic characters separated by spaces, even on multiple lines) with content similar to the following:

```
one two three
four five six
seven
```

Implement a bash script that prints the word with longer length.

Given the previous example, the script must print in output the word "three" or the word "seven", because both have 5 characters.

Risposta: Answer:

```
#!/bin/bash
# Exam 2020/06/16 - Exercise 4

longest_word=""
length_longest=0
# Scan the file word by word
for word in $(cat file.txt) do
	# Detect the length of $word
	length=$(echo $word|wc -c)
	#length=${#word} # Other possibility
	if [ $length -gt $length_longest ] then
		length_longest=$length
		longest_word=$word
	fi
done
echo $longest_word
```

Exam 2021/06/18 - Ex 8 (4.5 points)

Italiano

Scrivere uno script bash in grado di calcolare la somma delle lunghezze delle parole presenti sulla diagonale di una matrice quadrata che contiene solo stringhe.

La matrice quadrata viene memorizzata in un file di testo, specificato come parametro da linea di comando. Verificare il corretto passaggio di tale parametro.

English

Write a bash script able to compute the sum of lengths of the words appearing on the diagonal of a square matrix that contains just strings.

The square matrix is stored in a text file, specified as a command line argument to the script. Check that such an argument is correctly passed to the script.

Example

```
word11 word12 word13 word14
word21 word22 word23 word24
word31 word32 word33 word34
word41 word42 word43 word44
```

Result: 24

Risposta: Answer:

```
#!/bin/bash
################################################################################
Exercise 11 of exam 18/06/2021 # # Version A #
################################################################################
Check correct number of arguments passed
if [ $# -ne 1 ]; then
      echo "Usage: ./ex11a.sh "
      exit 1
fi

# Check input file existence
if [ ! -e $1 ]; then
      echo "File $1 not found!"
      exit 1
fi

# Scan matrix and sum lengths
i=0
j=0
tot=0
while read line; do
      for word in $line; do
            if [ $i -eq $j ]; then
                  l=$(echo -n $word | wc -m)
                  let "tot+=$l"
            fi
            let "j+=1"
      done
      let "j=0"
      let "i+=1"
done < $1
# Print result
echo "Result: $tot"


#!/bin/bash
################################################################################
Exercise 11 of exam 18/06/2021 # # Version B #
################################################################################
Check correct number of arguments passed
if [ $# -ne 1 ]; then
      echo "Usage: ./ex11a.sh "
      exit 1
fi

# Check input file existence
```

```
if [ ! -e $1 ]; then
      echo "File $1 not found!"
      exit 1
fi

# Retrieve square matrix dimension
N=$(cat $1 | wc -l)

# Compute sum of lengths
tot=0
for((i=1; i<=N; i++)); do
      word=$(head -n $i $1 | tail -n 1 | tr -s " " | cut -d " " -f $i)
      l=$(echo -n $word | wc -m)
      let "tot+=$l"
done
# Print result
echo "Result: $tot"


#!/bin/bash
################################################################################
Exercise 11 of exam 18/06/2021 # # Version C #
################################################################################
Check correct number of arguments passed
if [ $# -ne 1 ]; then
      echo "Usage: ./ex11a.sh "
      exit 1
fi

# Check input file existence
if [ ! -e $1 ]; then
      echo "File $1 not found!"
      exit 1
fi

# Scan matrix and sum lengths
i=1
tot=0
while read line; do
      word=$(echo -n $line | tr -s " " | cut -d " " -f $i)
      l=$(echo -n $word | wc -m)
      let "tot+=$l"
      let "i+=1"
done < $1
# Print result
echo "Result: $tot"
```

Exam 2021/09/07 - Ex 12 (5.0 points)

Italiano

Scrivere uno script BASH che riceva come argomenti un elenco di file e, per ogni file, se esistente, stampi le informazioni seguenti:

1. Se il file è un file regolare, stampa il nome, la dimensione e indica se l'utente che esegue lo script dispone delle autorizzazioni di lettura e di scrittura sul file.
2. Se il file è un direttorio, stampa il nome e il numero di sottodirettori in esso contenuti.

Si noti che l'output del comando `ls -l` ha il seguente formato:

```
drwx------ 12 user user 408 Oct 30 19:09 Desktop
-rw-r--r-- 1 user user 192 Jul 13 00:03 pgrm
-rwxr-xr-x 1 user user 74 Nov 3 10:02 ex.awk
drwxrwxrwx 22 user user 408 Oct 30 12:09 tmp
```

English

Write a BASH script that receives as arguments a list of files, and for each file, if it exists, it prints the following information: 1. If the file is a regular file, it prints its name, its dimension, and if the user that runs the script has read and write permissions for the file. 2. If the file is a directory, it prints its name, and how many sub-directories it contains. Notice that the output of command `ls -l` is like the following one:

```
drwx------ 12 user user 408 Oct 30 19:09 Desktop
-rw-r--r-- 1 user user 192 Jul 13 00:03 pgrm
-rwxr-xr-x 1 user user 74 Nov 3 10:02 ex.awk
drwxrwxrwx 22 user user 408 Oct 30 12:09 tmp
```

Risposta: Answer:

```
#!/bin/bash
################################################################################
Exercise 15 of exam 07/09/2021 - Version A #
################################################################################

for path in $*; do
      if [ -f $path ]; then
            dim=$(ls -l $path | tr -s " " | cut -d " " -f 5)
            read_perm=""
            if [ -r $path ]; then
                  read_perm="R"
            fi
            write_perm=""
            if [ -w $path ]; then
                  write_perm="W"
            fi
            echo "FILE: $path $dim $read_perm $write_perm"
      elif [ -d $path ]; then
            subdirs=$(find $path -maxdepth 1 -mindepth 1 -type d | wc -l)
            echo "DIR: $path $subdirs"
      fi
done

#!/bin/bash
################################################################################
Exercise 15 of exam 07/09/2021 - Version B #
################################################################################for
r path in $@; do
      if [ -f $path ]; then
            dim=$(ls -l $path | tr -s " " | cut -d " " -f 5)
            read_perm=""
            if [ -r $path ]; then
                  read_perm="R"
            fi
            write_perm=""
            if [ -w $path ]; then
                  write_perm="W"
            fi
            echo "FILE: $path $dim $read_perm $write_perm"
      elif [ -d $path ]; then
```

```
            subdirs=$(ls -l $path | grep -e "^d" | wc -l)
            echo "DIR: $path $subdirs"
      fi
done
```

Exam 2020/09/14 - Ex 10 (4.0 points)

Italiano

Realizzare uno script bash che richieda di introdurre il nome di un file contenente caratteri alfabetici, e fornisca errore nel caso in cui lo script sia lanciato con un numero non corretto di parametri. Il programma dovrà trasformare in lettere maiuscole le righe con un numero pari di parole ed in lettere minuscole le righe con un numero dispari di parole.

Ad esempio, se il file "file.txt" ha il seguente contenuto:

```
Nel mezzo del
Cammin di
Nostra vita mi ritrovai per una
Selva oscura
```

L'output del comando

```
> ./prog.sh file.txt
```

dovrà essere il seguente:

```
nel mezzo del
CAMMIN DI
NOSTRA VITA MI RITROVAI PER UNA
SELVA OSCURA
```

In particolare la prima riga è stata trasformata in minuscolo perché composta da 3 parole, mentre le restanti righe sono state trasformate in maiuscolo perché composte da 2, 6 e 2 parole, rispettivamente.

English

Produce a bash script that requires in input the name of a file containing alphabetic characters, and raises an error if the script is run with an incorrect number of parameters. The program must transform lines with an even number of words into capital letters, and lines with an odd number of words into lowercase letters.

For instance, if the file "file.txt" has the following content:

```
Nel mezzo del
Cammin di
Nostra vita mi ritrovai per una
Selva oscura
```

The output of the command:

```
> ./prog.sh file.txt
```

must be the following:

```
nel mezzo del
CAMMIN DI
NOSTRA VITA MI RITROVAI PER UNA
SELVA OSCURA
```

In particular, the first line has been transformed into lowercase letters because it consists of 3 words, while the remaining lines have been transformed into uppercase letters because they consist of 2, 6 and 2 words, respectively.

Risposta: Answer:

```
#!/bin/bash
# Exam 2020/09/14 - Exercise 10

# Check correct number of parameters
if [ $# -lt 1 ]; then
      echo "Usage: ./prog.sh "
      exit 1
fi
```

```
# Read file line by line
while read line; do
	# Compute number of words in line
	n=$(echo $line | wc -w)
	# If line has an odd number of words convert it to lower case
	if [ $[$n%2] -eq 1 ]; then
		echo $line | tr [A-Z] [a-z]
	fi
	# If line has an even number of words convert it to upper case
	if [ $[$n%2] -eq 0 ]; then
		echo $line | tr [a-z] [A-Z]
	fi
done < $1
```

Exam 2022/02/10 - Ex 3 (4.0 points)

Italiano

Scrivere uno script BASH di nome parse che prenda i seguenti parametri da linea di comando:
parse fileIn str1 str2 N str3 fileOut
Lo script deve leggere fileIn e scriverne una copia (di nome fileOut) effettuando le seguenti operazioni:

- Rimpiazzare ogni occorrenza della stringa str1 con la stringa str2.
- Aggiungere un padding destro e sinistro ad ogni riga: ogni riga del file di output deve iniziare e finire con N ripetizioni della stringa str3 (separate da un singolo spazio), come riportato in seguito.

Si supponga che nessuna delle stringhe str1, str2 e str3 contenga spazi bianchi. Lo script deve inoltre verificare che tutti i parametri attesi siano passati da linea di comando.

English

Write a BASH script called parse that takes the following parameters from the command line:
parse fileIn str1 str2 N str3 fileOut
The script must read fileIn and write a copy of it (called fileOut) performing the following two operations:

- Replace all instances of string str1 with string str2.
- Perform a left and right padding of each line: each line of the output file should begin and end with N repetitions of string str3 (separated by a single space), as reported in the following.

Suppose that none of the strings str1, str2 and str3 contains whitespaces. The script should also check that all the expected parameters are passed via the command line.

str3 str3 str3 … (N times) **<line 1 of the file fileOut>** str3 str3 str3 … (N times)
str3 str3 str3 … (N times) **<line 2 of the file fileOut>** str3 str3 str3 … (N times)
…
str3 str3 str3 … (N times) **<last line of the file fileOut>** str3 str3 str3 … (N times)

Risposta: Answer:

```
# Solution 1
#!/bin/bash

# Check corret number of parameters passed
if [ $# -lt 6 ]; then
    echo "Usage: parse fileIn str1 str2 N str3 fileOut"
    exit 1
fi

# Check input file exists (optional)
if [ ! -e $1 ]; then
    echo "Input file $1 does not exists"
    exit 1
```

```
fi

# Collect input parameters
fileIn=$1
str1=$2
str2=$3
N=$4
str3=$5
fileOut=$6

# Read file line-by-line
while read line; do

    # Print padding left
    i=0
    while [ $i -lt $N ]; do
        echo -n "$str3 "
        let i=i+1
    done

    # Print line subsituting str1 for str2
    for word in $line; do
        if [ $word == $str1 ]; then
            echo -n "$str2 "
        else
            echo -n "$word "
        fi
    done

    # Print padding right
    i=0
    while [ $i -lt $N ]; do
        echo -n "$str3 "
        let i=i+1
    done

    # Print newline
    echo ""
done < $fileIn > $fileOut
```

 # Solution 2

```
#!/bin/bash

# Check corret number of parameters passed
if [ $# -lt 6 ]; then
    echo "Usage: parse fileIn str1 str2 N str3 fileOut"
    exit 1
fi

# Check input file exists (optional)
if [ ! -e $1 ]; then
    echo "Input file $1 does not exists"
    exit 1
fi

# Collect input parameters
```

```
fileIn=$1
str1=$2
str2=$3
N=$4
str3=$5
fileOut=$6

# Read file line-by-line
while read line; do

    # Print padding left
    for i in $(seq 0 $N); do
        echo -n "$str3 "
    done

    # Print line subsituting str1 for str2
    for word in $line; do
        if [ $word == $str1 ]; then
            echo -n "$str2 "
        else
            echo -n "$word "
        fi
    done

    # Print padding right
    for i in $(seq 1 $N); do
        echo -n "$str3 "
    done

    # Print newline
    echo ""
done < $fileIn > $fileOut
```

 # Solution 3

```
#!/bin/bash

# Check corret number of parameters passed
if [ $# -lt 6 ]; then
    echo "Usage: parse fileIn str1 str2 N str3 fileOut"
    exit 1
fi

# Check input file exists (optional)
if [ ! -e $1 ]; then
    echo "Input file $1 does not exists"
    exit 1
fi

# Collect input parameters
fileIn=$1
str1=$2
str2=$3
N=$4
str3=$5
fileOut=$6
```

```
# Read file line-by-line
while read line; do

    # Print padding left
    for((i=0; i<$N; i++)); do
        echo -n "$str3 "
    done

    # Print line subsituting str1 for str2
    for word in $line; do
        if [ $word == $str1 ]; then
            echo -n "$str2 "
        else
            echo -n "$word "
        fi
    done

    # Print padding right
    for((i=0; i<$N; i++)); do
        echo -n "$str3 "
    done

    # Print newline
    echo ""
done < $fileIn > $fileOut
```

 # Solution 4

```
#!/bin/bash

# Check correct number of parameters passed
if [ $# -lt 6 ]; then
    echo "Usage: parse fileIn str1 str2 N str3 fileOut"
    exit 1
fi

# Check input file exists (optional)
if [ ! -e $1 ]; then
    echo "Input file $1 does not exists"
    exit 1
fi

# Collect input parameters
fileIn=$1
str1=$2
str2=$3
N=$4
str3=$5
fileOut=$6

# Compose padding
i=0
padding=""
while [ $i -lt $N ]; do
    padding=$padding"$str3 "
    let i=i+1
done
```

```
# Read file line-by-line
while read line; do

    # Print padding left
    echo -n "$padding"

    # Print line subsituting str1 for str2
    for word in $line; do
        if [ $word == $str1 ]; then
            echo -n "$str2 "
        else
            echo -n "$word "
        fi
    done

    # Print padding right
    echo "$padding"
done < $fileIn > $fileOut
```

Exam 2022/09/06 - Ex 3 (5.0 points)

Italiano

Il comando **`ps -aux`** in Linux visualizza le seguenti informazioni:

```
USER  PID %CPU %MEM    VSZ   RSS    TTY    STAT START  TIME COMMAND
root    1   0.0  0.0  167728  11568  ?      Ss   10:38  0:01 /sbin/init splash
root  223   0.0  0.1  109036  53488  ?      S<s  10:38  0:00 /lib/systemd/systemd
quer  1299 3.9  1.0  3772520 425772 ?      Rsl  10:38  3:50 /usr/bin/gnome-shell
quer  2004 0.2  0.1  824376  60760  ?      Ssl  10:46  0:15 /usr/lib/gnome-terminal
quer  4755 0.0  0.0  11696   3456   pts/0  R+   12:15  0:00 ps -aux
```

Scrivere uno script BASH che analizzi l'output di tale comando e invii un segnale di terminazione a tutti i processi **non** di root per cui l'utilizzo della memoria (`%MEM`) supera il 25%. Si assuma che le colonne dell'output siano separate da un singolo spazio.

English

The command **`ps -aux`** in Linux displays the following information:

```
USER  PID %CPU %MEM    VSZ   RSS    TTY    STAT START  TIME COMMAND
root    1   0.0  0.0  167728  11568  ?      Ss   10:38  0:01 /sbin/init splash
root  223   0.0  0.1  109036  53488  ?      S<s  10:38  0:00 /lib/systemd/systemd
quer  1299 3.9  1.0  3772520 425772 ?      Rsl  10:38  3:50 /usr/bin/gnome-shell
quer  2004 0.2  0.1  824376  60760  ?      Ssl  10:46  0:15 /usr/lib/gnome-terminal
quer  4755 0.0  0.0  11696   3456   pts/0  R+   12:15  0:00 ps -aux
```

Write a BASH script that analyzes the output of such a command and sends a termination signal to all the **non**-root processes that have a memory usage (`%MEM`) larger than 25%. Assume that the columns are separated from each other by a single space.

Risposta 1 / Answer 1

```
#!/usr/bin/env bash

################################################################################
# Version 1: temp file, while-read, cut
################################################################################

# Save ps output to a temporary file
ps -aux | tail -n +2 > tmp.txt

# Process the file line-by-line
```

```
while read line; do

  # Extract user, pid and memory percentage from the current line
  user=$(echo $line | tr -s " " | cut -d " " -f 1)
  pid=$(echo $line | tr -s " " | cut -d " " -f 2)
  mem=$(echo $line | tr -s " " | cut -d " " -f 4 | cut -d "." -f 1)

  # Send a termination signal if the process is non-root and has memory usage larger
than 25%
  if [ $user != "root" ] && [ $mem -ge 25 ]; then
      kill -9 $pid
  fi
done < tmp.txt

# Remove temporary file
rm tmp.txt
```

Risposta 2 / Answer 2

```
#!/usr/bin/env bash

################################################################################
# Version 2: pipelining, greps, xargs
################################################################################

ps -aux | tr -s " " | grep -ve "^root" | grep -E "^[^ ]+ [^ ]+ [^ ]+
(100|[3-9][0-9]|2[6-9]|25\.[1-9])" | cut -d ' ' -f 2 | xargs -n 1 echo
```