Introduction to Unix shell - ANSWERS TO EXERCISES

- 2015/16 Part III Systems Biology SEB module
- 10 Feb 2016, 10:00-13:00
- Bioinformatics Training Room, Craik-Marshall Building, Downing Site
- Avazeh Ghanbarian, Alexey Morgunov

Answers to exercises

1. Write a shell script that allows a user to enter his or her top three ice cream flavors. Your script should then print out the name of all three flavors.

```
#!/bin/sh
read -p "Enter your three ice cream flavors : " ice1 ice2 ice3
echo "Thanks $USER!"
echo "1# ${ice1}"
echo "2# ${ice2}"
echo "3# ${ice3}"
```

2. Write a shell script that allows a user to enter any existing file name. The program should then copy file to /tmp directory. An advanced solution would check whether the file exists.

```
#!/bin/sh
# Version 1 (blind copy)
read -p "Enter any file name : " filename
cp $filename /tmp
```

OR

```
#!/bin/sh
# Version 2 (first check for $filename and than copy it, else
display an error message)
read -p "Enter any file name : " filename
```

```
# if file exists, than copy it
if [ -f "${filename}" ]
then
cp -v "$filename" /tmp
else
echo "$0: $filename not found."
fi
```

3. Write a simple shell script where the user enters a pizza parlor bill total. Your script should then display a 10 percent tip.

```
#!/bin/sh
clear
echo "************************"
echo "*** Joes Pizza Parlor ***"
echo "************************"
echo
echo "Today is $(date)"
echo
read -p "Enter a pizza parlor bill : " bill

tip=$(echo "scale=2; (${bill}*10) / 100" | bc -l)
total=$(echo "scale=2; $tip + $bill" | bc -l)
echo "Pizza bill : $bill"
echo "Tip (10%)  : ${tip}"
echo "-------------------------"
echo "Total      : ${total}"
echo "-------------------------"
```

4. Write a simple calculator program that allows user to enter two numeric values and operand. The program should then print out the result of the operation on the two numbers. Make sure it works according to entered operand.

```
#!/bin/sh
read -p "Enter two values : " a b
read -p "Enter operand ( +, -, /, *) : " op
ans=$(( $a $op $b ))
echo "$a $op $b = $ans"
```

5. Write a shell script that, given a file name as the argument will count blank spaces, characters, number of line and symbols. The advanced version would also count the number of vowels.

```
#!/bin/sh
file=$1
v=0

if [ $# -ne 1 ]
then
echo "$0 fileName"
exit 1
fi
if [ ! -f $file ]
then
echo "$file not a file"
exit 2
fi

#advanced starts
while read -n 1 c
do
l=$(echo $c | tr [:upper:] [:lower:])
[[ "$l" == "a" || "$l" == "e" || "$l" == "i" || "$l" == "o" ||
"$l" == "u" ]] && (( v++ ))
done < $file

echo "Vowels : $v"
#advanced ends
echo "Characters : $(cat $file | wc -c)"
echo "Blank lines : $(grep  -c '^$' $file)"
echo "Lines : $(cat $file|wc -l )"
```

6. Write a shell script that, given a file name as the argument will write the even numbered line to a file with name `evenfile` and odd numbered lines in a text file called `oddfile`.

```
#!/bin/sh
file=$1
```

```
counter=0

if [ $# -ne 1 ]
then
echo "$0 fileName"
exit 1
fi
if [ ! -f $file ]
then
echo "$file not a file"
exit 2
fi

while read line
do
((counter++))
EvenNo=$(( counter%2 ))

if [ $EvenNo -eq 0 ]
then
 echo $line >> evenfile
else
 echo $line >> oddfile
fi

done < $file
```

7. Write a shell program to read a number (such as 123) and find the sum of digits (1+2+3=6).

```
#!/bin/sh

#store the no
num=$1

#store the value of sum
sum=0

if [ $# -ne 1 ]
```

```
then
echo "$0 number"
exit 1
fi

while [ $num -gt 0 ]
do
digit=$( num%10 ))
num=$( num/10 ))
sum=$( digit+sum ))
done

echo "Sum of digits = $sum"
```

8. Write a shell program to read two numbers and display all the odd numbers between those two numbers.

```
#!/bin/bash
# Shell program to read two numbers and display all the odd

echo -n "Enter first number : "
read n1

echo -n "Enter second number : "
read n2

if [ $n2 -gt $n1 ];
then
for(( i=$n1; i<=$n2; i++ ))
do
  # see if it is odd or even number
  test=$(( $i % 2 ))
  if [ $test -ne 0 ];
  then
    echo $i
  fi
done
else
echo "$n2 must be greater than $n1, try again..."
```

```
fi
```

9. Right click and save the `shakespeare.txt` exercises/shakespeare.txt file (all works of Shakespeare as text). Process it to output a list of words with frequency counts. Be careful with counting capitalised and non-capitalised words separately, and take care of the apostrophe!

```
tr -sc "A-Za-z\'" '\n' < shakespeare.txt | tr '[:upper:]'
'[:lower:]' | sort | uniq -c | sort -k 1n
```

10. Working with the file from 9, find the most common bigrams Shakespeare uses. Trigrams?

```
#bigrams
tr -sc "A-Za-z\'" '\n' < shakespeare.txt > sh.words
tail -n +2 sh.words > sh.nextwords
paste sh.words sh.nextwords > sh.bigrams
tr 'A-Z' 'a-z' < sh.bigrams | sort | uniq -c | sort -k 1n
#trigrams
tail -n +3 sh.words > sh.thirdwords
paste sh.words sh.nextwords sh.thirdwords > sh.trigrams
cat sh.trigrams | tr "[:upper:]" "[:lower:]" | sort | uniq -c |
sort -k 1n
```

Yeah, the file isn't filtered for the copyright notice. Can you do that?

License