Operating Systems and Networking Journal

Alexander Roque Rodrigues

October 19, 2020

Contents

1 Practice Lab Session for 25-09-2020

1.1 Write a program that prints your favorite movie name. It should print director name on the next line.

```
#Write a program that prints your favorite movie name. It should print director name on the next line.

movieName="Transformers"
director="Michael Bay"

echo $movieName
echo $director
```

1.2 Write a shell script that prints out your name and waits for the user to press the [Enter] key before the script ends.

```
#Write a shell script that prints out your name and waits for the user to press the [Enter] key before the script ends.

myName="Alexander Rodrigues"
echo $myName

# read blank value from keyboard.
read

# exit
exit
```

1.3 List 10 builtin and external commands.

```
#List 10 builtin and external commands.

echo "Built-in Commands."
type alias
type bg
type bind
type break
type builtin
type case
type cd
type command
type compgen
type complete
type continue
type declare


echo "External Commands"
type systemd-ask-password
```

```
type which
type busybox
type date
type gzip
type mkdir
type ntfsfallocate
type red
type systemd-escape
type whiptail
type bzcat
type dd
```

1.4 Make a backup of existing variable called PS1 to OLDPS1. Set PS1 to $. Reset your prompt using OLDPS1 variable.

```
#Make a backup of existing variable called PS1 to OLDPS1. Set PS1 to '£'.
↪  Reset your prompt using OLDPS1 variable.

OLDPS1=$PS1
PS1='$'

read

PS1=$OLDPS1

read
exit
```

1.5 Customize your bash prompt by setting PS1 variable to ”I Love Scripting”.

```
#Customize your bash prompt by setting PS1 variable to 'I Love Scripting '.

PS1="I Love Scripting "
```

1.6 Edit your .bashrc file and set your new PS1 variable.

```
# Edit your £HOME/.bashrc file and set your new PS1 variable.

sudo echo "PS1=\"I Love Scripting\"" >> /home/alexander/.bashrc
```

1.7 Write a command to display the environment.

```
# Write a command to display the environment.

printenv
```

1.8 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.

```
# 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.

read -p "Enter Flavours: " iceCreamFlavour1 iceCreamFlavour2 iceCreamFlavour3

echo "Thank you $USER!"
echo "1: ${iceCreamFlavour1}"
echo "2: ${iceCreamFlavour2}"
echo "3: ${iceCreamFlavour3}"
```

1.9 Write a shell script that allows a user to enter any Internet domain name (host name such as www.cyberciti.biz). Your script should than print out the IP address of the Internet domain name.

```
# Write a shell script that allows a user to enter any Internet domain name (host name such as www.cyberciti.biz). Your script should than print out the IP address of the Internet domain name.

read -p "Enter domain name : " userDomainName
host "${userDomainName}"
```

1.10 Write a shell script that allows a user to enter any existing file name. The program should then copy file to /tmp directory.

```
# Write a shell script that allows a user to enter any existing file name. The program should then copy file to /tmp directory.

read -p "Enter any file name : " filename
cp $filename /tmp
```

1.11 Write a shell script that allows a user to enter directory name. The program should then create directory name in /tmp directory.

```
# Write a shell script that allows a user to enter directory name. The program should then create directory name in /tmp directory.

read -p "Enter the name of a directory: " directoryName
mkdir "/tmp/${directoryName}"
```

1.12 Write a shell script that allows a user to enter three file names. The program should then copy all files to USB pen.

```
# Write a shell script that allows a user to enter three file names. The program should then copy all files to USB pen.
usbLocation="/media/usb"

read -p "Enter 3 file names : " fileOneLocation fileTwoLocation fileThreeLocation
cp -v "$fileOneLocation" "$fileTwoLocation" "$fileThreeLocation" $usbLocation
```

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

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

clear
echo
echo "$(date)"
echo
read -p "Enter bill : " bill

tip=$(echo "(${bill}*10) / 100" | bc -l)
total=$(echo "scale=2; $tip + $bill" | bc -l)

echo "Bill : $bill"
echo "Tip (10%)  : ${tip}"
echo "--------------------------"
echo "Total      : ${total}"
echo "--------------------------"
```

1.14 Write a simple calculator program that allows user to enter two numeric values and operand as follows. The program should then print out the sum of two numbers. Make sure it works according to entered operand.

```
# Write a simple calculator program that allows user to enter two numeric values and operand as follows. The program should then print out the sum of two numbers. Make sure it works according to entered operand.

read -p "Enter two values : " operandA operandB
read -p "Enter operand ( +, -, /, *) : " operation

ans=$(( $operandA $operation $operandB ))

echo "$operandA $operation $operandB = $ans"
```

1.15 Write a shell script that display one of ten unique fortune cookie message, at random each it is run.

```
# Write a shell script that display one of ten unique fortune cookie message, at random each it is run.
r=$(( $RANDOM%10+0 ))

# Quotes author name
author="\t --Alexander Rodrigues"

# Store cookies or quotes in an array
array=(
        "Awesome Quotes 1"
        "Awesome Quotes 2"
        "Awesome Quotes 3"
        "Awesome Quotes 4"
        "Awesome Quotes 5"
        "Awesome Quotes 6"
        "Awesome Quotes 7"
        "Awesome Quotes 8"
        "Awesome Quotes 9"
        "Awesome Quotes 10"
        )

# Display a random message
echo
echo ${array[$r]}
echo -e "$author"
echo
```

1.16 Write a shell script that, given a file name as the argument will count vowels, blank spaces, characters, number of line and symbols.

```
# Write a shell script that, given a file name as the argument will count vowels, blank spaces, characters, number of line and symbols.

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
```

```
16
17  # read vowels
18  exec 3<&0
19  while read -n 1 c
20  do
21    l="$(echo $c | tr '[A-Z]' '[a-z]')"
22    [ "$l" == "a" -o "$l" == "e" -o "$l" == "i" -o "$l" == "o" -o "$l" == "u" ]
      ↪  && (( v++ )) || :
23  done < $file
24
25  echo "Vowels : $v"
26  echo "Characters : $(cat $file | wc -c)"
27  echo "Blank lines : $(grep  -c '^$' $file)"
28  echo "Lines : $(cat $file|wc -l )"
```

1.17 Write a shell script that, given a file name as the argument will count English language articles such As 'A', 'An' and 'The'.

```
1   #Write a shell script that, given a file name as the argument will count
    ↪  English language articles such As 'A', 'An' and 'The'.
2
3   echo -n "Enter a file name : "
4   read file
5
6   # variables to store the count of articles.
7   a=0
8   the=0
9   an=0
10
11  # making sure the file exists
12  if  [ ! -f $file ]
13  then
14          echo "$file not a file!"
15          exit 1
16  fi
17
18  # put while loop to read a £file
19  while read line
20  do
21         #process each word
22          for w in $line
23          do
24                  # convert word to lowercase; so that we can count ThE, THE,
                    ↪  the, THe etc all
25                  lword="$(echo $w | tr '[A-Z]' '[a-z]')"
26
27                  # is it 'a' article?
28                  [ $lword = "a" ] && (( a++ )) || :
29                  [ $lword = "the" ] && (( the++ )) || :
30                  [ $lword = "an" ] && (( an++ )) || :
```

```
        done
done < $file

# display stats
echo "a: $a"
echo "the: $the"
echo "an: $an"
```

1.18 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.

```
# 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.

file=$1
counter=0

eout="evenfile.$$" # even file name
oout="oddfile.$$" # odd file name

if [ $# -eq 0 ]
then
        echo "$(basename $0) file"
        exit 1
fi

if [ ! -f $file ]
then
        echo "$file not a file"
        exit 2
fi

while read line
do
        # find out odd or even line number
        isEvenNo=$( expr $counter % 2 )

        if [ $isEvenNo -ne 0 ]
        then
                # even match
                echo $line >> $eout
        else
                # odd match
                echo $line >> $oout
        fi
        # increase counter by 1
        (( counter ++ ))
done < $file
```

```
echo "Even file - $eout"
echo "Odd file - $oout"
```

1.19 Write a shell script to monitor Linux server disk space using a while loop. Send an email alert when percentage of used disk space is greater than or equal to 90%.

```
# Write a shell script to monitor Linux server disk space using a while loop. Send an email alert when percentage of used disk space is >= 90%.

df -Ph | grep -vE '^Filesystem|tmpfs|cdrom' | awk '{ print $5,$1 }' | while read output;

do
  echo $output
  used=$(echo $output | awk '{print $1}' | sed s/%//g)
  partition=$(echo $output | awk '{print $2}')
  if [ $used -ge 60 ]; then
  echo "The partition \"$partition\" on $(hostname) has used $used% at $(date)" | mail -s "Disk Space Alert: $used% Used On $(hostname)" daygeek@gmail.com
  fi
done
```

1.20 Write a shell script to determine if an input number is a palindrome or not.

```
# Write a shell script to determine if an input number is a palindrome or not. A palindromic number is a number where the digits, with decimal representation usually assumed, are the same read backwards, for example, 58285.

echo -n "Enter number : "
read n

# store single digit
sd=0

# store number in reverse order
rev=""

# store original number
on=$n

while [ $n -gt 0 ]
do
    sd=$(( $n % 10 )) # get Remainder
    n=$(( $n / 10 ))  # get next digit
    # store previous number and current digit in reverse
```

```
    rev=$( echo ${rev}${sd} )
done

if [ $on -eq $rev ];
then
  echo "Number is palindrome"
else
  echo "Number is NOT palindrome"
fi
```

1.21 Write a shell program to read a number and find the sum of digits.

```
# Write a shell program to read a number *such as 123) and find the sum of digits (1+2+3=6).
# sum of all digits - shell script

echo "Enter a number"
read num

sum=0

while [ $num -gt 0 ]
do
    mod=`expr $num % 10`    #It will split each digits
    sum=`expr $sum + $mod`  #Add each digit to sum
    num=`expr $num / 10`    #divide num by 10.
done

echo $sum
```

1.22 Write a shell program to read a number and display reverse the number.

```
#Write a shell program to read a number and display reverse the number. For example, 123 should be printed as as 321.

echo -n "Enter number : "
read n

# store single digit
sd=0

# store number in reverse order
rev=""

# store original number
on=$n

# use while loop to caclulate the sum of all digits
```

```
while [ $n -gt 0 ]
do
    sd=$(( $n % 10 )) # get Remainder
    n=$(( $n / 10 ))  # get next digit
    # store previoues number and current digit in rev
    rev=$( echo ${rev}${sd} )
done

echo  "$on in a reverse order $rev"
```

1.23 Write a shell script that will count the number of files in each of your sub-directories using the for loop.

```
# Write a shell script that will count the number of files in each of your
↪  sub-directories using the for loop.

START=$HOME

# change your directory to command line if passed
# otherwise use home directory
[ $# -eq 1 ] && START=$1 || :

if [ ! -d $START ]
then
        echo "$START not a directory!"
        exit 1
fi

# use find command to get all subdirs name in DIRS variable
DIRS=$(find "$START" -type d)

# loop thought each dir to get the number of files in each of subdir
for d in $DIRS
do
   [ "$d" != "." -a "$d" != ".." ] &&  echo "$d dirctory has $(ls -l $d | wc
   ↪  -l) files" || :
done
```

1.24 Write a shell script that will count the number of files in each of your sub-directories using the for loop.

```
# Write a shell script that will count the number of files in each of your
↪  sub-directories using the for loop.

START=$HOME

# change your directory to command line if passed
# otherwise use home directory
[ $# -eq 1 ] && START=$1 || :

```

```
if [ ! -d $START ]
then
        echo "$START not a directory!"
        exit 1
fi

# use find command to get all subdirs name in DIRS variable
DIRS=$(find "$START" -type d)

# loop thought each dir to get the number of files in each of subdir
for d in $DIRS
do
   [ "$d" != "." -a "$d" != ".." ] &&  echo "$d dirctory has $(ls -l $d | wc
   ↪  -l) files" || :
done
```

1.25 Write a shell script that accepts two directory names as arguments and deletes those files in the first directory which are similarly named in the second directory.

```
# Write a shell script that accepts two directory names as arguments and
↪  deletes those files in the first directory which are similarly named in
↪  the second directory.

SRC="$1"
DST="$2"
if [ $# -ne 2 ]
then
        echo "$(basename $0) dir1 dir2"
        exit 1
fi

if [ ! -d $SRC ]
then
        echo "Directory $SRC does not exist!"
        exit 2
fi


if [ ! -d $DST ]
then
        echo "Directory $DST does not exist!"
        exit 2
fi

for f in $DST/*
do
echo Processing $f
        if [ -f $f ]
        then
```

```
                tFile="$SRC/$(basename $f)"
                if [ -f $tFile ]
                then
                        echo -n "Deleting $tFile..."
                        /bin/rm $tFile
                        [ $? -eq 0 ] && echo "done" || echo "failed"

                fi
        fi
done
```

1.26 Write a shell script to search for no password entries in /etc/passwd and lock all accounts.

```
# Write a shell script to search for no password entries in /etc/passwd and
↪  lock all accounts.

USERS="$(cut -d: -f 1 /etc/passwd)"
for u in $USERS
do
passwd -S $u | grep -Ew "NP" >/dev/null
if [ $? -eq 0 ]; then
passwd -l $u
fi
done
```

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

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

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
```

```
21    echo "$n2 must be greater than $n1, try again..."
22  fi
```

2 Practice Lab Session for 06-10-2020

2.1 Problem

```
# concatenate 2 strings and print the concatenated string length.
stringOne="Hello I am string One."
stringTwo="I am string Two."
stringJoin="$stringOne $stringTwo"
echo "$stringJoin"
echo "String length: ${#stringJoin}"
```

2.2 Problem

```
# write a shell script to find the length of a string
string="A demo string for testing purposes"
echo "{ $string } is ${#string} characters long."
```

2.3 Problem

```
# swap two variable values.
a=$1
b=$2
echo "a ->$a and b ->$b"
c=$b

b=$a
a=$c

echo "a ->$a and b ->$b"
```

2.4 Problem

```
# find a number in an array of numbers.
numbers=(0, 3, 1)
for i in "${numbers[@]}"; do
  if [[ ${numbers[i]}=="$1" ]]; then
          echo "$i is in the arr."
          fi
done
```

2.5 Problem

```
# alert user if string has less than 10 characters in a string
echo "Enter String"
read str

length=`echo $str | wc -c`
length=`echo $length - 1 |bc`
if [ $length -lt 10 ]
then
```

```
        echo "You enter have entered less than 10 characters!"
fi
```

2.6 Problem

```
# Write a shell script to input seven-digit no., reverse seven-digit no.
↪ &find the sumof all digits.
```

2.7 Problem

```
# number sequence
```

2.8 Problem

```
# check if file name has read write permissions.
echo -n "Enter file name : "
read file

# find out if file has write permission or not
[ -w $file ] && W="Write = yes" || W="Write = No"

# find out if file has excute permission or not
[ -x $file ] && X="Execute = yes" || X="Execute = No"

# find out if file has read permission or not
[ -r $file ] && R="Read = yes" || R="Read = No"

echo "$file permissions"
echo "$W"
echo "$R"
echo "$X"
```

2.9 Problem

```
# if file is a dir list files or else if file count number of lines.

echo -n \"Enter file or dir name : \"
read name

if [ -d $name ]
then
    echo \"Give name is directory\"
elif [ -f $name ]
then
    echo \"File name is : $name\"
    echo \"No of line in file is : \`wc -l $name | cut -d\" \" -f1\`\"
fi
```

2.10 Problem

```
# display all files who have more than 350 bytes of storage
ls -l | cut -c 31-48,56- > b1
tr -s ' ' < b1 >b2
grep "^ [3-9][0-9][0-9]." b2 | sort -rn
```

2.11 Problem

```
# format ls -l output
#!/bin/bash

#copying the out of ls -l command to a file
ls -l > /tmp/tmp.tmp

#initilizing values
sum=0
dir=0
file=0
link=0

#reading the file
while read line
do
        #getting the first character of each line to check the type of file
        read -n 1 c <<< $line

        #checking if the file is a directory or not
        if [ $c == "d" ]
        then
                ((dir++))
                echo "[DIR] ${line}/" | cut -d" " --fields="1 9" >>
                 /tmp/dir.tmp

        elif [ $c == "-" ] #true if the file is a regular file
        then
                ((file++))
                echo $line | cut -d" " -f8 >> /tmp/file.tmp

        elif [ $c == "l" ]  #true if the file is a symbolic link
        then
                ((link++))
        fi

        size=$( echo $line | cut -d" " -f5 ) #getting the size of the file
        sum=$(( sum+size )) #adding the size of all the files
done < /tmp/tmp.tmp

cat /tmp/file.tmp #output the name of all the files
cat /tmp/dir.tmp #output the name of all the directory
```

```
41
42 echo "Total regular files = $file"
43 echo "Total directories = $dir"
44 echo "Total symbolic links = $link"
45 echo "Total size of regular file = $size"
46
47 #removing the temporary files
48 rm /tmp/file.tmp
49 rm /tmp/dir.tmp
50 rm /tmp/tmp.tmp
```

3 System Calls - File Structure Related Calls 9-10-2020

3.1 Write a program to demonstrate create() system call. The program should create a file in read and write mode.

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include<sys/fcntl.h>

int main()
{
    int fd; // file descriptor
    fd = creat("TestCreateFile.txt", S_IREAD | S_IWRITE);
    if (fd == -1){
       printf("Error in opening TestCreateFile.txt\n");
   }
   else
   {
       printf("TestCreateFile.txt opened for read/write access\n");
       printf("TestCreateFile.txt is currently empty\n");
   }
   close(fd);
   exit (0);
}
```

3.2 Write a program to demonstrate open() system call. The program should open a file in read and write mode and writes a message in the file. If the file does not exist then it should create the file, permit read/write access to the file and then write the message to the file.

```c
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <stdio.h>

static char message[] = "Good, morning.\n";

int main()
{
        int fd;
        char buffer[80];
        fd = open("TestOpenFile.txt",O_RDWR | O_CREAT | S_IREAD | S_IWRITE);

        if(fd != -1)
        {
```

```
                printf("TestOpenFile.txt opened for read/write access\n");
                write(fd, message, sizeof(message));
                lseek(fd, 0L,0); /* go back to the beginning of the file */
                if(read(fd, buffer, sizeof(message)) == sizeof(message))
                        printf("\"%s\" was written to TestOpenFile.txt\n",
                        ↪  buffer);
                else
                        printf("*** Error Reading TestOpenFile.txt ***\n");

        }
        close(fd)
        return 0;

}
```

3.3 Write a program to demonstrate lseek() system call. The program should open a file in read and write mode, change the location of the read pointer of the file descriptor and calculate the length of the file in bytes. If the file does not exist, then display an error message.

```
#include <stdio.h>
#include <fcntl.h>
int main()
{
        int fd;
        long position;
        fd = open("creat.c", O_RDONLY);
        if ( fd != -1)
        {
                position = lseek(fd, 0L, 2); /* seek 0 bytes from end-of-file
                ↪  */
                if (position != -1)
                        printf("The length of datafile.dat is %ld bytes.\n",
                        ↪  position);
                else
                        perror("lseek error");
        }
                else
                {
                        printf("can't open datafile.dat\n");
                }

        close(fd);
}
```

3.4 Write a program to demonstrate dup() system call. The program should redirect the standard output to a file.

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
int main()
{
        int fd;
        fd= open("redirect.txt",O_WRONLY | O_CREAT, S_IREAD | S_IWRITE );
        if (fd == -1)
        {
                perror("redirect.txt");
                exit (1);
        }
        close(1); /* close standard output */
        dup(fd); /* fd will be duplicated into standard out's slot */
        close(fd); /* close the extra slot */
        printf("This text which you are reading has been redirected to redirect.txt!\n")
        ↪  /* should go to file redirect.txt */
        exit (0); /* exit() will close the files */
}
```

3.5 Write a program to demonstrate link() system call. The program should create a new link to the existing file.

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
if ((link("old_link.txt", "new_link.txt")) == -1)
{
perror(" ");
exit (1); /* return a non-zero exit code on error */
}
else
printf("File linked and contents copied successfully \n");
exit(0);
}
```

3.6 Write a program to demonstrate unlink() system call. The program should unlink(delete) to the existing file.

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
```

```
5           if ((unlink("old_link.txt")) == -1)
6           {
7                   perror(" ");
8                   exit (1); /* return a non-zero exit code on error */
9           }
10          else
11                  printf("File deleted successfully \n");
12          exit (0);
13  }
```