CENG 328 OPERATING SYSTEM

Quiz 1

1) Create a directory named as your name under your home directory.

$ mkdir betul

2) Create a file called “cars” in this directory in which the cars mercedes,bmw,opel and audi are listed.

$ cat > cars

mercedes

bmw

opel

audi

^D

$

3) Sort the car list in the file and store the sorted list to the file called “sorted”

$ sort < cars > sorted

4) Create a folder called “new” under the directory that you’ve previously created and copy the file “sorted” into this folder.

$ mkdir betul\ new

$ cp sorted betul/ new

5) Rename the file “sorted” as “sorted_file”

$ mv sorted sorted_file

6) Delete the file called “cars”

$ rm cars

7) Write a command that displays the number of the lines which doesn't have the word "bmw" in the file "sorted" and stores the result to the file result under your home directory

$ grep –vc bmw sorted > result

8) Write a command that displays the number of the online users at the moment

$ who | wc -l

9) Run a sleep process,send it to background and kill it.

1.

$ sleep 100 &

[223] 770

$ ps

770 sleep 100

$ kill -9 770

2.

$ sleep 100

^Z

$ps

625 sleep

$ kill -9 625

10) Give read and execute permissions to the user and group and remove rest of the permissions for one of the files that you've created

- r – x r – x - - -

5 5 0

$ chmod 550 result

<u>Lab3. Shell Programming</u>

Shell programs are a file that holds the one or more unix or linux commands.

A .Input and Output

echo : this command is used for standard output

Use echo command to display text or value of variable.

echo [options] [string, variables...]
Displays text or variables value on screen.
Options
-n Do not output the trailing new line.
-e Enable interpretation of the following backslash escaped characters in the strings:
\a alert (bell)
\b backspace
\c suppress trailing new line
\n new line
\r carriage return
\t horizontal tab
\\ backslash

For e.g. **$ echo -e "An apple a day keeps away \a\t\tdoctor\n"**

Ex1.

```
clear
echo "Knowledge is Power"
```

ex2. Script to print user information who currently login , current date & time

```
clear
echo "Hello $USER"
echo –e "Today is \c ";date
echo –e "Number of user login : \c" ; who | wc -l
echo "Calendar"
cal
```

B. Variables:

In shell programming declaring a variable before using it is not necessary. The value of the variable can be seen using $ sign before its name:

$ cat > myprog

message=”Hello world”

echo $message

^D

$ ./myprog

Hello world

$

read : this command is used for standard input

$ cat > myprog

echo input number

read number

echo your number is: $number

^D

$ ./myprog

input number

5

your number is: 5

$

C. Arithmetic Operations

Operations: / ,- ,+ ,% ,* ,= ,>= ,<= ,< ,> ,& ,| ,(,), a,o,!

expr op1 math-operator op2

Examples:

```
$ expr 1 + 3
$ expr 2 - 1
$ expr 10 / 2
$ expr 20 % 3
$ expr 10 \* 3
$ echo ‘expr 6 + 3‘
```

x=‘expr 3 * 5‘

echo $x

D. Testing

test : test command is used for testing operations. If test result is true, it returns true.

test –d myfile : test the file called myfile whether it is a directory or not

test –f myfile : test the file called myfile whether it is a file or not

test –r myfile : test the file called my file whether it has read permission or not

test –w myfile : test the file called my file whether it has write permission or not

test –x myfile : test the file called my file whether it has execution permission or not

test str1=str2 : is str1 equal to str2

test str1!=str2 : is str1 not equal to str2

test n1 –eq n2 : is n1 equal to n2

test n1 –le n2 : is n1 less than or equal to n2

test n1 –ge n2 : is n1 greater than or equal to n2

test n1 –ne n2 : is n1 not equal n2

test n1 –lt n2 : is n1 less than n2

E. If Statements

Syntax of the if statement:

if *condition*

then

    command1

    command2

elif

then

    command3

    command4

else

     command5

fi

example:

```
if test –r file1
then
    echo read permission
else
    echo no read permission
fi
```

F. Case Statement

Syntax of the case statement:

```
case variable-name in
option1 )
    command1
    command2
;;
```

option2)

    command3

    command4

;;

*)

    command5

;;

esac

example:

read number

case $number in

1)

     clear

     ls;;

2)

    ls

    ;;

*)

    echo wrong ;;

esac

G. For statement

for *variable name* in *list*

do

execute one for each item in the list until the list is

not finished (And repeat all statement between do and done)

done

```
for i in 1 2 3 4 5
do
echo "Welcome $i times"
done
```

Lists all words in the file called mylist

```
for i in ‘cat mylist‘
do
    echo $i
done
```

for i in *

do

echo $i

done

H. While Statement

```
while [ condition ]
do
                command1
                command2
                command3
                ..
                ....
done
```

i=1

while test $i –le 10

do

echo “i= $i”

i=‘expr $i + 1‘

done

I. Cut Statement

```
Usage: cut [OPTION]... [FILE]...
Print selected parts of lines from each FILE to standard output.
 -b, --bytes=LIST    output only these bytes
 -c, --characters=LIST  output only these characters
 -d, --delimiter=DELIM  use DELIM instead of TAB for field delimiter
 -f, --fields=LIST    output only these fields; also print any line
              that contains no delimiter character, unless
              the -s option is specified
```

```
-n           with -b: don’t split multibyte characters
-s, --only-delimited  do not print lines not containing delimiters
  --output-delimiter=STRING use STRING as the output delimiter
             the default is to use the input delimiter
  --help   display this help and exit
  --version output version information and exit
```

With no FILE, or when FILE is -, read standard input.

cut –c 4-7 file2

$cat > deneme

deneme1;deneme2;deneme3;

deneme4;deneme5

^D

$cut –f2 –d ‘;’ deneme

deneme2

deneme5

$

Lab. Work 1

1. Write Script to find out biggest number from given three numbers. Numbers are supplies as command line argument. Print error if sufficient arguments are not supplied

```
echo "Enter three number:"
read number1
read number2
read number3
if ((test $number1 -gt $number2) &&  (test $number1 -gt $number3))
then
      echo "the biggest number is $number1"
elif ((test $number2 -gt $number1) &&  (test $number2 -gt $number3))
then
      echo "the biggest number is $number2"
elif ((test $number3 -gt $number1) &&  (test $number3 -gt $number2))
then
      echo "the biggest number is $number3"
elif ((test $number3 -eq $number1) &&  (test $number3 -eq $number2))
then
       echo "numbers are equal"
else
      echo “error in numbers”
fi
```

2. Write script to print numbers as 5,4,3,2,1 using while loop

```
i=5
while test $i -ge 1
do
  echo -n " $i"
  i=‘expr $i - 1‘
done

echo
```

```
if test $# -ne 3
then
   echo "ERROR..."
fi

case $2 in
"+" )
     out=‘expr $1 + $3‘
     echo "$1 + $3= $out "
     ;;
"-" )
    out=‘expr $1 - $3‘
    echo "$1 - $3= $out "
    ;;
"x" )
   out=‘expr $1 \* $3‘
   echo "$1 * $3= $out "
   ;;
"/" )
   out=‘expr $1 / $3‘
   echo "$1 / $3= $out "
   ;;
esac
```

3. Write Script, using case statement to perform basic math operation as +,-,*,/

4. Write Script to see current date, time, username, and current directory

```
echo "Username = $USER"

echo -e "Date= \c";date

echo "Current working directory= ‘pwd‘"
```

5. Write a shell script that writes “I love network Lectures” into a file then ask user, the word that will be changed and the new word that will be replaced with the old one.

```
echo "Enter input file name:"; read ifile
echo "Enter output file name:"; read ofile
echo I love network lecture > $ifile
echo "Enter the word that you want to replace: ";read word1
echo "Enter the word that will replace with old one: ";read word2

for i in `cat $ifile`
do
  if test $i = $word1
  then
     echo -n $word2 >> $ofile
  else
     echo -n $i >> $ofile
  fi
  echo -n ' ' >> ofile
done
echo -n "." >> ofile
```

6. Write script to determine whether given file exist or not, file name is supplied as command line argument, also check for sufficient number of command line argument

```
if test $# -ne 1
then
   echo "Usage - $0  file-name"
fi

if test -f $1
then
   echo "$1 file exist"
else
   echo "Sorry, $1 file does not exist"
fi
```