Name : Zeinab Abdelghaffar Radwan

Lab 2

:**Bonus**: enhance the above script to support the following Synopsis

myls -option1 –option2
myls –option2 –option1
myls –option1option2
myls –option2option1

```
localhost:~# usage() {
>   echo "Usage: $0 [OPTIONS] [DIRECTORY]"
>   echo "Options:"
>   echo "  -l   List in long format"
>   echo "  -a   List all entries, including hidden files"
>   echo "  -d   If an argument is a directory, list only its name"
>   echo "  -i   Print inode number"
>   echo "  -R   Recursively list subdirectories"
>   exit 1
> }
localhost:~# if [ $# -eq 0 ]; then
>   ls
> else
>   while [[ $1 == -* ]]; do
>     case $1 in
>       -l|-a|-d|-i|-R)
>         ls_options="${ls_options}${1}"
>         ;;
>       *)
>         usage
>         ;;
>     esac
>     shift
>   done
> ls $ls_options "$@"
> fi
bench.py    hello.c     hello.js    readme.txt
localhost:~#
```

7- :Create a script called mytest where

It check the type of the given argument (file/directory)

```
localhost:~# usage() {
>   echo "Usage: $0 FILE_OR_DIRECTORY_PATH"
>   exit 1
> }
localhost:~# if [ $# -eq 0 ]; then
>   usage
> fi
Usage: sh FILE_OR_DIRECTORY_PATH
localhost:~# file_or_directory="$1"
localhost:~# if [ -f "$file_or_directory" ]; then
>   echo "$file_or_directory is a regular file."
> else
>   echo "$file_or_directory is a directory."
> fi
 is a directory.
```

It check the permissions of the given argument (read/write/execute)

```
localhost:~# echo "Permissions of $file_or_directory:"
Permissions of :
localhost:~# echo "Read: $(test -r "$file_or_directory" && echo "Yes" || echo "N
o")"
Read: No
localhost:~# echo "Write: $(test -w "$file_or_directory" && echo "Yes" || echo "
No")"
Write: No
localhost:~# echo "Execute: $(test -x "$file_or_directory" && echo "Yes" || echo
 "No")"
Execute: No
```

or

```
localhost:~# echo "Permissions of $file_or_directory:"
Permissions of :
localhost:~# for permission in Read Write Execute; do
>   echo "$permission: $(test -rwx "$file_or_directory" && echo "Yes" || echo "N
o")"
> done
sh: : unknown operand
Read: No
sh: : unknown operand
Write: No
sh: : unknown operand
Execute: No
```

8- :Create a script called myinfo where

.It asks the user about his/her logname

It print full info about files and directories in his/her home directory

.Copy his/her files and directories as much as you can in /tmp directory

.Gets his current processes status

```
localhost:~# read -p "Enter your logname: " user_logname
Enter your logname: zeinab
localhost:~# user_home_directory=$(eval echo ~$user_logname)
localhost:~# echo -e "\nFull info about files and directories in $user_home_dire
ctory:"

Full info about files and directories in ~zeinab:
localhost:~# ls -l "$user_home_directory"
ls: ~zeinab: No such file or directory
localhost:~# echo -e "\nCopying files and directories to /tmp:"

Copying files and directories to /tmp:
localhost:~# cp -r "$user_home_directory"/* /tmp/
cp: can't stat '~zeinab/*': No such file or directory
localhost:~# echo -e "\nCurrent processes status for user $user_logname:"

Current processes status for user zeinab:
localhost:~# ps -u "$user_logname"
ps: unrecognized option: u
BusyBox v1.31.1 () multi-call binary.

Usage: ps [-o COL1,COL2=HEADER]

Show list of processes

        -o COL1,COL2=HEADER     Select columns for display
```

Lab 3

1- Write a script called mycase, using the case utility to checks the type of character entered by a user:

Upper Case.

Lower Case.

Number.

Nothing.

```
localhost:~# read -p "Enter a character: " char
Enter a character: Z
localhost:~# case $char in
>     [A-Z])
>         echo "Upper Case."
>         ;;
>     [a-z])
>         echo "Lower Case."
>         ;;
>     [0-9])
>         echo "Number."
>         ;;
>     *)
>         echo "Nothing."
>         ;;
> esac
Upper Case.
```

```
localhost:~# read -p "Enter a character: " char
Enter a character: 2
localhost:~# case $char in
>     [A-Z])
>         echo "Upper Case."
>         ;;
>     [a-z])
>         echo "Lower Case."
>         ;;
>     [0-9])
>         echo "Number."
>         ;;
>     *)
>         echo "Nothing."
>         ;;
> esac
Number.
```

```
localhost:~# read -p "Enter a character: " char
Enter a character: a
localhost:~# case $char in
>     [A-Z])
>         echo "Upper Case."
>         ;;
>     [a-z])
>         echo "Lower Case."
>         ;;
>     [0-9])
>         echo "Number."
>         ;;
>     *)
>         echo "Nothing."
>         ;;
> esac
Lower Case.
```

2- Enhanced the previous script, by checking the type of string entered by a user:

Upper Cases.
Lower Cases.
Numbers.
Mix.
Nothing.

```
localhost:~# read -p "Enter a string: " input_string
Enter a string: zeinab
localhost:~# case "$input_string" in
>     *[A-Z]*)
>         echo "Upper Cases."
>         ;;
>     *[a-z]*)
>         echo "Lower Cases."
>         ;;
>     *[0-9]*)
>         echo "Numbers."
>         ;;
>     *[a-zA-Z0-9]*)
>         echo "Mix."
>         ;;
>     *)
>         echo "Nothing."
>         ;;
> esac
Lower Cases.
```

```
localhost:~# read -p "Enter a string: " input_string
Enter a string: 01289939695
localhost:~# case "$input_string" in
>     *[A-Z]*)
>         echo "Upper Cases."
>         ;;
>     *[a-z]*)
>         echo "Lower Cases."
>         ;;
>     *[0-9]*)
>         echo "Numbers."
>         ;;
>     *[a-zA-Z0-9]*)
>         echo "Mix."
>         ;;
>     *)
>         echo "Nothing."
>         ;;
> esac
Numbers.
```

```
localhost:~# read -p "Enter a string: " input_string
Enter a string: #$
localhost:~# case "$input_string" in
>     *[A-Z]*)
>         echo "Upper Cases."
>         ;;
>     *[a-z]*)
>         echo "Lower Cases."
>         ;;
>     *[0-9]*)
>         echo "Numbers."
>         ;;
>     *[a-zA-Z0-9]*)
>         echo "Mix."
>         ;;
>     *)
>         echo "Nothing."
>         ;;
> esac
Nothing.
```

3- Write a script called mychmod using for utility to give execute permission to all files and directories in your home directory.

```
localhost:~# home_directory=$HOME
localhost:~# for file_or_dir in "$home_directory"/*
> do
>     if [ -f "$file_or_dir" ] || [ -d "$file_or_dir" ]; then
>         chmod +x "$file_or_dir"
>         echo "Execute permission for: $file_or_dir"
>     fi
> done
Execute permission for: /root/bench.py
Execute permission for: /root/hello.c
Execute permission for: /root/hello.js
n $home_directory."Execute permission for: /root/readme.txt
localhost:~# echo "Execution permissions to all files and directories in $home_d
irectory."
Execution permissions to all files and directories in /root.
```

4- Write a script called mybackup using for utility to create a backup of only files in your home directory.

```
localhost:~# home_directory=$HOME
localhost:~# backup_directory="$home_directory/backup"
localhost:~# mkdir -p "$backup_directory"
localhost:~# for file in "$home_directory"/*
> do
>     if [ -f "$file" ]; then
>         filename=$(basename "$file")
>         cp "$file" "$backup_directory/$filename"
>         echo "Backup created for: $file"
>     fi
> done
Backup created for: /root/bench.py
Backup created for: /root/hello.c
Backup created for: /root/hello.js
Backup created for: /root/readme.txt
localhost:~# echo "Backup completed. Files are stored in $backup_directory."
Backup completed. Files are stored in /root/backup.
```

5- Write a script called mymail using for utility to send a mail to all users in the system. Note: write the mail body in a file called mtemplate.

```
localhost:~# email_body=$(<mtemplate)
sh: can't open mtemplate: no such file
localhost:~# user_list=$(getent passwd | cut -d: -f1)
eetings from mymail script" "$user"
    echo "Email sent to: $user"
done
echo "Emails sent to all users in the system."localhost:~# for user in $user_lis
t
> do
>     echo "$email_body" | mail -s "Greetings from mymail script" "$user"
>     echo "Email sent to: $user"
> done
sh: mail: not found
Email sent to: root
sh: mail: not found
Email sent to: bin
sh: mail: not found
Email sent to: daemon
sh: mail: not found
Email sent to: adm
sh: mail: not found
Email sent to: lp
sh: mail: not found
Email sent to: sync
sh: mail: not found
Email sent to: shutdown
sh: mail: not found
Email sent to: halt
sh: mail: not found
```

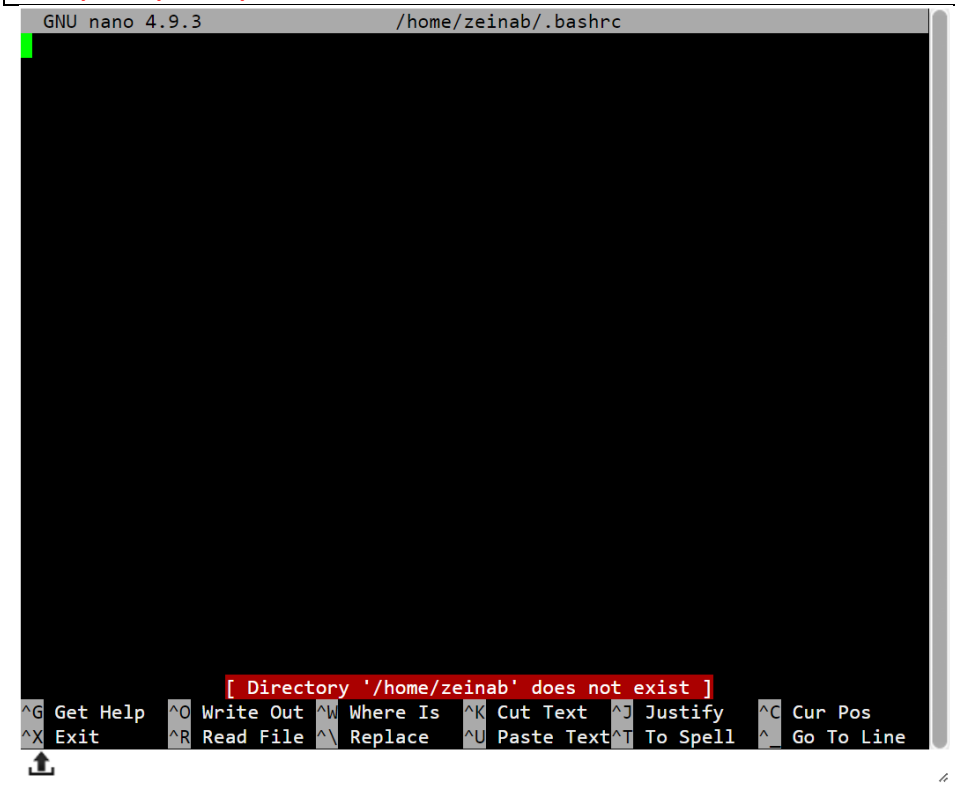
6- Write a script called chkmail to check for new mails every 10 seconds. Note: mails are saved in /var/mail/username.

```
localhost:~# mail_dir="/var/mail"
localhost:~# username=$(whoami)
localhost:~# mail_file="$mail_dir/$username"
localhost:~# while true
> do
>     new_mail_count=$(mail -H | grep -c "N  ")
>     if [ "$new_mail_count" -gt 0 ]; then
>         echo "You have $new_mail_count new mail(s)."
>     else
>         echo "No new mails."
>     fi
>     sleep 10
> done
sh: mail: not found
No new mails.
sh: mail: not found
No new mails.
```

Bonus: Open a talk session to a certain user when she/he logs into the system.

1- Edit the User's Shell Configuration:

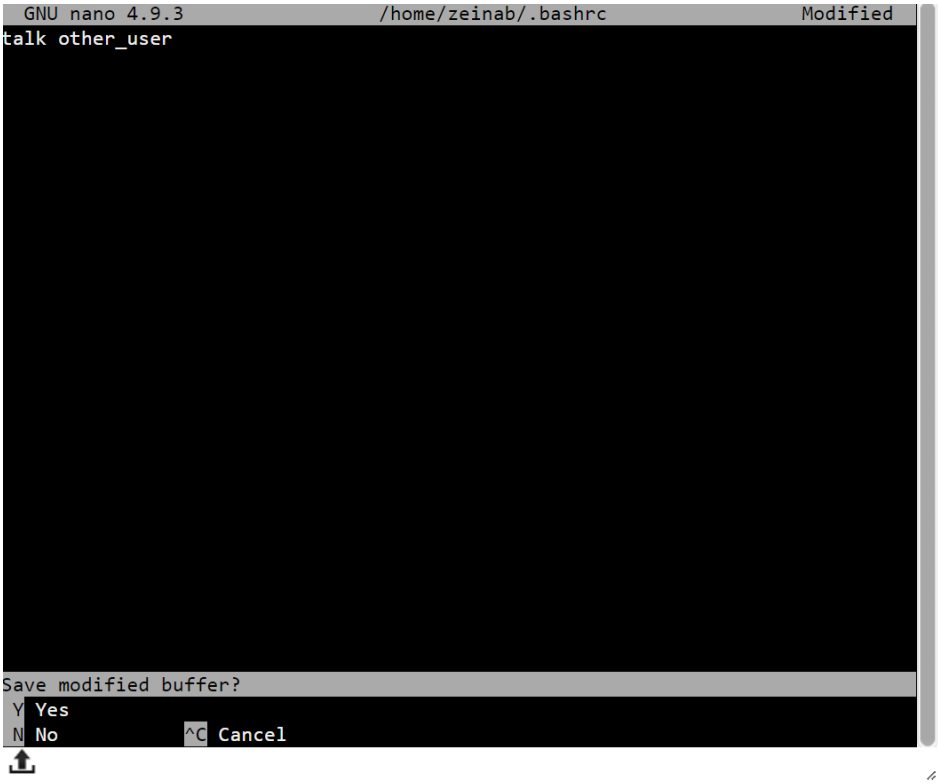
nano /home/zeinab/.bashrc



2- Add the talk Command:

talk other_user

3- Save the changes and exit the editor : In nano, press Ctrl + X, then Y to confirm the changes, and finally Enter to exit.



7- What is the output of the following script

```
typeset –i n1
typeset –i n2
n1=1
n2=1
while test $n1 –eq $n2
do
        n2=$n2+1
        print $n1
if [ $n1 –gt $n2 ]
then
        break
else
        continue
fi
n1=$n1+1
print $n2
done
```

The Output is : 1

```
localhost:~# n1=1
localhost:~# n2=1
localhost:~#
localhost:~# while [ $n1 -eq $n2
> do
>     n2=$n2+1
>     echo $n1
>
>     if [ $n1 -gt $n2 ]
>     then
>         break
>     else
>         continue
>     fi
>
>     n1=$n1+1
>     echo $n2
> done
1
```

8- Create the following menu:

Press 1 to ls

Press 2 to ls –a

Press 3 to exit

Using select utility then while utility.

```
localhost:~# while true; do
>     echo "Menu:"
>     echo "1. Press 1 to ls"
>     echo "2. Press 2 to ls -a"
>     echo "3. Press 3 to exit"
>     read -p "Enter your choice: " choice
> case $choice in
>         1)
>             ls
>             ;;
>         2)
>             ls -a
>             ;;
>         3)
>             echo "Exiting the menu."
>             break
>             ;;
>         *)
>             echo "Invalid choice. Please enter 1, 2, or 3."
>             ;;
>     esac
> done
Menu:
1. Press 1 to ls
2. Press 2 to ls -a
3. Press 3 to exit
Enter your choice:
Enter your choice: 1
bench.py    hello.c     hello.js    readme.txt
Menu:
1. Press 1 to ls
2. Press 2 to ls -a
3. Press 3 to exit
Enter your choice: 2
.             .ash_history  .mozilla      bench.py      hello.js
..            .cache        .wine         hello.c       readme.txt
Menu:
1. Press 1 to ls
2. Press 2 to ls -a
3. Press 3 to exit
Enter your choice: 3
Exiting the menu.
```

9- Write a script called myarr that ask a user how many elements he wants to enter in an array, fill the array and then print it.

JSLinux doesnot work in declaration array so I used ubuntu here

```
zeinab@Zeinab:~$ #!/bin/bash
zeinab@Zeinab:~$ echo -n "Enter the number of elements in the array: "
m_elemeEnter the number of elements in the array: ntszeinab@Zeinab:~$ read num_elements
3
zeinab@Zeinab:~$ my_array=()
zeinab@Zeinab:~$ for i in $(seq 1 "$num_elements"); do
   echo >     echo -n "Enter element $i: "
read ele>     read element
>     my_array+=("$element")
one> done
Enter element 1: 1
Enter element 2: 2
Enter element 3: 3
zeinab@Zeinab:~$ echo "The array is: ${my_array[@]}"
The array is: 1 2 3
```

10- Write a script called myavg that calculate average of all numbers entered by a user. Note: use arrays

JSLinux doesnot work in declaration array so I used ubuntu here

zeinab@Zeinab: ~

```
zeinab@Zeinab:~$ #!/bin/bash
ho -n "zeinab@Zeinab:~$ echo -n "Enter the number of elements: "
Enter the number of elements: zeinab@Zeinab:~$ read num_elements
3
zeinab@Zeinab:~$ my_array=()
zeinab@Zeinab:~$ for i in $(seq 1 "$num_elements"); do
>     echo -n "Enter number $i: "
>     read number
>     my_array+=("$number")
> done
Enter number 1: 12
Enter number 2: 13
Enter number 3: 11
zeinab@Zeinab:~$ sum=0
t in "${my_azeinab@Zeinab:~$ for element in "${my_array[@]}"; do
>     sum=$((sum + element))
> done
zeinab@Zeinab:~$ if [ "$num_elements" -gt 0 ]; then
ge=$((>     average=$((sum / num_elements))
>     echo "The average is: $average"
> else
>     echo "Cannot calculate average. Number of elements must be greater than 0."
> fi
The average is: 12
```

11- Write a function called mysq that calculate square if its argument.

```
localhost:~# mysq() {
>     local num=$1
>     local square=$((num * num))
>     echo "The square of $num is: $square"
> }
localhost:~# echo -n "Enter a number: "
Enter a number: localhost:~# read user_input
3
localhost:~# mysq "$user_input"
The square of 3 is: 9
```