12112001

LAB FILE

Linux programming

EXPERIMENT 2

Q1. Commands:

1. Man

```
skaezr73@skaezr73-VirtualBox:/home$ man ls
skaezr73@skaezr73-VirtualBox:/home$ man ps
skaezr73@skaezr73-VirtualBox:/home$
```

```
PS(1)                          User Commands                          PS(1)

NAME
       ps - report a snapshot of the current processes.

SYNOPSIS
       ps [options]

DESCRIPTION
       ps displays information about a selection of the active processes.  If you want a repetitive update of the selection and
       the displayed information, use top instead.
```

2. PS

```
skaezr73@skaezr73-VirtualBox:/home$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000    2282    2148  0  80   0 -  4948 do_wai pts/0    00:00:00 bash
4 R  1000    3864    2282  0  80   0 -  5331 -      pts/0    00:00:00 ps
```

3.who

```
skaezr73@skaezr73-VirtualBox:/home$ who
skaezr73 tty2         2023-02-27 20:37 (tty2)
skaezr73@skaezr73-VirtualBox:/home$
```

4. Date ,time

```
skaezr73@skaezr73-VirtualBox:/home$ date
Monday 27 February 2023 08:57:16 PM IST
skaezr73@skaezr73-VirtualBox:/home$ time

real    0m0.000s
user    0m0.000s
sys     0m0.000s
skaezr73@skaezr73-VirtualBox:/home$
```

5. Pwd ,cd , ls

```
skaezr73@skaezr73-VirtualBox:/home$ pwd
/home
skaezr73@skaezr73-VirtualBox:/home$ ls
skaezr73
skaezr73@skaezr73-VirtualBox:/home$ cd skaezr73/Documents
skaezr73@skaezr73-VirtualBox:~/Documents$ pwd
/home/skaezr73/Documents
skaezr73@skaezr73-VirtualBox:~/Documents$
```

6. History

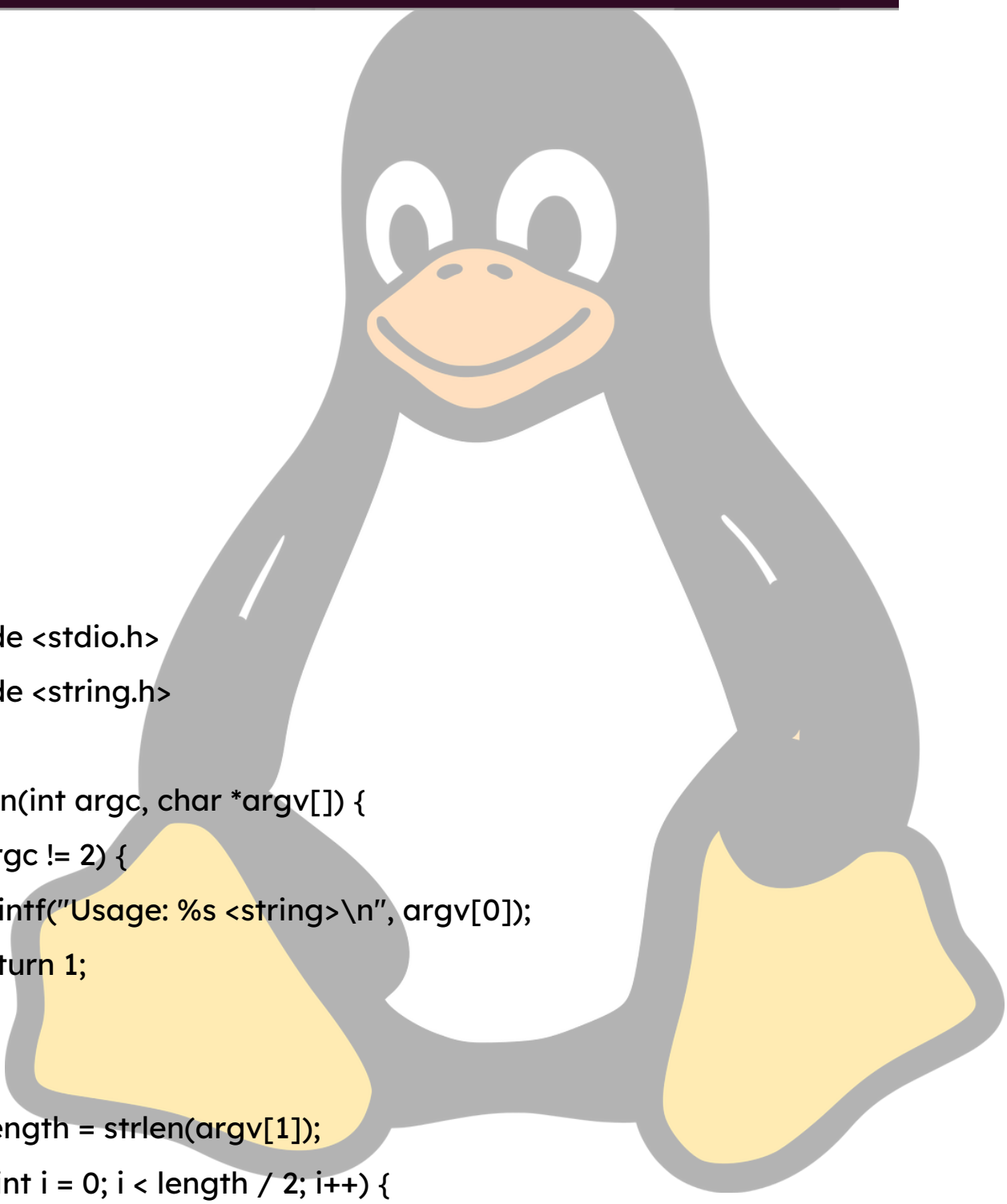
```
skaezr73@skaezr73-VirtualBox:~/Documents$ history
    1  ./test.sh
    2  ./linuxx.sh
    3  #!/bin/bash
    4  ./linuxx.sh
    5  echo $path
    6  echo @PATH
    7  echo $PATH
    8  nano
    9  sh linuxx.sh
   10  nano
   11  sh linuxx.sh
   12  vi
   13  sh linuxx.sh
   14  vi
   15  ./cro.awk
   16  chmod +x cro.awk
   17  ./cro.awk
   18  chmod +x cro.awk
   19  ./cro.awk
```

7.echo

```
skaezr73@skaezr73-VirtualBox:~/Documents$ echo $"hello world"
hello world
skaezr73@skaezr73-VirtualBox:~/Documents$
```

8. Rmdir

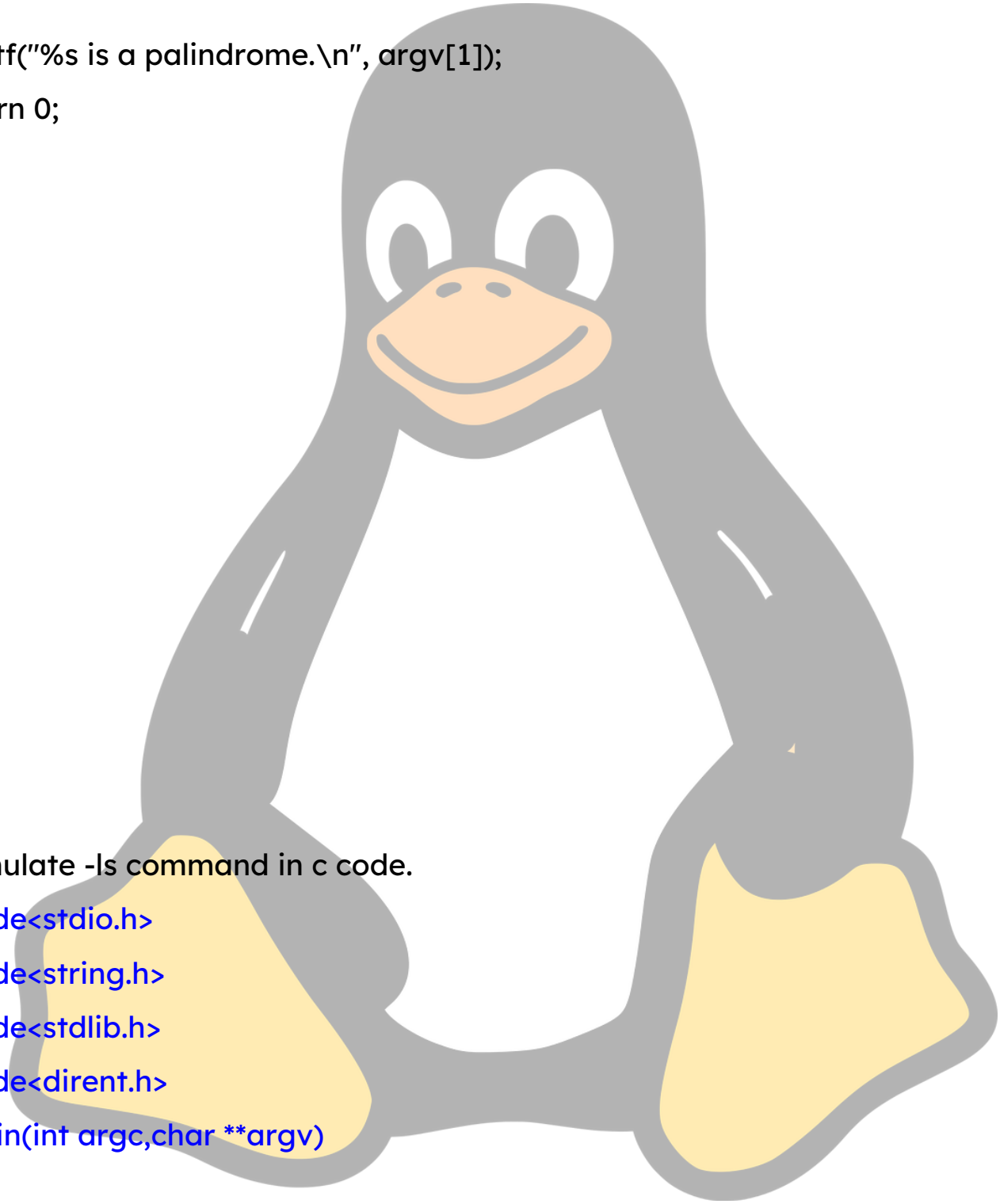
```
skaezr73@skaezr73-VirtualBox:~/Documents$ rmdir tree
skaezr73@skaezr73-VirtualBox:~/Documents$ ls
skaezr73@skaezr73-VirtualBox:~/Documents$
```



Q1.

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("Usage: %s <string>\n", argv[0]);
        return 1;
    }

    int length = strlen(argv[1]);
    for (int i = 0; i < length / 2; i++) {
        if (argv[1][i] != argv[1][length - i - 1]) {
            printf("%s is not a palindrome.\n", argv[1]);
            return 0;
```

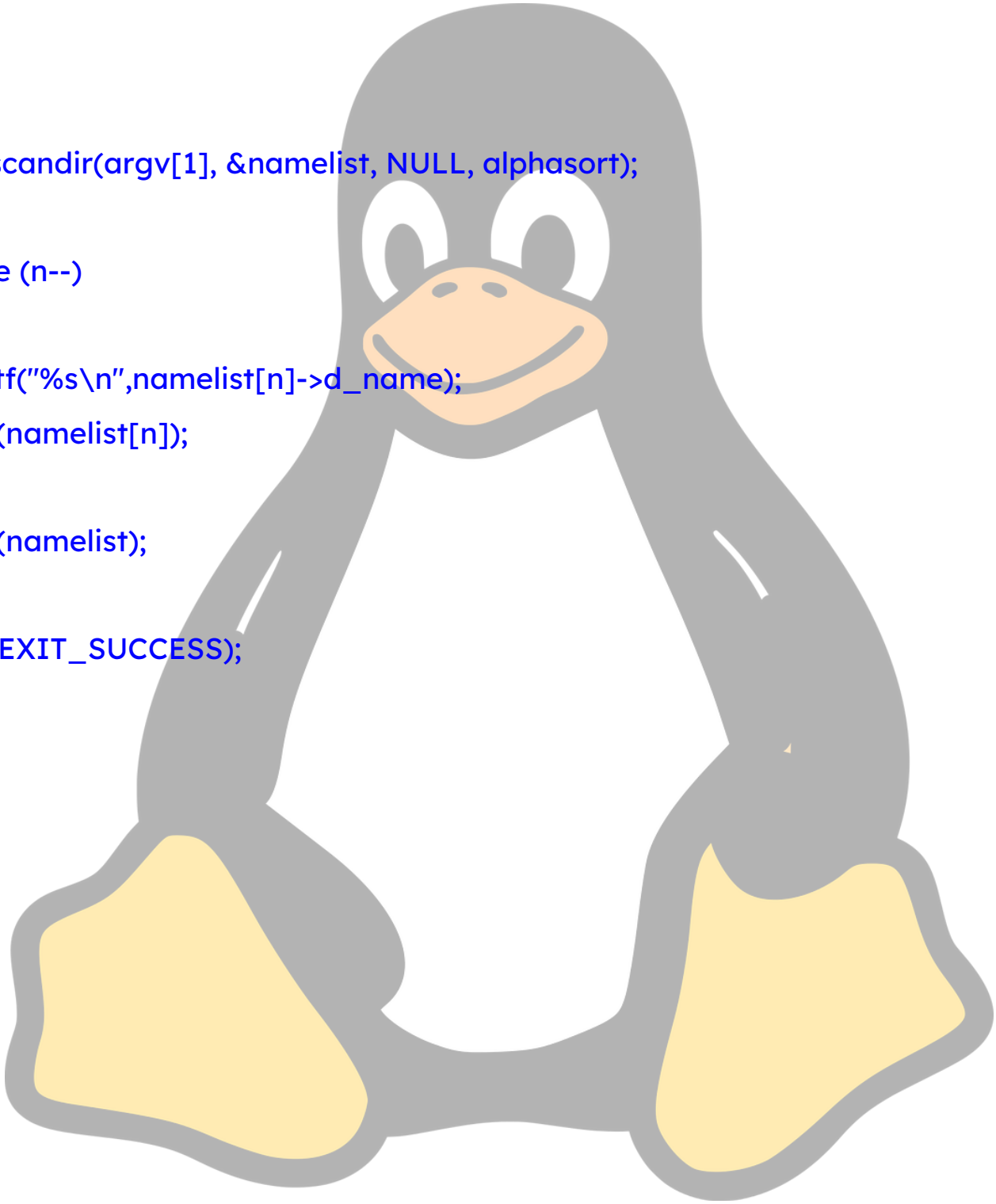
```
        }
    }

    printf("%s is a palindrome.\n", argv[1]);
    return 0;
}
```



Q2. emulate -ls command in c code.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<dirent.h>
int main(int argc,char **argv)
{
    struct dirent **namelist;
    int n;
  if (argc == 1)
```

```
    {
    n=scandir(".",&namelist,NULL,alphasort);
    }
    else
    {
    n = scandir(argv[1], &namelist, NULL, alphasort);
    }
    while (n--)
    {
    printf("%s\n",namelist[n]->d_name);
    free(namelist[n]);
    }
    free(namelist);

    exit(EXIT_SUCCESS);
}
```

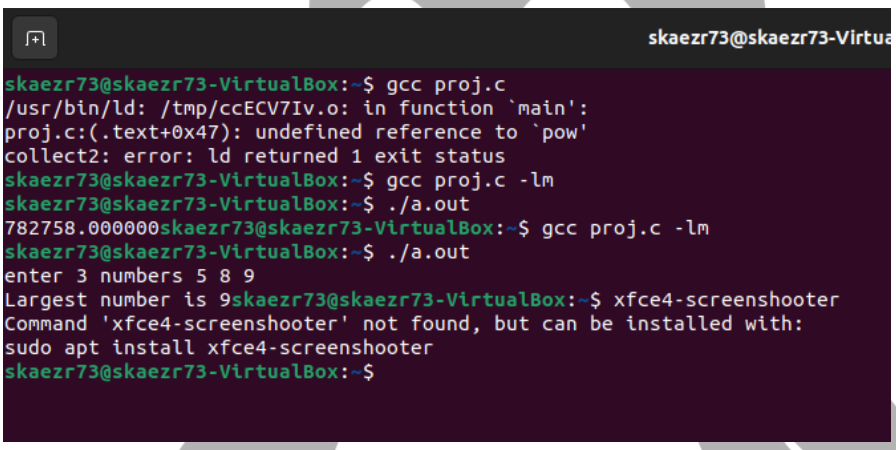


```
skaezr73@skaezr73-VirtualBox:~$ gcc proj.c -lm
skaezr73@skaezr73-VirtualBox:~$ ./a.out
snap
proj.c
nano.5206.save
nano.4877.save
linuxx.sh
linux.c
car.txt
a.out
Videos
Templates
Public
Pictures
Music
Downloads
Documents
Desktop
.thunderbird
.sudo_as_admin_successful
.ssh
.profile
.mozilla
.local
.lesshst
.gnupg
.config
.cache
.bashrc
.bash_logout
.bash_history
..
.
```

Q3. program to display largest of 3 integers.

```
#include<stdio.h>
#include<math.h>
int max(int a,int b){
   if (a>b) return a;
   else return b;
}

int main(){
   printf("enter 3 numbers");
```

```
   int a,b,c; scanf("%d %d %d",&a,&b,&c);
   int ans=max(a,max(b,c));
   printf("Largest number is %d",ans);

}
```



Q4. check whether number is avogadro or not.

```
#include <stdio.h>
int main() {
   int num, originalNum, remainder, result = 0;
   printf("Enter a three-digit integer: ");
   scanf("%d", &num);
   originalNum = num;

   while (originalNum != 0) {
   remainder = originalNum % 10;
   result += remainder * remainder * remainder;
   originalNum /= 10;
   }

   if (result == num)
```

```
        printf("%d is an Armstrong number.", num);
    else
        printf("%d is not an Armstrong number.", num);

    return 0;
}
```

```
skaezr73@skaezr73-VirtualBox:~$ ./a.out
Enter a three-digit integer: 789
789 is not an Armstrong number.skaezr73@skaezr73-VirtualBox:~$
```

Q5. find factorial of number.

```
#include <stdio.h>
int fact(int n){
   if ( n==1) return 1;
   else return n*fact(n-1);
}
int main() {
   int n; printf("enter the number...");
   scanf("%d",&n);
   int fs=fact(n);
   printf("factorial of %d is : %d",n,fs);
}
```

```
skaezr73@skaezr73-VirtualBox:~$ gcc proj.c -lm
skaezr73@skaezr73-VirtualBox:~$ ./a.out
enter the number... 6
factorial of 6 is : 720skaezr73@skaezr73-VirtualBox:~$
```

Q6. C program to check prime number or not.

Q7. find power of first number raised to second.

```
#include <stdio.h>
#include<math.h>
int fact(int n){
   if ( n==1) return 1;
   else return n*fact(n-1);



}
int main() {
   int a,b; printf("enter two numbers");
   scanf("%d %d",&a,&b);
   printf("a raised to power b is: %f",pow(a,b));


}
```

```
skaezr73@skaezr73-VirtualBox:~$ gcc proj.c -lm
skaezr73@skaezr73-VirtualBox:~$ ./a.out
enter two numbers 6 4
a raised to power b is: 1296.000000skaezr73@skaezr73-VirtualBox:~$
```

EXPERIMENT 3

12112001

Q1.

Shell script program to check number of users logged in.

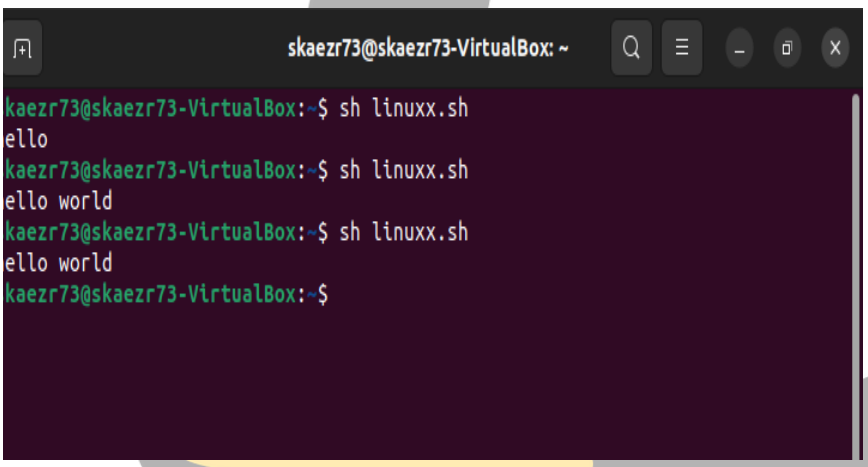
```
#!/bin/bash

num_users=$(who | wc -l)
echo "Number of users logged in: $num_users"
```

Q2.

Shell script program to display hello world.

#!/bin/sh

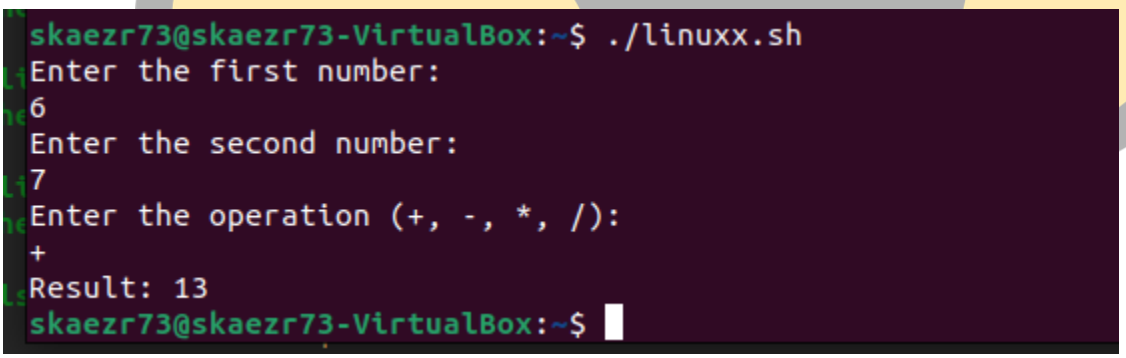
echo "hello world"



Q3. Develop a scientific calculator.

#!/bin/bash

```
echo "Enter the first number:"
read num1

echo "Enter the second number:"
read num2
echo "Enter the operation (+, -, *, /):"
read operator
if [ "$operator" = "+" ]
then
    result=$((num1 + num2))
elif [ "$operator" = "-" ]
then
    result=$((num1 - num2))
elif [ "$operator" = "*" ]
then
    result=$((num1 * num2))
elif [ "$operator" = "/" ]
then
    result=$((num1 / num2))
else
    echo "Invalid operator"
    exit 1
fi

echo "Result: $result"
```



Q4. Check whether even or odd number.

```
#!/bin/bash

echo "Enter a number:"
read number

if [ $((number % 2)) -eq 0 ]
then
   echo "$number is even"
else
   echo "$number is odd"
fi
```

```
./linuxx.sh: line 17: syntax error: unexpected end of file
skaezr73@skaezr73-VirtualBox:~$ ./linuxx.sh
Enter a number:
6
6 is even
skaezr73@skaezr73-VirtualBox:~$
```

Q5. Check whether element is in list or not.

```
#!/bin/bash

# Define the list
list=("qatar" "yemen" "syria" "iran" "iraq")

element=”qatar"
for element in "${list[@]}"; do
   # Check if the current element is equal to the search element
   if [ "$element" == "$element" ]; then
      echo "$element is in the list."
      exit 0
   fi
```

done

echo "$element is not in the list."

exit 1

EXPERIMENT -4

Q1. Write a shell program to check whether file is directory or not.

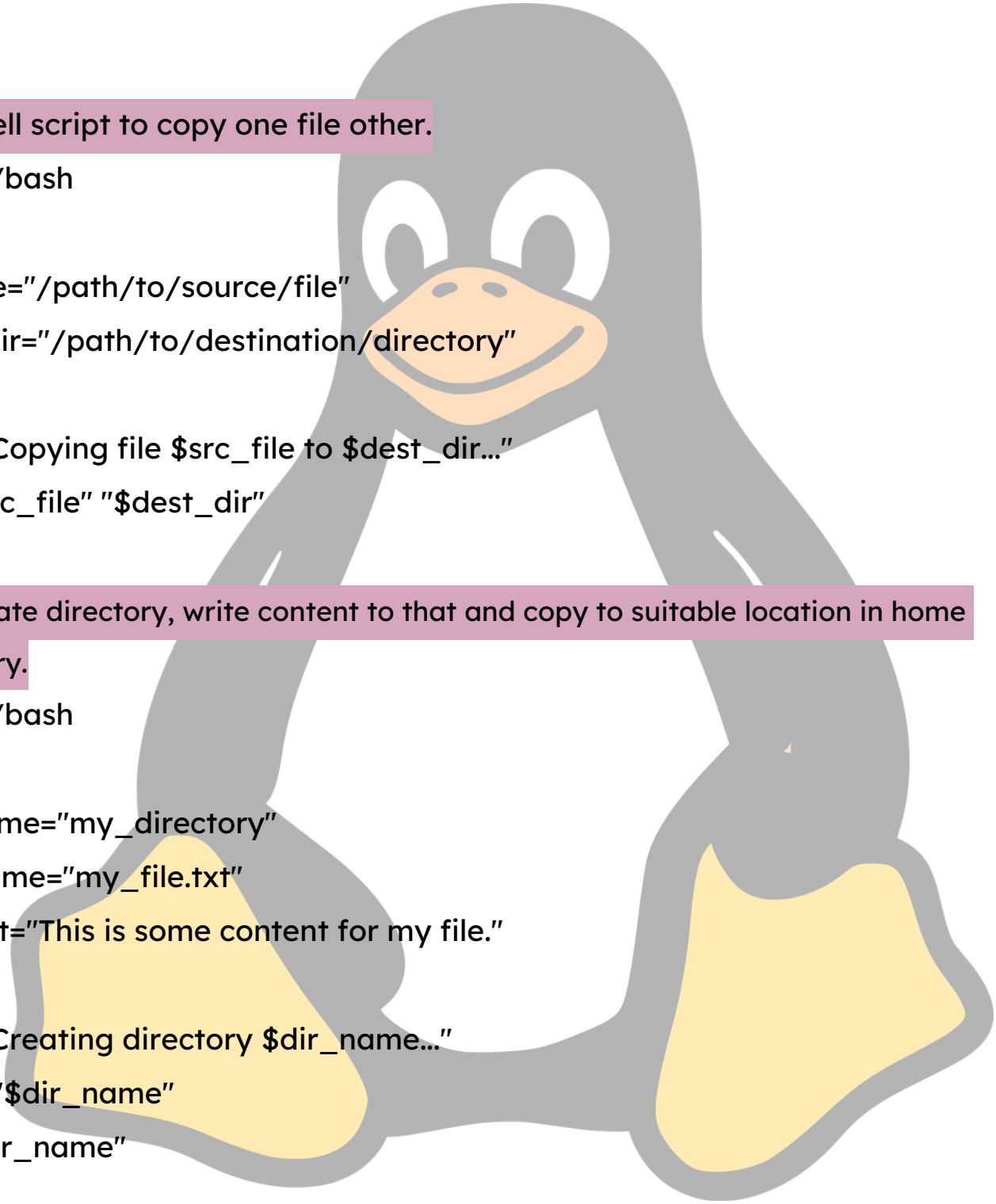
```
#!/bin/bash
# Prompt the user to enter a file name
echo "Enter a file name: "
read filename
# Check if the file is a directory
if [ -d "$filename" ]
then
   echo "$filename is a directory"
else
   echo "$filename is not a directory"
fi
```

```
Done
skaezr73@skaezr73-VirtualBox:~$ ./linuxx.sh
Enter a file name:
tree
tree is not a directory
skaezr73@skaezr73-VirtualBox:~$
```

Q2.

```
#!/bin/bash
# Prompt user to enter directory path
echo "Enter directory path: "
read dir_path
# Count the number of files in the directory
```

```
num_files=$(ls -l "$dir_path" | wc -l)
# Display the result to the user
echo "The number of files in $dir_path is: $num_files"
```



Q3. Shell script to copy one file other.

```
#!/bin/bash

src_file="/path/to/source/file"
dest_dir="/path/to/destination/directory"

echo "Copying file $src_file to $dest_dir..."
cp "$src_file" "$dest_dir"
```

Q4.Create directory, write content to that and copy to suitable location in home directory.

```
#!/bin/bash

dir_name="my_directory"
file_name="my_file.txt"
content="This is some content for my file."

echo "Creating directory $dir_name..."
mkdir "$dir_name"
cd "$dir_name"

echo "Writing content to $file_name..."
echo "$content" > "$file_name"
```

echo "Copying directory to home directory..."

cp -r . ~/"$dir_name"

cd ..

rm -r "$dir_name"

echo "Done"

```
Creating directory my_directory...
Writing content to my_file.txt...
Copying directory to home directory...
cp: '.' and '/home/skaezr73/my_directory/.' are the same f
Done
skaezr73@skaezr73-VirtualBox:~$
```

Q5.use pipeline and command line substitution to set length of line a file to a variable.

```
#!/bin/bash

file="muscat.txt"

length=$(cat "$file" | awk '{ print length }' | sort -n | tail -1)

echo "The longest line in $file has length: $length"
```

```
General help using GNU software: <https://www.gnu.org/geth
skaezr73@skaezr73-VirtualBox:~$ sed -n 'p' muscat.txt
text
skaezr73@skaezr73-VirtualBox:~$ ./linuxx.sh
The longest line in muscat.txt has length: 4
skaezr73@skaezr73-VirtualBox:~$
```

Q6. use sed command to print multiple lines of input

sed -n 'p' muscat.txt

EXPERIMENT -5

Q1:

A..Write grep /egrep script to count number of words / number of lines and number of characters in a file.

Text file

```
1 mehraba nasiliniz
2 selam
3 turkey== ankara
4 bahrain == manama
5 yemen == sanaa
6 somalia==mogadishu
7
```

Script to find number of words.

grep -0 '\w\+' bahrain.txt| wc -w

```
skaezr73@skaezr73-VirtualBox:~$ grep -0 '\w\+' bahrain.txt| wc -w
12
skaezr73@skaezr73-VirtualBox:~$ ^C
```

Script to find number of characters

Script to find number of lines

grep -c '^' bahrain.txt

```
skaezr73@skaezr73-VirtualBox:~$ grep -c '^' bahrain.txt
7
skaezr73@skaezr73-VirtualBox:~$
```

B..Write awk script to print fibonacci series

awk 'BEGIN{a=1;b=1}{print a; c=a+b; a=b; b=c}'

```
skaezr73@skaezr73-VirtualBox:~$ awk 'BEGIN{a=1;b=1}{print a; c=a+b; a=b; b=c}'
^[[A
1

1
^[[A
2
^[[A
3
^[[A
5

8

13

21

34

55
```

Q.2

1. Perl script to compute nth power of given number.

** always use semi- colon

** perl file is saved with extension ".pl"

** Perl is an interpreted language, which means that your code can be run as is, without a compilation stage that creates a non portable executable program.

```
** print "What is your name?\n";
$name = <>;  #take input in perl
print "Your name is ",$name #output in perl
```

```
#!/usr/bin/perl
print "enter the base number";
$a= <>;
print "enter the number by which we will raise the base";
$b= <>;

$c= $a**$b;
print "answer is ",$c;
```

```
syntax error at heb.pl line 5, near "$b"
Execution of heb.pl aborted due to compilation errors.
skaezr73@skaezr73-VirtualBox:~$ perl heb.pl
enter the base number 4
enter the number byt which we will raise the base 5
answer is 1024skaezr73@skaezr73-VirtualBox:~$ perl heb.pl
enter the base number 3
enter the number by which we will raise the base 3
answer is 27skaezr73@skaezr73-VirtualBox:~$
```

2. Awk script to display pattern of given string or number.

```
#!/usr/bin/awk -f

BEGIN {
    printf "Enter a string or number: "
    getline input < "/dev/tty"
    pattern = ""
    for (i = 1; i <= length(input); i++) {
        c = substr(input, i, 1)
        if (c ~ /[[:digit:]]/) {
            pattern = pattern "d"
        } else if (c ~ /[[:alpha:]]/) {
            pattern = pattern "a"
        } else {
            pattern = pattern "x"
        }
    }
            print "Pattern for \"" input "\" is:"
            print pattern
        }
```

```
skaezr73@skaezr73-VirtualBox:~$ chmod +x pat.awk
skaezr73@skaezr73-VirtualBox:~$ ./pat.awk
Enter a string or number: dtgsa
Pattern for "dtgsa" is:
aaaaa
skaezr73@skaezr73-VirtualBox:~$
```

Q3.

1 .Perl script to check whether the number is prime or not .

```
#!/usr/bin/perl
print "enter the number which we need to find whether prime or not ..";
$a= <>;
$flg=0;
for ($i=2;$i<$a/2;$i++){
   if( $a % $i==0) {
            $flg=1;
            break;


  }
}
if ($flg==0) {
  print " prime number\n";
  }
else
{
 print "not prime ";
 }
```

```
skaezr73@skaezr73-VirtualBox:~$ perl heb.pl
enter the number which we need to find whether prime or not .. 6
not prime skaezr73@skaezr73-VirtualBox:~$ perl heb.pl
enter the number which we need to find whether prime or not ..7
 prime number
skaezr73@skaezr73-VirtualBox:~$
```

2.

Taking multiply input in one line.

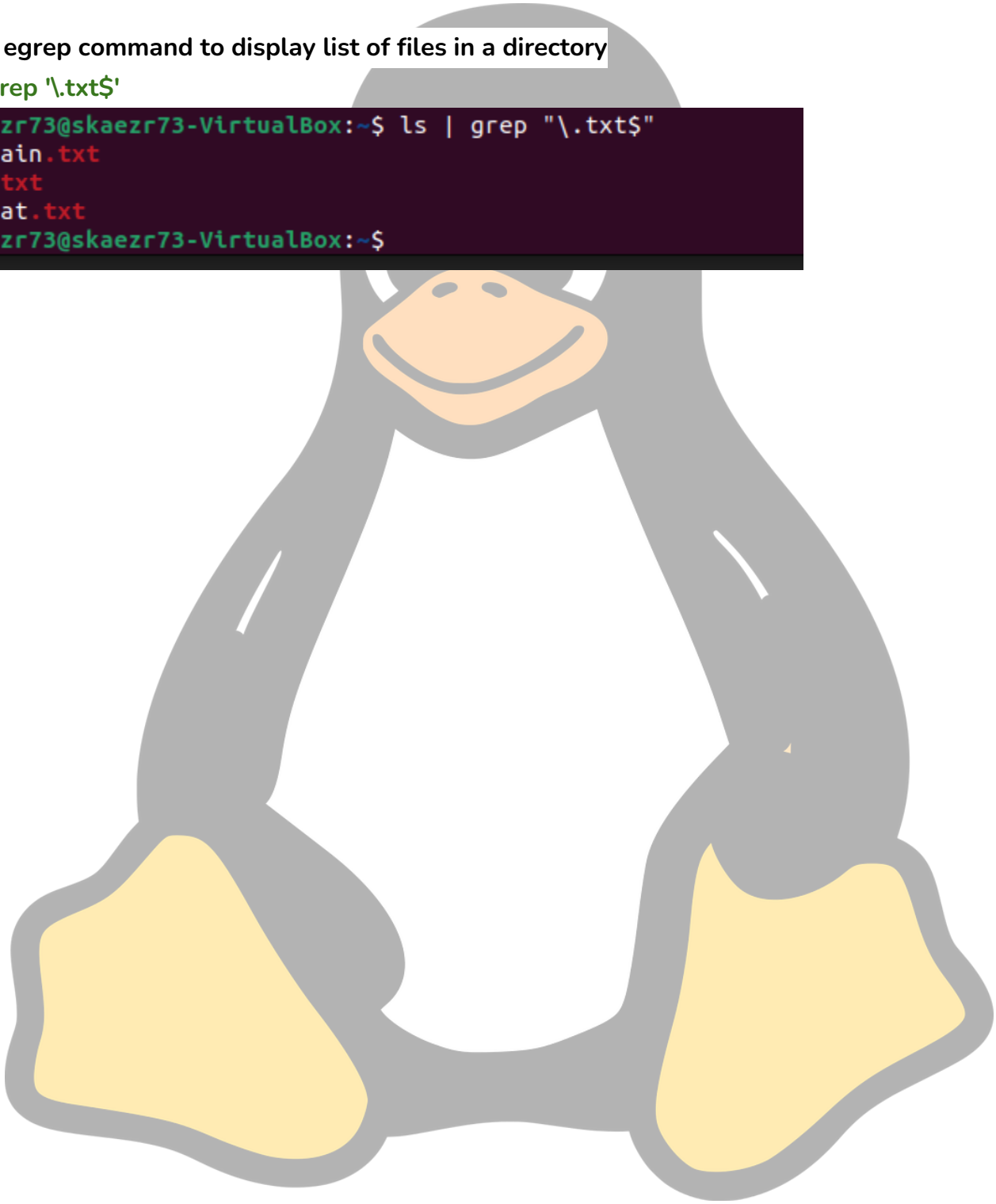
```
$t=<>;
for ($i=0;$i<$t;$i++){
   chomp(my $line = <>);     // chomp remove nextLine character!
   ($n, $k) = split " ", $line;
   if ($n%2!=0 && $k%2==0){
      print "NO\n";
   }
   else{
```

```
        print "YES\n";
    }
}
```

3. **Write egrep command to display list of files in a directory**

   ls | egrep '\.txt$'

```
skaezr73@skaezr73-VirtualBox:~$ ls | grep "\.txt$"
bahrain.txt
car.txt
muscat.txt
skaezr73@skaezr73-VirtualBox:~$
```



How to run shell script file...

Chmod+x linux.sh

./linuxx

1. Write a shell script program to display process of attributes.

```
!/bin/bash
for r in 1,2,3
do
  echo "process are: \n"
  ps -e
done
```

```
skaezr73@skaezr73-VirtualBox:~$ chmod +x linuxx.sh
skaezr73@skaezr73-VirtualBox:~$ ./linuxx.sh
./linuxx.sh: line 1: !/bin/bash: No such file or directory
process are: \n
    PID TTY          TIME CMD
      1 ?        00:00:04 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 rcu_gp
      4 ?        00:00:00 rcu_par_gp
      5 ?        00:00:00 slub_flushwq
      6 ?        00:00:00 netns
      8 ?        00:00:00 kworker/0:0H-events_highpri
     10 ?        00:00:00 mm_percpu_wq
     11 ?        00:00:00 rcu_tasks_rude_
     12 ?        00:00:00 rcu_tasks_trace
     13 ?        00:00:01 ksoftirqd/0
     14 ?        00:00:02 rcu_sched
     15 ?        00:00:00 migration/0
     16 ?        00:00:00 idle_inject/0
     18 ?        00:00:00 cpuhp/0
     19 ?        00:00:00 cpuhp/1
```

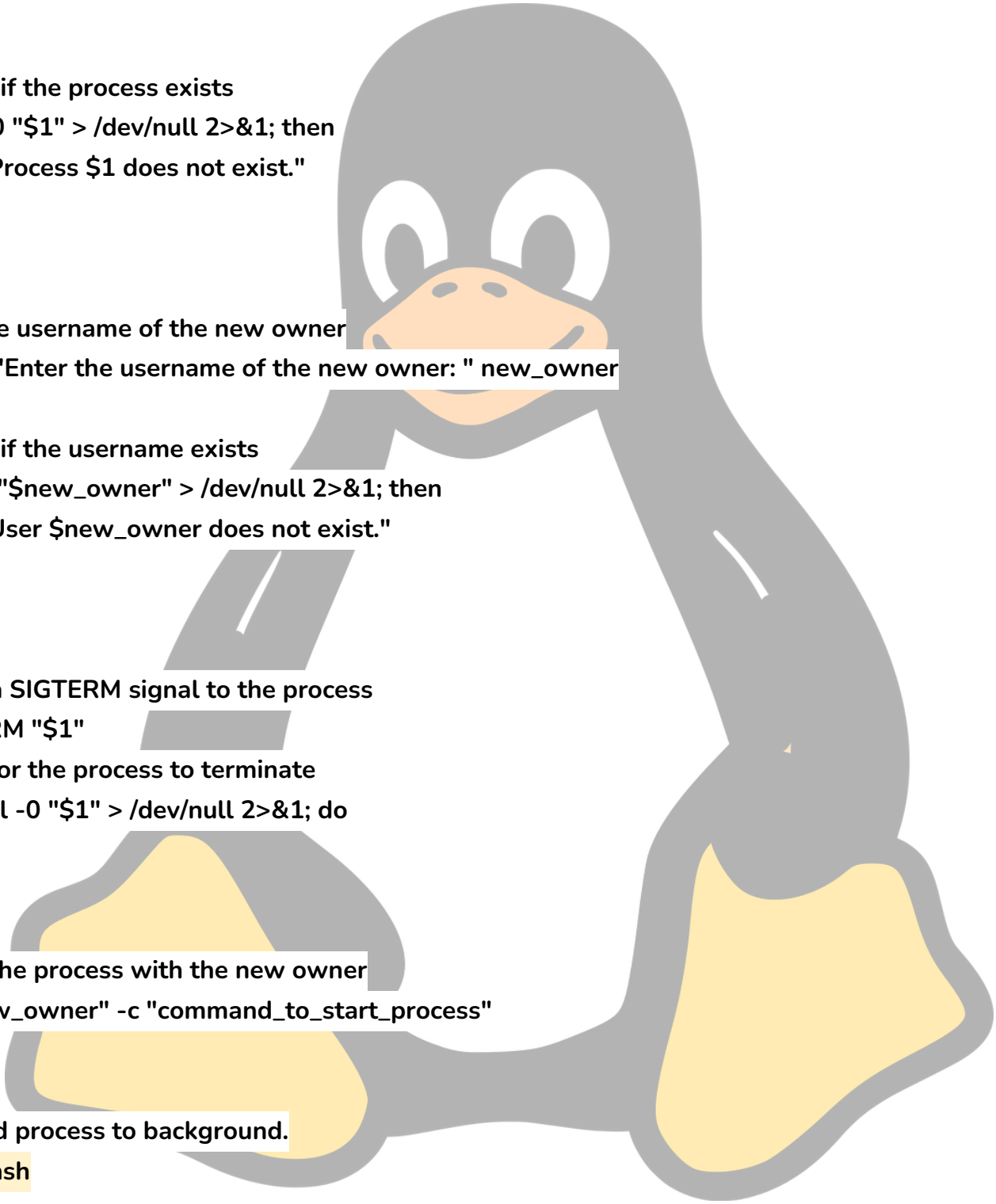
Q2 . Change priority of process using shell script

```
!/bin/bash
echo "priority changed"
renice -n 19 23900
```

Q3. write shell script program to change ownership of process

#!/bin/bash

```
if [ -z "$1" ]; then
  echo "Please provide a process ID."
  exit 1
fi

# Check if the process exists
if ! kill -0 "$1" > /dev/null 2>&1; then
  echo "Process $1 does not exist."
  exit 1
fi

# Get the username of the new owner
read -p "Enter the username of the new owner: " new_owner

# Check if the username exists
if ! id -u "$new_owner" > /dev/null 2>&1; then
  echo "User $new_owner does not exist."
  exit 1
fi

# Send a SIGTERM signal to the process
kill -TERM "$1"
# Wait for the process to terminate
while kill -0 "$1" > /dev/null 2>&1; do
  sleep 1
done

# Start the process with the new owner
su "$new_owner" -c "command_to_start_process"
```



Q4. Send process to background.

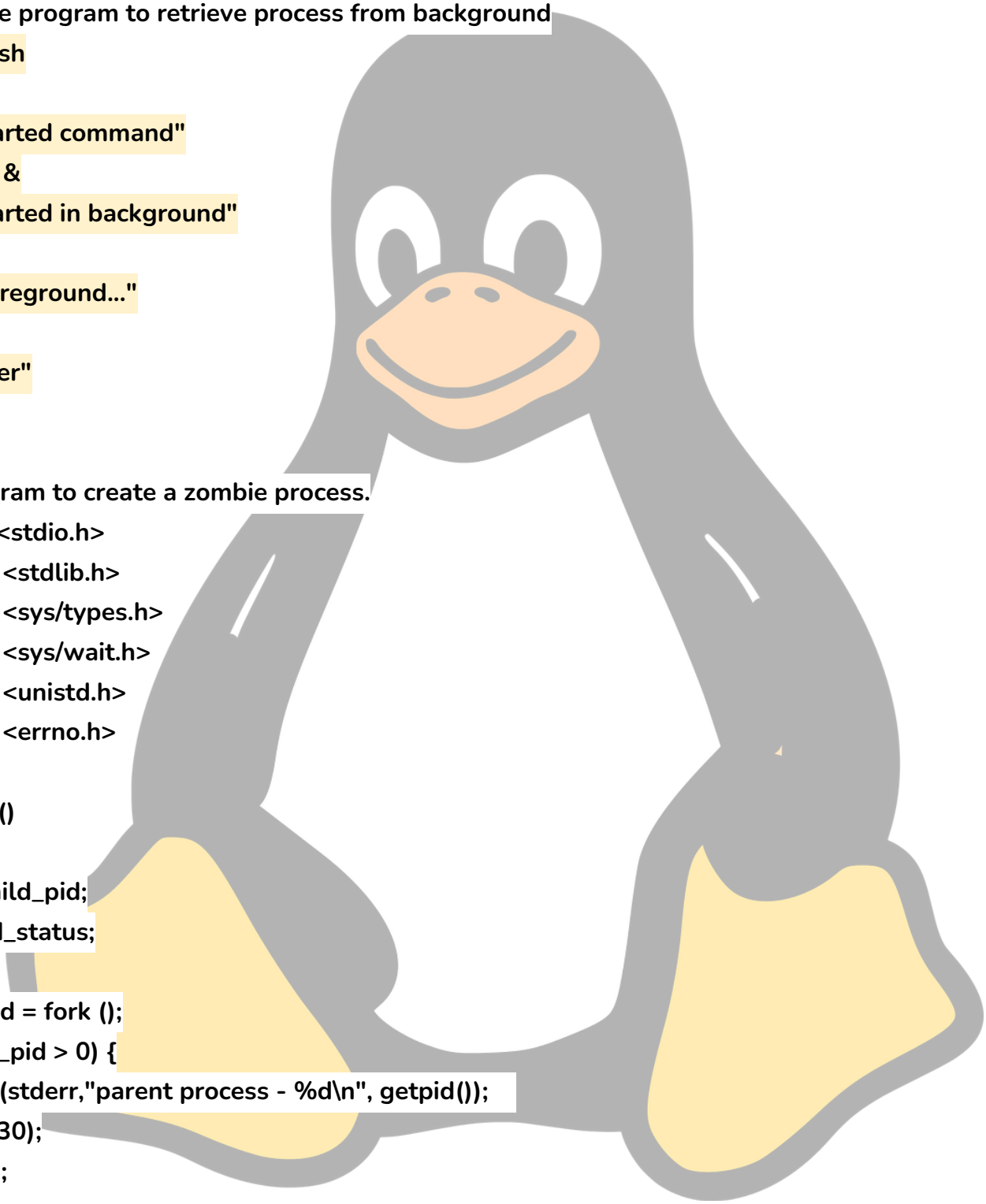
```
#!/bin/bash

echo "Start  command..."
sleep 10 &
echo "background"
```

Q5. Write program to retrieve process from background

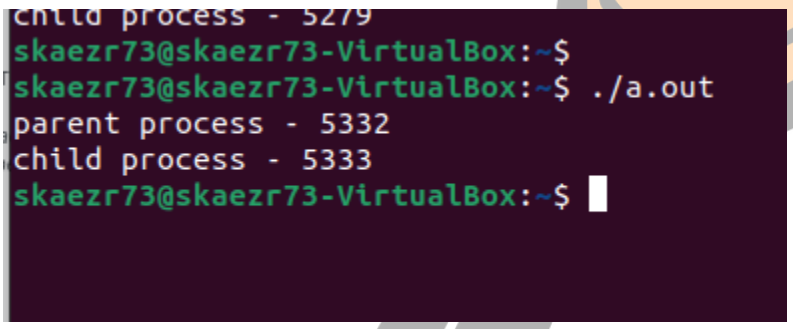
```
#!/bin/bash

echo "started command"
sleep 20 &
echo "started in background"

echo " foreground..."
fg
echo "over"
```

Q6. Program to create a zombie process.

```
#include<stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <errno.h>

int main ()
{
  pid_t child_pid;
  int child_status;

  child_pid = fork ();
  if (child_pid > 0) {
    fprintf(stderr,"parent process - %d\n", getpid());
    sleep(30);
    exit(0);
  }
  else if (child_pid == 0) {

    fprintf(stderr,"child process - %d\n", getpid());
```



```
    exit(0);
  }
  else if (child_pid == -1) {
    perror("fork() call failed");
    exit (-1);
  }
  else {
    fprintf(stderr, "unknown return value of %d from fork() call", child_pid);
    exit (-2);
  }
  return 0;
}
```



Q7. Write a program to create a child process and allow the parent to display "parent" on the screen and child to display " child" on screen.

```
#include <stdio.h>
#include <sys/wait.h>
#include<stdlib.h>
#include<unistd.h>
int main(void)
{
int pid;
int status;
printf("Hello World!\n");
pid = fork( );
if(pid == -1)
{
 perror("bad fork");
 exit(1);
}
 if (pid == 0)  printf("I am the child process.\n");
```

```
else
{
  wait(&status);
  printf("I am the parent process.\n");
}
}
```

```
skaezr73@skaezr73-VirtualBox:~$ gcc proj.c
skaezr73@skaezr73-VirtualBox:~$ gcc proj.c
skaezr73@skaezr73-VirtualBox:~$ ./a.out
Hello World!
I am the child process.
I am the parent process.
skaezr73@skaezr73-VirtualBox:~$
```

EXPERIMENT - 7

Q1. shell script program to check variable attribute of file and processes.

```
 #!/bin/sh
file="/home/skaezr73/muscat.txt"
if [ -r $file ]
then
   echo "File has read access"
else
   echo "File does not have read access"
fi

if [ -w $file ]
then
   echo "File has write permission"
else
   echo "File does not have write permission"
fi

if [ -x $file ]
then
   echo "File has execute permission"
else
   echo "File does not have execute permission"
fi

if [ -f $file ]
```

```
then
   echo "File is an ordinary file"
else
   echo "This is special file"
fi

if [ -d $file ]
then
   echo "File is a directory"
else
   echo "This is not a directory"
fi

if [ -s $file ]
then
   echo "File size is not zero"
else
   echo "File size is zero"
fi

if [ -e $file ]
then
   echo "File exists"
else
   echo "File does not exist"
fi
```

```
skaezr73@skaezr73-VirtualBox:~$ ./linuxx.sh
File has read access
File has write permission
File does not have execute permission
File is an ordinary file
This is not a directory
File size is not zero
File exists
skaezr73@skaezr73-VirtualBox:~$
```

Q2. Shell script program to check and list attributes of process.

```
#!/bin/bash
read -p "Enter process ID (PID): " pid
if ps -p $pid > /dev/null; then
   echo "Process attributes are:"
   ps -o pid , ppid, user, stat, %cpu, %mem, cmd -p $pid
else
   echo "Process does not exist."
```

fi

```
skaezr73@skaezr73-VirtualBox:~$ ./linuxx.sh
Enter process ID (PID): 56
Process does not exist.
skaezr73@skaezr73-VirtualBox:~$
```

Q3. Shell script to implement read,write, execute permissions

```
echo "The name of all files having all permissions :"
for file in *
do
if [ -f $file ]
then
# if all presmission are there
if [ -r $file -a -w $file -a -x $file ]
then
ls -l $file
fi
fi
done
```

```
skaezr73@skaezr73-VirtualBox:~$ ./linuxx.sh
The name of all files having all permissions :
-rwxrwxr-x 1 skaezr73 skaezr73 16016 Apr 10 12:18 a.out
-rwxrwxr-x 1 skaezr73 skaezr73 51 Feb 14 15:10 cro.awk
-rwxrwxr-x 1 skaezr73 skaezr73 197 Apr 25 13:42 linuxx.sh
skaezr73@skaezr73-VirtualBox:~$
```

Q4. change process priority.

```
!/bin/bash
echo "priority changed"
renice -n 19 23900
```

EXPERIMENT - 8

Q1. Execute command using gdb to utilize features like breakpoints ,conditional breakpoints.

gdb -ex 'break 10' ./myprogram

```
skaezr73@skaezr73-VirtualBox:~$ gdb -ex 'break 10' ./linux
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
./linux: No such file or directory.
No symbol table is loaded.  Use the "file" command.
(gdb) run
Starting program:
No executable file specified.
Use the "file" or "exec-file" command.
(gdb)
```

Q2.shell script to include verbose debug option for debugging.

```
#!/bin/bash
flg=false
if [[ $1 == "-d" ]]; then
  flag=true
fi
if [[ $flag == true ]]; then
  set -x # enable verbose debug output
fi
```

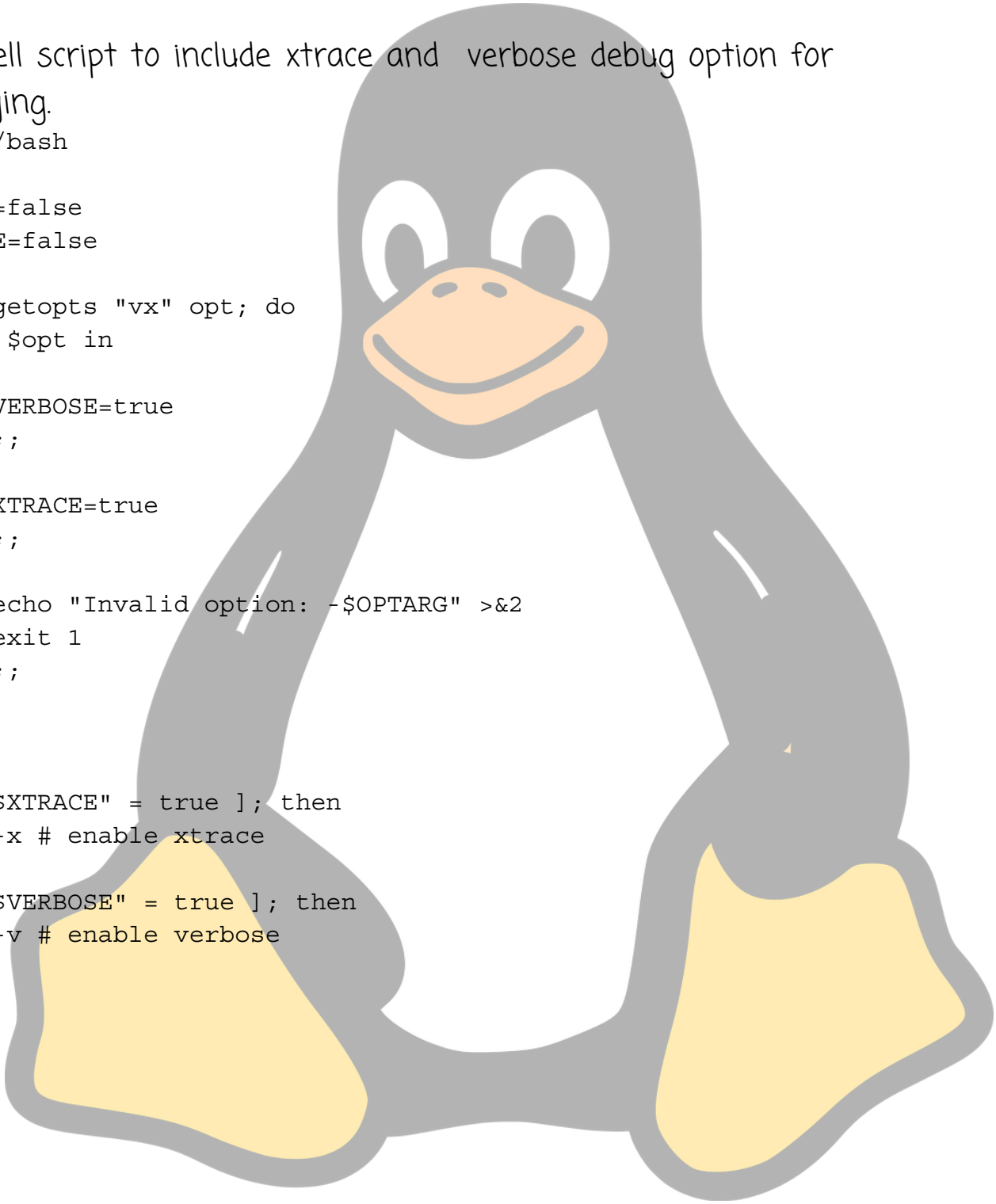
Q3. shell script to include xtrace debug option for debugging.

```
#!/bin/bash
flg=false

while getopts "x" opt; do
  case $opt in
  x)
        flg=true
        ;;
  *)
        echo "Invalid option: -$OPTARG" >&2
        exit 1
        ;;
  esac
```

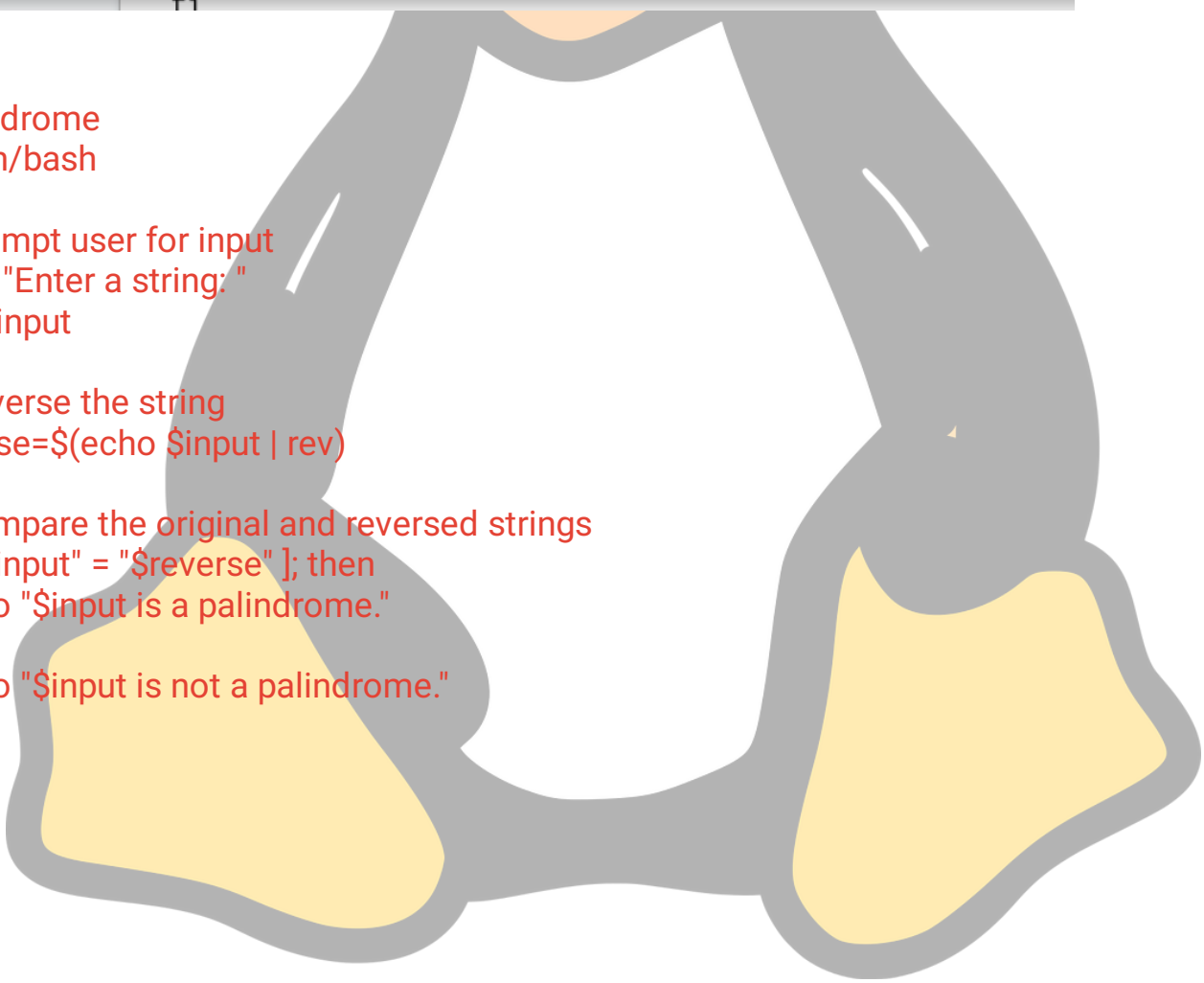
```
done

if [ "$flg" = true ]; then
  set -x # enable xtrace debug output
fi
```

Q4. shell script to include xtrace and verbose debug option for debugging.

```
#!/bin/bash

XTRACE=false
VERBOSE=false

while getopts "vx" opt; do
  case $opt in
    v)
      VERBOSE=true
      ;;
    x)
      XTRACE=true
      ;;
    *)
      echo "Invalid option: -$OPTARG" >&2
      exit 1
      ;;
  esac
done

if [ "$XTRACE" = true ]; then
  set -x # enable xtrace
fi
if [ "$VERBOSE" = true ]; then
  set -v # enable verbose
fi
```



```
^C
skaezr73@skaezr73-VirtualBox:~$
skaezr73@skaezr73-VirtualBox:~$ gdb -ex 'break 10' ./linux
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
./linux: No such file or directory.
No symbol table is loaded.  Use the "file" command.
(gdb)
```



pallindrome

```
#!/bin/bash

# Prompt user for input
echo "Enter a string: "
read input

# Reverse the string
reverse=$(echo $input | rev)

# Compare the original and reversed strings
if [ "$input" = "$reverse" ]; then
  echo "$input is a palindrome."
else
  echo "$input is not a palindrome."
fi
```

Fibonacci

```
#!/bin/bash

# Define the Fibonacci function
fibonacci() {
  if [ $1 -le 1 ]; then
    echo $1
  else
    echo $(( $(fibonacci $(( $1 - 1 )) ) + $(fibonacci $(( $1 - 2 )) ) ))
  fi
}

# Prompt user for input
echo "Enter a number: "
read num

# Call the Fibonacci function with the user's input
result=$(fibonacci $num)

# Print the result
echo "The Fibonacci value of $num is $result."
```

Factorial

```
#!/bin/bash

# Prompt user for input
echo "Enter a number: "
read num

# Initialize variables
factorial=1
counter=1

# Calculate the factorial
while [ $counter -le $num ]
do
  factorial=$(( $factorial * $counter ))
  counter=$(( $counter + 1 ))
done

# Print the result
echo "Factorial of $num is $factorial"
```