Lab Report 2

1. **Create a Process:** code in the figure 1 shows how to create a process

```
#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

int main()

{

 pid_t myPid;

 pid_t myParentPid;

 gid_t myGid;

 uid_t myUid;

 myPid = getpid();

 myParentPid = getppid();

 myGid = getgid();

 myUid = getuid();

 printf( “my process id is %d\n”, myPid );

 printf( “my parent’s process id is %d\n”, myParentPid );

 printf( “my group id is %d\n”, myGid );

 printf( “my user id is %d\n”, myUid );



 return 0;

}
```

Figure 1: A source code to create a process

Description: The compiling output of the code in figure 1 is:

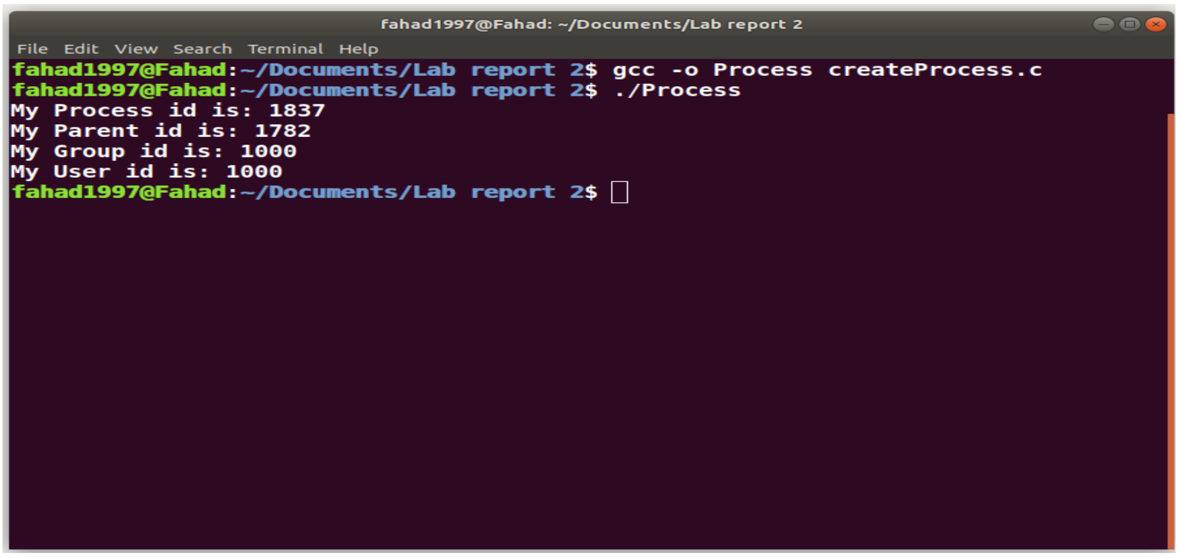
```
$gcc –o process createProcess.c

$ ./process
```

Getpid()-> returns the process id

Getppid()-> returns the process id of the parent of the calling process.

Getgid()-> returns the real group id of the calling process

Getuid()-> returns the real user id of the calling process


fahad1997@Fahad: ~/Documents/Lab report 2
File Edit View Search Terminal Help

```
fahad1997@Fahad:~/Documents/Lab report 2$ gcc -o Process createProcess.c
fahad1997@Fahad:~/Documents/Lab report 2$ ./Process
My Process id is: 1837
My Parent id is: 1782
My Group id is: 1000
My User id is: 1000
fahad1997@Fahad:~/Documents/Lab report 2$ 
```

2. **Create a Child Process:** code in the figure 2 shows how to create a child process

```
#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

int main()

{

 int ret;

 ret= fork();

 if(ret>0)

 {

   printf("I'm parent\n");

   printf("Parent ID: %d\n",getpid());

  }

  if(ret==0)

 {

    printf("I'm child\n");

   printf("Child ID= %d\n",getpid());

   printf("Parent ID= %d\n",getppid());

 }return  0;

}
```

Figure 2: A source code to create a child process

Description: The compiling output of the code in figure 2 is:

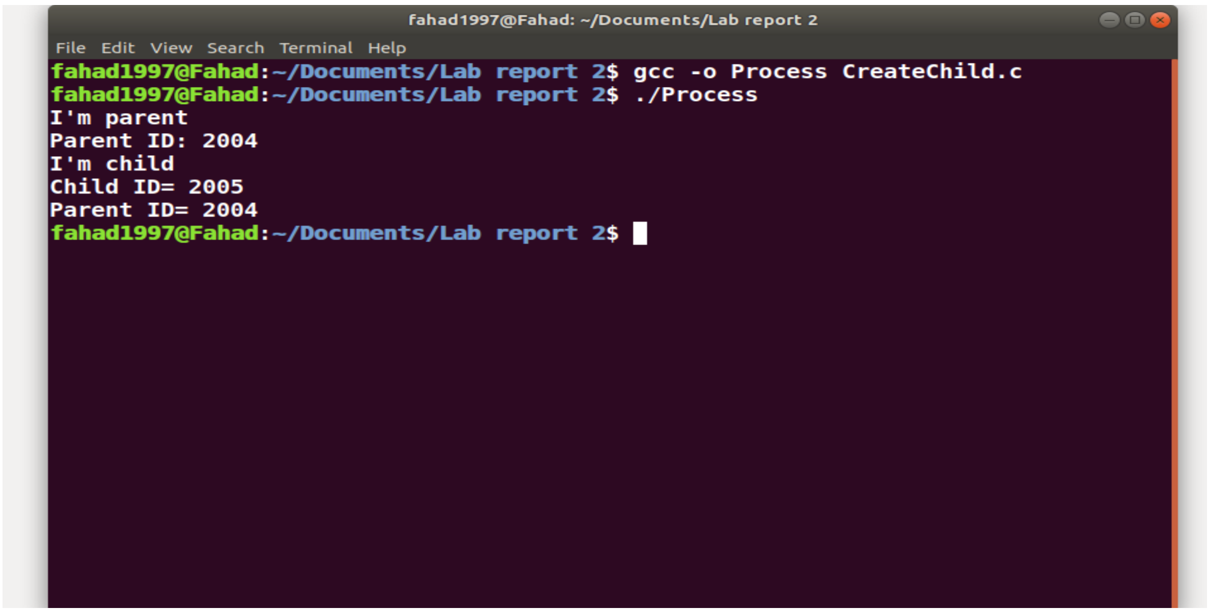
$gcc –o process createChild.c

$ ./process

Getpid()-> returns the process id

Getppid()-> returns the process id of the parent of the calling process.

Fork-> Create a child process


fahad1997@Fahad: ~/Documents/Lab report 2
File Edit View Search Terminal Help

```
fahad1997@Fahad:~/Documents/Lab report 2$ gcc -o Process CreateChild.c
fahad1997@Fahad:~/Documents/Lab report 2$ ./Process
I'm parent
Parent ID: 2004
I'm child
Child ID= 2005
Parent ID= 2004
fahad1997@Fahad:~/Documents/Lab report 2$ 
```

3. **Create multiple child process:** code in the figure 3 shows how to create multiple child process.

```
#include<stdio.h>

int main()

{

  for(int i=0;i<5;i++)

  {

    if(fork() == 0)

    {

      printf("[son] pid %d from [parent] pid %d\n",getpid(),getppid());

      exit(0);

    }

  }

  for(int i=0;i<5;i++)

  wait(NULL);

}
```

Description: The compiling output of the code in figure 3 is:

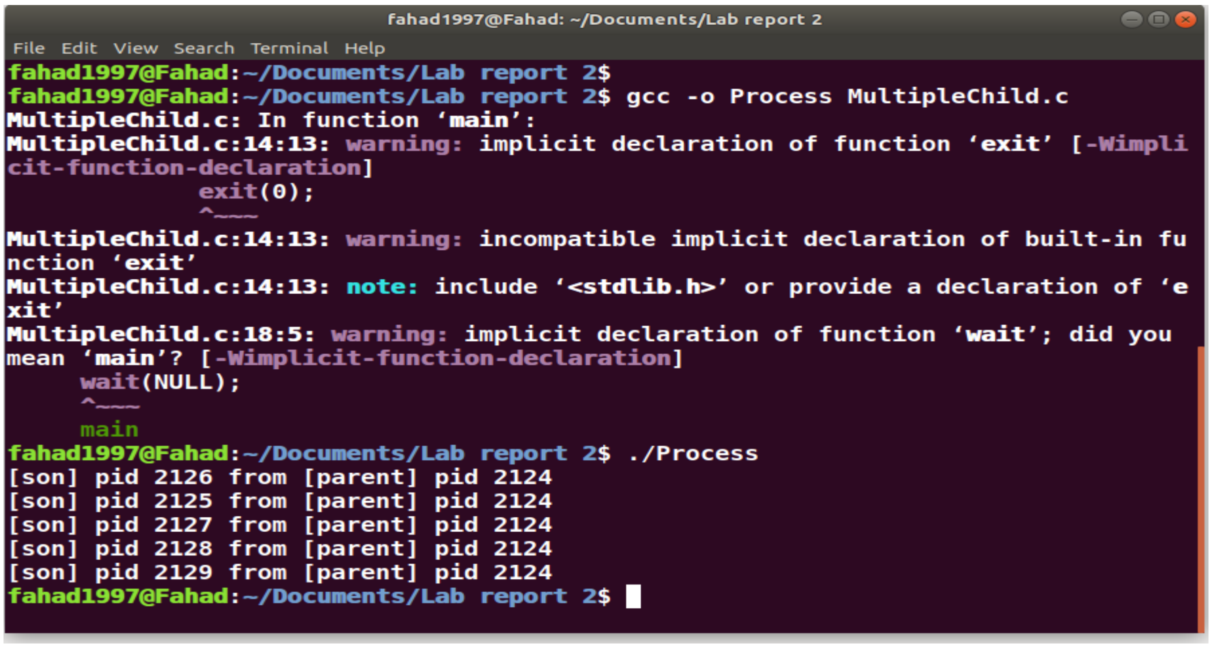
$gcc –o process MultipleChild.c

$ ./process

Getpid()-> returns the process id

Getppid()-> returns the process id of the parent of the calling process.

Fork-> Create a child process


fahad1997@Fahad: ~/Documents/Lab report 2
File Edit View Search Terminal Help

```
fahad1997@Fahad:~/Documents/Lab report 2$
fahad1997@Fahad:~/Documents/Lab report 2$ gcc -o Process MultipleChild.c
MultipleChild.c: In function ‘main’:
MultipleChild.c:14:13: warning: implicit declaration of function ‘exit’ [-Wimpli
cit-function-declaration]
             exit(0);
             ^~~~
MultipleChild.c:14:13: warning: incompatible implicit declaration of built-in fu
nction ‘exit’
MultipleChild.c:14:13: note: include ‘<stdlib.h>’ or provide a declaration of ‘e
xit’
MultipleChild.c:18:5: warning: implicit declaration of function ‘wait’; did you
mean ‘main’? [-Wimplicit-function-declaration]
     wait(NULL);
     ^~~~
     main
fahad1997@Fahad:~/Documents/Lab report 2$ ./Process
[son] pid 2126 from [parent] pid 2124
[son] pid 2125 from [parent] pid 2124
[son] pid 2127 from [parent] pid 2124
[son] pid 2128 from [parent] pid 2124
[son] pid 2129 from [parent] pid 2124
fahad1997@Fahad:~/Documents/Lab report 2$
```

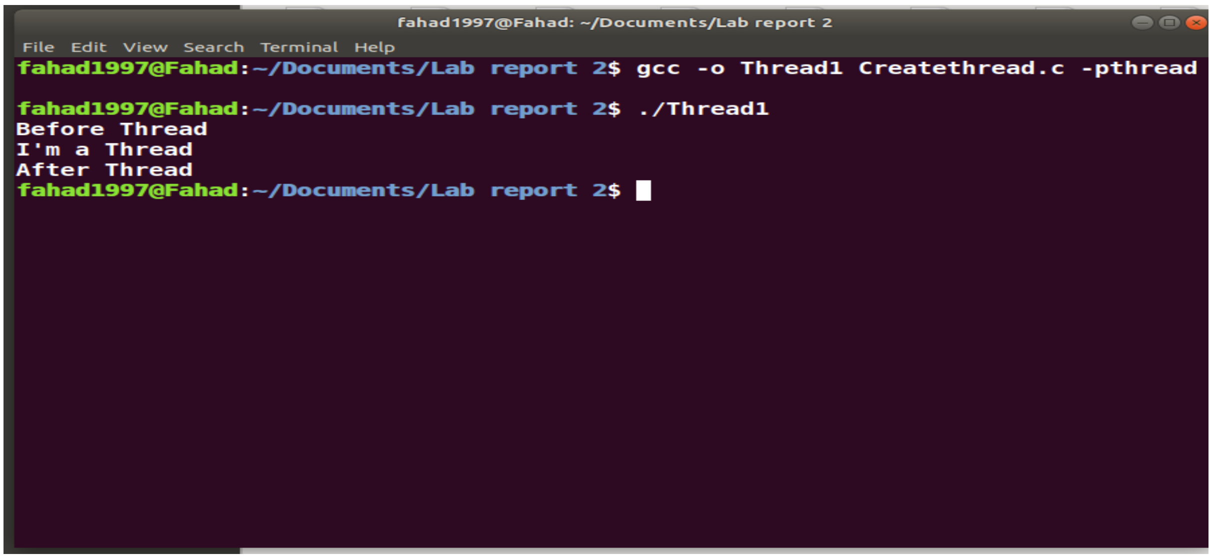
4. **Create a Thread:** code in the figure 4 shows how to create a thread

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

void *myThreadFun(void *vargp)
{
    sleep(1);

    printf("I’m a Thread \n");

    return NULL;
}

int main()
{
    pthread_t thread_id;

    printf("Before Thread\n");

    pthread_create(&thread_id, NULL, myThreadFun, NULL);

    pthread_join(thread_id, NULL);

    printf("After Thread\n");

    exit(0);
}
```

Description: The compiling output of the code in figure 4 is:

$gcc –o Thread1 Createthread.c -pthread

$ ./Thread1



fahad1997@Fahad: ~/Documents/Lab report 2
File Edit View Search Terminal Help

```
fahad1997@Fahad:~/Documents/Lab report 2$ gcc -o Thread1 Createthread.c -pthread

fahad1997@Fahad:~/Documents/Lab report 2$ ./Thread1
Before Thread
I'm a Thread
After Thread
fahad1997@Fahad:~/Documents/Lab report 2$
```

5. **Create Multiple Threads:** code in the figure 5 shows how to create multiple threads.

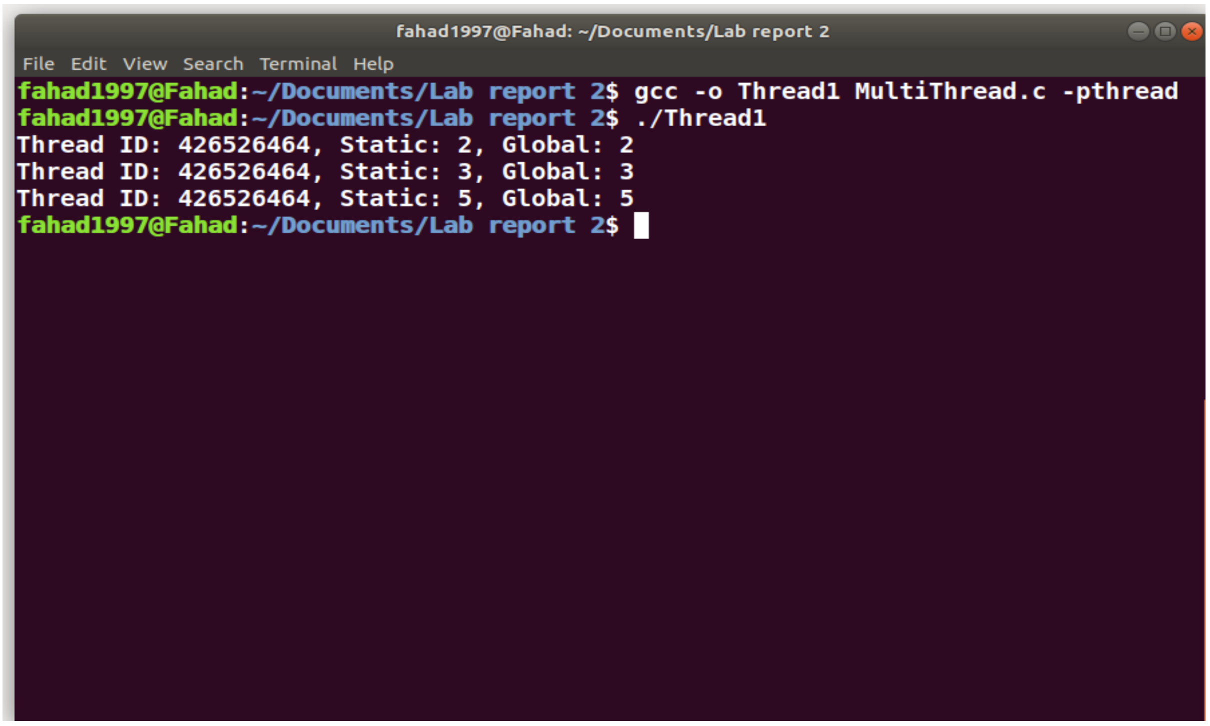
```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

int g = 0;

void *myThreadFun(void *vargp)

{

  int *myid = (int *)vargp;

  static int s = 0;

  ++s; ++g;

  printf("Thread ID: %d, Static: %d, Global: %d\n", *myid, ++s, ++g);

}

int main()

{

  int i;

  pthread_t tid;

  for (i = 0; i < 3; i++)

    pthread_create(&tid, NULL, myThreadFun, (void *)&tid);

  pthread_exit(NULL);

  return 0;

}
```

Figure 5: A source code to create multiple threads

Description: The compiling output of the code in figure 5 is:

$gcc –o Thread1 MultiThread.c -pthread

$ ./Thread1


fahad1997@Fahad: ~/Documents/Lab report 2
File Edit View Search Terminal Help

```
fahad1997@Fahad:~/Documents/Lab report 2$ gcc -o Thread1 MultiThread.c -pthread
fahad1997@Fahad:~/Documents/Lab report 2$ ./Thread1
Thread ID: 426526464, Static: 2, Global: 2
Thread ID: 426526464, Static: 3, Global: 3
Thread ID: 426526464, Static: 5, Global: 5
fahad1997@Fahad:~/Documents/Lab report 2$
```

6. **Access message queue:** code in the figure 6,7,8,9 shows access message queue for inter process communication.

```
#define MAX_LINE 80

#define MY_MQ_ID 111

typedef struct

{

  long type; // Msg Type (> 0)

  float fval; // User Message

  unsigned int uival; // User Message

  char strval[MAX_LINE+1]; // User Message

} MY_TYPE_T;
```

```
#include <stdio.h>

#include <sys/msg.h>

#include "common.h"

int main()

{

 int msgid;

 msgid = msgget( MY_MQ_ID, 0666 | IPC_CREAT );

    if (msgid >= 0) {

       printf( "Created a Message Queue %d\n", msgid );

    }

 return 0;

}
```

Figure7 : A source code to create message

```
#include <sys/msg.h>

#include <stdio.h>

#include "common.h"

int main()

{

 MY_TYPE_T myObject;

 int qid, ret;

 qid = msgget( MY_MQ_ID, 0 )

 if (qid >= 0) {

   myObject.type = 1L;

   myObject.fval = 128.256;

   myObject.uival = 512;

 strncpy( myObject.strval, "This is a test.\n",MAX_LINE );

 ret = msgsnd( qid, (struct msgbuf *)&myObject, sizeof(MY_TYPE_T), 0 );

  if (ret != -1) {

   printf( "Message successfully sent to queue %d\n",qid );

  } } return 0;}
```

Figure 8 : A source code to send message

```
#include <sys/msg.h>

#include <stdio.h>

#include "common.h"

int main()

{

 MY_TYPE_T myObject;

 int qid, ret;

 qid = msgget( MY_MQ_ID, 0 );

 if (qid >= 0) {

  ret = msgrcv( qid, (struct msgbuf *)&myObject,

  sizeof(MY_TYPE_T), 1, 0 );

 if (ret != -1) { printf( "Message Type: %ld\n", myObject.type );

  printf( "Float Value: %f\n", myObject.fval );
```

Figure 9 : A source code to receive message

Description:

- The compiling output of the code in figure 7 is:

```
$gcc –o Process CreateMessage.c

$ ./Process
```

This will create a message

- The compiling output of the code in figure 8 is:

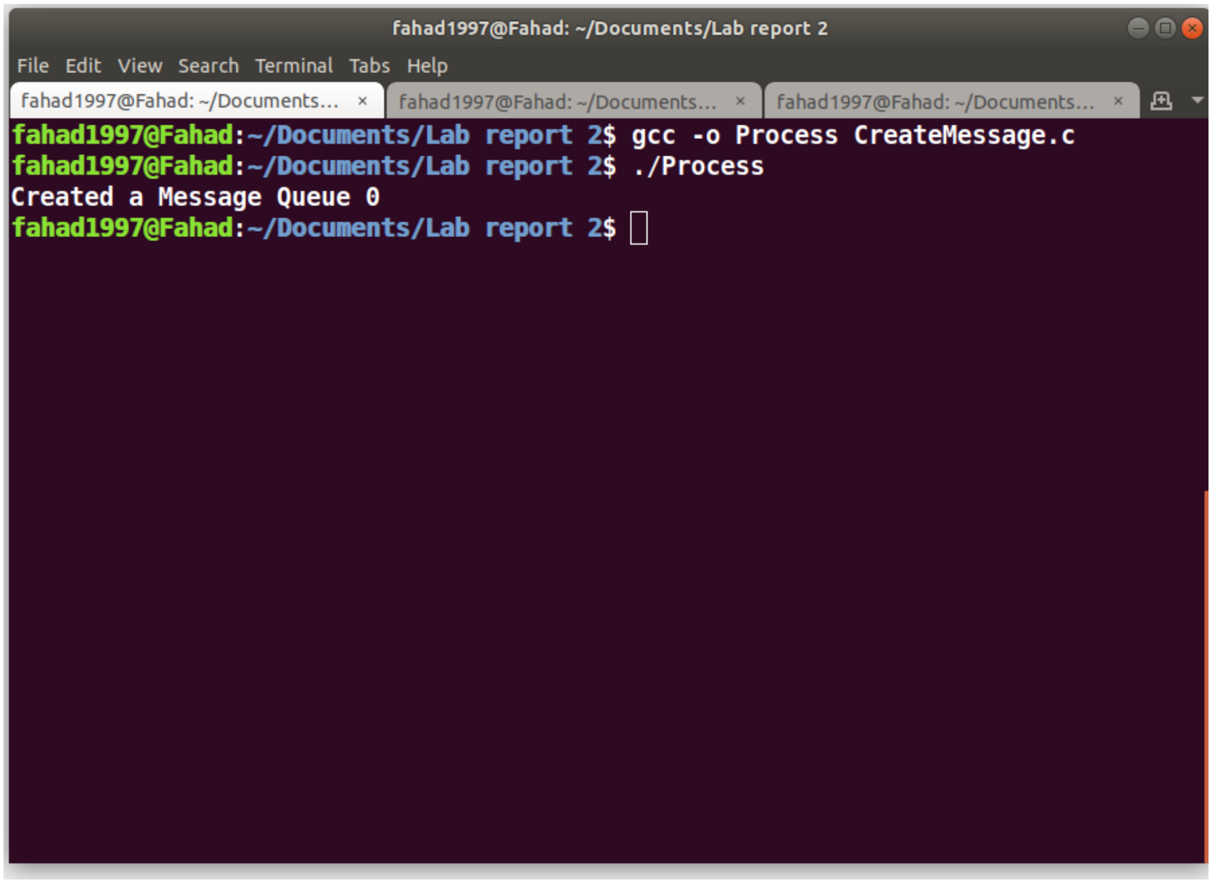
```
$gcc –o Process sendMssg .c

$ ./Process
```

This will send the message that was created in the code in figure 7

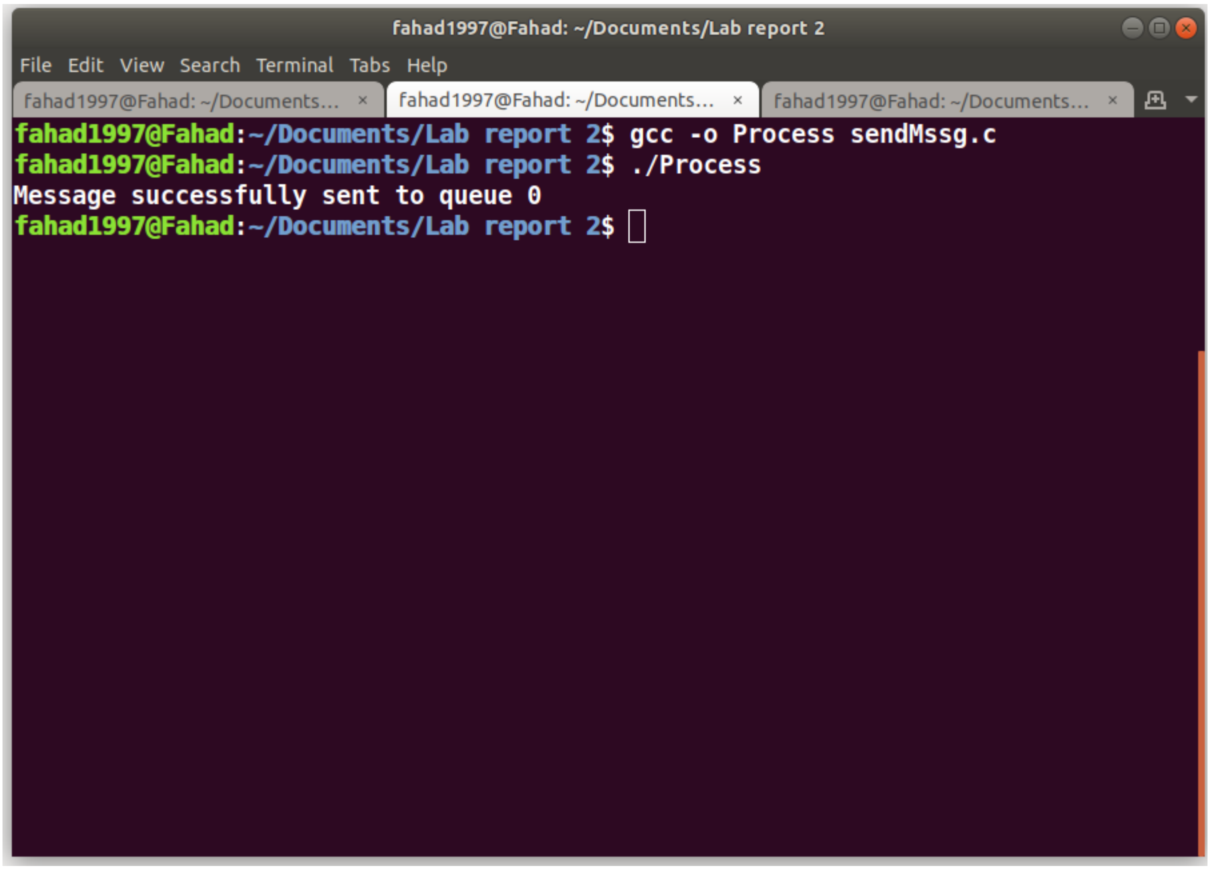
- The compiling output of the code in figure 9 is:

```
$gcc –o Process receiveMssg .c

$ ./Process
```

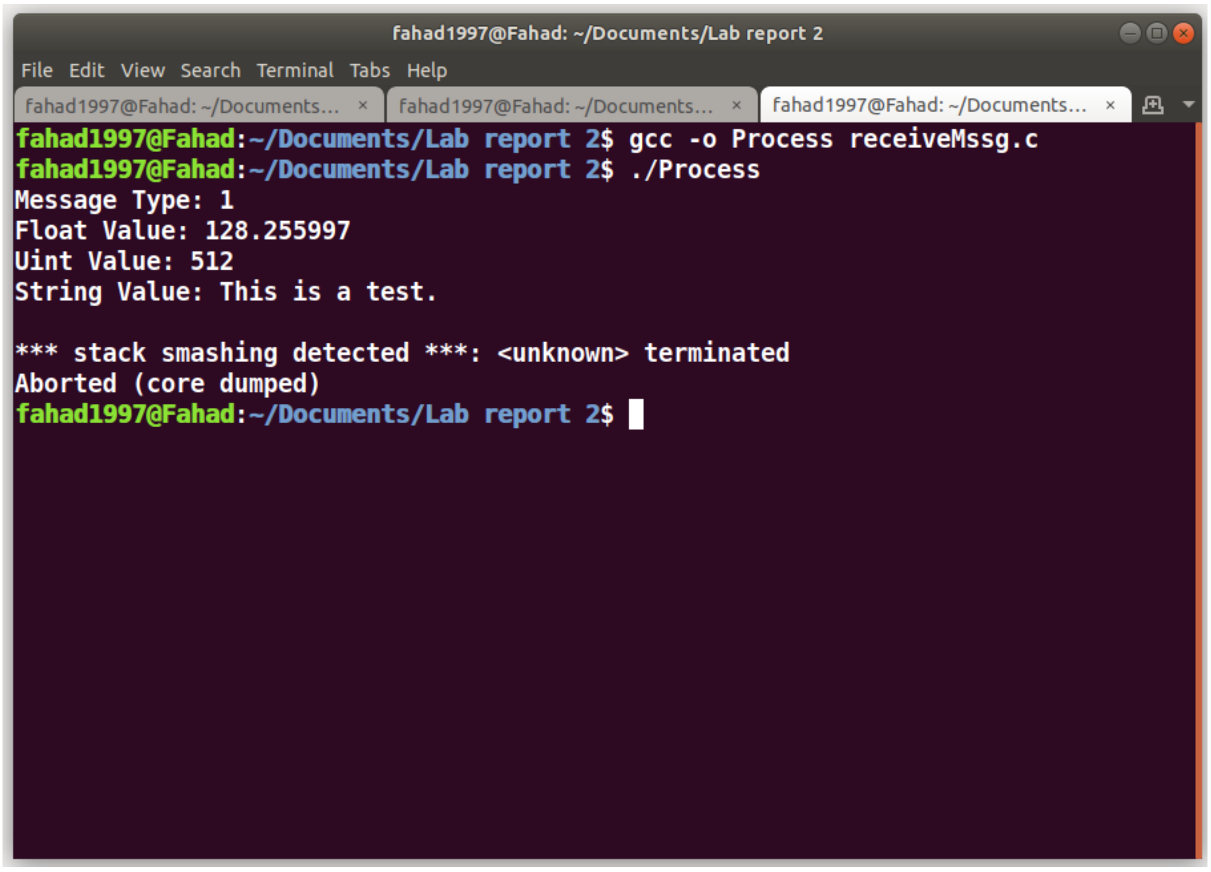
Receive the message that was sent in the code in figure 8, receiver can read the message


fahad1997@Fahad: ~/Documents/Lab report 2
File Edit View Search Terminal Tabs Help
fahad1997@Fahad: ~/Documents...
fahad1997@Fahad: ~/Documents...
fahad1997@Fahad: ~/Documents...

```
fahad1997@Fahad:~/Documents/Lab report 2$ gcc -o Process CreateMessage.c
fahad1997@Fahad:~/Documents/Lab report 2$ ./Process
Created a Message Queue 0
fahad1997@Fahad:~/Documents/Lab report 2$ 
```


fahad1997@Fahad: ~/Documents/Lab report 2
File Edit View Search Terminal Tabs Help
fahad1997@Fahad: ~/Documents...
fahad1997@Fahad: ~/Documents...
fahad1997@Fahad: ~/Documents...

```
fahad1997@Fahad:~/Documents/Lab report 2$ gcc -o Process sendMssg.c
fahad1997@Fahad:~/Documents/Lab report 2$ ./Process
Message successfully sent to queue 0
fahad1997@Fahad:~/Documents/Lab report 2$ 
```


fahad1997@Fahad: ~/Documents/Lab report 2
File Edit View Search Terminal Tabs Help
fahad1997@Fahad: ~/Documents…
fahad1997@Fahad: ~/Documents…
fahad1997@Fahad: ~/Documents…

```
fahad1997@Fahad:~/Documents/Lab report 2$ gcc -o Process receiveMssg.c
fahad1997@Fahad:~/Documents/Lab report 2$ ./Process
Message Type: 1
Float Value: 128.255997
Uint Value: 512
String Value: This is a test.

*** stack smashing detected ***: <unknown> terminated
Aborted (core dumped)
fahad1997@Fahad:~/Documents/Lab report 2$
```