**NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY**

(AN AUTONOMOUS INSTITUTION)

(AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA)

******

**OS MINI PROJECT**

**ON**

**“INTERPROCESS COMMUNICATION”**

***SUBMITTED BY:*** **REG NO. USN:**

MOHAN SAI KIRAN B 17692 1NT16CS056

PAWAN R TANKSALI 17992 1NT16CS070

*Submitted in partial fulfillment of the requirement for the award of Degree of BACHELOR OF COMPUTER SCIENCE AND ENGINEERING*

Department of Computer Science and Engineering

Nitte Meenakshi Institute of Technology,

Yelahanka, Bangalore– 560064

Academic Year 2018-2019

**NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY**

(AN AUTONOMOUS INSTITUTION)

(AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA)



**CERTIFICATE**

**This is to certify that the OS Mini Project Report entitled**

**“INTERPROCESS COMMUNICATION”**

is an authentic work carried out by,

***NAME:***  **REG NO. USN**

MOHAN SAI KIRAN B 17692 1NT16CS056

PAWAN R TANKSALI 17992 1NT16CS070

In partial fulfillment of the requirements for the completion of OS Project for the academic year 2018-2019.

Name & Signature of the Guide Name & Signature of HOD

**Mrs. Chaitra H V Dr. Thippeswamy M N**

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guide and encouragement crowned our efforts with success.

We would like express our gratitude to our **Principal**, **Dr. H C NAGARAJ**.

We would like to thank our **HOD, Dr. Thippeswamy MN** for his valuable support without whose support this project would not have been possible.

We would also like to express our greatest thanks to our guide, **Mrs. Chaitra HV** for giving us such a concept to work on and for her valuable effort, support and guidance without which this project would not have been possible.

We would like to thank our friends and parents for providing inspiration to successfully complete the project.

We would like to mention special thanks to the staff of **CSE Department, NMIT** for their support.

**ABSTRACT**

Processes execute to accomplish specified computations. An interesting and innovative way to use a computer system is to spread a given computation over several processes. The need for such communicating processes arises in parallel and distributed processing contexts. Often it is possible to partition a computational task into segments which can be distributed amongst several processes. Clearly, these processes would then form a set of communicating processes which cooperate in advancing a solution. In a highly distributed, multi-processor system, these processes may even be resident on different machines. In such a case the communication is supported over a network. It also covers many basic concepts such as:

* How to spawn (or create) a new process.
* How to assign a task for execution to this newly spawned process.
* A few mechanisms to enable communication amongst processes.
* Synchronization amongst these processes.

This Mini Project is an elaboration of how Inter-process communication can be achieved and also about communication in client-server systems.

The purpose of this project is to understand how processes can communicate using shared memory. The techniques can be used for communication in client-server systems. It also explores about the strategies for communication in client-server systems that is using sockets pipes.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **Description** | **Page No.** |
|  | INTRODUCTION  1.1 OBJECTIVE | 1 |
|  | QUESTIONS AND ANSWERS  2.1 QUESTION 1  2.1.1 CODE  2.1.2 SNAPS  2.2 QUESTION 2  2.2.1 CODE  2.2.2 SNAPS | 2  6 |
|  | CONCLUSION | 10 |
| 4. | REFERENCES | 11 |

**INTRODUCTION**

Inter-process communication or Interprocess communication (IPC) refers specifically to the mechanisms an operating system provides to allow the processes to manage shared data. Typically, applications can use IPC, categorized as clients and servers, where the client requests data and the server responds to client requests. Many applications are both clients and servers, as commonly seen in distributed computing. Methods for doing IPC are divided into categories which vary based on software requirements, such as performance and modularity requirements, and system circumstances, such as network bandwidth and latency.

**Message passing** allows multiple programs to communicate using message queues and/or non-OS managed channels, commonly used in concurrency models.

**Sockets** are data sent over a network interface, either to a different process on the same computer or to another computer on the network. Stream-oriented (TCP; data written through a socket requires formatting to preserve message boundaries) or more rarely message-oriented (UDP, SCTP).

* 1. **1.1 OBJECTIVE**

The main objective of this Mini Project is to,

* Understanding of Inter-process communication.
* Creating processes.
* Communication using shared memory segment.
* Information retrieval will become easy.
* Reduce delay in processing time.
* Use sockets to communicate between the processes running on different hosts.

# Questions and Answers

## 2.1 Question 1

1. Create two processes which communicate using a shared memory segment. The first process finds out the list of all processes running on the system with their name, process id, number of files opened and total time running and creates a linked list containing this data about every process running in the shared memory. The second process reads this linked list and formats it in HTML and saves it in a file. This is done by the processes every half an hour. (Hint: Use jproc filesystem on Linux to find out process information.)

### 2.1.1 Code

**To Generate Data**

import subprocess

import pickle

import time

from copy import deepcopy

def write\_process():

response = subprocess.check\_output(['ps','aux'])

response = str(response)

processes = response.split('\\n')

processes = processes[1:-1]

num\_of\_processes = len(processes)

# print(num\_of\_processes)

process\_json = {'name\_of\_process':'','process\_id':'','process\_time':''}

all\_processes = []

for process in processes:

row\_json = deepcopy(process\_json)

li = process.split()

pid = li[1]

time = li[9]

name = ''.join(li[10:])

row\_json['name\_of\_process'] = name

row\_json['process\_id'] = pid

row\_json['process\_time'] = time

all\_processes.append(row\_json)

process\_data = {'number\_of\_processes': num\_of\_processes,

'processes\_info':all\_processes}

with open('process\_data.pkl', 'wb') as f:

pickle.dump(process\_data, f, pickle.HIGHEST\_PROTOCOL)

return

def main():

while(1):

write\_process()

time.sleep(10)

main()

**To Generate Webpage**

import pickle

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def display\_process\_data():

html = ''

with open('process\_data.pkl', 'rb') as f:

process\_data = pickle.load(f)

num\_process = process\_data['number\_of\_processes']

processes = process\_data['processes\_info']

html = html + '<h1>Number of processes running currently are ' + str(num\_process) + '</h1>\n\n'

html = html + '''<table border=2>

<thead>

<tr>

<th>Process ID</th>

<th>Time running</th>

<th style="text-align: left;">Process name</th>

</tr>

</thead>

<tbody>'''

all\_rows = ''

for i in range(len(processes)-1,0,-1):

row\_value = '<tr><th>' + str(processes[i]['process\_id']) + '</th><th>' + str(processes[i]['process\_time']) + '</th><th style="text-align: left;">' + str(processes[i]['name\_of\_process']) + '</th></tr>'

all\_rows += row\_value

html = html + all\_rows

html = html + '''

</tbody>

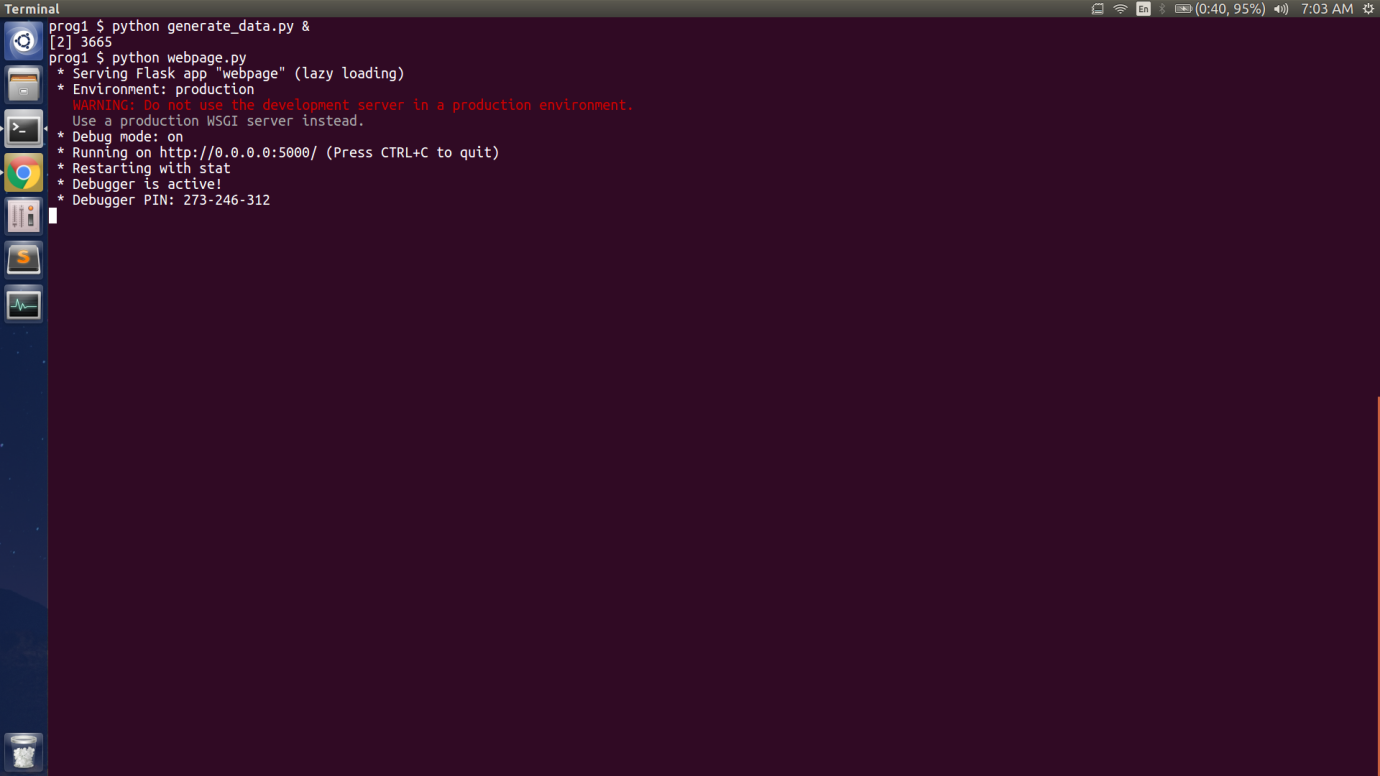
</table>'''

return html

if \_\_name\_\_ == '\_\_main\_\_':

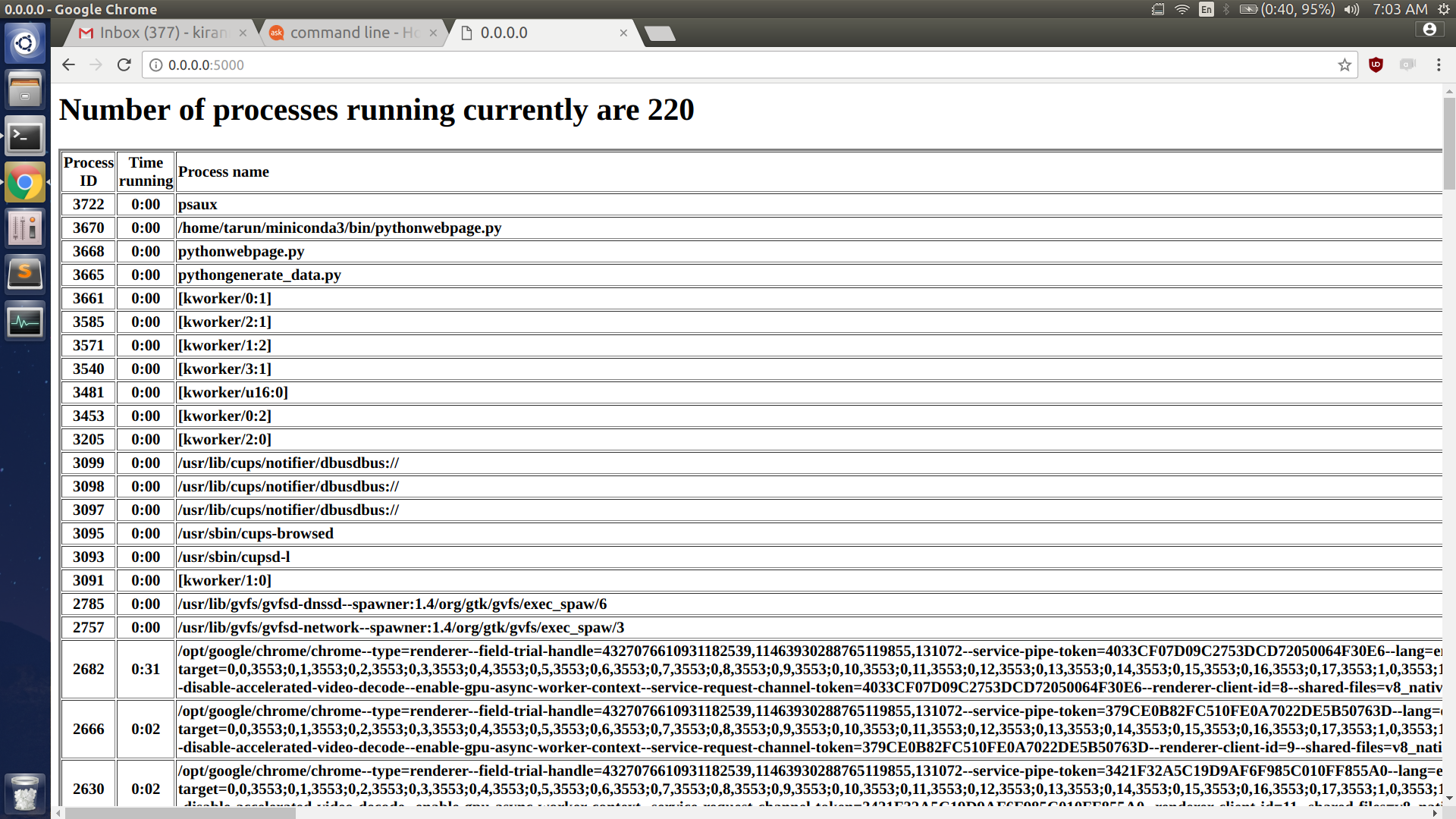
app.run(host='0.0.0.0',debug=True)

### 2.1.2 Snaps



**Fig 1. Running of generate\_data.py and webpage.py**

Figure describes how the data is generated and displayed on the webpage.



**Fig 2. Figure showing the data on webpage**

Figure shows the information which is displayed which includes the process ID and the time running as well as process name.

## 

## 2.2 Question 2

Create a process which creates a list of prime numbers in a given range of numbers. The process has the ability to run in two modes - supervisor mode and slave mode. In the supervisor mode the process takes a range of numbers from the user between which prime numbers have to be calculated. It also takes a list of tuples containing the hostname and port numbers. Each of this host runs another similar process in slave mode. The supervisor divides the range into equal parts and sends each range to different slave processes. The supervisor and the slave then calculate the prime numbers in their respective ranges. The slaves return back the prime numbers found to the supervisor. When all slaves have communicated back the prime numbers, the supervisor prints the list in ascending.

### 2.2.1 Code

**Master**

#include <stdio.h>

#include<stdlib.h>

#include <arpa/inet.h>

int main(int argc, char\* argv[])

{

char fname[50], buffer[1025];

int sd, n , low, high,arr[100],prime,i,j,m;

struct sockaddr\_in address;

address.sin\_family = AF\_INET;

address.sin\_port = htons(15000);

address.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

if((sd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

perror("socket");

connect(sd, (struct sockaddr\*)&address, sizeof(address));

printf("Enter lower limit :");

scanf("%d",&low);

printf("Enter upper limit :");

scanf("%d",&high);

FILE \*f = fopen("data.txt", "w");

printf("Enter a filename: ");

scanf("%s", fname);

printf("Sending request... \n");

send(sd, fname, sizeof(fname), 0); // send a message on a socket

printf("Received response: \n");

while((m = recv(sd, buffer, sizeof(buffer), 0)) > 0) ; // receive a message from a socket

for(n=low+1; n<high; n++)

{

prime = 1;

for(i=2; i<n/2; i++)

if(n%i == 0)

{

prime = 0;

break;

}

if(prime)

{

arr[j]=n;

j++;

}

}

for(i=0;i<11;i++)

printf("%d\n",arr[i]);

return 0;

}

**Slave**

#include <stdio.h>

#include <arpa/inet.h>

#include <fcntl.h>

#include<stdlib.h>

int main()

{

char fname[50], buffer[1025],s1[100];

int sd, source, size, file, n,num, arr[100],lower,upper,prime,i,j=0;

struct sockaddr\_in address;

address.sin\_family = AF\_INET;

address.sin\_port = htons(15000);

address.sin\_addr.s\_addr = INADDR\_ANY;

printf("Waiting for request... \n");

sd = socket(AF\_INET, SOCK\_STREAM, 0);

bind(sd, (struct sockaddr\*)&address, sizeof(address));

listen(sd, 3);

source = accept(sd, (struct sockaddr\*)NULL, NULL);

recv(source, fname, sizeof(fname), 0);

printf("Received request for: %s \n", fname);

FILE \*fptr;

fptr = fopen(fname,"r");

printf("\nPerforming Operations...\n");

while ( (num = getw(fptr)) != EOF ) {

lower=num;

upper=lower;

}

lower=lower-40;

for(n=lower+1; n<upper; n++)

{

prime = 1;

for(i=2; i<n/2; i++)

if(n%i == 0)

{

prime = 0;

break;

}

if(prime)

{

arr[j]=n;

j++;

}}

for(i=0;i<11;i++){

s1[i]=arr[i];

send(source, buffer, s1[i], 0);

}

fclose(fptr);

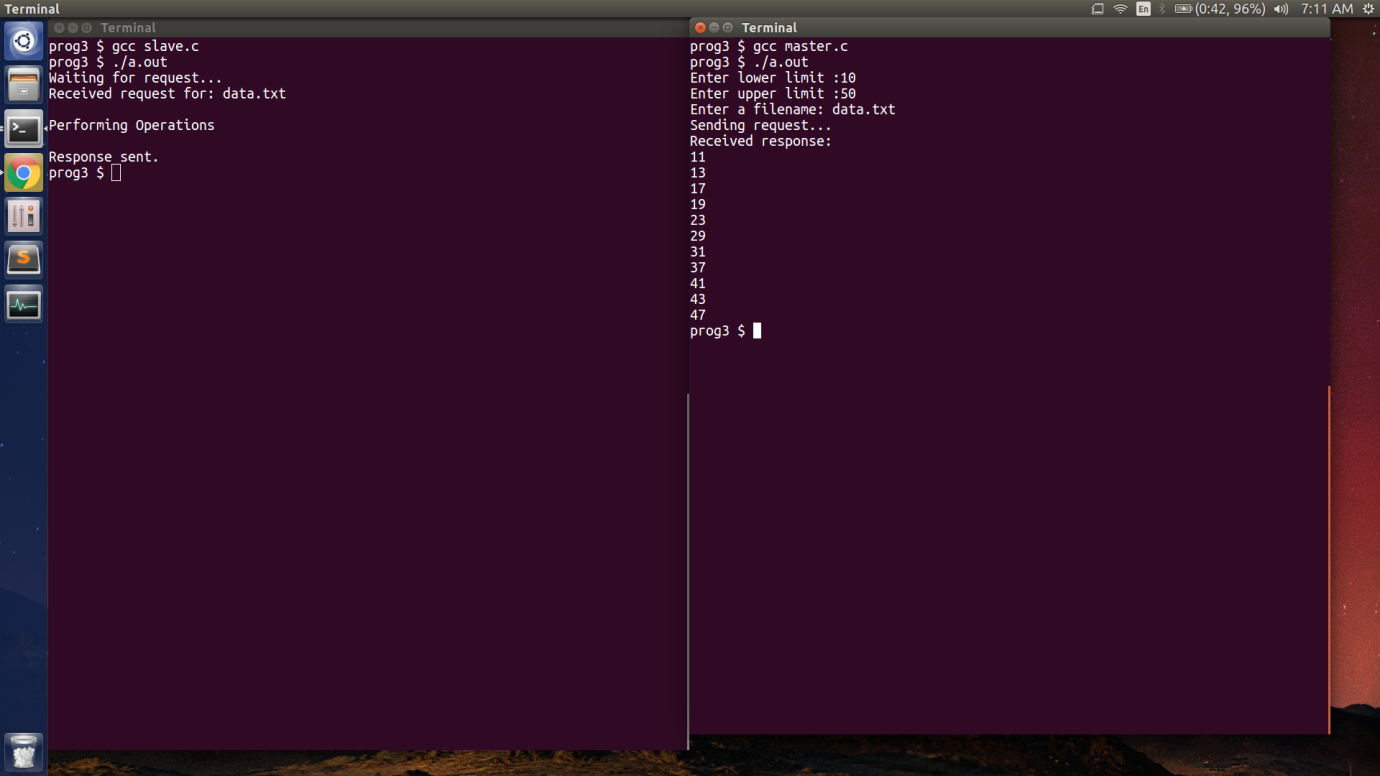
printf("\nResponse sent. \n");

//close(file);

return 0;

}

### 2.2.2 Snaps



**Fig 3. Figure showing the input taken from the user**

Figure above shows the request for response for slave program which acknowledges about the operations performed and as well as the response is sent to the master program. The master program takes the upper and the lower limit and prints the received response.

# CONCLUSION

From this project, we have learnt the importance of the coding language C and python. There are numerous benefits from learning this language, however, the most important benefit is that the C and python programming language is recognized worldwide, one of the most powerful and is used in a multitude of applications, including advanced scientific systems, games, creations of simulations, operating systems and many more.

We have also, through this project understood in depth how the Inter-process communication works, it’s different kinds. How client server communication can be achieved using sockets.

# REFERENCES

1. Operating System Concepts by Abraham Silberschatz, Peter Baer Galvin, Greg Gagne
2. YouTube videos

* GATE Smasher
* Knowledge GATE