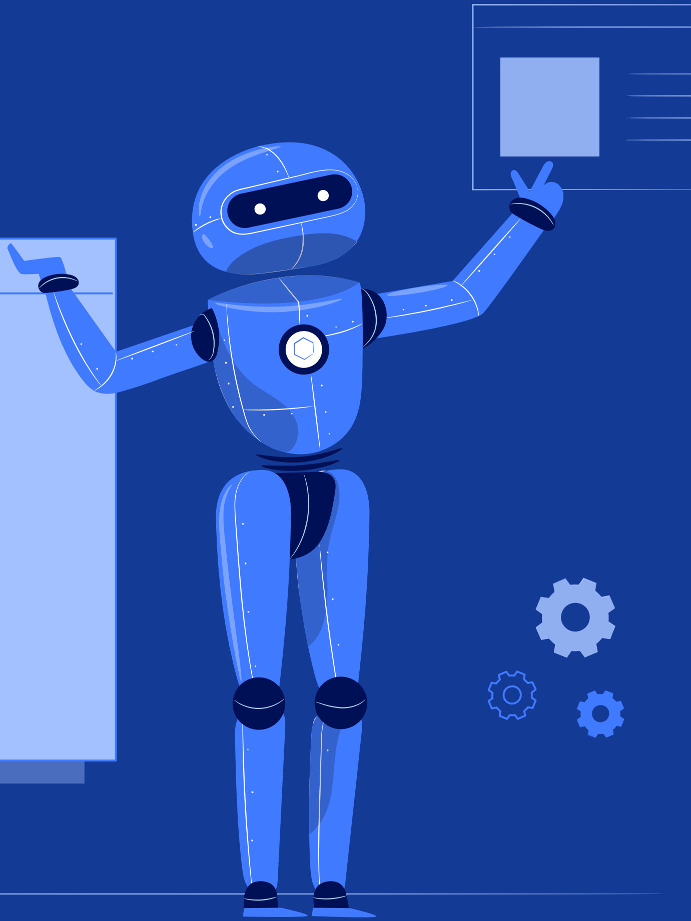
Operating Systems Lab - 4

Group - 6



Members

Abhijeet Sonkar	(IIB2019009)
Avneesh Kumar	(IIB2019010)
Ambika Singh	(IIB2019017)
Aditya Aggarwal	(IIT2019210)
Divy Agrawal	(IIT2019211)

Question Statement

Using the concepts of shared-memory based programming, design and implement a client-server based project.

Components:

1.	service1.c	find LCM of 4 given numbers
2.	service2.c	find GCD of 2 numbers
3.	service3.c	find LCM and GCD of 2 numbers
4.	client.c	to create particular request to server
5.	server.c	to manage all client requests

Table of Contents

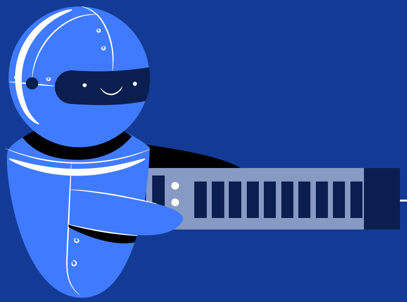
01 Basic Concepts

02 Code Explanation

03 Output Screenshots

01

Basic Concepts



Basic Concepts

exec syscalls

The exec family of system calls replaces the program executed by a process. When a process calls exec, all code (text) and data in the process is lost and replaced with the executable of the new program.
The exec() functions return only if an error has occurred. The return value is -1, and errno is set to indicate the error.

execl execle execlp execv execve execvp are some syscalls in exec family.

e: It is an array of pointers that points to environment variables and is passed explicitly to the newly loaded process.

l: l is for the command line arguments passed a list to the function
p: p is the path environment variable which helps to find the file passed as an argument to be loaded into process.
v: v is for the command line arguments. These are passed as an array of pointers to the function.

Shared memory IPC

Inter Process Communication through shared memory is a concept where two or more process can access the common memory. And communication is done via this shared memory where changes made by one process can be viewed by another process.

Shared memory provides a way by letting two or more processes share a memory segment. With Shared Memory the data is only copied twice – from input file into shared memory and from shared memory to the output file.

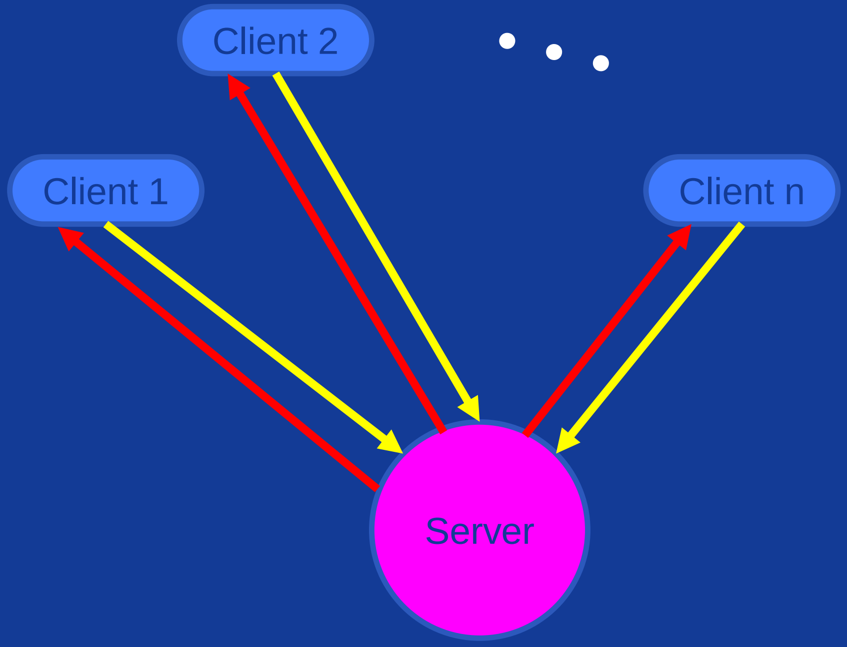
Syscalls for Shm-IPC

- ftok(),
- shmget(),
- shmat(),
- shmdt(),
- shmctl()

Client-Server system

- Client-server relationship is a relation in which one program (client) requests a service or resource from another program (server).

- Advantage of the client-server model is that its centralized architecture helps make it easier to protect data with access controls that are enforced by security policies.

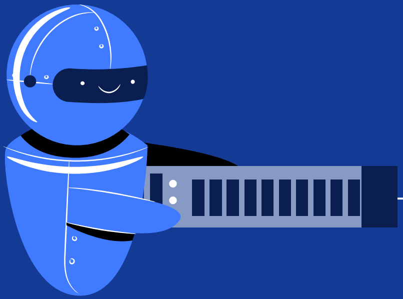


🟨 Client requesting a service from server

🟥 Server granting service to client

02

Code Explanation



server.c

msg_buffer structure is used here to store information related to a request made by client, while creating a request, client process provides process id(PID), service_no, sharedmemory id(shmid), and inputs needed (service_input) for invoking particular service.

```
struct msg_buffer{
    int service_no;
    int PID;
    int shmid;
    int service_input[4];
} message;
```

Array arr is implemented as a circular queue of type msg_buffer, which basically stores client's request information. front, end represent start and end index of circular queue respectively. Lock is used to create mutual exclusion between other clients and server.

```
typedef struct{
    struct msg_buffer arr[100000];
    int front;
    int end;
    pthread_mutex_t lock;
}queue;
```

queue *q; declares a variable of type queue
shmid_queue stores the shared memory id for queue.

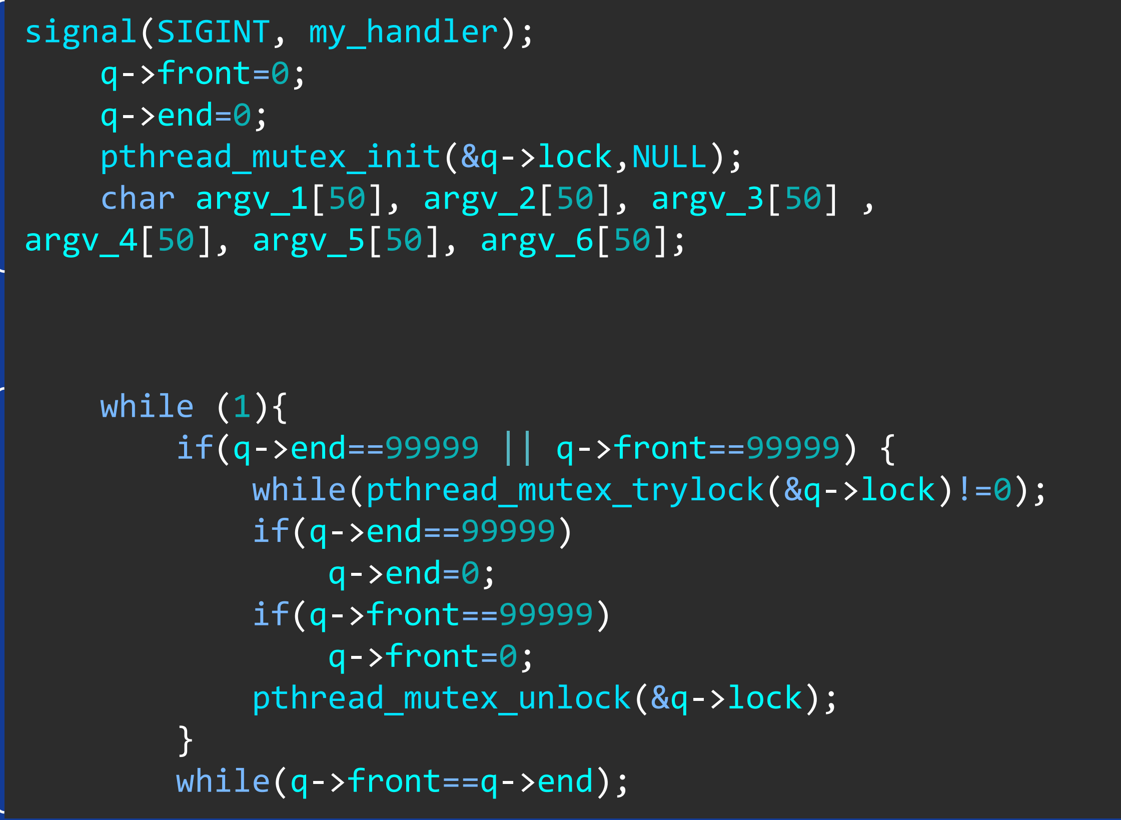
```
queue *q;
int shmid_queue;
```

```
void my_handler(){
    pthread_mutex_destroy(&q->lock);
    shmctl(shmid_queue, IPC_RMID, NULL);
    printf("Server ended\n");
    exit(1);
}
```

This handler is called when ctrl+c is pressed in server process, it destroys the mutex lock from queue, clears shared memory allotted to request queue, and quits server program.

- Initiate interrupt handler
- Initially queue is empty so, set front and end to 0
- Initiate mutex lock.
- argv_... store the parameters to be passed to exec sys call.

- Implementing circular queue using array
- Using mutex locking to ensure that no other process changes, data in queue.
- Last line is used here to wait for client process to request a service when, there are no processes pending in request queue.

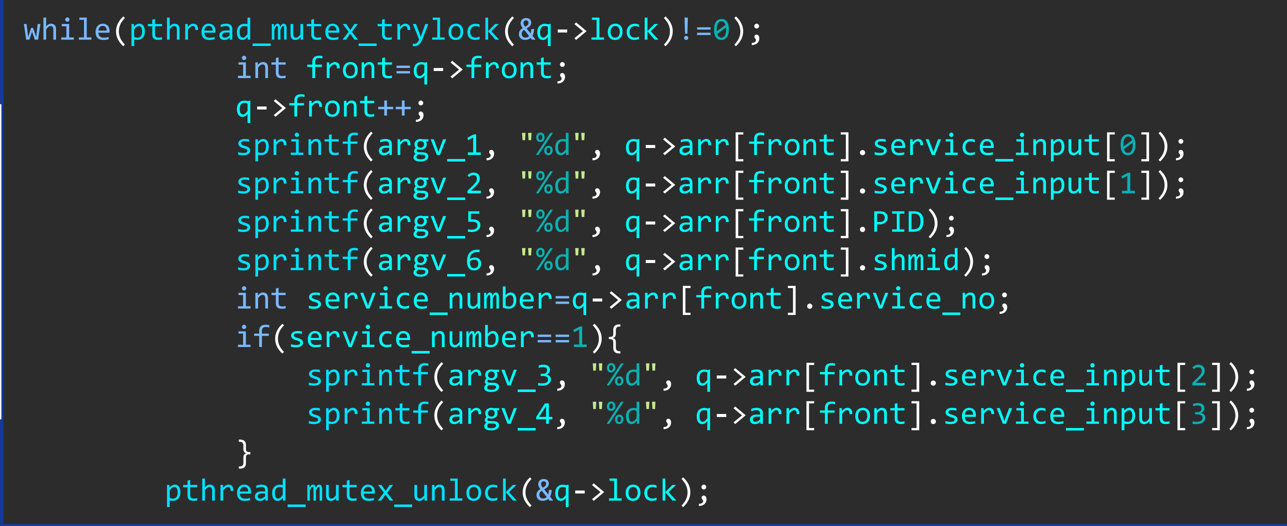


```
signal(SIGINT, my_handler);
    q->front=0;
    q->end=0;
    pthread_mutex_init(&q->lock,NULL);
    char argv_1[50], argv_2[50], argv_3[50] ,
argv_4[50], argv_5[50], argv_6[50];



    while (1){
        if(q->end==99999 || q->front==99999) {
            while(pthread_mutex_trylock(&q->lock)!=0);
            if(q->end==99999)
                q->end=0;
            if(q->front==99999)
                q->front=0;
            pthread_mutex_unlock(&q->lock);
        }
        while(q->front==q->end);
```

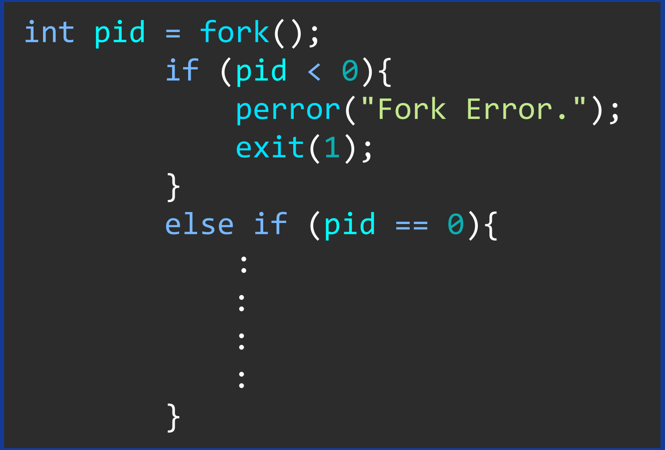
- Tries to lock the mutex

- Get the oldest request from queue
- And other info like PID, shmid, etc.

- And at last unlocks the mutex on queue.



```
while(pthread_mutex_trylock(&q->lock)!=0);
            int front=q->front;
            q->front++;
            sprintf(argv_1, "%d", q->arr[front].service_input[0]);
            sprintf(argv_2, "%d", q->arr[front].service_input[1]);
            sprintf(argv_5, "%d", q->arr[front].PID);
            sprintf(argv_6, "%d", q->arr[front].shmid);
            int service_number=q->arr[front].service_no;
            if(service_number==1){
                sprintf(argv_3, "%d", q->arr[front].service_input[2]);
                sprintf(argv_4, "%d", q->arr[front].service_input[3]);
            }
        pthread_mutex_unlock(&q->lock);
```



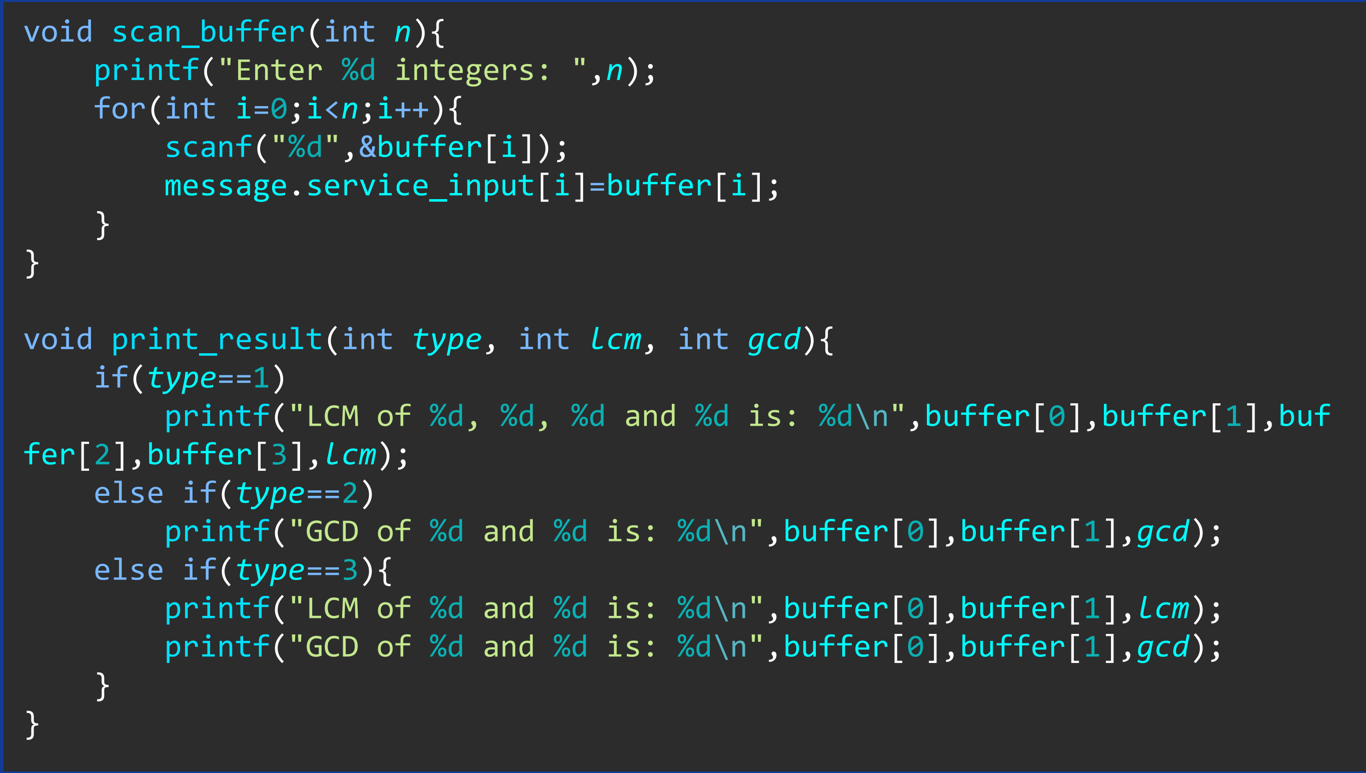
```
int pid = fork();
        if (pid < 0){
            perror("Fork Error.");
            exit(1);
        }
        else if (pid == 0){
            :
            :
            :
            :
        }
```

- Once all the details of client process, and its service request are gathered, using `fork()` we create a new process.
- If new process is not created, then result is shown.
- If child process is created, then particular program is invoked using `execl()` syscall, within child process.

client.c

- scan_buffer is used here to take input of certain no. of items in buffer array.

- print_result is used here to print the result, after response is received from server for particular service



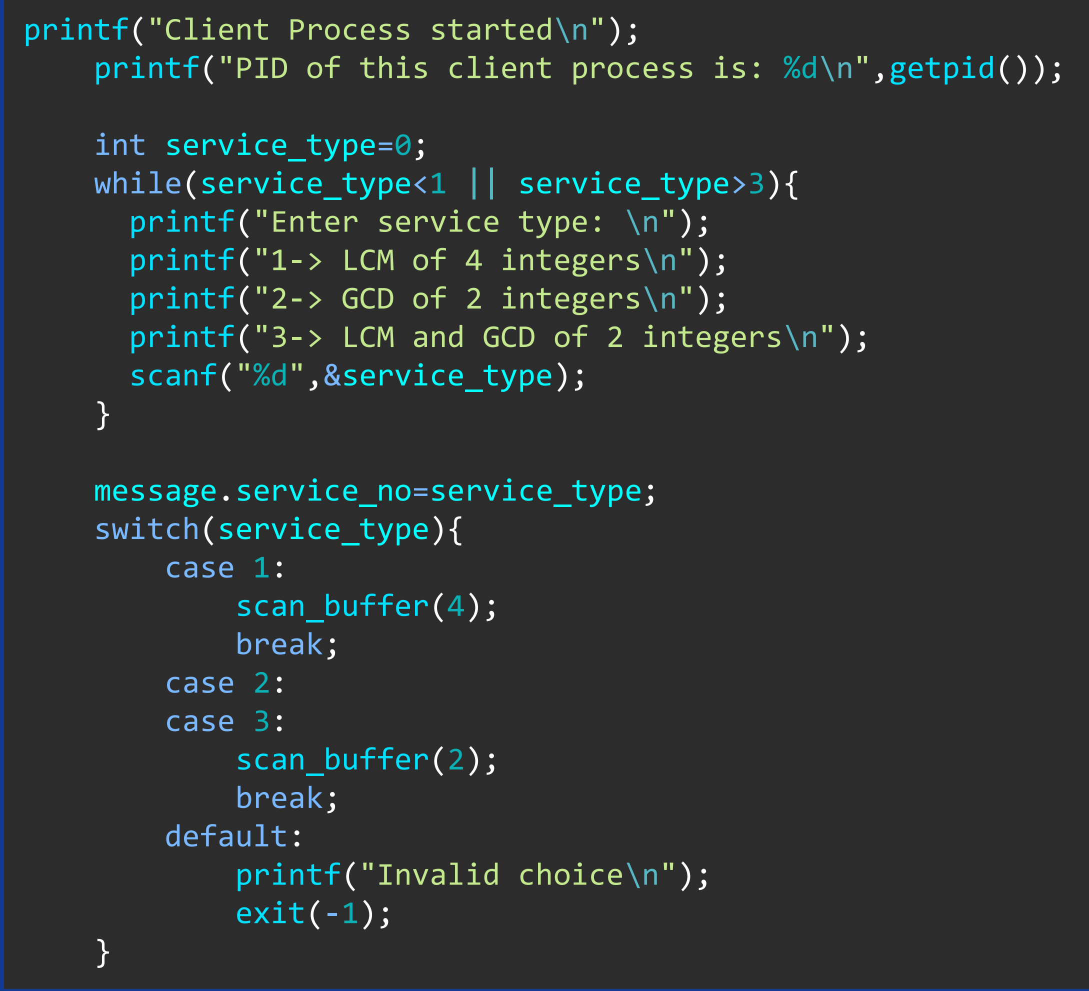
```c
void scan_buffer(int n){
    printf("Enter %d integers: ",n);
    for(int i=0;i<n;i++){
        scanf("%d",&buffer[i]);
        message.service_input[i]=buffer[i];
    }
}

void print_result(int type, int lcm, int gcd){
    if(type==1)
        printf("LCM of %d, %d, %d and %d is: %d\n",buffer[0],buffer[1],buffer[2],buffer[3],lcm);
    else if(type==2)
        printf("GCD of %d and %d is: %d\n",buffer[0],buffer[1],gcd);
    else if(type==3){
        printf("LCM of %d and %d is: %d\n",buffer[0],buffer[1],lcm);
        printf("GCD of %d and %d is: %d\n",buffer[0],buffer[1],gcd);
    }
}
```

- PID of client process is printed on screen

- Initiate service_type with 0.
- Keep taking service_type until a correct service type is provided by user i.e. 1, 2 or 3.

- service_type is saved as service_no in message variable of type msg_buffer.

- Then according to service_type, number of inputs are taken, from user using scan_buffer().



```
printf("Client Process started\n");
    printf("PID of this client process is: %d\n",getpid());

    int service_type=0;
    while(service_type<1 || service_type>3){
      printf("Enter service type: \n");
      printf("1-> LCM of 4 integers\n");
      printf("2-> GCD of 2 integers\n");
      printf("3-> LCM and GCD of 2 integers\n");
      scanf("%d",&service_type);
    }

    message.service_no=service_type;
    switch(service_type){
        case 1:
            scan_buffer(4);
            break;
        case 2:
        case 3:
            scan_buffer(2);
            break;
        default:
            printf("Invalid choice\n");
            exit(-1);
    }
```

- Once we have gathered all information required to invoke a service using, server, we try to lock the mutex.
- Once the mutex is locked, we add all relate details like PID, service_no, service_input etc. to queue.
- Now after successful write operation, we release the lock.

```
while(pthread_mutex_trylock(&q->lock)!=0);
        q->arr[q->end].PID = getpid();
        q->arr[q->end].service_no = service_type;
        q->arr[q->end].shmid=shmid;
        for(int i=0;i<4;i++)
            q->arr[q->end].service_input[i]=buffer[i];
        q->end++;
    pthread_mutex_unlock(&q->lock);
```

- Now the processed is paused, after putting its request to server
- When server encounters a request of this client, it invokes a relevant service.
- Service after successful completion, sends a signal to client(this) process, with result in result_buffer.
- Now, we print the result using print_result().
- And then we free result_buffer from memory, using shmctl()

```
pause();

    switch(service_type){
        case 1:
            print_result(1,result_buffer[0],0);
            break;
        case 2:
            print_result(2,0,result_buffer[0]);
            break;
        case 3:
            print_result(3,result_buffer[0],result_buffer[1])
;
            break;
        default:
            printf("Invalid choice\n");
            exit(-1);
    }
    shmctl(shmid, IPC_RMID, NULL);
```

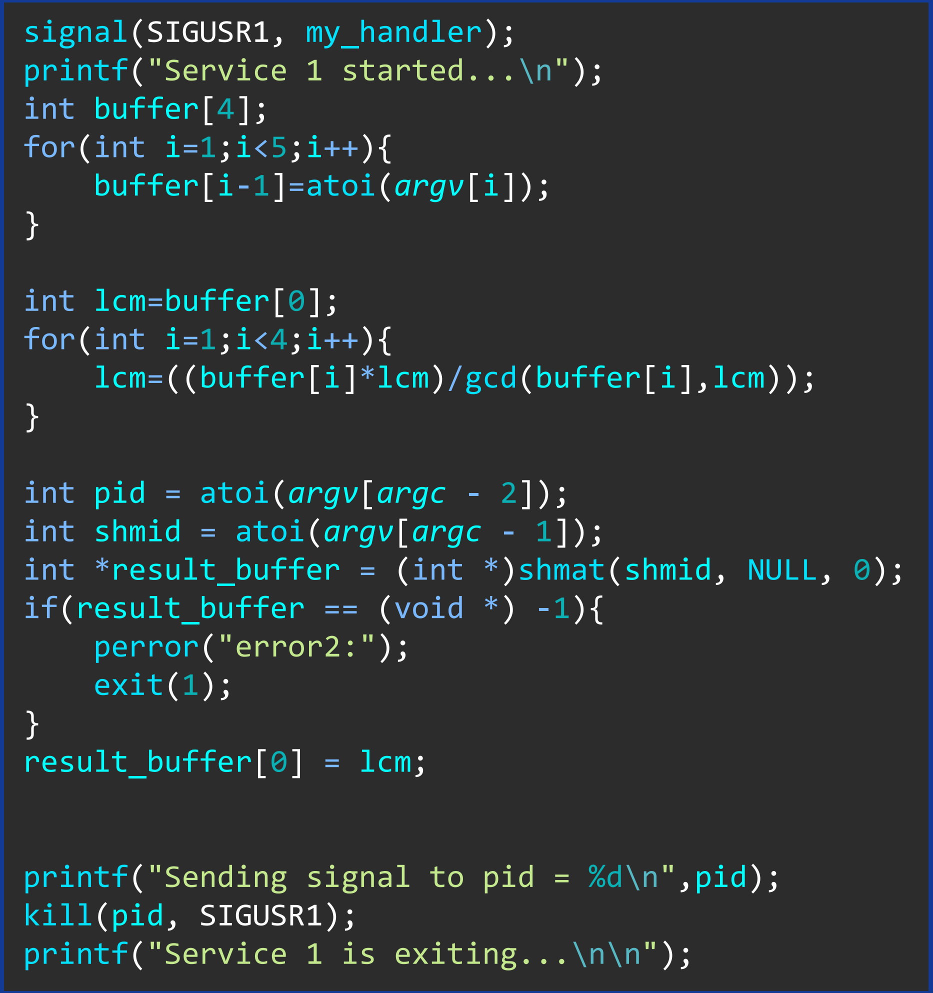
service1.c

- Required argv elements are copied to buffer array.

- Particular calculation is performed, in this case we have to find lcm of 4 number so, we perform that operation, and store result in lcm variable.

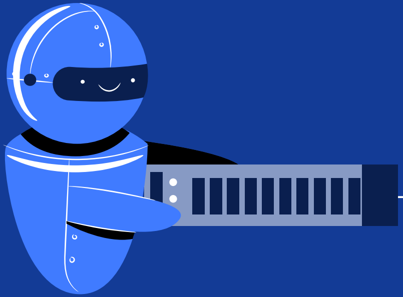
- pid of client process(from where request is generated) is stored in pid variable if int datatype.
- Similarly shmid of result buffer is stored in shmid variable of type int.
- Shared memory is created of same shmid to create same result_buffer which is at client.

- Using kill(), we signal the client process with pid to continue its execution.
- Same applies for other services also.

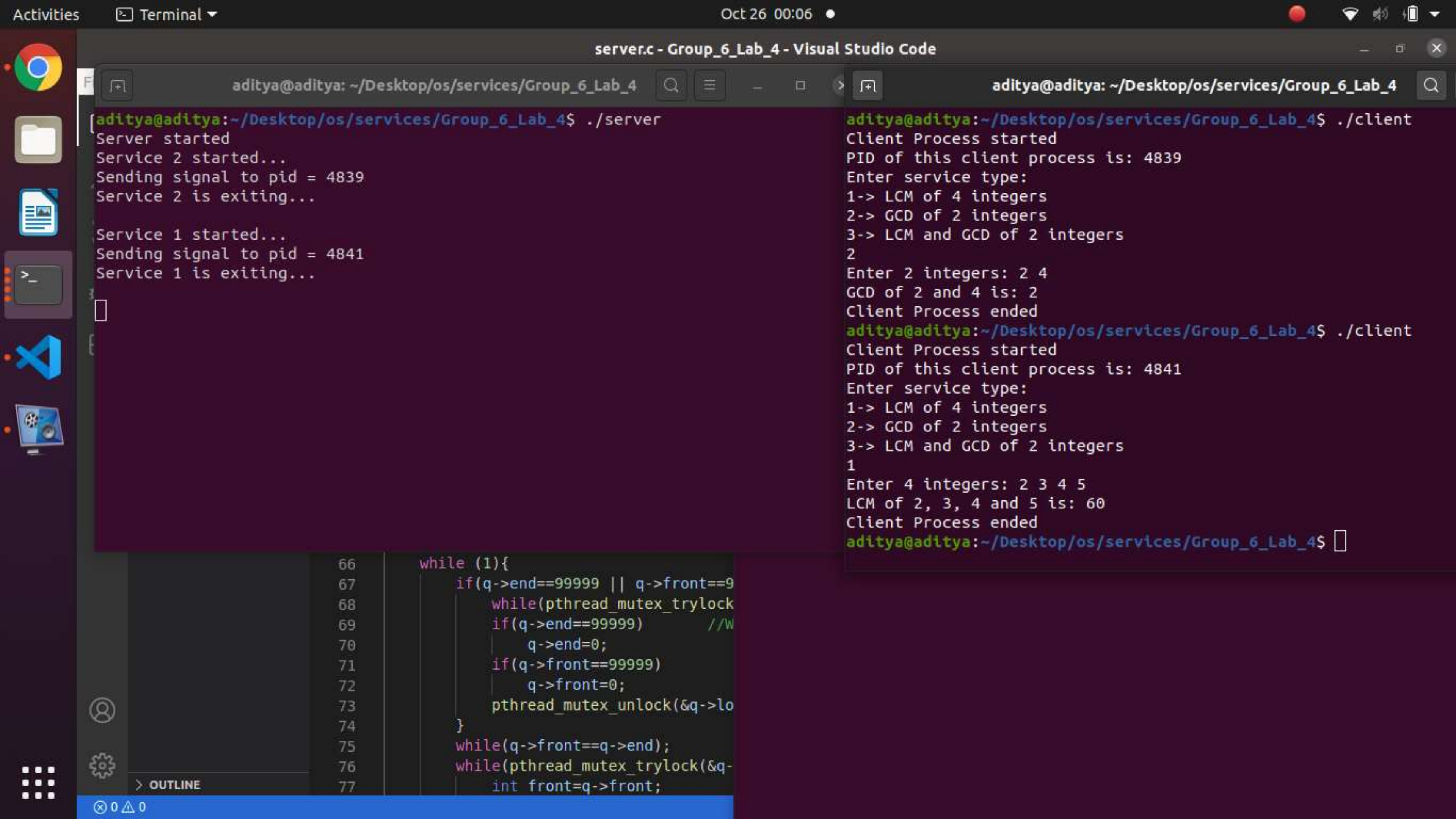


```
signal(SIGUSR1, my_handler);
printf("Service 1 started...\n");
int buffer[4];
for(int i=1;i<5;i++){
    buffer[i-1]=atoi(argv[i]);
}

int lcm=buffer[0];
for(int i=1;i<4;i++){
    lcm=((buffer[i]*lcm)/gcd(buffer[i],lcm));
}

int pid = atoi(argv[argc - 2]);
int shmid = atoi(argv[argc - 1]);
int *result_buffer = (int *)shmat(shmid, NULL, 0);
if(result_buffer == (void *) -1){
    perror("error2:");
    exit(1);
}
result_buffer[0] = lcm;


printf("Sending signal to pid = %d\n",pid);
kill(pid, SIGUSR1);
printf("Service 1 is exiting...\n\n");
```

03 Output Screenshots*



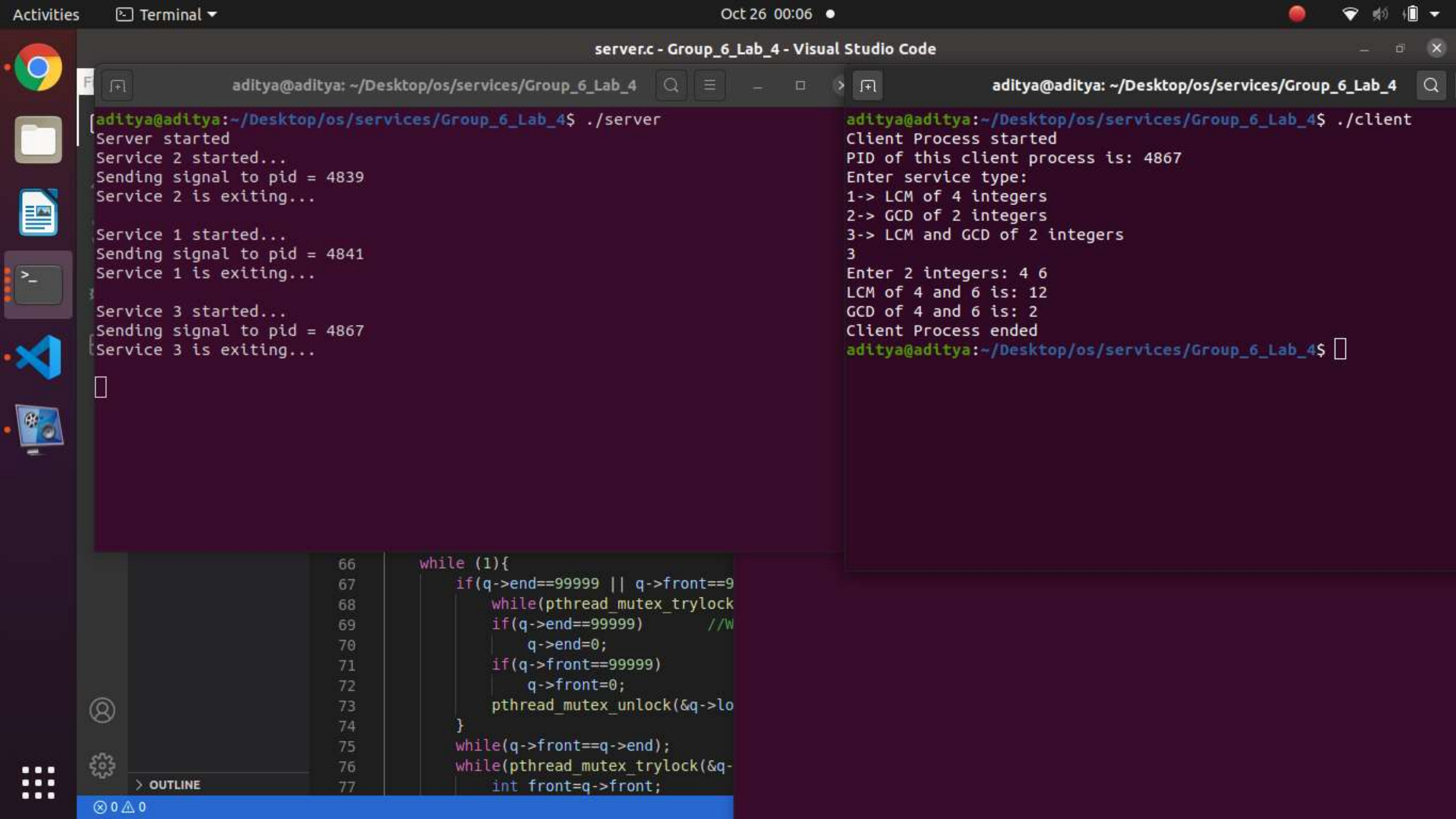
* *In ppt we are showing few screenshots of output, we have also attached a video demonstrating the complete work.*


Activities
Terminal
Oct 26 00:06
server.c - Group_6_Lab_4 - Visual Studio Code
aditya@aditya: ~/Desktop/os/services/Group_6_Lab_4
aditya@aditya: ~/Desktop/os/services/Group_6_Lab_4
OUTLINE
⊗ 0 ⚠ 0

```
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./server
Server started
Service 2 started...
Sending signal to pid = 4839
Service 2 is exiting...

Service 1 started...
Sending signal to pid = 4841
Service 1 is exiting...

```

```
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./client
Client Process started
PID of this client process is: 4839
Enter service type:
1-> LCM of 4 integers
2-> GCD of 2 integers
3-> LCM and GCD of 2 integers
2
Enter 2 integers: 2 4
GCD of 2 and 4 is: 2
Client Process ended
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./client
Client Process started
PID of this client process is: 4841
Enter service type:
1-> LCM of 4 integers
2-> GCD of 2 integers
3-> LCM and GCD of 2 integers
1
Enter 4 integers: 2 3 4 5
LCM of 2, 3, 4 and 5 is: 60
Client Process ended
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ 
```

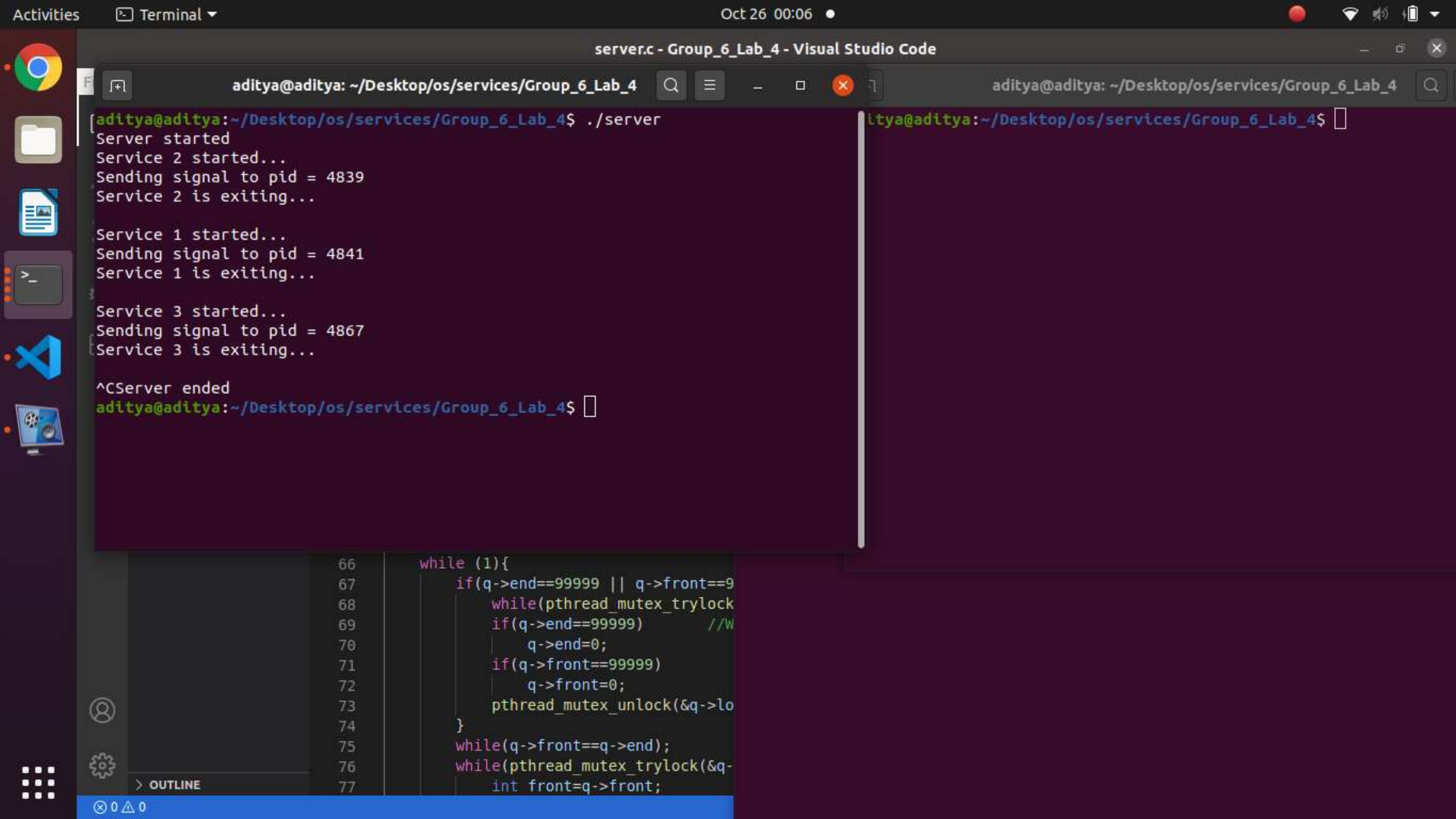
```
    while (1){
        if(q->end==99999 || q->front==9
            while(pthread_mutex_trylock
            if(q->end==99999)        //W
                q->end=0;
            if(q->front==99999)
                q->front=0;
            pthread_mutex_unlock(&q->lo
        }
        while(q->front==q->end);
        while(pthread_mutex_trylock(&q-
            int front=q->front;
```


Activities
Terminal
Oct 26 00:06
server.c - Group_6_Lab_4 - Visual Studio Code
aditya@aditya: ~/Desktop/os/services/Group_6_Lab_4
aditya@aditya: ~/Desktop/os/services/Group_6_Lab_4
OUTLINE
⊗ 0 ⚠ 0

```
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./server
Server started
Service 2 started...
Sending signal to pid = 4839
Service 2 is exiting...

Service 1 started...
Sending signal to pid = 4841
Service 1 is exiting...

Service 3 started...
Sending signal to pid = 4867
Service 3 is exiting...
```

```
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./client
Client Process started
PID of this client process is: 4867
Enter service type:
1-> LCM of 4 integers
2-> GCD of 2 integers
3-> LCM and GCD of 2 integers
3
Enter 2 integers: 4 6
LCM of 4 and 6 is: 12
GCD of 4 and 6 is: 2
Client Process ended
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$
```

```
    while (1){
        if(q->end==99999 || q->front==9
            while(pthread_mutex_trylock
            if(q->end==99999)        //W
                q->end=0;
            if(q->front==99999)
                q->front=0;
            pthread_mutex_unlock(&q->lo
        }
        while(q->front==q->end);
        while(pthread_mutex_trylock(&q-
            int front=q->front;
```

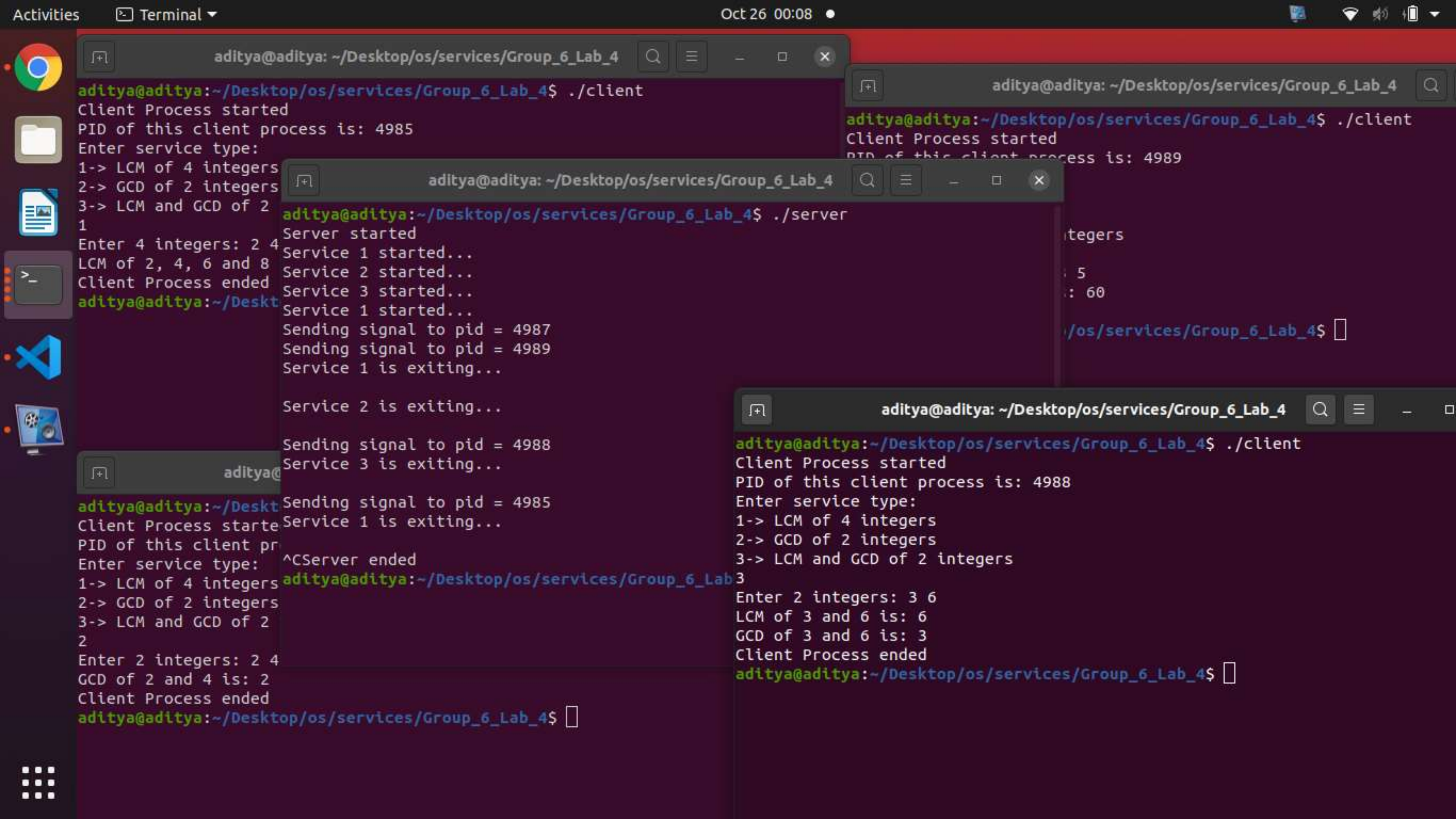


Activities
Terminal
Oct 26 00:06
server.c - Group_6_Lab_4 - Visual Studio Code
aditya@aditya: ~/Desktop/os/services/Group_6_Lab_4
aditya@aditya: ~/Desktop/os/services/Group_6_Lab_4
OUTLINE
⊗ 0 ⚠ 0

```
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./server
Server started
Service 2 started...
Sending signal to pid = 4839
Service 2 is exiting...

Service 1 started...
Sending signal to pid = 4841
Service 1 is exiting...

Service 3 started...
Sending signal to pid = 4867
Service 3 is exiting...

^CServer ended
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ 
```

```
itya@aditya:~/Desktop/os/services/Group_6_Lab_4$ 
```

```
    while (1){
        if(q->end==99999 || q->front==9
            while(pthread_mutex_trylock
            if(q->end==99999)        //W
                q->end=0;
            if(q->front==99999)
                q->front=0;
            pthread_mutex_unlock(&q->lo
        }
        while(q->front==q->end);
        while(pthread_mutex_trylock(&q-
            int front=q->front;
```



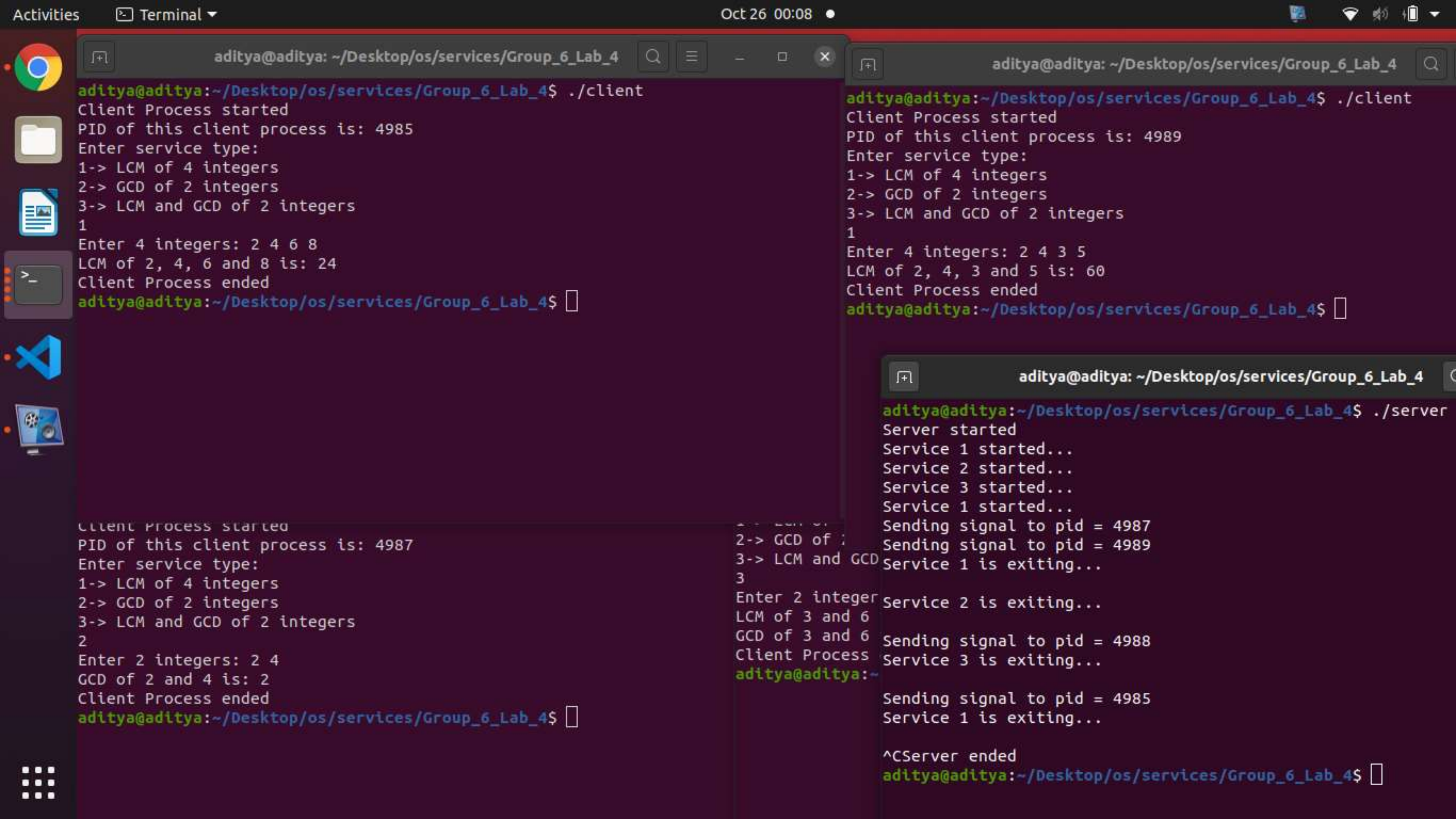
```
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./client
Client Process started
PID of this client process is: 4985
Enter service type:
1-> LCM of 4 integers
2-> GCD of 2 integers
3-> LCM and GCD of 2
1
Enter 4 integers: 2 4
LCM of 2, 4, 6 and 8
Client Process ended
aditya@aditya:~/Deskt
```

```
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./client
Client Process started
PID of this client process is: 4989
                               tegers

                               5
                               : 60
                               /os/services/Group_6_Lab_4$
```

```
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./server
Server started
Service 1 started...
Service 2 started...
Service 3 started...
Service 1 started...
Sending signal to pid = 4987
Sending signal to pid = 4989
Service 1 is exiting...

Service 2 is exiting...

Sending signal to pid = 4988
Service 3 is exiting...

Sending signal to pid = 4985
Service 1 is exiting...

^CServer ended
aditya@aditya:~/Desktop/os/services/Group_6_Lab
```

```
aditya@aditya:~/Deskt
Client Process starte
PID of this client pr
Enter service type:
1-> LCM of 4 integers
2-> GCD of 2 integers
3-> LCM and GCD of 2
2
Enter 2 integers: 2 4
GCD of 2 and 4 is: 2
Client Process ended
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$
```

```
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./client
Client Process started
PID of this client process is: 4988
Enter service type:
1-> LCM of 4 integers
2-> GCD of 2 integers
3-> LCM and GCD of 2 integers
3
Enter 2 integers: 3 6
LCM of 3 and 6 is: 6
GCD of 3 and 6 is: 3
Client Process ended
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$
```


Activities
Terminal
Oct 26 00:08
aditya@aditya: ~/Desktop/os/services/Group_6_Lab_4
aditya@aditya: ~/Desktop/os/services/Group_6_Lab_4
aditya@aditya: ~/Desktop/os/services/Group_6_Lab_4

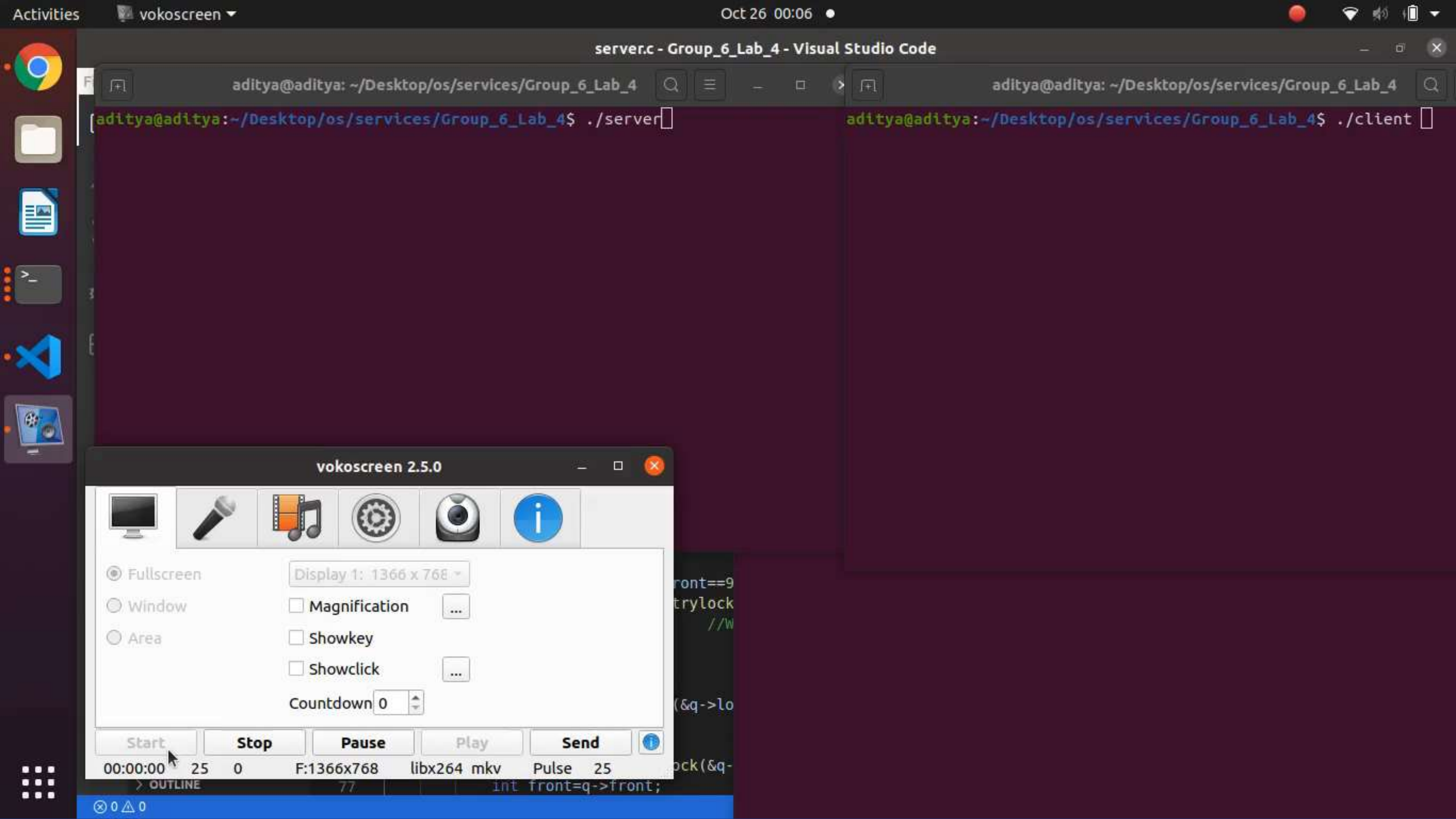
```
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./client
Client Process started
PID of this client process is: 4985
Enter service type:
1-> LCM of 4 integers
2-> GCD of 2 integers
3-> LCM and GCD of 2 integers
1
Enter 4 integers: 2 4 6 8
LCM of 2, 4, 6 and 8 is: 24
Client Process ended
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$
```

```
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./client
Client Process started
PID of this client process is: 4989
Enter service type:
1-> LCM of 4 integers
2-> GCD of 2 integers
3-> LCM and GCD of 2 integers
1
Enter 4 integers: 2 4 3 5
LCM of 2, 4, 3 and 5 is: 60
Client Process ended
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$
```

```
Client Process started
PID of this client process is: 4987
Enter service type:
1-> LCM of 4 integers
2-> GCD of 2 integers
3-> LCM and GCD of 2 integers
2
Enter 2 integers: 2 4
GCD of 2 and 4 is: 2
Client Process ended
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$
```

```
2-> GCD of
3-> LCM and GCD
3
Enter 2 integer
LCM of 3 and 6
GCD of 3 and 6
Client Process
aditya@aditya:~
```

```
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./server
Server started
Service 1 started...
Service 2 started...
Service 3 started...
Service 1 started...
Sending signal to pid = 4987
Sending signal to pid = 4989
Service 1 is exiting...

Service 2 is exiting...

Sending signal to pid = 4988
Service 3 is exiting...

Sending signal to pid = 4985
Service 1 is exiting...

^CServer ended
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$
```



server.c - Group_6_Lab_4 - Visual Studio Code

aditya@aditya: ~/Desktop/os/services/Group_6_Lab_4

```
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./server
```

aditya@aditya: ~/Desktop/os/services/Group_6_Lab_4

```
aditya@aditya:~/Desktop/os/services/Group_6_Lab_4$ ./client
```

```
ront==9
trylock
    //W
(&q->lo
ock(&q-
    int front=q->front;
```

vokoscreen 2.5.0

● Fullscreen
○ Window
○ Area

Display 1: 1366 x 768

□ Magnification
□ Showkey
□ Showclick
Countdown 0

00:00:00 25 0 F:1366x768 libx264 mkv Pulse 25

THANK YOU