OS_Lab6

K. Ram Mohan

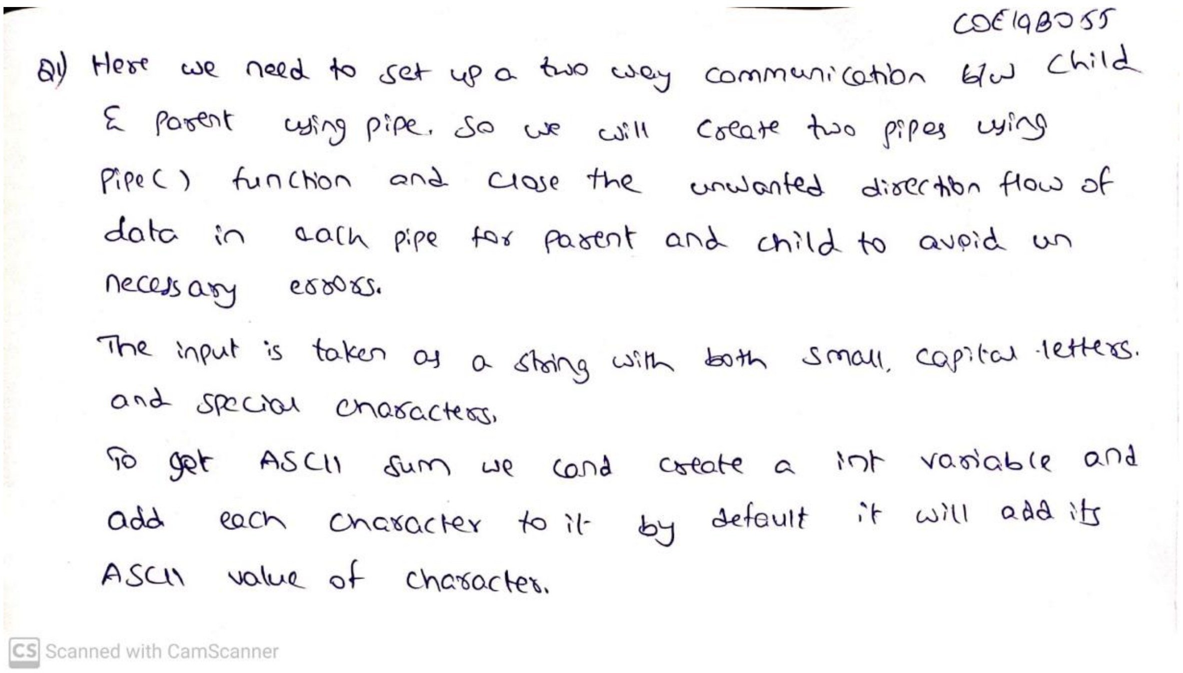
COE19B055

Q1)

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

int main()
{
	int pipefds1[2], pipefds2[2];
	int returnstatus1, returnstatus2;
	int pid;
	char writemessage[50], readmessage[50], buffer[50];

	returnstatus1 = pipe(pipefds1);
	if(returnstatus1 == -1)
	{
		printf("Failed to create pipe1\n");
		return 0;
	}

	returnstatus2 = pipe(pipefds2);
	if(returnstatus2 == -1)
```

```
{
        printf("Failed to create pipe2\n");
        return 0;
}

pid = fork();
if(pid == 0)
{
        close(pipefds1[1]);
        close(pipefds2[0]);

        read(pipefds1[0], readmessage, sizeof(readmessage));
        strcpy(buffer, readmessage);

        int i, sum=0;
        //-1 is given to exclude the line feed character
        for(i=0; i<strlen(buffer)-1; i++)
        {
                sum = sum + buffer[i];
        }

        sprintf(writemessage, "%d", sum);

        write(pipefds2[1], writemessage, sizeof(writemessage));
}
else
{
        close(pipefds1[0]);
        close(pipefds2[1]);
```

```
                printf("In parent process: Enter the string: ");
                fgets(buffer, 50, stdin);
                strcpy(writemessage, buffer);
                write(pipefds1[1], writemessage, sizeof(writemessage));

                read(pipefds2[0], readmessage, sizeof(readmessage));

                printf("In parent process: Sum read from pipe is: %s\n", readmessage);
        }
}
```



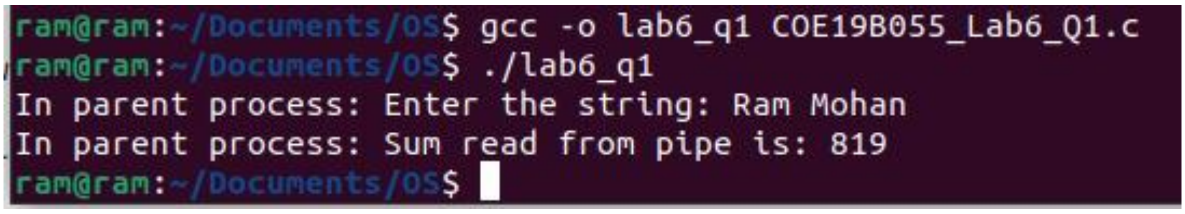
COE19B055

Q1) Here we need to set up a two way communication b/w child & parent using pipe. So we will create two pipes using pipe( ) function and close the unwanted direction flow of data in each pipe for parent and child to avoid un necessary errors.

The input is taken as a string with both small, capital letters. and special characters.

To get ASCII sum we cand create a int variable and add each character to it by default it will add its ASCII value of character.

Output:



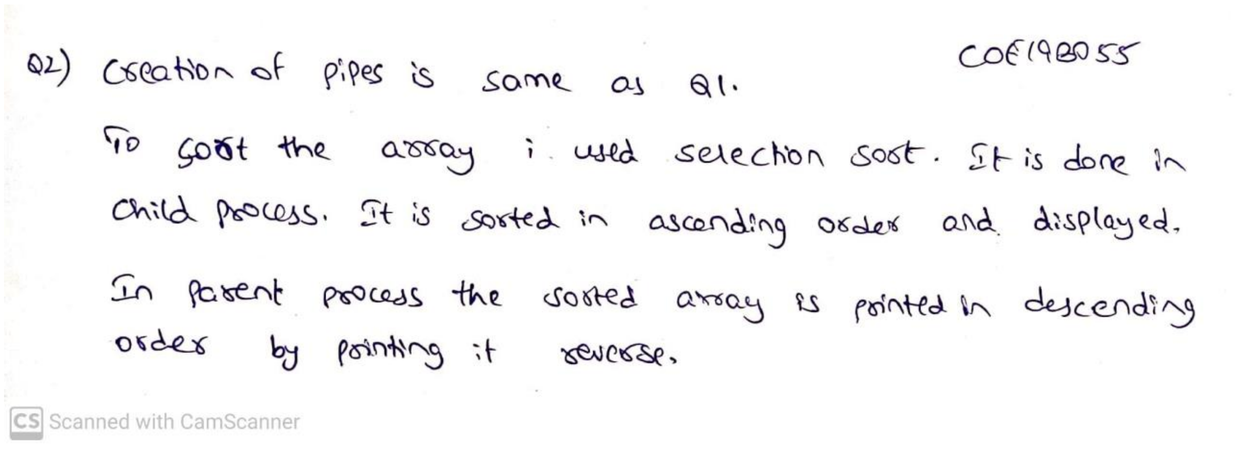
```
ram@ram:~/Documents/OS$ gcc -o lab6_q1 COE19B055_Lab6_Q1.c
ram@ram:~/Documents/OS$ ./lab6_q1
In parent process: Enter the string: Ram Mohan
In parent process: Sum read from pipe is: 819
ram@ram:~/Documents/OS$ 
```

Q2)

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/wait.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<string.h>


void swap(int *p, int *q)

{

	int temp;

	temp = *p;

	*p = *q;

	*q = temp;

}


int main()

{

	int pipefds1[2],pipefds2[2];

	int returnstatus1, returnstatus2;

	int pid;


	int writemessage[10], readmessage[10];


	returnstatus1 = pipe(pipefds1);

	if(returnstatus1 < 0)
```

```
{
        printf("Failed to create pipe1\n");
        return 0;
}

returnstatus2 = pipe(pipefds2);
if(returnstatus2 < 0)
{
        printf("Failed to create pipe2\n");
        return 0;
}

pid = fork();
if(pid == 0)
{
        close(pipefds1[1]);
        close(pipefds2[0]);

        read(pipefds1[0], readmessage, sizeof(readmessage));
        int i, j;

        printf("In child process: Read from pipe:  ");
        for(i=0; i<10; i++)
        {
                printf("%d ", readmessage[i]);
        }
        printf("\n");

        for(i=9, j=0; i>-1; i--, j++)
        {
                writemessage[j] = readmessage[i];
        }

        write(pipefds2[1], writemessage, sizeof(writemessage));
}
else
{
        close(pipefds1[0]);
        close(pipefds2[1]);

        printf("In parent process: Enter 10 number\n");
        int i, j, min;

        for(i=0; i<10; i++)
        {
                printf("Enter num %d: ", i+1);
                scanf("%d", &writemessage[i]);
        }

        for(i=0; i<9; i++)
        {
                min = i;
                for(j=i; j<10; j++)
                {
                        if(writemessage[j] < writemessage[min])
                        {
                                min = j;
                        }
```

```
            }
            swap(&writemessage[i], &writemessage[min]);
        }
        write(pipefds1[1], writemessage, sizeof(writemessage));


        read(pipefds2[0], readmessage, sizeof(readmessage));


        printf("In parent process: Read from pipe: ");
        for(i=0; i<10; i++)
        {
            printf("%d ", readmessage[i]);
        }
        printf("\n");
    }
}
```

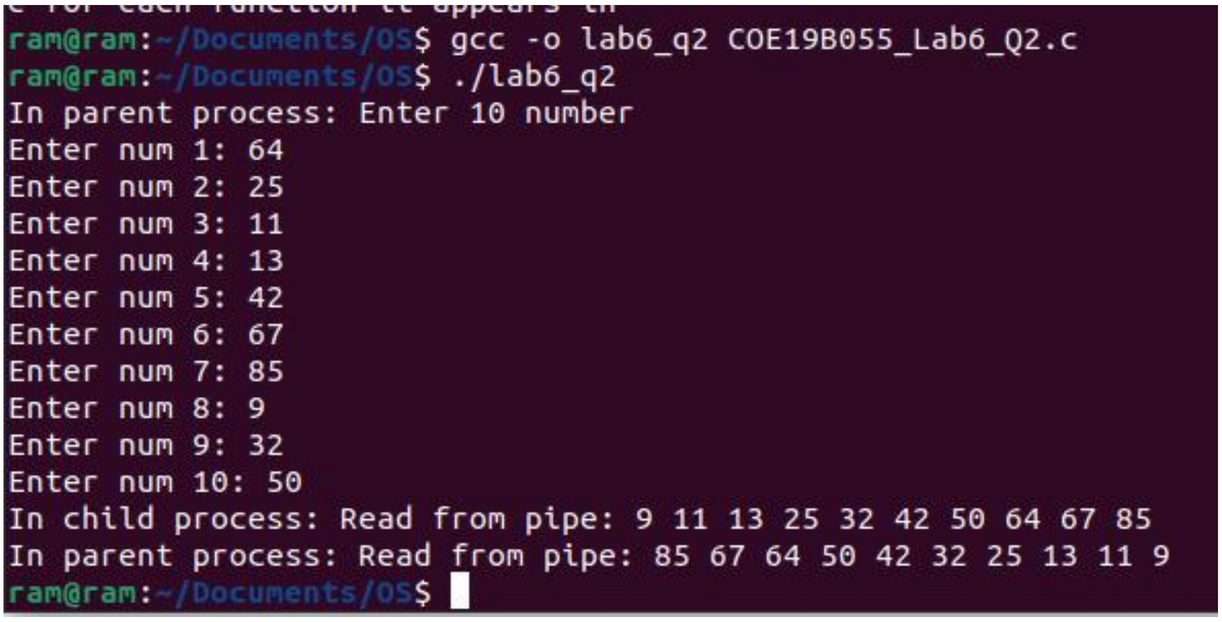


Q2) Creation of pipes is same as Q1. COE19B055

To sort the array i used selection sort. It is done in Child process. It is sorted in ascending order and displayed.

In parent process the sorted array is printed in descending order by printing it reverse.

Output:



```
e for each function it appears in
ram@ram:~/Documents/OS$ gcc -o lab6_q2 COE19B055_Lab6_Q2.c
ram@ram:~/Documents/OS$ ./lab6_q2
In parent process: Enter 10 number
Enter num 1: 64
Enter num 2: 25
Enter num 3: 11
Enter num 4: 13
Enter num 5: 42
Enter num 6: 67
Enter num 7: 85
Enter num 8: 9
Enter num 9: 32
Enter num 10: 50
In child process: Read from pipe: 9 11 13 25 32 42 50 64 67 85
In parent process: Read from pipe: 85 67 64 50 42 32 25 13 11 9
ram@ram:~/Documents/OS$
```

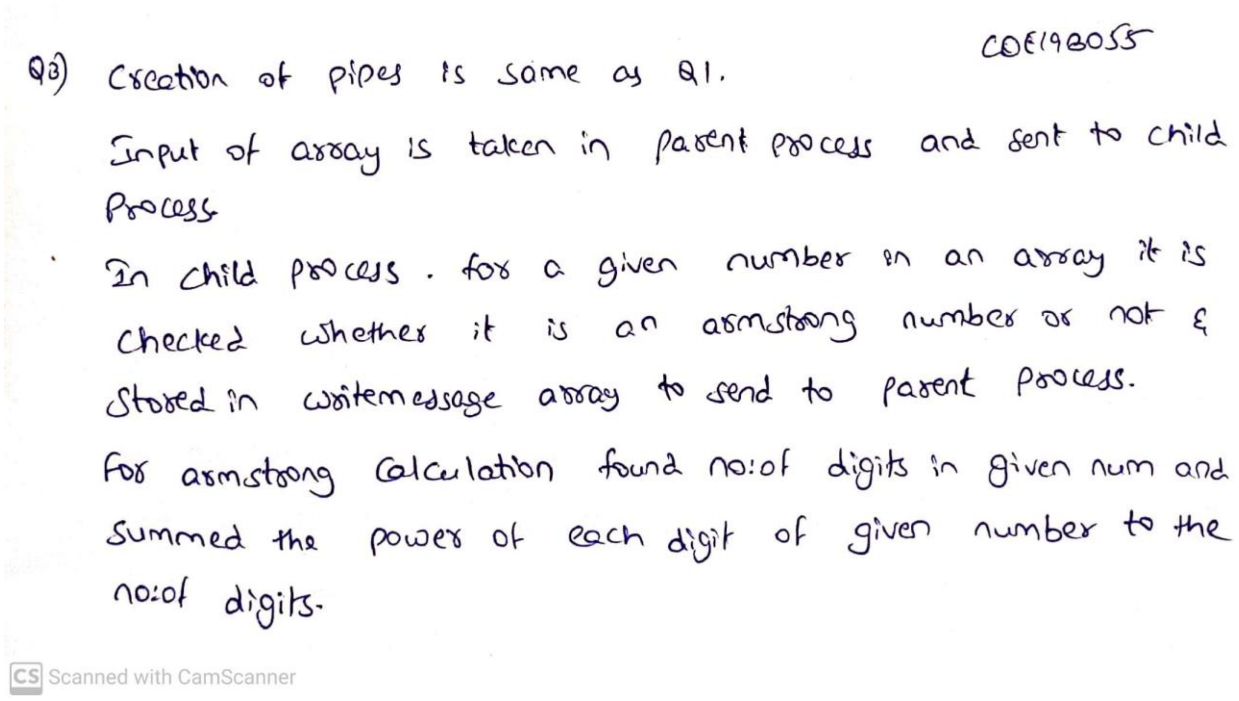
Q3)

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/wait.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<string.h>

#include<math.h>


int armstrong(int num)

{

        int digits, sum, temp, rem;

        digits = 0;

        sum = 0;

        temp = num;

        while(temp>0)

        {

                temp = temp/10;

                digits++;

        }


        temp = num;

        while(temp>0)

        {

                rem = temp%10;

                sum = sum + pow(rem, digits);

                temp = temp/10;
```

```
	}

	if(sum == num){
		return  1;
	}else{
		return 0;
	}
}

int main()
{
	int pipefds1[2],pipefds2[2];
	int returnstatus1, returnstatus2;
	int pid;

	int writemessage[10], readmessage[10];

	returnstatus1 = pipe(pipefds1);
	if(returnstatus1 < 0)
	{
		printf("Failed to create pipe1\n");
		return 0;
	}

	returnstatus2 = pipe(pipefds2);
	if(returnstatus2 < 0)
	{
		printf("Failed to create pipe2\n");
		return 0;
```

```
}

pid = fork();
if(pid == 0)
{
	close(pipefds1[1]);
	close(pipefds2[0]);

	read(pipefds1[0], readmessage, sizeof(readmessage));
	int i, j, digits, temp, rem, sum;

	for(i=0; i<10; i++)
	{
		writemessage[i] = armstrong(readmessage[i]);
	}
	write(pipefds2[1], writemessage, sizeof(writemessage));
}
else
{
	close(pipefds1[0]);
	close(pipefds2[1]);

	printf("In parent process: Enter 10 number\n");
	int i;

	for(i=0; i<10; i++)
	{
		printf("Enter num %d: ", i+1);
		scanf("%d", &writemessage[i]);
```

```
                }

                write(pipefds1[1], writemessage, sizeof(writemessage));

                read(pipefds2[0], readmessage, sizeof(readmessage));

                printf("In parent process: Result = 1 if armstrong 0 if not\n");
                for(i=0; i<10; i++)
                {
                        printf("%d - %d \n", writemessage[i], readmessage[i]);
                }
        }
}
```



COE19B055

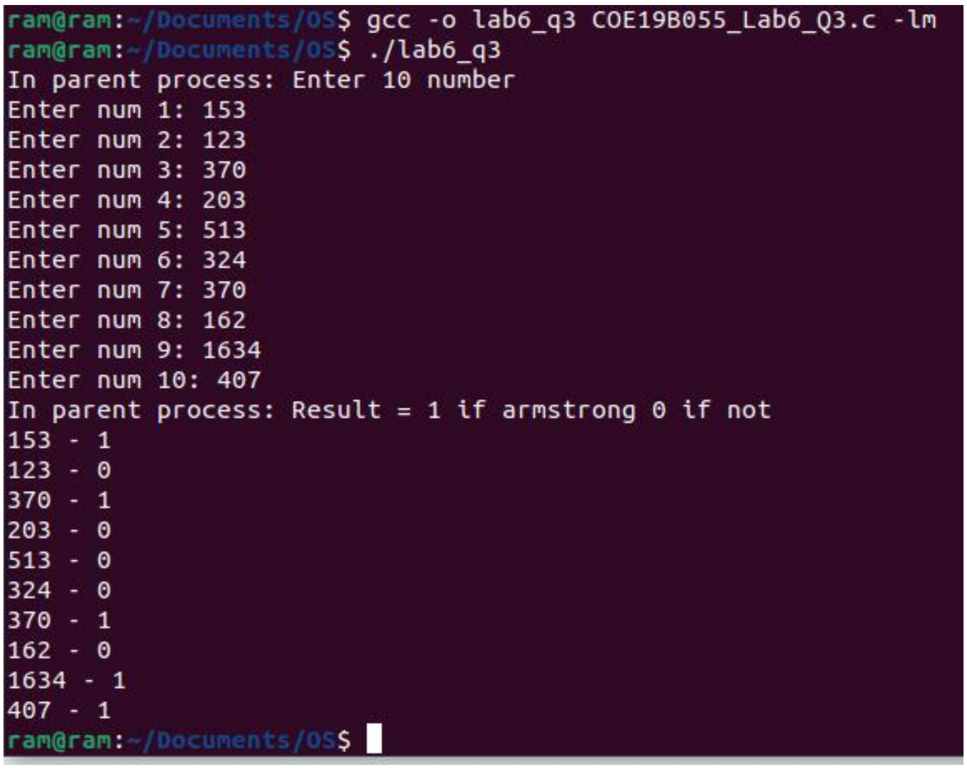
Q3) Creation of pipes is same as Q1.

Input of array is taken in parent process and sent to child Process

In child process . for a given number in an array it is Checked whether it is an armstrong number or not & Stored in writemessage array to send to parent process.

For armstrong Calculation found no:of digits in given num and summed the power of each digit of given number to the no:of digits.

Output:



```
ram@ram:~/Documents/OS$ gcc -o lab6_q3 COE19B055_Lab6_Q3.c -lm
ram@ram:~/Documents/OS$ ./lab6_q3
In parent process: Enter 10 number
Enter num 1: 153
Enter num 2: 123
Enter num 3: 370
Enter num 4: 203
Enter num 5: 513
Enter num 6: 324
Enter num 7: 370
Enter num 8: 162
Enter num 9: 1634
Enter num 10: 407
In parent process: Result = 1 if armstrong 0 if not
153 - 1
123 - 0
370 - 1
203 - 0
513 - 0
324 - 0
370 - 1
162 - 0
1634 - 1
407 - 1
ram@ram:~/Documents/OS$ 
```

Q4)

Process1:

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/shm.h>

#include<string.h>


#define SIZE 100


struct memory

{

	char data[SIZE];

	int status;

	//status=0 process read data, status=2 process wrote something

	int pal;

};


int main()

{

	int shmid;

	struct memory *shm;

	char buffer[100];


	key_t key = ftok("shmfile", 65);

	shmid = shmget(key, SIZE, IPC_CREAT|0666);
```

```
if(shmid<0)
{
	printf("Creation of shared memory failed\n");
	return 0;
}

shm = shmat(shmid, NULL, 0);

printf("Enter end-end to end chat\n");

while(1)
{
	printf("Enter message: ");
	fgets(buffer, 100, stdin);

	strcpy(shm->data, buffer);
	shm->status = 1;

	if(strncmp(shm->data, "end-end", 7)==0)
	{
		break;
	}

	while(shm->status==1)
		sleep(1);

	while(shm->status==0)
		sleep(1);
		if(shm->pal == 1){
			printf("It is a palindrome\n");
		}else{
			printf("It is not a palindrome\n");
		}
		shm->status=0;
	}
	shmdt(shm);
	shmctl(shmid, IPC_RMID, NULL);
}
```

Process2:

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<string.h>

#define SIZE 100

struct memory
{
	char data[SIZE];
	int status;
	//status=0 process read data, status=2 process wrote something
	int pal;
};

int main()
{
	int shmid;
	struct memory *shm;
	char buffer[100], buffer1[100];

	key_t key = ftok("shmfile", 65);
	shmid = shmget(key, SIZE, IPC_CREAT|0666);

	if(shmid<0)
	{
		printf("Creation of shared memory failed\n");
		return 0;
	}

	shm = shmat(shmid, NULL, 0);

	printf("Enter end-end to end chat\n");
	while(1)
	{
		while(shm->status!=1)
			sleep(1);

		printf("Read from shared memory: %s", shm->data);
		shm->status = 0;
```

```
if(strncmp(shm->data, "end-end", 7)==0)
{
        break;
}

int i, len, temp = 1;

len = strlen(shm->data) - 1;
strncpy(buffer, shm->data, len);

for(i=0; i<(len)/2; i++)
{
        if(strncmp(&buffer[i], &buffer[len-1-i], 1)!=0)
        {
                temp = 0;
        }
}

if(temp==1){
        printf("It is a palindrome :%d\n", temp);
        shm->pal = 1;
}else{
        printf("It is not a palindrome :%d\n", temp);
        shm->pal = 0;
}

//it is just to prevent to exit the code
printf("Enter 1 if given is palindrome or 0(or else press enter to continue): ");
fgets(buffer, 100, stdin);
```

```
                shm->status=1;


                while(shm->status==1)
                        sleep(1);
        }
        shmdt(shm);
        shmctl(shmid, IPC_RMID, NULL);
}
```

COE19B055

Q4) IPC using Shared memory:-

Since SM will be in userspace we donot need any System calls to read or write. We just need to create a SM.

Shmget - we can get unique id for SM

Shmat - we can get the SM pointer
↓
It returns a pointer through which we can access it.

Shmdt - to detach SM

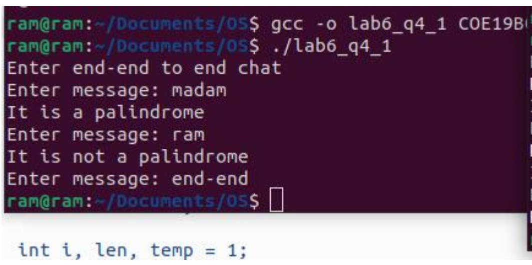
Shmctl - to destroy SM

To find whether it is palindrome or not compared the first & last and similarly second & last second & soon.

Input is taken from process 1 & palindrome check is done in process 2. output is displayed in process 1.

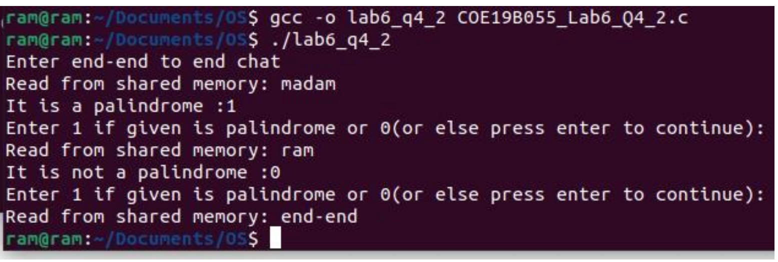
Here SM is a struct of data, status, Pal. Status is to check whether other process read or using SM.

Output:



```
ram@ram:~/Documents/OS$ gcc -o lab6_q4_1 COE19B
ram@ram:~/Documents/OS$ ./lab6_q4_1
Enter end-end to end chat
Enter message: madam
It is a palindrome
Enter message: ram
It is not a palindrome
Enter message: end-end
ram@ram:~/Documents/OS$
```

```
int i, len, temp = 1;
```



```
ram@ram:~/Documents/OS$ gcc -o lab6_q4_2 COE19B055_Lab6_Q4_2.c
ram@ram:~/Documents/OS$ ./lab6_q4_2
Enter end-end to end chat
Read from shared memory: madam
It is a palindrome :1
Enter 1 if given is palindrome or 0(or else press enter to continue):
Read from shared memory: ram
It is not a palindrome :0
Enter 1 if given is palindrome or 0(or else press enter to continue):
Read from shared memory: end-end
ram@ram:~/Documents/OS$
```