Ayush Goyal
190905522 CSE D 62

Operating Systems Week 7: Lab 6: IPC 2- Message Queue, Shared Memory

1. Process A wants to send a number to Process B. Once received, process B has to check whether the number is palindrome or not. Write a C program to implement this interprocess communication using message queue.

Code:

Producer code:

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

struct my_msg_st{
	long int my_msg_type;
	int num;
};

int main(){
	int running = 1;
	int msgid;
	struct my_msg_st some_data;
	long int msg_to_receive = 0;
	msgid = msgget((key_t)1234, 0666 | IPC_CREAT);
	if(msgid == -1){
		perror("msgget failed");
		exit(EXIT_FAILURE);
	}
	while(running){
		printf("Enter a number : ");
		int n;
		scanf("%d", &n);
		some_data.my_msg_type = 1;
		some_data.num = n;
		if(msgsnd(msgid, (void *)&some_data, sizeof(msgid), 0) == -1){
			perror("msgsnd failed");
			exit(EXIT_FAILURE);
		}
		if(n == -1){
			running = 0;
		}
	}
```

```
	exit(EXIT_SUCCESS);
}
```

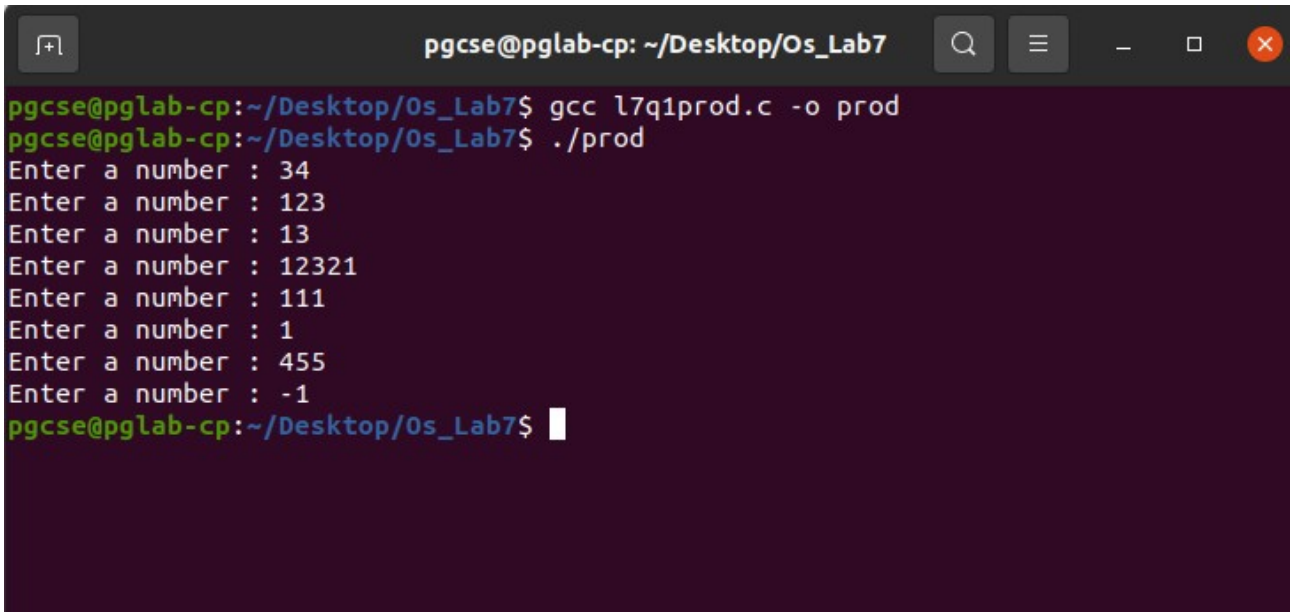
Consumer Code:

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

struct my_msg_st{
	long int my_msg_type;
	int num;
};

int ispalin(int t){
	int arr[100], n = 0;
	while(t >0){
		int dig = t%10;
		t/=10;
		arr[n++] = dig;
	}
	for(int i = 0; i<n/2; i++){
		if(arr[i] != arr[n-i-1]){
			return 0;
		}
	}
	return 1;
}

int main(){
	int running = 1;
	int msgid;
	struct my_msg_st some_data;
	long int msg_to_receive = 0;
	msgid = msgget((key_t)1234, 0666 | IPC_CREAT);
	if(msgid == -1){
		perror("msgget failed");
		exit(EXIT_FAILURE);
	}
	while (running){
		if(msgrcv(msgid, (void *)&some_data, sizeof(msgid), msg_to_receive, 0) == -1){
			perror("msgrcv failed");
			exit(EXIT_FAILURE);
		}
		if(some_data.num == -1){
			running = 0;
		}
```

```
		else if(ispalin(some_data.num)){
			printf("%d is a palindrome\n", some_data.num);
		}
		else{
			printf("%d is not a palindrome\n", some_data.num);
		}
	}
	if(msgctl(msgid, IPC_RMID, 0) == -1){
		perror("msgctl failed");
		exit(EXIT_FAILURE);
	}
	exit(EXIT_SUCCESS);
}
```

Output:

Producer Terminal:



Consumer Terminal:

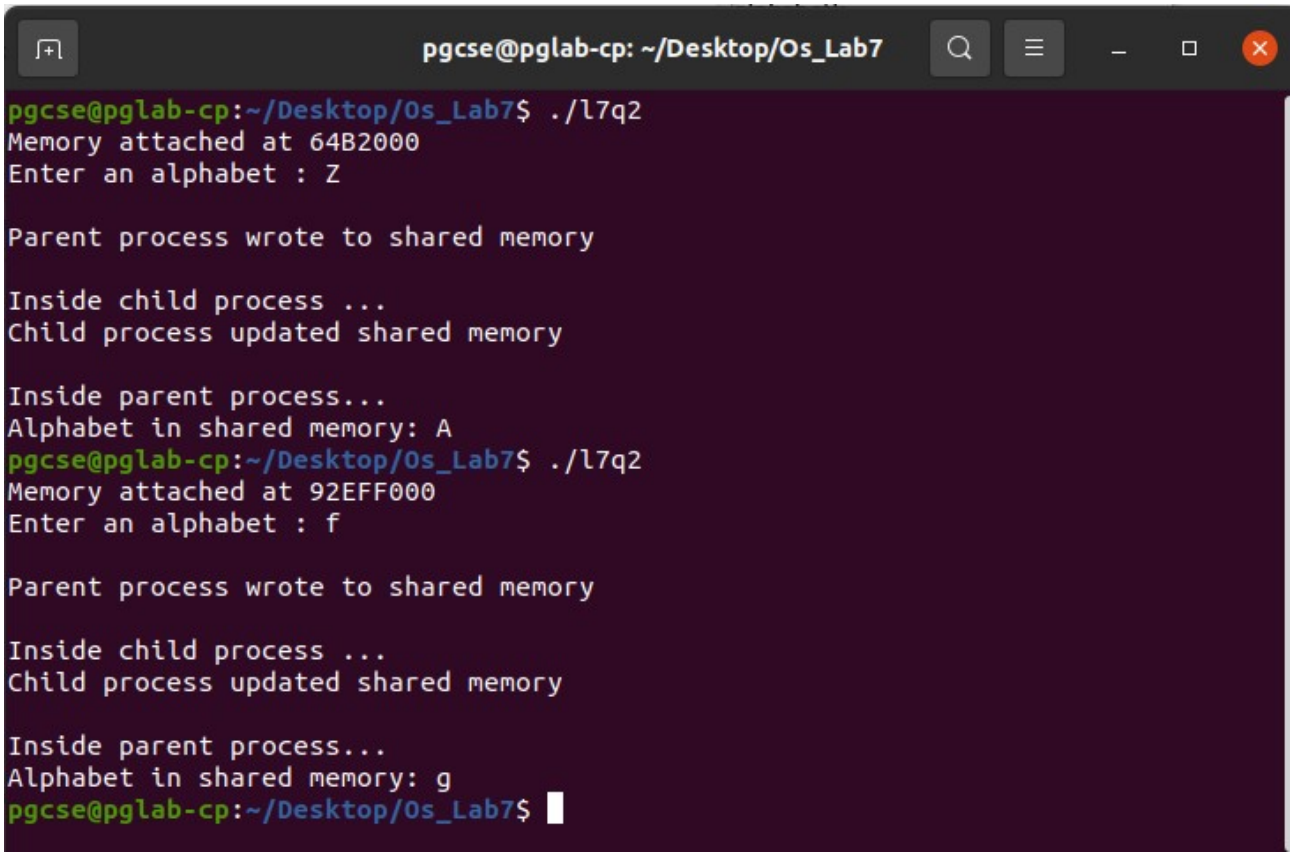
2. Implement a parent process, which sends an English alphabet to a child process using shared memory. The child process responds with the next English Alphbet to the parent. The parent displays the reply from the child.

Code:

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

struct shared_use_st{
	char alphabet;
};

int main(){
	void *shared_memory = (void *)0;
	struct shared_use_st *shared_stuff;
	char alphatosend;
	pid_t pid;
	int status;
	int smid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666 | IPC_CREAT);
	if(smid == -1){
		printf("shmget failed!\n");
		exit(0);
	}
	shared_memory = shmat(smid, (void *)0, 0);
	if(shared_memory == (void *)-1){
		printf("shmat failed!\n");
		exit(0);
	}
	printf("Memory attached at %X\n", (int)shared_memory);
	shared_stuff = (struct shard_use_st *)shared_memory;
	printf("Enter an alphabet : ");
	scanf("%c", &alphatosend);
	shared_stuff->alphabet = alphatosend;
	printf("\nParent process wrote to shared memory\n");
	pid = fork();
	if(pid == 0){
		printf("\nInside child process ...\n");
		char alphabet = shared_stuff->alphabet;
		if(alphabet == 'z'){
			alphabet = 'a';
		}
		else if (alphabet == 'Z'){
			alphabet = 'A';
		}
		else
```

```
                alphabet = (char)((int)alphabet + 1);
            shared_stuff->alphabet = alphabet;
            printf("Child process updated shared memory\n");
        }
        else{
            wait(&status);
            printf("\nInside parent process...\n");
            printf("Alphabet in shared memory: %c\n", shared_stuff->alphabet);
        }
}
```

Output:



THE END