## Term work 1: Implementing IPC using Pipes and message queues.

## TW-1 (Message Queue)

### Writer

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>

#include <stdlib.h>

#define MAX 50

struct msg\_buffer {

long mesg\_type;

char mesg\_text[100];

}message;

int main() {

key\_t key;

int msgid;

key = ftok("progfile", 65);

msgid = msgget(key, 0666 | IPC\_CREAT);

message.mesg\_type = 1;

printf("Write data: \n");

fgets(message.mesg\_text, MAX, stdin);

msgsnd(msgid, &message, sizeof(message), 0);

printf("Data sent is : %s\n", message.mesg\_text);

return 0;

}

### 

### Reader

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>

#include <stdlib.h>

#define MAX 50

struct msg\_buffer {

long mesg\_type;

char mesg\_text[100];

}message;

int main() {

key\_t key;

int msgid;

key = ftok("progfile", 65);

msgid = msgget(key, 0666 | IPC\_CREAT);

msgrcv(msgid, &message, sizeof(message), 1, 0);

printf("Data read is: %s\n", message.mesg\_text);

msgctl(msgid, IPC\_RMID, NULL);

return 0;

}

## Output

### Writer

lab2@lab2-virtual-machine:~ NP-Lab/TW-1$ gcc TW-1\_writer.c

lab2@lab2-virtual-machine:~ NP-Lab/TW-1$ ./a.out

Write data: Hello World

Data sent is : Hello World

### Reader

lab2@lab2-virtual-machine:~ TW-1$ gcc TW-1\_reader.c

lab2@lab2-virtual-machine:~ TW-1$ ./a.out

Data read is: Hello World

## 

## TW-1 (Pipe)

#include <unistd.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/wait.h>

int main() {

int fd[2], n;

char buffer[100];

pid\_t p;

pipe(fd);

p = fork();

if (p > 0) {

printf("Parent process pid: %d\n", getppid());

printf("Child process pid: %d\n", p);

printf("Passing value child\n");

write(fd[1], "Hello World!\n", 13);

}

else {

printf("Child process pid: %d\n", getpid());

printf("Parent process pid: %d\n", getppid());

n = read(fd[0], buffer, 100);

printf("Data received by child process: \n");

write(1, buffer, n);

}

return 0;

}

## Output

lab2@lab2-virtual- TW-1/pipe$ gcc TW-1.c

lab2@lab2-virtual- TW-1/pipe$ ./a.out

Parent process pid: **7437**

Child process pid: **7618**

**Passing value child**

lab2@lab2-virtual- TW-1/pipe$ Child process pid: **7618**

Parent process pid: **1511**

Data received by child process:

**Hello World!**