Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра кібербезпеки та програмного забезпечення

**Звіт**

Лабораторная работа № 5.

Дисципліна: Операційні системи

Виконав :

студент академічної групи КІ-15

Аннаєв А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перевірив :

Викладач

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кропивницкий- 2018

Лабораторная работа № 5.

Тема: Средства межпроцессного взаимодействия

Задания к лабораторной работе:

Написать программу, которая порождает дочерний процесс, и общается с ним через средства взаимодействия согласно варианту (табл А.).

Передачу и получение информации каждым из процессов сопровождать выводом на экран информации типа "процесс такой-то передал / получил-информации".

|  |  |  |  |
| --- | --- | --- | --- |
| **Вар.** | **Средство взаимодействия** | **Вар.** | **Средство взаимодействия** |
| 3 | Очередь сообщений | 13 | Очередь сообщений |

#include <sys/types.h>

#include <unistd.h>

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <string.h>

// structure for message queue

struct mesg\_buffer {

long mesg\_type;

char mesg\_text[100];

} message;

void ChildProcess(void); /\* child process prototype \*/

void ParentProcess(void); /\* parent process prototype \*/

int main(void)

{

pid\_t pid;

pid = fork();

if (pid == 0)

{

ChildProcess();

}

else

{

ParentProcess();

}

return 0;

}

void ChildProcess(void)

{

printf("[writer] = [child]\n child process is start\n");

key\_t key;

int msgid;

// ftok to generate unique key

key = ftok("progfile", 65);

// msgget creates a message queue

// and returns identifier

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

message.mesg\_type = 1;

printf("Write Data : ");

strcpy(message.mesg\_text , "Annaev Arslan KI 15 06.10.2018") ;

// msgsnd to send message

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

// display the message

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

printf("child process is done\n");

}

void ParentProcess(void)

{

printf("[reader] = [parent]\n parent is start\n");

key\_t key;

int msgid;

// ftok to generate unique key

key = ftok("progfile", 65);

// msgget creates a message queue

// and returns identifier

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

// msgrcv to receive message

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

// display the message

printf("Data Received is : %s \n",

message.mesg\_text);

// to destroy the message queue

msgctl(msgid, IPC\_RMID, NULL);

printf("parent is done\n");

}

