Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

“Inițiere în Cloud Computing și Sisteme de Operare Distribuite”

Лабораторная работа 7  
“Межпроцессное взаимодействие Linux”

Проверил: Препелица Аурелия

Выполнил: Чобану Артём

Кишинев 2021

# Требования к лабораторной работе:

Разработайте программу, которая должна создать 3 дочерних процесса. Каждый процесс генерирует по одному произвольному целому числу. Процессы должны сообща найти максимальное значение среди генерируемых случайных чисел, посылая и принимая сообщения через неименованные каналы, по кольцу. Затем дочерние процессы, отправив родительскому процессу сообщение через неименованные каналы -- N – номер процесса, завершают работу с выводом на консоль сообщения вида: N, случайное число, pid, ppid. Родительский процесс читает эти сообщения (N – номера дочерних процессов), выводит их на стандартный вывод, ожидает завершения сыновей, выводит на консоль максимальное значение, после чего завершается.

# Код программы::

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <time.h>

int childProcessAmount = 3;

int main(int argc, char \*\*argv)

{

    int processPIDs[childProcessAmount];

    int fd[childProcessAmount][2];

    int buffer[childProcessAmount];

    int parentPid = getpid();

    printf("Lab7!\n");

    printf("Parent pid: %i\n", parentPid);

    printf("Creating %i child processes\n", childProcessAmount);

    int max = -1;

    int currentPid;

    int currentProcessIndex;

    for(int i = 0; i < childProcessAmount; i++)

    {

        pipe(fd[i]);

        currentPid = getpid();

        int pid = fork();

        if(pid == 0)

        {

            srand(time(0) ^ getpid());

            close(fd[i][0]);

            int generatedNumber = rand() % 10;

            printf("Process with pid %i generates number %i\n", getpid());

            write(fd[i][1], &generatedNumber, sizeof(int));

            \_exit(EXIT\_SUCCESS);

        }

        else if (pid > 0)

        {

            int status;

            (void)waitpid(pid, &status, 0);

            close(fd[i][1]);

            int receivedMax;

            read(fd[i][0], &receivedMax, sizeof(int));

            if(max < receivedMax)

            {

                max = receivedMax;

            }

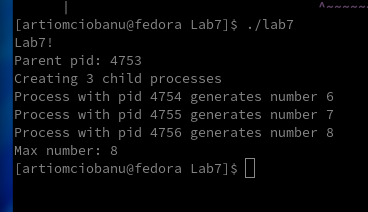
        }

    }

    printf("Max number: %i\n", max);

}

# Скриншоты работы программы:



# Библиография

<https://ru.wikipedia.org/wiki/Fork>

<http://www.c-cpp.ru/content/write-rtlwrite>

<https://www.cplusplus.com/reference/istream/istream/read/>

<https://www.geeksforgeeks.org/pipe-system-call/>

<http://www.c-cpp.ru/content/close-rtlclose>

<https://www.opennet.ru/man.shtml?topic=waitpid&category=2&russian=0>

<http://cppstudio.com/post/339/>