Министерство образования Республики Беларусь Учреждение образования

«Брестский государственный технический университет» Кафедра ИИТ

Лабораторная работа №6

по дисциплине «Операционные системы»

Тема«Средства межпроцессного взаимодействия»

Вариант 12

Выполнил: 220203

студент 2-го курса

группы ПО-11

Антонюк Н.А.

Проверила:

Давидюк Ю.И.

Брест 2023

**Цель работы:** ознакомиться с компилятором GCC; изучить создание, завершение и изменение пользовательского контекста процессов в операционной системе UNIX.

**Ход работы**

***Задание для выполнения:***

Ознакомиться с руководством, теоретическими сведениями и лекционным

материалом по использованию и функционированию средств взаимодействия.

Написать программу, которая порождает дочерний процесс, и общается с ним через

средства взаимодействия согласно варианту (табл.А), передавая и получая информацию

согласно варианту (табл.Б). Передачу и получение информации каждым из процессов

сопровождать выводом на экран информации типа "процесс такой-то передал/получил

такую-то информацию". Дочерние процессы начинают операции после получения

сигнала SIGUSR1 от родительского процесса.

***Вариант индивидуального задания:***

|  |  |  |  |
| --- | --- | --- | --- |
| *№* | *Средства взаимодействия* | *Задание* | |
| 12 | Разделяемая  память | Родитель передает 5 случайных чисел, потомок возвращает их  сумму и произведение. |

Текст программы:

#include <unistd.h>

#include <sys/types.h>

#include <stdio.h>

#include <fcntl.h>

#include <sys/mman.h>

#include <stdlib.h>

#include <signal.h>

#include <string.h>

#define NUMBERS\_AMOUNT 5

void hdl(int sig)

{

if(sig == SIGUSR1)

printf("SIGUSR1...\n");

else if(sig == SIGUSR2)

printf("SIGUSR2...\n");

else

printf("Something else...\n");

}

int main()

{

srand(time(NULL));

signal(SIGUSR1, hdl);

signal(SIGUSR2, hdl);

int number = 0;

int sum = 0, product = 1;

// имя разделяемой памяти

const char \*SHARED\_MEMORY\_NAME = "/my\_shared\_memory";

// размер разделяемой памяти

off\_t shm\_length = NUMBERS\_AMOUNT;

int shm\_fd = shm\_open(SHARED\_MEMORY\_NAME, O\_CREAT | O\_RDWR, 0666);

if (shm\_fd == -1)

{

printf("shm\_open - ошибка\n");

return -1;

}

if (ftruncate(shm\_fd, shm\_length) == -1)

{

printf("ftruncate - ошибка\n");

return -1;

}

int \*shared\_memory = (int \*)mmap(NULL, shm\_length, PROT\_READ | PROT\_WRITE, MAP\_SHARED, shm\_fd, 0);

if (shared\_memory == MAP\_FAILED)

{

printf("mmap ошибка\n");

return -1;

}

pid\_t pid = fork();

if (pid > 0)

{

for (int i = 0; i < NUMBERS\_AMOUNT; i++)

{

number = rand() % 100;

printf("\nСлучайное число: %d\n", number);

\*shared\_memory = number;

printf("Родительский процесс (pid=%d) отправил: %d\n", getpid(), number);

kill(pid, SIGUSR1);

pause();

}

printf("\nРодительский процесс ждёт сумму...\n");

kill(pid, SIGUSR1);

pause();

printf("Родительский процесс (pid=%d) получил: %d\n", getpid(), \*shared\_memory);

printf("\nРодительский процесс ждёт произведение...\n");

kill(pid, SIGUSR1);

pause();

printf("Родитель (pid=%d) получил произведение: %d\n", getpid(), \*shared\_memory);

if (munmap(shared\_memory, shm\_length) == -1)

{

printf("munmap ошибка\n");

return -1;

}

if (shm\_unlink(SHARED\_MEMORY\_NAME) == -1)

{

printf("shm\_unlink ошибка\n");

return -1;

}

printf("Родительский процесс завершен\n");

}

else

if (pid == 0)

{

for (int i = 0; i < NUMBERS\_AMOUNT; i++)

{

pause();

number = \*shared\_memory;

printf("Дочерний процесс (pid=%d) получил: %d\n", getpid(), number);

sum += number;

product \*= number;

kill(getppid(), SIGUSR2);

}

pause();

\*shared\_memory = sum;

printf("Дочерний процесс (pid=%d) отправил сумму: %d\n", getpid(), sum);

kill(getppid(), SIGUSR2);

pause();

\*shared\_memory = product;

printf("Дочерний процесс (pid=%d) отправил произведение: %d\n", getpid(), product);

kill(getppid(), SIGUSR2);

printf("Дочерний процесс завершен\n\n");

exit(0);

}

close(shm\_fd);

return 0;

}

**Результат выполнения**

**программы:**

