Отчет по лабораторной работе №8

«Взаимодействие процессов posix»

Выполнил: студент группы РИС-22-1б

Поважный Виталий Евгеньевич

Вариант 12 (4)

02.03.2025

**Задание**

Цель работы: научиться организовывать взаимодействие двух процессов

Необходимо выполнить следующие задачи:

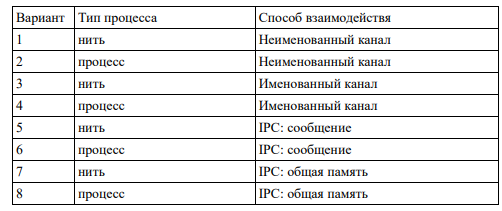
1. Создать два процесса, тип процесса определить по номеру варианта.

2. Процесс ОДИН должен передать ВТОРОМУ сигнал SIGUSR1

3. ВТОРОЙ процесс должен послать первому произвольную строку, используя способ взаимодействия процессов согласно варианту.

4. ПЕРВЫЙ процесс должен напечатать на экране полученные данные.

Варианты:



Мой вариант: 12 % 8 = 4;

**Ход работы**

Программа демонстрирует межпроцессное взаимодействие (IPC) с помощью именованного канала (FIFO) и сигналов.

1. Создаётся именованный канал (mkfifo()).
2. Форкается дочерний процесс (fork()).
3. Дочерний процесс:
   * Ждёт сигнал SIGUSR1 от родителя (pause()).
   * Открывает FIFO для записи и отправляет строку.
4. Родительский процесс:
   * Посылает SIGUSR1 дочернему процессу (kill()).
   * Открывает FIFO для чтения и получает сообщение.
5. Закрываются дескрипторы, FIFO удаляется, процессы завершаются.

Пример работы программы ниже:

root@minix:~# clang -o test test.c

root@minix:~# ./test

Received: Hello from Process 2!

root@minix:~#

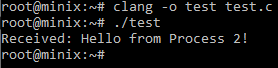


Рисунок 1 – Пример работы программы.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <string.h>

#include <errno.h>

#include <sys/wait.h>

#define FIFO\_NAME "/tmp/myfifo"

volatile sig\_atomic\_t flag = 0;

void handler(int sig) {

flag = 1;

}

int main() {

// Создаем именованный канал (FIFO)

if (mkfifo(FIFO\_NAME, 0666) == -1 && errno != EEXIST) {

perror("mkfifo");

exit(EXIT\_FAILURE);

}

pid\_t pid = fork();

if (pid == -1) {

perror("fork");

exit(EXIT\_FAILURE);

}

if (pid == 0) {

// Дочерний процесс (Процесс 2)

signal(SIGUSR1, handler); // Установка обработчика сигнала

// Ожидание сигнала SIGUSR1

while (!flag) {

pause();

}

// Открываем FIFO для записи

int fd = open(FIFO\_NAME, O\_WRONLY);

if (fd == -1) {

perror("open");

exit(EXIT\_FAILURE);

}

// Отправка строки через FIFO

const char \*message = "Hello from Process 2!";

if (write(fd, message, strlen(message)) == -1) {

perror("write");

exit(EXIT\_FAILURE);

}

close(fd);

exit(EXIT\_SUCCESS);

} else {

// Родительский процесс (Процесс 1)

// Даем время дочернему процессу зарегистрировать обработчик

sleep(1);

// Отправляем SIGUSR1 дочернему процессу

if (kill(pid, SIGUSR1) == -1) {

perror("kill");

exit(EXIT\_FAILURE);

}

// Открываем FIFO для чтения

int fd = open(FIFO\_NAME, O\_RDONLY);

if (fd == -1) {

perror("open");

exit(EXIT\_FAILURE);

}

// Чтение данных из FIFO

char buffer[256];

ssize\_t num\_bytes = read(fd, buffer, sizeof(buffer));

if (num\_bytes > 0) {

buffer[num\_bytes] = '\0';

printf("Received: %s\n", buffer);

} else {

perror("read");

}

close(fd);

unlink(FIFO\_NAME); // Удаляем FIFO

wait(NULL); // Ожидаем завершения дочернего процесса

}

return 0;

}