Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЕТ

к лабораторной работе №4

на тему

**УПРАВЛЕНИЕ ПРОЦЕССАМИ И ВЗАИМОДЕЙСТВИЕ ПРОЦЕССОВ**

Студент М. Ю. Пухов

Преподаватель С. И. Сиротко

Минск 2024

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 5](#_Toc146752070)

[Заключение 6](#_Toc146752071)

[Список использованных источников 7](#_Toc146752072)

[Приложение А (обязательное) Листинг кода 8](#_Toc146752073)

1. **ЦЕЛЬ РАБОТЫ**

Изучение основных особенностей подсистемы управления процессами и средств взаимодействия процессов в Unix.  
Практическое проектирование, реализация и отладка программных комплексов из нескольких взаимодействующих процессов.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Процесс-демон в Unix – это фоновый процесс, не имеющий контрольного терминала и выполняющий служебные или системные задачи.

Основные особенности демонов:

* Работают в фоновом режиме, независимо от контрольного терминала
* Запускаются при старте системы или по мере необходимости другими процессами
* Часто являются родительскими по отношению к другим фоновым процессам

Для превращения обычного процесса в демон используются следующие действия:

* Игнорирование сигналов SIGHUP и SIGINT
* Сброс маски создания файла по умолчанию
* Установка рабочего каталога в корневой каталог /
* Закрытие всех открытых файловых дескрипторов
* Потеря лидерства сеанса путем создания нового сеанса

Для перехвата и протоколирования сигналов в Linux используются системные вызовы sigaction() и signal(). В обработчике сигнала происходит запись информации о сигнале в лог-файл.

Таким образом, реализуется демон, выполняющий в фоновом режиме функцию регистрации заданных сигналов.

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате выполнения данной работы был разработан процесс-демон, осуществляющий перехват и протоколирование сигналов в соответствии с конфигурационным файлом.

Процесс после запуска преобразуется в демон путем закрытия файловых дескрипторов, игнорирования сигналов управления сеансом, смены рабочей директории на корневую.

При старте процесс регистрирует обработчики для заданного в конфигурационном файле набора сигналов. При возникновении этих сигналов запускается обработчик, который записывает в лог-файл информацию о времени возникновения и идентификаторе сигнала.

По сигналу SIGHUP происходит перечитывание конфигурационного файла и обновление набора регистрируемых сигналов.

В ходе тестирования успешно проверена работа демона по регистрации сигналов SIGINT, SIGTERM, SIGUSR1, а также его реконфигурация.

Данный процесс-демон может служить компонентом систем аварийной сигнализации и регистрации сбоев.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы был реализован процесс-демон, осуществляющий перехват и протоколирование заданных сигналов.

Были изучены особенности процессов-демонов и способы их создания в Unix. Реализованы функции преобразования дочернего процесса в демон путем игнорирования управляющих сигналов, сброса настроек терминала, смены рабочего каталога и закрытия файловых дескрипторов.

С использованием системных вызовов sigaction() и signal() настроен перехват заданного набора сигналов. Информация о полученных сигналах протоколируется в лог-файл с указанием времени и идентификатора сигнала.

Реализована возможность реконфигурации набора перехватываемых сигналов по сигналу SIGHUP на основе данных конфигурационного файла.

Полученный процесс-демон может использоваться как компонент систем мониторинга для регистрации возникающих аварийных ситуаций и отказов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

# Introduction aux scripts shell [Электронный ресурс]. – Режим доступа: https://doc.ubuntu-fr.org/tutoriel/script\_shell.

# Introduction au Script Bash [Электронный ресурс]. – Режим доступа: https://www.hostinger.fr/tutoriels/introduction-au-script-bash-avec-exemples.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Файл *main.c*

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <sys/wait.h>

// Глобальная переменная для отслеживания запущенного дочернего процесса

int child\_running = 0;

// Обработчик сигнала SIGINT (Ctrl+C)

void signal\_handler(int signal) {

// Создание дочернего процесса

pid\_t child\_pid = fork();

// Если это родительский процесс

if (child\_pid != 0) {

// Выход из родительского процесса

exit(0);

}

}

int main() {

// Инициализация переменной-счетчика

int counter = 0;

// Установка обработчика сигнала SIGINT

if (signal(SIGINT, signal\_handler) == SIG\_ERR) {

perror("Failed to set signal handler");

return EXIT\_FAILURE;

}

// Открытие файла "output.txt" в режиме добавления

FILE \*file = fopen("output.txt", "a");

if (file == NULL) {

perror("Failed to open file");

return EXIT\_FAILURE;

}

// Бесконечный цикл

while (1) {

// Запись значения счетчика в файл

fprintf(file, "%d\n", counter++);

// Принудительная синхронизация буферов вывода с файлом

fflush(file);

// Пауза на 1 секунду

sleep(1);

}

// Не достигнутый код, но добавлен для полноты

return 0;

}