Московский авиационный институт
(Национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Операционные системы»

Лабораторная работа № 2

Тема: Управление процессами в ОС

Студент: Туманов Георгий
Группа: 80-201
Преподаватель: Соколов А.А.
Дата:
Оценка:

Москва, 2019

1. **Постановка задачи**

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe).
Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Вариант 16: На вход программе подается команда интерпретатора команд и имя файла. Программа должна перенаправить стандартный ввод команды с этого файла и вывести результат команды в стандартный выходной поток. Использование операций write и printf запрещено.

2. **Описание программы**

Программа qwuk является вспомогательной и нужна для демонстрации основной программы. Она меняет регистр букв, а если встречает разделитель, выводит его ASCII-код. Остальные символы выводятся без изменений. Компилируется строкой: gcc qwuk.c -o qwuk
Основная программа читает две строки: команду интерпретатора и имя файла. Если вторая строка пустая, программа выполняет команду интерпретатора без изменений. Иначе, на стандартный ввод программе подаётся файл, имя которого написано во второй строке.

3. **Набор testcases**

№	Описание	Ввод
1	Тест с двумя строками	./qwuk lines.txt
2	Тест с одной строкой	date
3	Тест с /dev/null	./qwuk /dev/null

4. Результаты выполнения тестов.

test 1:
./qwuk
lines.txt
big letters SMALL LETTERS
aaBBccDD
oooo OOOOO PPPPpppp

test 2:
date

Чт окт 17 10:46:21 MSK 2019
test 3:
./qwuk
/dev/null

5. Листинг программы

```
#include <unistd.h>
#include <sys/wait.h>
#include <fcntl.h>

#define READ_END 0
#define WRITE_END 1
#define BUF_SZ 100

char rid(int from, char where[])
{
	char c;
	for (int i = 0; i < BUF_SZ - 1; i++)
	{
	read(from, &c, sizeof(c));
	if (c == '\n') { where[i] = '\0'; return c; }
	where[i] = c;
	}
	return 0;
}

int main()
{
	char prog[BUF_SZ], fname[BUF_SZ];
	prog[BUF_SZ - 1] = fname[BUF_SZ - 1] = '\0';

	rid(STDIN_FILENO, prog); //read prog name
	rid(STDIN_FILENO, fname); //read file name

	if (fname[0] == '\0')
```

```
        {
        execlp(prog, prog, NULL);
        return 0;
        }
        int file = open(fname, O_RDONLY);
        if (file == -1) return -1; //cannot open file
        if (dup2(file, STDIN_FILENO) == -1) { close(file); return -2; } //dup2 had been failing

        pid_t pid = fork();
        if (pid < 0) { close(file); return -3; } //fork was not sharp enough
        if (pid)
        {
        int s;
        waitpid(pid, &s, 0); //wait untill child is end
        close(file);
        }
        else execlp(prog, prog, NULL); //do the job
        return 0;
}
```

6. Выводы:

Освоил работу с несколькими процессами в Linux, а также технологию перенаправления входных и выходных потоков с помощью dup2.