Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу «Операционные системы»

Управление процессами в операционной системе

Студент: Симонов Сергей Яковлевич
Группа: М80 – 206Б-18
Вариант: 16
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2019

Содержание

Постановка задачи

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.

Общие сведения о программе

Программа компилируется из файла main.c. В программе используются следующие системные вызовы:

1. fork – для создания дочернего процесса.
2. Execlp – заменяет текущий образ процесса новым образом процесса.
3. Waitpid – приостанавливает выполнение текущего процесса до тех пор, пока дочерний процесс, указанный в параметре *pid*, не завершит выполнение, или пока не появится сигнал, который либо завершает текущий процесс либо требует вызвать функцию-обработчик.

Общий метод и алгоритм решения

С помощью fork создается дочерний процесс. Затем считывается команда и имя файла. Файл открывается для чтения (подразумевается, что данные в него уже внесены). Если файла не существует то выводится ошибка. После вызывается execlp заменяющий текущий образ процесса новым образом процесса, тем самым выполняя введенную команду. Закрытие файла и завершение дочернего процесса.

Файлы программы

main.c

```
#include <errno.h>
#include <malloc.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

const int ARG_MAX = 2097152;
int main()
{
pid_t pid;
int rv;
char* command = (char*)malloc(ARG_MAX);
char* file = (char*)malloc(ARG_MAX);

switch (pid = fork()) {
case -1: {
perror("fork"); //~ произошла ошибка~
exit(1); // ~выход из родительского процесса~
}
default: {

case 0: {
//~ Потомок~
scanf("%s\n%s", command, file);
FILE* my = freopen(file, "r", stdin);
if (!my) {
perror("File ERROR");
exit(errno);
}
rv = execlp(command, command, NULL);
if (rv)
```

```
perror("Exec ERROR");
fclose(my);
_exit(rv);
}
//~Родитель~
waitpid(pid, &rv, 0);
exit(WEXITSTATUS(rv));
}
}
free(file);
free(command);
}
```

Демонстрация работы программы

```
sergey@sergey-RedmiBook-14:~/labs/OS/lab2$ ./a.out
wc file
2  2 10
```

Вывод

Я научился создавать процессы, используя системный вызов fork(). Также я получил новые знания о файловых дискрипторах и выполнении команд интерпретатора команд с помощью семейства системных вызовов exec().