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

Лабораторная работа №4 по курсу «Операционные системы»

Отображение файлов в память

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

Москва, 2019

Содержание

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 решения задачи один или несколько дочерних процессов. Взаимодействие между процессами происходит черех системные сигналы\события\или через отображаемые файлы.
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.

Общие сведения о программе

Программа состоит из одного файла main.c.
В программе используются следующие заголовочные файлы: #include <ctype.h>,#include <errno.h>,#include <fcntl.h>,#include <malloc.h>,#include <semaphore.h>,#include <signal.h>,#include <stdio.h>,#include <stdlib.h>,#include <string.h>,#include <sys/mman.h>,#include <sys/stat.h>,#include <sys/types.h>,#include <sys/wait.h>,#include <unistd.h>
Используются следующие системные вызовы.

- shm_open() – используется для выделения разделяемой памяти.
- ftruncate – устанавливают длину обычного файла с именем *path* или файловым дескриптором *fd* в *length* байт.
- mmap – позволяет выполнить отбражение файла или устройства в память.
- fork – создает дочерний процесс.
- execlp – дублируют действия оболочки, относящиеся к поиску исполняемого файла, если указанное имя файла не содержит символ черты.
- waitpid – приостанавливает выполнение текущего процесса до тех пор, пока дочерний процесс не завершится, или до появления сигнала, который либо завершает текущий процесс, либо требует вызвать функцию-обработчик.
- munmap – отражает *length* байтов, начиная со смещения *offset* файла (или другого объекта), определенного файловым описателем *fd*, в память, начиная с адреса *start*.

Общий метод и алгоритм решения

Собственно, цель программы вызвать с помощью exec* некую программу и перенаправить ввод с указанного файла. Отличие от второй лабораторной лишь в том, что нужно использовать mmap для отображения виртуальной памяти. Логично создавать для вызова exec отдельный процесс, так как эта функция является системным вызовом и замещает собой породивший её процесс.

Основные файлы программы

main.c

```
#include <ctype.h>
#include <errno.h>
#include <fcntl.h>
#include <malloc.h>
#include <semaphore.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

const int ARG_MAX = 2097152 + 1;

int main()
{
   pid_t pid;
   int rv, fd;
   char* memory;
   // создаем разделяемую память
   if ((fd = shm_open("shared_file", O_RDWR | O_CREAT | O_TRUNC, S_IRUSR |
S_IWUSR)) == -1) {
      perror("shm::open_fail");
      exit(-1);
   }

   // задаем ей нужный размер
   if (ftruncate(fd, ARG_MAX * 2) == -1) {
      perror("trucate::fail");
      exit(-1);
   }

   memory = mmap(NULL, ARG_MAX * 2, PROT_READ | PROT_WRITE,
MAP_SHARED, fd, 0); // отображаем на вирт. память

   if (memory == MAP_FAILED) {
      perror("mmap::mapping_fail");
```

```
        fprintf(stderr, "%p", memory);
        exit(-1);
    }
    close(fd);

    char* command = memory;
    char* filename = memory + ARG_MAX;

    *command = *filename = '\0';

    switch (pid = fork()) {
        case -1: {
            perror("fork");
            exit(1);
        }

        case 0: {
            // Потомок
            while (*command == '\0' || *filename == '\0')
                sleep(1);
            FILE* my = freopen(filename, "r", stdin);
            if (!my) {
                perror("File ERROR");
                exit(errno);
            }
            rv = execlp(command, command, NULL);
            if (rv)
                perror("Exec ERROR");
            fclose(my);
            _exit(rv);
        }

        default: {
            //Родитель
            scanf("%s %s", command, filename);
            waitpid(pid, &rv, 0);
            exit(WEXITSTATUS(rv));
        }
    }
    if (munmap(memory, ARG_MAX * 2)) {
        perror("munmap");
        exit(-1);
    }
}
```

Демонстрация работы программы

```
sergey@sergey-RedmiBook-14:~/labs/OS/lab4/l4$ gcc main.c
-lrt -lpthread
sergey@sergey-RedmiBook-14:~/labs/OS/lab4/l4$ ./a.out
wc file
1  4 15
```

Вывод

В результате выаолнения данной лабораторной работы мной был изучен механизм отображения файлов в виртуальное адресное пространство. Были приобретены навыки отладки программ, имеющих больше одного процесса с помощью GDB. Также я научился синхранизировать работу процессов, используя семафоры.