Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа
по курсу «Операционные системы»
III Семестр

Задание 4
Вариант 10

Студент:	Алексеева М.А.
Группа:	М80-208Б-18
Преподаватель:	Миронов Е.С
Оценка:	
Дата:	

Москва 2019

1. Описание задания

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.
Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Вариант 10:

Программа позволяющая перенаправлять стандартный вывод. На вход программе подается имя выходного файла. Далее программа должна принимать различные команды интерпретатора команд и весь их вывод перенаправлять в файл.

2. Код программы

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <semaphore.h>
#include <string.h>

int main(int argc, char **argv){

    char semIn[] = "/semIn";
    int BUFFER_SIZE = 100;

    char* tmp_name = strdup("/tmp/tmp_file.XXXXXX");
    int tmp_fd = mkstemp(tmp_name);

    int file_size = BUFFER_SIZE + 1;
    char file_filler[file_size];
    for (int i = 0; i < file_size; ++i) {
        file_filler[i] = '\0';
    }
    write(tmp_fd, file_filler, file_size);
    char* map = (char*)mmap(NULL, BUFFER_SIZE, PROT_WRITE | PROT_READ,
MAP_SHARED, tmp_fd, 0);
```

```
    sem_t* sem_in;
    sem_unlink(semIn);
    if ((sem_in = sem_open(semIn, O_CREAT, 0777, 0)) == SEM_FAILED ) {
        perror("sem_in");
        return 1;
    }

    int f;
  char buf[160];
  pid_t pid; // создание процесса
  if ((f = open(argv[1], O_RDWR | O_CREAT | O_APPEND)) == -1){
    printf("Cannot open file.\n");
    return -1;
  }
  pid = fork();
  if(pid == -1){
    printf("Can\'t fork\n"); // процесс не создался
    return -1;
  }//ребенок
  else if(pid == 0){
    sem_wait(sem_in);
    char c;
    while(read(f, &c, sizeof(char)) == sizeof(char)) {
        write(f, &c, sizeof(char));
    }
  }//родитель
  else{
      scanf("%s", map);
      if(dup2(f, 1) < 0) {
          printf("Error: can’t dup");
          exit(-1);
      }
      sem_post(sem_in);
      execlp(map, map, NULL); //выполнение
  }
  return 0;
}
```

3. Протокол работы программы

```
masha@masha-VirtualBox:~$ cd 2kurs/OS/os_03
masha@masha-VirtualBox:~/2kurs/OS/os_03$ ls
1.txt  main  main.c  report.odt
masha@masha-VirtualBox:~/2kurs/OS/os_03$ clear

masha@masha-VirtualBox:~/2kurs/OS/os_03$ ls
main  main.c  report.odt
masha@masha-VirtualBox:~/2kurs/OS/os_03$ ./main 1.txt
ls
masha@masha-VirtualBox:~/2kurs/OS/os_03$ ls
1.txt  main  main.c  report.odt
masha@masha-VirtualBox:~/2kurs/OS/os_03$ sudo cat 1.txt
```

1.txt
main
main.c
report.odt

5. Объяснение работы программы

Межпроцессорное взаимодействие осуществляется через файл, отражённый в память.

Процесс-родитель считывает из стандартного потока ввода команду интерпретатора командной строки в переменную map, и записывает его в отражённый на память файл, после чего оповещает дочерний процесс, что данные готовы для вычисления посредством семафора sem_in. Родитель при помощи команды dup2 заменяет стандартный вывод на файл, а затем вызывает команду, записанную в переменной map.В программе использован именнованный семафор , так как и у дочернего процесса и у родительского одно пространство имен и это позволяет из двух процессов обращаться к одному семафору , а не создавать ещё один.

Вывод

Memory-mapped files – механизм, позволяющий отображать файлы на участок памяти. Их использование может дать существенный прирост в производительности, по сравнению с обычной буферизированной работе с файлами. Так же они могут использоваться для межпроцессорного взаимодействия, т.к. при отображении файла на участок памяти все процессы разделяют эту память и имеют доступ к данным.