

Московский авиационный институт
(Национальный исследовательский университет)
Факультет "Информационные технологии и прикладная математика"
Кафедра "Вычислительная математика и программирование"

**Лабораторная работа №2 по курсу
“Операционные системы”**

Студент: Иванов Андрей Кириллович

Группа: М8О-208Б-22

Преподаватель: Миронов Евгений Сергеевич

Вариант: 13

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2023

Содержание

1 Репозиторий

<https://github.com/kumaroid/osLabs>

2 Цель работы

Приобретение практических навыков в:

- Освоении принципов работы с файловыми системами
- Обеспечении обмена данных между процессами посредством технологии «File mapping»

3 Задание

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 через отображаемые файлы (memory-mapped files). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

4 Описание работы программы

Задание аналогично первой лабораторной работе.

В ходе выполнения лабораторной работы я использовал следующие системные вызовы:

- `fork()` - создание нового процесса
- `sem_open()` - создание/открытие семафора
- `sem_post()` - увеличение значения семафора и разблокировка ожидающих потоков
- `sem_wait()` - уменьшение значения семафора. Если 0, то вызывающий поток блокируется
- `sem_close()` - закрытие семафора
- `shm_open()` - создание/открытие разделяемой памяти POSIX
- `shm_unlink()` - закрытие разделяемой памяти
- `ftruncate()` - уменьшение длины файла до указанной
- `mmap()` - отражение файла или устройства в памяти
- `munmap()` - снятие отражения
- `execvp()` - запуск файла на исполнение

5 Исходный код

utils.hpp

```
1 #pragma once
2
3 #include <iostream>
4 #include <fstream>
5 #include <string>
6 #include <unistd.h>
7 #include <sys/wait.h>
8 #include <sys/stat.h>
9 #include <fcntl.h>
10 #include <string.h>
11 #include <sys/mman.h>
12 #include <sys/stat.h>
13 #include <fcntl.h>
14 #include <unistd.h>
15 #include <semaphore.h>
16
17 sem_t* CreateSemaphore(const char *name, int value);
18 int CreateShm(const char* name);
19 char* MapSharedMemory(const int size, int fd);
20 int CreateFork();
21
22 constexpr const char *SEM_1 = "SEM_1";
23 constexpr const char *SEM_2 = "SEM_2";
24 constexpr const char *SEM_3 = "SEM_3";
25
26 const int FILE_SIZE = 1024;
```

utils.cpp

```
1 #include "utils.hpp"
2
3 sem_t* CreateSemaphore(const char *name, int value) {
4     sem_t *semptr = sem_open(name, O_CREAT, S_IRUSR | S_IWUSR,
5     value);
6     if (semptr == SEM_FAILED){
7         perror("Couldn't open the semaphore");
8         exit(EXIT_FAILURE);
9     }
10    return semptr;
11}
12
13 int CreateShm(const char* name) {
14     int fd = shm_open(name, O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);
15     if (fd == -1) {
16         std::cerr << "Failed shm_open\n";
17         exit(-1);
18     }
19     ftruncate(fd, 1024);
20     return fd;
21}
22
23 char* MapSharedMemory(const int size, int fd) {
24     char *memptr = (char*)mmap(nullptr, size, PROT_READ |
25     PROT_WRITE, MAP_SHARED, fd, 0);
26     if (memptr == MAP_FAILED) {
27         perror("Error with file mapping");
28         exit(EXIT_FAILURE);
29     }
30 }
```

```

27     }
28     return memptr;
29 }
30
31 int CreateFork() {
32     int pid;
33     pid = fork();
34     if (pid == -1) {
35         std::cerr << "Failed fork()\n";
36         exit(-2);
37     }
38     return pid;
39 }

```

parent.hpp

```

1 #pragma once
2
3
4 #include "utils.hpp"
5
6 int ParentWork();

```

parent.cpp

```

1 #include "parent.hpp"
2 #include "utils.hpp"
3
4 int ParentWork() {
5     pid_t pid;
6     int status;;
7     char *child1;
8     char *child2;
9     std::string line;
10
11     char* name1 = "/shm1";
12     sem_t *semptr1 = CreateSemaphore(SEM_1, 0);
13     int shd_fd1 = CreateShm(name1);
14     char *memptr1 = MapSharedMemory(FILE_SIZE, shd_fd1);
15
16     char* name2 = "/shm2";
17     sem_t *semptr2 = CreateSemaphore(SEM_2, 0);
18     int shd_fd2 = CreateShm(name2);
19     char *memptr2 = MapSharedMemory(FILE_SIZE, shd_fd2);
20
21     char* name3 = "/shm3";
22     sem_t *semptr3 = CreateSemaphore(SEM_3, 0);
23     int shd_fd3 = CreateShm(name3);
24     char *memptr3 = MapSharedMemory(FILE_SIZE, shd_fd3);
25
26     child1 = getenv("PATH_CHILD1");
27     child2 = getenv("PATH_CHILD2");
28
29     pid = CreateFork();
30     if (pid == 0) { // child 1
31         if (execlp(child1, "lower.out", name1, name2, nullptr) ==
-1) {
32             std::cerr << "Failed execlp()\n";
33             exit(-5);
34         }
35     }

```

```

36
37     pid = CreateFork();
38     if (pid == 0) { // child 2
39         if (execlp(child2, "underscore.out", name2, name3, nullptr
) == -1) {
40             std::cerr << "Failed execlp()\n";
41             exit(-5);
42         }
43     }
44
45     if (pid != 0) {
46         while (std::getline(std::cin, line)) {
47             line += '\n';
48             strcpy(memptr1, line.c_str());
49             sem_post(sem_ptr1);
50
51             if (line == "\n") {
52                 strcpy(memptr1, "\0");
53                 sem_post(sem_ptr1);
54                 break;
55             }
56             sem_wait(sem_ptr3);
57             std::string_view st(memptr3);
58             std::string s = {st.begin(), st.end()};
59             if (s != "\n") {
60                 std::cout << s << std::endl;
61                 strcpy(memptr3, "\n");
62             }
63         }
64     }
65
66     waitpid(-1, &status, 0);
67     waitpid(-1, &status, 0);
68
69     sem_close(sem_ptr1);
70     sem_unlink(SEM_1);
71     shm_unlink("/shm1");
72     munmap(memptr1, FILE_SIZE);
73     close(shd_fd1);
74
75     sem_close(sem_ptr2);
76     sem_unlink(SEM_2);
77     shm_unlink("/shm2");
78     munmap(memptr2, FILE_SIZE);
79     close(shd_fd2);
80
81     sem_close(sem_ptr3);
82     sem_unlink(SEM_3);
83     shm_unlink("/shm3");
84     munmap(memptr3, FILE_SIZE);
85     close(shd_fd3);
86     return 0;
87 }

```

underscore.cpp

```

1 #include "utils.hpp"
2
3 int main (int argc, char** argv) {
4     if (argc != 3) {

```

```

5         std::cerr << "Wrong argc in underscore.out\n";
6         exit(-6);
7     }
8
9     const char* fileName = argv[1];
10    const char* fileName2 = argv[2];
11
12    sem_t *semptr1 = CreateSemaphore(SEM_2, 0);
13    int shd_fd1 = CreateShm(fileName);
14    char *memptr1 = MapSharedMemory(FILE_SIZE, shd_fd1);
15
16    sem_t *semptr2 = CreateSemaphore(SEM_3, 0);
17    int shd_fd2 = CreateShm(fileName2);
18    char *memptr2 = MapSharedMemory(FILE_SIZE, shd_fd2);
19
20    while(true) {
21        sem_wait(semptr1);
22        std::string_view st(memptr1);
23        std::string s = {st.begin(), st.end()};
24        for (int i = 0; i < s.size(); ++i) {
25            if (s[i] == ' ') {
26                s[i] = '_';
27            }
28        }
29
30        if (s != "\n") {
31            strcpy(memptr2, s.c_str());
32            strcpy(memptr1, "\n");
33            sem_post(semptr2);
34        }
35
36        if (s == "\0") {
37            break;
38        }
39    }
40
41    sem_close(semptr1);
42    sem_unlink(SEM_2);
43    shm_unlink(fileName);
44    munmap(memptr1, FILE_SIZE);
45    close(shd_fd1);
46
47    sem_close(semptr2);
48    sem_unlink(SEM_3);
49    shm_unlink(fileName2);
50    munmap(memptr2, FILE_SIZE);
51    close(shd_fd2);
52
53 }

```

6 Тесты

```
1 #include <iostream>
2 #include <gtest/gtest.h>
3
4 #include "parent.hpp"
5
6 void TestParent(std::string &src, std::string &res) {
7     std::istringstream srcStream(src);
8
9     std::streambuf* buf = std::cin.rdbuf(srcStream.rdbuf());
10
11     testing::internal::CaptureStdout();
12     ParentWork();
13     ASSERT_EQ(testing::internal::GetCapturedStdout(), res + '\n');
14
15     std::cin.rdbuf(buf);
16 }
17
18
19 TEST(cin_test, ONE) {
20     std::string src = "        \n\n";
21     std::string res = "_____\n";
22     TestParent(src, res);
23 }
24
25 TEST(cin_test, TWO) {
26     std::string src = "AHAHAHAHAH\n\n";
27     std::string res = "ahahahahah\n";
28     TestParent(src, res);
29 }
30
31 TEST(cin_test, THREE) {
32     std::string src = "    HELLO wOrLd 12 3\n\n";
33     std::string res = "___hello_world_12_3\n";
34     TestParent(src, res);
35 }
```


7 Запуск тестов

```
andrew@DESKTOP-K3DH39N:~/MAI/osLabs/build/tests$ ./lab3_test
Running main() from /home/andrew/MAI/osLabs/build/_deps/gtest-src/googletest/
[=====] Running 3 tests from 1 test suite.
[-----] Global test environment set-up.
[-----] 3 tests from cin_test
[ RUN      ] cin_test.ONE
[       OK ] cin_test.ONE (14 ms)
[ RUN      ] cin_test.TWO
[       OK ] cin_test.TWO (1 ms)
[ RUN      ] cin_test.THREE
[       OK ] cin_test.THREE (1 ms)
[-----] 3 tests from cin_test (16 ms total)

[-----] Global test environment tear-down
[=====] 3 tests from 1 test suite ran. (16 ms total)
[ PASSED  ] 3 tests.
```

8 Выводы

В результате выполнения данной лабораторной работы была написана программа на языке C++, осуществляющая работу с процессами и взаимодействие между ними через системные сигналы и отображаемые файлы. Были приобретены практические навыки в освоении принципов работы с файловыми системами и обеспечении обмена данных между процессами посредством технологии «File mapping».