Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №3 по курсу

«Операционные системы»

Группа: М8О-213Б-23

Студент: Колесник Д.С.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 17.11.24

Москва, 2024

Постановка задачи

Вариант 15.

Родительский процесс создает дочерний процесс. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child проверяет строки на валидность правилу. Если строка соответствует правилу, то она выводится в стандартный поток вывода дочернего процесса, иначе в pipe2 выводится информация об ошибке. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода.

Вместо каналов используется shared_memory.

Общий метод и алгоритм решения

Использованные системные вызовы:

- shmget – создание новой области разделяемой памяти
- shmat – привязка области разделяемой памяти к адресному пространству процесса
- read – чтение данных из стандартного ввода
- write – запись данных в стандартных вывод
- shmdt – отсоединение области разделяемой памяти от адресного пространства процесса
- shmctl – управление областью разделяемой памяти, удаление ее

Сервер создает область разделяемой памяти, которая будет использоваться для обмена данными между клиентом и сервером. Сервер запрашивает у пользователя имя файла и строки, которые будут записаны в разделяемую память. Для подключения клиента используется идентификатор разделяемой памяти. Клиент проверяет корректность строк и записывает их в файл, указанный в сервере, либо выводит сообщение об ошибке.
При запуске программы сервер создает новую область разделяемой памяти с помощью shmget. Он выделяет память для хранения данных, а также для имени файла. Затем с помощью shmat область разделяемой памяти привязывается к адресному пространству процесса, что позволяет серверу работать с ней как с обычным массивом.
Сервер запрашивает у пользователя ввод имени файла, который будет храниться в области разделяемой памяти. Имя файла записывается в память, начиная с определенного смещения, чтобы оставить место для других данных.

Сервер формирует сообщение, которое включает идентификатор созданной области разделяемой памяти, и выводит его на стандартный вывод. Этот идентификатор может быть полезен для клиентов, которые хотят подключиться к этой области.

Сервер входит в бесконечный цикл, в котором он запрашивает ввод строки от пользователя.

Введенные строки записываются в область разделяемой памяти. Если пользователь вводит пустую строку или завершает ввод, сервер завершает цикл.

Код программы

server.c

```
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>

#define SHM_SIZE 4096
#define FILENAME_SIZE 4096

int intToStr(int num, char *str) {
    int i = 0;
    if (num < 0) {
        str[i++] = '-';
        num = -num;
    }
    int start = i;
    do {
        str[i++] = (num % 10) + '0';
        num /= 10;
    } while (num > 0);


    for (int j = start, end = i - 1; j < end; j++, end--) {
        char temp = str[j];
        str[j] = str[end];
        str[end] = temp;
    }
    str[i] = '\0';
    return i;
}

int main() {
    int shmid;
    char *shared_memory;

    shmid = shmget(IPC_PRIVATE, SHM_SIZE + FILENAME_SIZE, IPC_CREAT | 0666);
    if (shmid < 0) {
        const char msg[] = "ошибка: не удалось создать разделяемую память\n";
             write(STDERR_FILENO, msg, sizeof(msg));
             exit(EXIT_FAILURE);
    }

    shared_memory = (char *)shmat(shmid, NULL, 0);
    if (shared_memory == (char *)(-1)) {
```

```
        const char msg[] = "ошибка: не удалось привязать разделяемую память\n";
             write(STDERR_FILENO, msg, sizeof(msg));
             exit(EXIT_FAILURE);
    }

    const char msg[] = "Введите имя файла: ";
       write(STDIN_FILENO, msg, sizeof(msg));
    read(STDIN_FILENO, shared_memory + SHM_SIZE, FILENAME_SIZE);

    shared_memory[SHM_SIZE + strcspn(shared_memory + SHM_SIZE, "\n")] = 0;

    char msgForShmid[] = "Идентификатор разделяемой памяти: ";
    msgForShmid[strlen(msgForShmid)] = intToStr(shmid, msgForShmid +
strlen(msgForShmid));
    int len = strlen(msgForShmid);
       write(STDOUT_FILENO, msgForShmid, len);
    write(STDOUT_FILENO, "\n", 1);

    while (1) {
        const char msg[] = "Введите строку (или нажмите CTRL+D для завершения): ";
        write(STDIN_FILENO, msg, sizeof(msg));
        if (read(STDIN_FILENO, shared_memory, SHM_SIZE) <= 0) {
            break;
        }
        shared_memory[strcspn(shared_memory, "\n")] = 0;

        if (strlen(shared_memory) == 0) {
            break;
        }
    }
    strcpy(shared_memory, "exit");

    shmdt(shared_memory);
    shmctl(shmid, IPC_RMID, NULL);

    write(STDOUT_FILENO, "Сервер завершен.\n", strlen("Сервер завершен.\n"));
    return 0;
}
```

client.c

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#include <fcntl.h> // Для open
#include <sys/types.h> // Для ssize_t
```

```
#include <sys/stat.h> // Для mode_t


#define SHM_SIZE 4096
#define FILENAME_SIZE 4096

void write_to_stdout(const char *msg) {
    write(STDOUT_FILENO, msg, strlen(msg));
}

void write_error(const char *msg) {
    write(STDERR_FILENO, msg, strlen(msg));
}

int main() {
    int shmid;
    char *shared_memory;

    write_to_stdout("Введите идентификатор разделяемой памяти: ");
    char input[10];
    read(STDIN_FILENO, input, sizeof(input));
    shmid = atoi(input);

    shared_memory = (char *)shmat(shmid, NULL, 0);
    if (shared_memory == (char *)(-1)) {
        write_error("shmat: ошибка привязки разделяемой памяти\n");
        exit(1);
    }

    char *filename = shared_memory + SHM_SIZE;

    int file = open(filename, O_WRONLY | O_APPEND | O_CREAT, S_IRUSR | S_IWUSR);
    if (file < 0) {
        write_error("Ошибка при открытии файла\n");
        exit(1);
    }

    while (1) {

        sleep(1);

        if (strlen(shared_memory) > 0) {

            if (strcmp(shared_memory, "exit") == 0) {
                break;
            }

            if (isupper(shared_memory[0])) {
                write_to_stdout("Ввод: ");
```

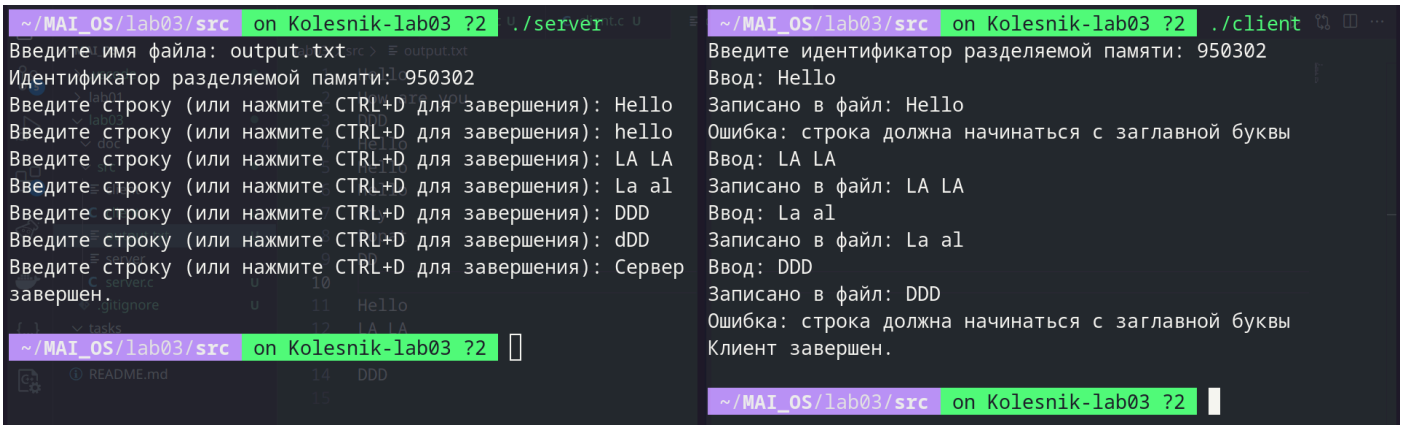
```
                write_to_stdout(shared_memory);
                write_to_stdout("\n");

                write(file, shared_memory, strlen(shared_memory));
                write(file, "\n", 1);
                write_to_stdout("Записано в файл: ");
                write_to_stdout(shared_memory);
                write_to_stdout("\n");
            } else {
                write_error("Ошибка: строка должна начинаться с заглавной буквы\n");
            }

            memset(shared_memory, 0, SHM_SIZE);
        }
    }

    close(file);
    shmdt(shared_memory);
    write_to_stdout("Клиент завершен.\n");
    return 0;
}
```

Протокол работы программы



Strace:

```
$ strace -f ./client
execve("./client", ["./client"], 0x7ffc46d8f2e8 /* 64 vars */) = 0
brk(NULL)                               = 0x62e635c30000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe31681550) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7475cf0d7000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=117899, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 117899, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7475cf0ba000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
```

```
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"...,
68, 896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7475cee00000
mprotect(0x7475cee28000, 2023424, PROT_NONE) = 0
mmap(0x7475cee28000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000) = 0x7475cee28000
mmap(0x7475cefbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) =
0x7475cefbd000
mmap(0x7475cf016000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x215000) = 0x7475cf016000
mmap(0x7475cf01c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0)
= 0x7475cf01c000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7475cf0b7000
arch_prctl(ARCH_SET_FS, 0x7475cf0b7740) = 0
set_tid_address(0x7475cf0b7a10)         = 918648
set_robust_list(0x7475cf0b7a20, 24)     = 0
rseq(0x7475cf0b80e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7475cf016000, 16384, PROT_READ) = 0
mprotect(0x62e634d7a000, 4096, PROT_READ) = 0
mprotect(0x7475cf111000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7475cf0ba000, 117899)          = 0
write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\270\320\264\320\265\320\275\321\202\320\270\321\204\320\270\320"..., 79Введите
идентификатор разделяемой памяти: ) = 79
read(0,
```

strace -f ./server

```
execve("./server", ["./server"], 0x7ffd69a3fc78 /* 64 vars */) = 0
brk(NULL)                               = 0x5cd5f821f000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffcf6340c40) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x786de7077000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=117899, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 117899, PROT_READ, MAP_PRIVATE, 3, 0) = 0x786de705a000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"...,
```

```
68, 896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x786de6e00000
mprotect(0x786de6e28000, 2023424, PROT_NONE) = 0
mmap(0x786de6e28000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000) = 0x786de6e28000
mmap(0x786de6fbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) =
0x786de6fbd000
mmap(0x786de7016000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x215000) = 0x786de7016000
mmap(0x786de701c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0)
= 0x786de701c000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x786de7057000
arch_prctl(ARCH_SET_FS, 0x786de7057740) = 0
set_tid_address(0x786de7057a10)         = 918703
set_robust_list(0x786de7057a20, 24)     = 0
rseq(0x786de70580e0, 0x20, 0, 0x53053053) = 0
mprotect(0x786de7016000, 16384, PROT_READ) = 0
mprotect(0x5cd5f7aaf000, 4096, PROT_READ) = 0
mprotect(0x786de70b1000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x786de705a000, 117899)          = 0
shmget(IPC_PRIVATE, 1280, IPC_CREAT|0666) = 950306
shmat(950306, NULL, 0)                  = 0x786de70b0000
write(0, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\270\320\274\321\217
\321\204\320\260\320\271\320\273\320\260"..., 35Введите имя файла: ) = 35
read(0, output.txt
"output.txt\n", 256)            = 11
write(1,
"\320\230\320\264\320\265\320\275\321\202\320\270\321\204\320\270\320\272\320\260\321\202\320
\276\321\200 \321\200\320\260\320"..., 71Идентификатор разделяемой памяти: 950306) = 71
write(1, "\n", 1
)                       = 1
write(0, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\201\321\202\321\200\320\276\320\272\321\203 (\320\270\320"..., 89Введите строку (или
нажмите CTRL+D для завершения): ) = 89
read(0, Hello
"Hello\n", 1024)                = 6
write(0, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\201\321\202\321\200\320\276\320\272\321\203 (\320\270\320"..., 89Введите строку (или
нажмите CTRL+D для завершения): ) = 89
read(0, lal
"lal\n", 1024)                  = 4
write(0, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\201\321\202\321\200\320\276\320\272\321\203 (\320\270\320"..., 89Введите строку (или
нажмите CTRL+D для завершения): ) = 89
read(0, "", 1024)                       = 0
```

```
shmdt(0x786de70b0000)                   = 0
shmctl(950306, IPC_RMID, NULL)          = 0
write(1, "\320\241\320\265\321\200\320\262\320\265\321\200
\320\267\320\260\320\262\320\265\321\200\321\210\320\265\320\275.\n", 31Сервер завершен.
) = 31
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

Язык Си предоставляет большую гибкость в вопросе синхронизации разных приложений между собой. Shared_memory - механизм, позволяющий гибко настраивать приложения для использования одинакового ресурса и взаимодействия с файлом.