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: М80-206Б-22

Студентка: Коломытцева Е. А.

Преподаватель: Миронов Е.С.

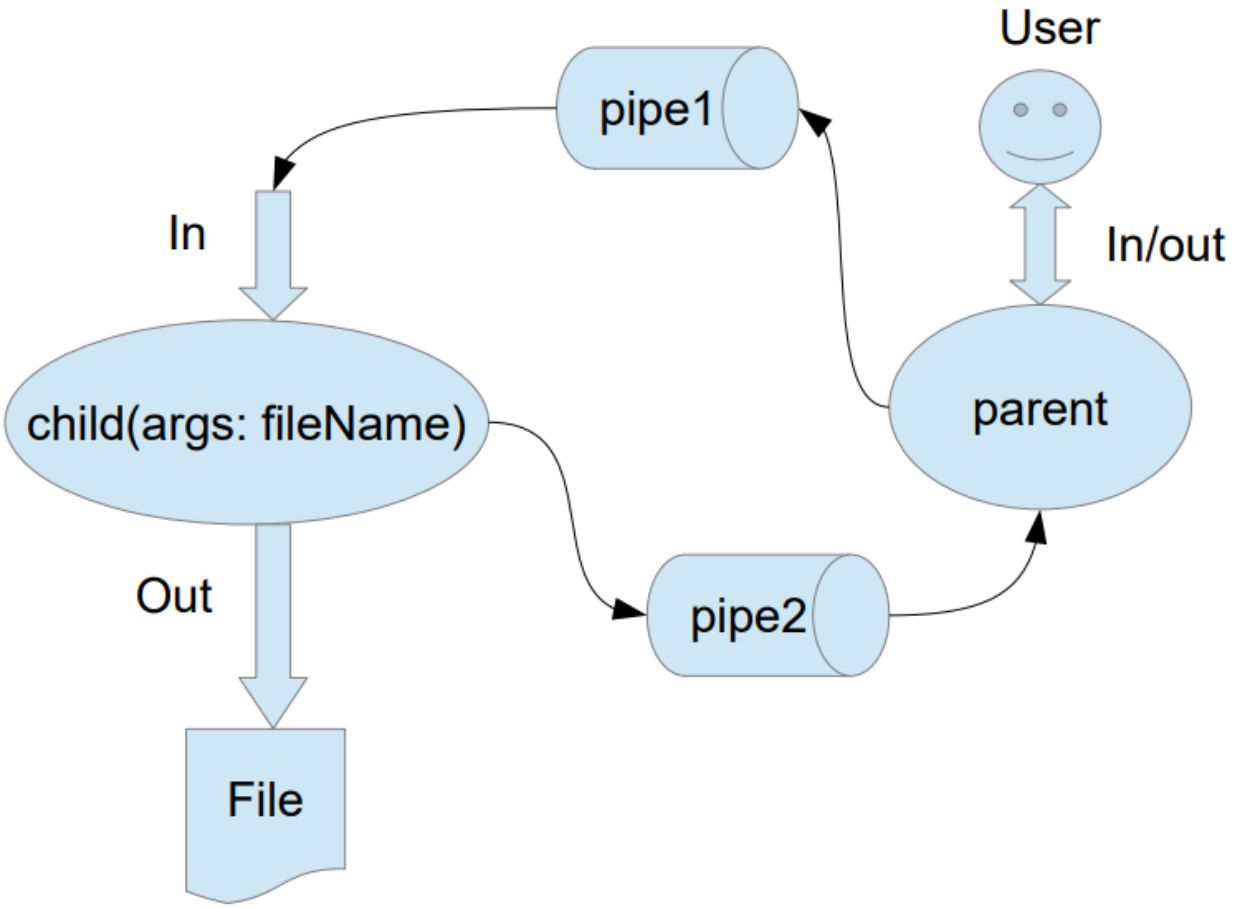
Оценка: _____

Дата: 30.11.23

Москва, 2023

# Постановка задачи

**Вариант 1.**




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.

Задание: Пользователь вводит команды вида: «число число число<endline>». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс считает их сумму и выводит её в файл. Числа имеют тип int. Количество чисел может быть произвольным.

# Общий метод и алгоритм решения

**Используемые системные вызовы:**

- pid_t fork(void); – создает дочерний процесс.
- int execl(const char *pathname, const char *arg, .../*, (char *) NULL */) – исполняет указанные файлы.
- int shm_open(const char* name, int flags, mode_t mode); – создает или открывает объект разделяемой памяти с заданным именем.
- int ftruncate(int fd, off_t length); – используется для изменения размера файла, указанного дескриптором файла fd, на указанный размер length.

- void* mmap(void* addr, size_t length, int prot, int flags, int fd, off_t offset); – используется для отображения файлов в память, создаёт новое отображение в виртуальном адресном пространстве вызывающего процесса.
- int munmap(void* addr, size_t length); – отменяет отображение области памяти, начинающейся с адреса addr и имеющей размер length.
- pid_t wait(int* status); – используется для ожидания завершения дочернего процесса в родительском процессе.
- int shm_unlink(const char* name); – используется для удаления объекта разделяемой памяти из системы.

Программа состоит из 5 файлов:
main.cpp – родительский процесс
child.cpp – дочерний процесс
func.cpp – функция вычисления суммы чисел из передаваемой строки
func.h – заголовочный файл для func.cpp
Makefile – файл, который собирает программу для выполнения задания

Команды для запуска программы:
make
./main
Пример ввода и вывода:
Enter filename:
sum.txt
Enter numbers:
12 3 4
Сумма чисел: 19

Описание программы и алгоритм выполнения работы:
Данная программа повторяет логику программы из лабораторной работы 1, однако вместо каналов взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы(memory-mapped files).

Подробное описание логики программы есть в отчете лабораторной работы 1, далее опишу использование memory-mapped files в это работе. В данном случае, родительский процесс создает дочерний процесс и оба общаются через общую область памяти. Родительский процесс создает область разделяемой памяти с помощью shm_open, и задает ей размер, равный BUFFER_SIZE, с помощью ftruncate. После этого родительский процесс отображает область разделяемой памяти в свое адресное пространство с помощью mmap. При помощи strncpy необходимые для передачи данные копируются в общую область памяти. При завершении каждого процесса вызывается munmap для отмены отображения области памяти. В конце родительского процесса вызывается shm_unlink для удаления объекта разделяемой памяти из системы.

# Код программы

**main.cpp**

```
#include <iostream>
#include <iostream>
using namespace std;
#include "func.h"

int main() {
 int fd = shm_open(SHARED_MEMORY_NAME, O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);
 ftruncate(fd, BUFFER_SIZE);

 char* shared_memory =
     (char*)mmap(NULL, BUFFER_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);

 pid_t child_pid = fork();

 if (child_pid == -1) {
   cerr << "Ошибка при создании дочернего процесса." << endl;
   return 1;
 }
 ////////////////////////////////////////////////////////////////////////////
 if (child_pid == 0) {  // Дочерний процесс

   if (execl("./child", "./child", NULL) == -1) {
     perror("Call execl was ended with erorr: ");
     exit(-1);
   }
//////////////////////////////////////////////////////////////////////////////////
 } else {
   // Родительский процесс

   char filename_nums[100];
   char filename[40];
   printf("Enter filename:\n");
   fgets(filename, sizeof(filename), stdin);
   filename[strcspn(filename, "\n")] = '\0';

   char input_nums[200];

   printf("Enter numbers:\n");
   fgets(input_nums, sizeof(input_nums), stdin);
   strcpy(filename_nums, filename);
   strcat(filename_nums, "\n");
   strcat(filename_nums, input_nums);

   strncpy(shared_memory, filename_nums, BUFFER_SIZE);
```

```
  wait(NULL);
  char suma[100];
  strncpy(suma, shared_memory, sizeof(suma));
  int summ = atoi(suma);
  cout << "Summa = " << summ << endl;
  close(fd);
  munmap(shared_memory, BUFFER_SIZE);
  shm_unlink(SHARED_MEMORY_NAME);
  wait(NULL);
 }
}
```

**child.cpp**

```
#include <iostream>

#include "func.h"
using namespace std;

int main() {
 int fd = shm_open(SHARED_MEMORY_NAME, O_RDWR, S_IRUSR | S_IWUSR);

 char* shared_memory =
     (char*)mmap(NULL, BUFFER_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
 while (strlen(shared_memory) == 0) {
   // Можно добавить небольшую задержку, чтобы не нагружать процессор
 }
 char fname_nums[256];
 char fname[100];
 char nums[100];
 strncpy(fname_nums, shared_memory, sizeof(fname_nums));

 char* token = strtok(fname_nums, "\n");
 if (token != NULL) {
   strncpy(fname, token, sizeof(fname));
   fname[sizeof(fname) - 1] = '\0';
 } else {
   printf("Ошибка разбора строки\n");
   return 1;
 }
 token = strtok(NULL, "\n");
 if (token != NULL) {
   strncpy(nums, token, sizeof(nums));
   nums[sizeof(nums) - 1] = '\0';
 } else {
   printf("Ошибка разбора строки\n");
   return 1;
 }
 // Читаем имя файла из канала
 FILE* file = fopen(fname, "w");
```

```
 if (file == NULL) {
   fprintf(stderr, "Failed to open the file\n");
   return 1;
 }

 int suma = sum_numbers(nums);
 fprintf(file, "%d", suma);

 sprintf(shared_memory, "%d", suma);
 close(fd);
 munmap(shared_memory, BUFFER_SIZE);
}
```

**func.cpp**

```
#include "func.h"

int sum_numbers(char input_nums[100]) {
  int sum = 0;
  int numbers[100];
  int count = 0;
  char* token = strtok(input_nums, " ");
  while (token != NULL && count < 100) {
    numbers[count] = atoi(token);  // Convert string to integer
    count++;
    token = strtok(NULL, " ");
  }
  for (int num = 0; num < count; num++) {
    sum += numbers[num];
  }
  return sum;
}
```

**func.h**

```
#include <fcntl.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

#include <iostream>

int sum_numbers(char input_nums[100]);
```

**Makefile**

```
CC = g++
CFLAGS = -Wall -Wextra

all: main child

main: main.cpp
```

```
	$(CC) $(CFLAGS) -o main main.cpp

child: child.cpp func.cpp
	$(CC) $(CFLAGS) -o child child.cpp func.cpp


clean:
	rm -f main child *.txt
```

# Протокол работы программы

**Тестирование:**

```
katya@katya:~/MAI_2/OS/OS_MAI/lab3/program$ ./main
Enter filename:
file.txt
Enter numbers:
1 5 2 3 45
Summa = 56
katya@katya:~/MAI_2/OS/OS_MAI/lab3/program$ cat file.txt
56
```

**Strace:**

```
$ strace -f ./main

execve("./main", ["./main"], 0x7ffc378e9098 /* 47 vars */) = 0

brk(NULL)                               = 0x556e44f35000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffdb5fd44b0) = -1 EINVAL (Недопустимый аргумент)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fd89db51000

access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=103055, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 103055, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd89db37000

close(3)                                = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.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\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2522552, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 2535872, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd89d800000

mmap(0x7fd89d89c000, 1249280, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x9c000) = 0x7fd89d89c000

mmap(0x7fd89d9cd000, 577536, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1cd000) = 0x7fd89d9cd000
```

```
     mmap(0x7fd89da5a000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x25a000) = 0x7fd89da5a000

     mmap(0x7fd89da68000, 12736, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7fd89da68000

     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<\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

     newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2072888, ...}, 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, 2117488, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd89d400000

     mmap(0x7fd89d422000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x22000) = 0x7fd89d422000

     mmap(0x7fd89d59a000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x19a000) = 0x7fd89d59a000

     mmap(0x7fd89d5f2000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1f1000) = 0x7fd89d5f2000

     mmap(0x7fd89d5f8000, 53104, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7fd89d5f8000

     close(3)                                = 0

     openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.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\0\0\0\0\0\0\0\0\0"..., 832) = 832

     newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=948816, ...}, AT_EMPTY_PATH) = 0

     mmap(NULL, 950520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd89d717000

     mmap(0x7fd89d725000, 516096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe000) = 0x7fd89d725000

     mmap(0x7fd89d7a3000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8c000) = 0x7fd89d7a3000

     mmap(0x7fd89d7fe000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe6000) = 0x7fd89d7fe000

     close(3)                                = 0

     openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3

     read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

     newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=141872, ...}, AT_EMPTY_PATH) = 0

     mmap(NULL, 144232, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd89db13000

     mmap(0x7fd89db16000, 110592, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x3000) = 0x7fd89db16000

     mmap(0x7fd89db31000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e000)
= 0x7fd89db31000
```

```
     mmap(0x7fd89db35000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x21000) = 0x7fd89db35000

     close(3)                                = 0

     mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fd89db11000

     arch_prctl(ARCH_SET_FS, 0x7fd89db12440) = 0

     set_tid_address(0x7fd89db12710)         = 17260

     set_robust_list(0x7fd89db12720, 24)     = 0

     rseq(0x7fd89db12d60, 0x20, 0, 0x53053053) = 0

     mprotect(0x7fd89d5f2000, 16384, PROT_READ) = 0

     mprotect(0x7fd89db35000, 4096, PROT_READ) = 0

     mprotect(0x7fd89d7fe000, 4096, PROT_READ) = 0

     mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fd89db0f000

     mprotect(0x7fd89da5a000, 45056, PROT_READ) = 0

     mprotect(0x556e44ec1000, 4096, PROT_READ) = 0

     mprotect(0x7fd89db86000, 8192, PROT_READ) = 0

     prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

     munmap(0x7fd89db37000, 103055)          = 0

     futex(0x7fd89da687fc, FUTEX_WAKE_PRIVATE, 2147483647) = 0

     getrandom("\x53\xf6\xbd\x66\x97\x42\xce\x8c", 8, GRND_NONBLOCK) = 8

     brk(NULL)                               = 0x556e44f35000

     brk(0x556e44f56000)                     = 0x556e44f56000

     openat(AT_FDCWD, "/dev/shm/my_shared_memory", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC,
0600) = 3

     ftruncate(3, 256)                       = 0

     mmap(NULL, 256, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7fd89db50000

     clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7fd89db12710) = 17261

     strace: Process 17261 attached

     [pid 17260] newfstatat(1, "",  <unfinished ...>

     [pid 17261] set_robust_list(0x7fd89db12720, 24 <unfinished ...>

     [pid 17260] <... newfstatat resumed>{st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0),
...}, AT_EMPTY_PATH) = 0

     [pid 17261] <... set_robust_list resumed>) = 0

     [pid 17260] write(1, "Enter filename:\n", 16Enter filename:

     ) = 16

     [pid 17261] execve("./child", ["./child"], 0x7ffdb5fd4618 /* 47 vars */ <unfinished
...>
```

```
     [pid 17260] newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...},
AT_EMPTY_PATH) = 0

     [pid 17260] read(0,  <unfinished ...>

     [pid 17261] <... execve resumed>)       = 0

     [pid 17261] brk(NULL)                   = 0x55f1088fd000

     [pid 17261] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc96d89560) = -1 EINVAL
(Недопустимый аргумент)

     [pid 17261] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f98d5c14000

     [pid 17261] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или
каталога)

     [pid 17261] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

     [pid 17261] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=103055, ...},
AT_EMPTY_PATH) = 0

     [pid 17261] mmap(NULL, 103055, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f98d5bfa000

     [pid 17261] close(3)                    = 0

     [pid 17261] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6",
O_RDONLY|O_CLOEXEC) = 3

     [pid 17261] read(3,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

     [pid 17261] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2522552, ...},
AT_EMPTY_PATH) = 0

     [pid 17261] mmap(NULL, 2535872, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f98d5800000

     [pid 17261] mmap(0x7f98d589c000, 1249280, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9c000) = 0x7f98d589c000

     [pid 17261] mmap(0x7f98d59cd000, 577536, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1cd000) = 0x7f98d59cd000

     [pid 17261] mmap(0x7f98d5a5a000, 57344, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25a000) = 0x7f98d5a5a000

     [pid 17261] mmap(0x7f98d5a68000, 12736, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f98d5a68000

     [pid 17261] close(3)                    = 0

     [pid 17261] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

     [pid 17261] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P<\2\0\0\0\0\0"...,
832) = 832

     [pid 17261] 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

     [pid 17261] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2072888, ...},
AT_EMPTY_PATH) = 0

     [pid 17261] 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

     [pid 17261] mmap(NULL, 2117488, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f98d5400000
```

```
     [pid 17261] mmap(0x7f98d5422000, 1540096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x7f98d5422000

     [pid 17261] mmap(0x7f98d559a000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19a000) = 0x7f98d559a000

     [pid 17261] mmap(0x7f98d55f2000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1f1000) = 0x7f98d55f2000

     [pid 17261] mmap(0x7f98d55f8000, 53104, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f98d55f8000

     [pid 17261] close(3)                    = 0

     [pid 17261] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3

     [pid 17261] read(3,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

     [pid 17261] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=948816, ...},
AT_EMPTY_PATH) = 0

     [pid 17261] mmap(NULL, 950520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f98d5b11000

     [pid 17261] mmap(0x7f98d5b1f000, 516096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x7f98d5b1f000

     [pid 17261] mmap(0x7f98d5b9d000, 372736, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8c000) = 0x7f98d5b9d000

     [pid 17261] mmap(0x7f98d5bf8000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe6000) = 0x7f98d5bf8000

     [pid 17261] close(3)                    = 0

     [pid 17261] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC)
= 3

     [pid 17261] read(3,
"\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

     [pid 17261] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=141872, ...},
AT_EMPTY_PATH) = 0

     [pid 17261] mmap(NULL, 144232, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f98d5aed000

     [pid 17261] mmap(0x7f98d5af0000, 110592, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f98d5af0000

     [pid 17261] mmap(0x7f98d5b0b000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1e000) = 0x7f98d5b0b000

     [pid 17261] mmap(0x7f98d5b0f000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x21000) = 0x7f98d5b0f000

     [pid 17261] close(3)                    = 0

     [pid 17261] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f98d5aeb000

     [pid 17261] arch_prctl(ARCH_SET_FS, 0x7f98d5aec440) = 0

     [pid 17261] set_tid_address(0x7f98d5aec710) = 17261

     [pid 17261] set_robust_list(0x7f98d5aec720, 24) = 0

     [pid 17261] rseq(0x7f98d5aecd60, 0x20, 0, 0x53053053) = 0
```

```
[pid 17261] mprotect(0x7f98d55f2000, 16384, PROT_READ) = 0

[pid 17261] mprotect(0x7f98d5b0f000, 4096, PROT_READ) = 0

[pid 17261] mprotect(0x7f98d5bf8000, 4096, PROT_READ) = 0

[pid 17261] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f98d5ae9000

[pid 17261] mprotect(0x7f98d5a5a000, 45056, PROT_READ) = 0

[pid 17261] mprotect(0x55f108561000, 4096, PROT_READ) = 0

[pid 17261] mprotect(0x7f98d5c49000, 8192, PROT_READ) = 0

[pid 17261] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0

[pid 17261] munmap(0x7f98d5bfa000, 103055) = 0

[pid 17261] futex(0x7f98d5a687fc, FUTEX_WAKE_PRIVATE, 2147483647) = 0

[pid 17261] getrandom("\x91\xb1\x1e\x83\xca\x3b\xa4\x71", 8, GRND_NONBLOCK) = 8

[pid 17261] brk(NULL)                   = 0x55f1088fd000

[pid 17261] brk(0x55f10891e000)         = 0x55f10891e000

[pid 17261] openat(AT_FDCWD, "/dev/shm/my_shared_memory", O_RDWR|O_NOFOLLOW|O_CLOEXEC)
= 3

[pid 17261] mmap(NULL, 256, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f98d5c13000



[pid 17260] <... read resumed>"\n", 1024) = 1

[pid 17260] write(1, "Enter numbers:\n", 15Enter numbers:

) = 15

[pid 17260] read(0,

"\n", 1024)         = 1

[pid 17260] wait4(-1,  <unfinished ...>

[pid 17261] newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...},
AT_EMPTY_PATH) = 0

[pid 17261] write(1, "\320\236\321\210\320\270\320\261\320\272\320\260
\321\200\320\260\320\267\320\261\320\276\321\200\320\260 \321\201\321\202"..., 41Ошибка
разбора строки

) = 41

[pid 17261] exit_group(1)               = ?

[pid 17261] +++ exited with 1 +++

<... wait4 resumed>NULL, 0, NULL)       = 17261

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=17261, si_uid=1000,
si_status=1, si_utime=3898 /* 38.98 s */, si_stime=1 /* 0.01 s */} ---

write(1, "Summa = 0\n", 10Summa = 0

)             = 10

close(3)                                = 0
```

```
munmap(0x7fd89db50000, 256)             = 0
unlink("/dev/shm/my_shared_memory")     = 0
wait4(-1, NULL, 0, NULL)                = -1 ECHILD (Нет дочерних процессов)
exit_group(0)                           = ?
+++ exited with 0 +++
```

## Вывод

Данная работа научила меня пользоваться общей областью памяти для передачи данных между двумя процессами. В этой лабораторной я узнала какие файловые системы используются в таких системах как Windows, UNIX, узнала о существовании разных структур файлов, в виде последовательности байтов, в виде односвязного списка и в виде дерева. Узнала о преимуществах и недостатках структур, а также с помощью чего эти недостатки решаются. Также узнала, что каталоги тоже являются файлами. Это работа показала мне иной способ передачи данных между процессами, тем самым расширив мои знания в области операционных систем.