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: Барменков А.С.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 26.11.24

Москва, 2024

Постановка задачи

Вариант 19.

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 каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы
Условие лабораторной работы №3 повторяет условие лабораторной работы №1, но вместо каналов необходимо использовать shared memory.
Правило фильтрации: с вероятностью 80% строки отправляются в pipe1, иначе в pipe2.
Дочерние процессы удаляют все гласные из строк.

Общий метод и алгоритм решения

Использованные системные вызовы:

- **`sem_open()`** – Создает или открывает именованный семафор.
- **`sem_post()`** – Увеличивает значение семафора (сигнализирует о доступности ресурса).
- **`sem_wait()`** – Уменьшает значение семафора (ожидание доступа к ресурсу).
- **`sem_close()`** – Закрывает дескриптор семафора.
- **`sem_unlink()`** – Удаляет именованный семафор из системы.
- **`shm_open()`** – Создает или открывает объект разделяемой памяти.
- **`ftruncate()`** – Устанавливает размер разделяемой памяти.
- **`mmap()`** – Отображает разделяемую память в адресное пространство процесса.
- **`munmap()`** – Удаляет отображение разделяемой памяти из адресного пространства.
- **`shm_unlink()`** – Удаляет объект разделяемой памяти из системы.

Программа начинается с создания разделяемой памяти и двух семафоров, которые используются для обмена данными между родительским процессом и двумя дочерними процессами. Разделяемая память служит общей областью для передачи строк от родителя к дочерним процессам, а семафоры обеспечивают синхронизацию доступа к этим данным.

Родительский процесс сначала создает дочерние процессы с помощью fork(). После этого он ожидает пользовательский ввод. Введенные строки записываются в разделяемую память, а затем, с вероятностью 80%, передаются на обработку первому дочернему процессу, иначе — второму. Выбор осуществляется случайным образом, используя генерацию случайного числа.

Каждый дочерний процесс работает параллельно. Получив строку из разделяемой памяти через свой семафор, процесс удаляет из нее все гласные буквы и записывает результат в файл, имя которого пользователь вводит при запуске программы.

Родительский процесс продолжает передавать строки до тех пор, пока пользователь не введет команду "exit". После этого он отправляет специальный сигнал завершения дочерним процессам, завершает их выполнение и корректно освобождает все ресурсы (разделяемую память и семафоры).

В итоге программа организует параллельное выполнение двух дочерних процессов, которые обрабатывают строки и записывают результаты в свои файлы, а родительский процесс управляет их работой и обеспечивает распределение задач.

Код программы

parent.c

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

#define SHM_NAME "/shm_example"
#define SEM_CHILD1 "/sem_child1"
#define SEM_CHILD2 "/sem_child2"

#define SHM_SIZE 1024

void HandleError(const char *msg) {
    write(STDERR_FILENO, msg, strlen(msg));
    write(STDERR_FILENO, "\n", 1);
    exit(1);
}

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

ssize_t Getline(char **lineptr, size_t *n, int fd) {
    if (*lineptr == NULL) {
        *lineptr = malloc(128);
        *n = 128;
    }

    size_t pos = 0;
    char c;
    while (read(fd, &c, 1) == 1) {
        if (pos >= *n - 1) {
            *n *= 2;
            *lineptr = realloc(*lineptr, *n);
        }
        (*lineptr)[pos++] = c;
        if (c == '\n') {
            break;
        }
    }

    if (pos == 0) {
        return -1;
    }

    (*lineptr)[pos] = '\0';
    return pos;
}

int main() {
    char *file1 = NULL, *file2 = NULL;
    size_t file_len = 0;
    char *input = NULL;
    size_t len = 0;
    ssize_t nread;

    Print("Введите имя файла для дочернего процесса 1: ");
    Getline(&file1, &file_len, STDIN_FILENO);
    file1[strcspn(file1, "\n")] = 0;
```

```
Print("Введите имя файла для дочернего процесса 2: ");
Getline(&file2, &file_len, STDIN_FILENO);
file2[strcspn(file2, "\n")] = 0;

// Создаем shared memory
int shm_fd = shm_open(SHM_NAME, O_CREAT | O_RDWR, 0644);
if (shm_fd == -1) {
    HandleError("Ошибка создания разделяемой памяти");
}

if (ftruncate(shm_fd, SHM_SIZE) == -1) {
    HandleError("Ошибка установки размера разделяемой памяти");
}

char *shm_ptr = mmap(NULL, SHM_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
if (shm_ptr == MAP_FAILED) {
    HandleError("Ошибка отображения разделяемой памяти");
}

// Создаем семафоры
sem_t *sem_child1 = sem_open(SEM_CHILD1, O_CREAT, 0644, 0);
sem_t *sem_child2 = sem_open(SEM_CHILD2, O_CREAT, 0644, 0);
if (sem_child1 == SEM_FAILED || sem_child2 == SEM_FAILED) {
    HandleError("Ошибка создания семафоров");
}

pid_t child1, child2;

if ((child1 = fork()) == 0) {
    execlp("./child1", "./child1", file1, NULL);
    HandleError("Ошибка запуска дочернего процесса 1");
}

if ((child2 = fork()) == 0) {
    execlp("./child2", "./child2", file2, NULL);
    HandleError("Ошибка запуска дочернего процесса 2");
}

srand(time(NULL));

while (1) {
    Print("Введите строку (или 'exit' для завершения): ");
    nread = Getline(&input, &len, STDIN_FILENO);
    input[strcspn(input, "\n")] = 0;

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

    // Копируем данные в shared memory
    strncpy(shm_ptr, input, SHM_SIZE - 1);

    int r = rand() % 5 + 1;
    //printf("r = %d\n", r);
    if (r == 3) {
        sem_post(sem_child2);
    } else {
        sem_post(sem_child1);
    }
}

Print("Работа завершена.\n");

// Удаляем ресурсы
munmap(shm_ptr, SHM_SIZE);
shm_unlink(SHM_NAME);
sem_unlink(SEM_CHILD1);
sem_unlink(SEM_CHILD2);
```

```
    free(file1);
    free(file2);
    free(input);

    return 0;
}
```

child1.c/child2,c

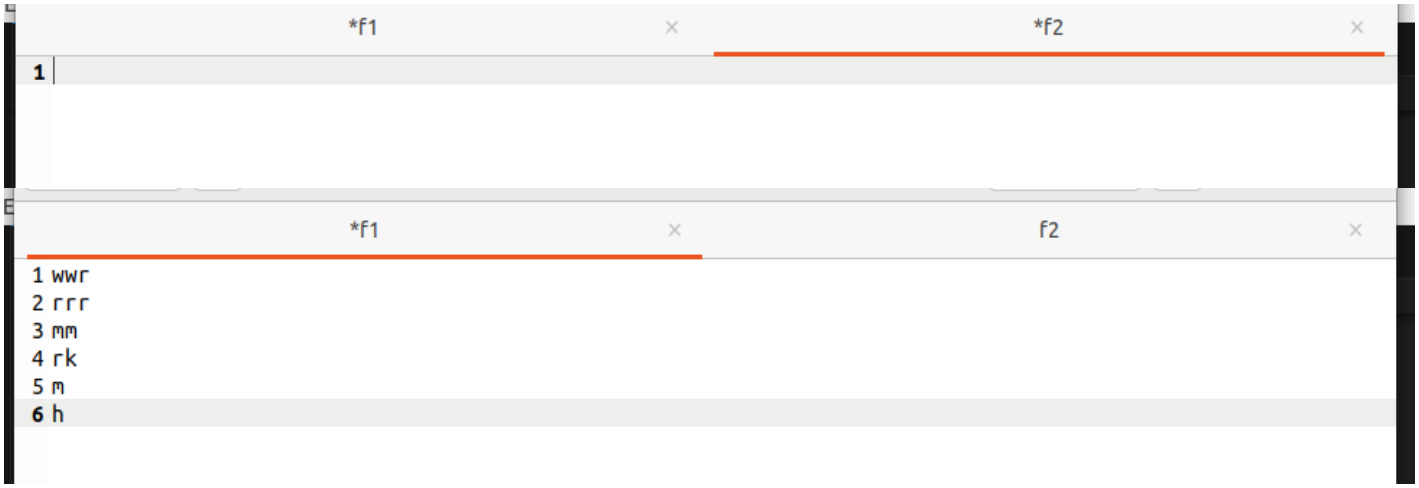
```
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <semaphore.h>

#define SHM_NAME "/shm_example"
#define SEM_CHILD1 "/sem_child1"
#define SHM_SIZE 1024

void HandleError(const char *msg) {
    write(STDERR_FILENO, msg, strlen(msg));
    write(STDERR_FILENO, "\n", 1);
    exit(1);
}

void RemoveVowels(char *str) {
    char *p = str, *q = str;
    while (*p) {
        if (!strchr("AEIOUaeiou", *p)) {
            *q++ = *p;
        }
        p++;
    }
    *q = '\0';
}

int main(int argc, char *argv[]) {
    if (argc < 2) {
        HandleError("Usage: <program> <output_file>");
    }

    int fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fd == -1) {
        HandleError("Cannot open file");
    }

    int shm_fd = shm_open(SHM_NAME, O_RDWR, 0644);
    if (shm_fd == -1) {
        HandleError("Ошибка доступа к разделяемой памяти");
    }

    char *shm_ptr = mmap(NULL, SHM_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
    if (shm_ptr == MAP_FAILED) {
        HandleError("Ошибка отображения разделяемой памяти");
    }

    sem_t *sem_child1 = sem_open(SEM_CHILD1, 0);
    if (sem_child1 == SEM_FAILED) {
        HandleError("Ошибка доступа к семафору");
    }

    while (1) {
        sem_wait(sem_child1);

        char buffer[SHM_SIZE];
        strncpy(buffer, shm_ptr, SHM_SIZE - 1);
```

```
        buffer[SHM_SIZE - 1] = '\0';

        RemoveVowels(buffer);
        write(fd, buffer, strlen(buffer));
        write(fd, "\n", 1);
    }

    munmap(shm_ptr, SHM_SIZE);
    close(fd);

    return 0;
}
```

Протокол работы программы

Тестирование:

```
artemdelgray@artemdelgray-VirtualBox:~/Загрузки/Telegram Desktop$ ./p
Введите имя файла для дочернего процесса 1: f1
Введите имя файла для дочернего процесса 2: f2
Введите строку (или 'exit' для завершения): wwer
Введите строку (или 'exit' для завершения): error
Введите строку (или 'exit' для завершения): mama
Введите строку (или 'exit' для завершения): reka
Введите строку (или 'exit' для завершения): mai
Введите строку (или 'exit' для завершения): hi
Введите строку (или 'exit' для завершения): exit
Работа завершена.
```



Strace:

```
$ strace -f ./p
execve("./p", ["./p"], 0x7fffbf05c598 /* 46 vars */) = 0
brk(NULL)                               = 0x59ac97eb6000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fffc9308c40) = -1 EINVAL (Недопустимый аргумент)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x79a34d0d3000
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=58047, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 58047, PROT_READ, MAP_PRIVATE, 3, 0) = 0x79a34d0c4000
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) = 0x79a34ce00000
mprotect(0x79a34ce28000, 2023424, PROT_NONE) = 0
mmap(0x79a34ce28000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) =
0x79a34ce28000
mmap(0x79a34cfbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) =
0x79a34cfbd000
mmap(0x79a34d016000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) =
0x79a34d016000
mmap(0x79a34d01c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x79a34d01c000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x79a34d0c1000
arch_prctl(ARCH_SET_FS, 0x79a34d0c1740) = 0
set_tid_address(0x79a34d0c1a10)         = 4145
set_robust_list(0x79a34d0c1a20, 24)     = 0
rseq(0x79a34d0c20e0, 0x20, 0, 0x53053053) = 0
mprotect(0x79a34d016000, 16384, PROT_READ) = 0
mprotect(0x59ac97ac9000, 4096, PROT_READ) = 0
mprotect(0x79a34d10d000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x79a34d0c4000, 58047)           = 0
write(1, "\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"..., 79Введите имя файла для дочернего процесса 1: ) = 79
getrandom("\x30\x5c\x57\x10\x22\x8e\x81\x60", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x59ac97eb6000
brk(0x59ac97ed7000)                     = 0x59ac97ed7000
read(0,
"\n", 1)                        = 1
write(1, "\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"..., 79Введите имя файла для дочернего процесса 2: ) = 79
read(0, exit
"e", 1)                         = 1
read(0, "x", 1)                         = 1
read(0, "i", 1)                         = 1
read(0, "t", 1)                         = 1
read(0, "\n", 1)                        = 1
openat(AT_FDCWD, "/dev/shm/shm_example", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0644) = 3
ftruncate(3, 1024)                      = 0
mmap(NULL, 1024, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x79a34d10c000
openat(AT_FDCWD, "/dev/shm/sem.sem_child1", O_RDWR|O_NOFOLLOW) = 4
newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x79a34d0d2000
close(4)                                = 0
openat(AT_FDCWD, "/dev/shm/sem.sem_child2", O_RDWR|O_NOFOLLOW) = 4
newfstatat(4, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) = 0x79a34d0d1000
close(4)                                = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 4146
attached
, child_tidptr=0x79a34d0c1a10) = 4146
[pid  4145] clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished
...>
[pid  4146] set_robust_list(0x79a34d0c1a20, 24) = 0
[pid  4146] execve("./child1", ["./child1", ""], 0x7fffc9308e18 /* 46 vars */strace: Process 4147
attached
 <unfinished ...>
[pid  4147] set_robust_list(0x79a34d0c1a20, 24 <unfinished ...>
[pid  4145] <... clone resumed>, child_tidptr=0x79a34d0c1a10) = 4147
[pid  4147] <... set_robust_list resumed>) = 0
[pid  4147] execve("./child2", ["./child2", "exit"], 0x7fffc9308e18 /* 46 vars */ <unfinished ...>
[pid  4145] write(1, "\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"..., 73Введите строку (или 'exit' для
завершения): ) = 73
```

```
[pid  4145] read(0,  <unfinished ...>
[pid  4147] <... execve resumed>)       = 0
[pid  4147] brk(NULL)                   = 0x62b3a3b9d000
[pid  4146] <... execve resumed>)       = 0
[pid  4146] brk(NULL)                   = 0x5eaf443d3000
[pid  4147] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffec3e34060) = -1 EINVAL (Недопустимый аргумент)
[pid  4147] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7a90e77e1000
[pid  4147] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или каталога)
[pid  4147] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
[pid  4147] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=58047, ...}, AT_EMPTY_PATH) = 0
[pid  4147] mmap(NULL, 58047, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7a90e77d2000
[pid  4147] close(3)                    = 0
[pid  4147] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
[pid  4147] 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
[pid  4147] 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  4147] 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
[pid  4147] 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
[pid  4147] newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
[pid  4147] 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  4147] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7a90e7400000
[pid  4147] mprotect(0x7a90e7428000, 2023424, PROT_NONE) = 0
[pid  4147] mmap(0x7a90e7428000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000) = 0x7a90e7428000
[pid  4147] mmap(0x7a90e75bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000)
= 0x7a90e75bd000
[pid  4147] mmap(0x7a90e7616000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x215000) = 0x7a90e7616000
[pid  4147] mmap(0x7a90e761c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1,
0) = 0x7a90e761c000
[pid  4147] close(3)                    = 0
[pid  4147] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7a90e77cf000
[pid  4147] arch_prctl(ARCH_SET_FS, 0x7a90e77cf740) = 0
[pid  4147] set_tid_address(0x7a90e77cfa10) = 4147
[pid  4147] set_robust_list(0x7a90e77cfa20, 24) = 0
[pid  4147] rseq(0x7a90e77d00e0, 0x20, 0, 0x53053053) = 0
[pid  4147] mprotect(0x7a90e7616000, 16384, PROT_READ <unfinished ...>
[pid  4146] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffcdd0042b0 <unfinished ...>
[pid  4147] <... mprotect resumed>)     = 0
[pid  4147] mprotect(0x62b3a3502000, 4096, PROT_READ <unfinished ...>
[pid  4146] <... arch_prctl resumed>)   = -1 EINVAL (Недопустимый аргумент)
[pid  4147] <... mprotect resumed>)     = 0
[pid  4146] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid  4147] mprotect(0x7a90e781b000, 8192, PROT_READ) = 0
[pid  4146] <... mmap resumed>)         = 0x743b7ed33000
[pid  4146] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid  4147] prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
[pid  4146] <... access resumed>)       = -1 ENOENT (Нет такого файла или каталога)
[pid  4147] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid  4146] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid  4147] munmap(0x7a90e77d2000, 58047 <unfinished ...>
[pid  4146] <... openat resumed>)       = 3
[pid  4146] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=58047, ...}, AT_EMPTY_PATH) = 0
[pid  4147] <... munmap resumed>)       = 0
[pid  4146] mmap(NULL, 58047, PROT_READ, MAP_PRIVATE, 3, 0) = 0x743b7ed24000
[pid  4146] close(3)                    = 0
[pid  4147] openat(AT_FDCWD, "exit", O_WRONLY|O_CREAT|O_TRUNC, 0644 <unfinished ...>
[pid  4146] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
[pid  4147] <... openat resumed>)       = 3
[pid  4146] 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
[pid  4146] 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  4146] 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
[pid  4146] 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
[pid  4146] newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
[pid  4146] 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  4147] openat(AT_FDCWD, "/dev/shm/shm_example", O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>
[pid  4146] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x743b7ea00000
[pid  4147] <... openat resumed>)       = 4
[pid  4147] mmap(NULL, 1024, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0 <unfinished ...>
[pid  4146] mprotect(0x743b7ea28000, 2023424, PROT_NONE <unfinished ...>
[pid  4147] <... mmap resumed>)         = 0x7a90e781a000
[pid  4146] <... mprotect resumed>)     = 0
[pid  4147] openat(AT_FDCWD, "/dev/shm/sem.sem_child2", O_RDWR|O_NOFOLLOW <unfinished ...>
[pid  4146] mmap(0x743b7ea28000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000 <unfinished ...>
[pid  4147] <... openat resumed>)       = 5
[pid  4146] <... mmap resumed>)         = 0x743b7ea28000
[pid  4147] newfstatat(5, "", {st_mode=S_IFREG|0644, st_size=32, ...}, AT_EMPTY_PATH) = 0
[pid  4146] mmap(0x743b7ebbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000
<unfinished ...>
[pid  4147] getrandom( <unfinished ...>
[pid  4146] <... mmap resumed>)         = 0x743b7ebbd000
[pid  4147] <... getrandom resumed>"\xeb\x6f\x2c\x0f\xbb\xdd\x61\x98", 8, GRND_NONBLOCK) = 8
[pid  4147] brk(NULL <unfinished ...>
[pid  4146] mmap(0x743b7ec16000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x215000 <unfinished ...>
[pid  4147] <... brk resumed>)          = 0x62b3a3b9d000
[pid  4146] <... mmap resumed>)         = 0x743b7ec16000
[pid  4147] brk(0x62b3a3bbe000 <unfinished ...>
[pid  4146] mmap(0x743b7ec1c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1,
0 <unfinished ...>
[pid  4147] <... brk resumed>)          = 0x62b3a3bbe000
[pid  4146] <... mmap resumed>)         = 0x743b7ec1c000
[pid  4146] close(3 <unfinished ...>
[pid  4147] mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0 <unfinished ...>
[pid  4146] <... close resumed>)        = 0
[pid  4146] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid  4147] <... mmap resumed>)         = 0x7a90e77e0000
[pid  4146] <... mmap resumed>)         = 0x743b7ed21000
[pid  4147] close(5)                    = 0
[pid  4146] arch_prctl(ARCH_SET_FS, 0x743b7ed21740 <unfinished ...>
[pid  4147] futex(0x7a90e77e0000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid  4146] <... arch_prctl resumed>)   = 0
[pid  4146] set_tid_address(0x743b7ed21a10) = 4146
[pid  4146] set_robust_list(0x743b7ed21a20, 24) = 0
[pid  4146] rseq(0x743b7ed220e0, 0x20, 0, 0x53053053) = 0
[pid  4146] mprotect(0x743b7ec16000, 16384, PROT_READ) = 0
[pid  4146] mprotect(0x5eaf43a49000, 4096, PROT_READ) = 0
[pid  4146] mprotect(0x743b7ed6d000, 8192, PROT_READ) = 0
[pid  4146] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid  4146] munmap(0x743b7ed24000, 58047) = 0
[pid  4146] openat(AT_FDCWD, "", O_WRONLY|O_CREAT|O_TRUNC, 0644) = -1 ENOENT (Нет такого файла или
каталога)
[pid  4146] write(2, "Cannot open file", 16Cannot open file) = 16
[pid  4146] write(2, "\n", 1
)           = 1
[pid  4146] exit_group(1)               = ?
[pid  4146] +++ exited with 1 +++
[pid  4145] <... read resumed>0x7fffc9308c5f, 1) = ? ERESTARTSYS (To be restarted if SA_RESTART is
set)
[pid  4145] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=4146, si_uid=1000, si_status=1,
si_utime=0, si_stime=0} ---
[pid  4145] read(0, exit
"e", 1)             = 1
[pid  4145] read(0, "x", 1)             = 1
[pid  4145] read(0, "i", 1)             = 1
[pid  4145] read(0, "t", 1)             = 1
[pid  4145] read(0, "\n", 1)            = 1
```

```
[pid  4145] write(1, "\320\240\320\260\320\261\320\276\321\202\320\260
\320\267\320\260\320\262\320\265\321\200\321\210\320\265\320\275\320\260."..., 33Работа завершена.
) = 33
[pid  4145] munmap(0x79a34d10c000, 1024) = 0
[pid  4145] unlink("/dev/shm/shm_example") = 0
[pid  4145] unlink("/dev/shm/sem.sem_child1") = 0
[pid  4145] unlink("/dev/shm/sem.sem_child2") = 0
[pid  4145] exit_group(0)               = ?
[pid  4145] +++ 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.