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-211Б-23

Студент: Ласточкин М.В.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 10.12.24

Москва, 2024

Постановка задачи

Вариант 17.

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процесс должны быть представлены разными программами.

Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в file1 или в file2 в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Правило фильтрации: строки длины больше 10 символов отправляются в file1, иначе в file2. Дочерние процессы удаляют все гласные из строк.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- int pipe(int *fd); – создает канал и помещает дескрипторы файла для чтения и записи в fd[0] и fd[1].
- pid_t getpid(void); – возвращает ID вызывающего процесса.
- int open(const char *__file, int __oflag, …); – используется для открытия файла для чтения, записи или и того, и другого.
- ssize_t write(int __fd, const void *__buf, size_t __n); – Записывает N байт из буфер(BUF) в файл (FD). Возвращает количество записанных байт или -1.
- void exit(int __status); – выполняет немедленное завершение программы. Все используемые программой потоки закрываются, и временные файлы удаляются, управление возвращается ОС или другой программе.
- int close(int __fd); – сообщает операционной системе об окончании работы с файловым дескриптором, и закрывает файл(FD).
- int dup2(int __fd, int __fd2); – копирует FD в FD2, закрыв FD2 если это требуется.
- int execv(const char *__path, char *const *__argv); – заменяет образ текущего процесса на образ нового процесса, определённого в пути path.
- ssize_t read(int __fd, void *__buf, size_t __nbytes); – считывает указанное количество байт из файла(FD) в буфер(BUF).
- pid_t wait(int *__stat_loc); – используются для ожидания изменения состояния процесса-потомка вызвавшего процесса и получения информации о потомке, чьё состояние изменилось.
- int shm_open(const char *name, int oflag, mode_t mode); – создает и открывает новый (или открывает уже существующий) объект разделяемой памяти POSIX.
- int shm_unlink(const char *name); – удаляется имя объекта разделяемой памяти и, как только все процессы завершили работу с объектом и отменили его распределение, очищают пространство и уничтожают связанную с ним область памяти.
- void * mmap(void *start, size_t length, int prot , int flags, int fd, off_t offset); – отражает length байтов, начиная со смещения offset файла (или другого объекта), определенного файловым дескриптором fd, в память, начиная с адреса start.

- int ftruncate(int fd, off_t length); – устанавливают длину файла с файловым дескриптором fd в length байт.
- int sem_wait(sem_t *sem); – уменьшает значение семафора на 1. Если семафор в данный момент имеет нулевое значение, то вызов блокируется до тех пор, пока либо не станет возможным выполнить уменьшение.
- int sem_post(sem_t *sem); – увеличивает значение семафора на 1.
- int sem_destroy(sem_t *sem); - уничтожает безымянный семафор, расположенный по адресу sem

Для выполнения данной лабораторной работы я изучил указанные выше системные вызовы, а также пример выполнения подобного задания.

Программа parent.c получает на вход два аргумента – два имени файла для двух дочерних процессов. Далее создаются два файла для общей памяти, в которые будут записываться строки. Создаются два семафора для каждого дочернего процесса для синхронизации работы с общей памятью.

Для каждого процесса с помощью fork() создается новый процесс. После успешного создания, родитель запускает child.c, передавая ей параметры: имя файла, в который дочерний процесс будет записывать результат, и название общей памяти и семафоров, с которыми дочерний процесс будет работать.

Родитель считывает строки с консоли, если символов в строки меньше или равно 10, то отправляется в первую область общей памяти – file2, иначе – во вторую - file1.

В child.c получаются данные, открывается файл для записи, создается общая память для обмена строчками и подключаются семафоры. После получения строчки дочерний процесс удаляет из нее все гласны.

После окончания ввода закрывает общую память и семафоры, ждем завершения дочерних процессов с помощью wait()

Код программы

client.c

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


#define SHM_NAME "/shared_memory"
```

```c
#define BUFFER_SIZE 4096

typedef struct {
    char buffer1[BUFFER_SIZE];
    char buffer2[BUFFER_SIZE];
    sem_t sem_parent;
    sem_t sem_child1;
    sem_t sem_child2;
    bool exit_flag;
} SharedMemory;

void delete_vowels(char *str) {
    if (!str) return;

    const char *vowels = "AEIOUYaeiouy";
    size_t j = 0;
    for (size_t i = 0; str[i] != '\0'; ++i) {
        if (!strchr(vowels, str[i])) {
            str[j++] = str[i];
        }
    }
    str[j] = '\0';
}

int main(int argc, char **argv) {
    if (argc != 3) {
        fprintf(stderr, "Usage: %s <client_id> <output_file>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    int shm_fd = shm_open(SHM_NAME, O_RDWR, 0600);
    if (shm_fd == -1) {
        perror("shm_open");
        exit(EXIT_FAILURE);
    }
```

```
   SharedMemory *shm = mmap(NULL, sizeof(SharedMemory), PROT_READ |
PROT_WRITE, MAP_SHARED, shm_fd, 0);
   if (shm == MAP_FAILED) {
       perror("mmap");
       exit(EXIT_FAILURE);
   }

   int output_file = open(argv[2], O_WRONLY | O_CREAT | O_TRUNC, 0600);
   if (output_file == -1) {
       perror("open");
       exit(EXIT_FAILURE);
   }

   sem_t *my_sem;
   char *my_buffer;

   if (strcmp(argv[1], "client1") == 0) {
       my_sem = &shm->sem_child1;
       my_buffer = shm->buffer1;
   } else {
       my_sem = &shm->sem_child2;
       my_buffer = shm->buffer2;
   }

   while (true) {
       sem_wait(my_sem);

       if (shm->exit_flag) break;

       delete_vowels(my_buffer);

       size_t len = strlen(my_buffer);
       if (write(output_file, my_buffer, len) == -1) {
           perror("write");
           exit(EXIT_FAILURE);
```

```
        }

        write(output_file, "\n", 1);

        sem_post(&shm->sem_parent);
    }

    close(output_file);
    munmap(shm, sizeof(SharedMemory));
    return 0;
}
```

server.c

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

#define SHM_NAME "/shared_memory"
#define BUFFER_SIZE 4096

typedef struct {
    char buffer1[BUFFER_SIZE];
    char buffer2[BUFFER_SIZE];
    sem_t sem_parent;
    sem_t sem_child1;
    sem_t sem_child2;
    bool exit_flag;
} SharedMemory;
```

```c
int main(int argc, char **argv) {
    if (argc < 3) {
        fprintf(stderr, "Usage: %s file1 file2\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    int shm_fd = shm_open(SHM_NAME, O_CREAT | O_RDWR, 0600);
    if (shm_fd == -1) {
        perror("shm_open");
        exit(EXIT_FAILURE);
    }

    if (ftruncate(shm_fd, sizeof(SharedMemory)) == -1) {
        perror("ftruncate");
        exit(EXIT_FAILURE);
    }

    SharedMemory *shm = mmap(NULL, sizeof(SharedMemory), PROT_READ |
PROT_WRITE, MAP_SHARED, shm_fd, 0);
    if (shm == MAP_FAILED) {
        perror("mmap");
        exit(EXIT_FAILURE);
    }

    sem_init(&shm->sem_parent, 1, 1);
    sem_init(&shm->sem_child1, 1, 0);
    sem_init(&shm->sem_child2, 1, 0);
    shm->exit_flag = false;

    pid_t child1 = fork();
    if (child1 == 0) {
        execlp("./client", "./client", "client1", argv[1], NULL);
        perror("execlp");
        exit(EXIT_FAILURE);
    }
```

```
pid_t child2 = fork();
if (child2 == 0) {
    execlp("./client", "./client", "client2", argv[2], NULL);
    perror("execlp");
    exit(EXIT_FAILURE);
}

char input[BUFFER_SIZE];
while (true) {
    sem_wait(&shm->sem_parent);

    printf("Input strings (press ENTER to exit): ");
    if (fgets(input, sizeof(input), stdin) == NULL) {
        perror("fgets");
        continue;
    }

    size_t len = strlen(input);
    if (len > 0 && input[len - 1] == '\n') {
        input[len - 1] = '\0';
    }

    if (input[0] == '\0') {
        shm->exit_flag = true;
        sem_post(&shm->sem_child1);
        sem_post(&shm->sem_child2);
        break;
    }

    if (strlen(input) > 10) {
        strncpy(shm->buffer1, input, BUFFER_SIZE - 1);
        shm->buffer1[BUFFER_SIZE - 1] = '\0';
        sem_post(&shm->sem_child1);
    } else {
        strncpy(shm->buffer2, input, BUFFER_SIZE - 1);
```

```
            shm->buffer2[BUFFER_SIZE - 1] = '\0';
            sem_post(&shm->sem_child2);
        }
    }

    wait(NULL);
    wait(NULL);

    sem_destroy(&shm->sem_parent);
    sem_destroy(&shm->sem_child1);
    sem_destroy(&shm->sem_child2);
    munmap(shm, sizeof(SharedMemory));
    shm_unlink(SHM_NAME);

    return 0;
}
```

Протокол работы программы

max@DESKTOP-L04A0IM:/mnt/c/Users/lasto/CLionProjects/Osi/laba3$ gcc -o server server.c
max@DESKTOP-L04A0IM:/mnt/c/Users/lasto/CLionProjects/Osi/laba3$ gcc -o client client.c

max@DESKTOP-L04A0IM:/mnt/c/Users/lasto/CLionProjects/Osi/laba3$ strace -f ./server file1.txt file2.txt
execve("./server", ["./server", "file1.txt", "file2.txt"], 0x7ffdaacb9f08 /* 27 vars */) = 0
brk(NULL) = 0x55bba0a00000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffee804fcc0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f8b2dc2b000
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=18463, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 18463, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f8b2dc26000
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) = 0x7f8b2d9fd000
mprotect(0x7f8b2da25000, 2023424, PROT_NONE) = 0
mmap(0x7f8b2da25000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f8b2da25000

```
mmap(0x7f8b2dbba000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000)
= 0x7f8b2dbba000
mmap(0x7f8b2dc13000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x215000) = 0x7f8b2dc13000
mmap(0x7f8b2dc19000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f8b2dc19000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f8b2d9fa000
arch_prctl(ARCH_SET_FS, 0x7f8b2d9fa740) = 0
set_tid_address(0x7f8b2d9faa10)         = 752
set_robust_list(0x7f8b2d9faa20, 24)     = 0
rseq(0x7f8b2d9fb0e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7f8b2dc13000, 16384, PROT_READ) = 0
mprotect(0x55bb8f6c3000, 4096, PROT_READ) = 0
mprotect(0x7f8b2dc65000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f8b2dc26000, 18463)           = 0
openat(AT_FDCWD, "/dev/shm/shared_memory", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC,
0600) = 3
ftruncate(3, 8296)                      = 0
mmap(NULL, 8296, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f8b2dc28000
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace:
Process 753 attached
, child_tidptr=0x7f8b2d9faa10) = 753
[pid   752] clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD
<unfinished ...>
[pid   753] set_robust_list(0x7f8b2d9faa20, 24) = 0
strace: Process 754 attached
[pid   753] execve("./client", ["./client", "client1", "file1.txt"], 0x7ffee804fea8 /* 27 vars */ <unfinished ...>
[pid   752] <... clone resumed>, child_tidptr=0x7f8b2d9faa10) = 754
[pid   754] set_robust_list(0x7f8b2d9faa20, 24 <unfinished ...>
[pid   752] newfstatat(1, "",  <unfinished ...>
[pid   754] <... set_robust_list resumed>) = 0
[pid   752] <... newfstatat resumed>{st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH)
= 0
[pid   752] getrandom( <unfinished ...>
[pid   754] execve("./client", ["./client", "client2", "file2.txt"], 0x7ffee804fea8 /* 27 vars */ <unfinished ...>
[pid   752] <... getrandom resumed>"\x23\x2a\xe5\x66\x79\x5d\xe6\xd1", 8, GRND_NONBLOCK) = 8
[pid   752] brk(NULL)                   = 0x55bba0a00000
[pid   752] brk(0x55bba0a21000)         = 0x55bba0a21000
[pid   752] newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0
[pid   752] write(1, "Input strings (press ENTER to ex"..., 37Input strings (press ENTER to exit): ) = 37
[pid   752] read(0,  <unfinished ...>
[pid   753] <... execve resumed>)       = 0
[pid   754] <... execve resumed>)       = 0
[pid   753] brk(NULL)                   = 0x555e6bd9c000
[pid   754] brk(NULL)                   = 0x55ab3f9b2000
[pid   753] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffcf55463a0 <unfinished ...>
[pid   754] arch_prctl(0x3001 /* ARCH_??? */, 0x7fff84875fe0) = -1 EINVAL (Invalid argument)
[pid   753] <... arch_prctl resumed>)   = -1 EINVAL (Invalid argument)
[pid   754] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
```

[pid 753] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 754] <... mmap resumed>) = 0x7fb250f0a000
[pid 753] <... mmap resumed>) = 0x7fb68abf8000
[pid 754] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid 753] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid 754] <... access resumed>) = -1 ENOENT (No such file or directory)
[pid 753] <... access resumed>) = -1 ENOENT (No such file or directory)
[pid 753] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 754] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 753] <... openat resumed>) = 3
[pid 754] <... openat resumed>) = 3
[pid 753] newfstatat(3, "", <unfinished ...>
[pid 754] newfstatat(3, "", <unfinished ...>
[pid 753] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=18463, ...}, AT_EMPTY_PATH) = 0
[pid 754] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=18463, ...}, AT_EMPTY_PATH) = 0
[pid 753] mmap(NULL, 18463, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
[pid 754] mmap(NULL, 18463, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
[pid 753] <... mmap resumed>) = 0x7fb68abf3000
[pid 754] <... mmap resumed>) = 0x7fb250f05000
[pid 753] close(3 <unfinished ...>
[pid 754] close(3 <unfinished ...>
[pid 753] <... close resumed>) = 0
[pid 754] <... close resumed>) = 0
[pid 753] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 754] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 753] <... openat resumed>) = 3
[pid 754] <... openat resumed>) = 3
[pid 753] read(3, <unfinished ...>
[pid 754] 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"..., 832) = 832
[pid 753] <... read resumed>"\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"..., 832) = 832
[pid 754] pread64(3, <unfinished ...>
[pid 753] pread64(3, <unfinished ...>
[pid 754] <... pread64 resumed>"\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 754] pread64(3, <unfinished ...>
[pid 753] <... pread64 resumed>"\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 754] <... pread64 resumed>"\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"..., 48, 848) = 48
[pid 753] pread64(3, <unfinished ...>
[pid 754] pread64(3, <unfinished ...>
[pid 753] <... pread64 resumed>"\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"..., 48, 848) = 48
[pid 754] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 68, 896) = 68
[pid 753] pread64(3, <unfinished ...>
[pid 754] newfstatat(3, "", <unfinished ...>
[pid 753] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 68, 896) = 68
[pid 754] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
[pid 754] pread64(3, <unfinished ...>
[pid 753] newfstatat(3, "", <unfinished ...>
[pid 754] <... pread64 resumed>"\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 753] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
[pid 754] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>
[pid 753] pread64(3, <unfinished ...>

```
[pid   754] <... mmap resumed>)        = 0x7fb250cdc000
[pid   753] <... pread64 resumed>"\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   754] mprotect(0x7fb250d04000, 2023424, PROT_NONE <unfinished ...>
[pid   753] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>
[pid   754] <... mprotect resumed>)    = 0
[pid   753] <... mmap resumed>)        = 0x7fb68a9ca000
[pid   754] mmap(0x7fb250d04000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>
[pid   753] mprotect(0x7fb68a9f2000, 2023424, PROT_NONE <unfinished ...>
[pid   754] <... mmap resumed>)        = 0x7fb250d04000
[pid   753] <... mprotect resumed>)    = 0
[pid   754] mmap(0x7fb250e99000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1bd000 <unfinished ...>
[pid   753] mmap(0x7fb68a9f2000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>
[pid   754] <... mmap resumed>)        = 0x7fb250e99000
[pid   753] <... mmap resumed>)        = 0x7fb68a9f2000
[pid   754] mmap(0x7fb250ef2000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7fb250ef2000
[pid   753] mmap(0x7fb68ab87000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1bd000 <unfinished ...>
[pid   754] mmap(0x7fb250ef8000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid   753] <... mmap resumed>)        = 0x7fb68ab87000
[pid   754] <... mmap resumed>)        = 0x7fb250ef8000
[pid   753] mmap(0x7fb68abe0000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000 <unfinished ...>
[pid   754] close(3 <unfinished ...>
[pid   753] <... mmap resumed>)        = 0x7fb68abe0000
[pid   754] <... close resumed>)       = 0
[pid   753] mmap(0x7fb68abe6000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid   754] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
[pid   753] <... mmap resumed>)        = 0x7fb68abe6000
[pid   754] <... mmap resumed>)        = 0x7fb250cd9000
[pid   754] arch_prctl(ARCH_SET_FS, 0x7fb250cd9740 <unfinished ...>
[pid   753] close(3 <unfinished ...>
[pid   754] <... arch_prctl resumed>)  = 0
[pid   753] <... close resumed>)       = 0
[pid   754] set_tid_address(0x7fb250cd9a10 <unfinished ...>
[pid   753] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
[pid   754] <... set_tid_address resumed>) = 754
[pid   753] <... mmap resumed>)        = 0x7fb68a9c7000
[pid   754] set_robust_list(0x7fb250cd9a20, 24 <unfinished ...>
[pid   753] arch_prctl(ARCH_SET_FS, 0x7fb68a9c7740 <unfinished ...>
[pid   754] <... set_robust_list resumed>) = 0
[pid   753] <... arch_prctl resumed>)  = 0
[pid   754] rseq(0x7fb250cda0e0, 0x20, 0, 0x53053053 <unfinished ...>
[pid   753] set_tid_address(0x7fb68a9c7a10 <unfinished ...>
[pid   754] <... rseq resumed>)        = 0
[pid   753] <... set_tid_address resumed>) = 753
```

[pid 754] mprotect(0x7fb250ef2000, 16384, PROT_READ <unfinished ...>
[pid 753] set_robust_list(0x7fb68a9c7a20, 24 <unfinished ...>
[pid 754] <... mprotect resumed>) = 0
[pid 753] <... set_robust_list resumed>) = 0
[pid 753] rseq(0x7fb68a9c80e0, 0x20, 0, 0x53053053 <unfinished ...>
[pid 754] mprotect(0x55ab0ed2e000, 4096, PROT_READ) = 0
[pid 753] <... rseq resumed>) = 0
[pid 754] mprotect(0x7fb250f44000, 8192, PROT_READ) = 0
[pid 753] mprotect(0x7fb68abe0000, 16384, PROT_READ <unfinished ...>
[pid 754] prlimit64(0, RLIMIT_STACK, NULL, <unfinished ...>
[pid 753] <... mprotect resumed>) = 0
[pid 754] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid 753] mprotect(0x555e62dac000, 4096, PROT_READ) = 0
[pid 754] munmap(0x7fb250f05000, 18463) = 0
[pid 753] mprotect(0x7fb68ac32000, 8192, PROT_READ <unfinished ...>
[pid 754] openat(AT_FDCWD, "/dev/shm/shared_memory", O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished ...>
[pid 753] <... mprotect resumed>) = 0
[pid 754] <... openat resumed>) = 3
[pid 754] mmap(NULL, 8296, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0 <unfinished ...>
[pid 753] prlimit64(0, RLIMIT_STACK, NULL, <unfinished ...>
[pid 754] <... mmap resumed>) = 0x7fb250f07000
[pid 754] openat(AT_FDCWD, "file2.txt", O_WRONLY|O_CREAT|O_TRUNC, 0600 <unfinished ...>
[pid 753] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid 753] munmap(0x7fb68abf3000, 18463) = 0
[pid 753] openat(AT_FDCWD, "/dev/shm/shared_memory", O_RDWR|O_NOFOLLOW|O_CLOEXEC) = 3
[pid 753] mmap(NULL, 8296, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7fb68abf5000
[pid 753] openat(AT_FDCWD, "file1.txt", O_WRONLY|O_CREAT|O_TRUNC, 0600 <unfinished ...>
[pid 754] <... openat resumed>) = 4
[pid 754] futex(0x7fb250f09040, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 753] <... openat resumed>) = 4
[pid 753] futex(0x7fb68abf7020, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY.
<unfinished ...>
[pid 752] <... read resumed>".\n", 1024) = 2
[pid 752] futex(0x7f8b2dc2a040, FUTEX_WAKE, 1) = 1
[pid 754] <... futex resumed>) = 0
[pid 752] futex(0x7f8b2dc2a000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 754] write(4, ".", 1) = 1
[pid 754] write(4, "\n", 1) = 1
[pid 754] futex(0x7fb250f09000, FUTEX_WAKE, 1 <unfinished ...>
[pid 752] <... futex resumed>) = 0
[pid 754] <... futex resumed>) = 1
[pid 752] write(1, "Input strings (press ENTER to ex"..., 37 <unfinished ...>
Input strings (press ENTER to exit): **[pid 754] futex(0x7fb250f09040, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>**
[pid 752] <... write resumed>) = 37
[pid 752] read(0, fhdsfjhdf
"fhdsfjhdf\n", 1024) = 10
[pid 752] futex(0x7f8b2dc2a040, FUTEX_WAKE, 1) = 1

[pid 754] <... futex resumed>) = 0
[pid 752] futex(0x7f8b2dc2a000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 754] write(4, "fhdsfjhdf", 9) = 9
[pid 754] write(4, "\n", 1) = 1
[pid 754] futex(0x7fb250f09000, FUTEX_WAKE, 1 <unfinished ...>
[pid 752] <... futex resumed>) = 0
[pid 754] <... futex resumed>) = 1
[pid 752] write(1, "Input strings (press ENTER to ex"..., 37 <unfinished ...>
Input strings (press ENTER to exit): **[pid 754] futex(0x7fb250f09040, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>**
[pid 752] <... write resumed>) = 37
[pid 752] read(0, fsjfdhjsdhfjsdfh
"fsjfdhjsdhfjsdfh\n", 1024) = 17
[pid 752] futex(0x7f8b2dc2a020, FUTEX_WAKE, 1) = 1
[pid 753] <... futex resumed>) = 0
[pid 752] futex(0x7f8b2dc2a000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 753] write(4, "fsjfdhjsdhfjsdfh", 16) = 16
[pid 753] write(4, "\n", 1) = 1
[pid 753] futex(0x7fb68abf7000, FUTEX_WAKE, 1 <unfinished ...>
[pid 752] <... futex resumed>) = 0
[pid 753] <... futex resumed>) = 1
[pid 752] write(1, "Input strings (press ENTER to ex"..., 37 <unfinished ...>
Input strings (press ENTER to exit): **[pid 753] futex(0x7fb68abf7020, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>**
[pid 752] <... write resumed>) = 37
[pid 752] read(0, fdjskfhsdfkdshfjdhfjdf
"fdjskfhsdfkdshfjdhfjdf\n", 1024) = 23
[pid 752] futex(0x7f8b2dc2a020, FUTEX_WAKE, 1) = 1
[pid 753] <... futex resumed>) = 0
[pid 752] futex(0x7f8b2dc2a000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 753] write(4, "fdjskfhsdfkdshfjdhfjdf", 22) = 22
[pid 753] write(4, "\n", 1) = 1
[pid 753] futex(0x7fb68abf7000, FUTEX_WAKE, 1 <unfinished ...>
[pid 752] <... futex resumed>) = 0
[pid 753] <... futex resumed>) = 1
[pid 752] write(1, "Input strings (press ENTER to ex"..., 37 <unfinished ...>
Input strings (press ENTER to exit): [pid 753] futex(0x7fb68abf7020, **FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>**
[pid 752] <... write resumed>) = 37
[pid 752] read(0, kfdjsfdshfjsdhfjdhfjd
"kfdjsfdshfjsdhfjdhfjd\n", 1024) = 22
[pid 752] futex(0x7f8b2dc2a020, FUTEX_WAKE, 1) = 1
[pid 753] <... futex resumed>) = 0
[pid 752] futex(0x7f8b2dc2a000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 753] write(4, "kfdjsfdshfjsdhfjdhfjd", 21) = 21
[pid 753] write(4, "\n", 1) = 1

[pid 753] futex(0x7fb68abf7000, FUTEX_WAKE, 1 <unfinished ...>
[pid 752] <... futex resumed>) = 0
[pid 753] <... futex resumed>) = 1
[pid 752] write(1, "Input strings (press ENTER to ex"..., 37 <unfinished ...>
Input strings (press ENTER to exit): [pid 753] futex(0x7fb68abf7020,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 752] <... write resumed>) = 37
[pid 752] read(0, ejasgagafgafgafg
"ejasgagafgafgafg\n", 1024) = 17
[pid 752] futex(0x7f8b2dc2a020, FUTEX_WAKE, 1) = 1
[pid 753] <... futex resumed>) = 0
[pid 752] futex(0x7f8b2dc2a000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 753] write(4, "jsggfgfgfg", 10) = 10
[pid 753] write(4, "\n", 1) = 1
[pid 753] futex(0x7fb68abf7000, FUTEX_WAKE, 1 <unfinished ...>
[pid 752] <... futex resumed>) = 0
[pid 753] <... futex resumed>) = 1
[pid 752] write(1, "Input strings (press ENTER to ex"..., 37Input strings (press ENTER to exit): <unfinished ...>
[pid 753] futex(0x7fb68abf7020, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 752] <... write resumed>) = 37
[pid 752] read(0, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"..., 1024) = 33
[pid 752] futex(0x7f8b2dc2a020, FUTEX_WAKE, 1) = 1
[pid 753] <... futex resumed>) = 0
[pid 752] futex(0x7f8b2dc2a000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 753] write(4, "", 0) = 0
[pid 753] write(4, "\n", 1) = 1
[pid 753] futex(0x7fb68abf7000, FUTEX_WAKE, 1 <unfinished ...>
[pid 752] <... futex resumed>) = 0
[pid 753] <... futex resumed>) = 1
[pid 752] write(1, "Input strings (press ENTER to ex"..., 37 <unfinished ...>
[pid 753] futex(0x7fb68abf7020, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL, FUTEX_BITSET_MATCH_ANYInput strings (press ENTER to exit): <unfinished ...>
[pid 752] <... write resumed>) = 37
[pid 752] read(0,
"\n", 1024) = 1
[pid 752] futex(0x7f8b2dc2a020, FUTEX_WAKE, 1) = 1
[pid 753] <... futex resumed>) = 0
[pid 752] futex(0x7f8b2dc2a040, FUTEX_WAKE, 1 <unfinished ...>
[pid 753] close(4 <unfinished ...>
[pid 752] <... futex resumed>) = 1
[pid 754] <... futex resumed>) = 0
[pid 752] wait4(-1, <unfinished ...>
[pid 754] close(4 <unfinished ...>
[pid 753] <... close resumed>) = 0
[pid 753] munmap(0x7fb68abf5000, 8296 <unfinished ...>
[pid 754] <... close resumed>) = 0
[pid 753] <... munmap resumed>) = 0
[pid 754] munmap(0x7fb250f07000, 8296 <unfinished ...>

```
[pid   753] exit_group(0 <unfinished ...>
[pid   754] <... munmap resumed>)       = 0
[pid   753] <... exit_group resumed>)   = ?
[pid   754] exit_group(0)               = ?
[pid   754] +++ exited with 0 +++
[pid   753] +++ exited with 0 +++
<... wait4 resumed>NULL, 0, NULL)       = 753
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=754, si_uid=1000, si_status=0, si_utime=0,
si_stime=1} ---
wait4(-1, NULL, 0, NULL)                = 754
munmap(0x7f8b2dc28000, 8296)            = 0
unlink("/dev/shm/shared_memory")        = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

В процессе выполнения данной лабораторной работы я изучил новые системные вызовы на языке Си, которые позволяют эффективно работать с разделяемой памятью и семафорами. Освоил передачу данных между процессами через shared memory и управление доступом с использованием семафоров. Затруднений в ходе выполнения лабораторной работы не возникло, все задачи удалось успешно реализовать.