# Курсовой проект по курсу "Операционные системы"

Студент группы: М80-207Б-21, Крючков Артемий Владимирович
Контакты: artemkr2003@mail.ru
Работа выполнена: 17.09.2022
Преподаватель: Миронов Евгений Сергеевич

## Задание

Варианты
Вариант на удовлетворительно (может быть выбран студентом по собственному усмотрению):

Необходимо написать 3-и программы. Далее будем обозначать эти программы A, B, C. Программа A принимает из стандартного потока ввода строки, а далее их отправляет программе C. Отправка строк должна производится построчно. Программа C печатает в стандартный вывод, полученную строку от программы A. После получения программа C отправляет программе A сообщение о том, что строка получена. До тех пор пока программа A не примет «сообщение о получение строки» от программы C, она не может отправлять следующую строку программе C. Программа B пишет в стандартный вывод количество отправленных символов программой A и количество принятых символов программой C. Данную информацию программа B получает от программ A и C соответственно. Способ организация межпроцессорного взаимодействия выбирает студент.

## Методы и алгоритмы решения

a.c

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

#define BUF_SIZE 255
#define SHARED_MEMORY "/shm_file"
#define S_1 "/sem1"
#define S_2 "/sem2"
#define S_3 "/sem3"

int main()
{
    int fd_shm;
    char* shmem;
    char* tmp = (char*)malloc(sizeof(char) * BUF_SIZE);
    char* buf_size = (char*)malloc(sizeof(char) * 10);
```

```
    sem_t* sem1 = sem_open(S_1, O_CREAT, 0660, 0);
    sem_t* sem2 = sem_open(S_2, O_CREAT, 0660, 0);
    sem_t* sem3 = sem_open(S_3, O_CREAT, 0660, 0);

    if (sem1 == SEM_FAILED || sem2 == SEM_FAILED || sem3 == SEM_FAILED) {
        perror("Sem opening error in program 'a'\n");
        exit(1);
    }

    if ((fd_shm = shm_open(SHARED_MEMORY, O_RDWR | O_CREAT, 0660)) == -1) {
        perror("shm_open error in program 'a'\n");
        exit(1);
    }

    if (ftruncate(fd_shm, BUF_SIZE) == -1) {
        perror("ftruncate error in program 'a'\n");
        exit(-1);
    }

    shmem = (char*)mmap(NULL, BUF_SIZE, PROT_WRITE | PROT_READ, MAP_SHARED,
fd_shm, 0);
    sprintf(buf_size, "%d", BUF_SIZE);
    char* argv[] = {buf_size, SHARED_MEMORY, S_2, S_3, NULL};

    while (scanf("%s", tmp) != EOF) {
        pid_t pid = fork();

        if (pid == 0) {
            pid_t pid_1 = fork();

            if (pid_1 == 0) {
                sem_wait(sem1);
                printf("program A sent:\n");

                if (execve("./b.out", argv, NULL) == -1) {
                    perror("Could not execve in program 'a'\n");
                }
            } else if (pid_1 > 0) {
                sem_wait(sem3); // блокирует семафор

                if (execve("./c.out", argv, NULL) == -1) {
                    perror("Could not execve in program 'a'\n");
                }
            }
        } else if (pid > 0) {
            sprintf(shmem, "%s", tmp);
            sem_post(sem1); // разблокирует семафор
            sem_wait(sem2); // блокирует семафор
            printf("    \n\n");
        }
    }
```

```
    shm_unlink(SHARED_MEMORY);
    sem_unlink(S_1);
    sem_unlink(S_2);
    sem_unlink(S_3);
    sem_close(sem1);
    sem_close(sem2);
    sem_close(sem3);
    close(fd_shm);
}
```

b.c

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

int main(int argc, char const* argv[])
{
    if (argc < 2) {
        perror("args < 2 in program 'b'\n");
        exit(1);
    }

    int buf_size = atoi(argv[0]);
    char const* shared_memory_name = argv[1];
    char const* sem3_name = argv[3];
    int fd_shm;

    if ((fd_shm = shm_open(shared_memory_name, O_RDWR, 0660)) == -1) {
        perror("shm_open error in program 'b'\n");
        exit(1);
    }

    sem_t* sem3 = sem_open(sem3_name, 0, 0, 0);
    if (sem3 == SEM_FAILED) {
        perror("sem3 error in program 'b'\n");
        exit(1);
    }

    char* shmem = (char*)mmap(NULL, buf_size, PROT_WRITE | PROT_READ,
MAP_SHARED, fd_shm, 0);
    int size = strlen(shmem);

    printf("%d symbols\n", size);
    sem_post(sem3); // разблокирует семафор
}
```

c.c

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

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

int main(int argc, char* const argv[])
{
    if (argc < 2) {
        printf("args < 2 in program 'c'\n");
        return 0;
    }

    int buf_size = atoi(argv[0]);
    char const* shared_memory_name = argv[1];
    char const* sem2_name = argv[2];
    char const* sem3_name = argv[3];
    int fd_shm;

    if ((fd_shm = shm_open(shared_memory_name, O_RDWR, 0660)) == -1) {
        perror("shm_open error in program 'c'\n");
        exit(1);
    }

    sem_t* sem2 = sem_open(sem2_name, 0, 0, 0);
    sem_t* sem3 = sem_open(sem3_name, 0, 0, 0);

    if (sem2 == SEM_FAILED || sem3 == SEM_FAILED) {
        perror("sem2 || sem3 error in program 'c'\n");
        exit(1);
    }

    char* shmem = (char*)mmap(NULL, buf_size, PROT_WRITE | PROT_READ,
MAP_SHARED, fd_shm, 0);
    pid_t p = fork();

    if (p == 0) {
        printf("program C got:\n");
        if (execve("b.out", argv, NULL) == -1) {
            perror("execve error in program 'c'\n");
            exit(1);
        }
    } else if (p > 0) {
        sem_wait(sem3); // блокирует семафор
        printf("%s\n", shmem);
    }

    sem_post(sem2); // разблокирует семафор
}
```

### Strace

```
[crewch@pc build]$ strace ./a.out
```

```
execve("./a.out", ["./a.out"], 0x7ffec0eeaf00 /* 83 vars */) = 0
brk(NULL)                               = 0x55cf1e9aa000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe97ac3f50) = -1 EINVAL (Недопустимый
аргумент)
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=158627, ...}, AT_EMPTY_PATH) =
0
mmap(NULL, 158627, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f8f2c18d000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/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\0P4\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|0755, st_size=1953472, ...}, AT_EMPTY_PATH)
= 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f8f2c18b000
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, 1994384, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f8f2bfa4000
mmap(0x7f8f2bfc6000, 1421312, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) = 0x7f8f2bfc6000
mmap(0x7f8f2c121000, 356352, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x17d000) = 0x7f8f2c121000
mmap(0x7f8f2c178000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d4000) = 0x7f8f2c178000
mmap(0x7f8f2c17e000, 52880, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f8f2c17e000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f8f2bfa2000
arch_prctl(ARCH_SET_FS, 0x7f8f2c18c680) = 0
set_tid_address(0x7f8f2c18c950)         = 12665
set_robust_list(0x7f8f2c18c960, 24)     = 0
rseq(0x7f8f2c18cfa0, 0x20, 0, 0x53053053) = 0
mprotect(0x7f8f2c178000, 16384, PROT_READ) = 0
mprotect(0x55cf1e7ab000, 4096, PROT_READ) = 0
mprotect(0x7f8f2c1e5000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f8f2c18d000, 158627)          = 0
getrandom("\x6c\x94\xd5\x19\xe8\xea\xa6\x4e", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x55cf1e9aa000
brk(0x55cf1e9cb000)                     = 0x55cf1e9cb000
openat(AT_FDCWD, "/dev/shm/sem.sem1", O_RDWR|O_NOFOLLOW) = -1 ENOENT (Нет
такого файла или каталога)
getrandom("\x18\xb7\xef\x61\x7b\x3d\x76\x43", 8, GRND_NONBLOCK) = 8
newfstatat(AT_FDCWD, "/dev/shm/sem.IPsMVM", 0x7ffe97ac3bb0,
```

```
AT_SYMLINK_NOFOLLOW) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/dev/shm/sem.IPsMVM", O_RDWR|O_CREAT|O_EXCL, 0660) = 3
write(3, "\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0",
32) = 32
mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f8f2c1b3000
link("/dev/shm/sem.IPsMVM", "/dev/shm/sem.sem1") = 0
newfstatat(3, "", {st_mode=S_IFREG|0640, st_size=32, ...}, AT_EMPTY_PATH) = 0
unlink("/dev/shm/sem.IPsMVM")           = 0
close(3)                                = 0
openat(AT_FDCWD, "/dev/shm/sem.sem2", O_RDWR|O_NOFOLLOW) = -1 ENOENT (Нет
такого файла или каталога)
getrandom("\x5e\x8f\x7d\xa2\xfb\xea\xac\x72", 8, GRND_NONBLOCK) = 8
newfstatat(AT_FDCWD, "/dev/shm/sem.O7ViFc", 0x7ffe97ac3bb0,
AT_SYMLINK_NOFOLLOW) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/dev/shm/sem.O7ViFc", O_RDWR|O_CREAT|O_EXCL, 0660) = 3
write(3, "\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0",
32) = 32
mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f8f2c1b2000
link("/dev/shm/sem.O7ViFc", "/dev/shm/sem.sem2") = 0
newfstatat(3, "", {st_mode=S_IFREG|0640, st_size=32, ...}, AT_EMPTY_PATH) = 0
unlink("/dev/shm/sem.O7ViFc")           = 0
close(3)                                = 0
openat(AT_FDCWD, "/dev/shm/sem.sem3", O_RDWR|O_NOFOLLOW) = -1 ENOENT (Нет
такого файла или каталога)
getrandom("\x48\xa2\xea\xf1\xf1\xc7\x40\x2a", 8, GRND_NONBLOCK) = 8
newfstatat(AT_FDCWD, "/dev/shm/sem.QBhtBX", 0x7ffe97ac3bb0,
AT_SYMLINK_NOFOLLOW) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/dev/shm/sem.QBhtBX", O_RDWR|O_CREAT|O_EXCL, 0660) = 3
write(3, "\0\0\0\0\0\0\0\200\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0",
32) = 32
mmap(NULL, 32, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f8f2c1b1000
link("/dev/shm/sem.QBhtBX", "/dev/shm/sem.sem3") = 0
newfstatat(3, "", {st_mode=S_IFREG|0640, st_size=32, ...}, AT_EMPTY_PATH) = 0
unlink("/dev/shm/sem.QBhtBX")           = 0
close(3)                                = 0
openat(AT_FDCWD, "/dev/shm/shm_file", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC,
0660) = 3
ftruncate(3, 255)                       = 0
mmap(NULL, 255, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f8f2c1b0000
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...},
AT_EMPTY_PATH) = 0
read(0, dasdasdas
"dasdasdas\n", 1024)            = 10
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f8f2c18c950) = 12687
futex(0x7f8f2c1b2000, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0, NULL,
FUTEX_BITSET_MATCH_ANYprogram A sent:
9 symbols
program C got:
9 symbols
dasdasdas
) = 0
```

```
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...},
AT_EMPTY_PATH) = 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=12687, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
write(1, "\t\n", 2
)                       = 2
write(1, "\n", 1
)                         = 1
read(0, "", 1024)                       = 0
unlink("/dev/shm/shm_file")             = 0
unlink("/dev/shm/sem.sem1")             = 0
unlink("/dev/shm/sem.sem2")             = 0
unlink("/dev/shm/sem.sem3")             = 0
munmap(0x7f8f2c1b3000, 32)              = 0
munmap(0x7f8f2c1b2000, 32)              = 0
munmap(0x7f8f2c1b1000, 32)              = 0
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```