Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный исследовательский университет "Высшая школа экономики"
Московский институт электроники и математики
им. А. Н. Тихонова НИУ ВШЭ
Департамент компьютерной инженерии

Лабораторная работа № 7
по теме:
«Семафоры»

По дисциплине

«Операционные системы»

Группа: **БПМ 243**

Студент: **Николаев Степан Денисович**

Москва 2026

Цель работы

Изучить семафоры System V и добавить их в программы из практической работы 6, чтобы доступ к разделяемой памяти был синхронизирован.

Ход работы

Все программы собирал и запускал в виртуальной машине с Ubuntu в UTM под обычным пользователем marmos, компилятор gcc с ключами -Wall -Wextra, предупреждений нет.

В писателе кроме сегмента (ключ 0x4F5306) создаётся набор из одного семафора (ключ 0x4F5307) вызовом semget. Начальное значение 1 задаётся через semctl с командой SETVAL; union semun в Linux приходится описывать самому. Функции semaphore_wait и semaphore_signal — это semop со значением -1 и +1. Писатель захватывает семафор перед записью в память и освобождает после неё.

Читатель находит сегмент и семафор по ключам, перед чтением вызывает semaphore_wait, после — semaphore_signal. В конце он удаляет и сегмент, и набор семафоров (IPC_RMID), чтобы ничего не оставалось в системе.

```
marmos@marmos-QEMU-Virtual-Machine:~/practice6$ cd ~/practice7
gcc -Wall -Wextra -o writer writer.c
gcc -Wall -Wextra -o reader reader.c
./writer
ipcs -m -s | grep -E "key|0x004f530[67]"
./reader > out.txt
head -n 4 out.txt
tail -n 2 out.txt
ipcs -m -s | grep -E "key|0x004f530[67]"
Писатель: семафор захвачен, пишу в память
Писатель: записано 2113 байт, семафор освобождён
key        shmid      owner      perms      bytes      nattch     status
0x004f5306 6          marmos     666        2114       0
key        semid      owner      perms      nsems
0x004f5307 0          marmos     666        1
Читатель: жду семафор
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
}
--- Читатель: прочитано 2113 байт ---
key        shmid      owner      perms      bytes      nattch     status
key        semid      owner      perms      nsems
marmos@marmos-QEMU-Virtual-Machine:~/practice7$
```

Рис. 1. Обычный запуск: писатель, затем читатель.

После писателя в ipcs видны и сегмент (2114 байт), и семафор. Читатель вывел текст, а в первой строке файла out.txt он записал «Читатель: жду семафор». После читателя оба списка пустые.

При последовательном запуске семафор ничего не блокирует, поэтому, чтобы проверить его работу, добавил в писатель необязательный параметр — паузу в секундах внутри критической секции. Запустил ./writer 3 в фоне и через полсекунды читателя.

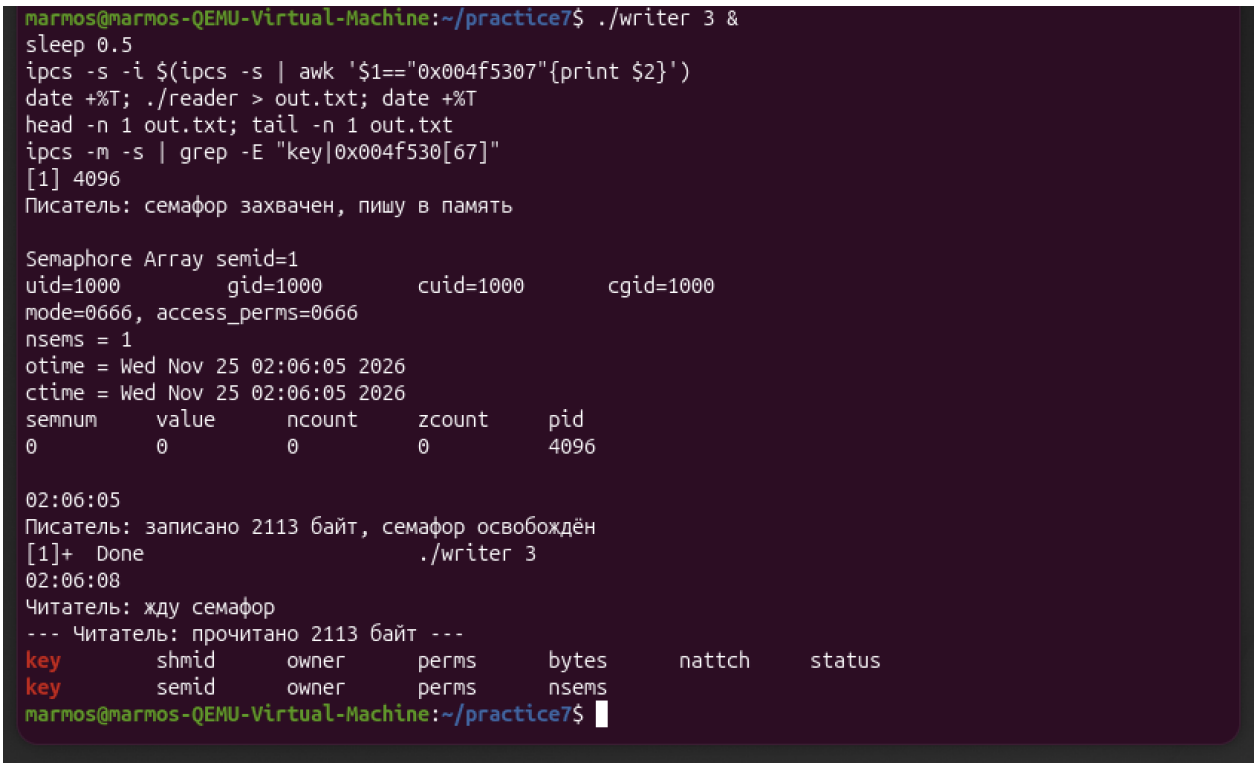

Рис. 2. Читатель ждёт, пока писатель держит семафор.

Пока писатель спит, значение семафора равно 0, а в поле pid стоит PID писателя 4096 (ipcs -s -i). Читатель стартовал в 02:06:05 и завершился в 02:06:08: он стоял на semop, пока писатель не напечатал «семафор освобождён», и только потом прочитал все 2113 байт.

Ограничение: семафор со значением 1 даёт взаимное исключение, но не задаёт порядок «сначала писатель». Если запустить читателя до писателя, shmget не найдёт сегмент и программа завершится с ошибкой. Для строгого порядка нужен второй семафор или другая схема.

Вывод. Добавил в программы из ПР6 семафор System V (semget, semop, semctl). Читатель теперь не читает данные, пока писатель не закончил запись, а гонки при одновременном доступе исключены. Все IPC-объекты удаляются читателем.

Приложение writer.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/sem.h>

#define SHM_KEY 0x4F5306
#define SEM_KEY 0x4F5307

union semun {
    int val;
    struct semid_ds *buf;
    unsigned short *array;
};

static void semaphore_wait(int semid) {
    struct sembuf op = {0, -1, 0};
    if (semop(semid, &op, 1) == -1) {
        perror("semop wait");
        exit(1);
    }
}

static void semaphore_signal(int semid) {
    struct sembuf op = {0, 1, 0};
    if (semop(semid, &op, 1) == -1) {
        perror("semop signal");
        exit(1);
    }
}

// Необязательный аргумент - пауза в секундах в критической секции (для демо)
int main(int argc, char *argv[]) {
    int delay = argc > 1 ? atoi(argv[1]) : 0;

    int fd = open("writer.c", O_RDONLY);
    if (fd == -1) {
        perror("open");
        return 1;
    }
    off_t size = lseek(fd, 0, SEEK_END);
    lseek(fd, 0, SEEK_SET);

    int shmid = shmget(SHM_KEY, size + 1, IPC_CREAT | 0666);
    if (shmid == -1) {
        perror("shmget");
        return 1;
    }
    char *mem = shmat(shmid, NULL, 0);
    if (mem == (void *)-1) {
        perror("shmat");
        return 1;
    }

    // Семафор создаётся здесь и получает начальное значение 1
    int semid = semget(SEM_KEY, 1, IPC_CREAT | 0666);
    if (semid == -1) {
        perror("semget");
        return 1;
    }
    union semun arg;
    arg.val = 1;
```

```
    if (semctl(semid, 0, SETVAL, arg) == -1) {
        perror("semctl SETVAL");
        return 1;
    }

    semaphore_wait(semid);
    printf("Писатель: семафор захвачен, пишу в память\n");
    fflush(stdout);
    sleep(delay);
    ssize_t n = read(fd, mem, size);
    if (n == -1) {
        perror("read");
        return 1;
    }
    mem[n] = '\0';
    close(fd);
    printf("Писатель: записано %zd байт, семафор освобождён\n", n);
    semaphore_signal(semid);

    shmdt(mem);
    return 0;
}
```

Приложение reader.c:

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

#define SHM_KEY 0x4F5306
#define SEM_KEY 0x4F5307

static void semaphore_wait(int semid) {
    struct sembuf op = {0, -1, 0};
    if (semop(semid, &op, 1) == -1) {
        perror("semop wait");
        exit(1);
    }
}

static void semaphore_signal(int semid) {
    struct sembuf op = {0, 1, 0};
    if (semop(semid, &op, 1) == -1) {
        perror("semop signal");
        exit(1);
    }
}

int main() {
    int shmid = shmget(SHM_KEY, 0, 0666);
    if (shmid == -1) {
        perror("shmget");
        return 1;
    }
    int semid = semget(SEM_KEY, 1, 0666);
    if (semid == -1) {
        perror("semget");
        return 1;
    }

    struct shmid_ds info;
    if (shmctl(shmid, IPC_STAT, &info) == -1) {
        perror("shmctl IPC_STAT");
        return 1;
    }
    char *mem = shmat(shmid, NULL, SHM_RDONLY);
    if (mem == (void *)-1) {
        perror("shmat");
        return 1;
    }

    printf("Читатель: жду семафор\n");
    fflush(stdout);
    semaphore_wait(semid);
    write(STDOUT_FILENO, mem, info.shm_segsz - 1);
    printf("--- Читатель: прочитано %zu байт ---\n", (size_t)info.shm_segsz - 1);
    semaphore_signal(semid);

    shmdt(mem);
    shmctl(shmid, IPC_RMID, NULL);
    semctl(semid, 0, IPC_RMID);
    return 0;
}
```