Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Курсовой проект по курсу
«Операционные системы»

Студент: Матвеев Данил
Группа: М8О-207Б-21
Вариант: на
удовлетворительно
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2023

Содержание

2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

Репозиторий

https://github.com/MrDenli/OsLabs

Постановка задачи

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.

Общие сведения о программе

sem_open() - инициализируем и открываем семафор

(char *)mmap(0, mapsize, PROT_READ | PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0) — создаём memory map для передачи информации между процессами, настраивая определёнными флагами

munmap(mapped, mapsize) — закрытие memory map

sem_close(semaphore) — отключение семафора

sem_unlink(sem_file) — закрываем семафор по данному имени

Общий метод и алгоритм решения

Метод и алгоритм решения несложен. Создаем файлы a.c, b.c и c.c .

a.c — получает от пользователя строки и отправляет с помощью семофора данные в програму C.

b.c — выводит пользователю полученные данные и отправляет так же по симофору в программу A данные.

c.c — взаимодействуя, между програмами A и B выводит длину строки.

Исходный код

a.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

```
#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) {
```

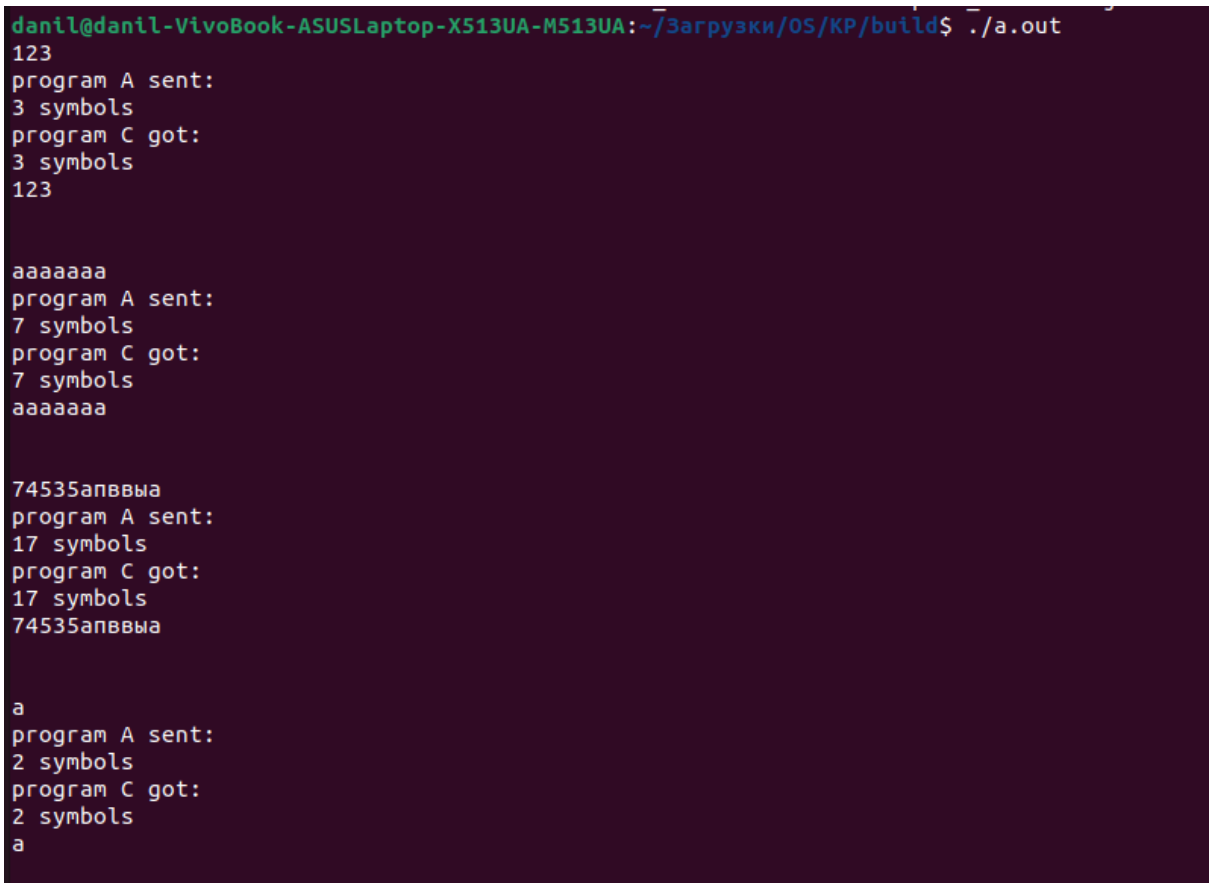
```
        perror("sem3 error in program 'b'\n");
        exit(1);
    }

    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

```
#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) {
        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
}

}
```

Демонстрация работы программы



Выводы

Мне понравилась данная лабораторная работа, мы уже работали в рамках курса с организацией межпроцессорными взаимодействиями и эта курсовая работа отличное ее завершение, чтобы закрепить полученные знания, которые многовероятно пригодятся нам в будущем.