Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Операционные системы»
III Семестр

Студент:	Анисимов В.А.
Группа:	М80-208Б-18
Преподаватель:	Миронов Е.С
Оценка:	
Дата:	

Москва 2019

1. Описание

Необходимо разработать 3 программы. Назовём их A, B и C. A принимает из стандартного потока ввода строки, а далее передаёт их программе C. Отправка строк должна производиться 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 получении строки» от программы C, она не может отправлять следующую строку программе C. Программа B пишет в стандартный поток вывода количество отправленных символов программой A и количество принятых символов программой C. Данную информацию программа B получает от программ A и C соответственно.

2. Алгоритм

Используются семафоры, разделяемая память и отображение файла в память.

1. A создаёт семафоры, разделяемую память и отображает 100 байт в память. Затем считывает строку со стандартного ввода и создаёт дочерний процесс.
2. Дочерний процесс создаёт ещё один процесс, вызывающий программу B, которая подсчитывает длину отправленной строки.
3. Следом выполняется C, которая также вызывает B.
4. B подсчитывает длину строки, полученную C.
5. A дожидается завершения дочерних процессов, после считывает новую строку из стандартного потока ввода.

3. Исходный код

a.c

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

#define BUF_SIZE 100
#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 |
O_EXCL, 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);
          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);
    sem_wait(sem2);
    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);
```

b.c

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

c.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>
#include <semaphore.h>
#include <sys/mman.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);
    printf("%s\n", shmem);
  }
  sem_post(sem2);
```

4. Работа программы

```
walien@PC-name:~/2kurs/OS/kp_3$ ./a.out
kek
program A sent:
3 symbols
program C got:
3 symbols
kek
```

```
lol
program A sent:
3 symbols
program C got:
3 symbols
lol
```

```
a-lot-of-letters
program A sent:
```

16 symbols
program C got:
16 symbols
a-lot-of-letters

5. Вывод

Благодаря знания, полученным в ходе изучения курса «Операционные системы», была написана многопроцессорная программа, использующая отображение файлов в память, общую память и семафоры. Данные довольно удобные инструменты являются одними из основных при написании многопроцессорных программ. Тема курсового проекта интересна и несложна в реализации, кроме того, она обобщает полученные в течение курса знания, закрепляя их в памяти.