# Министерство науки и высшего образования РФ

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

# «Московский Авиационный Институт»

**Национальный Исследовательский Университет**

**Институт** №8 «Информационные технологии и прикладная математика»

**Кафедра** 806 «Вычислительная математика и программирование»

# Курсовой проект

**по курсу «Операционные системы»**

|  |  |
| --- | --- |
| Студент: | Гуликов К.А. |
| Группа: | М8О-206Б-20 |
| Преподаватель: | Миронов Е. С. |
| Подпись: |  |
| Оценка: |  |
| Дата: |  |

Москва, 2022

# Содержание

1. Цель работы;
2. Постановка задачи;
3. Общий метод и алгоритм решения;
4. Код программ;
5. Демонстрация работы программы;
6. Вывод.

# Цель работы

1. Приобретение практических навыков в использовании знаний, полученных в течении курса;
2. Проведение исследования в выбранной предметной области.

# 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 С. Отправка строк должна производиться построчно. Программа C печатает в стандартный вывод, полученную строку от программы A. После получения программа C отправляет программе А сообщение о том, что строка получена. До тех пор, пока программа А не примет «сообщение о получение строки» от программы С, она не может отправлять следующую строку программе С. Программа B пишет в стандартный вывод количество отправленных символов программой А и количество принятых символов программой С. Данную информацию программа B получает от программ A и C соответственно. Способ организация межпроцессорного взаимодействия выбирает студент.

# Общий алгоритм решения

Используются семафоры, разделяемая память и отображение файла в память.

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

# Код программ

konstantin@LAPTOP-44CRFC1U:~/kp/os/33$ cat 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 255 //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");

//printf("%s", tmp);

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); close(fd\_shm);

}

konstantin@LAPTOP-44CRFC1U:~/kp/os/33$ cat 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); sem\_post(sem3);

}

konstantin@LAPTOP-44CRFC1U:~/kp/os/33$ cat 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);

}

# Демонстрация работы программы

konstantin@LAPTOP-44CRFC1U:~/kp/os/33$ ./a.out aaaaaaaaaaaaaaaaaaaaaaa

program A sent:

23 symbols program C got:

23 symbols aaaaaaaaaaaaaaaaaaaaaaa

adfjhgvh rkgjrkt program A sent: 8 symbols program C got: 8 symbols adfjhgvh

program A sent: 7 symbols program C got: 7 symbols rkgjrkt

sdsd

program A sent: 4 symbols program C got: 4 symbols

sdsd

s

program A sent: 1 symbols program C got: 1 symbols

s

s

program A sent: 1 symbols program C got: 1 symbols

s

# Выводы

Благодаря умениям, полученным в ходе курса по операционным системам, была написана многопроцессорная программа, которая использует отображение файла в память, общую память и семафоры. Эти инструменты довольно удобны и часто используются при написании многопроцессорных

программ. Данный курсовой проект прост для понимания и несложен в реализации, кроме того, он обобщает полученные во время изучения курса знания, закрепляя их в памяти.