Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовая работа по курсу**

**«Операционные системы»**

Группа: М80 – 201Б-19
Студент: Цыкин И.А.
Преподаватель: Миронов Е.С.
Оценка: _____
Дата: _____

Москва, 2020.

**Содержание**

## 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 стандартый 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 принятыхсимволов программой C. Данную информацию программа B получает от программ A и C соответственно.Способ организация межпроцессорного взаимодействия выбирает студент.

## Общие сведения о программе

Программа состоит из 3 файлов a.c, которая считывает строку и запускает сначала сначала программу b.c, которая подсчитывает количество символов полученных от пользователя и отправленных в разделяемую память, а затем программу c.c, которая прочитывает содержимое и запускает процесс b.c, который снова подсчитывает число символов в полученной строке, а затем программа c.c выводит содержимое. Все это время программа a.c ждет завершения работы программы c.c, а затем можно снова вводить строки.

Основные функции:

1. **fork** – для создания дочернего процесса.
2. **int execv(const char ****path***, char *const *argv*[]);** - замена образа памяти процесса
3. **int shm_open(const char ****name***, int *oflag*, mode_t *mode*);** - создает и открывает новый (или открывает уже существующий) объект разделяемой памяти POSIX. Объект разделяемой памяти POSIX - это обработчик, используемый несвязанными процессами для исполнения **mmap** на одну область разделяемой памяти(mode - **S_IRUSR** — чтения для владельца; **S_IWUSR** — запись для владельца)
4. **int shm_unlink(const char ****name***);** - выполняет обратную операцию, удаляя объект, предварительно созданный с помощью **shm_open**.
5. **void * mmap(void ****start***, size_t *length*, int *prot* , int *flags*, int *fd*, off_t *offset*);** - отражает *length* байтов, начиная со смещения *offset* файла (или

другого объекта), определенного файловым описателем *fd*, в память, начиная с адреса *start*. Последний параметр (адрес) необязателен, и обычно бывает равен 0.(**PROT_READ** - данные можно читать, **MAP_SHARED** - Запись информации в эту область памяти будет эквивалентна записи в файл)

6 **int munmap(void ****start***, size_t *length*);** - отключения отображения объекта в адресное пространство процесса

7 **int ftruncate(int** *fd*, **off_t** *length*); - устанавливают длину обычного файла с именем *path* или файловым дескриптором *fd* в *length* байт

8 **size_t strlen( const char * string ); -** выводит длину строки

## Общий метод и алгоритм решения.

С помощью функции fork() в программе a создаем дочерний процесс, в котором создаем 2 дочерний процесс, который запускает b.c, а первый дочерний процесс ждет выполнения 2 дочернего процесса, который запускает программу c.c, а главный родительский процесс жлет завершения. В свою очередь процесс c.c тоже создает дочерний процесс, который запускает b.c, а затем дожедается и выводит значения.

## Листинг программы

**a.c**

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

#define MEMORY_NAME "shm"
#define N 256

int main(int argc, char* argv[]){
      int status;
      char str[N];
      printf("write 0 to stop programm\n");
      for(;;){
            printf("Enter string: ");
            scanf("%s", str);
            if(str[0] == '0'){
                  return 0;
            }
            char* arg1[] = {NULL};
```

```
            char* arg2[] = {str, NULL};

            int fd = shm_open(MEMORY_NAME, O_CREAT | O_RDWR, 00600);
            if(fd == -1){
                  printf("Error with creating shared memory\n");
                  return -1;
            }

            if(ftruncate(fd, N) == -1){
                  printf("Error with rising\n");
                  return -2;
            }

            char* ptr = mmap(0, sizeof(str), PROT_READ | PROT_WRITE,
MAP_SHARED, fd, 0);
            if(ptr == MAP_FAILED){
                  printf("1. Error mmap\n");
                  return -3;
            }
            memcpy(ptr, str, sizeof(str));
            munmap(ptr, sizeof(str));
            close(fd);

            pid_t id = fork();

            if(id == 0){
                  pid_t ch_id = fork();
                  if(ch_id == 0){
                        execv("./b", arg2);
                  } else{
                        if(waitpid(ch_id, &status, 0) == -1){
                              printf("Error wait child process\n");
                              return -1;
                        }
                        if (WEXITSTATUS(status) != 0) {
                              return -2;
                        }
                        printf("was sent by process A\n");
                        execv("./c", arg1);
                  }
            }else{
                  if(waitpid(id, &status, 0) == -1){
                        printf("Error wait child process\n");
                        return -1;
                  }
                  if (WEXITSTATUS(status) != 0) {
                        return -2;
                  }
                  shm_unlink(MEMORY_NAME);
            }
      }
```

```
	return 0;
}
```

## b.c

```
#include <stdio.h>

#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main(int argc, char* argv[]){
	int n = strlen(argv[0]);
	printf("%d symbols ", n);
	return 0;
}
```

## c.c

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

#define MEMORY_NAME "shm"
#define N 256

int main(int argc, char* argv[]){
	int status;
	int fd = shm_open(MEMORY_NAME, O_RDWR, 0);
	if(fd == -1){
		printf("Error with creating shared memory\n");
		return -1;
	}

	struct stat file_st;
	if(fstat(fd, &file_st) == -1){
		printf("Error fstat\n");
		return -3;
	}

	char* ptr = mmap(NULL, file_st.st_size, PROT_READ, MAP_SHARED, fd, 0);
	if(ptr == MAP_FAILED){
		printf("Error mmap\n");
		return -3;
	}
```

```
	char* arg[] = {ptr, NULL};

	pid_t id = fork();
	if(id == 0){
		execv("./b", arg);
	} else{
		if(waitpid(id, &status, 0) == -1){
			printf("Error wait child process\n");
			return -1;
		}
		if (WEXITSTATUS(status) != 0) {
			return -2;
		}
		printf("was get by process C\n");
		printf("Result: %s\n", ptr);
		munmap(ptr, file_st.st_size);
		close(fd);
	}
	return 0;
}
```

## Результаты работы программы

```
vaney@V-box:~$ cd */os_kp
vaney@V-box:~/Examples/os_kp$ gcc -o a a.c -lrt
vaney@V-box:~/Examples/os_kp$ gcc -o b b.c
vaney@V-box:~/Examples/os_kp$ gcc -o c c.c -lrt
vaney@V-box:~/Examples/os_kp$ ./a
write 0 to stop programm
Enter string: hello
5 symbols was sent by process A
5 symbols was get by process C
Result: hello
Enter string: my
2 symbols was sent by process A
2 symbols was get by process C
Result: my
Enter string: name
4 symbols was sent by process A
4 symbols was get by process C
Result: name
Enter string: is
2 symbols was sent by process A
2 symbols was get by process C
Result: is
Enter string: Ivan
4 symbols was sent by process A
4 symbols was get by process C
Result: Ivan
Enter string: qwertyuiopasdfghjklzxcvbnm
26 symbols was sent by process A
26 symbols was get by process C
Result: qwertyuiopasdfghjklzxcvbnm
Enter string: 0
```

# Strace

```
vaney@V-box:~/Examples/os_kp$ strace ./a
execve("./a", ["./a"], 0x7ffd6c923d50 /* 58 vars */) = 0
brk(NULL)                               = 0x55c08eb93000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffdfe0c42b0) = -1 EINVAL (Invalid
argument)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or
directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=67999, ...}) = 0
mmap(NULL, 67999, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f451e6c6000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/librt.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 7\0\0\0\0\0\0"...,
832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=40040, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f451e6c4000
mmap(NULL, 44000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f451e6b9000
mprotect(0x7f451e6bc000, 24576, PROT_NONE) = 0
mmap(0x7f451e6bc000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x3000) = 0x7f451e6bc000
mmap(0x7f451e6c0000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x7000) = 0x7f451e6c0000
mmap(0x7f451e6c2000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x8000) = 0x7f451e6c2000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\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
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\
0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\363\377?\332\200\270\27\304d\245n\
355Y\377\t\334"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0
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
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\
0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\363\377?\332\200\270\27\304d\245n\
355Y\377\t\334"..., 68, 880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f451e4c7000
mprotect(0x7f451e4ec000, 1847296, PROT_NONE) = 0
mmap(0x7f451e4ec000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x25000) = 0x7f451e4ec000
mmap(0x7f451e664000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x19d000) = 0x7f451e664000
mmap(0x7f451e6af000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1e7000) = 0x7f451e6af000
```

```
mmap(0x7f451e6b5000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f451e6b5000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpthread.so.0", O_RDONLY|O_CLOEXEC)
= 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\
0\1\0\0\0\220\201\0\0\0\0\0\0"..., 832) = 832
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O\305\3743\364B\2216\244\224\306@\
261\23\327o"..., 68, 824) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=157224, ...}) = 0
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O\305\3743\364B\2216\244\224\306@\
261\23\327o"..., 68, 824) = 68
mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f451e4a4000
mmap(0x7f451e4ab000, 69632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x7000) = 0x7f451e4ab000
mmap(0x7f451e4bc000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x18000) = 0x7f451e4bc000
mmap(0x7f451e4c1000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1c000) = 0x7f451e4c1000
mmap(0x7f451e4c3000, 13432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f451e4c3000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f451e4a1000
arch_prctl(ARCH_SET_FS, 0x7f451e4a1740) = 0
mprotect(0x7f451e6af000, 12288, PROT_READ) = 0
mprotect(0x7f451e4c1000, 4096, PROT_READ) = 0
mprotect(0x7f451e6c2000, 4096, PROT_READ) = 0
mprotect(0x55c08d769000, 4096, PROT_READ) = 0
mprotect(0x7f451e704000, 4096, PROT_READ) = 0
munmap(0x7f451e6c6000, 67999)           = 0
set_tid_address(0x7f451e4a1a10)         = 3653
set_robust_list(0x7f451e4a1a20, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7f451e4abbf0, sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f451e4b93c0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7f451e4abc90, sa_mask=[],
sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f451e4b93c0},
NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
brk(NULL)                               = 0x55c08eb93000
brk(0x55c08ebb4000)                     = 0x55c08ebb4000
write(1, "write 0 to stop programm\n", 25write 0 to stop programm
) = 25
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(1, "Enter string: ", 14Enter string: )          = 14
read(0, hello
"hello\n", 1024)                = 6
statfs("/dev/shm/", {f_type=TMPFS_MAGIC, f_bsize=4096, f_blocks=382937,
f_bfree=377224, f_bavail=377224, f_files=382937, f_ffree=382855,
f_fsid={val=[0, 0]}, f_namelen=255, f_frsize=4096, f_flags=ST_VALID|
ST_NOSUID|ST_NODEV}) = 0
futex(0x7f451e4c6390, FUTEX_WAKE_PRIVATE, 2147483647) = 0
```

```
openat(AT_FDCWD, "/dev/shm/shm", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0600) =
3
ftruncate(3, 256)                       = 0
mmap(NULL, 256, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f451e703000
munmap(0x7f451e703000, 256)             = 0
close(3)                                = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
SIGCHLD, child_tidptr=0x7f451e4a1a10) = 3654
wait4(3654, 5 symbols was sent by process A
5 symbols was get by process C
Result: hello
[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 3654
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=3654, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
unlink("/dev/shm/shm")                  = 0
write(1, "Enter string: ", 14Enter string: )          = 14
read(0, 0
"0\n", 1024)                    = 2
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
+++ exited with 0 +++
```

## Вывод

Выполняя данный курсовой проект, мне в очередной раз удалось попрактиковаться в знаниях, полученных на протяжени всего семестра. Полученный знания в двльнейшем очень сильно помогут во многих работах.