Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу

«Операционные системы»

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

Москва, 2020.

Содержание

1 Постановка задачи
2 Общие сведения о программе
3 Общий метод и алгоритм решения
4 Листинг программы
5 Результаты работы программы
6 Strace
7 Вывод

Постановка задачи

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы (memory-mapped files).

Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

6 вариант) В файле записаны команды вида: «число числочисло<endline>». Дочерний процесс считает их сумму и выводит результат в стандартный поток вывода. Числа имеют тип int

Общие сведения о программе

Программа компилируется из двух файлов child.c и main.c. Для работы программы необходимо использование библиотеку sys/mman.h. Библиотека sys/mman.h служит для управления памяти.

Основные функции:

1. **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)
2. **int shm_unlink(const char **name*);** - выполняет обратную операцию, удаляя объект, предварительно созданный с помощью **shm_open**.
3. **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)
4. **int munmap(void **start*, size_t *length*);** - отключения отображения объекта в адресное пространство процесса

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

Общий метод и алгоритм решения.

Изначально родительский процесс создает дочерний процесс. Дочерний процесс должен открыть файл в разделяемой памяти и расширить его до необходимого значения, а также отобразить в своё адресное пространство. Затем дочерний процесс открывает файл тестовый файл и помещает результаты в разделяемую память, после же удаляет отображение разделяемой памятии и закрыть файловый дескриптор. Дождавшись завершения дочернего процесса, родительский процесс также как и дочерний открывает и отображает файл разделяемой памяти. Затем выводит результаты и удаляет все файлы.

Листинг программы

child.c

```
#include <stdio.h>

#include <stdlib.h>
#include <unistd.h>
#include <sys/mman.h>
#include <fcntl.h>

#define MEMORY_NAME "file"
#define MEMORY_SIZE 4096
#define N 256

typedef struct {
	size_t size;
	int sum[N];
}Data;

void func(FILE *input,Data *data) {
	char c;
	int a, b, res, flag;
	a = 0; res =0; flag = 0;
	data->size = 0;
	for(;;) {
		c = fgetc(input);

		if (c >= '0' && c <= '9') {
			b = atoi(&c);
			a = a*10 + b;
		}else if (c == '-') {
			flag = 1;
		}else if(c == ' ') {
			if(flag){
				res-=a;
				flag = 0;
			}else{
				res+=a;
			}
			a = 0;
		}else if(c == '\n') {
			if(flag){
				res-=a;
				flag = 0;
			}else{
				res+=a;
			}
			data->sum[data->size++] = res;
			res = 0;
			a = 0;
		}else if(c == EOF){
```

```
                return;
            }
        }
}

int main(int argc, char *argv[]) {
        int fd = shm_open(MEMORY_NAME, O_EXCL | O_CREAT | O_RDWR, S_IRUSR |
S_IWUSR);
        if(fd < 0){
            printf("Error sh_open\n");
            return -2;
        }
        if(ftruncate(fd, MEMORY_SIZE)){
            printf("Error with length of file\n");
            return -3;
        }

        Data* data = mmap(NULL, MEMORY_SIZE, PROT_WRITE, MAP_SHARED, fd, 0);
        if(data == MAP_FAILED){
            printf("Error map\n");
            return -4;
        }
        FILE* input = fopen(argv[0], "r");
        if(input == NULL){
            printf("Error open file\n");
            return -5;
        }
        func(input, data);
        fclose(input);
        munmap(data, MEMORY_SIZE);
        close(fd);
        return 0;
}
```

main.c

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

#define MEMORY_NAME "file"
#define MEMORY_SIZE 4096
#define N 256

typedef struct {
        size_t size;
        int sum[N];
}Data;
```

```
int main(int argc, char *argv[]){
	int status;
	char name[N];
	printf("Enter file name: ");
	scanf("%s", name);
	char* arg[] = {name, NULL};

	pid_t id = fork();

	if (id == 0) {
		execv("./child", arg);
	}else if(id > 0){
		if(waitpid(id, &status, 0) == -1){
			printf("Error wait child process\n");
			return -1;
		}
		if (WEXITSTATUS(status) != 0) {
			return -2;
		}
		int fd = shm_open(MEMORY_NAME, O_RDONLY, S_IRUSR | S_IWUSR);
		if(fd == -1){
			printf("Error open shared memory file\n");
			return -2;
		}

		Data *data = mmap(NULL, MEMORY_SIZE, PROT_READ, MAP_SHARED, fd,
0);
		if(data == MAP_FAILED){
			printf("Error map\n");
			return -4;
		}

		for (int i = 0; i < data->size; i++) {
			printf("%d Result from child: %d\n", i + 1, data->sum[i]);
		}

		munmap(data, MEMORY_SIZE);
		shm_unlink(MEMORY_NAME);
		close(fd);
	} else{
		printf("Error fork\n");
		return -5;
	}
	return 0;
}
```

Результаты работы программы

```
vaney@V-box:~/Examples/os_lab4$ gcc -o child child.c -lrt
vaney@V-box:~/Examples/os_lab4$ gcc -o main main.c -lrt
vaney@V-box:~/Examples/os_lab4$ cat test
3 3 3
5 5 5
5 5 -10
vaney@V-box:~/Examples/os_lab4$ ./main
Enter file name: test
1 Result from child: 9
2 Result from child: 15
3 Result from child: 0
vaney@V-box:~/Examples/os_lab4$ cat test
2 2 2
3 3 3
4 4 4
5 5 5
1 1 1
-3 -3 -3
-3 5 -2
245 678 -783
vaney@V-box:~/Examples/os_lab4$ ./main
Enter file name: test
1 Result from child: 6
2 Result from child: 9
3 Result from child: 12
4 Result from child: 15
5 Result from child: 3
6 Result from child: -9
7 Result from child: 0
8 Result from child: 140
```

Strace

```
vaney@V-box:~/Examples/os_lab4$ strace ./main
execve("./main", ["./main"], 0x7fff073bbb20 /* 58 vars */) = 0
brk(NULL)                                 = 0x55cbe39cb000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe565b83a0) = -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) = 0x7f408f241000
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) =
0x7f408f23f000
mmap(NULL, 44000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f408f234000
mprotect(0x7f408f237000, 24576, PROT_NONE) = 0
mmap(0x7f408f237000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x3000) = 0x7f408f237000
mmap(0x7f408f23b000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x7000) = 0x7f408f23b000
```

```
mmap(0x7f408f23d000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x8000) = 0x7f408f23d000
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\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) =
0x7f408f042000
mprotect(0x7f408f067000, 1847296, PROT_NONE) = 0
mmap(0x7f408f067000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x25000) = 0x7f408f067000
mmap(0x7f408f1df000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x19d000) = 0x7f408f1df000
mmap(0x7f408f22a000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1e7000) = 0x7f408f22a000
mmap(0x7f408f230000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f408f230000
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\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) =
0x7f408f01f000
mmap(0x7f408f026000, 69632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x7000) = 0x7f408f026000
mmap(0x7f408f037000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x18000) = 0x7f408f037000
mmap(0x7f408f03c000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1c000) = 0x7f408f03c000
mmap(0x7f408f03e000, 13432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f408f03e000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f408f01c000
arch_prctl(ARCH_SET_FS, 0x7f408f01c740) = 0
mprotect(0x7f408f22a000, 12288, PROT_READ) = 0
mprotect(0x7f408f03c000, 4096, PROT_READ) = 0
mprotect(0x7f408f23d000, 4096, PROT_READ) = 0
```

```
mprotect(0x55cbe2aff000, 4096, PROT_READ) = 0
mprotect(0x7f408f27f000, 4096, PROT_READ) = 0
munmap(0x7f408f241000, 67999)           = 0
set_tid_address(0x7f408f01ca10)         = 7139
set_robust_list(0x7f408f01ca20, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7f408f026bf0, sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f408f0343c0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7f408f026c90, sa_mask=[],
sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f408f0343c0},
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)                               = 0x55cbe39cb000
brk(0x55cbe39ec000)                     = 0x55cbe39ec000
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(1, "Enter file name: ", 17Enter file name: )       = 17
read(0, test
"test\n", 1024)                 = 5
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
SIGCHLD, child_tidptr=0x7f408f01ca10) = 7140
wait4(7140, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 7140
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=7140, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
statfs("/dev/shm/", {f_type=TMPFS_MAGIC, f_bsize=4096, f_blocks=382937,
f_bfree=377378, f_bavail=377378, f_files=382937, f_ffree=382843,
f_fsid={val=[0, 0]}, f_namelen=255, f_frsize=4096, f_flags=ST_VALID|
ST_NOSUID|ST_NODEV}) = 0
futex(0x7f408f041390, FUTEX_WAKE_PRIVATE, 2147483647) = 0
openat(AT_FDCWD, "/dev/shm/file", O_RDONLY|O_NOFOLLOW|O_CLOEXEC) = 3
mmap(NULL, 4096, PROT_READ, MAP_SHARED, 3, 0) = 0x7f408f27e000
write(1, "1 Result from child: 6\n", 231 Result from child: 6
) = 23
write(1, "2 Result from child: 9\n", 232 Result from child: 9
) = 23
write(1, "3 Result from child: 12\n", 243 Result from child: 12
) = 24
munmap(0x7f408f27e000, 4096)            = 0
unlink("/dev/shm/file")                 = 0
close(3)                                = 0
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

Выполняя данную лабораторную работу мной была изучена идея с изображением файла в память с помощью file-mapping . Отображение файлов в память оказывается очень полезным для коммуникации между процессами. Также мной была изучена функция shm_open, так как процессы были разделены на разные файлы.