Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №3 по курсу

«Операционные системы»

Группа: М80-206Б-20

Студент: Шипилова Т.П.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 16.12.23

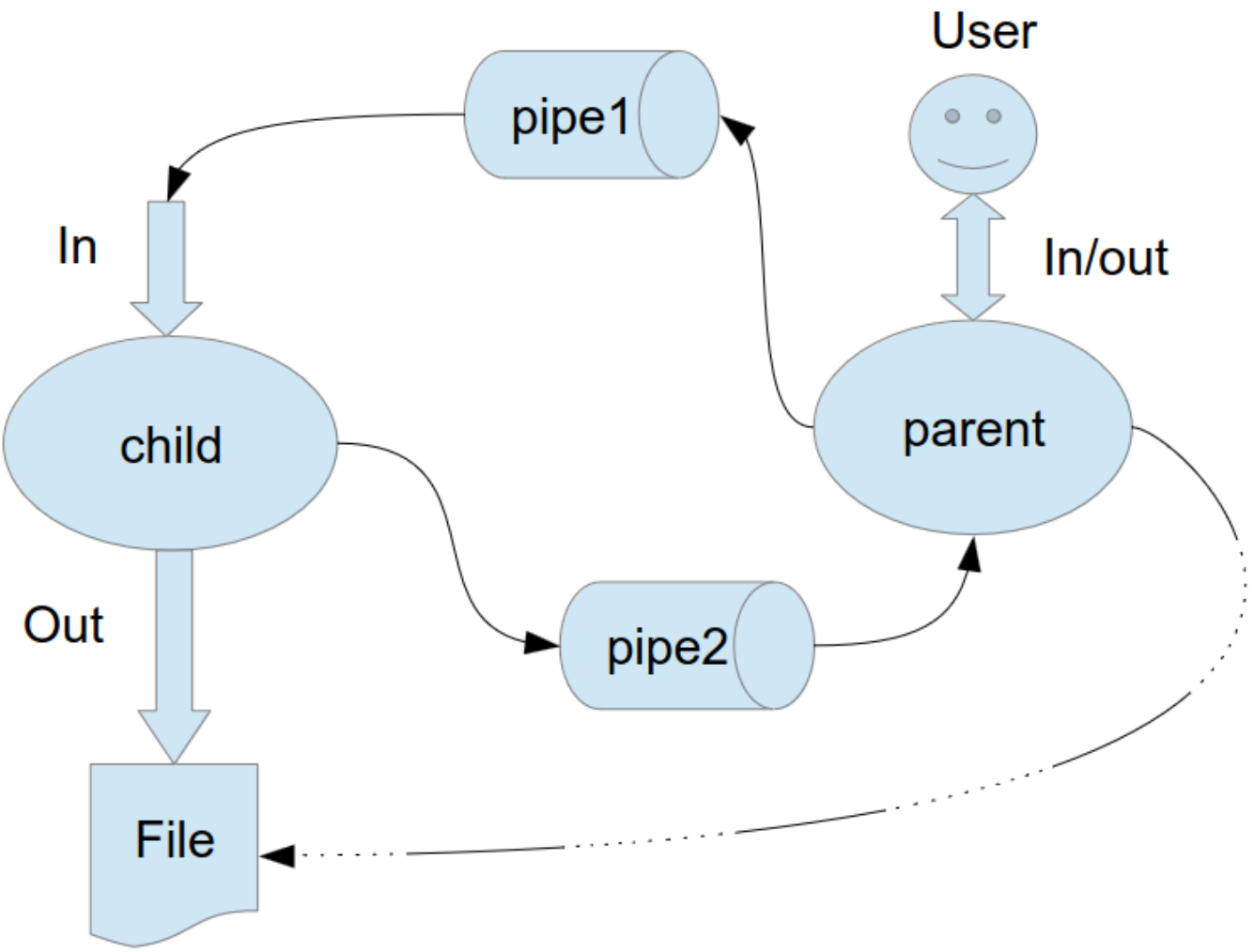
Москва, 2023

Постановка задачи

Вариант 16, группа 4.

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.

Родительский процесс создает дочерний процесс. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child проверяет строки на валидность правилу. Если строка соответствует правилу, то она выводится в стандартный поток вывода дочернего процесса, иначе в pipe2 выводится информация об ошибке. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода.



Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t **fork**(void); – создает дочерний процесс.
- int **open**(const char *pathname, int flags, mode_t mode); – используется для открытия(создания) файла.
- int **execl**(const char *path, const char *arg0, ... /* (char *) NULL */); – используется для запуска другой программы.
- int **close**(int fd); – используется для закрытия файловых дескрипторов.
- int **shm_open** (const char *__name, int __oflag, mode_t __mode); - используется для создания или открытия сегмента общей памяти.
- int **ftruncate** (int __fd, __off_t __length) - используется для изменения размера файла.
- void ***mmap** (void *__addr, size_t __len, int __prot, int __flags, int __fd, __off_t __offset) - используется для создания или открытия области памяти, которая сопоставляется с файлом или другим ресурсом
- int **shm_unlink** (const char *__name) - используется для удаления сегмента общей памяти
- int **munmap** (void *__addr, size_t __len) - используется для отсоединения области памяти, сопоставленной с помощью mmap, от файла или ресурса.

Код программы

main.cpp

```
#include "unistd.h"

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <sys/mman.h>

#include <iostream>

#include <fcntl.h>

#include <sys/wait.h>

const int SIZE_SH = 1024;

using namespace std;

int main() {

   int fd = shm_open("/myshm", O_CREAT | O_RDWR, 0666);

   if(ftruncate(fd, SIZE_SH) == -1) {

       cout << "truncate throwed\n";

       exit(1);
```

```
    }
    char* str = (char*)mmap(NULL, SIZE_SH, PROT_READ | PROT_WRITE,
MAP_SHARED, fd, 0);
    if(str == MAP_FAILED) {
        cout << "mmap failed\n";
        exit(1);
    }
    string input;
    getline(cin, input);
    strncpy(str, input.c_str(), SIZE_SH);
    // Создание дочернего процесса
    pid_t pid = fork();
    if (pid < 0){
        perror("can't create fork");
        exit(EXIT_FAILURE);
    }


    if (pid == 0) {
        // Дочерний процесс
        execl("child", "child", "/myshm", "output.txt", nullptr);
        munmap(str, SIZE_SH);
        cout << "EXECL ERROR" << endl;
        return 1;
    } else {
        // Родительский процесс
        wait(NULL);
        munmap(str, SIZE_SH);
        close(fd);
        shm_unlink("/myshm");
    }
```

```
    return 0;
}
```

child.cpp

```
#include "unistd.h"
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <sys/mman.h>
#include <iostream>
#include <fcntl.h>
#include <sys/wait.h>
const int SIZE_SH = 1024;
using namespace std;
int main(int argc, char* argv[]) {
    int fd = shm_open(argv[1], O_RDWR, 0666);

    char* str = (char*)mmap(NULL, SIZE_SH, PROT_READ | PROT_WRITE,
MAP_SHARED, fd, 0);

    // Проверка строки на соответствие правилу
    if (str[strlen(str) - 1] == '.' || str[strlen(str) - 1] == ';') {
        cout << str << endl;
    }  else {
            FILE* file = fopen(argv[2], "w");
            fprintf(file, "%s", str);
            fclose(file);
        }
    munmap(str, SIZE_SH);
    return 0;
}
```

Протокол работы программы

Тестирование:

```
tanya@tanya:~/Рабочий стол/OOS/OS3sem/3$ ./main
kjsdncskd;
kjsdncskd;
tanya@tanya:~/Рабочий стол/OOS/OS3sem/3$ ./main
dsclksdm
tanya@tanya:~/Рабочий стол/OOS/OS3sem/3$ cat output.txt
dsclksdm
```

Strace:

```
tanya@tanya:~/Рабочий стол/OOS/OS3sem/3$ strace ./main
execve("./main", ["./main"], 0x7fffc20921c0 /* 74 vars */) = 0
brk(NULL)                               = 0x5600223ce000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffebcbf3f10) = -1 EINVAL (Недопустимый аргумент)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f20b4c59000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=68035, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 68035, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f20b4c48000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.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\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2260296, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 2275520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f20b4a00000
mprotect(0x7f20b4a9a000, 1576960, PROT_NONE) = 0
mmap(0x7f20b4a9a000, 1118208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x9a000) = 0x7f20b4a9a000
mmap(0x7f20b4bab000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1ab000) = 0x7f20b4bab000
mmap(0x7f20b4c1b000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x21a000) = 0x7f20b4c1b000
mmap(0x7f20b4c29000, 10432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f20b4c29000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", 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\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=125488, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 127720, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f20b49e0000
```

```
     mmap(0x7f20b49e3000, 94208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x3000) = 0x7f20b49e3000

     mmap(0x7f20b49fa000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a000)
= 0x7f20b49fa000

     mmap(0x7f20b49fe000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1d000) = 0x7f20b49fe000

     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\0P\237\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 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48,
848) = 48

     pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\244;\374\204(\337f#\315I\214\234\f\256\271\32"..., 68, 896)
= 68

     newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2216304, ...}, AT_EMPTY_PATH) = 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

     mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f20b4600000

     mmap(0x7f20b4628000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x7f20b4628000

     mmap(0x7f20b47bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1bd000) = 0x7f20b47bd000

     mmap(0x7f20b4815000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x214000) = 0x7f20b4815000

     mmap(0x7f20b481b000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f20b481b000

     close(3)                                = 0

     openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.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\0\0\0\0\0\0\0\0"..., 832) = 832

     newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=940560, ...}, AT_EMPTY_PATH) = 0

     mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f20b48f9000

     mmap(0x7f20b4907000, 507904, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe000) = 0x7f20b4907000

     mmap(0x7f20b4983000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8a000) = 0x7f20b4983000

     mmap(0x7f20b49de000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe4000) = 0x7f20b49de000

     close(3)                                = 0

     mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f20b4c46000
```

```
arch_prctl(ARCH_SET_FS, 0x7f20b4c473c0) = 0

set_tid_address(0x7f20b4c47690)        = 12347

set_robust_list(0x7f20b4c476a0, 24)    = 0

rseq(0x7f20b4c47d60, 0x20, 0, 0x53053053) = 0

mprotect(0x7f20b4815000, 16384, PROT_READ) = 0

mprotect(0x7f20b49de000, 4096, PROT_READ) = 0

mprotect(0x7f20b49fe000, 4096, PROT_READ) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f20b4c44000

mprotect(0x7f20b4c1b000, 45056, PROT_READ) = 0

mprotect(0x56002167c000, 4096, PROT_READ) = 0

mprotect(0x7f20b4c93000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7f20b4c48000, 68035)          = 0

getrandom("\x93\xc1\xb7\x98\xc8\x96\x26\x74", 8, GRND_NONBLOCK) = 8

brk(NULL)                              = 0x5600223ce000

brk(0x5600223ef000)                    = 0x5600223ef000

futex(0x7f20b4c2977c, FUTEX_WAKE_PRIVATE, 2147483647) = 0

openat(AT_FDCWD, "/dev/shm/myshm", O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC, 0666) = 3

ftruncate(3, 1024)                     = 0

mmap(NULL, 1024, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f20b4c92000

newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...},
AT_EMPTY_PATH) = 0

read(0, hiodxjsiv;

"hiodxjsiv;\n", 1024)            = 11

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f20b4c47690) = 12372

wait4(-1, hiodxjsiv;

NULL, 0, NULL)             = 12372

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=12372, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---

munmap(0x7f20b4c92000, 1024)           = 0

close(3)                               = 0

unlink("/dev/shm/myshm")               = 0

exit_group(0)                          = ?

+++ exited with 0 +++
```

Вывод

В ходе работы научилась пользоваться отображением файла или другого ресурса на область памяти, создавать сегмент памяти без привязки к файлу. К такой памяти удобно обращаться по имени, а также динамически ее изменять, так как она является объектом ядра. Она упрощает коммуникацию процессов.