Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №1 по курсу

«Операционные системы»

Группа: М8О-211Б-23

Студент: Сергеева А. А.

Преподаватель: Бахарев В.Д. (ФИИТ)

Оценка: _____

Дата: 09.10.24

Москва, 2024

Постановка задачи

Вариант 1.

Пользователь вводит команды вида: «число число число». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс считает их сумму и выводит её в файл. Числа имеют тип int. Количество чисел может быть произвольным.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- int pipe(int *fd); – создаёт канал.
- int execv (const char * path, char *const argv[]) – вызов для замещения тела процесса, в случае успешного выполнения системного вызова виртуальное адресное, пространство процесса полностью заменяется. В path записывается абсолютный или относительный путь к исполняемой файлу для запуска, в argv массив аргументов командной строки.
- ssize_t write (int fd, const void * buf, size_t n) – записывает n бит из указанного буфера в файл, соответствующий файловому дескриптору, возвращает количество записанных байт в случае успеха, иначе -1.
- ssize_t read (int fd, void * buf, size_t nbytes) – считывает nbytes из файла, соответствующего файловому дескриптору fd в буфер buf, возвращает количество прочитанных байт, 0 – если достигнут конец файла, –1 в случае ошибки.
- int open(const char *pathname, int oflag, ... /* mode_t mode*/) – Открывает файл в соответствие с указанными модами, возвращает дескриптор файла в случае успеха, –1 в случае ошибки.
- int close (int fd) – закрывает файл.
- pid_t wait(&pstatus) – приостанавливает выполнение текущего процесса до тех пор, пока какой-либо сыновний процесс не завершит своё выполнение, либо пока в текущий процесс не поступит сигнал, который вызовет обработчик сигнала или завершит выполнение процесса, записывает статус завершенного процесса в pstatus

Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. В *posix_ipc-server.c* родительский процесс создает дочерний процесс с помощью системного вызова fork. Родительский процесс передает команды пользователя через pipe, который связан с стандартным входным потоком дочернего процесса, с помощью функции dup2. С помощью системного вызова execv, заменяет образ дочернего процесса. Происходит выполнение клиентской программы. Дочерний процесс записывает результат суммы в файл, путь к которому вводил пользователь с консоли. Происходит ожидание завершения дочернего процесса с помощью системного вызова wait. Записываем в статус то, что возвращает дочерний процесс. Анализируем статус, если -1, то выводим сообщение об ошибке.

Код программы

posix_ipc-client.c

```c
#include <stdint.h>

#include <stdbool.h>


#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <stdio.h>

#include <limits.h>


#include <string.h>

#include <ctype.h>


enum err

{

    OK,

    LONG_INT_OVERFLOW,

    NOT_NUM

};

int check_long_int(char *num, int size, long int *res)

{

    char *end_num = NULL;

    *res = strtol(num, &end_num, 10);

    if ((*res == LONG_MAX) || (*res == LONG_MIN))

    {

        return LONG_INT_OVERFLOW;

    }
```

```
    if ((*end_num != '\0') || (*res == 0 && strcmp(num, "0") != 0))
    {
        return NOT_NUM;
    }
    return 0;
}

int main(int argc, char **argv)
{
    char buf[4096];
    ssize_t bytes;
    pid_t pid = getpid();
    int32_t file = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC | O_APPEND, 0600);
    if (file == -1)
    {
        const char msg[] = "error: failed to open requested file\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }

    {
        char msg[128];
        int32_t len = snprintf(msg, sizeof(msg) - 1, "%d: Start typing lines of
text. Press 'Ctrl-D' or 'Enter' with no input to exit\n", pid);
        write(STDOUT_FILENO, msg, len);
    }
    while (bytes = read(STDIN_FILENO, buf, sizeof(buf)))
    {
        if (bytes < 0)
```

```
{
    const char msg[] = "error: failed to read from stdin\n";
    write(STDERR_FILENO, msg, sizeof(msg));
    exit(EXIT_FAILURE);
}
else if (buf[0] == '\n')
{
    break;
}
{
    int j = 0, length = 0;
    long int res = 0, sum = 0;
    char msg[33];
    char num[20];
    for (int i = 0; i < bytes / sizeof(char); ++i)
    {
        if (!isspace(buf[i]))
        {
            num[j++] = buf[i];
        }
        else
        {
            if (j != 0)
            {
                num[j] = '\0';
                switch (check_long_int(num, j, &res))
                {
                case LONG_INT_OVERFLOW:
```

```
                            length = snprintf(msg, sizeof(msg) - 1, "error:
overflow long int type\n");
                            write(STDERR_FILENO, msg, length);
                            exit(EXIT_FAILURE);
                            break;
                        case NOT_NUM:
                            length = snprintf(msg, sizeof(msg) - 1, "error:
lecsema not number\n");
                            write(STDERR_FILENO, msg, length);
                            exit(EXIT_FAILURE);
                            break;
                        case OK:
                            if (((res > 0) && (sum > LONG_MAX - res)) || ((res <
0) && (sum < LONG_MIN - res)))
                            {
                                length = snprintf(msg, sizeof(msg) - 1, "error:
overflow long int type\n");
                                write(STDERR_FILENO, msg, sizeof(msg));
                                exit(EXIT_FAILURE);
                            }
                            else
                            {
                                sum += res;
                            }
                            break;
                        }
                        j = 0;
                    }
                }
            }
```

```
            int32_t len = snprintf(msg, sizeof(msg) - 1, "%ld -- sum\n", sum);
            int32_t written = write(file, msg, len);
            if (written != len)
            {
                length = snprintf(msg, sizeof(msg) - 1, "error: failed to write to
file\n");
                write(STDERR_FILENO, msg, sizeof(msg));
                exit(EXIT_FAILURE);
            }
        }
    }
    const char term = '\0';
    write(file, &term, sizeof(term));
    close(file);
}
```

posix_ipc-server.c

```
#include <stdint.h>
#include <stdbool.h>

#include <unistd.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <stdio.h>

static char CLIENT_PROGRAM_NAME[] = "posix_ipc-client";
int main(int argc, char **argv)
{
    if (argc == 1)
    {
```

```
            char msg[1024];
            uint32_t len = snprintf(msg, sizeof(msg) - 1, "usage: %s filename\n",
argv[0]);
            write(STDERR_FILENO, msg, len);
            exit(EXIT_SUCCESS);
        }
        char progpath[1024];
        {
            ssize_t len = readlink("/proc/self/exe", progpath,
                                   sizeof(progpath) - 1);
            if (len == -1)
            {
                const char msg[] = "error: failed to read full program path\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                exit(EXIT_FAILURE);
            }
            while (progpath[len] != '/')
                --len;
            progpath[len] = '\0';
        }
        int channel[2];
        if (pipe(channel) == -1)
        {
            const char msg[] = "error: failed to create pipe\n";
            write(STDERR_FILENO, msg, sizeof(msg));
            exit(EXIT_FAILURE);
        }
        const pid_t child = fork();
        switch (child)
```

```
        {
        case -1:
        {
            const char msg[] = "error: failed to spawn new process\n";
            write(STDERR_FILENO, msg, sizeof(msg));
            exit(EXIT_FAILURE);
        }
        break;
        case 0:
        {
            pid_t pid = getpid();

            dup2(STDIN_FILENO, channel[STDIN_FILENO]);
            close(channel[STDOUT_FILENO]);
            {
                char msg[64];
                const int32_t length = snprintf(msg, sizeof(msg), "%d: I'm a child\n",
pid);
                write(STDOUT_FILENO, msg, length);
            }
            {
                char path[1050];
                snprintf(path, sizeof(path) - 1, "%s/%s", progpath,
CLIENT_PROGRAM_NAME);

                char *const args[] = {CLIENT_PROGRAM_NAME, argv[1], NULL};
                int32_t status = execv(path, args);
                if (status == -1)
                {
```

```
                    const char msg[] = "error: failed to exec into new exectuable
image\n";
                    write(STDERR_FILENO, msg, sizeof(msg));
                    exit(EXIT_FAILURE);
                }
            }
        }
        break;
        default:
        {
            pid_t pid = getpid();
            {
                char msg[64];
                const int32_t length = snprintf(msg, sizeof(msg), "%d: I'm a parent, my
child has PID %d\n", pid, child);
                write(STDOUT_FILENO, msg, length);
            }
            int child_status;
            wait(&child_status);

            if (child_status != EXIT_SUCCESS)
            {
                const char msg[] = "error: child exited with error\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                exit(child_status);
            }
        }
        break;
        }
    }
```

Протокол работы программы

Тестирование:

ali_@LAPTOP-TG8SK2OI:~/OS_2/lab_1_$ ./posix_ipc-server test2.txt

30062: I'm a parent, my child has PID 30063

30063: I'm a child

30063: Start typing lines of text. Press 'Ctrl-D' or 'Enter' with no input to exit

-1 12 45 0 -1

23 10000 3

1

ali_@LAPTOP-TG8SK2OI:~/OS_2/lab_1_$ cat test2.txt

55 -- sum

10026 -- sum

1 -- sum

ali_@LAPTOP-TG8SK2OI:~/OS_2/lab_1_$ ./posix_ipc-server test2.txt

30488: I'm a parent, my child has PID 30489

30489: I'm a child

30489: Start typing lines of text. Press 'Ctrl-D' or 'Enter' with no input to exit

100000 -1000000 9

2000000000000000000000 7 8

error: overflow long int type

error: child exited with error

ali_@LAPTOP-TG8SK2OI:~/OS_2/lab_1_$ cat test2.txt

-899991 -- sum

ali_@LAPTOP-TG8SK2OI:~/OS_2/lab_1_$ ./posix_ipc-server test2.txt

30690: I'm a parent, my child has PID 30691

30691: I'm a child

30691: Start typing lines of text. Press 'Ctrl-D' or 'Enter' with no input to exit

2661 8 1

27j 2

error: lecsema not number

error: child exited with error

ali_@LAPTOP-TG8SK2OI:~/OS_2/lab_1_$ cat test2.txt

2670 -- sum

ali_@LAPTOP-TG8SK2OI:~/OS_2/lab_1_$

Starce 1:

ali_@LAPTOP-TG8SK2OI:~/OS_2/lab_1_$ strace ./posix_ipc-server test2.txt test3.txt

execve("./posix_ipc-server", ["./posix_ipc-server", "test2.txt", "test3.txt"], 0x7fffd3211880 /* 28 vars */) = 0

brk(NULL) = 0x7fffd84b4000

arch_prctl(0x3001 /* ARCH_??? */, 0x7fffe0936f60) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f69cc240000

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

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=17675, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 17675, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f69cc24b000

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"..., 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\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 68, 896) = 68

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, 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, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f69cc010000

mprotect(0x7f69cc038000, 2023424, PROT_NONE) = 0

mmap(0x7f69cc038000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f69cc038000

mmap(0x7f69cc1cd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f69cc1cd000

mmap(0x7f69cc226000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7f69cc226000

mmap(0x7f69cc22c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f69cc22c000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f69cc000000

arch_prctl(ARCH_SET_FS, 0x7f69cc000740) = 0

set_tid_address(0x7f69cc000a10) = 28545

set_robust_list(0x7f69cc000a20, 24) = 0

rseq(0x7f69cc0010e0, 0x20, 0, 0x53053053) = -1 ENOSYS (Function not implemented)

mprotect(0x7f69cc226000, 16384, PROT_READ) = 0

mprotect(0x7f69cc291000, 4096, PROT_READ) = 0

mprotect(0x7f69cc288000, 8192, PROT_READ) = 0

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

munmap(0x7f69cc24b000, 17675) = 0

readlink("/proc/self/exe", "/home/ali_/OS_2/lab_1_/posix_ipc"..., 1023) = 39

pipe2([3, 4], 0) = 0

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD28546: I'm a child

, child_tidptr=0x7f69cc000a10) = 28546

getpid(28546: Start typing lines of text. Press 'Ctrl-D' or 'Enter' with no input to exit

) = 28545

write(1, "28545: I'm a parent, my child ha"..., 4428545: I'm a parent, my child has PID 28546

) = 44

wait4(-1, 12 45 0 -1

23 10000 3

1

[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 28546

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=28546, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

exit_group(0) = ?

+++ exited with 0 +++

Strace 2:

ali_@LAPTOP-TG8SK2OI:~/OS_2/lab_1_$ strace ./posix_ipc-server test2.txt

execve("./posix_ipc-server", ["./posix_ipc-server", "test2.txt"], 0x7fffd80abe18 /* 28 vars */) = 0

brk(NULL) = 0x7fffea2de000

```
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffff254bcc0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f2b44d80000
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=17675, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 17675, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f2b44d3b000
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\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 68, 896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, 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, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f2b44b10000
mprotect(0x7f2b44b38000, 2023424, PROT_NONE) = 0
mmap(0x7f2b44b38000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f2b44b38000
mmap(0x7f2b44ccd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f2b44ccd000
mmap(0x7f2b44d26000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7f2b44d26000
mmap(0x7f2b44d2c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f2b44d2c000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f2b44b00000
arch_prctl(ARCH_SET_FS, 0x7f2b44b00740) = 0
set_tid_address(0x7f2b44b00a10)         = 29531
set_robust_list(0x7f2b44b00a20, 24)     = 0
rseq(0x7f2b44b010e0, 0x20, 0, 0x53053053) = -1 ENOSYS (Function not implemented)
mprotect(0x7f2b44d26000, 16384, PROT_READ) = 0
mprotect(0x7f2b44d89000, 4096, PROT_READ) = 0
```

mprotect(0x7f2b44d78000, 8192, PROT_READ) = 0

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

munmap(0x7f2b44d3b000, 17675) = 0

readlink("/proc/self/exe", "/home/ali_/OS_2/lab_1_/posix_ipc"..., 1023) = 39

pipe2([3, 4], 0) = 0

clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD29534: I'm a child

, child_tidptr=0x7f2b44b00a10) = 29534

getpid() = 29531

write(1, "29531: I'm a parent, my child ha"..., 4429531: I'm a parent, my child has PID 29534

) = 44

wait4(-1, 29534: Start typing lines of text. Press 'Ctrl-D' or 'Enter' with no input to exit

6666666666666666666666666666666 2222

error: overflow long int type

[{WIFEXITED(s) && WEXITSTATUS(s) == 1}], 0, NULL) = 29534

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=29534, si_uid=1000, si_status=1, si_utime=0, si_stime=0} ---

write(2, "error: child exited with error\n\0", 32error: child exited with error

) = 32

exit_group(256) = ?

+++ exited with 0 +++

Вывод

Я приобрела навыки управления процессами в ОС и обеспечение обмена данных между процессами посредством каналов. Отработала создание дочернего процесса с помощью fork(), перенаправила ввод из родительского процесса в дочерний, с помощью канала, сделала замещение тела процесса, виртуальное адресное пространство процесса полностью меняется, загружается в виртуальное адресное пространство новая программа и необходимые библиотеки, передается управление на точку входа новой программы. Отладила программу на языке Си.