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: М8О-213Б-23

Студент: Иванов В. М.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 18.10.24

Москва, 2024

Постановка задачи

Вариант 20.

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в child1 или child2 в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Вариант 20) Правило фильтрации: строки длины больше 10 символов отправляются Child2, иначе Child1. Дочерние процессы инвертируют строки.

Общий метод и алгоритм решения

Кратко опишите системные вызовы, которые вы использовали в лабораторной работе.

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- shmget() - выделяет общую обла ть памяти
- shmat() - присединяет к программе общую память
- shmdt() - отсоединяет память
- semget() - создает семафор
- semctl() - управляет семафором
- semop() - ожидать семафор
- void exit(int status) — завершение выполнения процесса и возвращение статуса.
- int dup2(int oldfd, int newfd) — переназначение файлового дескриптора.
- int close(int fd) — закрыть файл.
- int execlp() - заменяет текущий процесс на новый процесс, загружая исполняемый файл.
- int open() - открытие/создание файла.
- int write() - вывод на экран сообщение.
- int read() - чтение с файла.

Общий алгоритм:

- Запросить у пользователя названия файлов
- Выделить общую память
- форкнуть процесс и переназначить stdout на открытый файл
- запустить дочерний процесс через execl(), передав в качестве параметра коммандной строки id семафора
- запрашивать у пользователя строки и в зависимости от длины строки дёргаем семафоры разных процессов

Код программы

main.c

```
#include <stdio.h>

#include <stdlib.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/types.h>

#include <sys/sem.h>
```

```
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

#define SHM_SIZE 1024
#define SEM_KEY1 1234
#define SEM_KEY2 5678

void P(int semid)
{
    struct sembuf p = {0, -1, 0};
    semop(semid, &p, 1);
}

void V(int semid)
{
    struct sembuf v = {0, 1, 0};
    semop(semid, &v, 1);
}

int main()
{
    char buffer[SHM_SIZE];
    char filename1[BUFSIZ];
    char filename2[BUFSIZ];
    write(1, "Enter a filename for child1: ", 30);
    read(0, filename1, BUFSIZ);
    filename1[strcspn(filename1, "\n")] = '\0';

    write(1, "Enter a filename for child2: ", 30);
    read(0, filename2, BUFSIZ);
    filename2[strcspn(filename2, "\n")] = '\0';

    int fd1 = open(filename1, O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fd1 == -1)
    {
```

```
    write(2, "Error Opening File 1", 20);
    exit(EXIT_FAILURE);
}
int fd2 = open(filename2, O_WRONLY | O_CREAT | O_TRUNC, 0644);
if (fd2 == -1)
{
    close(fd1);
    write(2, "Error Opening File 2", 20);
    exit(EXIT_FAILURE);
}

// Создание семафоров
int semid1 = semget(SEM_KEY1, 1, IPC_CREAT | 0666);
semctl(semid1, 0, SETVAL, 1);
int semid2 = semget(SEM_KEY2, 1, IPC_CREAT | 0666);
semctl(semid2, 0, SETVAL, 1);
if (semid1 == -1 || semid2 == -1)
{
    close(fd1);
    close(fd2);
    write(2, "Error in Semget", 15);
    exit(1);
}
// Создание разделяемой памяти
int shmid1 = shmget(1234, SHM_SIZE, IPC_CREAT | 0666);
char *shmemes = shmat(shmid1, NULL, 0);

if (shmemes == (char *)-1)
{
    close(fd1);
    close(fd2);
    semctl(semid1, 0, IPC_RMID);
    semctl(semid2, 0, IPC_RMID);
    write(2, "Error in Shmat", 14);
    exit(1);
}
```

```
// Создаём дочерний процесс
pid_t pid = fork();
if (pid == -1)
{
    close(fd1);
    close(fd2);
    shmdt(shmemes);
    shmctl(shmid1, IPC_RMID, NULL);
    semctl(semid1, 0, IPC_RMID);
    semctl(semid2, 0, IPC_RMID);
    write(2, "Error in Child1", 15);
    exit(1);
}
if (pid == 0)
{
    dup2(fd1, 1);
    close(fd1);
    // Дочерний процесс 1
    execlp("./child", "./child", "1234", (char *)NULL);
    perror("execlp failed");
    exit(1);
}
pid = fork();
if (pid == -1)
{
    close(fd1);
    close(fd2);
    shmdt(shmemes);
    shmctl(shmid1, IPC_RMID, NULL);
    semctl(semid1, 0, IPC_RMID);
    semctl(semid2, 0, IPC_RMID);
    write(2, "Error in Child1", 15);
    exit(1);
}
if (pid == 0)
```

```
{
    dup2(fd2, 1);
    close(fd2);
    execlp("./child", "./child", "5678", (char *)NULL);
    perror("execlp failed");
    exit(1);
}
// Записываем данные в общую память
while (1)
{
    write(1, "Enter line: ", 13);
    ssize_t bytes_read = read(0, buffer, SHM_SIZE);
    if (bytes_read == 0)
    { // EOF
        break;
    }
    else if (bytes_read == -1)
    {
        write(2, "Error reading Line", 18);
        close(fd1);
        close(fd2);
        shmdt(shmemes);
        shmctl(shmid1, IPC_RMID, NULL);
        semctl(semid1, 0, IPC_RMID);
        semctl(semid2, 0, IPC_RMID);
        exit(1);
    }
    buffer[bytes_read - 1] = '\0';

    if (strlen(buffer) == 0)
    {
        break;
    }

    if (strlen(buffer) > 10)
    {
```

```
            // write(pipe2[1], buffer, bytes_read);
            semctl(semid2, 0, SETVAL, 1);
            strcpy(shmemes, buffer);
            semctl(semid2, 0, SETVAL, 0);
        }
        else
        {
            // P(semid); // ждем 0
            semctl(semid1, 0, SETVAL, 1);
            strcpy(shmemes, buffer);
            semctl(semid1, 0, SETVAL, 0);
        }
    }
    // Удаление разделяемой памяти и семафоров
    P(semid1);
    P(semid2); // ждем пока семафоры поднимутся в 1
    memset(shmemes, 0, SHM_SIZE);
    semctl(semid1, 0, SETVAL, 0);
    semctl(semid2, 0, SETVAL, 0); // Шлем сигнал с пустым буфером == завершение
    shmdt(shmemes);
    shmctl(shmid1, IPC_RMID, NULL);
    semctl(semid1, 0, IPC_RMID);
    semctl(semid2, 0, IPC_RMID);

    return 0;
}
child.c
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/sem.h>
#include <unistd.h>
#include <string.h>

#define SHM_SIZE 1024

void reverse_string(char *str)
```

```
{
    // sleep(2);
    int len = strlen(str);
    for (int i = 0; i < len / 2; i++)
    {
        char tmp = str[i];
        str[i] = str[len - 1 - i];
        str[len - 1 - i] = tmp;
    }
}

void P(int semid) // ождидаем семафор
{
    struct sembuf p = {0, 0, 0};
    semop(semid, &p, 1);
}

int is_digit(char c)
{
    return (c >= '0' && c <= '9');
}
int str_to_i(char *str)
{
    int buf = 0;
    int len = strlen(str);
    for (int i = 0; i < len; i++)
    {
        if (!is_digit(str[i]))
            return -1;
        buf *= 10;
        buf += (str[i] - '0');
    }
    return buf;
}

int main(int argc, char **argv)
```

```
{
    // write(1, argv[1], strlen(argv[1]));
    int id = str_to_i(argv[1]);
    // Получаем семафор
    int semid = semget(id, 1, 0666);
    // Получаем доступ к общей памяти
    int shmid = shmget(1234, SHM_SIZE, 0666);
    char *shared_memory = shmat(shmid, NULL, 0);
    P(semid);
    while (strlen(shared_memory) > 0)
    {
        semctl(semid, 0, SETVAL, 1);
        reverse_string(shared_memory);
        shared_memory[strlen(shared_memory)]='\n';
        write(1, shared_memory, strlen(shared_memory));
        memset(shared_memory, 0, strlen(shared_memory));
        // semctl(semid, 0, SETVAL, 2);
        P(semid);
    }
    close(1);
    return 0;
}
```

Протокол работы программы

Тестирование:

```
$ ./main
Enter a filename for child1: out1.txt
Enter a filename for child2: out2.txt
Enter line: asd
Enter line: 123
Enter line: 12345678901
Enter line: asdfghjklqw
Enter line:
$ cat out1.txt
dsa
```

```
321

$ cat out2.txt

10987654321

wqlkjhgfdsa
```

Strace:

```
    23896 execve("./main", ["./main"], 0x7ffeee5b0eb8 /* 95 vars */) = 0

    23896 brk(NULL)                           = 0xb9a000

    23896  arch_prctl(0x3001  /*  ARCH_???  */,  0x7ffd735dea60)  =  -1  EINVAL
(Недопустимый аргумент)

    23896 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или
каталога)

    23896 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

    23896 fstat(3, {st_mode=S_IFREG|0644, st_size=104655, ...}) = 0

    23896 mmap(NULL, 104655, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f00b2004000

    23896 close(3)                            = 0

    23896 openat(AT_FDCWD, "/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

    23896       read(3,       "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\
227\2\0\0\0\0\0"..., 832) = 832

    23896     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

    23896           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

    23896              pread64(3,              "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0T\
247\253\1\356\366\342\334\242\306\260\332\270\306V\241"..., 68, 896) = 68

    23896 fstat(3, {st_mode=S_IFREG|0755, st_size=2592552, ...}) = 0

    23896 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x7f00b2002000

    23896     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

    23896 mmap(NULL, 2133936, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f00b1c00000

    23896 mprotect(0x7f00b1c28000, 1892352, PROT_NONE) = 0

    23896  mmap(0x7f00b1c28000,  1527808,  PROT_READ|PROT_EXEC,  MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f00b1c28000

    23896  mmap(0x7f00b1d9d000,  360448,  PROT_READ,  MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x19d000) = 0x7f00b1d9d000

    23896  mmap(0x7f00b1df6000,  24576,  PROT_READ|PROT_WRITE,  MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x1f5000) = 0x7f00b1df6000

    23896  mmap(0x7f00b1dfc000,  53168,  PROT_READ|PROT_WRITE,  MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f00b1dfc000

    23896 close(3)                            = 0
```

```
    23896 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x7f00b2000000

    23896 arch_prctl(ARCH_SET_FS, 0x7f00b2003600) = 0

    23896 set_tid_address(0x7f00b20038d0)   = 23896

    23896 set_robust_list(0x7f00b20038e0, 24) = 0

    23896 rseq(0x7f00b2003fa0, 0x20, 0, 0x53053053) = 0

    23896 mprotect(0x7f00b1df6000, 16384, PROT_READ) = 0

    23896 mprotect(0x403000, 4096, PROT_READ) = 0

    23896 mprotect(0x7f00b2052000, 8192, PROT_READ) = 0

    23896     prlimit64(0,     RLIMIT_STACK,     NULL,     {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0

    23896 munmap(0x7f00b2004000, 104655)    = 0

    23896 write(1, "Enter a filename for child1: \0", 30) = 30

    23896 read(0, "out1.txt\n", 8192)       = 9

    23896 write(1, "Enter a filename for child2: \0", 30) = 30

    23896 read(0, "out2.txt\n", 8192)       = 9

    23896 openat(AT_FDCWD, "out1.txt", O_WRONLY|O_CREAT|O_TRUNC, 0644) = 3

    23896 openat(AT_FDCWD, "out2.txt", O_WRONLY|O_CREAT|O_TRUNC, 0644) = 4

    23896 semget(0x4d2, 1, IPC_CREAT|0666)  = 26

    23896 semctl(26, 0, SETVAL, 0x1)        = 0

    23896 semget(0x162e, 1, IPC_CREAT|0666) = 27

    23896 semctl(27, 0, SETVAL, 0x1)        = 0

    23896 shmget(0x4d2, 1024, IPC_CREAT|0666) = 63

    23896 shmat(63, NULL, 0)                = 0x7f00b2051000

    23896          clone(child_stack=NULL,          flags=CLONE_CHILD_CLEARTID|
CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7f00b20038d0) = 23995

    23995 set_robust_list(0x7f00b20038e0, 24 <unfinished ...>

    23896          clone(child_stack=NULL,          flags=CLONE_CHILD_CLEARTID|
CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>

    23995 <... set_robust_list resumed>)    = 0

    23995 dup2(3, 1 <unfinished ...>

    23996 set_robust_list(0x7f00b20038e0, 24 <unfinished ...>

    23995 <... dup2 resumed>)               = 1

    23996 <... set_robust_list resumed>)    = 0

    23995 close(3 <unfinished ...>

    23996 dup2(4, 1 <unfinished ...>

    23995 <... close resumed>)              = 0
```

```
    23896 <... clone resumed>, child_tidptr=0x7f00b20038d0) = 23996

    23996 <... dup2 resumed>)               = 1

    23995 execve("./child", ["./child", "1234"], 0x7ffd735deb88 /* 95 vars */
<unfinished ...>

    23996 close(4 <unfinished ...>

    23896 write(1, "Enter line: \0", 13 <unfinished ...>

    23996 <... close resumed>)              = 0

    23896 <... write resumed>)              = 13

    23996 execve("./child", ["./child", "5678"], 0x7ffd735deb88 /* 95 vars */
<unfinished ...>

    23896 read(0,  <unfinished ...>

    23996 <... execve resumed>)             = 0

    23995 <... execve resumed>)             = 0

    23996 brk(NULL <unfinished ...>

    23995 brk(NULL)                         = 0x13fb000

    23996 <... brk resumed>)                = 0xcb1000

    23995 arch_prctl(0x3001 /* ARCH_??? */, 0x7fff5b412400 <unfinished ...>

    23996 arch_prctl(0x3001 /* ARCH_??? */, 0x7fff77a2f030 <unfinished ...>

    23995 <... arch_prctl resumed>)              = -1 EINVAL (Недопустимый
аргумент)

    23996 <... arch_prctl resumed>)              = -1 EINVAL (Недопустимый
аргумент)

    23996 access("/etc/ld.so.preload", R_OK <unfinished ...>

    23995 access("/etc/ld.so.preload", R_OK <unfinished ...>

    23996 <... access resumed>)              = -1 ENOENT (Нет такого файла или
каталога)

    23995 <... access resumed>)              = -1 ENOENT (Нет такого файла или
каталога)

    23996     openat(AT_FDCWD,     "/etc/ld.so.cache",     O_RDONLY|O_CLOEXEC
<unfinished ...>

    23995     openat(AT_FDCWD,     "/etc/ld.so.cache",     O_RDONLY|O_CLOEXEC
<unfinished ...>

    23996 <... openat resumed>)             = 4

    23995 <... openat resumed>)             = 3

    23996 fstat(4,  <unfinished ...>

    23995 fstat(3,  <unfinished ...>

    23996 <... fstat resumed>{st_mode=S_IFREG|0644, st_size=104655, ...}) = 0

    23995 <... fstat resumed>{st_mode=S_IFREG|0644, st_size=104655, ...}) = 0

    23996 mmap(NULL, 104655, PROT_READ, MAP_PRIVATE, 4, 0 <unfinished ...>
```

```
23995 mmap(NULL, 104655, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>

23996 <... mmap resumed>)               = 0x7f35ca65b000

23995 <... mmap resumed>)               = 0x7f106da14000

23996 close(4 <unfinished ...>

23995 close(3 <unfinished ...>

23996 <... close resumed>)              = 0

23995 <... close resumed>)              = 0

23996     openat(AT_FDCWD,      "/lib64/libc.so.6",      O_RDONLY|O_CLOEXEC
<unfinished ...>

23995     openat(AT_FDCWD,      "/lib64/libc.so.6",      O_RDONLY|O_CLOEXEC
<unfinished ...>

23996 <... openat resumed>)             = 4

23995 <... openat resumed>)             = 3

23996 read(4,  <unfinished ...>

23995 read(3,  <unfinished ...>

23996  <...  read  resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\
227\2\0\0\0\0\0"..., 832) = 832

23996     pread64(4,      "\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

23996           pread64(4,           "\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

23996              pread64(4,              "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0T\
247\253\1\356\366\342\334\242\306\260\332\270\306V\241"..., 68, 896) = 68

23996 fstat(4, {st_mode=S_IFREG|0755, st_size=2592552, ...}) = 0

23996 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x7f35ca659000

23996     pread64(4,      "\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

23996  mmap(NULL,  2133936,  PROT_READ,  MAP_PRIVATE|MAP_DENYWRITE,  4,  0
<unfinished ...>

23995  <...  read  resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\
227\2\0\0\0\0\0"..., 832) = 832

23996 <... mmap resumed>)               = 0x7f35ca400000

23995 pread64(3,  <unfinished ...>

23996 mprotect(0x7f35ca428000, 1892352, PROT_NONE <unfinished ...>

23995     <...      pread64      resumed>"\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

23996 <... mprotect resumed>)           = 0

23996  mmap(0x7f35ca428000,  1527808,  PROT_READ|PROT_EXEC,  MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 4, 0x28000) = 0x7f35ca428000

23995           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
```

```
    23996   mmap(0x7f35ca59d000,   360448,   PROT_READ,   MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 4, 0x19d000 <unfinished ...>

    23995              pread64(3,              "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0T\
247\253\1\356\366\342\334\242\306\260\332\270\306V\241"..., 68, 896) = 68

    23995 fstat(3,  <unfinished ...>

    23996 <... mmap resumed>)               = 0x7f35ca59d000

    23995 <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2592552, ...}) = 0

    23996   mmap(0x7f35ca5f6000,   24576,   PROT_READ|PROT_WRITE,   MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 4, 0x1f5000 <unfinished ...>

    23995  mmap(NULL,  8192,  PROT_READ|PROT_WRITE,  MAP_PRIVATE|MAP_ANONYMOUS,  -
1, 0 <unfinished ...>

    23996 <... mmap resumed>)               = 0x7f35ca5f6000

    23995 <... mmap resumed>)               = 0x7f106da12000

    23996   mmap(0x7f35ca5fc000,   53168,   PROT_READ|PROT_WRITE,   MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

    23995 pread64(3,  <unfinished ...>

    23996 <... mmap resumed>)               = 0x7f35ca5fc000

    23995      <...      pread64      resumed>"\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

    23996 close(4 <unfinished ...>

    23995  mmap(NULL,  2133936,  PROT_READ,  MAP_PRIVATE|MAP_DENYWRITE,  3,  0
<unfinished ...>

    23996 <... close resumed>)              = 0

    23996  mmap(NULL,  8192,  PROT_READ|PROT_WRITE,  MAP_PRIVATE|MAP_ANONYMOUS,  -
1, 0 <unfinished ...>

    23995 <... mmap resumed>)               = 0x7f106d800000

    23996 <... mmap resumed>)               = 0x7f35ca657000

    23996 arch_prctl(ARCH_SET_FS, 0x7f35ca65a600) = 0

    23996 set_tid_address(0x7f35ca65a8d0 <unfinished ...>

    23995 mprotect(0x7f106d828000, 1892352, PROT_NONE <unfinished ...>

    23996 <... set_tid_address resumed>)    = 23996

    23996 set_robust_list(0x7f35ca65a8e0, 24) = 0

    23995 <... mprotect resumed>)           = 0

    23996 rseq(0x7f35ca65afa0, 0x20, 0, 0x53053053) = 0

    23995   mmap(0x7f106d828000,   1527808,   PROT_READ|PROT_EXEC,   MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f106d828000

    23996 mprotect(0x7f35ca5f6000, 16384, PROT_READ <unfinished ...>

    23995   mmap(0x7f106d99d000,   360448,   PROT_READ,   MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x19d000 <unfinished ...>

    23996 <... mprotect resumed>)           = 0
```

```
23995 <... mmap resumed>)               = 0x7f106d99d000

23996 mprotect(0x403000, 4096, PROT_READ <unfinished ...>

23995  mmap(0x7f106d9f6000,  24576,  PROT_READ|PROT_WRITE,  MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x1f5000) = 0x7f106d9f6000

23996 <... mprotect resumed>)           = 0

23996 mprotect(0x7f35ca6a9000, 8192, PROT_READ <unfinished ...>

23995  mmap(0x7f106d9fc000,  53168,  PROT_READ|PROT_WRITE,  MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

23996 <... mprotect resumed>)           = 0

23996 prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>

23995 <... mmap resumed>)               = 0x7f106d9fc000

23996         <...          prlimit64         resumed>{rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0

23995 close(3 <unfinished ...>

23996 munmap(0x7f35ca65b000, 104655 <unfinished ...>

23995 <... close resumed>)              = 0

23996 <... munmap resumed>)             = 0

23995 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x7f106da10000

23995 arch_prctl(ARCH_SET_FS, 0x7f106da13600 <unfinished ...>

23996 semget(0x162e, 1, 0666 <unfinished ...>

23995 <... arch_prctl resumed>)         = 0

23996 <... semget resumed>)             = 27

23995 set_tid_address(0x7f106da138d0 <unfinished ...>

23996 shmget(0x4d2, 1024, 0666 <unfinished ...>

23995 <... set_tid_address resumed>)    = 23995

23996 <... shmget resumed>)             = 63

23996 shmat(63, NULL, 0)                = 0x7f35ca6a8000

23995 set_robust_list(0x7f106da138e0, 24 <unfinished ...>

23996  semtimedop(27,  [{sem_num=0,  sem_op=0,  sem_flg=0}],  1,  NULL
<unfinished ...>

23995 <... set_robust_list resumed>)    = 0

23995 rseq(0x7f106da13fa0, 0x20, 0, 0x53053053) = 0

23995 mprotect(0x7f106d9f6000, 16384, PROT_READ) = 0

23995 mprotect(0x403000, 4096, PROT_READ) = 0

23995 mprotect(0x7f106da62000, 8192, PROT_READ) = 0

23995     prlimit64(0,     RLIMIT_STACK,     NULL,     {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
```

```
23995 munmap(0x7f106da14000, 104655)    = 0
23995 semget(0x4d2, 1, 0666)            = 26
23995 shmget(0x4d2, 1024, 0666)         = 63
23995 shmat(63, NULL, 0)                = 0x7f106da61000
23995   semtimedop(26,   [{sem_num=0,   sem_op=0,   sem_flg=0}],   1,   NULL
<unfinished ...>
23896 <... read resumed>"asd\n", 1024)  = 4
23896 semctl(26, 0, SETVAL, 0x1)        = 0
23896 semctl(26, 0, SETVAL, NULL)       = 0
23995 <... semtimedop resumed>)         = 0
23896 write(1, "Enter line: \0", 13 <unfinished ...>
23995 semctl(26, 0, SETVAL, 0x1 <unfinished ...>
23896 <... write resumed>)              = 13
23995 <... semctl resumed>)             = 0
23896 read(0,  <unfinished ...>
23995 write(1, "dsa", 3)                = 3
23995   semtimedop(26,   [{sem_num=0,   sem_op=0,   sem_flg=0}],   1,   NULL
<unfinished ...>
23896 <... read resumed>"123\n", 1024)  = 4
23896 semctl(26, 0, SETVAL, 0x1)        = 0
23896 semctl(26, 0, SETVAL, NULL)       = 0
23995 <... semtimedop resumed>)         = 0
23896 write(1, "Enter line: \0", 13 <unfinished ...>
23995 semctl(26, 0, SETVAL, 0x1 <unfinished ...>
23896 <... write resumed>)              = 13
23896 read(0,  <unfinished ...>
23995 <... semctl resumed>)             = 0
23995 write(1, "321", 3)                = 3
23995   semtimedop(26,   [{sem_num=0,   sem_op=0,   sem_flg=0}],   1,   NULL
<unfinished ...>
23896 <... read resumed>"12345678901\n", 1024) = 12
23896 semctl(27, 0, SETVAL, 0x1)        = 0
23896 semctl(27, 0, SETVAL, NULL)       = 0
23996 <... semtimedop resumed>)         = 0
23896 write(1, "Enter line: \0", 13)    = 13
23996 semctl(27, 0, SETVAL, 0x1 <unfinished ...>
23896 read(0,  <unfinished ...>
```

23996 <... semctl resumed>)   = 0

23996 write(1, "10987654321", 11)   = 11

23996 semtimedop(27, [{sem_num=0, sem_op=0, sem_flg=0}], 1, NULL <unfinished ...>

23896 <... read resumed>"asdfghjklqw\n", 1024) = 12

23896 semctl(27, 0, SETVAL, 0x1)   = 0

23896 semctl(27, 0, SETVAL, NULL)   = 0

23996 <... semtimedop resumed>)   = 0

23896 write(1, "Enter line: \0", 13 <unfinished ...>

23996 semctl(27, 0, SETVAL, 0x1 <unfinished ...>

23896 <... write resumed>)   = 13

23996 <... semctl resumed>)   = 0

23896 read(0, <unfinished ...>

23996 write(1, "wqlkjhgfdsa", 11)   = 11

23996 semtimedop(27, [{sem_num=0, sem_op=0, sem_flg=0}], 1, NULL <unfinished ...>

23896 <... read resumed>"\n", 1024)   = 1

23896 semtimedop(26, [{sem_num=0, sem_op=-1, sem_flg=0}], 1, NULL) = 0

23995 <... semtimedop resumed>)   = 0

23896 semtimedop(27, [{sem_num=0, sem_op=-1, sem_flg=0}], 1, NULL <unfinished ...>

23995 exit_group(0 <unfinished ...>

23896 <... semtimedop resumed>)   = 0

23996 <... semtimedop resumed>)   = 0

23995 <... exit_group resumed>)   = ?

23896 **semctl(26, 0, SETVAL, NULL <unfinished ...>**

23996 exit_group(0 <unfinished ...>

23995 +++ exited with 0 +++

23896 <... semctl resumed>)   = 0

23896 --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=23995, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

23996 <... exit_group resumed>)   = ?

23896 semctl(27, 0, SETVAL, NULL)   = 0

23896 shmdt(0x7f00b2051000 <unfinished ...>

23996 +++ exited with 0 +++

23896 <... shmdt resumed>)   = 0

23896 --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=23996, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

```
23896 shmctl(63, IPC_RMID, NULL)        = 0
23896 semctl(26, 0, IPC_RMID, NULL)     = 0
23896 semctl(27, 0, IPC_RMID, NULL)     = 0
23896 exit_group(0)                     = ?
23896 +++ exited with 0 +++
```

Вывод

Я научился созадваьб процессы вlinux с помощщью системных вызовов. Научился использовать общую память и синхронизировать доступ к ней с помошью семафоров. Так же я использовал dup2() для переопределения файловых дескрипторов. Эти знания помогут мне лучше разобраться в принципах написания низкоуровнего системного ПО и в устройстве операционных систем.