Московский авиационный институт
(Национальный исследовательский университет)
Факультет "Информационные технологии и прикладная математика"
Кафедра "Вычислительная математика и программирование"

Лабораторная работа №1 по курсу “Операционные системы”

Студент: Живалев Е.А.

Группа: М8О-206Б

Преподаватель: Соколов А.А.

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2019

1 Задание

Продемонстрировать работу с утилитой strace на примере одной из лабораторных работ.

Программе на вход поступает число n - количество процессов, которое будет создано в ходе выполнения и программы, и n названий файлов, в которые эти процессы будут записывать данные. Родительский процесс создает n дочерних процессов, которым поочередно передает символы с входной строки. Дочерний процесс записывает полученные символ в переданный ему файл.

В ходе выполнения лабораторной работы были использованы следующие системные вызовы:

- open - открытие файла и получение его дескриптора
- read - использовался для чтения данных с входной строки
- sem_open - создание семафора или открытие ранее созданного
- sem_post - увеличение счетчика семафора
- sem_wait - уменьшение счетчика семафора
- fork - создание дочерних процессов
- mmap - отображение файла на память

2 Описание работы программы

С помощью функции readLine, которая, используя read считывает одну строку с входной строки, программа получает количество процессов(число n), которые необходимо создать и имена файлов, в которые должна производиться запись. Затем создаётся n пайпов и n процессов. Затем в родительском процессе в цикле с условием, что возвращенное функцией read число больше 0 происходит запись считанного символа в определенный элемент массива char, который был записан в файл, отображенный в память, и увеличивает значение семафора, связанного с чтением. В то же время дочерний процесс считывает символ и записывает его в свой файл, а также увеличивает значение семафора, связанного с записью символов, тем самым давая родительскому процессу понять, что можно записать новый символ.

3 Исходный код

main.c

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

#define FILE_NAME_LENGTH 129


void readLine(char* result) {
    char c;
    int byteCounter = 0;
    while(byteCounter < FILE_NAME_LENGTH) {
        if(read(STDIN_FILENO, &c, 1) == -1) {
            perror("Reading failed");
        }
        if(c == '\n') {
            break;
        } else {
            result[byteCounter++] = c;
        }
    }
    result[byteCounter] = '\0';
}

int main() {
    pid_t pid;
    char c;
    int numberOfProcesses = 1;
    char buf[FILE_NAME_LENGTH];
    readLine(buf);
    char* inputSemaphoreName = strdup("/semaphore?input");
    char* outputSemaphoreName = strdup("/semaphore?output");
    if(atoi(buf) > 0) {
        numberOfProcesses = atoi(buf);
    } else {
        printf("Invalid argment - must be a positive number\n");
        exit(-1);
    }
    char* map;
    char fileNames[numberOfProcesses][FILE_NAME_LENGTH];
    int processNumber = 0;
    sem_t* inputSemaphores[numberOfProcesses];
    sem_t* outputSemaphores[numberOfProcesses];
    for(int i = 0; i < numberOfProcesses; ++i) {
        readLine(buf);
        strcpy(fileNames[i], buf);
    }
    for(int i = 0; i < numberOfProcesses; ++i) {
        inputSemaphoreName[10] = '0' + i;
```

```
57          outputSemaphoreName[10] = '0' + i;
58          inputSemaphores[i] = sem_open(inputSemaphoreName, O_CREAT,
     777, 0);
59          outputSemaphores[i] = sem_open(outputSemaphoreName,
    O_CREAT, 777, 1);
60          if(inputSemaphores[i] == SEM_FAILED ||
61                  outputSemaphores[i] == SEM_FAILED) {
62              perror("Fault during semaphore init");
63              return -1;
64          }
65          sem_unlink(inputSemaphoreName);
66          sem_unlink(outputSemaphoreName);
67      }
68      char* tempFileName = strdup("/tmp/tmp_file.XXXXXX");
69      int tempFileDescriptor = mkstemp(tempFileName);
70      free(tempFileName);
71      int fileSize = numberOfProcesses + 1;
72      char temp[fileSize];
73      for(int i = 0; i < fileSize; ++i) {
74          temp[i] = ' ';
75      }
76      write(tempFileDescriptor, temp, fileSize);
77      if((map = (char*)mmap(NULL, fileSize, PROT_WRITE | PROT_READ,
    MAP_SHARED,
78                      tempFileDescriptor, 0)) == MAP_FAILED) {
79          perror("Mapping failed");
80          return -1;
81      }
82      for(int i = 0; i < numberOfProcesses; ++i) {
83         pid = fork();
84         if(pid == 0) {
85              processNumber = i;
86              break;
87         } else if (pid == -1) {
88              perror("Fork failed");
89              return -1;
90         }
91      }
92      if(pid > 0) {
93          while(read(STDIN_FILENO, &c, 1) > 0) {
94              sem_wait(outputSemaphores[processNumber]);
95              map[processNumber] = c;
96              sem_post(inputSemaphores[processNumber]);
97              processNumber++;
98              processNumber %= numberOfProcesses;
99          }
100         for(int i = 0; i < numberOfProcesses; ++i) {
101             map[i] = '\0';
102             sem_post(inputSemaphores[i]);
103         }
104     } else if(pid == 0) {
105         int fd = open(fileNames[processNumber], O_CREAT | O_WRONLY
    , S_IREAD |
106                 S_IWRITE);
107         if(ftruncate(fd, 0) == -1) {
108             perror("truncation failed");
109         }
110         while(1) {
111             sem_wait(inputSemaphores[processNumber]);
```

```
112             char c;
113             c = map[processNumber];
114             sem_post(outputSemaphores[processNumber]);
115             if(c == '\0') {
116                 break;
117             } else {
118                 if(write(fd, &c, 1) == -1) {
119                     perror("Writing failed");
120                 }
121             }
122         }
123         close(fd);
124         exit(0);
125     }
126     close(tempFileDescriptor);
127     munmap(map, fileSize);
128     return 0;
129 }
```

4 Консоль

```
qelderdelta@qelderdelta-UX331UA:~/Study/OS/os_lab_4/src$ ./a.out
3
1.txt
2.txt
3.txt
ttthhhiiisss   ssshhhooouuullldddd   bbbeee   iiinnn   fffiiillleee123
qelderdelta@qelderdelta-UX331UA:~/Study/OS/os_lab_4/src$ cat 1.txt
this should be in file1
qelderdelta@qelderdelta-UX331UA:~/Study/OS/os_lab_4/src$ cat 2.txt
this should be in file2
qelderdelta@qelderdelta-UX331UA:~/Study/OS/os_lab_4/src$ cat 3.txt
this should be in file3


qelderdelta@qelderdelta-UX331UA:~/Study/OS/os_lab_1/src$ strace ./a.out
execve("./a.out", ["./a.out"], 0x7ffda39b0c70 /* 54 vars */) = 0
brk(NULL)                                = 0x55f985fda000
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
fstat(3, \{st\_mode=S\_IFREG|0644, st\_size=142252, ...\}) = 0
mmap(NULL, 142252, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fe4d59cf000
close(3)                                 = 0
openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libpthread.so.0", O\_RDONLY|O\_CLOEXEC) =
read(3, "\textbackslash\{\}177ELF\textbackslash\{\}2\textbackslash\{\}1\textbacksla
fstat(3, \{st\_mode=S\_IFREG|0755, st\_size=149840, ...\}) = 0
mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7
mmap(NULL, 132288, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fe4d59ac000
mmap(0x7fe4d59b3000, 61440, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DE
mmap(0x7fe4d59c2000, 20480, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3,
mmap(0x7fe4d59c7000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DE
mmap(0x7fe4d59c9000, 13504, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_A
close(3)                                 = 0
openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3
read(3, "\textbackslash\{\}177ELF\textbackslash\{\}2\textbackslash\{\}1\textbacksla
fstat(3, \{st\_mode=S\_IFREG|0755, st\_size=2000480, ...\}) = 0
mmap(NULL, 2008696, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fe4d57c100
mmap(0x7fe4d57e6000, 1519616, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_
mmap(0x7fe4d5959000, 299008, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3
mmap(0x7fe4d59a2000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DE
mmap(0x7fe4d59a8000, 13944, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_A
close(3)                                 = 0
mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x
arch\_prctl(ARCH\_SET\_FS, 0x7fe4d57be740) = 0
mprotect(0x7fe4d59a2000, 12288, PROT\_READ) = 0
mprotect(0x7fe4d59c7000, 4096, PROT\_READ) = 0
mprotect(0x55f9844ee000, 4096, PROT\_READ) = 0
mprotect(0x7fe4d5a1c000, 4096, PROT\_READ) = 0
```

```
munmap(0x7fe4d59cf000, 142252)          = 0
set\_tid\_address(0x7fe4d57bea10)         = 3655
set\_robust\_list(0x7fe4d57bea20, 24)     = 0
rt\_sigaction(SIGRTMIN, \{sa\_handler=0x7fe4d59b36c0, sa\_mask=[], sa\_flags=SA\_R
rt\_sigaction(SIGRT\_1, \{sa\_handler=0x7fe4d59b3760, sa\_mask=[], sa\_flags=SA\_R
rt\_sigprocmask(SIG\_UNBLOCK, [RTMIN RT\_1], NULL, 8) = 0
prlimit64(0, RLIMIT\_STACK, NULL, \{rlim\_cur=8192*1024, rlim\_max=RLIM64\_INFINIT
read(0, 2
"2", 1)                          = 1
read(0, "\textbackslash\{\}n", 1)                       = 1
brk(NULL)                               = 0x55f985fda000
brk(0x55f985ffb000)                     = 0x55f985ffb000
read(0, 1.txt
"1", 1)                          = 1
read(0, ".", 1)                         = 1
read(0, "t", 1)                         = 1
read(0, "x", 1)                         = 1
read(0, "t", 1)                         = 1
read(0, "\textbackslash\{\}n", 1)                       = 1
read(0, 2.txt
"2", 1)                          = 1
read(0, ".", 1)                         = 1
read(0, "t", 1)                         = 1
read(0, "x", 1)                         = 1
read(0, "t", 1)                         = 1
read(0, "\textbackslash\{\}n", 1)                       = 1
statfs("/dev/shm/", \{f\_type=TMPFS\_MAGIC, f\_bsize=4096, f\_blocks=1001828, f\_b
futex(0x7fe4d59cc3b0, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0
openat(AT\_FDCWD, "/dev/shm/sem.semaphore0input", O\_RDWR|O\_NOFOLLOW) = -1 ENOENT
getpid()                                = 3655
lstat("/dev/shm/iHu5Yh", 0x7ffd496ec340) = -1 ENOENT (Нет такого файла или каталога
openat(AT\_FDCWD, "/dev/shm/iHu5Yh", O\_RDWR|O\_CREAT|O\_EXCL, 01411) = 3
write(3, "\textbackslash\{\}0\textbackslash\{\}0\textbackslash\{\}0\textbackslash\
mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7fe4d59f1000
link("/dev/shm/iHu5Yh", "/dev/shm/sem.semaphore0input") = 0
fstat(3, \{st\_mode=S\_IFREG|S\_ISVTX|0411, st\_size=32, ...\}) = 0
unlink("/dev/shm/iHu5Yh")               = 0
close(3)                                = 0
openat(AT\_FDCWD, "/dev/shm/sem.semaphore0output", O\_RDWR|O\_NOFOLLOW) = -1 ENOENT
getpid()                                = 3655
lstat("/dev/shm/CYRcbL", 0x7ffd496ec340) = -1 ENOENT (Нет такого файла или каталога
openat(AT\_FDCWD, "/dev/shm/CYRcbL", O\_RDWR|O\_CREAT|O\_EXCL, 01411) = 3
write(3, "\textbackslash\{\}1\textbackslash\{\}0\textbackslash\{\}0\textbackslash\
mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7fe4d59f0000
link("/dev/shm/CYRcbL", "/dev/shm/sem.semaphore0output") = 0
fstat(3, \{st\_mode=S\_IFREG|S\_ISVTX|0411, st\_size=32, ...\}) = 0
unlink("/dev/shm/CYRcbL")               = 0
close(3)                                = 0
unlink("/dev/shm/sem.semaphore0input")  = 0
```

```
unlink("/dev/shm/sem.semaphore0output") = 0
openat(AT\_FDCWD, "/dev/shm/sem.semaphore1input", O\_RDWR|O\_NOFOLLOW) = -1 ENOENT
getpid()                                = 3655
lstat("/dev/shm/U0Srne", 0x7ffd496ec340) = -1 ENOENT (Нет такого файла или каталога
openat(AT\_FDCWD, "/dev/shm/U0Srne", O\_RDWR|O\_CREAT|O\_EXCL, 01411) = 3
write(3, "\textbackslash\{\}0\textbackslash\{\}0\textbackslash\{\}0\textbackslash\
mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7fe4d59ef000
link("/dev/shm/U0Srne", "/dev/shm/sem.semaphore1input") = 0
fstat(3, \{st\_mode=S\_IFREG|S\_ISVTX|0411, st\_size=32, ...\}) = 0
unlink("/dev/shm/U0Srne")               = 0
close(3)                                = 0
openat(AT\_FDCWD, "/dev/shm/sem.semaphore1output", O\_RDWR|O\_NOFOLLOW) = -1 ENOENT
getpid()                                = 3655
lstat("/dev/shm/rdhNzH", 0x7ffd496ec340) = -1 ENOENT (Нет такого файла или каталога
openat(AT\_FDCWD, "/dev/shm/rdhNzH", O\_RDWR|O\_CREAT|O\_EXCL, 01411) = 3
write(3, "\textbackslash\{\}1\textbackslash\{\}0\textbackslash\{\}0\textbackslash\
mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7fe4d59ee000
link("/dev/shm/rdhNzH", "/dev/shm/sem.semaphore1output") = 0
fstat(3, \{st\_mode=S\_IFREG|S\_ISVTX|0411, st\_size=32, ...\}) = 0
unlink("/dev/shm/rdhNzH")               = 0
close(3)                                = 0
unlink("/dev/shm/sem.semaphore1input")  = 0
unlink("/dev/shm/sem.semaphore1output") = 0
getpid()                                = 3655
openat(AT\_FDCWD, "/tmp/tmp\_file.uZNfMa", O\_RDWR|O\_CREAT|O\_EXCL, 0600) = 3
write(3, "   ", 3)                       = 3
mmap(NULL, 3, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7fe4d59ed000
clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD
clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD
read(0, xxxddd
"x", 1)                           = 1
futex(0x7fe4d59f1000, FUTEX\_WAKE, 1)    = 1
read(0, "x", 1)                         = 1
futex(0x7fe4d59ef000, FUTEX\_WAKE, 1)    = 1
read(0, "x", 1)                         = 1
futex(0x7fe4d59f1000, FUTEX\_WAKE, 1)    = 1
read(0, "d", 1)                         = 1
futex(0x7fe4d59ef000, FUTEX\_WAKE, 1)    = 1
read(0, "d", 1)                         = 1
futex(0x7fe4d59f1000, FUTEX\_WAKE, 1)    = 1
read(0, "d", 1)                         = 1
futex(0x7fe4d59ef000, FUTEX\_WAKE, 1)    = 1
read(0, "\textbackslash\{\}n", 1)                       = 1
futex(0x7fe4d59f1000, FUTEX\_WAKE, 1)    = 1
read(0, "", 1)                          = 0
futex(0x7fe4d59f1000, FUTEX\_WAKE, 1)    = 1
futex(0x7fe4d59ef000, FUTEX\_WAKE, 1)    = 1
--- SIGCHLD \{si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=3656, si\_uid=1000,
close(3)                                = 0
```

```
munmap(0x7fe4d59ed000, 3)                = 0
--- SIGCHLD \{si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=3657, si\_uid=1000,
exit\_group(0)                            = ?
+++ exited with 0 +++
```

5 Выводы

В ходе выполнения лабораторной работы я познакомился с такими интересным механизмом, как отображение файла в память, который позволяет выиграть в производительности по сравнению с обычным чтением из файла за счет уменьшения количества системных вызовов, а также лишнего копирования.