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: Кочев Д.О.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 01.12.2023

Москва, 2023

Постановка задачи

Группа вариантов 5.

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 первый отображаемый файл (memory-mapped file) или во второй отображаемый файл в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Вариант 18.

Правило фильтрации: нечетные строки отправляются в первый отображаемый файл, четные во второй отображаемый файл. Дочерние процессы удаляют все гласные из строк.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- int shm_open(const char *name, int oflag, mode_t mode) - создает и открывает новый (или открывает уже существующий) объект разделяемой памяти POSIX.
- void * mmap(void *start, size_t length, int prot , int flags, int fd, off_t offset) - отражает файлы или устройства в памяти или снимает их отражение. При удачном выполнении mmap возвращает указатель на область с отраженными данными.
- int ftruncate(int fd, off_t length) - приводит файл к заданному размеру.
- int execl(char *name, char *arg0, ... /*NULL*/) – загружает и запускает указанную программу. Таким образом, новая программа полностью замещает текущий процесс.
- int dup2(int oldfd, int newfd) – делает newfd копией oldfd, закрывая newfd, если требуется.
- int close(int fd) - закрывает файловый дескриптор.
- void (*signal (int signal, void (*sigfunc) (int func)))(int) - дает указание выполнить функцию, на которую указывает sigfunc, в случае получения сигнала signal.

- int kill(pid_t pid, int sig) - посылает сигнал процессу или выводит список допустимых сигналов.

- int unlink(const char *pathname) - unlink удаляет имя из файловой системы
- void exit(int status) - приводит к обычному завершению программы

После запуска программы пользователю нужно ввести в командную строку имя первого файла, затем на следующей строке имя второго файла. После этого функция open открывает файл с данным названием и очищает его. Если данного файла не было, то создаст его. Если все было введено корректно и два файла доступны для работы, то создаются два новых файла в разделяемой памяти (далее mmf файл), получим указатели на них. Далее будет создан первый дочерний процесс. В нем мы заменяем стандартный поток вывода на запись в файл с помощью функции dup2. Затем первый дочерний процесс запускает программу child.c, в которую передаем число, отвечающее за имя файла, который находится в разделяемой памяти, и программа main для этого процесса завершается. Аналогичные действия мы проделываем с вторым дочерним процессом. Далее в родительском процессе мы считываем все символы, которые вводит пользователь, записываем их в mmf файл. После символа ‘\n’ родительский процесс отправляет сигнал дочернему. Приняв сигнал, первый дочерний процесс запускает функцию, в которой считывает из mmf файла введенные данные, обрабатывает их и записывает в стандартный вывод, который подменен на вывод в файл в данном процессе. Далее все описанные действия выполняться для второго родительского процесса. Программа завершает работу, когда встретит символ EOF.

Код программы

laba3.c

```
#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <fcntl.h>

#include <string.h>

#include <stdbool.h>

#include <sys/mman.h>

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

```
#define MEMORY_NAME1 "first_mmf"
#define MEMORY_NAME2 "second_mmf"

char* get_fileaname() {
    int len = 0;
    int capacity = 1;
    char *s = (char*) malloc(sizeof(char));
    char c = getchar();
    while (c != '\n') {
        s[(len)++] = c;
        if (len >= capacity) {
            capacity *= 2;
            s = (char*) realloc(s, capacity * sizeof(char));
        }
                    if (capacity > 256) {
            s = NULL;
            return s;
                    }
        c = getchar();
    }
    s[len] = '\0';
    return s;
}
int main (){
            enum {
                    READ = 0,
```

```
        WRITE = 1
    };

    char * first_file = NULL;
    first_file = get_fileaname();
    if (first_file == NULL) {
        perror ("Large file name or no memory \n");
        return -1;
    }
    int out = open(first_file, O_WRONLY| O_CREAT | O_TRUNC ,
S_IWUSR);
    if (out == -1) {
        perror ("There is no such file \n");
        return -1;
    }
    char * second_file = NULL;
    second_file = get_fileaname();
    if (second_file == NULL) {
        perror ("Large file name or no memory\n");
        return -1;
    }
    int out2 = open(second_file, O_WRONLY| O_CREAT | O_TRUNC ,
S_IWUSR);
    if (out2 == -1) {
        perror ("There is no such file \n");
        return -1;
    }
```

```
        //очистка память названия файлов
        free(first_file);
        free(second_file);
   int fd_for_input = shm_open(MEMORY_NAME1, O_RDWR | O_CREAT | O_TRUNC , 0777); //S_IWUSR
   // printf("%d \n", fd_for_input);
        ftruncate (fd_for_input , 500*sizeof(int));
        char *file_mmf =  mmap(NULL, 500*sizeof(int), PROT_WRITE | PROT_READ , MAP_SHARED ,fd_for_input,0);
        int fd_for_input2 = shm_open(MEMORY_NAME2, O_RDWR | O_CREAT | O_TRUNC , 0777); //S_IWUSR
   // printf("%d \n", fd_for_input2);
        ftruncate (fd_for_input2 , 500*sizeof(int)); // Изменяет длинну файла на нужную
        char *file_mmf2 =  mmap(NULL, 500*sizeof(int), PROT_WRITE | PROT_READ , MAP_SHARED ,fd_for_input2,0);
        pid_t id = fork();
        if (id == 0){
      if (dup2(out, STDOUT_FILENO) == -1){ // fileno(stdout)
                        perror ("dup2");
                }
      char str1[sizeof(int)];
      str1[0] = '1';
      // sprintf(str1, "%d", fd_for_input); // int в строку чаров
                execl("./child", "./child", str1, NULL);
                perror("execl");
        }
   else if (id < 0){
```

```
    perror ("fork\n");
                exit(0);
}
    pid_t id2 = fork();
    if (id2 == 0){
                if (dup2(out2, STDOUT_FILENO) == -1){ // fileno(stdout)
                        perror ("dup2");
                }
    char str2[sizeof(int)];
    str2[0] = '2';
    // sprintf(str2, "%d", fd_for_input2);
                execl("./child", "./child", str2, NULL);
                perror("execl");
    }
else if (id2 < 0){
    perror ("fork\n");
                exit(0);
}

    if (id > 0 ) {
                int c;
                int flag = 0;
    int k_1 = 0;
    int k_2 = 0;

                while  ((c = getchar()) != EOF) {
    if (k_1 == 500*sizeof(int)){
```

```
    k_1 = 0;
    flag ++;
}
else if (k_2 == 500*sizeof(int)){
    k_2 = 0;
    flag ++;
}


                if (flag % 2 == 0){
                        // write(fd[WRITE], &c, sizeof(int));
    file_mmf[k_1] = (char)c;
    k_1 ++;
                } else {
                        file_mmf2[k_2] = (char)c;
    k_2 ++;
                }


                if (c == '\n'){
    if (flag % 2 == 0){
        // msync(file_mmf, 500*sizeof(int), MS_SYNC| MS_INVALIDATE);
        // printf("here1\n");
        // for(int y = 0; y < 100; y++)
        //     printf("%c", file_mmf[y]);
        // printf("\n");
                                kill(id, SIGUSR1);
    }
    else{
```

```
                // msync(file_mmf2, 500*sizeof(int), MS_SYNC| MS_INVALIDATE);
                // printf("here2\n");
                // for(int y = 0; y < 100; y++)
                //     printf("%c", file_mmf2[y]);
                // printf("\n");
                                        kill(id2, SIGUSR1);
            }
                                flag ++;
                        }


                }
        kill(id, SIGUSR2);
        kill(id2, SIGUSR2);
                close(fd_for_input);
                close(fd_for_input2);
                unlink(file_mmf);
                unlink(file_mmf2);
          }
}
```

child.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <fcntl.h>
```

```
#include <string.h>
#include <stdbool.h>
#include <sys/mman.h>
#include <sys/stat.h>

#define MEMORY_NAME1 "first_mmf"
#define MEMORY_NAME2 "second_mmf"

char * file_mmf_global;
int i_global = 0;

void writer(){
   // msync(file_mmf_global, 500*sizeof(int), MS_SYNC| MS_INVALIDATE);
   int c;
   for (int i = i_global ; i < 500*sizeof(int); i++) {
      // printf("ЯЯ здесьььuuuuuuuuuu\n");
      c = file_mmf_global[i];
                  if ((c != 'a') && (c != 'e') && (c != 'i') && (c != 'u') && (c != 'y') &&
(c != 'o') &&
                  (c != 'A') && (c != 'E') && (c != 'I') && (c != 'U') && (c != 'Y') &&
(c != 'O')) {
                        write(fileno(stdout), &c, sizeof(char));
         if (c == '\n'){
            i_global = i + 1;
            break;
         }
                  }
         }
```

```
}

void quit(){
        unlink(file_mmf_global);
   exit(0);
}

int main (int argc, const char *argv[]){
   int a = atoi(argv[1]);

   // int out = atoi(argv[1]);
        int out = 0;
        if (a == 1) {
                out = shm_open(MEMORY_NAME1, O_RDWR, S_IRUSR);
        }
        else {
                out = shm_open(MEMORY_NAME2, O_RDWR, S_IRUSR);
        }

        char *file_mmf =  mmap(NULL, 500*sizeof(int),  PROT_WRITE
|PROT_READ , MAP_SHARED ,out,0);
        if (file_mmf == NULL) {
                perror ("Ошибка mmap\n");
        }

   file_mmf_global = file_mmf;
```

```
        // int c;
   // printf("ЯЯ здесььь\n");
        // for (int i = 0 ; i < 500*sizeof(int); i++) {
   //     msync(file_mmf, 500*sizeof(int), MS_SYNC| MS_INVALIDATE);
   //     // printf("ЯЯ здесьььuuuuuuuuuu\n");
   //     c = file_mmf[i];
        //         if ((c != 'a') && (c != 'e') && (c != 'i') && (c != 'u') && (c != 'y') &&
(c != 'o') &&
        //         (c != 'A') && (c != 'E') && (c != 'I') && (c != 'U') && (c != 'Y') &&
(c != 'O')) {
        //                 write(fileno(stdout), &c, sizeof(char));
        //         }
        // }
        // close(fileno(stdin));
   signal (SIGUSR1, writer);
        signal (SIGUSR2, quit);
        while (true);
}
```

Протокол работы программы

Тестирование:

```
$ ./parent
aaaa
bbbb
qwertyyyy
dddderayt
adassssddeeee
uuuyturrrtt
iooottttt
$ cat < aaaa
qwrt
dssssdd
ttttt
```

```
$ cat < bbbb
ddddrt
trrrtt
```

Strace:

```
$ strace -f ./parent
execve("./parent", ["./parent"], 0x7ffd95cc05e8 /* 36 vars */) = 0
brk(NULL)                               = 0x556b2bc64000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc089c41f0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7f9a09803000
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=18151, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 18151, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f9a097fe000
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) =
0x7f9a095d6000
mmap(0x7f9a095fe000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f9a095fe000
mmap(0x7f9a09793000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f9a09793000
mmap(0x7f9a097eb000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x7f9a097eb000
mmap(0x7f9a097f1000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f9a097f1000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0) = 0x7f9a095d3000
arch_prctl(ARCH_SET_FS, 0x7f9a095d3740) = 0
set_tid_address(0x7f9a095d3a10)         = 4189
set_robust_list(0x7f9a095d3a20, 24)     = 0
rseq(0x7f9a095d40e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7f9a097eb000, 16384, PROT_READ) = 0
mprotect(0x556b2ab99000, 4096, PROT_READ) = 0
mprotect(0x7f9a0983d000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) =
0
```

```
munmap(0x7f9a097fe000, 18151)           = 0
getrandom("\xac\x8d\xc6\x8c\xd7\x6c\x03\xc7", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x556b2bc64000
brk(0x556b2bc85000)                     = 0x556b2bc85000
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x4), ...},
AT_EMPTY_PATH) = 0
read(0, aaaa
"aaaa\n", 1024)                 = 5
openat(AT_FDCWD, "aaaa", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
read(0, bbbb
"bbbb\n", 1024)                 = 5
openat(AT_FDCWD, "bbbb", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 4
openat(AT_FDCWD, "/dev/shm/first_mmf",
O_RDWR|O_CREAT|O_TRUNC|O_NOFOLLOW|O_CLOEXEC, 0777) = 5
ftruncate(5, 2000)                      = 0
mmap(NULL, 2000, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) = 0x7f9a0983c000
openat(AT_FDCWD, "/dev/shm/second_mmf",
O_RDWR|O_CREAT|O_TRUNC|O_NOFOLLOW|O_CLOEXEC, 0777) = 6
ftruncate(6, 2000)                      = 0
mmap(NULL, 2000, PROT_READ|PROT_WRITE, MAP_SHARED, 6, 0) = 0x7f9a09802000
clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 4208
attached
, child_tidptr=0x7f9a095d3a10) = 4208
[pid  4208] set_robust_list(0x7f9a095d3a20, 24 <unfinished ...>
[pid  4189] clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>
[pid  4208] <... set_robust_list resumed>) = 0
[pid  4208] dup2(3, 1strace: Process 4209 attached
 <unfinished ...>
[pid  4189] <... clone resumed>, child_tidptr=0x7f9a095d3a10) = 4209
[pid  4208] <... dup2 resumed>)         = 1
[pid  4189] read(0,  <unfinished ...>
[pid  4209] set_robust_list(0x7f9a095d3a20, 24 <unfinished ...>
[pid  4208] execve("./child", ["./child", "5"], 0x7ffc089c43c8 /* 36 vars */ <unfinished ...>
[pid  4209] <... set_robust_list resumed>) = 0
[pid  4209] dup2(4, 1)                  = 1
[pid  4209] execve("./child", ["./child", "6"], 0x7ffc089c43c8 /* 36 vars */) = 0
[pid  4209] brk(NULL <unfinished ...>
[pid  4208] <... execve resumed>)       = 0
[pid  4209] <... brk resumed>)          = 0x55b5200db000
[pid  4209] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd5eee27f0) = -1 EINVAL (Invalid
argument)
[pid  4209] mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fefe252e000
[pid  4209] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
[pid  4209] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 7
```

```
[pid  4209] newfstatat(7, "", {st_mode=S_IFREG|0644, st_size=18151, ...},
AT_EMPTY_PATH) = 0
[pid  4209] mmap(NULL, 18151, PROT_READ, MAP_PRIVATE, 7, 0 <unfinished ...>
[pid  4208] brk(NULL <unfinished ...>
[pid  4209] <... mmap resumed>)         = 0x7fefe2529000
[pid  4208] <... brk resumed>)          = 0x55c86e4af000
[pid  4209] close(7)                    = 0
[pid  4208] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe45e420d0 <unfinished ...>
[pid  4209] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>
[pid  4208] <... arch_prctl resumed>)   = -1 EINVAL (Invalid argument)
[pid  4209] <... openat resumed>)       = 7
[pid  4208] mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid  4209] read(7,  <unfinished ...>
[pid  4208] <... mmap resumed>)         = 0x7fded8be3000
[pid  4209] <... read
resumed>"\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
[pid  4208] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid  4209] pread64(7,  <unfinished ...>
[pid  4208] <... access resumed>)       = -1 ENOENT (No such file or directory)
[pid  4209] <... 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
[pid  4208] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished
...>
[pid  4209] pread64(7,  <unfinished ...>
[pid  4208] <... openat resumed>)       = 7
[pid  4209] <... pread64 resumed>"\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
[pid  4208] newfstatat(7, "",  <unfinished ...>
[pid  4209] pread64(7,  <unfinished ...>
[pid  4208] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=18151, ...},
AT_EMPTY_PATH) = 0
[pid  4209] <... pread64
resumed>"\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
[pid  4208] mmap(NULL, 18151, PROT_READ, MAP_PRIVATE, 7, 0 <unfinished ...>
[pid  4209] newfstatat(7, "",  <unfinished ...>
[pid  4208] <... mmap resumed>)         = 0x7fded8bde000
[pid  4209] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2216304, ...},
AT_EMPTY_PATH) = 0
[pid  4208] close(7 <unfinished ...>
[pid  4209] pread64(7,  <unfinished ...>
[pid  4208] <... close resumed>)        = 0
[pid  4209] <... 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
[pid  4208] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>
```

```
[pid  4209] mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 7, 0
<unfinished ...>
[pid  4208] <... openat resumed>)       = 7
[pid  4209] <... mmap resumed>)         = 0x7fefe2301000
[pid  4208] read(7,  <unfinished ...>
[pid  4209] mmap(0x7fefe2329000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x28000 <unfinished ...>
[pid  4208] <... read
resumed>"\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
[pid  4209] <... mmap resumed>)         = 0x7fefe2329000
[pid  4208] pread64(7,  <unfinished ...>
[pid  4209] mmap(0x7fefe24be000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x1bd000 <unfinished ...>
[pid  4208] <... 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
[pid  4209] <... mmap resumed>)         = 0x7fefe24be000
[pid  4208] pread64(7,  <unfinished ...>
[pid  4209] mmap(0x7fefe2516000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x214000 <unfinished ...>
[pid  4208] <... pread64 resumed>"\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
[pid  4209] <... mmap resumed>)         = 0x7fefe2516000
[pid  4208] pread64(7,  <unfinished ...>
[pid  4209] mmap(0x7fefe251c000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid  4208] <... pread64
resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\244;\374\204(\337f#\315l\214\234\f\256\271\32"...
, 68, 896) = 68
[pid  4209] <... mmap resumed>)         = 0x7fefe251c000
[pid  4208] newfstatat(7, "",  <unfinished ...>
[pid  4209] close(7 <unfinished ...>
[pid  4208] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2216304, ...},
AT_EMPTY_PATH) = 0
[pid  4209] <... close resumed>)        = 0
[pid  4208] pread64(7,  <unfinished ...>
[pid  4209] mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid  4208] <... 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
[pid  4209] <... mmap resumed>)         = 0x7fefe22fe000
[pid  4208] mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 7, 0
<unfinished ...>
[pid  4209] arch_prctl(ARCH_SET_FS, 0x7fefe22fe740 <unfinished ...>
[pid  4208] <... mmap resumed>)         = 0x7fded89b6000
[pid  4209] <... arch_prctl resumed>)   = 0
[pid  4208] mmap(0x7fded89de000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x28000 <unfinished ...>
[pid  4209] set_tid_address(0x7fefe22fea10 <unfinished ...>
```

```
[pid  4208] <... mmap resumed>)        = 0x7fded89de000
[pid  4209] <... set_tid_address resumed>) = 4209
[pid  4208] mmap(0x7fded8b73000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x1bd000 <unfinished ...>
[pid  4209] set_robust_list(0x7fefe22fea20, 24 <unfinished ...>
[pid  4208] <... mmap resumed>)        = 0x7fded8b73000
[pid  4209] <... set_robust_list resumed>) = 0
[pid  4208] mmap(0x7fded8bcb000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x214000 <unfinished ...>
[pid  4209] rseq(0x7fefe22ff0e0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  4208] <... mmap resumed>)        = 0x7fded8bcb000
[pid  4209] <... rseq resumed>)        = 0
[pid  4208] mmap(0x7fded8bd1000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid  4209] mprotect(0x7fefe2516000, 16384, PROT_READ <unfinished ...>
[pid  4208] <... mmap resumed>)        = 0x7fded8bd1000
[pid  4209] <... mprotect resumed>)    = 0
[pid  4209] mprotect(0x55b51fdc9000, 4096, PROT_READ <unfinished ...>
[pid  4208] close(7 <unfinished ...>
[pid  4209] <... mprotect resumed>)    = 0
[pid  4208] <... close resumed>)       = 0
[pid  4209] mprotect(0x7fefe2568000, 8192, PROT_READ <unfinished ...>
[pid  4208] mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid  4209] <... mprotect resumed>)    = 0
[pid  4208] <... mmap resumed>)        = 0x7fded89b3000
[pid  4209] prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
[pid  4208] arch_prctl(ARCH_SET_FS, 0x7fded89b3740 <unfinished ...>
[pid  4209] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid  4208] <... arch_prctl resumed>)  = 0
[pid  4209] munmap(0x7fefe2529000, 18151 <unfinished ...>
[pid  4208] set_tid_address(0x7fded89b3a10 <unfinished ...>
[pid  4209] <... munmap resumed>)      = 0
[pid  4208] <... set_tid_address resumed>) = 4208
[pid  4209] mmap(NULL, 2000, PROT_READ|PROT_WRITE, MAP_SHARED, 6, 0
<unfinished ...>
[pid  4208] set_robust_list(0x7fded89b3a20, 24 <unfinished ...>
[pid  4209] <... mmap resumed>)        = 0x7fefe2567000
[pid  4208] <... set_robust_list resumed>) = 0
[pid  4209] rt_sigaction(SIGUSR1, {sa_handler=0x55b51fdc7229, sa_mask=[USR1],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7fefe2343520},  <unfinished ...>
[pid  4208] rseq(0x7fded89b40e0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  4209] <... rt_sigaction resumed>{sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) =
0
[pid  4208] <... rseq resumed>)        = 0
[pid  4209] rt_sigaction(SIGUSR2, {sa_handler=0x55b51fdc7339, sa_mask=[USR2],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7fefe2343520},
{sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0
```

```
[pid  4208] mprotect(0x7fded8bcb000, 16384, PROT_READ) = 0
[pid  4208] mprotect(0x55c86c89b000, 4096, PROT_READ) = 0
[pid  4208] mprotect(0x7fded8c1d000, 8192, PROT_READ) = 0
[pid  4208] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
[pid  4208] munmap(0x7fded8bde000, 18151) = 0
[pid  4208] mmap(NULL, 2000, PROT_READ|PROT_WRITE, MAP_SHARED, 5, 0) =
0x7fded8c1c000
[pid  4208] rt_sigaction(SIGUSR1, {sa_handler=0x55c86c899229, sa_mask=[USR1],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7fded89f8520},
{sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0
[pid  4208] rt_sigaction(SIGUSR2, {sa_handler=0x55c86c899339, sa_mask=[USR2],
sa_flags=SA_RESTORER|SA_RESTART, sa_restorer=0x7fded89f8520},
{sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0
qwertyuuutrttt
[pid  4189] <... read resumed>"qwertyuuutrttt\n", 1024) = 15
[pid  4189] msync(0x7f9a0983c000, 2000, MS_SYNC|MS_INVALIDATE) = 0
[pid  4189] kill(4208, SIGUSR1)         = 0
[pid  4208] --- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=4189, si_uid=1000}
---
[pid  4189] read(0,  <unfinished ...>
[pid  4208] write(1, "q", 1)            = 1
[pid  4208] write(1, "w", 1)            = 1
[pid  4208] write(1, "r", 1)            = 1
[pid  4208] write(1, "t", 1)            = 1
[pid  4208] write(1, "t", 1)            = 1
[pid  4208] write(1, "r", 1)            = 1
[pid  4208] write(1, "t", 1)            = 1
[pid  4208] write(1, "t", 1)            = 1
[pid  4208] write(1, "t", 1)            = 1
[pid  4208] write(1, "\n", 1)           = 1
[pid  4208] rt_sigreturn({mask=[]})     = 0
ddkkkdkoooi
[pid  4189] <... read resumed>"ddkkkdkoooi\n", 1024) = 12
[pid  4189] msync(0x7f9a09802000, 2000, MS_SYNC|MS_INVALIDATE) = 0
[pid  4189] kill(4209, SIGUSR1)         = 0
[pid  4209] --- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=4189, si_uid=1000}
---
[pid  4189] read(0,  <unfinished ...>
[pid  4209] write(1, "d", 1)            = 1
[pid  4209] write(1, "d", 1)            = 1
[pid  4209] write(1, "k", 1)            = 1
[pid  4209] write(1, "k", 1)            = 1
[pid  4209] write(1, "k", 1)            = 1
[pid  4209] write(1, "d", 1)            = 1
[pid  4209] write(1, "k", 1)            = 1
[pid  4209] write(1, "\n", 1)           = 1
[pid  4209] rt_sigreturn({mask=[]})     = 0
```

```
wertdffooi
[pid  4189] <... read resumed>"wertdffooi\n", 1024) = 11
[pid  4189] msync(0x7f9a0983c000, 2000, MS_SYNC|MS_INVALIDATE) = 0
[pid  4189] kill(4208, SIGUSR1)         = 0
[pid  4208] --- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=4189, si_uid=1000}
---
[pid  4189] read(0,  <unfinished ...>
[pid  4208] write(1, "w", 1)            = 1
[pid  4208] write(1, "r", 1)           = 1
[pid  4208] write(1, "t", 1)           = 1
[pid  4208] write(1, "d", 1)            = 1
[pid  4208] write(1, "f", 1)           = 1
[pid  4208] write(1, "f", 1)           = 1
[pid  4208] write(1, "\n", 1)          = 1
[pid  4208] rt_sigreturn({mask=[]})     = 0
qwerrteeeaannnnn
[pid  4189] <... read resumed>"qwerrteeeaannnnn\n", 1024) = 17
[pid  4189] msync(0x7f9a09802000, 2000, MS_SYNC|MS_INVALIDATE) = 0
[pid  4189] kill(4209, SIGUSR1)         = 0
[pid  4209] --- SIGUSR1 {si_signo=SIGUSR1, si_code=SI_USER, si_pid=4189, si_uid=1000}
---
[pid  4189] read(0,  <unfinished ...>
[pid  4209] write(1, "q", 1)           = 1
[pid  4209] write(1, "w", 1)            = 1
[pid  4209] write(1, "r", 1)           = 1
[pid  4209] write(1, "r", 1)           = 1
[pid  4209] write(1, "t", 1)           = 1
[pid  4209] write(1, "n", 1)           = 1
[pid  4209] write(1, "n", 1)           = 1
[pid  4209] write(1, "n", 1)           = 1
[pid  4209] write(1, "n", 1)           = 1
[pid  4209] write(1, "n", 1)           = 1
[pid  4209] write(1, "\n", 1)          = 1
[pid  4209] rt_sigreturn({mask=[]})     = 0
[pid  4189] <... read resumed>"", 1024) = 0
[pid  4189] kill(4208, SIGUSR2)         = 0
[pid  4208] --- SIGUSR2 {si_signo=SIGUSR2, si_code=SI_USER, si_pid=4189, si_uid=1000}
---
[pid  4189] kill(4209, SIGUSR2)         = 0
[pid  4209] --- SIGUSR2 {si_signo=SIGUSR2, si_code=SI_USER, si_pid=4189, si_uid=1000}
---
[pid  4208] exit_group(0 <unfinished ...>
[pid  4189] unlink("first_mmf.txt" <unfinished ...>
[pid  4209] exit_group(0 <unfinished ...>
[pid  4208] <... exit_group resumed>)   = ?
[pid  4209] <... exit_group resumed>)   = ?
[pid  4208] +++ exited with 0 +++
[pid  4209] +++ exited with 0 +++
```

```
<... unlink resumed>)                   = 0
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=4208, si_uid=1000,
si_status=0, si_utime=2508, si_stime=1} ---
unlink("second_mmf.txt")                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

В ходе выполнения данной лабораторной работы я узнал, что существуют memory-mapped файлы, которые являются отличной альтернативой pipe. Было очень интересно разбираться в том, как именно работают все системные вызовы и где хранятся созданные memory-mapped файлы. Уверен, эти знания помогут мне в понимании внутренних процессов компьютера во время выполнения более сложных задач.