

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»**

Выполнил: Н. В. Храмов  
Группа: М8О-208БВ-24  
Преподаватель: Е. С. Миронов

Москва, 2025

## Условие

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано как отображаемый в память файл для взаимодействия с child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процессы представлены разными исполняемыми файлами. Родительский процесс принимает от пользователя строки произвольной длины и записывает их в соответствующую область отображаемой памяти в зависимости от правила фильтрации. Процессы child1 и child2 читают данные из своих отображаемых файлов, обрабатывают строки и записывают результаты в файлы с суффиксом .out.

## Цель работы

Изучение механизмов создания процессов, организации межпроцессного взаимодействия через отображаемые в память файлы (memory-mapped files) и обработки данных в многопроцессной архитектуре.

## Задание

Правило фильтрации: строки длины больше 10 отправляются в отображаемый файл для child2, длиной 10 и менее — для child1. Дочерние процессы удаляют все гласные из строк.

## Вариант

17

## Метод решения

Данная программа реализует многопроцессную обработку текстовых данных с использованием отображаемых в память файлов (memory-mapped files) для межпроцессного взаимодействия. Основной алгоритм: родительский процесс читает строки из стандартного ввода и записывает строки длиной больше 10 символов во второй отображаемый файл, длиной 10 и менее — в первый. Каждый дочерний процесс читает данные из своего отображаемого файла, удаляет все гласные из строк и записывает результат в файл с суффиксом .out.

Ключевые компоненты:

MappedFile — обёртка над системными вызовами mmap, munmap, open, ftruncate

ChildProcess — логика работы дочернего процесса

ChildProcessor — удаление гласных букв

Системные вызовы:

Linux: mmap, munmap, open, ftruncate, fork, exec1

Программа использует объектно-ориентированный подход с инкапсуляцией.

## Описание программы

Программа реализует многопроцессную обработку текстовых данных через отображаемые в память файлы.

Родительский процесс читает строки из стандартного ввода и распределяет их между двумя дочерними процессами по правилу длины.  
Каждый дочерний процесс удаляет все гласные буквы и записывает результат в файл с суффиксом .out и в стандартный вывод.

Программа состоит из двух исполняемых файлов: `parent` и `child`.

`parent (src/parent.cpp)` — родительский процесс:

создаёт отображаемые файлы (`MappedFile`), порождает дочерние процессы (`fork/exec1`), считывает строки и записывает их в соответствующую область памяти.

`child (src/child.cpp)` — точка входа дочернего процесса:

через `ChildProcess` опрашивает отображаемую память,

удаляет гласные с помощью `ChildProcessor`

и записывает результат в файл и на экран.

## Результаты

Разработанная программа успешно реализует многопроцессную архитектуру для параллельной обработки текстовых данных.

В ходе решения были достигнуты следующие ключевые результаты:

Корректная работа системы межпроцессного взаимодействия

Реализованы два независимых канала передачи данных между родительским и дочерними процессами через отображаемые в память файлы

Обеспечено четкое распределение строк по принципу больше/меньше 10

Достигнута синхронизация процессов через опрос нулевого байта в общей памяти

Реализована унифицированная абстракция для работы с отображаемыми файлами через класс `MappedFile`

## Выводы

В ходе лабораторной работы успешно разработана многопроцессная система обработки текстовых данных с использованием межпроцессного взаимодействия через отображаемые в память файлы (`memory-mapped files`). Программа демонстрирует корректную работу на Unix-системах.

## Исходная программа

Листинг 1: src/parent.cpp — родительский процесс

```
1  #include "mapped_file.hpp"
2  #include <iostream>
3  #include <string>
4  #include <unistd.h>
5  #include <sys/wait.h>
6  #include <cstring>
7
8  int main()
9  {
10     std::string file1, file2;
11     std::getline(std::cin, file1);
12     std::getline(std::cin, file2);
13
14     const size_t BUFFER_SIZE = 4096;
15
16     MappedFile map1(file1, BUFFER_SIZE);
17     MappedFile map2(file2, BUFFER_SIZE);
18
19     if (!map1.isValid() || !map2.isValid())
20     {
21         std::cerr << "Failed to create mapped files" << std::endl;
22         return 1;
23     }
24
25     pid_t pid1 = fork();
26     if (pid1 == 0)
27     {
28         execl("./bin/child", "child", file1.c_str(), nullptr);
29         std::cerr << "Exec failed for child1" << std::endl;
30         _exit(1);
31     }
32     else if (pid1 < 0)
33     {
34         std::cerr << "Fork failed for child1" << std::endl;
35         return 1;
36     }
37
38     pid_t pid2 = fork();
39     if (pid2 == 0)
40     {
41         execl("./bin/child", "child", file2.c_str(), nullptr);
42         std::cerr << "Exec failed for child2" << std::endl;
43         _exit(1);
44     }
45     else if (pid2 < 0)
46     {
47         std::cerr << "Fork failed for child2" << std::endl;
48         return 1;
49     }
50
51     sleep(1);
52
53     char *buf1 = static_cast<char *>(map1.getData());
54     char *buf2 = static_cast<char *>(map2.getData());
```

```

55
56     std::string line;
57     while (std::getline(std::cin, line))
58     {
59         line += '\n';
60         char *target = (line.length() - 1 <= 10) ? buf1 : buf2;
61
62         while (target[0] != '\0')
63         {
64             usleep(1000);
65         }
66
67         std::strncpy(target, line.c_str(), BUFFER_SIZE - 1);
68         target[BUFFER_SIZE - 1] = '\0';
69     }
70
71     std::memset(buf1, 0, BUFFER_SIZE);
72     std::memset(buf2, 0, BUFFER_SIZE);
73
74     waitpid(pid1, nullptr, 0);
75     waitpid(pid2, nullptr, 0);
76
77     return 0;
78 }

```

Листинг 2: src/child.cpp — точка входа дочернего процесса

```

1  #include "childProcess.hpp"
2  #include <iostream>
3
4  int main(int argc, char *argv[])
5  {
6      if (argc != 2)
7      {
8          std::cerr << "Usage: " << argv[0] << " <mapped_filename>" << std::endl;
9          return 1;
10     }
11
12     ChildProcess child(argv[1]);
13     child.run();
14     return 0;
15 }

```

Листинг 3: src/childProcess.cpp — обработка в child

```

1  #include "childProcess.hpp"
2  #include "mapped_file.hpp"
3  #include <iostream>
4  #include <fstream>
5  #include <cstring>
6  #include <unistd.h>
7
8  ChildProcess::ChildProcess(const std::string &mapped_filename)
9      : mapped_filename_(mapped_filename) {}
10

```

```

11 void ChildProcess::run()
12 {
13     const size_t BUFFER_SIZE = 4096;
14     MappedFile mapped_file(mapped_filename_, BUFFER_SIZE);
15     if (!mapped_file.isValid())
16     {
17         std::cerr << "Child failed to map file: " << mapped_filename_ << std::endl;
18         return;
19     }
20
21     std::ofstream output_file(mapped_filename_ + ".out");
22     if (!output_file.is_open())
23     {
24         std::cerr << "Failed to open output file: " << mapped_filename_ << ".out" <<
                std::endl;
25         return;
26     }
27
28     char *buffer = static_cast<char *>(mapped_file.getData());
29     size_t pos = 0;
30
31     while (true)
32     {
33         if (buffer[pos] == '\0' && pos > 0)
34         {
35             std::string line(buffer, pos);
36             std::string processed = processor_.removeVowels(line);
37             std::cout << processed;
38             output_file << processed;
39             output_file.flush();
40
41             std::memset(buffer, 0, BUFFER_SIZE);
42             pos = 0;
43         }
44         else if (buffer[pos] != '\0')
45         {
46             if (++pos >= BUFFER_SIZE - 1)
47             {
48                 std::cerr << "Buffer overflow in child" << std::endl;
49                 break;
50             }
51         }
52         usleep(1000);
53     }
54 }

```

Листинг 4: src/childProcessor.cpp — удаление гласных

```

1 #include "childProcessor.hpp"
2 #include <algorithm>
3 #include <cctype>
4
5 std::string ChildProcessor::removeVowels(const std::string &input)
6 {
7     std::string result = input;
8     std::string vowels = "aeiouAEIOU";
9     result.erase(std::remove_if(result.begin(), result.end(),

```

```

10         [&vowels](char c)
11         { return vowels.find(c) != std::string::npos; }},
12         result.end());
13     return result;
14 }

```

Листинг 5: src/mapped\_file.cpp~mmap()

```

1  #include "mapped_file.hpp"
2  #include <iostream>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <sys/mman.h>
6  #include <sys/stat.h>
7
8  MappedFile::MappedFile(const std::string &filename, size_t size)
9      : fd_(-1), data_((void *)-1), size_(size)
10 {
11     fd_ = open(filename.c_str(), O_RDWR | O_CREAT | O_TRUNC, 0666);
12     if (fd_ == -1)
13     {
14         std::cerr << "Failed to open file for mapping: " << filename << std::endl;
15         return;
16     }
17
18     if (ftruncate(fd_, size) == -1)
19     {
20         std::cerr << "Failed to truncate file: " << filename << std::endl;
21         close(fd_);
22         fd_ = -1;
23         return;
24     }
25
26     data_ = mmap(nullptr, size, PROT_READ | PROT_WRITE, MAP_SHARED, fd_, 0);
27     if (data_ == MAP_FAILED)
28     {
29         std::cerr << "mmap failed for: " << filename << std::endl;
30         data_ = nullptr;
31         close(fd_);
32         fd_ = -1;
33     }
34 }
35
36 MappedFile::~MappedFile()
37 {
38     if (data_ != nullptr && data_ != (void *)-1)
39     {
40         munmap(data_, size_);
41     }
42     if (fd_ != -1)
43     {
44         close(fd_);
45     }
46 }

```

## Системные вызовы

```
37936 execve("./parent",["./parent"],0x7ffc9d70ab98 /* 37 vars */) = 0
37936 brk(NULL) = 0x642acffaa000
37936 mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) =
0x7fab9c263000
37936 access("/etc/ld.so.preload",R_OK) = -1 ENOENT (No such file or directory)
37936 openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC) = 3
37936 fstat(3,{st_mode=S_IFREG|0644,st_size=31547,...}) = 0
37936 mmap(NULL,31547,PROT_READ,MAP_PRIVATE,3,0) = 0x7fab9c25b000
37936 close(3) = 0
37936 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libstdc++.so.6",O_RDONLY|O_CLOEXEC)
= 3
37936 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"...,832)
= 832
37936 fstat(3,{st_mode=S_IFREG|0644,st_size=2592224,...}) = 0
37936 mmap(NULL,2609472,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7fab9be00000
37936 mmap(0x7fab9be9d000,1343488,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x7fab9be9d000
37936 mmap(0x7fab9bfe5000,552960,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x1e
= 0x7fab9bfe5000
37936 mmap(0x7fab9c06c000,57344,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x7fab9c06c000
37936 mmap(0x7fab9c07a000,12608,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_ANONYM
= 0x7fab9c07a000
37936 close(3) = 0
37936 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libgcc_s.so.1",O_RDONLY|O_CLOEXEC)
= 3
37936 read(3,"\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0"...,832)
= 832
37936 fstat(3,{st_mode=S_IFREG|0644,st_size=183024,...}) = 0
37936 mmap(NULL,185256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7fab9c22d000
37936 mmap(0x7fab9c231000,147456,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x7fab9c231000
37936 mmap(0x7fab9c255000,16384,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x280
= 0x7fab9c255000
37936 mmap(0x7fab9c259000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x7fab9c259000
37936 close(3) = 0
37936 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libc.so.6",O_RDONLY|O_CLOEXEC)
= 3
37936 read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"...,
= 832
37936 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"...,78
= 784
37936 fstat(3,{st_mode=S_IFREG|0755,st_size=2125328,...}) = 0
37936 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"...,78
= 784
37936 mmap(NULL,2170256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7fab9ba00000
```

```

37936 mmap(0x7fab9ba28000,1605632,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x7fab9ba28000
37936 mmap(0x7fab9bbb0000,323584,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x1b
= 0x7fab9bbb0000
37936 mmap(0x7fab9bbff000,24576,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x7fab9bbff000
37936 mmap(0x7fab9bc05000,52624,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_ANONYM
= 0x7fab9bc05000
37936 close(3) = 0
37936 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libm.so.6",O_RDONLY|O_CLOEXEC)
= 3
37936 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"...,832)
= 832
37936 fstat(3,{st_mode=S_IFREG|0644,st_size=952616,...}) = 0
37936 mmap(NULL,950296,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7fab9c144000
37936 mmap(0x7fab9c154000,520192,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x7fab9c154000
37936 mmap(0x7fab9c1d3000,360448,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x8f
= 0x7fab9c1d3000
37936 mmap(0x7fab9c22b000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRI
= 0x7fab9c22b000
37936 close(3) = 0
37936 mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) =
0x7fab9c142000
37936 mmap(NULL,12288,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x7fab9c13f000
37936 arch_prctl(ARCH_SET_FS,0x7fab9c13f740) = 0
37936 set_tid_address(0x7fab9c13fa10) = 37936
37936 set_robust_list(0x7fab9c13fa20,24) = 0
37936 rseq(0x7fab9c140060,0x20,0,0x53053053) = 0
37936 mprotect(0x7fab9bbff000,16384,PROT_READ) = 0
37936 mprotect(0x7fab9c22b000,4096,PROT_READ) = 0
37936 mprotect(0x7fab9c259000,4096,PROT_READ) = 0
37936 mprotect(0x7fab9c06c000,45056,PROT_READ) = 0
37936 mprotect(0x642ab5fba000,4096,PROT_READ) = 0
37936 mprotect(0x7fab9c29b000,8192,PROT_READ) = 0
37936 prlimit64(0,RLIMIT_STACK,NULL,{rlim_cur=8192*1024,rlim_max=RLIM64_INFINITY})
= 0
37936 munmap(0x7fab9c25b000,31547) = 0
37936 futex(0x7fab9c07a7bc,FUTEX_WAKE_PRIVATE,2147483647) = 0
37936 getrandom("\x22\xc6\x6c\x7d\x8f\xb9\xc8\x61",8,GRND_NONBLOCK) = 8
37936 brk(NULL) = 0x642acffaa000
37936 brk(0x642acffcb000) = 0x642acffcb000
37936 fstat(0,{st_mode=S_IFREG|0644,st_size=53,...}) = 0
37936 read(0,"short.txt\nlong.txt\nhello\nthis is"...,4096) = 53
37936 openat(AT_FDCWD,"short.txt",O_RDWR|O_CREAT|O_TRUNC,0666) = 3
37936 ftruncate(3,4096) = 0
37936 mmap(NULL,4096,PROT_READ|PROT_WRITE,MAP_SHARED,3,0) = 0x7fab9c262000

```

```

37936 openat(AT_FDCWD,"long.txt",O_RDWR|O_CREAT|O_TRUNC,0666) = 4
37936 ftruncate(4,4096) = 0
37936 mmap(NULL,4096,PROT_READ|PROT_WRITE,MAP_SHARED,4,0) = 0x7fab9c261000
37936 clone(child_stack=NULL,flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,ch
= 37937
37937 set_robust_list(0x7fab9c13fa20,24 <unfinished ...>
37936 clone(child_stack=NULL,flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD
<unfinished ...>
37937 <... set_robust_list resumed>) = 0
37937 execve("./bin/child",["child","short.txt"],0x7ffebc19a158 /* 37 vars
*/ <unfinished ...>
37936 <... clone resumed>,child_tidptr=0x7fab9c13fa10) = 37938
37938 set_robust_list(0x7fab9c13fa20,24 <unfinished ...>
37937 <... execve resumed>) = -1 ENOENT (No such file or directory)
37938 <... set_robust_list resumed>) = 0
37936 clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=1,tv_nsec=0}, <unfinished ...>
37938 execve("./bin/child",["child","long.txt"],0x7ffebc19a158 /* 37 vars */
<unfinished ...>
37937 write(2,"Exec failed for child1",22 <unfinished ...>
37938 <... execve resumed>) = -1 ENOENT (No such file or directory)
37937 <... write resumed>) = 22
37937 write(2,"\n",1 <unfinished ...>
37938 write(2,"Exec failed for child2",22) = 22
37937 <... write resumed>) = 1
37937 exit_group(1 <unfinished ...>
37938 write(2,"\n",1 <unfinished ...>
37937 <... exit_group resumed>) = ?
37938 <... write resumed>) = 1
37938 exit_group(1 <unfinished ...>
37937 +++ exited with 1 +++
37938 <... exit_group resumed>) = ?
37936 <... clock_nanosleep resumed>{tv_sec=0,tv_nsec=999369899}) = ? ERESTART_RESTART
(Interrupted by signal)
37936 ---SIGCHLD {si_signo=SIGCHLD,si_code=CLD_EXITED,si_pid=37937,si_uid=1000,si_sta
---
37938 +++ exited with 1 +++
37936 ---SIGCHLD {si_signo=SIGCHLD,si_code=CLD_EXITED,si_pid=37938,si_uid=1000,si_sta
---
37936 restart_syscall(<... resuming interrupted clock_nanosleep ...>) = 0
37936 read(0,"",4096) = 0
37936 clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=0,tv_nsec=1000000},NULL) = 0

```