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: Т. В. Балдынов
Группа: М8О-208БВ-24
Преподаватель: Е. С. Миронов

Москва, 2025

Содержание

1 Условие **2**

2 Метод решения **3**

3 Фрагмент кода **3**

4 Результаты **9**

5 Выводы **10**

5.1 Анализ системных вызовов 10

5.2 Логи выполнения программы 10

1. Условие

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файлов, которые будут использованы для открытия файлов с таким именем на запись. Родительский и дочерний процесс должны быть представлены разными про- граммами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в дочерние процессы через системные сигналы/события и/или через отображаемые файлы (memory-mapped files). Дочерние процессы удаляют из строк гласные буквы и записывают строки в файлы, заданные в начале. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода.

Цель работы

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

Задание

Дочерние процессы удаляют из строк гласные буквы и записывают строки в файлы, заданные в начале.

Вариант

18

2. Метод решения

Данная программа реализует многопроцессную обработку текстовых данных с помощью shared memory: Ключевые компоненты:
• Shared memory − Linux
Системные вызовы:
• Windows: CreatePipe, CreateProcess, ReadFile, WriteFile
• Linux: pipe, fork, excel, read, write
Программа обеспечивает кроссплатформенность и четкое разделение ответственности между модулями.

3. Фрагмент кода

os.cpp:

```
#include <iostream>
#include <cstdlib>
#include <cstdio>
#include <unistd.h>
#include <string>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <semaphore.h>
#include <cstring>
#include <sys/wait.h>

#include "os.hpp"

SharedData* CreateSharedMemory(const char* name) {
    int shm_fd = shm_open(name, O_CREAT | O_RDWR, 0666);
    if (shm_fd == -1) {
        perror("shm_open");
        exit(1);
    }
    ftruncate(shm_fd, sizeof(SharedData));
    SharedData* data = (SharedData*)mmap(NULL, sizeof(SharedData
    ), PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
    if (data == MAP_FAILED) {
        perror("mmap");
        exit(1);
    }
    close(shm_fd);
    sem_init(&data->sem_file, 1, 1);
    sem_init(&data->sem_parent, 1, 0);
    sem_init(&data->sem_child, 1, 0);
```

```
    return data;
}
SharedData* OpenSharedMemory(const char* name) {
    int shm_fd = shm_open(name, O_RDWR, 0);
    if (shm_fd == -1) {
        perror("shm_open");
        exit(1);
    }
    SharedData* data = (SharedData*)mmap(NULL, sizeof(SharedData
    ), PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
    if (data == MAP_FAILED) {
        perror("mmap");
        exit(1);
    }
    close(shm_fd);
    return data;
}
void DestroySharedMemory(const char* name, SharedData* data) {
    sem_destroy(&data->sem_file);
    sem_destroy(&data->sem_parent);
    sem_destroy(&data->sem_child);
    munmap(data, sizeof(SharedData));
    shm_unlink(name);
}
ProcessRole ProcessCreate() {
    pid_t pid = fork();
    if (pid == -1) {
        std::cout << "Ошибка создания process" << std::endl;
        exit(-1);
    }
    if (pid == 0) {
        return IS_CHILD;
    }
    return IS_PARENT;
}
void ProcessExecute(const char* program, const char* arg, const
    char* shared_memory_name) {
    execl(program, arg, shared_memory_name, NULL);
    perror("execl");
    exit(1);
}
void SemaphorePost(sem_t* sem) {
    if (sem_post(sem) == -1) {
        perror("sem_post");
        exit(1);
    }
}
void SemaphoreWait(sem_t* sem) {
    if (sem_wait(sem) == -1) {
```

```
78          perror("sem_wait");
79          exit(1);
80      }
81  }
```

Listing 1: os.cpp

Parent.cpp - Код родительского процесса:

```
1  #include <iostream>
2  #include <cstdlib>
3  #include <cstdio>
4  #include <unistd.h>
5  #include <string>
6  #include <cstring>
7  #include <sys/wait.h>
8
9  #include "os.hpp"
10
11 int main() {
12     SharedData* data1 = CreateSharedMemory("/shared1");
13     SharedData* data2 = CreateSharedMemory("/shared2");
14
15     if (!data1 || !data2) {
16         std::cout << "Error creating shared memory" << std::endl
       ;
17         return 1;
18     }
19
20     std::cout << "Input file for child1: ";
21     std::string file1;
22     std::cin >> file1;
23
24     std::cout << "Input file for child2: ";
25     std::string file2;
26     std::cin >> file2;
27
28     strncpy(data1->file, file1.c_str(), sizeof(data1->file));
29     data1->signal = 0;
30     data1->number = 0;
31
32     strncpy(data2->file, file2.c_str(), sizeof(data2->file));
33     data2->signal = 0;
34     data2->number = 0;
35
36     ProcessRole role1 = ProcessCreate();
37     if (role1 == IS_CHILD) {
38         ProcessExecute("./child", "child", "/shared1");
39     }
40
```

```
41      ProcessRole role2 = ProcessCreate();
42      if (role2 == IS_CHILD) {
43          ProcessExecute("./child", "child", "/shared2");
44      }
45
46      if (role1 == IS_PARENT && role2 == IS_PARENT) {
47          std::cout << "Child1 started with PID: " << getpid() + 1
        << ", file: " << file1 << std::endl;
48          std::cout << "Child2 started with PID: " << getpid() + 2
        << ", file: " << file2 << std::endl;
49
50          SemaphoreWait(&data1->sem_parent);
51          SemaphoreWait(&data2->sem_parent);
52
53          std::cout << "\nWrite strings (Ctrl+D to finish):\n";
54          std::string input;
55
56          int cnt = 0;
57          while (std::cin >> input) {
58              SharedData* currentData = (cnt % 2 == 0) ? data1 :
        data2;
59
60              strncpy(currentData->file, input.c_str(), sizeof(
        currentData->file) - 1);
61              currentData->file[sizeof(currentData->file) - 1] = '
        \0';
62              currentData->signal = 0;
63
64              SemaphorePost(&currentData->sem_child);
65              SemaphoreWait(&currentData->sem_parent);
66
67              cnt++;
68          }
69
70          data1->signal = 1;
71          data2->signal = 1;
72          SemaphorePost(&data1->sem_child);
73          SemaphorePost(&data2->sem_child);
74
75          wait(NULL);
76          wait(NULL);
77
78          DestroySharedMemory("/shared1", data1);
79          DestroySharedMemory("/shared2", data2);
80
81          std::cout << "Parent process finished. Processed " <<
        cnt << " lines." << std::endl;
82      }
83
```

```
84      return 0;
85  }
```

Listing 2: Parent.cpp

Child.cpp - Код дочернего процесса:

```
1   #include <iostream>
2   #include <fstream>
3   #include <string>
4   #include <unistd.h>
5   #include <set>
6   #include <cstring>
7   #include <cctype>
8
9   #include "os.hpp"
10
11  std::string removeVowels(const std::string& str) {
12      std::string res;
13      std::set<char> ms = {'a', 'e', 'i', 'o', 'u', 'y'};
14
15      for (char c : str) {
16          if (c != 0 && ms.find(tolower(c)) == ms.end()) {
17              res.push_back(c);
18          }
19      }
20
21      return res;
22  }
23
24  int main(int argc, char** argv) {
25      SharedData* data = OpenSharedMemory(argv[1]);
26      if (!data) {
27          std::cout << "Error opening shared memory" << std::endl;
28          return 1;
29      }
30
31      std::string filename(data->file);
32      std::ofstream out(filename, std::ios::app);
33      if (!out.is_open()) {
34          std::cout << "Error, can't open " << filename << std::
    endl;
35          SemaphorePost(&data->sem_parent);
36          return 1;
37      }
38
39      SemaphorePost(&data->sem_parent);
40
41      bool running = true;
42      while (running) {
```

```
43          SemaphoreWait(&data->sem_child);
44
45          if (data->signal == 1) {
46              running = false;
47          } else if (data->signal == 0) {
48              std::string input(data->file);
49              if (!input.empty()) {
50                  std::string result = removeVowels(input);
51
52                  out << result << std::endl;
53                  std::cout << "Child " << getpid() << " processed
    : " << input
54                            << " -> " << result << std::endl;
55              }
56          }
57
58          SemaphorePost(&data->sem_parent);
59      }
60
61      out.close();
62      std::cout << "Child process " << getpid() << " finished." <<
        std::endl;
63      return 0;
64  }
```

Listing 3: Child.cpp

4. 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 дочер- ним процессами
• Обеспечено изменение строк по заданному правилу
• Кросс-платформенная функциональность
• Программа корректно работает как в Windows, так и в Linux/Unix системах
• Реализована унифицированная абстракция для взаимодействия между процессами через системные сигналы/события и/или через отображаемые файлы (memory-mapped files)
• Обеспечен единый интерфейс для создания процессов на разных платформах

5. Выводы

В ходе лабораторной работы успешно разработана многопроцессная система обработки тек- стовых данных с использованием межпроцессного взаимодействия через системные сигналы/события и/или через отображаемые файлы (memory-mapped files). Программа демонстрирует корректную работу в кросс-платформенном режиме на Windows и Unix системах.

5.1. Анализ системных вызовов

Для анализа работы программы был использован инструмент strace. Ниже будут представлены ключевые системные вызовы, используемые программой: Анализ системных вызовов подтверждает корректную работу программы:
• Созданы два канала для межпроцессного взаимодействия
• Корректно созданы дочерние процессы
• Осуществляется передача данных между процессами
• Программа правильно обрабатывает строки

5.2. Логи выполнения программы

output.txt - Логи:

```
execve("./parent", ["./parent"], 0x7ffd65371580 /* 97 vars */) =
    0
brk(NULL)                                       = 0x5774725ef000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc6442c1c0) = -1 EINVAL (
    Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d889062000
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=78560, ...},
    AT_EMPTY_PATH) = 0
mmap(NULL, 78560, PROT_READ, MAP_PRIVATE, 3, 0) = 0x75d88904e000
close(3)                                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libasan.so.6", O_RDONLY|
    O_CLOEXEC) = 3
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\0\0"...,
    832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=7619608, ...},
    AT_EMPTY_PATH) = 0
```

```
13 mmap(NULL, 10391048, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0)
       = 0x75d888600000
14 mprotect(0x75d888624000, 1089536, PROT_NONE) = 0
15 mmap(0x75d888624000, 880640, PROT_READ|PROT_EXEC, MAP_PRIVATE|
     MAP_FIXED|MAP_DENYWRITE, 3, 0x24000) = 0x75d888624000
16 mmap(0x75d8886fb000, 204800, PROT_READ, MAP_PRIVATE|MAP_FIXED|
     MAP_DENYWRITE, 3, 0xfb000) = 0x75d8886fb000
17 mmap(0x75d88872e000, 28672, PROT_READ|PROT_WRITE, MAP_PRIVATE|
     MAP_FIXED|MAP_DENYWRITE, 3, 0x12d000) = 0x75d88872e000
18 mmap(0x75d888735000, 9125384, PROT_READ|PROT_WRITE, MAP_PRIVATE|
     MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x75d888735000
19 close(3)                                        = 0
20 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6",
     O_RDONLY|O_CLOEXEC) = 3
21 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
22 newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2260296, ...},
     AT_EMPTY_PATH) = 0
23 mmap(NULL, 2275520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0)
     = 0x75d888200000
24 mprotect(0x75d88829a000, 1576960, PROT_NONE) = 0
25 mmap(0x75d88829a000, 1118208, PROT_READ|PROT_EXEC, MAP_PRIVATE|
     MAP_FIXED|MAP_DENYWRITE, 3, 0x9a000) = 0x75d88829a000
26 mmap(0x75d8883ab000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|
     MAP_DENYWRITE, 3, 0x1ab000) = 0x75d8883ab000
27 mmap(0x75d88841b000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|
     MAP_FIXED|MAP_DENYWRITE, 3, 0x21a000) = 0x75d88841b000
28 mmap(0x75d888429000, 10432, PROT_READ|PROT_WRITE, MAP_PRIVATE|
     MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x75d888429000
29 close(3)                                        = 0
30 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|
     O_CLOEXEC) = 3
31 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
32 newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=940560, ...},
     AT_EMPTY_PATH) = 0
33 mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
       0x75d888519000
34 mmap(0x75d888527000, 507904, PROT_READ|PROT_EXEC, MAP_PRIVATE|
     MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x75d888527000
35 mmap(0x75d8885a3000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|
     MAP_DENYWRITE, 3, 0x8a000) = 0x75d8885a3000
36 mmap(0x75d8885fe000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
     MAP_FIXED|MAP_DENYWRITE, 3, 0xe4000) = 0x75d8885fe000
37 close(3)                                        = 0
38 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY
     |O_CLOEXEC) = 3
```

```
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\0"...,
    832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=125488, ...},
    AT_EMPTY_PATH) = 0
mmap(NULL, 127720, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
     0x75d88902e000
mmap(0x75d889031000, 94208, PROT_READ|PROT_EXEC, MAP_PRIVATE|
    MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x75d889031000
mmap(0x75d889048000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x1a000) = 0x75d889048000
mmap(0x75d88904c000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_DENYWRITE, 3, 0x1d000) = 0x75d88904c000
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\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\0O{\f
    \225\\=\201\327\312\301P\32$\230\266\235"..., 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)
    = 0x75d887e00000
mprotect(0x75d887e28000, 2023424, PROT_NONE) = 0
mmap(0x75d887e28000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|
    MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x75d887e28000
mmap(0x75d887fbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x1bd000) = 0x75d887fbd000
mmap(0x75d888016000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x75d888016000
mmap(0x75d88801c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x75d88801c000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d88902c000
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_ANONYMOUS, -1, 0) = 0x75d889029000
arch_prctl(ARCH_SET_FS, 0x75d8890297c0) = 0
set_tid_address(0x75d889029a90)         = 18984
set_robust_list(0x75d889029aa0, 24)     = 0
rseq(0x75d88902a160, 0x20, 0, 0x53053053) = 0
mprotect(0x75d888016000, 16384, PROT_READ) = 0
```

```
mprotect(0x75d88904c000, 4096, PROT_READ) = 0
mprotect(0x75d8885fe000, 4096, PROT_READ) = 0
mprotect(0x75d88841b000, 45056, PROT_READ) = 0
mprotect(0x75d88872e000, 16384, PROT_READ) = 0
mprotect(0x57743524d000, 4096, PROT_READ) = 0
mprotect(0x75d88909c000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=
    RLIM64_INFINITY}) = 0
munmap(0x75d88904e000, 78560)           = 0
readlink("/proc/self/exe", "/home/hom1e/codes/OC/lab3/build
    /"..., 4096) = 38
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d88909b000
open("/proc/self/cmdline", O_RDONLY)    = 3
read(3, "./parent\0", 4096)             = 9
read(3, "", 4087)                       = 0
close(3)                                = 0
munmap(0x75d88909b000, 4096)            = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d88909b000
open("/proc/self/environ", O_RDONLY)    = 3
read(3, "SHELL=/bin/bash\0SESSION_MANAGER="..., 4096) = 4096
close(3)                                = 0
munmap(0x75d88909b000, 4096)            = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d889060000
open("/proc/self/environ", O_RDONLY)    = 3
read(3, "SHELL=/bin/bash\0SESSION_MANAGER="..., 8192) = 5638
read(3, "", 2554)                       = 0
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d88905e000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d88905c000
mmap(NULL, 3481600, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_ANONYMOUS, -1, 0) = 0x75d887aae000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d88909b000
prlimit64(0, RLIMIT_CORE, NULL, {rlim_cur=0, rlim_max=
    RLIM64_INFINITY}) = 0
prlimit64(0, RLIMIT_CORE, {rlim_cur=0, rlim_max=RLIM64_INFINITY
    }, NULL) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d88905b000
open("/proc/self/maps", O_RDONLY)       = 3
read(3, "577435246000-577435249000 r--p 0"..., 4096) = 4028
read(3, "75d889064000-75d889066000 r--p 0"..., 68) = 68
close(3)                                = 0
munmap(0x75d88905b000, 4096)            = 0
```

```
104 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d88905a000
105 open("/proc/self/maps", O_RDONLY)        = 3
106 read(3, "577435246000-577435249000 r--p 0"..., 8192) = 4028
107 read(3, "75d889064000-75d889066000 r--p 0"..., 4164) = 974
108 read(3, "", 3190)                        = 0
109 close(3)                                 = 0
110 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d889059000
111 open("/proc/self/maps", O_RDONLY)        = 3
112 read(3, "577435246000-577435249000 r--p 0"..., 4096) = 4028
113 read(3, "75d889064000-75d889066000 r--p 0"..., 68) = 68
114 close(3)                                 = 0
115 munmap(0x75d889059000, 4096)             = 0
116 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d889058000
117 open("/proc/self/maps", O_RDONLY)        = 3
118 read(3, "577435246000-577435249000 r--p 0"..., 8192) = 4028
119 read(3, "75d889064000-75d889066000 r--p 0"..., 4164) = 974
120 read(3, "", 3190)                        = 0
121 close(3)                                 = 0
122 munmap(0x75d889058000, 8192)             = 0
123 mmap(0x7fff7000, 268435456, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS|MAP_NORESERVE, -1, 0) = 0x7fff7000
124 madvise(0x7fff7000, 268435456, MADV_NOHUGEPAGE) = 0
125 madvise(0x7fff7000, 268435456, MADV_DONTDUMP) = 0
126 mmap(0x2008fff7000, 15392894357504, PROT_READ|PROT_WRITE,
    MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS|MAP_NORESERVE, -1, 0) = 0
    x2008fff7000
127 madvise(0x2008fff7000, 15392894357504, MADV_NOHUGEPAGE) = 0
128 madvise(0x2008fff7000, 15392894357504, MADV_DONTDUMP) = 0
129 mmap(0x8fff7000, 2199023255552, PROT_NONE, MAP_PRIVATE|MAP_FIXED
    |MAP_ANONYMOUS|MAP_NORESERVE, -1, 0) = 0x8fff7000
130 sigaltstack(NULL, {ss_sp=NULL, ss_flags=SS_DISABLE, ss_size=0})
    = 0
131 mmap(NULL, 61440, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_ANONYMOUS, -1, 0) = 0x75d88901a000
132 sigaltstack({ss_sp=0x75d88901a000, ss_flags=0, ss_size=58112},
    NULL) = 0
133 rt_sigaction(SIGSEGV, {sa_handler=0x75d8886ba3e0, sa_mask=[],
    sa_flags=SA_RESTORER|SA_ONSTACK|SA_NODEFER|SA_SIGINFO,
    sa_restorer=0x75d887e42520}, NULL, 8) = 0
134 rt_sigaction(SIGBUS, {sa_handler=0x75d8886ba3e0, sa_mask=[],
    sa_flags=SA_RESTORER|SA_ONSTACK|SA_NODEFER|SA_SIGINFO,
    sa_restorer=0x75d887e42520}, NULL, 8) = 0
135 rt_sigaction(SIGFPE, {sa_handler=0x75d8886ba3e0, sa_mask=[],
    sa_flags=SA_RESTORER|SA_ONSTACK|SA_NODEFER|SA_SIGINFO,
    sa_restorer=0x75d887e42520}, NULL, 8) = 0
136 mmap(0x500000000000, 4398046523392, PROT_NONE, MAP_PRIVATE|
```

```
    MAP_FIXED|MAP_ANONYMOUS|MAP_NORESERVE, -1, 0) = 0
    x500000000000
137 mmap(0x540000000000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x540000000000
138 mmap(NULL, 8388608, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|
    MAP_NORESERVE, -1, 0) = 0x75d887200000
139 mmap(NULL, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_ANONYMOUS, -1, 0) = 0x75d88900c000
140 mmap(NULL, 33554432, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_ANONYMOUS, -1, 0) = 0x75d885200000
141 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d889059000
142 getpid()                                 = 18984
143 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=
    RLIM64_INFINITY}) = 0
144 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d889058000
145 open("/proc/self/maps", O_RDONLY)        = 3
146 read(3, "7fff7000-8fff7000 rw-p 00000000 "..., 4096) = 4082
147 read(3, "75d88904c000-7", 14)            = 14
148 close(3)                                 = 0
149 munmap(0x75d889058000, 4096)             = 0
150 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d889057000
151 open("/proc/self/maps", O_RDONLY)        = 3
152 read(3, "7fff7000-8fff7000 rw-p 00000000 "..., 8192) = 4082
153 read(3, "75d88904c000-75d88904d000 r--p 0"..., 4110) = 1249
154 read(3, "", 2861)                        = 0
155 close(3)                                 = 0
156 munmap(0x75d88905a000, 8192)             = 0
157 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d88905b000
158 open("/proc/self/maps", O_RDONLY)        = 3
159 read(3, "7fff7000-8fff7000 rw-p 00000000 "..., 4096) = 4082
160 read(3, "75d88904c000-7", 14)            = 14
161 close(3)                                 = 0
162 munmap(0x75d88905b000, 4096)             = 0
163 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d88905a000
164 open("/proc/self/maps", O_RDONLY)        = 3
165 read(3, "7fff7000-8fff7000 rw-p 00000000 "..., 8192) = 4082
166 read(3, "75d88904c000-75d88904d000 r--p 0"..., 4110) = 1249
167 read(3, "", 2861)                        = 0
168 close(3)                                 = 0
169 munmap(0x75d88905a000, 8192)             = 0
170 mmap(0x10000c77e000, 1044480, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS|MAP_NORESERVE, -1, 0) = 0
    x10000c77e000
171 madvise(0x10000c77e000, 1044480, MADV_NOHUGEPAGE) = 0
```

```
madvise(0x10000c77e000, 1044480, MADV_DONTDUMP) = 0
sigaltstack(NULL, {ss_sp=0x75d88901a000, ss_flags=0, ss_size=58112}) = 0
mmap(NULL, 2097152, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d885000000
munmap(0x75d885100000, 1048576)          = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d88905b000
mmap(NULL, 2097152, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d884e00000
munmap(0x75d884f00000, 1048576)          = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d88905a000
mmap(NULL, 1179648, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d8880e0000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d889056000
mmap(NULL, 2097152, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d884c00000
munmap(0x75d884d00000, 1048576)          = 0
mmap(NULL, 2097152, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d884a00000
munmap(0x75d884b00000, 1048576)          = 0
munmap(0x75d889056000, 4096)             = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d889056000
munmap(0x75d889056000, 4096)             = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d889056000
munmap(0x75d889056000, 4096)             = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d889056000
munmap(0x75d889056000, 4096)             = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d889056000
munmap(0x75d889056000, 4096)             = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d889056000
munmap(0x75d889056000, 4096)             = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d889056000
munmap(0x75d889056000, 4096)             = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d889056000
munmap(0x75d889056000, 4096)             = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d889056000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75d889055000
```

```
203 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d889053000
204 clock_gettime(CLOCK_MONOTONIC, {tv_sec=751, tv_nsec=392248304})
    = 0
205 mmap(0x507000000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x507000000000
206 mmap(0x507e00000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x507e00000000
207 mmap(NULL, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_ANONYMOUS, -1, 0) = 0x75d888ffc000
208 clock_gettime(CLOCK_MONOTONIC, {tv_sec=751, tv_nsec=392373886})
    = 0
209 mmap(0x503000000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x503000000000
210 mmap(0x503e00000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x503e00000000
211 clock_gettime(CLOCK_MONOTONIC, {tv_sec=751, tv_nsec=392507435})
    = 0
212 mmap(0x50b000000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x50b000000000
213 mmap(0x50be00000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x50be00000000
214 clock_gettime(CLOCK_MONOTONIC, {tv_sec=751, tv_nsec=392669510})
    = 0
215 mmap(0x524000000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x524000000000
216 mmap(0x524e00000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x524e00000000
217 mmap(0x531000000000, 131072, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x531000000000
218 mmap(0x531e00000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x531e00000000
219 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d889052000
220 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d889051000
221 mmap(NULL, 36864, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_ANONYMOUS, -1, 0) = 0x75d888ff3000
222 futex(0x75d88842977c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
223 openat(AT_FDCWD, "/dev/shm/shared1", O_RDWR|O_CREAT|O_NOFOLLOW|
    O_CLOEXEC, 0666) = 3
224 ftruncate(3, 360)                          = 0
225 mmap(NULL, 360, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0
    x75d889050000
226 close(3)                                   = 0
227 openat(AT_FDCWD, "/dev/shm/shared2", O_RDWR|O_CREAT|O_NOFOLLOW|
    O_CLOEXEC, 0666) = 3
228 ftruncate(3, 360)                          = 0
229 mmap(NULL, 360, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0
```

```
    x75d88904f000
close(3)                                = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0
    x2), ...}, AT_EMPTY_PATH) = 0
mmap(0x519000000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x519000000000
mmap(0x519e00000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x519e00000000
write(1, "Input file for child1: ", 23) = 23
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0
    x2), ...}, AT_EMPTY_PATH) = 0
read(0, "file1\n", 1024)                = 6
write(1, "Input file for child2: ", 23) = 23
read(0, "file2\n", 1024)                = 6
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|
    CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x75d889029a90) =
    19044
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|
    CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x75d889029a90) =
    19045
getpid()                                = 18984
write(1, "Child1 started with PID: 18985, "..., 44) = 44
getpid()                                = 18984
write(1, "Child2 started with PID: 18986, "..., 44) = 44
futex(0x75d889050108, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
     NULL, FUTEX_BITSET_MATCH_ANY) = 0
futex(0x75d88904f108, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
     NULL, FUTEX_BITSET_MATCH_ANY) = 0
write(1, "\nWrite strings (Ctrl+D to finish"..., 35) = 35
read(0, "asdasd\n", 1024)               = 7
futex(0x75d889050128, FUTEX_WAKE, 1)    = 1
futex(0x75d889050108, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
     NULL, FUTEX_BITSET_MATCH_ANY) = 0
read(0, "ddddd\n", 1024)                = 6
futex(0x75d88904f128, FUTEX_WAKE, 1)    = 1
futex(0x75d88904f108, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 0,
     NULL, FUTEX_BITSET_MATCH_ANY) = 0
read(0, "", 1024)                       = 0
futex(0x75d889050128, FUTEX_WAKE, 1)    = 1
futex(0x75d88904f128, FUTEX_WAKE, 1)    = 1
wait4(-1, NULL, 0, NULL)                = 19044
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=19044,
     si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
wait4(-1, NULL, 0, NULL)                = 19045
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=19045,
     si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
munmap(0x75d889050000, 360)             = 0
unlink("/dev/shm/shared1")              = 0
munmap(0x75d88904f000, 360)             = 0
```

```
264 unlink("/dev/shm/shared2")                  = 0
265 write(1, "Parent process finished. Process"..., 44) = 44
266 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS
    , -1, 0) = 0x75d889050000
267 gettid()                                    = 18984
268 prctl(PR_GET_DUMPABLE)                      = 1 (SUID_DUMP_USER)
269 getpid()                                    = 18984
270 mmap(NULL, 2101248, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_ANONYMOUS, -1, 0) = 0x75d8847ff000
271 mprotect(0x75d8847ff000, 4096, PROT_NONE) = 0
272 rt_sigprocmask(SIG_BLOCK, ~[ILL ABRT BUS FPE SEGV XCPU XFSZ],
    [], 8) = 0
273 clone(child_stack=0x75d8849ffff0, flags=CLONE_VM|CLONE_FS|
    CLONE_FILES|CLONE_UNTRACED) = 19156
274 rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
275 getpid()                                    = 18984
276 prctl(PR_SET_PTRACER, 19156)                = 0
277 sched_yield()                               = 0
278 sched_yield()                               = 0
279 sched_yield()                               = 0
280 sched_yield()                               = 0
281 wait4(19156, NULL, __WALL, NULL)            = 19156
282 munmap(0x75d8847ff000, 2101248)             = 0
283 getpid()                                    = 18984
284 write(2, "==18984==LeakSanitizer has encou"..., 54) = 54
285 mmap(NULL, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|
    MAP_ANONYMOUS, -1, 0) = 0x75d888509000
286 getpid()                                    = 18984
287 write(2, "==18984==HINT: For debugging, tr"..., 102) = 102
288 getpid()                                    = 18984
289 write(2, "==18984==HINT: LeakSanitizer doe"..., 75) = 75
290 exit_group(1)                               = ?
291 +++ exited with 1 +++
```

Listing 4: output.txt