МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №1
по курсу «Операционные системы»

Выполнил:
Павлова Ю.В.
Группа: М80-207БВ-24

Преподаватель:
Е.С. Миронов

Москва
2025

1 Условие

Цель работы: Приобретение практических навыков в:

- Управлении процессами в ОС
- Обеспечение обмена данных между процессами посредством каналов

Задание: Составить и отладить программу на языке C++, осуществляющую работу с процессами и взаимодействие между ними в операционной системе. В результате работы программа (основной процесс) должен создать для решения задачи один дочерний процесс. Взаимодействие между процессами осуществляется через каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Вариант 3: Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс передает команды пользователя через pipe, который связан со стандартным входным потоком дочернего процесса.

Пользователь вводит команды вида: «число число число». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс производит деление первого числа на последующие, а результат выводит в файл. Если происходит деление на 0, то тогда дочерний и родительский процесс завершают свою работу. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. Числа имеют тип int. Количество чисел может быть произвольным.

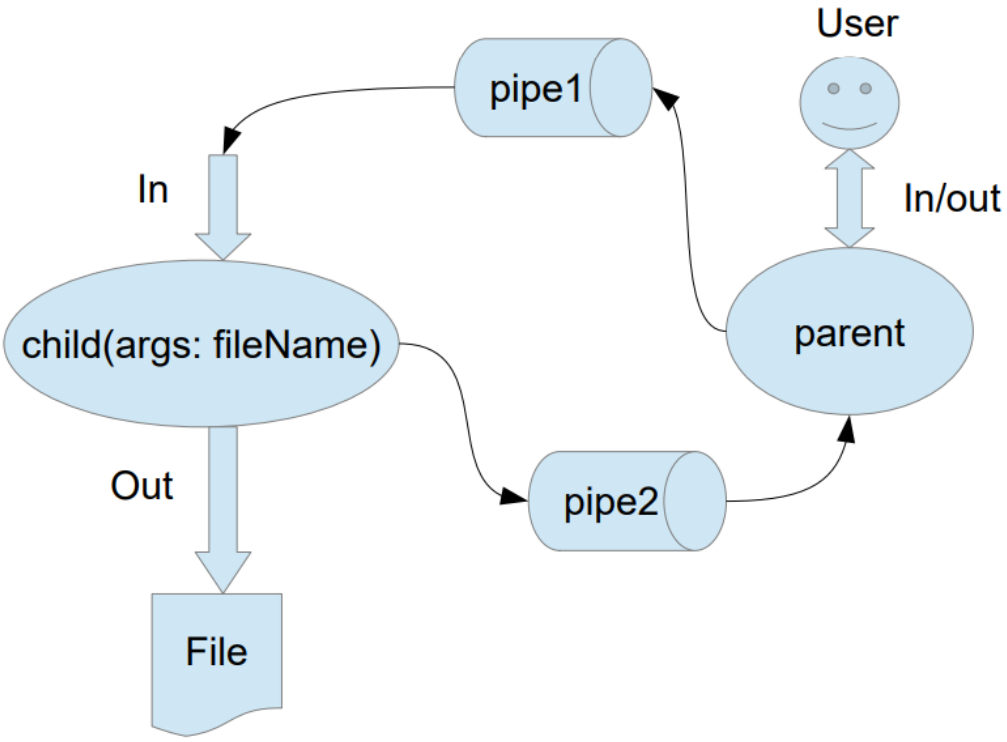

Рис. 1: Схема работы процессов

2 Метод решения

Алгоритм решения задачи:

1. Пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия файла дочерним процессом.
2. Создается объект класса Parent (родительский процесс).
3. Родительский процесс создает канал (pipe) для передачи данных между родительским и дочерним процессами. Затем делает fork для создания дочернего процесса.
4. Дочерний процесс закрывает соответствующий канал для записи, перенаправляет конец канала для чтения на стандартный ввод, закрывает второй конец канала.
5. Дочерний процесс пытается запустить бинарный файл для дочернего процесса, если не получилось — завершается.
6. Родительский процесс закрывает конец канала для чтения и отправляет имя файла для его открытия дочернему процессу.
7. Дочерний процесс пытается открыть указанный файл, если не получилось — процесс завершается.
8. Родительский процесс считывает пользовательский ввод и отправляет дочернему процессу через pipe.
9. Дочерний процесс читает строки чисел, производит деление первого числа на последующие, а результат выводит в файл.
10. Если происходит деление на 0, то дочерний процесс завершает свою работу с ошибкой, после чего родительский процесс также завершает работу.
11. По команде "exit"родитель останавливает дочерний процесс и программа завершается.

3 Архитектура программы

3.1 Структура проекта

```
lab1_os
|-- app
|   |-- childmain.cpp
|   |-- parentmain.cpp
|-- include
|   |-- child.h
|   |-- os.h
|   |-- parent.h
|-- src
```

```
|   |-- child.cpp
|   |-- oswin.cpp
|   |-- os.cpp
|   |-- parent.cpp
|-- main.tex
```

3.2 Назначение файлов

- **app/childmain.cpp** — точка входа в программу для дочернего процесса
- **app/parentmain.cpp** — точка входа в программу для родительского процесса
- **include/child.h** — объявление класса Child
- **include/parent.h** — объявление класса Parent
- **include/os.h** — объявление функций управления процессами ОС
- **src/child.cpp** — реализация класса Child
- **src/parent.cpp** — реализация класса Parent
- **src/oswin.cpp** — реализация функций для работы с процессами и каналами в Windows
- **src/os.cpp** — реализация функций для работы с процессами и каналами в Linux

4 Описание программы

Класс Child

Поля класса

- `std::string filename` — название файла для открытия
- `FILE* file` — указатель на файл для записи

Основные методы

- `Child(const std::string& filename_arg)` — конструктор, открывающий файл
- `void ProcessDivision()` — получает из канала строки, выполняет деление и записывает в файл
- `~Child()` — деструктор, закрывающий файл

Вспомогательные методы

- `std::vector<int> parse_numbers(const std::string& input_line)` — парсит строку с числами

Класс Parent

Поля класса

- `int pipe_write_end` — конец канала для записи в дочерний процесс
- `int pipe_read_end` — конец канала для чтения

Основные методы

- `Parent()` — конструктор
- `void CreateChildProcess(const std::string& filename)` — создает дочерний процесс
- `void Input()` — получает пользовательский ввод и записывает в канал
- `void EndChild()` — завершает дочерний процесс
- `~Parent()` — деструктор

Модуль OS

Основные функции

- `int CreateChildProcess(const StartProcess& args)` — создает дочерний процесс
- `bool CreatePipe(PipeHandle& readpipe, PipeHandle& writepipe)` — создает канал
- `int WritePipe(PipeHandle pipe, const void* buf, int count)` — запись в канал
- `int ReadPipe(PipeHandle pipe, void* buf, int count)` — чтение из канала
- `void ClosePipe(PipeHandle pipe)` — закрытие канала
- `void Exit(int code)` — завершение текущего процесса
- `int WaitForChild()` — ожидание завершения дочернего процесса

5 Результаты

Программа получает на вход название файла, создает дочерний процесс, который открывает этот файл. Далее все введенные пользователем строки с числами передаются в дочерний процесс, который выполняет деление первого числа на последующие и записывает результаты в файл.

5.1 Пример работы программы

```
Входные данные: "10 2 5"
Результат в файле:
10 / 2 = 5
```

```
10 / 5 = 2


Входные данные: "100 4 25"
Результат в файле:
100 / 4 = 25
100 / 25 = 4
```

6 Выводы

В ходе выполнения лабораторной работы были приобретены практические навыки в управлении процессами в ОС, обеспечение обмена данных между процессами посредством каналов. Была составлена и отлажена программа на языке C++, осуществляющая работу с процессами и взаимодействие между ними в операционной системе.

В результате работы программа (основной процесс) создает один дочерний процесс. Взаимодействие между процессами осуществляется через каналы (pipes). Обработаны системные ошибки, которые могут возникнуть в результате работы (деление на ноль, ошибки создания каналов и процессов).

Программа демонстрирует принципы межпроцессного взаимодействия и корректного управления ресурсами в многопроцессных приложениях. Была реализована проверка на деление на ноль на стороне дочернего процесса с последующим аварийным завершением обоих процессов в случае ошибки. Обеспечена кроссплатформенность за счет раздельной реализации системных функций для Windows и Linux.

7 Листинги программного кода

7.1 Родительский процесс

```
1  #include "parent.h"
2  #include <iostream>
3  #include <string>
4
5  int main() {
6      std::string filename;
7      std::cout << "Parent:
                                          :" << std::endl;
8      std::getline(std::cin, filename);
9
10     if (filename.empty()) {
11         std::cerr << "Parent:
                  ." << std::endl;
12         return 1;
13     }
14
```

```
    parent::Parent p;
    p.CreateChildProcess(filename);
    p.Input();
    p.EndChild();
    std::cout << "Parent:                                                    ." << std::endl;
    return 0;
}
```

Листинг 1: app/parentmain.cpp

```
#pragma once

#include <iostream>
#include <string>

namespace parent {
    class Parent {
    private:
        int pipe_write_end;
        int pipe_read_end;

    public:
        Parent();
        void CreateChildProcess(const std::string& filename);
        void Input();
        void EndChild();
        ~Parent();
    };
}
```

Листинг 2: include/parent.h

```
#include "parent.h"
#include "os.h"
#include <iostream>
#include <string>
#include <cstdlib>

namespace parent {
    Parent::Parent() : pipe_write_end(-1), pipe_read_end(-1) {}

    void Parent::CreateChildProcess(const std::string& filename) {
        os::PipeHandle read_pipe, write_pipe;

        if (!os::CreatePipe(read_pipe, write_pipe)) {
            std::cerr << "Parent:                                    pipe" << std::
   endl;
            exit(1);
        }

```

```
18          pipe_read_end = read_pipe;
19          pipe_write_end = write_pipe;
20
21          os::StartProcess child_info;
22          child_info.path = "./child";
23          child_info.filename = "child";
24          child_info.stdin_pipe = pipe_read_end;
25
26          if (os::CreateChildProcess(child_info) == -1) {
27              std::cerr << "Parent:
                                      " << std::endl;
28              exit(1);
29          }
30
31          os::ClosePipe(pipe_read_end);
32          pipe_read_end = -1;
33          std::string filename_with_newline = filename + "\n";
34          int written = os::WritePipe(pipe_write_end, filename_with_newline.c_str()
    , filename_with_newline.size());
35          if (written == -1) {
36              std::cerr << "Parent:                                    pipe" << std::endl
    ;
37          }
38      }
39
40      void Parent::Input() {
41          std::cout << "                                                    (
                : '                          1                  2          ...')." <<
    std::endl;
42          std::cout << "                                        'exit'." << std::endl;
43          std::string input;
44
45          while (std::getline(std::cin, input)) {
46              if (input == "exit") {
47                  break;
48              }
49              if (input.empty()) continue;
50
51              input += "\n";
52              int written = os::WritePipe(pipe_write_end, input.c_str(), input.size
    ());
53              if (written == -1) {
54                  std::cerr << "Parent:                                    pipe" << std::
    endl;
55                  break;
56              }
57          }
58      }
59
```

```
60      void Parent::EndChild() {
61          os::ClosePipe(pipe_write_end);
62          pipe_write_end = -1;
63          if (os::WaitForChild() == -1) {
64              std::cerr << "Parent:
                                        " << std::endl;
65          }
66      }
67
68      Parent::~Parent() {
69          if (pipe_write_end != -1) {
70              os::ClosePipe(pipe_write_end);
71          }
72          if (pipe_read_end != -1) {
73              os::ClosePipe(pipe_read_end);
74          }
75      }
76  }
```

Листинг 3: src/parent.cpp

7.2 Дочерний процесс

```
1  #include "child.h"
2  #include <iostream>
3  #include <string>
4
5  int main() {
6      std::string filename;
7      if (!std::getline(std::cin, filename) || filename.empty()) {
8          std::cerr << "Child:              !!!
                                      ." << std::endl;
9          return 1;
10     }
11
12     child::Child child_process(filename);
13     child_process.ProcessDivision();
14     return 0;
15 }
```

Листинг 4: app/childmain.cpp

```
1  #pragma once
2
3  #include <cstdio>
4  #include <iostream>
5  #include <sstream>
6  #include <string>
7  #include <vector>
```

```
8
9  namespace child {
10     class Child {
11     private:
12         std::string filename;
13         FILE* file;
14         std::vector<int> parse_numbers(const std::string& input_line);
15
16     public:
17         explicit Child(const std::string& filename_arg);
18         void ProcessDivision();
19         ~Child();
20     };
21 }
```

Листинг 5: include/child.h

```
1  #include "child.h"
2
3  namespace child {
4      Child::Child(const std::string& filename_arg) : filename(filename_arg), file(
       nullptr) {
5          file = std::fopen(filename.c_str(), "a");
6          if (!file) {
7              std::cerr << "Child:              !
               : " << filename << std::endl;
8          } else {
9              std::fprintf(file, "Child:                .                    : %s\
       n", filename.c_str());
10         }
11     }
12
13     std::vector<int> Child::parse_numbers(const std::string& input_line) {
14         std::vector<int> numbers;
15         std::stringstream ss(input_line);
16         int num;
17         while (ss >> num) {
18             numbers.push_back(num);
19         }
20         return numbers;
21     }
22
23     void Child::ProcessDivision() {
24         if (!file) {
25             return;
26         }
27
28         std::string line;
29         while (std::getline(std::cin, line)) {
```

```
30              if (line.empty()) continue;
31
32              std::vector<int> numbers = parse_numbers(line);
33              if (numbers.size() < 2) {
34                  std::fprintf(file, "             :                                  2
                 ,                  %zu\n", numbers.size());
35                  continue;
36              }
37
38              int dividend = numbers[0];
39              std::fprintf(file, "           %d     : ", dividend);
40              for (size_t i = 1; i < numbers.size(); i++) {
41                  std::fprintf(file, "%d", numbers[i]);
42                  if (i < numbers.size() - 1) std::fprintf(file, ", ");
43              }
44              std::fprintf(file, "\n");
45
46              for (size_t i = 1; i < numbers.size(); i++) {
47                  if (numbers[i] == 0) {
48                      std::fprintf(file, "                                        :
                                  ! %d / %d\n", dividend, numbers[i]);
49                      std::fprintf(file, "Child:
        -                                .\n");
50                      std::cerr << "
             .                                    ." << std::endl;
51                      std::fclose(file);
52                      exit(1);
53                  }
54                  int result = dividend / numbers[i];
55                  std::fprintf(file, "%d / %d = %d\n", dividend, numbers[i], result
    );
56              }
57              std::fprintf(file, "---\n");
58              std::fflush(file);
59          }
60      }
61
62      Child::~Child() {
63          if (file) {
64              std::fprintf(file, "Child:                                  .\n");
65              std::fclose(file);
66          }
67      }
68  }
```

Листинг 6: src/child.cpp

7.3 Системные функции

```h
#pragma once

#include <string>
#include <cstdint>

namespace os {
    using PipeHandle = intptr_t;

    struct StartProcess {
        std::string path;
        std::string filename;
        PipeHandle stdin_pipe = -1;
    };

    int CreateChildProcess(const StartProcess& args);
    bool CreatePipe(PipeHandle& readpipe, PipeHandle& writepipe);
    int WritePipe(PipeHandle pipe, const void* buf, int count);
    int ReadPipe(PipeHandle pipe, void* buf, int count);
    void ClosePipe(PipeHandle pipe);
    void Exit(int code);
    int WaitForChild();
}
```

Листинг 7: include/os.h

```cpp
#include "os.h"
#include <windows.h>
#include <string>

namespace os {

    int CreateChildProcess(const StartProcess& args) {
        STARTUPINFOA si{};
        PROCESS_INFORMATION pi{};
        si.cb = sizeof(si);
        si.dwFlags = STARTF_USESTDHANDLES;

        if (args.stdin_pipe != -1) {
            si.hStdInput = reinterpret_cast<HANDLE>(args.stdin_pipe);
        } else {
            si.hStdInput = GetStdHandle(STD_INPUT_HANDLE);
        }

        si.hStdOutput = GetStdHandle(STD_OUTPUT_HANDLE);
        si.hStdError = GetStdHandle(STD_ERROR_HANDLE);

        std::string command_line = args.path;

        BOOL success = CreateProcessA(
```

```
            nullptr,
            const_cast<char*>(command_line.c_str()),
            nullptr,
            nullptr,
            TRUE,
            0,
            nullptr,
            nullptr,
            &si,
            &pi
        );

        if (!success) {
            return -1;
        }

        CloseHandle(pi.hThread);
        CloseHandle(pi.hProcess);

        return static_cast<int>(pi.dwProcessId);
    }

    bool CreatePipe(PipeHandle& readpipe, PipeHandle& writepipe) {
        SECURITY_ATTRIBUTES sa{};
        sa.nLength = sizeof(sa);
        sa.bInheritHandle = TRUE;
        HANDLE hRead, hWrite;

        if (!CreatePipe(&hRead, &hWrite, &sa, 0)) {
            return false;
        }

        readpipe = reinterpret_cast<PipeHandle>(hRead);
        writepipe = reinterpret_cast<PipeHandle>(hWrite);
        return true;
    }

    int WritePipe(PipeHandle pipe, const void* buf, int count) {
        if (pipe == -1) {
            return -1;
        }
        HANDLE h = reinterpret_cast<HANDLE>(pipe);
        DWORD bytesWritten = 0;

        if (!WriteFile(h, buf, static_cast<DWORD>(count), &bytesWritten, nullptr)
    ) {
            return -1;
        }

```

```
73          return static_cast<int>(bytesWritten);
74      }
75
76      int ReadPipe(PipeHandle pipe, void* buf, int count) {
77          if (pipe == -1) {
78              return -1;
79          }
80          HANDLE h = reinterpret_cast<HANDLE>(pipe);
81          DWORD bytesRead = 0;
82
83          if (!ReadFile(h, buf, static_cast<DWORD>(count), &bytesRead, nullptr)) {
84              return -1;
85          }
86
87          return static_cast<int>(bytesRead);
88      }
89
90      void ClosePipe(PipeHandle pipe) {
91          if (pipe != -1) {
92              HANDLE h = reinterpret_cast<HANDLE>(pipe);
93              CloseHandle(h);
94          }
95      }
96
97      void Exit(int code) {
98          ExitProcess(static_cast<UINT>(code));
99      }
100
101     int WaitForChild() {
102         Sleep(1000);
103         return 0;
104     }
105
106 }
```

Листинг 8: src/oswin.cpp

```
1  #include "os.h"
2  #include <unistd.h>
3  #include <sys/wait.h>
4  #include <cstdlib>
5  #include <cstdio>
6
7  namespace os {
8
9      int CreateChildProcess(const StartProcess& args) {
10         pid_t pid = fork();
11         if (pid == -1) {
12             perror("fork failed");
```

```
            return -1;
        }

        if (pid == 0) {
            if (args.stdin_pipe != -1) {
                if (dup2(args.stdin_pipe, STDIN_FILENO) == -1) {
                    perror("dup2 failed");
                    Exit(1);
                }
                close(args.stdin_pipe);
            }

            execl(args.path.c_str(), args.filename.c_str(), nullptr);
            perror("execl failed");
            Exit(1);
        }

        return pid;
    }

    bool CreatePipe(PipeHandle& readpipe, PipeHandle& writepipe) {
        int pipefd[2];
        if (pipe(pipefd) != 0) {
            perror("pipe creation failed");
            return false;
        }
        readpipe = pipefd[0];
        writepipe = pipefd[1];
        return true;
    }

    int WritePipe(PipeHandle pipe, const void* buf, int count) {
        int result = write(pipe, buf, count);
        if (result == -1) {
            perror("write pipe failed");
        }
        return result;
    }

    int ReadPipe(PipeHandle pipe, void* buf, int count) {
        int result = read(pipe, buf, count);
        if (result == -1) {
            perror("read pipe failed");
        }
        return result;
    }

    void ClosePipe(PipeHandle pipe) {
        if (pipe != -1) {
```

```
62              close(pipe);
63          }
64      }
65
66      void Exit(int code) {
67          _exit(code);
68      }
69
70      int WaitForChild() {
71          int status = wait(nullptr);
72          return status;
73      }
74
75  }
```

Листинг 9: src/os.cpp

8 strace

```
1  execve("./parent", ["./parent"], 0x7ffe93a9f208 /* 37 vars */) = 0
2  brk(NULL)                                = 0x55f0fd08a000
3  mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
       x7f8093061000
4  access("/etc/ld.so.preload", R_OK)       = -1 ENOENT (No such file or directory)
5  openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
6  fstat(3, {st_mode=S_IFREG|0644, st_size=21863, ...}) = 0
7  mmap(NULL, 21863, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f809305b000
8  close(3)                                 = 0
9  openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = 3
10 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\0"..., 832)
       = 832
11 fstat(3, {st_mode=S_IFREG|0644, st_size=2592224, ...}) = 0
12 mmap(NULL, 2609472, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f8092ddd000
13 mmap(0x7f8092e7a000, 1343488, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
       MAP_DENYWRITE, 3, 0x9d000) = 0x7f8092e7a000
14 mmap(0x7f8092fc2000, 552960, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0
       x1e5000) = 0x7f8092fc2000
15 mmap(0x7f8093049000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
       MAP_DENYWRITE, 3, 0x26b000) = 0x7f8093049000
16 mmap(0x7f8093057000, 12608, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
       MAP_ANONYMOUS, -1, 0) = 0x7f8093057000
17 close(3)                                 = 0
18 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
19 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
20 fstat(3, {st_mode=S_IFREG|0644, st_size=183024, ...}) = 0
21 mmap(NULL, 185256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f8092daf000
```

```
22 mmap(0x7f8092db3000, 147456, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
       MAP_DENYWRITE, 3, 0x4000) = 0x7f8092db3000
23 mmap(0x7f8092dd7000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0
       x28000) = 0x7f8092dd7000
24 mmap(0x7f8092ddb000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
       MAP_DENYWRITE, 3, 0x2b000) = 0x7f8092ddb000
25 close(3)                                = 0
26 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
27 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) = 832
28 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
29 fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
30 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
31 mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f8092b9d000
32 mmap(0x7f8092bc5000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
       MAP_DENYWRITE, 3, 0x28000) = 0x7f8092bc5000
33 mmap(0x7f8092d4d000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0
       x1b0000) = 0x7f8092d4d000
34 mmap(0x7f8092d9c000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
       MAP_DENYWRITE, 3, 0x1fe000) = 0x7f8092d9c000
35 mmap(0x7f8092da2000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
       MAP_ANONYMOUS, -1, 0) = 0x7f8092da2000
36 close(3)                                = 0
37 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3
38 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\0"..., 832)
       = 832
39 fstat(3, {st_mode=S_IFREG|0644, st_size=952616, ...}) = 0
40 mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f8092ab4000
41 mmap(0x7f8092ac4000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
       MAP_DENYWRITE, 3, 0x10000) = 0x7f8092ac4000
42 mmap(0x7f8092b43000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0
       x8f000) = 0x7f8092b43000
43 mmap(0x7f8092b9b000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
       MAP_DENYWRITE, 3, 0xe7000) = 0x7f8092b9b000
44 close(3)                                = 0
45 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
       x7f8092ab2000
46 mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
       x7f8092aaf000
47 arch_prctl(ARCH_SET_FS, 0x7f8092aaf740) = 0
48 set_tid_address(0x7f8092aafa10)         = 208412
49 set_robust_list(0x7f8092aafa20, 24)     = 0
50 rseq(0x7f8092ab0060, 0x20, 0, 0x53053053) = 0
51 mprotect(0x7f8092d9c000, 16384, PROT_READ) = 0
52 mprotect(0x7f8092b9b000, 4096, PROT_READ) = 0
53 mprotect(0x7f8092ddb000, 4096, PROT_READ) = 0
54 mprotect(0x7f8093049000, 45056, PROT_READ) = 0
```

```
mprotect(0x55f0eef2a000, 4096, PROT_READ) = 0
mprotect(0x7f8093099000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
   = 0
munmap(0x7f809305b000, 21863)           = 0
futex(0x7f80930577bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
getrandom("\x7f\x05\x80\x1b\xb5\x8c\x3b\x66", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x55f0fd08a000
brk(0x55f0fd0ab000)                     = 0x55f0fd0ab000
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0xa), ...}) = 0
write(1, "Parent: \320\222\320\262\320\265\320\264\320\270\321\202\320\265
   \320\270\320\274\321\217 \321\204"..., 85Parent:
                                                          :
) = 85
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0xa), ...}) = 0
read(0, t.txt
"t.txt\n", 1024)                = 6
pipe2([3, 4], 0)                        = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
   SIGCHLDstrace: Process 208473 attached
, child_tidptr=0x7f8092aafa10) = 208473
[pid 208473] set_robust_list(0x7f8092aafa20, 24 <unfinished ...>
[pid 208412] close(3 <unfinished ...>
[pid 208473] <... set_robust_list resumed>) = 0
[pid 208412] <... close resumed>)       = 0
[pid 208412] write(4, "t.txt\n", 6 <unfinished ...>
[pid 208473] close(4 <unfinished ...>
[pid 208412] <... write resumed>)       = 6
[pid 208473] <... close resumed>)       = 0
[pid 208412] write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
   \321\207\320\270\321\201\320\273\320\260 \320\264\320\273\321\217"..., 122 <
   unfinished ...>
                                                 (            : ’
          1                 2         ...’).
[pid 208473] dup2(3, 0 <unfinished ...>
[pid 208412] <... write resumed>)       = 122
[pid 208473] <... dup2 resumed>)        = 0
[pid 208412] write(1, "\320\224\320\273\321\217
   \320\262\321\213\321\205\320\276\320\264\320\260
   \320\262\320\262\320\265\320\264\320\270\321\202"..., 43 <unfinished ...>
                                  ’exit’.
[pid 208473] close(3 <unfinished ...>
[pid 208412] <... write resumed>)       = 43
[pid 208473] <... close resumed>)       = 0
[pid 208412] read(0,  <unfinished ...>
[pid 208473] execve("./child", ["child"], 0x7ffc027962a8 /* 37 vars */) = 0
[pid 208473] brk(NULL)                  = 0x55e2bd7bd000
[pid 208473] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
   -1, 0) = 0x7f1ceb678000
```

```
94 [pid 208473] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or
   directory)
95 [pid 208473] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
96 [pid 208473] fstat(3, {st_mode=S_IFREG|0644, st_size=21863, ...}) = 0
97 [pid 208473] mmap(NULL, 21863, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f1ceb672000
98 [pid 208473] close(3)                  = 0
99 [pid 208473] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6", O_RDONLY|
   O_CLOEXEC) = 3
100 [pid 208473] 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
101 [pid 208473] fstat(3, {st_mode=S_IFREG|0644, st_size=2592224, ...}) = 0
102 [pid 208473] mmap(NULL, 2609472, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0
   x7f1ceb3f4000
103 [pid 208473] mmap(0x7f1ceb491000, 1343488, PROT_READ|PROT_EXEC, MAP_PRIVATE|
   MAP_FIXED|MAP_DENYWRITE, 3, 0x9d000) = 0x7f1ceb491000
104 [pid 208473] mmap(0x7f1ceb5d9000, 552960, PROT_READ, MAP_PRIVATE|MAP_FIXED|
   MAP_DENYWRITE, 3, 0x1e5000) = 0x7f1ceb5d9000
105 [pid 208473] mmap(0x7f1ceb660000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|
   MAP_FIXED|MAP_DENYWRITE, 3, 0x26b000) = 0x7f1ceb660000
106 [pid 208473] mmap(0x7f1ceb66e000, 12608, PROT_READ|PROT_WRITE, MAP_PRIVATE|
   MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f1ceb66e000
107 [pid 208473] close(3)                  = 0
108 [pid 208473] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|
   O_CLOEXEC) = 3
109 [pid 208473] 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
110 [pid 208473] fstat(3, {st_mode=S_IFREG|0644, st_size=183024, ...}) = 0
111 [pid 208473] mmap(NULL, 185256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0
   x7f1ceb3c6000
112 [pid 208473] mmap(0x7f1ceb3ca000, 147456, PROT_READ|PROT_EXEC, MAP_PRIVATE|
   MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x7f1ceb3ca000
113 [pid 208473] mmap(0x7f1ceb3ee000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|
   MAP_DENYWRITE, 3, 0x28000) = 0x7f1ceb3ee000
114 [pid 208473] mmap(0x7f1ceb3f2000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
   MAP_FIXED|MAP_DENYWRITE, 3, 0x2b000) = 0x7f1ceb3f2000
115 [pid 208473] close(3)                  = 0
116 [pid 208473] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|
   O_CLOEXEC) = 3
117 [pid 208473] 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) = 832
118 [pid 208473] 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
119 [pid 208473] fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
120 [pid 208473] 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
121 [pid 208473] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0
   x7f1ceb1b4000
122 [pid 208473] mmap(0x7f1ceb1dc000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|
   MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f1ceb1dc000
```

```
[pid 208473] mmap(0x7f1ceb364000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|
   MAP_DENYWRITE, 3, 0x1b0000) = 0x7f1ceb364000
[pid 208473] mmap(0x7f1ceb3b3000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|
   MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7f1ceb3b3000
[pid 208473] mmap(0x7f1ceb3b9000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|
   MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f1ceb3b9000
[pid 208473] close(3)                   = 0
[pid 208473] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|
   O_CLOEXEC) = 3
[pid 208473] 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\0"..., 832) = 832
[pid 208473] fstat(3, {st_mode=S_IFREG|0644, st_size=952616, ...}) = 0
[pid 208473] mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0
   x7f1ceb0cb000
[pid 208473] mmap(0x7f1ceb0db000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|
   MAP_FIXED|MAP_DENYWRITE, 3, 0x10000) = 0x7f1ceb0db000
[pid 208473] mmap(0x7f1ceb15a000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|
   MAP_DENYWRITE, 3, 0x8f000) = 0x7f1ceb15a000
[pid 208473] mmap(0x7f1ceb1b2000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
   MAP_FIXED|MAP_DENYWRITE, 3, 0xe7000) = 0x7f1ceb1b2000
[pid 208473] close(3)                   = 0
[pid 208473] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
   -1, 0) = 0x7f1ceb0c9000
[pid 208473] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
   -1, 0) = 0x7f1ceb0c6000
[pid 208473] arch_prctl(ARCH_SET_FS, 0x7f1ceb0c6740) = 0
[pid 208473] set_tid_address(0x7f1ceb0c6a10) = 208473
[pid 208473] set_robust_list(0x7f1ceb0c6a20, 24) = 0
[pid 208473] rseq(0x7f1ceb0c7060, 0x20, 0, 0x53053053) = 0
[pid 208473] mprotect(0x7f1ceb3b3000, 16384, PROT_READ) = 0
[pid 208473] mprotect(0x7f1ceb1b2000, 4096, PROT_READ) = 0
[pid 208473] mprotect(0x7f1ceb3f2000, 4096, PROT_READ) = 0
[pid 208473] mprotect(0x7f1ceb660000, 45056, PROT_READ) = 0
[pid 208473] mprotect(0x55e2a91c6000, 4096, PROT_READ) = 0
[pid 208473] mprotect(0x7f1ceb6b0000, 8192, PROT_READ) = 0
[pid 208473] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=
   RLIM64_INFINITY}) = 0
[pid 208473] munmap(0x7f1ceb672000, 21863) = 0
[pid 208473] futex(0x7f1ceb66e7bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
[pid 208473] getrandom("\x41\x84\xd1\xb4\x79\x26\x1a\x5d", 8, GRND_NONBLOCK) = 8
[pid 208473] brk(NULL)                  = 0x55e2bd7bd000
[pid 208473] brk(0x55e2bd7de000)        = 0x55e2bd7de000
[pid 208473] fstat(0, {st_mode=S_IFIFO|0600, st_size=0, ...}) = 0
[pid 208473] read(0, "t.txt\n", 4096)   = 6
[pid 208473] openat(AT_FDCWD, "t.txt", O_WRONLY|O_CREAT|O_APPEND, 0666) = 3
[pid 208473] lseek(3, 0, SEEK_END)      = 375
[pid 208473] fstat(3, {st_mode=S_IFREG|0644, st_size=375, ...}) = 0
[pid 208473] read(0, 10 2 5
 <unfinished ...>
```

```
[pid 208412] <... read resumed>"10 2 5 \n", 1024) = 8
[pid 208412] write(4, "10 2 5 \n", 8)   = 8
[pid 208473] <... read resumed>"10 2 5 \n", 4096) = 8
[pid 208412] read(0,  <unfinished ...>
[pid 208473] write(3, "Child:
   \320\227\320\260\320\277\321\203\321\211\320\265\320\275.
   \320\244\320\260\320\271\320\273 "..., 103) = 103
[pid 208473] read(0, exit
 <unfinished ...>
[pid 208412] <... read resumed>"exit\n", 1024) = 5
[pid 208412] write(1, "Parent:
   \320\227\320\260\320\272\321\200\321\213\320\262\320\260\321\216
   \320\277\320\260\320\271\320"..., 85Parent:
                                                ...
) = 85
[pid 208412] close(4)                   = 0
[pid 208473] <... read resumed>"", 4096) = 0
[pid 208412] wait4(208473,  <unfinished ...>
[pid 208473] write(2, "Child:
   \320\237\320\276\320\273\321\203\321\207\320\265\320\275 EOF,
   \320\267\320\260\320"..., 64Child:               EOF,
   ProcessDivision.) = 64
[pid 208473] write(2, "\n", 1
)          = 1
[pid 208473] write(3, "Child: \320\240\320\260\320\261\320\276\321\202\320\260
   \320\267\320\260\320\262\320\265\321\200\321\210"..., 40) = 40
[pid 208473] close(3)                   = 0
[pid 208473] exit_group(0)              = ?
[pid 208473] +++ exited with 0 +++
<... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 208473
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=208473, si_uid=1000,
   si_status=0, si_utime=0, si_stime=1 /* 0.01 s */} ---
write(1, "Parent:
   \320\224\320\276\321\207\320\265\321\200\320\275\320\270\320\271
   \320\277\321\200\320\276\321"..., 58Parent:
                    .
) = 58
write(1, "Parent:
   \320\237\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260
   \320\267\320\260\320"..., 47Parent:                                    .
) = 47
exit_group(0)                           = ?
+++ exited with 0 +++
```