

**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: Д. А. Алгиничев
Группа: М8О-208БВ-24
Преподаватель: Е. С. Миронов

Москва, 2025

Условие

Составить и отладить программу, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программы (основной процесс) должен создать для решения задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы (memory-mapped files). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Цель работы

Изучение механизмов межпроцессного взаимодействия через разделяемую память (shared memory) и синхронизации процессов с использованием семафоров.

Задание

Реализовать программу, состоящую из родительского и двух дочерних процессов, взаимодействующих через разделяемую память. Родительский процесс распределяет вводимые строки между дочерними процессами по принципу четности: нечетные строки передаются первому процессу, четные - второму. Каждый дочерний процесс инвертирует полученные строки и записывает результат в указанный файл..

Вариант

21

Архитектура программы

Общая структура

Программа реализует многопроцессную архитектуру с использованием разделяемой памяти для межпроцессного взаимодействия. Основные компоненты системы:

- **ParentProcess** - управляет созданием дочерних процессов и распределением данных
- **SharedMemory** - инкапсулирует работу с разделяемой памятью и семафорами
- **ChildProcess** - отвечает за запуск дочерних процессов
- **ChildProcessor** - обрабатывает данные в дочерних процессах

Ключевые алгоритмы

Синхронизация с помощью семафоров

Для координации доступа к разделяемой памяти используются два семафора:

- **sem_producer** - контролирует запись данных
- **sem_consumer** - контролирует чтение данных

Результаты тестирования

Пример работы программы

```
Введите имя файла для child1: file1
Введите имя файла для child2: file2
child1: запущен с shared memory /child1_shm и выходным файлом file1
child2: запущен с shared memory /child2_shm и выходным файлом file2
hello
big
create
table
Данные отправлены в дочерние процессы
child1: получено 'hello\ncreate\n'
child2: получено 'big\ntable\n'
child1: обработал и записал данные в file1
child2: обработал и записал данные в file2
Данные обработаны дочерними процессами
Дочерние процессы завершили работу
```

Содержимое выходных файлов

file1:

```
olleh
etaerc
```

file2:

```
gib
elbat
```

Анализ системных вызовов

В ходе работы программы были использованы следующие ключевые системные вызовы:

- `shm_open` - создание/открытие разделяемой памяти
- `mmap` - отображение памяти в адресное пространство процесса
- `sem_init` - инициализация семафоров
- `sem_wait/sem_post` - операции синхронизации
- `fork` - создание дочерних процессов
- `execve` - запуск исполняемых файлов
- `waitpid` - ожидание завершения дочерних процессов

Выводы

В процессе выполнения этой лабораторной работы я получил ценный практический опыт в нескольких ключевых областях:

Межпроцессное взаимодействие - я на практике освоил работу с разделяемой памятью (shared memory), что оказалось гораздо интереснее теоретического изучения. Особенno впечатлила скорость обмена данными между процессами по сравнению с другими механизмами IPC.

Синхронизация процессов - работа с семафорами показала мне, насколько важна правильная синхронизация в многопроцессных системах. Я столкнулся с проблемой взаимной блокировки (deadlock) и научился ее диагностировать и исправлять, что было ценным опытом отладки.

Приобретенные навыки

- **Практическое применение IPC** - теперь я уверенно могу использовать разделяемую память в своих проектах
- **Отладка многопроцессных приложений** - научился использовать strace, добавлять диагностику и анализировать сложные сценарии взаимодействия процессов
- **Проектирование архитектуры** - понял, как правильно разделять ответственность между процессами и организовывать их взаимодействие

Заключение

Эта лабораторная работа стала для меня настоящим прорывом в понимании многозадачности и межпроцессного взаимодействия. Из абстрактных концепций эти темы превратились в конкретные инструменты, которые я теперь могу применять в реальных проектах. Особую ценность имел процесс преодоления трудностей - каждая исправленная ошибка давала новое понимание работы операционной системы. Я не просто написал программу, а глубоко разобрался в том, как она взаимодействует с ядром ОС.

Полученные знания и навыки обязательно пригодятся мне в будущем, особенно при работе с высоконагруженными приложениями и распределенными системами. Эта работа показала мне, что системное программирование - это не только сложно, но и невероятно интересно!

Исходная программа

```
1 #include "process.hpp"
2
3 int main() {
4     ParentProcess parent;
5     parent.start();
6     return 0;
7 }
```

Листинг 1: main.cpp - точка входа, создает ParentProcess

```
1 #include "process.hpp"
2
3 #include <iostream>
4
5 #include <string>
6
7 void ParentProcess::start()
8 {
9     child1->execute();
10    child2->execute();
11
12    sleep(10);
13
14    std::string line;
15    int line_number = 0;
16    std::string data1, data2;
17
18    while (std::getline(std::cin, line))
19    {
20        ++line_number;
21        if (line_number % 2 == 1)
22        {
23            if (!data1.empty()) data1 += "\n";
24            data1 += line;
25        }
26        else
27        {
28            if (!data2.empty()) data2 += "\n";
29            data2 += line;
30        }
31    }
32
33    shm1->writeData(data1);
34    shm2->writeData(data2);
35
36    std::cout << "Данные отправлены в дочерние процессы" << std::endl;
37
38    shm1->waitConsumer();
39    shm2->waitConsumer();
40
41    std::cout << "Данные обработаны дочерними процессами" << std::endl;
42
43    int status1, status2;
44
45    waitpid(child1->getPid(), &status1, 0);
46    waitpid(child2->getPid(), &status2, 0);
47
48    std::cout << "Дочерние процессы завершили работу" << std::endl;
49 }
50 }
```

```

51 ParentProcess::ParentProcess()
52 {
53     std::cout << "Введите имя файла для child1: ";
54     std::getline(std::cin, file1);
55     std::cout << "Введите имя файла для child2: ";
56     std::getline(std::cin, file2);
57
58     shm1 = new SharedMemory("/child1_shm", 4096, true);
59     shm2 = new SharedMemory("/child2_shm", 4096, true);
60
61     child1 = new ChildProcess("/child1_shm", file1, true);
62     child2 = new ChildProcess("/child2_shm", file2, false);
63 }
64
65 ParentProcess::~ParentProcess()
66 {
67     delete child1;
68     delete child2;
69     delete shm1;
70     delete shm2;
71 }
```

Листинг 2: process.cpp - класс ParentProcess, управляет дочерними процессами

```

1 #include "childProcess.hpp"
2
3 ChildProcess::ChildProcess(const std::string& shm_n, const std::string& f, bool
4     is_c1)
5     : shm_name(shm_n), output_file_name(f), is_child1(is_c1), pid(-1)
6 {
7     prefix = is_child1 ? "child1" : "child2";
8 }
9
10 void ChildProcess::execute()
11 {
12     pid = fork();
13     if (pid == -1)
14     {
15         perror("Не удалось создать процесс");
16         exit(1);
17     }
18
19     if (pid == 0)
20     {
21         execl("./child", "./child", prefix.c_str(), shm_name.c_str(),
22             output_file_name.c_str(), nullptr);
23
24         perror("Не удалось запустить дочерний процесс");
25         exit(1);
26     }
27 }
```

```

25 }
26
27 pid_t ChildProcess::getPid() const
28 {
29     return pid;
30 }
```

Листинг 3: childProcess.cpp - класс ChildProcess, запуск дочерних процессов

```

1 #include <iostream>
2 #include <cstdlib>
3 #include <string>
4 #include "childProcessor.hpp"
5
6 int main(int argc, char* argv[]) {
7     if (argc != 4) {
8         std::cerr << "Использование: " << argv[0] << " <prefix> <shm_name> <
9             output_file>" << std::endl;
10    return 1;
11 }
12
13 std::string prefix = argv[1];
14 std::string shm_name = argv[2];
15 std::string output_file = argv[3];
16
17 std::cout << prefix << ": запущен с shared memory " << shm_name
18     << " и выходным файлом" << output_file << std::endl;
19
20 ChildProcessor processor(prefix);
21     return processor.process(shm_name, output_file);
22 }
```

Листинг 4: child main.cpp - точка входа дочернего процесса

```

1 #include "childProcessor.hpp"
2
3 #include <iostream>
4 #include <fstream>
5 #include <sstream>
6
7 #include <string>
8
9 #include "sharedMemory.hpp"
10
11 ChildProcessor::ChildProcessor(const std::string& pref) : prefix(pref) {}
12
13 std::string ChildProcessor::reverseString(const std::string& s) {
14     size_t len = s.length();
15     std::string result;
```

```

16     for (size_t i = len; i > 0; --i) {
17         result += s[i - 1];
18     }
19     return result;
20 }
21
22 int ChildProcessor::process(const std::string& shm_name, const std::string&
23                             output_file)
24 {
25     SharedMemory shm(shm_name, 4096, false);
26     std::string data = shm.readData();
27
28     if (data.empty()) {
29         std::cerr << prefix << ": нет данных" << std::endl;
30         return 1;
31     }
32
33     std::cout << prefix << ": получено'" << data << "'" << std::endl;
34
35     std::ofstream outfile(output_file);
36     if (!outfile) {
37         std::cerr << prefix << ": не удалось создать файл" << std::endl;
38         return 1;
39     }
40
41     std::istringstream stream(data);
42     std::string line;
43     int line_count = 0;
44
45     while (std::getline(stream, line)) {
46         if (!line.empty()) {
47             std::string reversed = reverseString(line);
48             outfile << reversed << std::endl;
49             line_count++;
50         }
51     }
52
53     std::cout << prefix << ": записано" << line_count << " строк" <<
54     output_file << std::endl;
55     return 0;
56 }
```

Листинг 5: childProcessor.cpp - класс ChildProcessor, обработка строк

sharedMemory.hpp - заголовочный файл разделяемой памяти

```

1 #pragma once
2
3 #include <string>
4
```

```

5  class SharedMemory
6  {
7      private:
8          std::string name;
9          int fd;
10         void * data;
11         size_t size;
12         bool is_owner;
13
14         struct Semaphore
15         {
16             void * sem_data;
17         };
18
19         Semaphore sem_prod;
20         Semaphore sem_cons;
21     public:
22         SharedMemory(const std::string& name, size_t size, bool shm_create =
23                         false);
24         ~SharedMemory() noexcept;
25
26         void * getData() const;
27         size_t getSize() const;
28         void writeData(const std::string& datas);
29         std::string readData();
30
31         void waitProducer();
32         void waitConsumer();
33         void postProducer();
34         void postConsumer();
35     };

```

Листинг 6: sharedMemory.hpp - заголовочный файл разделяемой памяти

sharedMemory.cpp - реализация разделяемой памяти

```

1 #include "sharedMemory.hpp"
2
3 #include <iostream>
4 #include <sys/mman.h>
5 #include <fcntl.h>
6 #include <unistd.h>
7 #include <sys/stat.h>
8 #include <cstring>
9 #include <semaphore.h>
10
11 SharedMemory::SharedMemory(const std::string& shm_name, size_t shm_size, bool
12                             shm_create)
13     : name(shm_name), size(shm_size), is_owner(shm_create)
14 {

```

```

14     if (shm_create)
15     {
16         fd = shm_open(name.c_str(), O_CREAT | O_RDWR, 0666);
17         if (fd == -1)
18         {
19             perror("shm create error");
20             exit(1);
21         }
22
23         size_t total_size = size + sizeof(sem_t) * 2;
24         if (ftruncate(fd, total_size) == -1)
25         {
26             perror("Не удалось установить размер");
27             exit(1);
28         }
29     }
30     else
31     {
32         fd = shm_open(name.c_str(), O_RDWR, 0666);
33         if (fd == -1)
34         {
35             perror("shm open error");
36             exit(1);
37         }
38     }
39
40     data = mmap(NULL, size + sizeof(sem_t) * 2, PROT_READ | PROT_WRITE,
41                 MAP_SHARED, fd, 0);
42     if (data == MAP_FAILED)
43     {
44         perror("mmap error");
45         exit(1);
46     }
47
48     sem_prod.sem_data = static_cast<char*>(data) + size;
49     sem_cons.sem_data = static_cast<char*>(data) + size + sizeof(sem_t);
50
51     if (shm_create)
52     {
53         if (sem_init(static_cast<sem_t*>(sem_prod.sem_data), 1, 0) == -1)
54         {
55             perror("sem_init producer error");
56             exit(1);
57         }
58         if (sem_init(static_cast<sem_t*>(sem_cons.sem_data), 1, 0) == -1)
59         {
60             perror("sem_init consumer error");
61             exit(1);
62         }
63     }

```

```
64
65 void SharedMemory::writeData(const std::string& datas)
66 {
67     if (datas.size() >= size)
68     {
69         std::cerr << "Слишком большой объём данных для shared memory" << std::endl;
70     }
71
72     size_t data_size = datas.size();
73     memcpy(data, &data_size, sizeof(size_t));
74     memcpy(static_cast<char*>(data) + sizeof(size_t), datas.c_str(), data_size);
75
76     postProducer();
77 }
78
79 std::string SharedMemory::readData()
80 {
81     waitProducer();
82
83     size_t data_size;
84     memcpy(&data_size, data, sizeof(size_t));
85
86     if (data_size >= size - sizeof(size_t))
87     {
88         return "";
89     }
90
91     std::string result(static_cast<char*>(data) + sizeof(size_t), data_size);
92
93     postConsumer();
94
95     return result;
96 }
97
98 void SharedMemory::waitProducer()
99 {
100     sem_wait(static_cast<sem_t*>(sem_prod.sem_data));
101 }
102
103 void SharedMemory::waitConsumer()
104 {
105     sem_wait(static_cast<sem_t*>(sem_cons.sem_data));
106 }
107
108 void SharedMemory::postProducer()
109 {
110     sem_post(static_cast<sem_t*>(sem_prod.sem_data));
111 }
112
113 void SharedMemory::postConsumer()
114 {
```

```

115     sem_post(static_cast<sem_t*>(sem_cons.sem_data));
116 }
117
118 SharedMemory::~SharedMemory()
119 {
120     if (data != MAP_FAILED)
121     {
122         if (is_owner)
123         {
124             sem_destroy(static_cast<sem_t*>(sem_prod.sem_data));
125             sem_destroy(static_cast<sem_t*>(sem_cons.sem_data));
126         }
127
128         munmap(data, size + sizeof(sem_t) * 2);
129     }
130
131     if (fd != -1)
132     {
133         close(fd);
134     }
135
136     if (is_owner)
137     {
138         shm_unlink(name.c_str());
139     }
140 }
141
142 void * SharedMemory::getData() const
143 {
144     return data;
145 }
146
147 size_t SharedMemory::getSize() const
148 {
149     return size;
150 }

```

Листинг 7: sharedMemory.cpp - реализация разделяемой памяти

Системные вызовы

```

302509 execve("./parent", ["../parent"], 0x7ffc7c2a3138 /* 36 vars */) = 0
302509 brk(NULL)                      = 0x6352a142f000
302509 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0x7ed6942cb000
302509 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
302509 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
302509 fstat(3, {st_mode=S_IFREG|0644, st_size=33863, ...}) = 0
302509 mmap(NULL, 33863, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ed6942c2000
302509 close(3)                        = 0

```



```
= 48
302509 fstat(0,{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x6),...}) = 0
302509 read(0,"file1\n",1024) = 6
302509 write(1,"\\320\\222\\320\\262\\320\\265\\320\\264\\320\\270\\321\\202\\320\\265 \\320\\270\\320\\321\\204\\320\\260\\320\\271\\320\\273\\320\\260 \\320\\264\\320\\273\\321\\217 child2: ",48)
= 48
302509 read(0,"file2\n",1024) = 6
302509 openat(AT_FDCWD,"/dev/shm/child1_shm",O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC,0666
= 3
302509 ftruncate(3,4160) = 0
302509 mmap(NULL,4160,PROT_READ|PROT_WRITE,MAP_SHARED,3,0) = 0x7ed6942c9000
302509 openat(AT_FDCWD,"/dev/shm/child2_shm",O_RDWR|O_CREAT|O_NOFOLLOW|O_CLOEXEC,0666
= 4
302509 ftruncate(4,4160) = 0
302509 mmap(NULL,4160,PROT_READ|PROT_WRITE,MAP_SHARED,4,0) = 0x7ed6942c7000
302509 clone(child_stack=NULL,flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,c
= 302535
302535 set_robust_list(0x7ed69428fa20,24 <unfinished ...>
302509 clone(child_stack=NULL,flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD
<unfinished ...>
302535 <... set_robust_list resumed>) = 0
302535 execve("./child",["./child","child1","/child1_shm","file1"],0x7ffc7a5af818
/* 36 vars */ <unfinished ...>
302509 <... clone resumed>,child_tidptr=0x7ed69428fa10) = 302536
302536 set_robust_list(0x7ed69428fa20,24 <unfinished ...>
302509 clock_nanosleep(CLOCK_REALTIME,0,{tv_sec=10,tv_nsec=0}, <unfinished
...>
302536 <... set_robust_list resumed>) = 0
302536 execve("./child",["./child","child2","/child2_shm","file2"],0x7ffc7a5af818
/* 36 vars */) = 0
302535 <... execve resumed>) = 0
302536 brk(NULL <unfinished ...>
302535 brk(NULL <unfinished ...>
302536 <... brk resumed>) = 0x5f2d70ae1000
302535 <... brk resumed>) = 0x631689272000
302536 mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x767eee48d000
302535 mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0 <unfinished
...>
302536 access("/etc/ld.so.preload",R_OK <unfinished ...>
302535 <... mmap resumed>) = 0x7381a264b000
302536 <... access resumed>) = -1 ENOENT (No such file or directory)
302535 access("/etc/ld.so.preload",R_OK <unfinished ...>
302536 openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC <unfinished ...>
302535 <... access resumed>) = -1 ENOENT (No such file or directory)
302536 <... openat resumed>) = 3
302535 openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC <unfinished ...>
302536 fstat(3, <unfinished ...>
```



```
302536 <... mmap resumed>) = 0x767eee47f000
302535 <... close resumed>) = 0
302536 arch_prctl(ARCH_SET_FS,0x767eee47f740 <unfinished ...>
302535 mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0 <unfinished ...
...>
302536 <... arch_prctl resumed>) = 0
302535 <... mmap resumed>) = 0x7381a2529000
302536 set_tid_address(0x767eee47fa10 <unfinished ...>
302535 mmap(NULL,12288,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0
<unfinished ...>
302536 <... set_tid_address resumed>) = 302536
302535 <... mmap resumed>) = 0x7381a2526000
302536 set_robust_list(0x767eee47fa20,24 <unfinished ...>
302535 arch_prctl(ARCH_SET_FS,0x7381a2526740 <unfinished ...>
302536 <... set_robust_list resumed>) = 0
302535 <... arch_prctl resumed>) = 0
302536 rseq(0x767eee480060,0x20,0,0x53053053 <unfinished ...>
302535 set_tid_address(0x7381a2526a10 <unfinished ...>
302536 <... rseq resumed>) = 0
302535 <... set_tid_address resumed>) = 302535
302535 set_robust_list(0x7381a2526a20,24 <unfinished ...>
302536 mprotect(0x767eedfff000,16384,PROT_READ <unfinished ...>
302535 <... set_robust_list resumed>) = 0
302536 <... mprotect resumed>) = 0
302535 rseq(0x7381a2527060,0x20,0,0x53053053 <unfinished ...>
302536 mprotect(0x767eee1d0000,4096,PROT_READ) = 0
302535 <... rseq resumed>) = 0
302536 mprotect(0x767eee1fe000,4096,PROT_READ) = 0
302535 mprotect(0x7381a1fff000,16384,PROT_READ) = 0
302535 mprotect(0x7381a2612000,4096,PROT_READ) = 0
302536 mprotect(0x767eee46c000,45056,PROT_READ <unfinished ...>
302535 mprotect(0x7381a2640000,4096,PROT_READ <unfinished ...>
302536 <... mprotect resumed>) = 0
302535 <... mprotect resumed>) = 0
302536 mprotect(0x5f2d62e0e000,4096,PROT_READ) = 0
302536 mprotect(0x767eee4c5000,8192,PROT_READ) = 0
302535 mprotect(0x7381a246c000,45056,PROT_READ <unfinished ...>
302536 prlimit64(0,RLIMIT_STACK,NULL, <unfinished ...>
302535 <... mprotect resumed>) = 0
302536 <... prlimit64 resumed>{rlim_cur=8192*1024,rlim_max=RLIM64_INFINITY})
= 0
302535 mprotect(0x631649d6e000,4096,PROT_READ <unfinished ...>
302536 munmap(0x767eee484000,33863 <unfinished ...>
302535 <... mprotect resumed>) = 0
302536 <... munmap resumed>) = 0
302535 mprotect(0x7381a2683000,8192,PROT_READ) = 0
302536 futex(0x767eee47a7bc,FUTEX_WAKE_PRIVATE,2147483647 <unfinished ...>
302535 prlimit64(0,RLIMIT_STACK,NULL, <unfinished ...>
```

```
302536 <... futex resumed> = 0
302535 <... prlimit64 resumed>{rlim_cur=8192*1024,rlim_max=RLIM64_INFINITY})
= 0
302536 getrandom( <unfinished ...>
302535 munmap(0x7381a2642000,33863 <unfinished ...>
302536 <... getrandom resumed>"\xab\xb0\x44\x84\x13\x a0\xcd\x3a",8,GRND_NONBLOCK)
= 8
302536 brk(NULL <unfinished ...>
302535 <... munmap resumed> = 0
302536 <... brk resumed> = 0x5f2d70ae1000
302535 futex(0x7381a247a7bc,FUTEX_WAKE_PRIVATE,2147483647 <unfinished ...>
302536 brk(0x5f2d70b02000 <unfinished ...>
302535 <... futex resumed> = 0
302536 <... brk resumed> = 0x5f2d70b02000
302535 getrandom( <unfinished ...>
302536 fstat(1, <unfinished ...>
302535 <... getrandom resumed>"\x4e\x0b\xc4\x54\xdd\x83\xfa\xd3",8,GRND_NONBLOCK)
= 8
302536 <... fstat resumed>{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x6),...})
= 0
302535 brk(NULL <unfinished ...>
302536 write(1,"child2: \320\267\320\260\320\277\321\203\321\211\320\265\320\275
\321\201 shared memory /child2_shm \320\270 \320\262\321\213\321\205\320\276\320\264\
\321\204\320\260\320\271\320\273\320\276\320\274 file2\n",91 <unfinished ...>
302535 <... brk resumed> = 0x631689272000
302536 <... write resumed> = 91
302535 brk(0x631689293000 <unfinished ...>
302536 openat(AT_FDCWD,"/dev/shm/child2_shm",O_RDWR|O_NOFOLLOW|O_CLOEXEC <unfinished
...>
302535 <... brk resumed> = 0x631689293000
302536 <... openat resumed> = 3
302536 mmap(NULL,4160,PROT_READ|PROT_WRITE,MAP_SHARED,3,0 <unfinished ...>
302535 fstat(1, <unfinished ...>
302536 <... mmap resumed> = 0x767eee48b000
302535 <... fstat resumed>{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x6),...})
= 0
302536 futex(0x767eee48c000,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSE
<unfinished ...>
302535 write(1,"child1: \320\267\320\260\320\277\321\203\321\211\320\265\320\275
\321\201 shared memory /child1_shm \320\270 \320\262\321\213\321\205\320\276\320\264\
\321\204\320\260\320\271\320\273\320\276\320\274 file1\n",91) = 91
302535 openat(AT_FDCWD,"/dev/shm/child1_shm",O_RDWR|O_NOFOLLOW|O_CLOEXEC) = 3
302535 mmap(NULL,4160,PROT_READ|PROT_WRITE,MAP_SHARED,3,0) = 0x7381a2649000
302535 futex(0x7381a264a000,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FUTEX_BITSE
<unfinished ...>
302509 <... clock_nanosleep resumed>0x7ffc7a5af570) = 0
302509 read(0,"hello\n",1024) = 6
```

```
302509 read(0,"big\n",1024) = 4
302509 read(0,"git\n",1024) = 4
302509 read(0,"create\n",1024) = 7
302509 read(0,"for\n",1024) = 4
302509 read(0,"down\n",1024) = 5
302509 read(0,"",1024) = 0
302509 futex(0x7ed6942ca000,FUTEX_WAKE,1) = 1
302535 <... futex resumed> = 0
302509 futex(0x7ed6942c8000,FUTEX_WAKE,1 <unfinished ...>
302535 write(1,"child1: \320\277\320\276\320\273\321\203\321\207\320\265\320\275\320\
'hello\ngit\n",36 <unfinished ...>
302509 <... futex resumed> = 1
302536 <... futex resumed> = 0
302535 <... write resumed> = 36
302509 write(1,"\"320\224\320\260\320\275\320\275\321\213\320\265 \320\276\321\202\320\
\320\262 \320\264\320\276\321\207\320\265\321\200\320\275\320\270\320\265 \320\277\320\
<unfinished ...>
302536 write(1,"child2: \320\277\320\276\320\273\321\203\321\207\320\265\320\275\320\
'big\ncreate\n",37 <unfinished ...>
302535 write(1,"for'\n",5 <unfinished ...>
302509 <... write resumed> = 71
302536 <... write resumed> = 37
302509 write(1,"\"320\224\320\260\320\275\320\275\321\213\320\265 \320\276\320\261\321\
\320\264\320\276\321\207\320\265\321\200\320\275\320\270\320\274\320\270 \320\277\321\
<unfinished ...>
302535 <... write resumed> = 5
302536 write(1,"down'\n",6 <unfinished ...>
302509 <... write resumed> = 70
302536 <... write resumed> = 6
302509 wait4(302535, <unfinished ...>
302536 openat(AT_FDCWD,"file2",O_WRONLY|O_CREAT|O_TRUNC,0666 <unfinished ...>
302535 openat(AT_FDCWD,"file1",O_WRONLY|O_CREAT|O_TRUNC,0666 <unfinished ...>
302536 <... openat resumed> = 4
302535 <... openat resumed> = 4
302536 write(4,"gib\n",4 <unfinished ...>
302535 write(4,"olleh\n",6 <unfinished ...>
302536 <... write resumed> = 4
302535 <... write resumed> = 6
302536 write(4,"etaerc\n",7 <unfinished ...>
302535 write(4,"tig\n",4 <unfinished ...>
302536 <... write resumed> = 7
302535 <... write resumed> = 4
302536 write(4,"nwod\n",5 <unfinished ...>
302535 write(4,"rof\n",4 <unfinished ...>
302536 <... write resumed> = 5
302535 <... write resumed> = 4
302536 write(1,"child2: \320\267\320\260\320\277\320\270\321\201\320\260\320\275\320\
3 \321\201\321\202\321\200\320\276\320\272 \320\262 file2\n",47 <unfinished
```

```
...>
302535 write(1,"child1: \320\267\320\260\320\277\320\270\321\201\320\260\320\275\320\
3 \321\201\321\202\321\200\320\276\320\272 \320\262 file1\n",47 <unfinished
...>
302536 <... write resumed> = 47
302535 <... write resumed> = 47
302536 close(4 <unfinished ...>
302535 close(4 <unfinished ...>
302536 <... close resumed> = 0
302536 munmap(0x767eee48b000,4160) = 0
302535 <... close resumed> = 0
302536 close(3 <unfinished ...>
302535 munmap(0x7381a2649000,4160 <unfinished ...>
302536 <... close resumed> = 0
302535 <... munmap resumed> = 0
302535 close(3) = 0
302536 exit_group(0 <unfinished ...>
302535 exit_group(0 <unfinished ...>
302536 <... exit_group resumed> = ?
302535 <... exit_group resumed> = ?
302536 +++ exited with 0 ===+
302535 +++ exited with 0 ===+
302509 <... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}],0,NULL) =
302535
302509 ---SIGCHLD {si_signo=SIGCHLD,si_code=CLD_EXITED,si_pid=302536,si_uid=1000,si_s
---
302509 wait4(302536,[{WIFEXITED(s) && WEXITSTATUS(s) == 0}],0,NULL) = 302536
302509 write(1,"\320\224\320\276\321\207\320\265\321\200\320\275\320\270\320\265
\320\277\321\200\320\276\321\206\320\265\321\201\321\201\321\213 \320\267\320\260\320
\321\200\320\260\320\261\320\276\321\202\321\203\n",66) = 66
302509 munmap(0x7ed6942c9000,4160) = 0
302509 close(3) = 0
302509 unlink("/dev/shm/child1_shm") = 0
302509 munmap(0x7ed6942c7000,4160) = 0
302509 close(4) = 0
302509 unlink("/dev/shm/child2_shm") = 0
302509 exit_group(0) = ?
302509 +++ exited with 0 ===+
```