

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

Москва, 2025

Условие

Родительский процесс создает два дочерних процесса. Пользователь вводит имена файлов для каждого дочернего процесса. Родительский процесс принимает от пользователя строки произвольной длины и поочередно пересыпает их дочерним процессам через отдельные каналы. Дочерние процессы удаляют гласные буквы из полученных строк и записывают результат в соответствующие файлы, а также возвращают обработанные строки родительскому процессу.

Цель работы

Изучение механизмов создания процессов, организации межпроцессного взаимодействия через pipes и обработки данных в многопроцессной архитектуре.

Задание

Реализовать программу, в которой родительский процесс создает два дочерних процесса, каждый из которых получает строки через отдельный канал, удаляет гласные буквы и записывает результат в файл.

Вариант

18

Метод решения

Данная программа реализует многопроцессную обработку текстовых данных с использованием каналов (pipes) для межпроцессного взаимодействия. Основной алгоритм: родительский процесс читает строки из стандартного ввода и направляет их поочередно двум дочерним процессам через отдельные каналы. Каждый дочерний процесс удаляет гласные буквы из полученных строк и записывает результат в указанный файл.

Ключевые компоненты:

- Pipe - реализация канала для межпроцессного взаимодействия
- Процессы - создание и управление дочерними процессами

Системные вызовы:

- Linux: pipe, fork, execl, read, write, close, dup2, waitpid

Программа обеспечивает четкое разделение ответственности между процессами и эффективное использование системных ресурсов.

Описание программы

Программа состоит из трех основных модулей:

- **Parent.cpp** - родительский процесс, который создает два дочерних процесса и управляет передачей данных между ними

- **Child.cpp** - дочерний процесс, который удаляет гласные буквы из полученных строк и записывает результат в файл
- **Pipe.cpp** - вспомогательные функции для работы с каналами и процессами

Родительский процесс создает два канала для взаимодействия с дочерними процессами, запрашивает имена файлов для каждого дочернего процесса, затем поочередно передает вводимые строки дочерним процессам. Дочерние процессы получают имя файла, открывают его для записи, затем в цикле читают строки из канала, обрабатывают их (удаляют гласные) и записывают результат в файл.

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 через каналы
- Реализованы два независимых канала передачи данных между родительским и дочерними процессами
- Обеспечена параллельная обработка строк двумя дочерними процессами
- Реализован алгоритм удаления гласных букв из текстовых строк
- Обеспечена запись результатов обработки в отдельные файлы
- Достигнута синхронизация процессов через блокирующие операции чтения/записи

Выводы

В ходе лабораторной работы успешно разработана многопроцессная система обработки текстовых данных с использованием межпроцессного взаимодействия через каналы. Программа демонстрирует корректную работу в Unix-системах, эффективно распределяя обработку данных между несколькими процессами. Были изучены и применены на практике основные механизмы работы с процессами и межпроцессным взаимодействием в операционных системах семейства Unix.

Системные вызовы

Для реализации программы использовались следующие системные вызовы Linux:

- **pipe()** - создание каналов для межпроцессного взаимодействия
- **fork()** - создание дочерних процессов
- **execl()** - запуск исполняемых файлов дочерних процессов
- **read()/write()** - чтение и запись данных в каналы
- **close()** - закрытие файловых дескрипторов
- **dup2()** - перенаправление стандартных потоков ввода-вывода
- **waitpid()** - ожидание завершения дочерних процессов

Фрагмент кода

Листинг 1: Родительский процесс (Parent.cpp)

```
1 #include <iostream>
2 #include <cstdlib>
3 #include <cstdio>
4 #include <unistd.h>
5 #include <string>
6
7 #include "Pipe.hpp"
8
9 #include <sys/wait.h>
10
11 int main() {
12     int pipe1[2];
13     int pipe2[2];
14     PipeCreate(pipe1);
15     PipeCreate(pipe2);
16
17     std::cout << "input file for child1\n";
18     std::string file1;
19     std::cin >> file1;
20
21     std::cout << "input file for child2\n";
22     std::string file2;
23     std::cin >> file2;
24
25     pid_t pid1 = ProcessCreate();
26     if (!pid1) {
27         PipeClose(pipe1[1]);
28         PipeRedirect(pipe1[0], STDIN_FILENO);
29         PipeClose(pipe1[0]);
30         ProcessExecute("./child", "child");
31     }
32     pid_t pid2 = ProcessCreate();
33     if (!pid2) {
34         PipeClose(pipe2[1]);
35         PipeRedirect(pipe2[0], STDIN_FILENO);
36         PipeClose(pipe2[0]);
37         ProcessExecute("./child", "child");
38     }
39     PipeClose(pipe1[0]);
40     PipeClose(pipe2[0]);
41
42     sleep(2);
43
44     std::cout << "Child1 started with PID: " << pid1 << ", file: " << file1 <<
45     std::endl;
46     std::cout << "Child2 started with PID: " << pid2 << ", file: " << file2 <<
47     std::endl;
48
49     PipeWrite(pipe1[1], file1.c_str(), sizeof(file1.c_str()));
50     PipeWrite(pipe2[1], file2.c_str(), sizeof(file2.c_str()));
51
52     std::cout << "\nWrite strings:\n";
53     std::string input;
54
55     int cnt = 0;
56     while (std::cin >> input) {
57         if (cnt % 2 == 0) {
```

```

56         PipeWrite(pipe1[1], input.c_str(), sizeof(input.c_str()));
57     } else {
58         PipeWrite(pipe2[1], input.c_str(), sizeof(input.c_str()));
59     }
60     cnt++;
61 }
62
63 PipeClose(pipe1[1]);
64 PipeClose(pipe2[1]);
65
66 waitpid(pid1, NULL, 0);
67 waitpid(pid2, NULL, 0);
68 std::cout << "Parent process finished. Processed " << cnt << " lines." <<
69             std::endl;
70 return 0;
}

```

Листинг 2: Дочерний процесс (Child.cpp)

```

1 #include <iostream>
2 #include <fstream>
3 #include <string>
4 #include <unistd.h>
5 #include <set>
6
7 #include "Pipe.hpp"
8
9 std::string removeVowels(const std::string& str) {
10     std::string res;
11     std::set<char> ms = {'a', 'e', 'i', 'o', 'u', 'y'};
12
13     for (char c : str) {
14         if (c != 0 && ms.find(tolower(c)) == ms.end()) {
15             res.push_back(c);
16         }
17     }
18
19     return res;
20 }
21
22 int main() {
23     char filename[256];
24     PipeRead(0, filename, sizeof(filename));
25     std::ofstream out(filename, std::ios::app);
26     if (!out.is_open()) {
27         std::cout << "Error, can't open " << filename << std::endl;
28         return 1;
29     }
30
31     char s[256];
32     while(PipeRead(0, &s, sizeof(s)) > 0) {
33         std::string str = removeVowels(s);
34
35         PipeWrite(1, str.c_str(), sizeof(str.c_str()));
36         out << str << std::endl;
37     }
38
39     out.close();
40     std::cout << "Child process " << getpid() << " finished." << std::endl;
41     return 0;
42 }

```

Листинг 3: Реализация работы с каналами (Pipe.cpp)

```
1 #include <iostream>
2 #include <unistd.h>
3 #include <cstdlib>
4
5 #include "Pipe.hpp"
6
7 pid_t ProcessCreate() {
8     pid_t pid = fork();
9     if (pid == -1) {
10         std::cout << "process" << std::endl;
11         exit(-1);
12     }
13     return pid;
14 }
15
16 int PipeCreate(int fd[2]) {
17     int err = pipe(fd);
18     if (err == -1) {
19         std::cout << "pipe" << std::endl;
20         exit(-1);
21     }
22
23     return err;
24 }
25
26 int PipeWrite(int fd, const void * buf, size_t count) {
27     int bytes = write(fd, buf, count);
28     if (bytes == -1) {
29         std::cout << "pipe" << std::endl;
30         exit(-1);
31     }
32
33     return bytes;
34 }
35
36 void PipeClose(int fd) {
37     close(fd);
38 }
39
40 int PipeRead(int fd, void * buf, size_t count) {
41     int bytes = read(fd, buf, count);
42     if (bytes == -1) {
43         std::cout << "pipe" << std::endl;
44         exit(-1);
45     }
46
47     return bytes;
48 }
49
50 void PipeRedirect(int oldfd, int newfd) {
51     if (dup2(oldfd, newfd) == -1) {
52         std::cerr << " " << std::endl;
53         exit(-1);
54     }
55 }
```

```
57 void ProcessExecute(const char* program, const char* arg) {
58     execl(program, program, arg, NULL);
59     std::cerr << "
60         << std::endl;
61     exit(-1);
62 }
```

Листинг 4: Системные вызовы


```

72 mprotect(0x79bdde70f000, 8192, PROT_READ) = 0
73 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY
    }) = 0
74 munmap(0x79bdde6c1000, 77848) = 0
75 readlink("/proc/self/exe", "/home/hom1e/codes/OC/lab1/build/..., 4096) = 38
76 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x79bdde70e000
77 open("/proc/self/cmdline", O_RDONLY) = 3
78 read(3, "./parent\0", 4096) = 9
79 read(3, "", 4087) = 0
80 close(3) = 0
81 munmap(0x79bdde70e000, 4096) = 0
82 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x79bdde70e000
83 open("/proc/self/environ", O_RDONLY) = 3
84 read(3, "COLORTERM=truecolor\0LC_ADDRESS=r"..., 4096) = 2123
85 read(3, "", 1973) = 0
86 close(3) = 0
87 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x79bdde6d3000
88 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x79bdde6d1000
89 mmap(NULL, 3481600, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x79bdd0ae000
90 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x79bdde6d0000
91 prlimit64(0, RLIMIT_CORE, NULL, {rlim_cur=0, rlim_max=0}) = 0
92 prlimit64(0, RLIMIT_CORE, {rlim_cur=0, rlim_max=0}, NULL) = 0
93 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x79bdde6cf000
94 open("/proc/self/maps", O_RDONLY) = 3
95 read(3, "56d38f43c000-56d38f43f000 r--p 0"..., 4096) = 4028
96 read(3, "79bdde6d7000-79bdde6d9000 r--p 0"..., 68) = 68
97 close(3) = 0
98 munmap(0x79bdde6cf000, 4096) = 0
99 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x79bdde6ce000
100 open("/proc/self/maps", O_RDONLY) = 3
101 read(3, "56d38f43c000-56d38f43f000 r--p 0"..., 8192) = 4028
102 read(3, "79bdde6d7000-79bdde6d9000 r--p 0"..., 4164) = 974
103 read(3, "", 3190) = 0
104 close(3) = 0
105 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x79bdde6cd000
106 open("/proc/self/maps", O_RDONLY) = 3
107 read(3, "56d38f43c000-56d38f43f000 r--p 0"..., 4096) = 4028
108 read(3, "79bdde6d7000-79bdde6d9000 r--p 0"..., 68) = 68
109 close(3) = 0
110 munmap(0x79bdde6cd000, 4096) = 0
111 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x79bdde6cc000
112 open("/proc/self/maps", O_RDONLY) = 3
113 read(3, "56d38f43c000-56d38f43f000 r--p 0"..., 8192) = 4028
114 read(3, "79bdde6d7000-79bdde6d9000 r--p 0"..., 4164) = 974
115 read(3, "", 3190) = 0
116 close(3) = 0
117 munmap(0x79bdde6cc000, 8192) = 0
118 mmap(0x7fff7000, 268435456, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_ANONYMOUS|MAP_NORESERVE, -1, 0) = 0x7fff7000
119 madvise(0x7fff7000, 268435456, MADV_NOHUGEPAGE) = 0

```

```

120 madvise(0x7fff7000, 268435456, MADV_DONTDUMP) = 0
121 mmap(0x2008fff7000, 15392894357504, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED
      |MAP_ANONYMOUS|MAP_NORESERVE, -1, 0) = 0x2008ffff7000
122 madvise(0x2008fff7000, 15392894357504, MADV_NOHUGEPAGE) = 0
123 madvise(0x2008fff7000, 15392894357504, MADV_DONTDUMP) = 0
124 mmap(0x8ffff7000, 2199023255552, PROT_NONE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS|
      MAP_NORESERVE, -1, 0) = 0x8ffff7000
125 sigaltstack(NULL, {ss_sp=NULL, ss_flags=SS_DISABLE, ss_size=0}) = 0
126 mmap(NULL, 61440, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde68d000
127 sigaltstack({ss_sp=0x79bdde68d000, ss_flags=0, ss_size=58112}, NULL) = 0
128 rt_sigaction(SIGSEGV, {sa_handler=0x79bdddcba3e0, sa_mask=[], sa_flags=
      SA_RESTORER|SA_ONSTACK|SA_NODEFER|SA_SIGINFO, sa_restorer=0x79bddd442520},
      NULL, 8) = 0
129 rt_sigaction(SIGBUS, {sa_handler=0x79bdddcba3e0, sa_mask=[], sa_flags=
      SA_RESTORER|SA_ONSTACK|SA_NODEFER|SA_SIGINFO, sa_restorer=0x79bddd442520},
      NULL, 8) = 0
130 rt_sigaction(SIGFPE, {sa_handler=0x79bdddcba3e0, sa_mask=[], sa_flags=
      SA_RESTORER|SA_ONSTACK|SA_NODEFER|SA_SIGINFO, sa_restorer=0x79bddd442520},
      NULL, 8) = 0
131 mmap(0x5000000000000, 4398046523392, PROT_NONE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS|MAP_NORESERVE, -1, 0) = 0x5000000000000
132 mmap(0x5400000000000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS, -1, 0) = 0x5400000000000
133 mmap(NULL, 8388608, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, -1, 0)
      = 0x79bddd800000
134 mmap(NULL, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde67f000
135 mmap(NULL, 33554432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
      0x79bdda800000
136 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6cd000
137 getpid() = 344145
138 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
      } = 0
139 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6cc000
140 open("/proc/self/maps", O_RDONLY) = 3
141 read(3, "7fff7000-8fff7000 rw-p 00000000 "..., 4096) = 4082
142 read(3, "79bdde6bf000-7", 14) = 14
143 close(3) = 0
144 munmap(0x79bdde6cc000, 4096) = 0
145 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6cb000
146 open("/proc/self/maps", O_RDONLY) = 3
147 read(3, "7fff7000-8fff7000 rw-p 00000000 "..., 8192) = 4082
148 read(3, "79bdde6bf000-79bdde6c0000 r--p 0"..., 4110) = 1249
149 read(3, "", 2861) = 0
150 close(3) = 0
151 munmap(0x79bdde6ce000, 8192) = 0
152 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6cf000
153 open("/proc/self/maps", O_RDONLY) = 3
154 read(3, "7fff7000-8fff7000 rw-p 00000000 "..., 4096) = 4082
155 read(3, "79bdde6bf000-7", 14) = 14
156 close(3) = 0
157 munmap(0x79bdde6cf000, 4096) = 0
158 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6ce000
159 open("/proc/self/maps", O_RDONLY) = 3

```

```

160 read(3, "7fff7000-8fff7000 rw-p 00000000 "..., 8192) = 4082
161 read(3, "79bdde6bf000-79bdde6c0000 r--p 0"..., 4110) = 1249
162 read(3, "", 2861) = 0
163 close(3) = 0
164 munmap(0x79bdde6ce000, 8192) = 0
165 mmap(0x10002ac40000, 1044480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS|MAP_NORESERVE, -1, 0) = 0x10002ac40000
166 madvise(0x10002ac40000, 1044480, MADV_NOHUGEPAGE) = 0
167 madvise(0x10002ac40000, 1044480, MADV_DONTDUMP) = 0
168 sigaltstack(NULL, {ss_sp=0x79bdde68d000, ss_flags=0, ss_size=58112}) = 0
169 mmap(NULL, 2097152, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdda600000
170 munmap(0x79bdda700000, 1048576) = 0
171 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6cf000
172 mmap(NULL, 2097152, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdda400000
173 munmap(0x79bdda500000, 1048576) = 0
174 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6ce000
175 mmap(NULL, 1179648, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bddd6e0000
176 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6ca000
177 mmap(NULL, 2097152, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdda200000
178 munmap(0x79bdda300000, 1048576) = 0
179 mmap(NULL, 2097152, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdda000000
180 munmap(0x79bdda100000, 1048576) = 0
181 munmap(0x79bdde6ca000, 4096) = 0
182 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6ca000
183 munmap(0x79bdde6ca000, 4096) = 0
184 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6ca000
185 munmap(0x79bdde6ca000, 4096) = 0
186 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6ca000
187 munmap(0x79bdde6ca000, 4096) = 0
188 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6ca000
189 munmap(0x79bdde6ca000, 4096) = 0
190 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6ca000
191 munmap(0x79bdde6ca000, 4096) = 0
192 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6ca000
193 munmap(0x79bdde6ca000, 4096) = 0
194 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6ca000
195 munmap(0x79bdde6ca000, 4096) = 0
196 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6ca000
197 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6c9000
198 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6c7000
199 clock_gettime(CLOCK_MONOTONIC, {tv_sec=34908, tv_nsec=325685991}) = 0
200 mmap(0x507000000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS|MAP_NORESERVE, -1, 0) = 0x507000000000

```

```

MAP_ANONYMOUS, -1, 0) = 0x507000000000
201 mmap(0x507e00000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS, -1, 0) = 0x507e00000000
202 mmap(NULL, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde66f000
203 clock_gettime(CLOCK_MONOTONIC, {tv_sec=34908, tv_nsec=325886042}) = 0
204 mmap(0x503000000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS, -1, 0) = 0x503000000000
205 mmap(0x503e00000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS, -1, 0) = 0x503e00000000
206 clock_gettime(CLOCK_MONOTONIC, {tv_sec=34908, tv_nsec=326097405}) = 0
207 mmap(0x50b000000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS, -1, 0) = 0x50b000000000
208 mmap(0x50be00000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS, -1, 0) = 0x50be00000000
209 clock_gettime(CLOCK_MONOTONIC, {tv_sec=34908, tv_nsec=326568604}) = 0
210 mmap(0x524000000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS, -1, 0) = 0x524000000000
211 mmap(0x524e00000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS, -1, 0) = 0x524e00000000
212 mmap(0x531000000000, 131072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS, -1, 0) = 0x531000000000
213 mmap(0x531e00000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS, -1, 0) = 0x531e00000000
214 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6c6000
215 mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde6c5000
216 mmap(NULL, 36864, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
      x79bdde666000
217 futex(0x79bddda2977c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
218 pipe2([3, 4], 0) = 0
219 pipe2([5, 6], 0) = 0
220 newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...},
      AT_EMPTY_PATH) = 0
221 mmap(0x519000000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS, -1, 0) = 0x519000000000
222 mmap(0x519e00000000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
      MAP_ANONYMOUS, -1, 0) = 0x519e00000000
223 write(1, "input file for child1\n", 22) = 22
224 newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...},
      AT_EMPTY_PATH) = 0
225 read(0, "file1\n", 1024) = 6
226 write(1, "input file for child2\n", 22) = 22
227 read(0, "file2\n", 1024) = 6
228 clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
      child_tidptr=0x79bdde69ca90) = 344271
229 clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
      child_tidptr=0x79bdde69ca90) = 344272
230 close(3) = 0
231 close(5) = 0
232 clock_nanosleep(CLOCK_REALTIME, 0, {tv_sec=2, tv_nsec=0}, 0x7ffd56a379a0) = 0
233 write(1, "Child1 started with PID: 344271,..., 45) = 45
234 write(1, "Child2 started with PID: 344272,..., 45) = 45
235 write(4, "file1\0\0\0", 8) = 8
236 write(6, "file2\0\0\0", 8) = 8
237 write(1, "\nWrite strings:\n", 16) = 16
238 read(0, "kek\0", 1024) = 6
239 write(4, "kek\0\0\0", 8) = 8
240 read(0, "pasdasd\n", 1024) = 8

```

```

241 write(6, "pasdasd\0", 8) = 8
242 read(0, "vmvm,asdasd\n", 1024) = 12
243 write(4, "vmvm,asd", 8) = 8
244 read(0, "sllsllslls\n", 1024) = 11
245 write(6, "sllsllslls", 8) = 8
246 read(0, "aabobabaaaaaaaaa\n", 1024) = 16
247 write(4, "aabobaba", 8) = 8
248 read(0, "pipiska\n", 1024) = 8
249 write(6, "pipiska\0", 8) = 8
250 read(0, "", 1024) = 0
251 close(4) = 0
252 close(6) = 0
253 wait4(344271, NULL, 0, NULL) = ? ERESTARTSYS (To be restarted if
      SA_RESTART is set)
--- SIGINT {si_signo=SIGINT, si_code=SI_KERNEL} ---
254 +++ killed by SIGINT +++
255

```