

**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: Е. Г. Туймуков
Группа: М8О-208БВ-24
Преподаватель: Е. С. Миронов**

Москва, 2025

Условие

Составить и отладить программу на языке Си, 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: Целью является приобретение практических навыков в Освоение принципов работы с файловыми системами, Обеспечение обмена данных между процессами посредством технологии «File mapping»

Задание: Правило фильтрации: строки длиной более 10 символов записываются в файл для «длинных» через memory-mapped file, остальные — в файл для «коротких». Дочерние процессы считывают данные из соответствующих файлов и инвертируют строки.

Вариант: 20

Метод решения

Программа реализует многопроцессное взаимодействие с использованием технологии file mapping (отображаемых файлов). Основной процесс создает два дочерних процесса. Родитель читает строки с клавиатуры и в зависимости от длины строки (более 10 символов или нет) записывает их в соответствующий mmap-файл для одного из детей. Дочерние процессы получают доступ к mmap-файлам, читают строки, инвертируют их и записывают результат в соответствующие выходные файлы.

Ключевые компоненты:

ParentProcess — управляет созданием mmap-файлов, записью строк и запуском дочерних процессов.

ChildProcess — отвечает за чтение данных из mmap-файла, инвертирование строк и запись результата в выходной файл.

os.h и его реализации (oslinux.cpp, oswin.cpp) — инкапсулируют системные вызовы для работы с процессами и файлами, обеспечивая кроссплатформенный интерфейс.

Системные вызовы:

Linux/macOS: fork, execvp, mmap, munmap, open, close, ftruncate, waitpid.

Windows: CreateProcess, MapViewOfFile, UnMapViewOfFile, CreateFile, CloseHandle, SetFilePointer/SetEndOfFile, WaitForSingleObject.

Программа использует объектно-ориентированный подход: управление процессами и синхронизацией реализовано через классы и обертки над системными вызовами, что обеспечивает модульность и кроссплатформенность

Описание программы

Программа выполняет фильтрацию строк по длине и их реверс с использованием нескольких процессов. Основной процесс читает строки и записывает их в два memory-mapped файла: строки длиной больше 10 символов — в один файл, строки длиной 10 и меньше — в другой. Каждый дочерний процесс мапит соответствующий файл, инвертирует строки и записывает результат в свой выходной файл.

Синхронизация процессов осуществляется через доступ к mmap-файлам: родитель постепенно пишет строки, а дочерние процессы читают их по мере появления данных. Таким

образом, обеспечивается корректное взаимодействие и обмен данными между процессами.

Результаты

разработанная программа реализует обработку строк с разделением по длине через memory-mapped файлы и порождает дочерние процессы для инверсии строк. Основные результаты:

- Корректная обработка данных с разделением строк: длина 10 символов — один файл, >10 символов — другой.
- Дочерние процессы инвертируют строки и записывают результат в отдельные выходные файлы.
- Используются memory-mapped файлы для обмена данными между процессами.
- Обработаны системные ошибки при работе с файлами и процессами.
- Реализована кроссплатформенность за счет оберток для Windows и Linux.

Выводы

В ходе лабораторной работы №3 я разработал программу на языке С, демонстрирующую взаимодействие между процессами с использованием технологии memory-mapped files. Я понял, что основным процессом можно эффективно управлять распределением данных между дочерними процессами, при этом дочерние процессы могут параллельно обрабатывать данные. Реализация правила фильтрации, при котором строки длиной более 10 символов направляются в один файл, а строки длиной 10 и меньше — в другой, показала на практике, как удобно использовать отображаемые файлы для обмена данными между процессами.

Я также усвоил, что memory-mapped файлы позволяют избежать лишнего копирования данных и обеспечивают прямой доступ к общей памяти, что значительно упрощает и ускоряет взаимодействие между процессами. Работая с файлами, я столкнулся с необходимостью тщательно проверять системные ошибки при открытии, отображении и закрытии файлов, а также при создании дочерних процессов — это помогло мне лучше понять, как низкоуровнево управлять ресурсами операционной системы.

Кроме того, я понял, как правильно организовать последовательную запись и параллельное чтение данных: родительский процесс постепенно записывает строки в memory-mapped файлы, а дочерние процессы считывают их и обрабатывают реверсирование. Такой опыт позволил мне осознать важность синхронизации и контроля за порядком доступа к общим ресурсам, даже если в этой работе синхронизация реализована упрощённо.

В результате лабораторной работы я научился проектировать межпроцессное взаимодействие с использованием memory-mapped файлов, оценивать производительность системы при параллельной обработке данных и видеть реальные преимущества разделяемой памяти для эффективного обмена информацией между процессами.

Исходная программа

```
#pragma once
#include <string>
#include "os.h"

class ChildProcess
{
private:
    ProcessHandle pid;
    std::string mmap_filename;
    std::string file_name;
    bool is_child1;

public:
    ChildProcess(const std::string &mmap_f,const std::string &out_f,bool is_c1);
    void execute();
    ProcessHandle getPid() const { return pid; }
};

#pragma once
#include <string>

#ifndef _WIN32
#include <windows.h>
typedef HANDLE FileHandle;
typedef HANDLE ProcessHandle;
typedef HANDLE PipeHandle;
#define INVALID_PIPE_HANDLE INVALID_HANDLE_VALUE
#else
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
typedef int FileHandle;
typedef int PipeHandle;
typedef pid_t ProcessHandle;
#define INVALID_PIPE_HANDLE -1
#endif

FileHandle OpenFile(const std::string &filename);
void SetFileSize(FileHandle file,size_t size);
void *MapFile(FileHandle file,size_t size);
void UnmapFile(void *mapped_memory,size_t size);

void CloseFile(FileHandle file);
ProcessHandle CreateChildProcess(const char *exe,char *const *argv);
int WaitProcess(ProcessHandle pid);
```

```

ProcessHandle Fork();
int Exec(const char *exe,char *const *argv);

#pragma once
#include "child_process.hpp"
#include <string>

class ParentProcess
{
private:
    ChildProcess *child1;
    ChildProcess *child2;
    std::string file1,file2;
    std::string filename;

public:
    ParentProcess();
    ~ParentProcess();
    void start();
};

#include "../include/os.h"
#include <pthread.h>
#include <errno.h>
#include <stdio.h>

int ThreadCreate(ThreadHandle* handle,void* (*func)(void*),void* arg) {
    int res = pthread_create(handle,nullptr,func,arg);
    if (res != 0) {
        perror("pthread_create failed");
        return -1;
    }
    return 0;
}

void ThreadJoin(ThreadHandle handle) {
    int res = pthread_join(handle,nullptr);
    if (res != 0) {
        perror("pthread_join failed");
    }
}

#include "../include/child_process.hpp"
#include "../include/os.h"

ChildProcess::ChildProcess(const std::string &mmap_f,const std::string &out_f,bool is_c1)

```

```
: mmap_filename(mmap_f),file_name(out_f),is_child1(is_c1),pid(INVALID_PIPE_HANDLE)
{ }

void ChildProcess::execute()
{
#ifdef _WIN32
const char *exe = "child.exe";
#else
const char *exe = "./child";
#endif
char *argv[] = {(char *)"child", (char *)mmap_filename.c_str(), (char *)file_name.c_str()};
pid = CreateChildProcess(exe, argv);
}

#include <iostream>
#include <fstream>
#include <string>
#include "../include/os.h"

int main(int argc,char **argv)
{
if (argc <2)
{
std::cerr <<"No filename provided" <<std::endl;
return 1;
}

std::string filename = argv[1];
size_t mapped_size = 1024;
std::string output_filename = argv[2];
std::ofstream outfile(output_filename);
int fd = OpenFile(filename);
void *mapped_memory = MapFile(fd,mapped_size);
CloseFile(fd);

char *mem_ptr = static_cast<char *>(mapped_memory);
size_t offset = 0;

while (offset <mapped_size)
{
char *mem_ptr = static_cast<char *>(mapped_memory) + offset;
if (mem_ptr[0] == 0)
{
continue;
}

std::string line(mem_ptr + 1);
size_t len = line.find('\n');
```

```

if (len != std::string::npos)
line.resize(len);

std::reverse(line.begin(),line.end());
std::cout << line << std::endl;
outfile << line << std::endl;

mem_ptr[0] = 0;
offset += len + 2;
UnmapFile(mapped_memory,mapped_size);
return 0;
}

#include "../include/os.h"
#include <fcntl.h>
#include <unistd.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <cstdio>
#include <cstring>
#include <string>

FileHandle OpenFile(const std::string &filename)
{
int fd = open(filename.c_str(),O_RDWR | O_CREAT,0666);
if (fd == -1)
perror("open");
return fd;
}

void SetFileSize(FileHandle file,size_t size)
{
if (ftruncate(file,size) == -1)
{
perror("ftruncate");
}
}

void *MapFile(FileHandle file,size_t size)
{
void *addr = mmap(nullptr,size,PROT_READ | PROT_WRITE,MAP_SHARED,file,0);
if (addr == MAP_FAILED)
{
perror("mmap");
return nullptr;
}
}

```

```
return addr;
}

void UnmapFile(void *mapped_memory,size_t size)
{
if (munmap(mapped_memory,size) == -1)
{
perror("munmap");
}
}

void CloseFile(FileHandle file)
{
if (close(file) == -1)
{
perror("close");
}
}

ProcessHandle CreateChildProcess(const char *exe,char *const *argv)
{
pid_t pid = Fork();
if (pid == -1)
{
perror("fork");
return -1;
}
if (pid == 0)
{
Exec(exe,argv);
}
return pid;
};

int WaitProcess(ProcessHandle pid)
{
int status;
return waitpid(pid,&status,0);
}

ProcessHandle Fork()
{
pid_t pid = fork();
if (pid == -1)
{
perror("fork");
}
return pid;
}
```

```
int Exec(const char *exe,char *const *argv)
{
execvp(exe,argv);
perror("execvp");
_exit(1);
return -1;
}

#include "../include/os.h"
#include <windows.h>
#include <iostream>
#include <string>

FileHandle OpenFile(const std::string &filename)
{
HANDLE hFile = CreateFileA(
filename.c_str(),
GENERIC_READ | GENERIC_WRITE,
FILE_SHARE_READ | FILE_SHARE_WRITE,
NULL,
OPEN_ALWAYS,
FILE_ATTRIBUTE_NORMAL,
NULL);

if (hFile == INVALID_HANDLE_VALUE)
std::cerr <<"OpenFile failed (" <<GetLastError() <<")" <<std::endl;

return hFile;
}

void SetFileSize(FileHandle file,size_t size)
{
LARGE_INTEGER li;
li.QuadPart = size;
if (!SetFilePointerEx(file,li,NULL,FILE_BEGIN) || !SetEndOfFile(file))
std::cerr <<"SetFileSize failed (" <<GetLastError() <<")" <<std::endl;
}

void* MapFile(FileHandle file,size_t size)
{
HANDLE hMap = CreateFileMapping(file,NULL,PAGE_READWRITE,0,(DWORD)size,NULL);
if (!hMap)
{
std::cerr <<"CreateFileMapping failed (" <<GetLastError() <<")" <<std::endl;
return nullptr;
}
```

```
void* addr = MapViewOfFile(hMap,FILE_MAP_ALL_ACCESS,0,0,size);
CloseHandle(hMap);

if (!addr)
std::cerr <<"MapViewOfFile failed (" <<GetLastError() <<")" <<std::endl;

return addr;
}

void UnmapFile(void* mapped_memory,size_t /*size*/)
{
if (!UnmapViewOfFile(mapped_memory))
std::cerr <<"UnmapViewOfFile failed (" <<GetLastError() <<")" <<std::endl;
}

void CloseFile(FileHandle file)
{
if (!CloseHandle(file))
std::cerr <<"CloseFile failed (" <<GetLastError() <<")" <<std::endl;
}

// Создание дочернего процесса (без пайпов)
ProcessHandle CreateChildProcess(const char* exe,char* const* argv)
{
std::string cmdline;
for (int i = 0; argv[i]; ++i)
{
if (i >0) cmdline += " ";
cmdline += argv[i];
}

STARTUPINFO si;
PROCESS_INFORMATION pi;
ZeroMemory(&si,sizeof(si));
si.cb = sizeof(si);
ZeroMemory(&pi,sizeof(pi));

if (!CreateProcessA(NULL,(LPSTR)cmdline.c_str(),NULL,NULL, FALSE,0,NULL,NULL,&si,&pi))
{
std::cerr <<"CreateProcess failed (" <<GetLastError() <<")" <<std::endl;
return INVALID_HANDLE_VALUE;
}

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

int WaitProcess(ProcessHandle pid)
```

```

{
WaitForSingleObject(pid,INFINITE);
CloseHandle(pid);
return 0;
}

ProcessHandle Fork() { std::cerr <<"Fork not supported on Windows" <<std::endl;
return INVALID_HANDLE_VALUE; }
int Exec(const char* /*exe*/,char* const* /*argv*/) { std::cerr <<"Exec not
supported on Windows" <<std::endl; return -1; }

#include "../include/parent_process.hpp"
#include "../include/os.h"
#include <iostream>

ParentProcess::ParentProcess()
: child1(nullptr),child2(nullptr)
{
}

ParentProcess::~ParentProcess()
{
delete child1;
delete child2;
}

void ParentProcess::start()
{
std::string mmap_file_short,mmap_file_long;
std::string out_file_short,out_file_long;

std::cout <<"Enter mmap file for short lines: ";
std::getline(std::cin,mmap_file_short);
std::cout <<"Enter mmap file for long lines: ";
std::getline(std::cin,mmap_file_long);
std::cout <<"Enter output file for short lines: ";
std::getline(std::cin,out_file_short);
std::cout <<"Enter output file for long lines: ";
std::getline(std::cin,out_file_long);

const size_t mapped_size = 1024;
void *mapped_short = nullptr;
void *mapped_long = nullptr;

int fd_short = OpenFile(mmap_file_short);
SetFileSize(fd_short,mapped_size);
mapped_short = MapFile(fd_short,mapped_size);
}

```

```
CloseFile(fd_short);

int fd_long = OpenFile(mmap_file_long);
SetFileSize(fd_long,mapped_size);
mapped_long = MapFile(fd_long,mapped_size);
CloseFile(fd_long);

child1 = new ChildProcess(mmap_file_short,out_file_short,false);
child1->execute();

child2 = new ChildProcess(mmap_file_long,out_file_long,false);
child2->execute();

std::cout <<"Enter lines (empty line to end):" <<std::endl;
std::string line;
size_t offset_short = 0;
size_t offset_long = 0;

while (std::getline(std::cin,line))
{
if (line.empty())
break;

const char *data = line.c_str();
size_t len = line.size();

if (len <= 10)
{
if (offset_short + len + 2 > mapped_size)
break;
char *mem_ptr = static_cast<char *>(mapped_short) + offset_short;
memcpy(mem_ptr + 1,data,len);
mem_ptr[len + 1] = '\n';
mem_ptr[0] = 1;

while (mem_ptr[0] != 0)
{
}

offset_short += len + 2;
}
else
{
if (offset_long + len + 2 > mapped_size)
break;
char *mem_ptr = static_cast<char *>(mapped_long) + offset_long;
memcpy(mem_ptr + 1,data,len);
mem_ptr[len + 1] = '\n';
}
```

```

mem_ptr[0] = 1;

while (mem_ptr[0] != 0)
{
}

offset_long += len + 2;
}

std::cout << "Parent: input finished." << std::endl;

WaitProcess(child1->getPid());
WaitProcess(child2->getPid());

std::cout << "Parent: children finished." << std::endl;

if (mapped_short)
UnmapFile(mapped_short,mapped_size);
if (mapped_long)
UnmapFile(mapped_long,mapped_size);
}

#include "../include/parent_process.hpp"

int main() {
ParentProcess parent;
parent.start();
return 0;
}

```

Логи

```

10341 execve("./main",["./main","16"],["SHELL=/bin/bash","SESSION_MANAGER=local/24243",
ERROR;JS LOG","HOME=/home/modmod","USERNAME=modmod","IM_CONFIG_PHASE=1","LANG=ru_RU.U
%s %s","XDG_SESSION_CLASS=user","TERM=xterm-256color","GTK_PATH=/snap/code/206/usr/li
/usr/bin/lesspipe %s","USER=modmod","GTK_PATH_VSCODE_SNAP_ORIG=","VSCODE_GIT_IPC_HAND
","FONTCONFIG_FILE=/etc/fonts/fonts.conf","VSCODE_GIT_ASKPASS_MAIN=/snap/code/206/usr
"GDK_BACKEND=x11","PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
= 0
10341 brk(NULL) = 0x615731163000
10341 mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) =
0x78d6e697d000
10341 access("/etc/ld.so.preload",R_OK) = -1 ENOENT (Нет такого файла или каталога)
10341 openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC) = 3
10341 fstat(3,{st_dev=makedev(0x103,0x8),st_ino=7080911,st_mode=S_IFREG|0644,st_nlink
/* 2025-10-13T12:49:14.467681259+0300 */,st_atime_nsec=467681259,st_mtime=1760348954

```



```
10341 mprotect(0x615728746000,4096,PROT_READ) = 0
10341 mprotect(0x78d6e69bb000,8192,PROT_READ) = 0
10341 prlimit64(0,RLIMIT_STACK,NULL,{rlim_cur=8192*1024,rlim_max=RLIM64_INFINITY}) = 0
10341 munmap(0x78d6e6965000,94467) = 0
10341 futex(0x78d6e687a7bc,FUTEX_WAKE_PRIVATE,2147483647) = 0
10341 getrandom("\x9e\x15\xa8\xd6\x30\xe3\xe4\x1c",8,GRND_NONBLOCK) = 8
10341 brk(NULL) = 0x615731163000
10341 brk(0x615731184000) = 0x615731184000
10341 fstat(1,{st_dev=makedev(0,0x1a),st_ino=3,st_mode=S_IFCHR|0620,st_nlink=1,st_uid /* 2025-10-13T13:04:48+0300 */,st_atime_nsec=0,st_mtime=1760349888 /* 2025-10-13T13:0 *//*,st_mtime_nsec=0,st_ctime=1760348474 /* 2025-10-13T12:41:14.222497184+0300 *//*,st_ctime_nsec=222497184}) = 0
10341 write(1,"Before sorting:\n",16) = 16
10341 write(1,"746 358 274 817 780 93 905 419 265 147 \n",40) = 40
10341 rt_sigaction(SIGRT_1,{sa_handler=0x78d6e6299530,sa_mask=[],sa_flags=SA_RESTORER} = 0
10341 rt_sigprocmask(SIG_UNBLOCK,[RTMIN RT_1],NULL,8) = 0
10341 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0) = 0x78d6e59ff000
10341 mprotect(0x78d6e5a00000,8388608,PROT_READ|PROT_WRITE) = 0
10341 rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
10341 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY =>{parent_tid=[10342]},88) = 10342
10341 rt_sigprocmask(SIG_SETMASK,[],NULL,8) = 0
10342 rseq(0x78d6e61ffffe0,0x20,0,0x53053053 <unfinished ...>
10341 futex(0x78d6e61ff990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,10342,NULL,FUTEX_BI <unfinished ...>
10342 <... rseq resumed>) = 0
10342 set_robust_list(0x78d6e61ff9a0,24) = 0
10342 rt_sigprocmask(SIG_SETMASK,[],NULL,8) = 0
10342 write(1,"Thread started. Active threads: 1\n",34) = 34
10342 mmap(NULL,134217728,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x78d6dd800000
10342 munmap(0x78d6dd800000,41943040) = 0
10342 munmap(0x78d6e4000000,25165824) = 0
10342 mprotect(0x78d6e0000000,135168,PROT_READ|PROT_WRITE) = 0
10342 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0) = 0x78d6e51fe000
10342 mprotect(0x78d6e51ff000,8388608,PROT_READ|PROT_WRITE) = 0
10342 rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
10342 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY =>{parent_tid=[10343]},88) = 10343
10343 rseq(0x78d6e59fefef0,0x20,0,0x53053053 <unfinished ...>
10342 rt_sigprocmask(SIG_SETMASK,[],<unfinished ...>
10343 <... rseq resumed>) = 0
10342 <... rt_sigprocmask resumed>NULL,8) = 0
10343 set_robust_list(0x78d6e59fe9a0,24 <unfinished ...>
10342 futex(0x78d6e59fe990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,10343,NULL,FUTEX_BI
```

```
<unfinished ...>
10343 <... set_robust_list resumed>      = 0
10343 rt_sigprocmask(SIG_SETMASK, [] ,NULL,8) = 0
10343 write(1,"Thread started. Active threads: 2\n",34) = 34
10343 mmap(NULL,134217728,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x78d6d8000000
10343 munmap(0x78d6dc000000,67108864)   = 0
10343 mprotect(0x78d6d8000000,135168,PROT_READ|PROT_WRITE) = 0
10343 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0)
= 0x78d6e49fd000
10343 mprotect(0x78d6e49fe000,8388608,PROT_READ|PROT_WRITE) = 0
10343 rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
10343 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY
=>{parent_tid=[10344]},88) = 10344
10344 rseq(0x78d6e51fdf0,0x20,0,0x53053053 <unfinished ...>
10343 rt_sigprocmask(SIG_SETMASK, [] , <unfinished ...>
10344 <... rseq resumed>                  = 0
10343 <... rt_sigprocmask resumed>NULL,8) = 0
10344 set_robust_list(0x78d6e51fd9a0,24 <unfinished ...>
10343 futex(0x78d6e51fd990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,10344,NULL,FUTEX_BI
<unfinished ...>
10344 <... set_robust_list resumed>      = 0
10344 rt_sigprocmask(SIG_SETMASK, [] ,NULL,8) = 0
10344 write(1,"Thread started. Active threads: 3\n",34) = 34
10344 mmap(0x78d6dc000000,67108864,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x78d6dc000000
10344 mprotect(0x78d6dc000000,135168,PROT_READ|PROT_WRITE) = 0
10344 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0)
= 0x78d6e41fc000
10344 mprotect(0x78d6e41fd000,8388608,PROT_READ|PROT_WRITE) = 0
10344 rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
10344 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY
=>{parent_tid=[10345]},88) = 10345
10345 rseq(0x78d6e49fcfe0,0x20,0,0x53053053 <unfinished ...>
10344 rt_sigprocmask(SIG_SETMASK, [] , <unfinished ...>
10345 <... rseq resumed>                  = 0
10344 <... rt_sigprocmask resumed>NULL,8) = 0
10345 set_robust_list(0x78d6e49fc9a0,24 <unfinished ...>
10344 futex(0x78d6e49fc990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,10345,NULL,FUTEX_BI
<unfinished ...>
10345 <... set_robust_list resumed>      = 0
10345 rt_sigprocmask(SIG_SETMASK, [] ,NULL,8) = 0
10345 write(1,"Thread started. Active threads: 4\n",34) = 34
10345 mmap(NULL,134217728,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x78d6d0000000
10345 munmap(0x78d6d4000000,67108864)   = 0
10345 mprotect(0x78d6d0000000,135168,PROT_READ|PROT_WRITE) = 0
10345 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0)
= 0x78d6d77ff000
10345 mprotect(0x78d6d7800000,8388608,PROT_READ|PROT_WRITE) = 0
```

```
10345 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0
10345 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY
=>{parent_tid=[10346]}, 88) = 10346
10346 rseq(0x78d6d7ffffe0, 0x20, 0, 0x53053053 <unfinished ...>
10345 rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>
10346 <... rseq resumed>) = 0
10345 <... rt_sigprocmask resumed>NULL, 8) = 0
10346 set_robust_list(0x78d6d7fff9a0, 24 <unfinished ...>
10345 futex(0x78d6d7fff990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 10346, NULL, FUTEX_BI
<unfinished ...>
10346 <... set_robust_list resumed>) = 0
10346 rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
10346 write(1, "Thread started. Active threads: 5\n", 34) = 34
10346 mmap(0x78d6d4000000, 67108864, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0x78d6cc000000
10346 mprotect(0x78d6cc000000, 135168, PROT_READ|PROT_WRITE) = 0
10346 mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0)
= 0x78d6d6ffe000
10346 mprotect(0x78d6d6ffff000, 8388608, PROT_READ|PROT_WRITE) = 0
10346 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0
10346 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY
<unfinished ...>
10347 rseq(0x78d6d77fef0, 0x20, 0, 0x53053053 <unfinished ...>
10346 <... clone3 resumed>=>{parent_tid=[10347]}, 88) = 10347
10347 <... rseq resumed>) = 0
10346 rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>
10347 set_robust_list(0x78d6d77fe9a0, 24 <unfinished ...>
10346 <... rt_sigprocmask resumed>NULL, 8) = 0
10347 <... set_robust_list resumed>) = 0
10347 rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>
10346 futex(0x78d6d77fe990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 10347, NULL, FUTEX_BI
<unfinished ...>
10347 <... rt_sigprocmask resumed>NULL, 8) = 0
10347 write(1, "Thread started. Active threads: 6\n", 34) = 34
10347 mmap(NULL, 134217728, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x78d6c4000000
10347 munmap(0x78d6c8000000, 67108864) = 0
10347 mprotect(0x78d6c4000000, 135168, PROT_READ|PROT_WRITE) = 0
10347 mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0)
= 0x78d6d67fd000
10347 mprotect(0x78d6d67fe000, 8388608, PROT_READ|PROT_WRITE) = 0
10347 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0
10347 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY
=>{parent_tid=[10348]}, 88) = 10348
10348 rseq(0x78d6d6fffdfe0, 0x20, 0, 0x53053053 <unfinished ...>
10347 rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>
10348 <... rseq resumed>) = 0
10347 <... rt_sigprocmask resumed>NULL, 8) = 0
10348 set_robust_list(0x78d6d6ffd9a0, 24 <unfinished ...>
```

```
10347 futex(0x78d6d6ffd990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,10348,NULL,FUTEX_BI
<unfinished ...>
10348 <... set_robust_list resumed>      = 0
10348 rt_sigprocmask(SIG_SETMASK,[] ,NULL,8) = 0
10348 write(1,"Thread started. Active threads: 7\n",34) = 34
10348 mmap(0x78d6c8000000,67108864,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x78d6c8000000
10348 mprotect(0x78d6c8000000,135168,PROT_READ|PROT_WRITE) = 0
10348 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0)
= 0x78d6d5ffc000
10348 mprotect(0x78d6d5ffd000,8388608,PROT_READ|PROT_WRITE) = 0
10348 rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
10348 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY
<unfinished ...>
10349 rseq(0x78d6d67fcfe0,0x20,0,0x53053053 <unfinished ...>
10348 <... clone3 resumed>=>{parent_tid=[10349]},88) = 10349
10349 <... rseq resumed>                  = 0
10348 rt_sigprocmask(SIG_SETMASK,[] , <unfinished ...>
10349 set_robust_list(0x78d6d67fc9a0,24 <unfinished ...>
10348 <... rt_sigprocmask resumed>NULL,8) = 0
10349 <... set_robust_list resumed>       = 0
10348 futex(0x78d6d67fc990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,10349,NULL,FUTEX_BI
<unfinished ...>
10349 rt_sigprocmask(SIG_SETMASK,[] ,NULL,8) = 0
10349 write(1,"Thread started. Active threads: 8\n",34) = 34
10349 mmap(NULL,134217728,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x78d6bc000000
10349 munmap(0x78d6c0000000,67108864)   = 0
10349 mprotect(0x78d6bc000000,135168,PROT_READ|PROT_WRITE) = 0
10349 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0)
= 0x78d6d57fb000
10349 mprotect(0x78d6d57fc000,8388608,PROT_READ|PROT_WRITE) = 0
10349 rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
10349 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY
<unfinished ...>
10350 rseq(0x78d6d5ffbfe0,0x20,0,0x53053053 <unfinished ...>
10349 <... clone3 resumed>=>{parent_tid=[10350]},88) = 10350
10350 <... rseq resumed>                  = 0
10349 rt_sigprocmask(SIG_SETMASK,[] , <unfinished ...>
10350 set_robust_list(0x78d6d5ffb9a0,24) = 0
10349 <... rt_sigprocmask resumed>NULL,8) = 0
10350 rt_sigprocmask(SIG_SETMASK,[] , <unfinished ...>
10349 futex(0x78d6d5ffb990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,10350,NULL,FUTEX_BI
<unfinished ...>
10350 <... rt_sigprocmask resumed>NULL,8) = 0
10350 write(1,"Thread started. Active threads: 9\n",34) = 34
10350 openat(AT_FDCWD,"/sys/devices/system/cpu/online",O_RDONLY|O_CLOEXEC)
= 3
10350 read(3,"0-19\n",1024)             = 5
```

```
10350 close(3) = 0
10350 mmap(0x78d6c0000000,67108864,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x78d6c0000000
10350 mprotect(0x78d6c0000000,135168,PROT_READ|PROT_WRITE) = 0
10350 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0)
= 0x78d6d4ffa000
10350 mprotect(0x78d6d4ffb000,8388608,PROT_READ|PROT_WRITE) = 0
10350 rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
10350 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY
<unfinished ...>
10351 rseq(0x78d6d57faf0,0x20,0,0x53053053 <unfinished ...>
10350 <... clone3 resumed>=>{parent_tid=[10351]},88) = 10351
10351 <... rseq resumed> = 0
10350 rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
10351 set_robust_list(0x78d6d57fa9a0,24 <unfinished ...>
10350 <... rt_sigprocmask resumed>NULL,8) = 0
10351 <... set_robust_list resumed> = 0
10350 futex(0x78d6d57fa990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,10351,NULL,FUTEX_BI
<unfinished ...>
10351 rt_sigprocmask(SIG_SETMASK,[],NULL,8) = 0
10351 write(1,"Thread started. Active threads: 10\n",35) = 35
10351 mmap(NULL,134217728,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x78d6b4000000
10351 munmap(0x78d6b8000000,67108864) = 0
10351 mprotect(0x78d6b4000000,135168,PROT_READ|PROT_WRITE) = 0
10351 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0)
= 0x78d6d47f9000
10351 mprotect(0x78d6d47fa000,8388608,PROT_READ|PROT_WRITE) = 0
10351 rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
10351 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY
=>{parent_tid=[10352]},88) = 10352
10352 rseq(0x78d6d4ff9fe0,0x20,0,0x53053053 <unfinished ...>
10351 rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
10352 <... rseq resumed> = 0
10351 <... rt_sigprocmask resumed>NULL,8) = 0
10352 set_robust_list(0x78d6d4ff99a0,24 <unfinished ...>
10351 futex(0x78d6d4ff9990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,10352,NULL,FUTEX_BI
<unfinished ...>
10352 <... set_robust_list resumed> = 0
10352 rt_sigprocmask(SIG_SETMASK,[],NULL,8) = 0
10352 write(1,"Thread started. Active threads: 11\n",35) = 35
10352 mmap(0x78d6b8000000,67108864,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x78d6b8000000
10352 mprotect(0x78d6b8000000,135168,PROT_READ|PROT_WRITE) = 0
10352 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0)
= 0x78d6b37ff000
10352 mprotect(0x78d6b3800000,8388608,PROT_READ|PROT_WRITE) = 0
10352 rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
10352 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY
```

```
=>{parent_tid=[10353],88) = 10353
10353 rseq(0x78d6b3ffffe0,0x20,0,0x53053053 <unfinished ...>
10352 rt_sigprocmask(SIG_SETMASK,[] , <unfinished ...>
10353 <... rseq resumed>) = 0
10352 <... rt_sigprocmask resumed>NULL,8) = 0
10353 set_robust_list(0x78d6b3ffff9a0,24 <unfinished ...>
10352 futex(0x78d6b3fff990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,10353,NULL,FUTEX_BI
<unfinished ...>
10353 <... set_robust_list resumed>) = 0
10353 rt_sigprocmask(SIG_SETMASK,[] ,NULL,8) = 0
10353 write(1,"Thread started. Active threads: 12\n",35) = 35
10353 mmap(NULL,134217728,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x78d6ab600000
10353 munmap(0x78d6ab600000,10485760) = 0
10353 munmap(0x78d6b0000000,56623104) = 0
10353 mprotect(0x78d6ac000000,135168,PROT_READ|PROT_WRITE) = 0
10353 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0)
= 0x78d6b2ffe000
10353 mprotect(0x78d6b2ffff000,8388608,PROT_READ|PROT_WRITE) = 0
10353 rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
10353 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY
=>{parent_tid=[10354],88) = 10354
10354 rseq(0x78d6b37fe0,0x20,0,0x53053053 <unfinished ...>
10353 rt_sigprocmask(SIG_SETMASK,[] , <unfinished ...>
10354 <... rseq resumed>) = 0
10353 <... rt_sigprocmask resumed>NULL,8) = 0
10354 set_robust_list(0x78d6b37fe9a0,24 <unfinished ...>
10353 futex(0x78d6b37fe990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,10354,NULL,FUTEX_BI
<unfinished ...>
10354 <... set_robust_list resumed>) = 0
10354 rt_sigprocmask(SIG_SETMASK,[] ,NULL,8) = 0
10354 write(1,"Thread started. Active threads: 13\n",35) = 35
10354 mmap(NULL,134217728,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x78d6a4000000
10354 munmap(0x78d6a8000000,67108864) = 0
10354 mprotect(0x78d6a4000000,135168,PROT_READ|PROT_WRITE) = 0
10354 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0)
= 0x78d6b27fd000
10354 mprotect(0x78d6b27fe000,8388608,PROT_READ|PROT_WRITE) = 0
10354 rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
10354 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY
<unfinished ...>
10355 rseq(0x78d6b2ffdf0,0x20,0,0x53053053 <unfinished ...>
10354 <... clone3 resumed>=>{parent_tid=[10355],88) = 10355
10355 <... rseq resumed>) = 0
10354 rt_sigprocmask(SIG_SETMASK,[] , <unfinished ...>
10355 set_robust_list(0x78d6b2ffd9a0,24 <unfinished ...>
10354 <... rt_sigprocmask resumed>NULL,8) = 0
10355 <... set_robust_list resumed>) = 0
10354 futex(0x78d6b2ffd990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,10355,NULL,FUTEX_BI
```

```
<unfinished ...>
10355 rt_sigprocmask(SIG_SETMASK, [],NULL,8) = 0
10355 write(1,"Thread started. Active threads: 14\n",35) = 35
10355 mmap(0x78d6a8000000,67108864,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x78d6a8000000
10355 mprotect(0x78d6a8000000,135168,PROT_READ|PROT_WRITE) = 0
10355 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0)
= 0x78d6b1fffc000
10355 mprotect(0x78d6b1ffd000,8388608,PROT_READ|PROT_WRITE) = 0
10355 rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
10355 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY
<unfinished ...>
10356 rseq(0x78d6b27fcfe0,0x20,0,0x53053053 <unfinished ...>
10355 <... clone3 resumed=>={parent_tid=[10356]},88) = 10356
10356 <... rseq resumed> = 0
10355 rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>
10356 set_robust_list(0x78d6b27fc9a0,24 <unfinished ...>
10355 <... rt_sigprocmask resumed>NULL,8) = 0
10356 <... set_robust_list resumed> = 0
10355 futex(0x78d6b27fc990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,10356,NULL,FUTEX_BI
<unfinished ...>
10356 rt_sigprocmask(SIG_SETMASK, [],NULL,8) = 0
10356 write(1,"Thread started. Active threads: 15\n",35) = 35
10356 mmap(NULL,134217728,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x78d69c000000
10356 munmap(0x78d6a0000000,67108864) = 0
10356 mprotect(0x78d69c000000,135168,PROT_READ|PROT_WRITE) = 0
10356 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0)
= 0x78d6b17fb000
10356 mprotect(0x78d6b17fc000,8388608,PROT_READ|PROT_WRITE) = 0
10356 rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
10356 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SY
=>{parent_tid=[10357]},88) = 10357
10357 rseq(0x78d6b1ffbfe0,0x20,0,0x53053053 <unfinished ...>
10356 rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>
10357 <... rseq resumed> = 0
10356 <... rt_sigprocmask resumed>NULL,8) = 0
10357 set_robust_list(0x78d6b1ffb9a0,24 <unfinished ...>
10356 futex(0x78d6b1ffb990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,10357,NULL,FUTEX_BI
<unfinished ...>
10357 <... set_robust_list resumed> = 0
10357 rt_sigprocmask(SIG_SETMASK, [],NULL,8) = 0
10357 write(1,"Thread started. Active threads: 16\n",35) = 35
10357 write(1,"Thread finished. Active threads: 16\n",36) = 36
10357 mmap(0x78d6a0000000,67108864,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x78d6a0000000
10357 mprotect(0x78d6a0000000,135168,PROT_READ|PROT_WRITE) = 0
10357 rt_sigprocmask(SIG_BLOCK,~[RT_1],NULL,8) = 0
10357 madvise(0x78d6b17fb000,8368128,MADV_DONTNEED) = 0
```

```
10357 exit(0) = ?
10357 +++
10356 <... futex resumed> = 0
10356 write(1,"Thread finished. Active threads: 15\n",36) = 36
10356 rt_sigprocmask(SIG_BLOCK,~[RT_1],NULL,8) = 0
10356 madvise(0x78d6b1ffc000,8368128,MADV_DONTNEED) = 0
10356 exit(0) = ?
10356 +++
10355 <... futex resumed> = 0
10355 write(1,"Thread finished. Active threads: 14\n",36) = 36
10355 rt_sigprocmask(SIG_BLOCK,~[RT_1],NULL,8) = 0
10355 madvise(0x78d6b27fd000,8368128,MADV_DONTNEED) = 0
10355 exit(0) = ?
10355 +++
10354 <... futex resumed> = 0
10354 write(1,"Thread finished. Active threads: 13\n",36) = 36
10354 rt_sigprocmask(SIG_BLOCK,~[RT_1],NULL,8) = 0
10354 madvise(0x78d6b2ffe000,8368128,MADV_DONTNEED) = 0
10354 exit(0) = ?
10354 +++
10353 <... futex resumed> = 0
10353 write(1,"Thread finished. Active threads: 12\n",36) = 36
10353 rt_sigprocmask(SIG_BLOCK,~[RT_1],NULL,8) = 0
10353 madvise(0x78d6b37ff000,8368128,MADV_DONTNEED) = 0
10353 exit(0) = ?
10353 +++
10352 <... futex resumed> = 0
10352 munmap(0x78d6b17fb000,8392704) = 0
10352 write(1,"Thread finished. Active threads: 11\n",36) = 36
10352 rt_sigprocmask(SIG_BLOCK,~[RT_1],NULL,8) = 0
10352 madvise(0x78d6d47f9000,8368128,MADV_DONTNEED) = 0
10352 exit(0) = ?
10352 +++
10351 <... futex resumed> = 0
10351 munmap(0x78d6b1ffc000,8392704) = 0
10351 write(1,"Thread finished. Active threads: 10\n",36) = 36
10351 rt_sigprocmask(SIG_BLOCK,~[RT_1],NULL,8) = 0
10351 madvise(0x78d6d4ffa000,8368128,MADV_DONTNEED) = 0
10351 exit(0) = ?
10351 +++
10350 <... futex resumed> = 0
10350 munmap(0x78d6b27fd000,8392704) = 0
10350 write(1,"Thread finished. Active threads: 9\n",35) = 35
10350 rt_sigprocmask(SIG_BLOCK,~[RT_1],NULL,8) = 0
10350 madvise(0x78d6d57fb000,8368128,MADV_DONTNEED) = 0
10350 exit(0) = ?
10350 +++
10349 <... futex resumed> = 0
```



```
10342 <... futex resumed>          = 0
10342 munmap(0x78d6d6ffe000,8392704) = 0
10342 write(1,"Thread finished. Active threads: 1\n",35) = 35
10342 rt_sigprocmask(SIG_BLOCK,[RT_1],NULL,8) = 0
10342 madvise(0x78d6e59ff000,8368128,MADV_DONTNEED) = 0
10342 exit(0)                      = ?
10341 <... futex resumed>          = 0
10342 +++ exited with 0 +++
10341 munmap(0x78d6d77ff000,8392704) = 0
10341 write(1,"After sorting:\n",15) = 15
10341 write(1,"0 0 4 9 14 22 23 24 25 26 \n",27) = 27
10341 write(1,"Time: 0.0184601 seconds\n",24) = 24
10341 exit_group(0)                = ?
10341 +++ exited with 0 +---
```