**Федеральное государственное автономное образовательное учреждение**

**Высшего профессионального образования**

Санкт-Петербургский политехнический университет

Институт информационных технологий и управления

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №3**

«Исключения в Linux»

Работу выполнил студент гр. № 53501/3 Цыганов А.А.

Работу принял преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Душутина Е.В.

Санкт-Петербург

2016

# Цель работы:

Для исключений деления на ноль и переполнение вещественного числа выполнить:

* Создать обработчик сигнала исключения
* Поймать исключения с помощью конструкции try-catch.
* Проанализировать работу программ с помощью средств профилирования

# Ход работы:

Для работы была использована виртуальная машина VMware Workstation 10.0.4, на которой была запущена ОС Linux Ubuntu 64-bit.В ходе работыбудут рассмотрены обработки исключений в Linux: с помощью механизма сигналов и с помощью блоков Try/Catch.

# Обработка исключения в Linux через сигналы

При возникновении исключений в ОС Linux система отправляет сигнал, соответствующий возникшему исключению. Таким образом, написав обработчик для необходимого сигнала, возможно имитировать обработку исключений, аналогично механизму SEH.

Далее рассмотрим программу, генерирующую исключение деления на ноль и исключение переполнения вещественного числа. На эти ошибки будут сгенерированы исключения и данные сохранятся в системный журнал.

#include <stdio.h>

#include <stdlib.h>

#include <signal.h>

int main()

{

int b = 0;

int result = 1 / b;

return 0;

}

После сборки и выполнения программы получены следующие данные:

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 3$ gcc -Wall app.cpp

app.cpp: In function ‘int main()’:

app.cpp:9:6: warning: unused variable ‘result’ [-Wunused-variable]

int result = 1 / b;

^

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 3$ ./a.out

Floating point exception (core dumped)

Сработало исключение и в системные логи добавилась информация об исключении.

cat /var/log/syslog

Feb 11 18:19:22 ubuntu kernel: [98486.220758] traps: a.out[18147] trap divide error ip:400507 sp:7ffe21d87df0 error:0 in a.out[400000+1000]

Далее выполним программу в утилите strace:

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 3$ strace ./a.out

execve("./a.out", ["./a.out"], [/\* 64 vars \*/]) = 0

brk(0) = 0xab5000

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f9fd0c90000

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

open("/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=86295, ...}) = 0

mmap(NULL, 86295, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f9fd0c7a000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

open("/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0`\v\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=1869392, ...}) = 0

mmap(NULL, 3972864, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f9fd06a5000

mprotect(0x7f9fd0865000, 2097152, PROT\_NONE) = 0

mmap(0x7f9fd0a65000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1c0000) = 0x7f9fd0a65000

mmap(0x7f9fd0a6b000, 16128, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f9fd0a6b000

close(3) = 0

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f9fd0c79000

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f9fd0c78000

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f9fd0c77000

arch\_prctl(ARCH\_SET\_FS, 0x7f9fd0c78700) = 0

mprotect(0x7f9fd0a65000, 16384, PROT\_READ) = 0

mprotect(0x600000, 4096, PROT\_READ) = 0

mprotect(0x7f9fd0c92000, 4096, PROT\_READ) = 0

munmap(0x7f9fd0c7a000, 86295) = 0

--- SIGFPE {si\_signo=SIGFPE, si\_code=FPE\_INTDIV, si\_addr=0x400507} ---

+++ killed by SIGFPE (core dumped) +++

Floating point exception (core dumped)

С помощью утилиты gdb можно получить дизассемблированный код и провести отладку программы:

(gdb) disassemble /m main

Dump of assembler code for function main():

6 {

0x00000000004004f6 <+0>: push %rbp

0x00000000004004f7 <+1>: mov %rsp,%rbp

7 int b = 0;

0x00000000004004fa <+4>: movl $0x0,-0x8(%rbp)

8 int result = 1 / b;

0x0000000000400501 <+11>: mov $0x1,%eax

0x0000000000400506 <+16>: cltd

=> 0x0000000000400507 <+17>: idivl -0x8(%rbp)

0x000000000040050a <+20>: mov %eax,-0x4(%rbp)

9 return 0;

0x000000000040050d <+23>: mov $0x0,%eax

10 }

0x0000000000400512 <+28>: pop %rbp

0x0000000000400513 <+29>: retq

End of assembler dump.

(gdb) run

Starting program: /home/alex/Dropbox/политех/Системное программирование/Л. р. № 3/a.out

Program received signal SIGFPE, Arithmetic exception.

0x0000000000400507 in main () at app.cpp:8

8 int result = 1 / b;

Strace вывело в консоль множество системных вызовов. Интересующий нас вызов - SIGFPE сообщает о том, что процесс получил сигнал об ошибки в делении. Код FPE\_INTDIV указывает на то, что причиной исключения является некорректное целочисленное деление. Поле si\_addr указывает на адрес инструкции, выполнение которой повлекло за собой исключение. Так как в программе не был определен пользовательский обработчик исключения, был использован системный обработчик, который принудительно завершает (Kill) процесс.

Далее проведем обработку исключения обращение к пустой области памяти, с реализацией обработчика сигналов. Ниже приведен код программы с генерацией исключения на обращение к пустой области памяти:

int main()

{

float\* d = 0;

d[10] = 1;

return 0;

}

После сборки и выполнения программы получены следующие данные:

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 3$ gcc -Wall app.cpp -lm

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 3$ ./a.out

Segmentation fault (core dumped)

В файле системных логов добавилась информация об полученном исключении:

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 3$ cat /var/log/syslog

Feb 11 19:09:12 ubuntu kernel: [101483.045552] a.out[18468]: segfault at 28 ip 0000000000400512 sp 00007ffe3ed98470 error 6 in a.out[400000+1000]

С помощью утилиты strace исследуем работы программы:

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 3$ strace ./a.out

execve("./a.out", ["./a.out"], [/\* 64 vars \*/]) = 0

brk(0) = 0xf1e000

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f82cda5d000

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

open("/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=86295, ...}) = 0

mmap(NULL, 86295, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f82cda47000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

open("/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0`\v\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=1869392, ...}) = 0

mmap(NULL, 3972864, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f82cd472000

mprotect(0x7f82cd632000, 2097152, PROT\_NONE) = 0

mmap(0x7f82cd832000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1c0000) = 0x7f82cd832000

mmap(0x7f82cd838000, 16128, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f82cd838000

close(3) = 0

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f82cda46000

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f82cda45000

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f82cda44000

arch\_prctl(ARCH\_SET\_FS, 0x7f82cda45700) = 0

mprotect(0x7f82cd832000, 16384, PROT\_READ) = 0

mprotect(0x600000, 4096, PROT\_READ) = 0

mprotect(0x7f82cda5f000, 4096, PROT\_READ) = 0

munmap(0x7f82cda47000, 86295) = 0

**--- SIGSEGV {si\_signo=SIGSEGV, si\_code=SEGV\_MAPERR, si\_addr=0x28} ---**

**+++ killed by SIGSEGV (core dumped) +++**

Segmentation fault (core dumped)

C помощью утилиты gdb можно получить дизассемблированный код:

(gdb) disassemble /m main

Dump of assembler code for function main():

8 {

0x00000000004004f6 <+0>: push %rbp

0x00000000004004f7 <+1>: mov %rsp,%rbp

9 float\* d = 0;

0x00000000004004fa <+4>: movq $0x0,-0x8(%rbp)

10 d[10] = 1;

0x0000000000400502 <+12>: mov -0x8(%rbp),%rax

0x0000000000400506 <+16>: add $0x28,%rax

0x000000000040050a <+20>: movss 0x92(%rip),%xmm0 # 0x4005a4

0x0000000000400512 <+28>: movss %xmm0,(%rax)

11

12 return 0;

0x0000000000400516 <+32>: mov $0x0,%eax

13 }

0x000000000040051b <+37>: pop %rbp

0x000000000040051c <+38>: retq

End of assembler dump.

(gdb) run

Starting program: /home/alex/Dropbox/политех/Системное программирование/Л. р. № 3/a.out

Program received signal SIGSEGV, Segmentation fault.

0x0000000000400512 in main () at app.cpp:10

10 d[10] = 1;

Ошибка в программе связана с генерацией сигнала SIGSEGV. Далее добавим в программу обработчик этого сигнала. Ниже приведен текст программы.

#include <signal.h>

#include <stdio.h>

#include <stdlib.h>

void mySignalHandler(int signum)

{

if (signum == SIGSEGV)

printf("Check signal: SIGFPE\n");

exit(0);

}

int main()

{

signal(SIGSEGV, &mySignalHandler);

float\* d = 0;

d[10] = 1;

return 0;

}

После сборки и выполнения программы получены следующие результаты:

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 3$ gcc -Wall -g app.cpp

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 3$ ./a.out

Check signal: SIGFPE

В консоль отобразилась информация о полученном исключении в программе. Ниже приведен анализ системных вызовов через утилиту strace.

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 3$ strace ./a.out

execve("./a.out", ["./a.out"], [/\* 64 vars \*/]) = 0

brk(0) = 0x11af000

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f1b191fd000

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

open("/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=86295, ...}) = 0

mmap(NULL, 86295, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f1b191e7000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

open("/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0`\v\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=1869392, ...}) = 0

mmap(NULL, 3972864, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f1b18c12000

mprotect(0x7f1b18dd2000, 2097152, PROT\_NONE) = 0

mmap(0x7f1b18fd2000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1c0000) = 0x7f1b18fd2000

mmap(0x7f1b18fd8000, 16128, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f1b18fd8000

close(3) = 0

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f1b191e6000

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f1b191e5000

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f1b191e4000

arch\_prctl(ARCH\_SET\_FS, 0x7f1b191e5700) = 0

mprotect(0x7f1b18fd2000, 16384, PROT\_READ) = 0

mprotect(0x600000, 4096, PROT\_READ) = 0

mprotect(0x7f1b191ff000, 4096, PROT\_READ) = 0

munmap(0x7f1b191e7000, 86295) = 0

**rt\_sigaction(SIGSEGV, {0x4005c6, [SEGV], SA\_RESTORER|SA\_RESTART, 0x7f1b18c472f0}, {SIG\_DFL, [], 0}, 8) = 0**

**--- SIGSEGV {si\_signo=SIGSEGV, si\_code=SEGV\_MAPERR, si\_addr=0x28} ---**

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 19), ...}) = 0

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f1b191fc000

write(1, "Check signal: SIGFPE\n", 21Check signal: SIGFPE

) = 21

exit\_group(0) = ?

+++ exited with 0 +++

Жирным отмечен вызов для установки обработчика сигнала и момент получения сигнала процессом. Таким образом можно отслеживать любые исключение внутри текста программы.

# Использование блоков try/catch для перехвата исключений в Linux

# Сгенерируем и перехватим исключения С++ в ОС Linux, используя операторы trow,try,catch. Блок catch перехватывает исключение, полученное в безопасным блоке try. Для генерации исключений можно использовать trow. Ниже приведен код обработки исключения.

#include <stdio.h>

#include <stdlib.h>

#include <signal.h>

const int DEVIDE\_BY\_ZERO = 1;

int main()

{

try {

throw DEVIDE\_BY\_ZERO;

printf("Doesn't print\n");

}

catch(int e) {

if (e == DEVIDE\_BY\_ZERO)

printf("Catch exception: Devide by zero\n");

else

printf("Catch exception: %d\n", e);

}

return 0;

}

После сборки и выполнения программы получены следующие результаты:

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 3$ ./a.out

Catch exception: Devide by zero

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 3$

В консоль отобразилась информация о полученном исключении в программе. Ниже приведен анализ системных вызовов через утилиту strace.

alex@ubuntu:~/Dropbox/политех/Системное программирование/Л. р. № 3$ strace ./a.out

execve("./a.out", ["./a.out"], [/\* 64 vars \*/]) = 0

brk(0) = 0x1d05000

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7facff8e9000

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

open("/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=86295, ...}) = 0

mmap(NULL, 86295, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7facff8d3000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

open("/usr/lib/x86\_64-linux-gnu/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 \240\10\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=1566568, ...}) = 0

mmap(NULL, 3675328, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7facff346000

mprotect(0x7facff4b9000, 2093056, PROT\_NONE) = 0

mmap(0x7facff6b8000, 49152, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x172000) = 0x7facff6b8000

mmap(0x7facff6c4000, 13504, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7facff6c4000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

open("/lib/x86\_64-linux-gnu/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360\*\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=92504, ...}) = 0

mmap(NULL, 2188352, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7facff12f000

mprotect(0x7facff145000, 2093056, PROT\_NONE) = 0

mmap(0x7facff344000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x15000) = 0x7facff344000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

open("/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0`\v\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=1869392, ...}) = 0

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7facff8d2000

mmap(NULL, 3972864, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7facfed65000

mprotect(0x7facfef25000, 2097152, PROT\_NONE) = 0

mmap(0x7facff125000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1c0000) = 0x7facff125000

mmap(0x7facff12b000, 16128, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7facff12b000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

open("/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\240U\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=1084840, ...}) = 0

mmap(NULL, 3174760, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7facfea5d000

mprotect(0x7facfeb64000, 2093056, PROT\_NONE) = 0

mmap(0x7facfed63000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x106000) = 0x7facfed63000

close(3) = 0

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7facff8d1000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7facff8cf000

arch\_prctl(ARCH\_SET\_FS, 0x7facff8cf740) = 0

mprotect(0x7facff125000, 16384, PROT\_READ) = 0

mprotect(0x7facfed63000, 4096, PROT\_READ) = 0

mprotect(0x7facff344000, 4096, PROT\_READ) = 0

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7facff8ce000

mprotect(0x7facff6b8000, 40960, PROT\_READ) = 0

mprotect(0x600000, 4096, PROT\_READ) = 0

mprotect(0x7facff8eb000, 4096, PROT\_READ) = 0

munmap(0x7facff8d3000, 86295) = 0

brk(0) = 0x1d05000

brk(0x1d37000) = 0x1d37000

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 19), ...}) = 0

mmap(NULL, 4096, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7facff8e8000

write(1, "Catch exception: Devide by zero\n", 32Catch exception: Devide by zero

) = 32

exit\_group(0) = ?

+++ exited with 0 +++

С помощью утилиты gdb можно получить дизассемблированный код и провести отладку программы. В полученном коде видна инструкция которая осуществляет переход в блок catch при генерации исключения:

Dump of assembler code for function main():

8 {

0x00000000004008c6 <+0>: push %rbp

0x00000000004008c7 <+1>: mov %rsp,%rbp

0x00000000004008ca <+4>: push %rbx

0x00000000004008cb <+5>: sub $0x18,%rsp

9 try {

10 throw DEVIDE\_BY\_ZERO;

0x00000000004008cf <+9>: mov $0x4,%edi

0x00000000004008d4 <+14>: callq 0x400770 <\_\_cxa\_allocate\_exception@plt>

0x00000000004008d9 <+19>: movl $0x1,(%rax)

0x00000000004008df <+25>: mov $0x0,%edx

0x00000000004008e4 <+30>: mov $0x601080,%esi

0x00000000004008e9 <+35>: mov %rax,%rdi

0x00000000004008ec <+38>: callq 0x400780 <\_\_cxa\_throw@plt>

0x00000000004008f1 <+43>: cmp $0x1,%rdx

**0x00000000004008f5 <+47>: je 0x4008ff <main()+57>**

0x00000000004008f7 <+49>: mov %rax,%rdi

0x00000000004008fa <+52>: callq 0x4007c0 <\_Unwind\_Resume@plt>

11 printf("Doesn't print\n");

12 }

13 catch(int e) {

0x00000000004008ff <+57>: mov %rax,%rdi

0x0000000000400902 <+60>: callq 0x4007a0 <\_\_cxa\_begin\_catch@plt>

0x0000000000400907 <+65>: mov (%rax),%eax

0x0000000000400909 <+67>: mov %eax,-0x14(%rbp)

0x0000000000400932 <+108>: callq 0x400790 <\_\_cxa\_end\_catch@plt>

0x0000000000400941 <+123>: callq 0x400790 <\_\_cxa\_end\_catch@plt>

0x0000000000400946 <+128>: mov %rbx,%rax

0x0000000000400949 <+131>: mov %rax,%rdi

0x000000000040094c <+134>: callq 0x4007c0 <\_Unwind\_Resume@plt>

14 if (e == DEVIDE\_BY\_ZERO)

0x000000000040090c <+70>: cmpl $0x1,-0x14(%rbp)

0x0000000000400910 <+74>: jne 0x40091e <main()+88>

15 printf("Catch exception: Devide by zero\n");

0x0000000000400912 <+76>: mov $0x4009f0,%edi

0x0000000000400917 <+81>: callq 0x400750 <puts@plt>

0x000000000040091c <+86>: jmp 0x400932 <main()+108>

16 else

17 printf("Catch exception: %d\n", e);

0x000000000040091e <+88>: mov -0x14(%rbp),%eax

0x0000000000400921 <+91>: mov %eax,%esi

0x0000000000400923 <+93>: mov $0x400a10,%edi

0x0000000000400928 <+98>: mov $0x0,%eax

0x000000000040092d <+103>: callq 0x400730 <printf@plt>

18 }

19 return 0;

0x0000000000400937 <+113>: mov $0x0,%eax

0x000000000040093c <+118>: jmp 0x400951 <main()+139>

0x000000000040093e <+120>: mov %rax,%rbx

20 }

0x0000000000400951 <+139>: add $0x18,%rsp

0x0000000000400955 <+143>: pop %rbx

0x0000000000400956 <+144>: pop %rbp

0x0000000000400957 <+145>: retq

End of assembler dump.

**Выводы:**

В ходе лабораторной работы были изучены основные методы обработки исключений в Linux. Сгенерированное исключения обрабатывалось и сохранялось в системный журнал. Для перехвата таких исключений в коде программы можно реализовать функцию и подписаться на соответствующие типы исключений. Отладку программы можно делать в утилите gdb, которая позволяет дизассемблировать код. Механизм с использованием блоков try/catch так же успешно работает и перехватывает сгенерированные исключения через trow.