Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе №2

по курсу «Системное программирование»

по теме «Обработка исключений в Linux»

Выполнил студент гр. 13541/2:
Ерниязов Т. Е.

Проверил преподаватель:
Душутина Е. В.

Санкт-Петербург
2019 г.

1 Цель работы

Познакомится с видами исключений в операционной системе Linux и со способами их обработки.

2 Характеристики системы

```
lorismelik@lorismelik-Aspire-Z5700:~/$ cat /proc/version
Linux version 4.15.0-29-generic (buildd@lgw01-amd64-057) (gcc version 7.3.0 (Ubuntu 7.3.0-16ubuntu3))
    #31-Ubuntu SMP Tue Jul 17 15:39:52 UTC 2018

lorismelik@lorismelik-Aspire-Z5700:~/$ gcc --version
gcc (Ubuntu 7.3.0-27ubuntu1~18.04) 7.3.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

lorismelik@lorismelik-Aspire-Z5700:~/$ strace -V
strace -- version UNKNOWN
Copyright (c) 1991-2018 The strace developers <https://strace.io>.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Optional features enabled: stack-unwind stack-demangle m32-mpers mx32-mpers

lorismelik@lorismelik-Aspire-Z5700:~/$ ltrace -V
ltrace version 0.7.3.
Copyright (C) 1997-2009 Juan Cespedes <cespedes@debian.org>.
This is free software; see the GNU General Public Licence
version 2 or later for copying conditions.  There is NO warranty.
```

3 Ход работы

3.1 Обработка исключений посредством сигналов

В отличие от Windows, в UNIX системах не предусмотрено единого средства для обработки аппаратных и программных исключений, как структурированная обработка исключений (SEH). Для фиксации и обработки аппаратных исключений в Unix системах предусмотрено многоцелевое средство межпроцессного взаимодействия – сигналы. Сигнал представляет собой асинхронное уведомление процесса о возникновении некоторого события.

С точки зрения назначения сигналы в Linux делятся на несколько категорий: исключение, отладка, пользовательские, управление и сигналы POSIX реального времени. В рамках данной работы рассмотрим сигналы, порождаемые программными или аппаратными исключениями:

- SIGBUS – Код: 10. Неправильное обращение в физическую память, по умолчанию завершает процесс с дампом памяти.
- SIGFPE – Код: 8. Ошибочная арифметическая операция, по умолчанию завершает процесс с дампом памяти.
- SIGILL – Код: 4. Недопустимая инструкция процессора, по умолчанию завершает процесс с дампом памяти.
- SIGSEGV – Код: 11. Нарушение при обращении в память, по умолчанию завершает процесс с дампом памяти.
- SIGSYS – Код: 12. Неправильный системный вызов, по умолчанию завершает процесс с дампом памяти.
- SIGXCPU – Код: 30. Процесс превысил лимит процессорного времени, по умолчанию завершает процесс с дампом памяти.
- SIGXFSZ - Код: 31. Процесс превысил допустимый размер файла, по умолчанию завершает процесс с дампом памяти.

В данной работе рассматриваются сигналы SIGFPE (деление на ноль) и SIGSEGV (ошибочный доступ к памяти) как наиболее просто генерируемые.

3.1.1 Генерация сигналов

Рассмотрим программу, генерирующую исключения SIGFPE и SIGSEGV, но не имеющую обработчика:

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>

void generateArythmeticException() {
    const int one = 1;
    const int zero = 0;
    const int exception = 1 / zero;
}

void generateMemoryException() {
    int* address = 0x0;
    *address = 1;
}

int main(int argc, char *argv[]){
    if (argc != 2) {
        return 0x1;
    }

    if (strcmp(argv[1], "0") == 0)
        generateArythmeticException();
    else if (strcmp(argv[1], "1") == 0)
        generateMemoryException();

    return 0x0;
}
```

В зависимости от аргумента командной строки, программа генерирует исключение типа SIGFPE или SIGSEGV.

```
lorismelik@lorismelik-Aspire-Z5700:~/$ gcc task1.c -o task1.o
lorismelik@lorismelik-Aspire-Z5700:~/$ ./task1.o 0
Floating point exception (core dumped)
lorismelik@lorismelik-Aspire-Z5700:~/$ ./task1.o 1
Segmentation fault (core dumped)
```

Программы завершились аварийно. Характер завершения различен в зависимости от аргумента командной строки, что можно заметить по выводу.

В ситуациях, когда сигнал-исключение не был обработан программой и она завершила свою работу в аварийном режиме, ядро системы автоматически вносит соответствующие записи об ошибках в системных лог. Рассмотрим подробности завершения программы с помощью утилиты strace, а также проверим системный журнал syslog на наличие информации об исключениях:

```
lorismelik@lorismelik-Aspire-Z5700:~/$ strace ./task.o 0
execve("./task.o", ["./task.o", "0"], 0x7ffdfd432048 /* 57 vars */) = 0
brk(NULL)                               = 0x55eb6df30000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=88178, ...}) = 0
mmap(NULL, 88178, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f95bc6a0000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/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\260\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f95bc69e000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f95bc09e000
mprotect(0x7f95bc285000, 2097152, PROT_NONE) = 0
mmap(0x7f95bc485000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0
    x7f95bc485000
mmap(0x7f95bc48b000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
    x7f95bc48b000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f95bc69f4c0) = 0
mprotect(0x7f95bc485000, 16384, PROT_READ) = 0
mprotect(0x55eb6c70e000, 4096, PROT_READ) = 0
mprotect(0x7f95bc6b6000, 4096, PROT_READ) = 0
munmap(0x7f95bc6a0000, 88178)           = 0
--- SIGFPE {si_signo=SIGFPE, si_code=FPE_INTDIV, si_addr=0x55eb6c50e662} ---
+++ killed by SIGFPE (core dumped) +++
Floating point exception (core dumped)



lorismelik@lorismelik-Aspire-Z5700:~/$ strace ./task.o 1
execve("./task.o", ["./task.o", "1"], 0x7fff136f1248 /* 57 vars */) = 0
brk(NULL)                               = 0x5629efffa000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
```

```
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=88178, ...}) = 0
mmap(NULL, 88178, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f432a474000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/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\260\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f432a472000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f4329e72000
mprotect(0x7f432a059000, 2097152, PROT_NONE) = 0
mmap(0x7f432a259000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0
    x7f432a259000
mmap(0x7f432a25f000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
    x7f432a25f000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f432a4734c0) = 0
mprotect(0x7f432a259000, 16384, PROT_READ) = 0
mprotect(0x5629ee7a3000, 4096, PROT_READ) = 0
mprotect(0x7f432a48a000, 4096, PROT_READ) = 0
munmap(0x7f432a474000, 88178)           = 0
--- SIGSEGV {si_signo=SIGSEGV, si_code=SEGV_MAPERR, si_addr=NULL} ---
+++ killed by SIGSEGV (core dumped) +++
Segmentation fault (core dumped)


lorismelik@lorismelik-Aspire-Z5700:~/$ cat /var/log/syslog | grep task
Jan 26 14:46:12 lorismelik-Aspire-Z5700 kernel: [ 1391.059850] task.o[16734]: segfault at 0 ip 0000561102
    d8167b sp 00007fff529b1b20 error 6 in task.o[561102d81000+1000]
Jan 26 14:46:15 lorismelik-Aspire-Z5700 kernel: [ 1394.684221] traps: task.o[16737] trap divide error ip
    :5599f8e6a662 sp:7ffd43296c60 error:0 in task.o[5599f8e6a000+1000]
```

Вывод утилиты strace подтверждает завершение процесса с помощью необработанных исключений SIGFPE и SIGSEGV.

3.1.2 Обработка сигналов функцией signal

Для назначения обработчика ненадежных сигналов в Unix-подобных системах используется функция signal со следующей сигнатурой:

```
sighandler_t signal(int signum, sighandler_t handler);
```

Прототип имеет два аргумента, первый номер сигнала и второй указатель на функцию обработчика сигналов. Функция обработчика сигналов возвращает void и принимает единственный целочисленный аргумент, который представляет номер сигнала, который был отправлен. Таким образом, имеется возможность использовать одну и ту же функцию обработчика сигнала для нескольких разных сигналов.

Для записи в системный журнал используется 3 функции:

```
void openlog(const char *ident, int option, int facility);
void syslog(int priority, const char *format, ...);
void closelog(void);
```

- openlog() устанавливает связь с программой, ведущей системный журнал. Строка ident добавляется к каждому сообщению и обычно представляет собой название программы. Аргумент option указывает флаг управляющий работой openlog() и соответствующих вызовов syslog().
- syslog() создает сообщение для журнала, которое передается syslogd.
- closelog() закрывает описатель, используемый для записи данных в журнал. Использование closelog() необязательно.

Рассмотрим программу, которая при возникновении исключения вызывает собственный обработчик, в котором составляет сообщение для системного журнала и завершает процесс:

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <syslog.h>

void generateArythmeticException() {
    const int one = 1;
    const int zero = 0;
    const int exception = one / zero;
}

void generateMemoryException() {
    int* address = 0x0;
```

```
    *address = 1;
}

void signalArythmeticHandler(const int signalCode) {
    const char* message = "Arythmetic exception occured! Code: 0x";
    printf("%s%x\n", message, signalCode);

    openlog("task2", 0, LOG_USER);
    syslog(LOG_NOTICE, "%s%x\n", message, signalCode);
    closelog();

    exit(EXIT_FAILURE);
}

void signalMemoryHandler(const int signalCode) {
    const char* message = "Memory exception occured! Code: 0x";
    printf("%s%x\n", message, signalCode);

    openlog("task2", 0, LOG_USER);
    syslog(LOG_NOTICE, "%s%x\n", message, signalCode);
    closelog();

    exit(EXIT_FAILURE);
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        return 0x1;
    }

    if (signal(SIGFPE, signalArythmeticHandler) == SIG_ERR)
        return 0x2;

    if (signal(SIGSEGV, signalMemoryHandler) == SIG_ERR)
        return 0x3;

    if (strcmp(argv[1], "0") == 0)
        generateArythmeticException();
    else if (strcmp(argv[1], "1") == 0)
        generateMemoryException();

    return 0x0;
}
```

```
lorismelik@lorismelik-Aspire-Z5700:~/$ gcc task2.c -o task2.o
lorismelik@lorismelik-Aspire-Z5700:~/$ ./task2.o 0
Arythmetic exception occured! Code: 0x8
lorismelik@lorismelik-Aspire-Z5700:~/$ ./task2.o 1
Memory exception occured! Code: 0xb
```

Вывод программы свидетельствует о том, что исключения действительно были обработаны. Рассмотрим поведение программы с точки зрения системы:

```
lorismelik@lorismelik-Aspire-Z5700:~/$ strace ./task2.o 0
execve("./task2.o", ["./task2.o", "0"], 0x7ffcb2842f08 /* 57 vars */) = 0
brk(NULL)                               = 0x5576c7123000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=88178, ...}) = 0
mmap(NULL, 88178, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f6e83be4000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/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\260\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f6e83be2000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f6e835e2000
mprotect(0x7f6e837c9000, 2097152, PROT_NONE) = 0
mmap(0x7f6e839c9000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0
    x7f6e839c9000
mmap(0x7f6e839cf000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
    x7f6e839cf000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f6e83be34c0) = 0
mprotect(0x7f6e839c9000, 16384, PROT_READ) = 0
mprotect(0x5576c6099000, 4096, PROT_READ) = 0
mprotect(0x7f6e83bfa000, 4096, PROT_READ) = 0
munmap(0x7f6e83be4000, 88178)           = 0
rt_sigaction(SIGFPE, {sa_handler=0x5576c5e99832, sa_mask=[FPE], sa_flags=SA_RESTORER|SA_RESTART,
    sa_restorer=0x7f6e83620f20}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0
rt_sigaction(SIGSEGV, {sa_handler=0x5576c5e998aa, sa_mask=[SEGV], sa_flags=SA_RESTORER|SA_RESTART,
    sa_restorer=0x7f6e83620f20}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0
--- SIGFPE {si_signo=SIGFPE, si_code=FPE_INTDIV, si_addr=0x5576c5e99810} ---
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
brk(NULL)                               = 0x5576c7123000
brk(0x5576c7144000)                     = 0x5576c7144000
write(1, "Arythmetic exception occured! Co"..., 40Arythmetic exception occured! Code: 0x8
) = 40
```

```
openat(AT_FDCWD, "/etc/localtime", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
fstat(3, {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
read(3, "TZif2\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\21\0\0\0\21\0\0\0\0"..., 4096) = 1544
lseek(3, -936, SEEK_CUR)                = 608
read(3, "TZif2\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\21\0\0\0\21\0\0\0\0"..., 4096) = 936
close(3)                                = 0
socket(AF_UNIX, SOCK_DGRAM|SOCK_CLOEXEC, 0) = 3
connect(3, {sa_family=AF_UNIX, sun_path="/dev/log"}, 110) = 0
sendto(3, "<13>Jan 26 15:02:43 task2: Aryth"..., 67, MSG_NOSIGNAL, NULL, 0) = 67
close(3)                                = 0
exit_group(1)                           = ?
+++ exited with 1 +++

lorismelik@lorismelik-Aspire-Z5700:~/$ cat /var/log/syslog | grep task2
Jan 26 15:02:43 lorismelik-Aspire-Z5700 task2: Arythmetic exception occured! Code: 0x8
Jan 26 15:33:53 lorismelik-Aspire-Z5700 task2: Memory exception occured! Code: 0xb
```

На выводе утилиты strace виден момент установки обработчиков сигналов, момент возникновения исключения, а также процесс его обработки.

Записи в журнале syslog, которые были добавлены непосредственно внутри собственных обработчиков, действительно были добавлены.

3.1.3 Генерация сигналов функцией raise

Исключение можно генерировать с помощью функции raise, имеющей следующую сигнатуру:

```
int raise(int sig);
```

Функция raise принимает единственный аргумент, являющийся кодом сигнала. Стоит отметить, что raise эквивалентна функции kill(getpid(), sig). Обработку сигналов можно осуществить с помощью функции sigaction, имеющей следующую сигнатуру:

```
int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);
```

Системный вызов sigaction используется для изменения действий процесса при получении соответствующего сигнала. Sigaction намного более опциональна, чем signal, а также позволяет задавать обработчики надежных сигналов. Лучше избегать использования функции signal и использовать вместо нее sigaction

Структура sigaction имеет следующий формат:

```
struct sigaction {
    void (*sa_handler)(int);
    void (*sa_sigaction)(int, siginfo_t *, void *);
    sigset_t sa_mask;
    int sa_flags;
    void (*sa_restorer)(void);
}
```

Рассмотрим модифицированную программу, использующую функции raise и sigaction:

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <syslog.h>

void signalArythmeticHandler(const int signalCode) {
    const char* message = "Arythmetic exception occured! Code: 0x";
    printf("%s%x\n", message, signalCode);

    openlog("task3", 0, LOG_USER);
    syslog(LOG_NOTICE, "%s%x\n", message, signalCode);
    closelog();

    exit(EXIT_FAILURE);
}

void signalMemoryHandler(const int signalCode) {
    const char* message = "Memory exception occured! Code: 0x";
    printf("%s%x\n", message, signalCode);

    openlog("task3", 0, LOG_USER);
    syslog(LOG_NOTICE, "%s%x\n", message, signalCode);
    closelog();

    exit(EXIT_FAILURE);
}
```

```
29
30 int main(int argc, char *argv[]) {
31     if (argc != 2) {
32         return 0x1;
33     }
34
35     struct sigaction arythmeticAction, memoryAction;
36
37     arythmeticAction.sa_handler = signalArythmeticHandler;
38     sigemptyset(&arythmeticAction.sa_mask);
39     arythmeticAction.sa_flags;
40
41     memoryAction.sa_handler = signalMemoryHandler;
42     sigemptyset(&memoryAction.sa_mask);
43     memoryAction.sa_flags;
44
45     sigaction(SIGFPE, &arythmeticAction, NULL);
46     sigaction(SIGSEGV, &memoryAction, NULL);
47
48     if (strcmp(argv[1], "0") == 0)
49         raise(SIGFPE);
50     else if (strcmp(argv[1], "1") == 0)
51         raise(SIGSEGV);
52
53     return 0x0;
54 }
```

Результат работы программы

```
1 lorismelik@lorismelik-Aspire-Z5700:~/$ gcc task3.c -o task3
2 lorismelik@lorismelik-Aspire-Z5700:~/$ ./task3 0
3 Arythmetic exception occured! Code: 0x8
4 lorismelik@lorismelik-Aspire-Z5700:~/$ ./task3 1
5 Memory exception occured! Code: 0xb
```

Рассмотрим поведение программы с точки зрения системы:

```
1  lorismelik@lorismelik-Aspire-Z5700:~/$ strace ./task3.o 0
2  execve("./task3.o", ["./task3.o", "0"], 0x7ffe877df248 /* 57 vars */) = 0
3  brk(NULL)                               = 0x561f00400000
4  access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
5  access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
6  openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
7  fstat(3, {st_mode=S_IFREG|0644, st_size=88178, ...}) = 0
8  mmap(NULL, 88178, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f692df70000
9  close(3)                                = 0
10 access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
11 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
12 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\260\34\2\0\0\0\0"..., 832) = 832
13 fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0
14 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f692df6e000
15 mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f692d96e000
16 mprotect(0x7f692db55000, 2097152, PROT_NONE) = 0
17 mmap(0x7f692dd55000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0
       x7f692dd55000
18 mmap(0x7f692dd5b000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
       x7f692dd5b000
19 close(3)                                = 0
20 arch_prctl(ARCH_SET_FS, 0x7f692df6f4c0) = 0
21 mprotect(0x7f692dd55000, 16384, PROT_READ) = 0
22 mprotect(0x561effcea000, 4096, PROT_READ) = 0
23 mprotect(0x7f692df86000, 4096, PROT_READ) = 0
24 munmap(0x7f692df70000, 88178)           = 0
25 rt_sigaction(SIGFPE, {sa_handler=0x561effae98fa, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0
       x7f692d9acf20}, NULL, 8) = 0
26 rt_sigaction(SIGSEGV, {sa_handler=0x561effae9972, sa_mask=[], sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|
       SA_INTERRUPT|SA_NODEFER|SA_RESETHAND|0x3ae97f0, sa_restorer=0x7f692d9acf20}, NULL, 8) = 0
27 rt_sigprocmask(SIG_BLOCK, ~[RTMIN RT_1], [], 8) = 0
28 getpid()                                = 19588
29 gettid()                                = 19588
30 tgkill(19588, 19588, SIGFPE)            = 0
31 rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
32 --- SIGFPE {si_signo=SIGFPE, si_code=SI_TKILL, si_pid=19588, si_uid=1000} ---
33 fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
34 brk(NULL)                               = 0x561f00400000
35 brk(0x561f00421000)                     = 0x561f00421000
36 write(1, "Arythmetic exception occured! Co"..., 40Arythmetic exception occured! Code: 0x8
37 ) = 40
38 openat(AT_FDCWD, "/etc/localtime", O_RDONLY|O_CLOEXEC) = 3
39 fstat(3, {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
40 fstat(3, {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
41 read(3, "TZif2\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\21\0\0\0\21\0\0\0\0"..., 4096) = 1544
42 lseek(3, -936, SEEK_CUR)                = 608
43 read(3, "TZif2\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\21\0\0\0\21\0\0\0\0"..., 4096) = 936
44 close(3)                                = 0
45 socket(AF_UNIX, SOCK_DGRAM|SOCK_CLOEXEC, 0) = 3
46 connect(3, {sa_family=AF_UNIX, sun_path="/dev/log"}, 110) = 0
47 sendto(3, "<13>Jan 26 15:29:11 task3: Aryth"..., 67, MSG_NOSIGNAL, NULL, 0) = 67
48 close(3)                                = 0
49 exit_group(1)                           = ?
```

```
+++ exited with 1 +++


lorismelik@lorismelik-Aspire-Z5700:~/$ cat /var/log/syslog | grep task3
Jan 26 15:29:11 lorismelik-Aspire-Z5700 task3: Arythmetic exception occured! Code: 0x8
Jan 26 15:30:02 lorismelik-Aspire-Z5700 task3: Memory exception occured! Code: 0xb
```

Результат аналогичен предыдущей программе, за исключением системного вызова tgkill на строке 28, которую raise использует для генерации исключения.

3.1.4 Обработка вложенных исключений ненадежными сигналами

Особенность ненадежных сигналов состоит в том, что если во время обработки такого сигнала возникнет еще один, то он обработан не будет. Именно из-за этого недостатка ненадежные сигналы не рекомендуется использовать в реальных программах.

Проиллюстрируем работу обработчика вложенных исключений, работающего на основе ненадежных сигналов:

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <syslog.h>

void signalArythmeticHandler(const int signalCode) {
    const char* message = "Arythmetic exception occured! Code: 0x";
    printf("%s%x\n", message, signalCode);
    raise(SIGSEGV);

    printf("Unreachable code, signals not in queue.\n");
    exit(EXIT_FAILURE);
}

void signalMemoryHandler(const int signalCode) {
    const char* message = "Memory exception occured! Code: 0x";
    printf("%s%x\n", message, signalCode);
    exit(EXIT_FAILURE);
}

int main(int argc, char *argv[]) {
    struct sigaction arythmeticAction, memoryAction;

    arythmeticAction.sa_handler = signalArythmeticHandler;
    sigemptyset(&arythmeticAction.sa_mask);
    arythmeticAction.sa_flags;

    memoryAction.sa_handler = signalMemoryHandler;
    sigemptyset(&memoryAction.sa_mask);
    memoryAction.sa_flags;

    sigaction(SIGFPE, &arythmeticAction, NULL);
    sigaction(SIGSEGV, &memoryAction, NULL);

    raise(SIGFPE);
    return 0x0;
}
```

В функции main был вызван сигнал SIGFPE, который инициировал обработчик signalArythmeticHandler. Внутри обработчика арифметического исключения был вызван сигнал SIGSEGV, который сразу же инициировал обработчик signalMemoryHandler, который завершил процесс. Данный пример иллюстрирует принцип работы ненадежных сигналов: они работают полностью асинхронно и не организуются в очередь.

```
lorismelik@lorismelik-Aspire-Z5700:~/$ gcc task4.c -o task4
lorismelik@lorismelik-Aspire-Z5700:~/$ ./task4
Arythmetic exception occured! Code: 0x8
Memory exception occured! Code: 0xb
```

Можно заметить, что вывод фразы "Unreachable code, signals not in queue."не произошел, потому что приложение завершилось до этого.

Рассмотрим системные вызовы данного процесса:

```
lorismelik@lorismelik-Aspire-Z5700:~/$ ./task4
Arythmetic exception occured! Code: 0x8
Memory exception occured! Code: 0xb

lorismelik@lorismelik-Aspire-Z5700:~/$ strace ./task4
execve("./task4", ["./task4"], 0x7ffef992ee70 /* 57 vars */) = 0
brk(NULL)                               = 0x55754315e000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
```

```
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=88178, ...}) = 0
mmap(NULL, 88178, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f5220e9d000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/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\260\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f5220e9b000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f522089b000
mprotect(0x7f5220a82000, 2097152, PROT_NONE) = 0
mmap(0x7f5220c82000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0
    x7f5220c82000
mmap(0x7f5220c88000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
    x7f5220c88000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f5220e9c4c0) = 0
mprotect(0x7f5220c82000, 16384, PROT_READ) = 0
mprotect(0x557543145000, 4096, PROT_READ) = 0
mprotect(0x7f5220eb3000, 4096, PROT_READ) = 0
munmap(0x7f5220e9d000, 88178)           = 0
rt_sigaction(SIGFPE, {sa_handler=0x557542f4581a, sa_mask=[], sa_flags=SA_RESTORER, sa_restorer=0
    x7f52208d9f20}, NULL, 8) = 0
rt_sigaction(SIGSEGV, {sa_handler=0x557542f4586b, sa_mask=[], sa_flags=SA_RESTORER|SA_NODEFER|0x2f45710,
    sa_restorer=0x7f52208d9f20}, NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[RTMIN RT_1], [], 8) = 0
getpid()                                = 19721
gettid()                                = 19721
tgkill(19721, 19721, SIGFPE)            = 0
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
--- SIGFPE {si_signo=SIGFPE, si_code=SI_TKILL, si_pid=19721, si_uid=1000} ---
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
brk(NULL)                               = 0x55754315e000
brk(0x55754317f000)                     = 0x55754317f000
write(1, "Arythmetic exception occured! Co"..., 40Arythmetic exception occured! Code: 0x8
) = 40
rt_sigprocmask(SIG_BLOCK, ~[RTMIN RT_1], [FPE], 8) = 0
getpid()                                = 19721
gettid()                                = 19721
tgkill(19721, 19721, SIGSEGV)           = 0
rt_sigprocmask(SIG_SETMASK, [FPE], NULL, 8) = 0
--- SIGSEGV {si_signo=SIGSEGV, si_code=SI_TKILL, si_pid=19721, si_uid=1000} ---
write(1, "Memory exception occured! Code: "..., 36Memory exception occured! Code: 0xb
) = 36
exit_group(1)                           = ?
+++ exited with 1 +++
```

Действительно, оба сигнала были порождены асинхронно, о чем свидетельствует вывод утилиты strace. Как только из первого обработчика был инициирован сигнал, сразу же был вызван второй обработчик.

3.1.5 Обработка вложенных исключений надежными сигналами

В отличие от ненадежных сигналов, надежные сигналы ставятся в очередь и не теряются, если один из сигналов находится в обработке. Надежные сигналы организуются при помощи структуры sigset_t, а также следующих функций:

```
int sigemptyset(sigset_t *sig_m);
int sigaddset(sigset_t *sig_m, int signr);
int sigprocmask(int mode, const sigset_t *sig_m, sigset_t *alt_sig_m);
```

С помощью этой функции можно блокировать процесс до тех пор, пока не придет нужный сигнал. Рассмотрим программу для обработки вложенных исключений, организованную посредством надежных сигналов:

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <syslog.h>

void signalArythmeticHandler(const int signalCode) {
    printf("Arythmetic exception occured! Code: 0x%x\n", signalCode);

    sigset_t sigset;
    sigemptyset(&sigset);
    sigaddset(&sigset, SIGSEGV);
    sigprocmask(SIG_SETMASK, &sigset, NULL);

    raise(SIGSEGV);

    printf("Next signal in queue.\n");
}

```

```
void signalMemoryHandler(const int signalCode) {
    printf("Memory exception occured! Code: 0x%x\n", signalCode);
    exit(EXIT_FAILURE);
}

int main(int argc, char *argv[]) {
    if (signal(SIGFPE, signalArythmeticHandler) == SIG_ERR)
        return 0x1;

    if (signal(SIGSEGV, signalMemoryHandler) == SIG_ERR)
        return 0x2;

    raise(SIGFPE);
    return 0x0;
}
```

Результат работы программы

```
lorismelik@lorismelik-Aspire-Z5700:~/$ ./task5
Arythmetic exception occured! Code: 0x8
Next signal in queue.
Memory exception occured! Code: 0xb
```

В этот раз, код следующий за функцией raise первого обработчика успешно выполнился, потому что обработка второго сигнала была поставлена в очередь, а не вызвана сразу же.

Рассмотрим системные вызовы данного процесса:

```
lorismelik@lorismelik-Aspire-Z5700:~/$ ./task5
Arythmetic exception occured! Code: 0x8
Next signal in queue.
Memory exception occured! Code: 0xb

lorismelik@lorismelik-Aspire-Z5700:~/$ strace ./task5
execve("./task5", ["./task5"], 0x7ffc7225f4b0 /* 57 vars */) = 0
brk(NULL)                               = 0x5604df8b3000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=88178, ...}) = 0
mmap(NULL, 88178, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff481437000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/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\260\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff481435000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff480e35000
mprotect(0x7ff48101c000, 2097152, PROT_NONE) = 0
mmap(0x7ff48121c000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0
    x7ff48121c000
mmap(0x7ff481222000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
    x7ff481222000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7ff4814364c0) = 0
mprotect(0x7ff48121c000, 16384, PROT_READ) = 0
mprotect(0x5604de095000, 4096, PROT_READ) = 0
mprotect(0x7ff48144d000, 4096, PROT_READ) = 0
munmap(0x7ff481437000, 88178)           = 0
rt_sigaction(SIGFPE, {sa_handler=0x5604dde958ba, sa_mask=[FPE], sa_flags=SA_RESTORER|SA_RESTART,
    sa_restorer=0x7ff480e73f20}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0
rt_sigaction(SIGSEGV, {sa_handler=0x5604dde9595c, sa_mask=[SEGV], sa_flags=SA_RESTORER|SA_RESTART,
    sa_restorer=0x7ff480e73f20}, {sa_handler=SIG_DFL, sa_mask=[], sa_flags=0}, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[RTMIN RT_1], [], 8) = 0
getpid()                                = 19731
gettid()                                = 19731
tgkill(19731, 19731, SIGFPE)            = 0
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
--- SIGFPE {si_signo=SIGFPE, si_code=SI_TKILL, si_pid=19731, si_uid=1000} ---
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
brk(NULL)                               = 0x5604df8b3000
brk(0x5604df8d4000)                     = 0x5604df8d4000
write(1, "Arythmetic exception occured! Co"..., 40Arythmetic exception occured! Code: 0x8
) = 40
rt_sigprocmask(SIG_SETMASK, [SEGV], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[RTMIN RT_1], [SEGV], 8) = 0
getpid()                                = 19731
gettid()                                = 19731
tgkill(19731, 19731, SIGSEGV)           = 0
rt_sigprocmask(SIG_SETMASK, [SEGV], NULL, 8) = 0
write(1, "Next signal in queue.\n", 22Next signal in queue.
) = 22
rt_sigreturn({mask=[]})                 = 0
--- SIGSEGV {si_signo=SIGSEGV, si_code=SI_TKILL, si_pid=19731, si_uid=1000} ---
write(1, "Memory exception occured! Code: "..., 36Memory exception occured! Code: 0xb
) = 36
exit_group(1)                           = ?
+++ exited with 1 +++
```

Можно заметить, что был вызван системный вызов rt_sigprocmask (строки 25, 29, 37, 38, 42), который устанавливает маску сигналов, а также вызов rt_sigreturn (строка 45).

3.2 Обработка программных исключений C++

В языке C++ была реализована программная обработка исключений. Конструкции языка try, catch, finally для программных исключений работают аналогично SEH в Windows.

3.2.1 Генерация и обработка исключений

Рассмотрим программу, вызывающую и обрабатывающую исключение out_of_range средствами C++:

```
1  #include <stdexcept>
2  #include <iostream>
3
4  int main(int argc, char *argv[]) {
5      try {
6          throw std::out_of_range("Some C++ exception.");
7      } catch (const std::out_of_range& exception) {
8          std::cout << exception.what() << std::endl;
9      }
10
11     return 0x0;
12 }
```

Результат работы программы

```
1  lorismelik@lorismelik-Aspire-Z5700:~/$ g++ task6.cpp -o task6
2  lorismelik@lorismelik-Aspire-Z5700:~/$ ./task6
3  Some C++ exception.
```

С помощью утилиты ltrace рассмотрим вызовы процесса к стандартной библиотеке C++:

```
1  lorismelik@lorismelik-Aspire-Z5700:~/$ ltrace -e "*exception*+*catch*+*wind*@*" ./task6
2  task6->__cxa_allocate_exception(16, 0x7ffd92c69028, 0x7ffd92c69038, 160)
                                        = 0x5650e118eef0
3  libstdc++.so.6->__cxa_init_primary_exception(0x5650e118eef0, 0x5650e03eed28, 0x7f6d34bc5fc0, 0
       x5650e118ef28) = 0x5650e118ee70
4  libstdc++.so.6->_Unwind_RaiseException(0x5650e118eed0, 0x5650e03eed28, 0x7f6d34bc5fc0, 0x5650e118ef28 <
       unfinished ...>
5  libgcc_s.so.1->_Unwind_Find_FDE(0x7f6d3491683d, 0x7ffd92c68bd8, 0x7f6d3491683e, 0)
                                = 0x7f6d3491b9c8
6  libgcc_s.so.1->_Unwind_Find_FDE(0x7f6d34bb0c66, 0x7ffd92c68cc8, 0x7f6d34bb0c67, 0)
                                = 0x7f6d34c6b510
7  libgcc_s.so.1->_Unwind_Find_FDE(0x5650e01edcb2, 0x7ffd92c68cc8, 0x5650e01edcb3, 0)
                                = 0x5650e01edf40
8  libstdc++.so.6->_Unwind_GetLanguageSpecificData(0x7ffd92c68c20, 1, 0x474e5543432b2b00, 0x5650e118eed0)
           = 0x5650e01ee010
9  libstdc++.so.6->_Unwind_GetRegionStart(0x7ffd92c68c20, 0x5650e01ee010, 0x7ffd92c68ab0, 0x5650e118eed0)
           = 0x5650e01edc6a
10 libstdc++.so.6->_Unwind_GetIPInfo(0x7ffd92c68c20, 0x7ffd92c68a8c, 23, 7)
                                        = 0x5650e01edcb3
11 libgcc_s.so.1->_Unwind_GetCFA(0x7ffd92c68c20, 0x5650e01ee010, 0x5650e118eef0, 0)
                                  = 0x7ffd92c68f00
12 libgcc_s.so.1->_Unwind_Find_FDE(0x7f6d34bb0c66, 0x7ffd92c68cc8, 0x7f6d34bb0c67, 0)
                                = 0x7f6d34c6b510
13 libgcc_s.so.1->_Unwind_GetCFA(0x7ffd92c68c20, 4, 4, 7)
                                                          = 0x7ffd92c68ee0
14 libgcc_s.so.1->_Unwind_Find_FDE(0x5650e01edcb2, 0x7ffd92c68cc8, 0x5650e01edcb3, 0)
                                = 0x5650e01edf40
15 libgcc_s.so.1->_Unwind_GetCFA(0x7ffd92c68c20, 5, 0x5650e01edd47, 0)
                                             = 0x7ffd92c68f00
16 libstdc++.so.6->_Unwind_SetGR(0x7ffd92c68c20, 0, 0x5650e118eed0, 0x5650e118eed0)
                                = 8
17 libstdc++.so.6->_Unwind_SetGR(0x7ffd92c68c20, 1, 1, 0x7ffd92c68e98)
                                             = 8
18 libstdc++.so.6->_Unwind_SetIP(0x7ffd92c68c20, 0x5650e01edcc9, 1, 0x7ffd92c68ea0)
                                = 8
19 task6->__cxa_begin_catch(0x5650e118eed0, 0x5650e01edcc9, 1, 0x7ffd92c68ef8)
                                       = 0x5650e118eef0
20 Some C++ exception.
21 task6->__cxa_end_catch(0x7f6d34901760, 0x7f6d349028c0, 0, 2880 <unfinished ...>
22 libstdc++.so.6->_Unwind_DeleteException(0x5650e118eed0, 0x7f6d349028c0, 0, 2880 <unfinished ...>
23 libstdc++.so.6->_ZNSt9exceptionD2Ev(0x5650e118eef0, 0x5650e117d020, 0x5650e117d010, 1)
                           = 0
24 libstdc++.so.6->__cxa_free_exception(0x5650e118eef0, 0x5650e117d020, 0x5650e117d010, 1)
                         = 0
25 <... _Unwind_DeleteException resumed> )
                                                                        = 0
26 <... __cxa_end_catch resumed> )
                                                                              = 0
27 +++ exited (status 0) +++                                                    = 0
```

В первую очередь была вызвана библиотечная функция Unwind_RaiseException, которая выбрасывает исключение. Далее происходит длительный процесс раскрутки стека, о чем свидетельствуют вызовы с

префиксом Unwind. Затем вызывается обработчик блока catch функцией cxa_begin_catch и завершается функцией cxa_end_catch. После чего приложение нормально завершается.

3.2.2 Обработка вложенных исключений

В C++ обработка вложенных исключений выглядит просто и логично. При генерации исключения во внутреннем блоке ищется обработчик catch в направлении от внутренних к внешним. Рассмотрим программу, иллюстрирующую работу вложенных исключений:

```
#include <stdexcept>
#include <iostream>

int main(int argc, char *argv[]) {
    try {
        try {
            try {
                throw std::out_of_range("");
            } catch (const std::length_error& exception) {
                // Unreachable code.
                std::cout << "Length error exception." << std::endl;
            }
        } catch (const std::out_of_range& exception) {
            std::cout << "Out of range exception." << std::endl;
            throw std::overflow_error("");
        }
    } catch (const std::overflow_error& exception) {
        std::cout << "Overflow exception." << std::endl;
    }

    return 0x0;
}
```

Результат работы программы

```
lorismelik@lorismelik-Aspire-Z5700:~/$ g++ task7.cpp -o task7
lorismelik@lorismelik-Aspire-Z5700:~/$ ./task7
Out of range exception.
Overflow exception.
```

С помощью утилиты ltrace рассмотрим вызовы процесса к стандартной библиотеке C++:

```
lorismelik@lorismelik-Aspire-Z5700:~/$ ltrace -e "*exception*+*catch*+*wind*@libstdc*.so.6+task7" ./task7
task7->__cxa_allocate_exception(16, 0x7ffe9047ed08, 0x7ffe9047ed18, 160)
                                        = 0x555e46a94ef0
libstdc++.so.6->__cxa_init_primary_exception(0x555e46a94ef0, 0x555e46932d18, 0x7f4b12042fc0, 117)
             = 0x555e46a94e70
libstdc++.so.6->_Unwind_RaiseException(0x555e46a94ed0, 0x555e46932d18, 0x7f4b12042fc0, 117 <unfinished
    ...>
libstdc++.so.6->_Unwind_GetLanguageSpecificData(0x7ffe9047e8f0, 1, 0x474e5543432b2b00, 0x555e46a94ed0)
         = 0x555e46732280
libstdc++.so.6->_Unwind_GetRegionStart(0x7ffe9047e8f0, 0x555e46732280, 0x7ffe9047e780, 0x555e46a94ed0)
         = 0x555e46731dca
libstdc++.so.6->_Unwind_GetIPInfo(0x7ffe9047e8f0, 0x7ffe9047e75c, 48, 7)
                                        = 0x555e46731e13
libstdc++.so.6->_Unwind_SetGR(0x7ffe9047e8f0, 0, 0x555e46a94ed0, 0x555e46a94ed0)
                                = 8
libstdc++.so.6->_Unwind_SetGR(0x7ffe9047e8f0, 1, 2, 0x7ffe9047eb68)
                                            = 8
libstdc++.so.6->_Unwind_SetIP(0x7ffe9047e8f0, 0x555e46731e33, 2, 0x7ffe9047eb70)
                                = 8
task7->__cxa_begin_catch(0x555e46a94ed0, 0x555e46731e33, 2, 2)
                                                  = 0x555e46a94ef0
Out of range exception.
task7->__cxa_allocate_exception(16, 0x7f4b11d7f8c0, 0, 2880)
                                                   = 0x555e46a953a0
libstdc++.so.6->__cxa_init_primary_exception(0x555e46a953a0, 0x555e46932d00, 0x7f4b120430f0, 117)
             = 0x555e46a95320
libstdc++.so.6->_Unwind_RaiseException(0x555e46a95380, 0x555e46932d00, 0x7f4b120430f0, 117 <unfinished
    ...>
libstdc++.so.6->_Unwind_GetLanguageSpecificData(0x7ffe9047e8f0, 1, 0x474e5543432b2b00, 0x555e46a95380)
         = 0x555e46732280
libstdc++.so.6->_Unwind_GetRegionStart(0x7ffe9047e8f0, 0x555e46732280, 0x7ffe9047e780, 0x555e46a95380)
         = 0x555e46731dca
libstdc++.so.6->_Unwind_GetIPInfo(0x7ffe9047e8f0, 0x7ffe9047e75c, 48, 7)
                                        = 0x555e46731f01
libstdc++.so.6->_Unwind_SetGR(0x7ffe9047e8f0, 0, 0x555e46a95380, 0x555e46a95380)
                                = 8
libstdc++.so.6->_Unwind_SetGR(0x7ffe9047e8f0, 1, 1, 0x7ffe9047eb68)
                                            = 8
libstdc++.so.6->_Unwind_SetIP(0x7ffe9047e8f0, 0x555e46731f14, 1, 0x7ffe9047eb70)
                                = 8
task7->__cxa_end_catch(0x7ffe9047ebd0, 0x555e46731f14, 1, 0x7ffe9047ebc8 <unfinished ...>
libstdc++.so.6->_Unwind_DeleteException(0x555e46a94ed0, 0x555e46731f14, 0, 0x7ffe9047ebc8 <unfinished
    ...>
libstdc++.so.6->_ZNSt9exceptionD2Ev(0x555e46a94ef0, 0x555e46a94ed0, 0, 0x7ffe9047ebc8)
                        = 0x7f4b12322d78
libstdc++.so.6->__cxa_free_exception(0x555e46a94ef0, 0x555e46a94ed0, 0, 0x7ffe9047ebc8)
                          = 0
<... _Unwind_DeleteException resumed> )
                                                                                = 0
<... __cxa_end_catch resumed> )
                                                                                      = 0
```

```
task7->__cxa_begin_catch(0x555e46a95380, 0x555e46a83050, 1, 1)
                                                    = 0x555e46a953a0
Overflow exception.
task7->__cxa_end_catch(0x7f4b11d7e760, 0x7f4b11d7f8c0, 0, 2880 <unfinished ...>
libstdc++.so.6->_Unwind_DeleteException(0x555e46a95380, 0x7f4b11d7f8c0, 0, 2880 <unfinished ...>
libstdc++.so.6->_ZNSt9exceptionD2Ev(0x555e46a953a0, 0x555e46a95380, 0, 2880)
                                    = 0x7f4b12322d78
libstdc++.so.6->__cxa_free_exception(0x555e46a953a0, 0x555e46a95380, 0, 2880)
                                   = 0
<... _Unwind_DeleteException resumed> )
                                                                        = 0
<... __cxa_end_catch resumed> )
                                                                              = 0
+++ exited (status 0) +++
```

Можно заметить, что после каждого из двух выброшенных исключений произошла раскрутка стека и был найден соответствующий обработчик.

3.2.3 Выход из охраняемого кода инструкцией goto

Рассмотрим программу, которая выходит из охраняемого кода инструкцией goto:

```
#include <stdexcept>
#include <iostream>

int main(int argc, char *argv[]) {
    try {
        try {
            try {
                goto out;
            } catch (const std::length_error& exception) {
                std::cout << "Length error exception." << std::endl;
            }
        } catch (const std::out_of_range& exception) {
            std::cout << "Out of range exception." << std::endl;
        }
    } catch (const std::overflow_error& exception) {
        std::cout << "Overflow exception." << std::endl;
    }

    return 0x0;

out:
    std::cout << "Exit with goto instruction." << std::endl;
    return 0x0;
}
```

Результат работы программы свидетельствует о том, что выход из блока был успешно осуществлен:

```
lorismelik@lorismelik-Aspire-Z5700:~/$ g++ task8.cpp -o task8
lorismelik@lorismelik-Aspire-Z5700:~/$ ./task8
Exit with goto instruction.
```

С помощью утилиты ltrace рассмотрим вызовы процесса к стандартной библиотеке C++:

```
lorismelik@lorismelik-Aspire-Z5700:~/$ ltrace -e "*exception*+*catch*+*wind*@libstdc*.so.6+task8" ./task8
Exit with goto instruction.
+++ exited (status 0) +++
```

Можно заметить, что раскрутки стека осуществлено не было.

4 Вывод

В ходе выполнения данной работы был проведен анализ таких механизмов обработки аппаратных и программных исключений, доступных в unix-подобных системах, как системные сигналы и стандартные средства языка C++.

Метод обработки исключений посредством сигналов в Unix-подобных системах является намного менее удобным для программиста, чем SEH в Windows. Кроме того всегда необходимо задумываться о ненадежности сигналов или организовывать надежные. Систему вложенных исключений посредством сигналов также очень неудобно организовывать.

Программные исключения C++ являются полноценным универсальным средством для обработки программных исключений. Программисту предоставляется множество возможностей для разработки собственной системы исключений, для вложенных исключений, для очистки ресурсов (finally). Однако, исключения в C++ не умеют обрабатывать аппаратные прерывания, а также работают не так быстро из-за раскрутки стека.