Московский авиационный институт
(Национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа № 6-8 по курсу «Операционные системы»

Студент: Саженов Константин Станиславович

Группа: 8О-208

Преподаватель: Е. С. Миронов

Вариант: 38

Дата:

Оценка:

Москва, 2020

Лабораторная работа №6-8

1 Постановка задачи. Вариант 38

Данная лабораторная работа будет выполняться в ОС Unix.

Реализовать распределенную систему по асинхронной обработке запросов. В данной распределенной системе должно существовать 2 вида узлов: «управляющий» и «вычислительный». Необходимо объединить данные узлы в соответствии с той топологией, которая определена вариантом. Связь между узлами необходимо осуществить при помощи технологии очередей сообщений. Также в данной системе необходимо предусмотреть проверку доступности узлов в соответствии с вариантом. При убийстве («kill -9») любого вычислительного узла система должна пытаться максимально сохранять свою работоспособность, а именно все дочерние узлы убитого узла могут стать недоступными, но родительские узлы должны сохранить свою работоспособность. Управляющий узел отвечает за ввод команд от пользователя и отправку этих команд на
вычислительные узлы.
Список основных поддерживаемых команд:

- Создание нового вычислительного узла
- Удаление существующего вычислительного узла
- Исполнение команды на вычислительном уровне

Вариант № 38: топология – жестко сбаланисированное бинарное дерево, команда – посчитать сумму чисел, проверка доступности – ping id.

2 Метод решения

Используемые методы и системные вызовы для выполнения работы:

context_t::context_t(int io_threads)	Сопоставляется с функцией zmq_init (), как описано в zmq_init(3).
socket_t::socket_t(context_t &context, int type)	Сопоставляется с функцией zmq_socket (), как описано в zmq_socket(3).
void *zmq_init (int io_threads);	Функция zmq_init () инициализирует контекст ØMQ.
void *zmq_socket (void *context, int type);	Создать сокет ØMQ
int zmq_setsockopt (void *socket, int option_name, const void *option_value, size_t option_len);	Установить параметры сокета ØMQ
int zmq_bind (void *socket, const char *endpoint);	Функция zmq_bind () создает конечную точку для приема соединений и привязывает ее к сокету, на который ссылается аргумент socket.
int execv(const char *path, char *const argv[]);	Семейство функций exec () заменяет текущий образ процесса новым образом процесса.
void *memcpy(void *dest, const void *src, size_t n);	Функция memcpy() копирует n байтов из области памяти src в область памяти dest.

	Области памяти не могут пересекаться. Используйте memmove(3), если области памяти перекрываются.
int kill(pid_t pid, int sig);	kill - посылает сигнал процессу:
сигнал SIGTERM	запрашивает остановку работы процесса. Он может быть проигнорирован. Процессу дается время на корректное завершение. Если программа завершается корректно, значит она использовала данное время на то, чтобы сохранить свое состояние или результаты работы и освободить ресурсы. Другими словами, ее не заставляли остановиться.
сигнал SIGKILL	заставляет процесс прекратить работу немедленно. Программа не может проигнорировать этот сигнал. Несохраненные результаты будут потеряны.
int zmq_recv (void *socket, zmq_msg_t *msg, int flags);	Функция zmq_recv () должна получить сообщение от сокета, на который ссылается аргумент socket, и сохранить его в сообщении, на которое ссылается аргумент msg.
pid_t waitpid(pid_t pid, int *status, int options)	Функция wait приостанавливает выполнение текущего процесса до тех пор, пока дочерний процесс не завершится, или до появления сигнала, который либо завершает текущий процесс, либо требует вызвать функцию-обработчик.
zmq::active_poller_t()	Обработчик, который получает и обрабатывает полученные сообщения в отдельном контексте(которому можно передать лямбда-функцию)
сигнал SIGCHLD	сигнал, посылаемый при изменении статуса дочернего процесса (завершён, приостановлен или возобновлен).

Программа состоит из четырех файлов, в которых представлены основная программа, средства для логгирования, система сборки CMakeLists.txt, а также система сериализации и десериализации – protobuf.

Логика программы описывается следующим образом. Управляющий узел обрабатывает все входящие команды и сообщения. После получения команды, управляющий узел передает все команды всем дочерним узлам, которые, в свою очередь, если команда относится не к ним, отправляют команды дальше по дереву, а если узел является листом, то возвращает статус выполнения команды обратно. После получения статуса выполнения команды, родительский узел отправляет статусы «наверх», в зависимости от полученного статуса. И после всех отправлений, головной узел получает данные и выводит их на экран.

Проверка узлов на доступность осуществляется вышеописанном методом. В случае, если узел оказался недоступен(превышен лимит времени ожидания), то узел считается удаленным(как и все его дети).

Создание узла происходит в два этапа: сначала находится наименьшее расстояние от корня до дочернего узла, затем посылается команда создания ребенка этому узлу, после которого ожидается статус создания или тайм-аут, который означает недоступность узла.

Удаление узлов происходит аналогично другим операциям: посылается сигнал о команде дочерним узлам на удаление определенного узла, затем, по достижении необходимого узла, производится отправка удаления всех узлов до листьев и происходит ожидание выхода детей.

Запуск команды происходит аналогично проверки узлов на доступность.

3 Консоль

Тест 1:

```
┌─sakost@sakost-pc ~/university/2 course/os/6-8 lab ‹master*›
└─$ cmake-build-debug/6_8_lab
Using 0MQ version 4.3.4
Using Protobuf version 3.12.4
> create 1
Ok: 870242
> create 2
Ok: 870258
> create 3
Ok: 870296
> create 4
Ok: 870672
> exec 4 4 1 2 3 4
> Ok: 10
ping 1
Ok: 1
> ping 0
Ok: 0
> exec 10 1 4
Error: node is unavailable
```

Тест 2:

```
┌─sakost@sakost-pc ~/university/2 course/os/6-8 lab ‹master*›
└─$ cmake-build-debug/6_8_lab
Using 0MQ version 4.3.4
Using Protobuf version 3.12.4
> create 1
Ok: 872180
> create 2
Ok: 872225
> create 3
Ok: 872263
> create 4
Ok: 872274
> create 5
```

```
Ok: 872287
> create 6
Ok: 872296
> create 7
Ok: 872386
> create 8
Ok: 872407
> create 8
Error: node already exists
> create 9
Ok: 872523
> create 10
Ok: 872767
> create 11
Ok: 872775
> create 12
Ok: 872794
> ping 13
Ok: 0
> ping 12
Ok: 1
> remove 6
Ok
> ping 8
Ok: 1
> ping 9
Ok: 0
> exec 10 4 1 2 3 4
> Ok: 10
```

4 strace

```
┌─sakost@sakost-pc ~/university/2 course/os/6-8 lab ‹master*›
└─$ strace cmake-build-debug/6_8_lab
execve("cmake-build-debug/6_8_lab", ["cmake-build-debug/6_8_lab"], 0x7ffe9402a5c0 /* 55 vars */) = 0
brk(NULL)                               = 0x563423041000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fff90bae260) = -1 EINVAL (Недопустимый аргумент)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/tls/haswell/x86_64/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1
ENOENT (Нет такого файла или каталога)
stat("/usr/local/lib/tls/haswell/x86_64", 0x7fff90bad400) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/tls/haswell/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
stat("/usr/local/lib/tls/haswell", 0x7fff90bad400) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/tls/x86_64/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
stat("/usr/local/lib/tls/x86_64", 0x7fff90bad400) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/tls/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
stat("/usr/local/lib/tls", 0x7fff90bad400) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/haswell/x86_64/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(Нет такого файла или каталога)
stat("/usr/local/lib/haswell/x86_64", 0x7fff90bad400) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/haswell/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
```

```
stat("/usr/local/lib/haswell", 0x7fff90bad400) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/x86_64/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
stat("/usr/local/lib/x86_64", 0x7fff90bad400) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
stat("/usr/local/lib", {st_mode=S_IFDIR|0755, st_size=12288, ...}) = 0
openat(AT_FDCWD, "tls/haswell/x86_64/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "tls/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "haswell/x86_64/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "haswell/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла
или каталога)
openat(AT_FDCWD, "x86_64/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла
или каталога)
openat(AT_FDCWD, "libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "/usr/local/lib/libprotobuf.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=329160, ...}) = 0
mmap(NULL, 329160, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f9e6d87c000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libprotobuf.so.23", 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 \300\v\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=3257544, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f9e6d87a000
mmap(NULL, 3266496, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6d55c000
mprotect(0x7f9e6d618000, 2433024, PROT_NONE) = 0
mmap(0x7f9e6d618000, 1871872, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xbc000) = 0x7f9e6d618000
mmap(0x7f9e6d7e1000, 557056, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x285000)
= 0x7f9e6d7e1000
mmap(0x7f9e6d86a000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x30d000) = 0x7f9e6d86a000
mmap(0x7f9e6d878000, 6080, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f9e6d878000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/lib/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "tls/haswell/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла
или каталога)
openat(AT_FDCWD, "tls/x86_64/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла
или каталога)
openat(AT_FDCWD, "tls/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
каталога)
```

```
openat(AT_FDCWD, "haswell/x86_64/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libpthread.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libpthread.so.0", 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\0000\201\0\0\0\0\0\0"..., 832) = 832
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\307Y\373z\3054\277z\21\35\225\341\273\304<\223"..., 68, 824) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=158744, ...}) = 0
mmap(NULL, 135600, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6d53a000
mmap(0x7f9e6d541000, 65536, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7f9e6d541000
mmap(0x7f9e6d551000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17000) = 0x7f9e6d551000
mmap(0x7f9e6d556000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b000) = 0x7f9e6d556000
mmap(0x7f9e6d558000, 12720, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f9e6d558000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/lib/libzmq.so.5", 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\0\272\4\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1183096, ...}) = 0
mmap(NULL, 992600, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6d447000
mmap(0x7f9e6d48e000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x47000) = 0x7f9e6d48e000
mmap(0x7f9e6d50d000, 139264, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xc6000) = 0x7f9e6d50d000
mmap(0x7f9e6d52f000, 45056, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe7000) = 0x7f9e6d52f000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/lib/libgnutls.so.30", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libgnutls.so.30", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libgnutls.so.30", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libgnutls.so.30", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libgnutls.so.30", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libgnutls.so.30", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libgnutls.so.30", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libgnutls.so.30", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libgnutls.so.30", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libgnutls.so.30", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
```

```
openat(AT_FDCWD, "/usr/lib/libgnutls.so.30", 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@@\3\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2098592, ...}) = 0
mmap(NULL, 2107176, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6d244000
mmap(0x7f9e6d278000, 1183744, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x34000) = 0x7f9e6d278000
mmap(0x7f9e6d399000, 626688, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x155000)
= 0x7f9e6d399000
mmap(0x7f9e6d432000, 77824, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1ed000) = 0x7f9e6d432000
mmap(0x7f9e6d445000, 5928, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f9e6d445000
close(3)                        = 0
openat(AT_FDCWD, "/usr/local/lib/libbsd.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла
или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libbsd.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "tls/haswell/libbsd.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "tls/x86_64/libbsd.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "tls/libbsd.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "haswell/x86_64/libbsd.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "haswell/libbsd.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "x86_64/libbsd.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "libbsd.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libbsd.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла
или каталога)
openat(AT_FDCWD, "/usr/lib/libbsd.so.0", 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\0P@\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=96528, ...}) = 0
mmap(NULL, 102768, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6d22a000
mprotect(0x7f9e6d22e000, 77824, PROT_NONE) = 0
mmap(0x7f9e6d22e000, 61440, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x4000) = 0x7f9e6d22e000
mmap(0x7f9e6d23d000, 12288, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x13000) =
0x7f9e6d23d000
mmap(0x7f9e6d241000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x16000) = 0x7f9e6d241000
mmap(0x7f9e6d243000, 368, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f9e6d243000
close(3)                        = 0
openat(AT_FDCWD, "/usr/local/lib/libsodium.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libsodium.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет
такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libsodium.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libsodium.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла
или каталога)
openat(AT_FDCWD, "tls/libsodium.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
```

каталога)
openat(AT_FDCWD, "haswell/x86_64/libsodium.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libsodium.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libsodium.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libsodium.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libsodium.so.23", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libsodium.so.23", 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 \320\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=362968, ...}) = 0
mmap(NULL, 365576, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6d1d0000
mmap(0x7f9e6d1dd000, 233472, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xd000) = 0x7f9e6d1dd000
mmap(0x7f9e6d216000, 73728, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x46000) = 0x7f9e6d216000
mmap(0x7f9e6d228000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x57000) = 0x7f9e6d228000
close(3)   = 0
openat(AT_FDCWD, "/usr/local/lib/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/librt.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/librt.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\0P7\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=39408, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f9e6d1ce000
mmap(NULL, 43520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6d1c3000
mmap(0x7f9e6d1c6000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f9e6d1c6000
mmap(0x7f9e6d1ca000, 8192, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7f9e6d1ca000
mmap(0x7f9e6d1cc000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8000) = 0x7f9e6d1cc000
close(3)   = 0
openat(AT_FDCWD, "/usr/local/lib/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

```
openat(AT_FDCWD, "tls/haswell/x86_64/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/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@`\t\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=20955936, ...}) = 0
mmap(NULL, 1951744, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6cfe6000
mprotect(0x7f9e6d07c000, 1269760, PROT_NONE) = 0
mmap(0x7f9e6d07c000, 966656, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x96000) = 0x7f9e6d07c000
mmap(0x7f9e6d168000, 299008, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x182000) = 0x7f9e6d168000
mmap(0x7f9e6d1b2000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1cb000) = 0x7f9e6d1b2000
mmap(0x7f9e6d1c0000, 10240, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f9e6d1c0000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/lib/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libm.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/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\260\363\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1332096, ...}) = 0
mmap(NULL, 1331224, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6cea0000
```

```
mmap(0x7f9e6ceaf000, 638976, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xf000) = 0x7f9e6ceaf000
mmap(0x7f9e6cf4b000, 626688, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xab000) = 0x7f9e6cf4b000
mmap(0x7f9e6cfe4000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x143000) = 0x7f9e6cfe4000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/lib/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/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 0\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=594704, ...}) = 0
mmap(NULL, 103144, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6ce86000
mmap(0x7f9e6ce89000, 69632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f9e6ce89000
mmap(0x7f9e6ce9a000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x14000) = 0x7f9e6ce9a000
mmap(0x7f9e6ce9e000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17000) = 0x7f9e6ce9e000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
```

```
каталога)
openat(AT_FDCWD, "/usr/lib/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\220\202\2\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\207\360\21\247\344\314?\306\nT\320\323\335i\16t"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2159552, ...}) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 1868448, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6ccbd000
mmap(0x7f9e6cce3000, 1363968, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x26000) = 0x7f9e6cce3000
mmap(0x7f9e6ce30000, 311296, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x173000)
= 0x7f9e6ce30000
mmap(0x7f9e6ce7c000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1be000) = 0x7f9e6ce7c000
mmap(0x7f9e6ce82000, 12960, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f9e6ce82000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/lib/libz.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libz.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "tls/haswell/libz.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "tls/x86_64/libz.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "tls/libz.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libz.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла
или каталога)
openat(AT_FDCWD, "haswell/libz.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "x86_64/libz.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "libz.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libz.so.1", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "/usr/lib/libz.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0 0\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=100096, ...}) = 0
mmap(NULL, 102416, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6cca3000
mprotect(0x7f9e6cca6000, 86016, PROT_NONE) = 0
mmap(0x7f9e6cca6000, 57344, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x3000) = 0x7f9e6cca6000
mmap(0x7f9e6ccb4000, 24576, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x11000) =
0x7f9e6ccb4000
mmap(0x7f9e6ccbb000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x17000) = 0x7f9e6ccbb000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/lib/libp11-kit.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла
или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libp11-kit.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого
файла или каталога)
openat(AT_FDCWD, "tls/haswell/libp11-kit.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла
или каталога)
openat(AT_FDCWD, "tls/x86_64/libp11-kit.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла
```

или каталога)
openat(AT_FDCWD, "tls/libp11-kit.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libp11-kit.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libp11-kit.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libp11-kit.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libp11-kit.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libp11-kit.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libp11-kit.so.0", 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@\220\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1261048, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f9e6cca1000
mmap(NULL, 1264616, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6cb6c000
mmap(0x7f9e6cb95000, 647168, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x29000) = 0x7f9e6cb95000
mmap(0x7f9e6cc33000, 368640, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xc7000) = 0x7f9e6cc33000
mmap(0x7f9e6cc8d000, 81920, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x120000) = 0x7f9e6cc8d000
close(3)   = 0
openat(AT_FDCWD, "/usr/local/lib/libidn2.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libidn2.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libidn2.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libidn2.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libidn2.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libidn2.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libidn2.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libidn2.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libidn2.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libidn2.so.0", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libidn2.so.0", 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 \0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=128696, ...}) = 0
mmap(NULL, 131096, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6cb4b000
mmap(0x7f9e6cb4d000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f9e6cb4d000
mmap(0x7f9e6cb51000, 102400, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x6000) = 0x7f9e6cb51000
mmap(0x7f9e6cb6a000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e000) = 0x7f9e6cb6a000

close(3)   = 0
openat(AT_FDCWD, "/usr/local/lib/libunistring.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libunistring.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libunistring.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libunistring.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libunistring.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libunistring.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libunistring.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libunistring.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libunistring.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libunistring.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libunistring.so.2", 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 \20\1\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1574712, ...}) = 0
mmap(NULL, 1579272, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6c9c9000
mmap(0x7f9e6c9da000, 217088, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x11000) = 0x7f9e6c9da000
mmap(0x7f9e6ca0f000, 1273856, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x46000) = 0x7f9e6ca0f000
mmap(0x7f9e6cb46000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17c000) = 0x7f9e6cb46000
close(3)   = 0
openat(AT_FDCWD, "/usr/local/lib/libtasn1.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libtasn1.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libtasn1.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libtasn1.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libtasn1.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libtasn1.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libtasn1.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libtasn1.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libtasn1.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libtasn1.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libtasn1.so.6", 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 0\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=83720, ...}) = 0

mmap(NULL, 86568, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6c9b3000
mprotect(0x7f9e6c9b6000, 69632, PROT_NONE) = 0
mmap(0x7f9e6c9b6000, 49152, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f9e6c9b6000
mmap(0x7f9e6c9c2000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xf000) = 0x7f9e6c9c2000
mmap(0x7f9e6c9c7000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x13000) = 0x7f9e6c9c7000
close(3) = 0
openat(AT_FDCWD, "/usr/local/lib/libnettle.so.8", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libnettle.so.8", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libnettle.so.8", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libnettle.so.8", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libnettle.so.8", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libnettle.so.8", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libnettle.so.8", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libnettle.so.8", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libnettle.so.8", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libnettle.so.8", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libnettle.so.8", 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 \300\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=288552, ...}) = 0
mmap(NULL, 290872, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6c96b000
mprotect(0x7f9e6c977000, 233472, PROT_NONE) = 0
mmap(0x7f9e6c977000, 139264, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xc000) = 0x7f9e6c977000
mmap(0x7f9e6c999000, 90112, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2e000) = 0x7f9e6c999000
mmap(0x7f9e6c9b0000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x44000) = 0x7f9e6c9b0000
close(3) = 0
openat(AT_FDCWD, "/usr/local/lib/libhogweed.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libhogweed.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libhogweed.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libhogweed.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libhogweed.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libhogweed.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libhogweed.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)

```
openat(AT_FDCWD, "x86_64/libhogweed.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libhogweed.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libhogweed.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libhogweed.so.6", 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 \220\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=292608, ...}) = 0
mmap(NULL, 294928, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6c922000
mmap(0x7f9e6c92b000, 77824, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9000) = 0x7f9e6c92b000
mmap(0x7f9e6c93e000, 172032, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c000) = 0x7f9e6c93e000
mmap(0x7f9e6c968000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x45000) = 0x7f9e6c968000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/lib/libgmp.so.10", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libgmp.so.10", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libgmp.so.10", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libgmp.so.10", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libgmp.so.10", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libgmp.so.10", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libgmp.so.10", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libgmp.so.10", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libgmp.so.10", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libgmp.so.10", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libgmp.so.10", 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\200\20\1\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=649512, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f9e6c920000
mmap(NULL, 651784, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6c880000
mprotect(0x7f9e6c891000, 573440, PROT_NONE) = 0
mmap(0x7f9e6c891000, 475136, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x11000) = 0x7f9e6c891000
mmap(0x7f9e6c905000, 94208, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x85000) = 0x7f9e6c905000
mmap(0x7f9e6c91d000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9c000) = 0x7f9e6c91d000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/lib/libffi.so.7", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libffi.so.7", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
```

```
openat(AT_FDCWD, "tls/haswell/libffi.so.7", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libffi.so.7", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libffi.so.7", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libffi.so.7", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libffi.so.7", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libffi.so.7", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libffi.so.7", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libffi.so.7", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libffi.so.7", 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@ \0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=42960, ...}) = 0
mmap(NULL, 46376, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6c874000
mmap(0x7f9e6c876000, 24576, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f9e6c876000
mmap(0x7f9e6c87c000, 8192, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8000) = 0x7f9e6c87c000
mmap(0x7f9e6c87e000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9000) = 0x7f9e6c87e000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/lib/libdl.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/x86_64/libdl.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/haswell/libdl.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/x86_64/libdl.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "tls/libdl.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/x86_64/libdl.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "haswell/libdl.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "x86_64/libdl.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "libdl.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/local/lib/libdl.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/usr/lib/libdl.so.2", 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\20\22\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=18608, ...}) = 0
mmap(NULL, 20624, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9e6c86e000
mmap(0x7f9e6c86f000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f9e6c86f000
mmap(0x7f9e6c871000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f9e6c871000
mmap(0x7f9e6c872000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f9e6c872000
```

```
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f9e6c86c000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f9e6c86a000
arch_prctl(ARCH_SET_FS, 0x7f9e6c86b040) = 0
mprotect(0x7f9e6ce7c000, 12288, PROT_READ) = 0
mprotect(0x7f9e6c872000, 4096, PROT_READ) = 0
mprotect(0x7f9e6c87e000, 4096, PROT_READ) = 0
mprotect(0x7f9e6c91d000, 8192, PROT_READ) = 0
mprotect(0x7f9e6c9b0000, 8192, PROT_READ) = 0
mprotect(0x7f9e6c968000, 8192, PROT_READ) = 0
mprotect(0x7f9e6c9c7000, 4096, PROT_READ) = 0
mprotect(0x7f9e6cb46000, 16384, PROT_READ) = 0
mprotect(0x7f9e6cb6a000, 4096, PROT_READ) = 0
mprotect(0x7f9e6cc8d000, 40960, PROT_READ) = 0
mprotect(0x7f9e6ccbb000, 4096, PROT_READ) = 0
mprotect(0x7f9e6ce9e000, 4096, PROT_READ) = 0
mprotect(0x7f9e6cfe4000, 4096, PROT_READ) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f9e6c868000
mprotect(0x7f9e6d1b2000, 53248, PROT_READ) = 0
mprotect(0x7f9e6d556000, 4096, PROT_READ) = 0
mprotect(0x7f9e6d1cc000, 4096, PROT_READ) = 0
mprotect(0x7f9e6d228000, 4096, PROT_READ) = 0
mprotect(0x7f9e6d241000, 4096, PROT_READ) = 0
mprotect(0x7f9e6d432000, 69632, PROT_READ) = 0
mprotect(0x7f9e6d52f000, 32768, PROT_READ) = 0
mprotect(0x7f9e6d86a000, 53248, PROT_READ) = 0
mprotect(0x5634212f0000, 4096, PROT_READ) = 0
mprotect(0x7f9e6d8f9000, 4096, PROT_READ) = 0
munmap(0x7f9e6d87c000, 329160)          = 0
set_tid_address(0x7f9e6c86b310)         = 875700
set_robust_list(0x7f9e6c86b320, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7f9e6d541b90, sa_mask=[], sa_flags=SA_RESTORER|SA_SIGINFO,
sa_restorer=0x7f9e6d54e0f0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7f9e6d541c30, sa_mask=[], sa_flags=SA_RESTORER|SA_RESTART|
SA_SIGINFO, sa_restorer=0x7f9e6d54e0f0}, NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
brk(NULL)                               = 0x563423041000
brk(0x563423062000)                     = 0x563423062000
brk(0x563423083000)                     = 0x563423083000
getrandom("\x9b", 1, GRND_NONBLOCK)     = 1
stat("/etc/gnutls/config", 0x7fff90bae1c0) = -1 ENOENT (Нет такого файла или каталога)
futex(0x7f9e6d1c06bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f9e6d1c06c8, FUTEX_WAKE_PRIVATE, 2147483647) = 0
readlink("/proc/self/exe", "/home/sakost/university/2 course"..., 4096) = 69
eventfd2(0, EFD_CLOEXEC)                = 3
fcntl(3, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(3, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
fcntl(3, F_GETFL)                       = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(3, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
getpid()                                = 875700
getpid()                                = 875700
```

```
getrandom("\x13\x41\x12\x1d\x28\x44\x1a\x67\x26\x15\x10\x44\x95\xa4\xfb\xa7", 16, 0) = 16
getrandom("\x85\x62\xed\x35\xa3\x0f\xea\xe7\x9c\x4d\x28\xcd\x78\x5b\x0e\xbb", 16, 0) = 16
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
write(1, "Using 0MQ version 4.3.4\n", 24Using 0MQ version 4.3.4
) = 24
write(1, "Using Protobuf version 3.12.4\n", 30Using Protobuf version 3.12.4
) = 30
eventfd2(0, EFD_CLOEXEC)                = 4
fcntl(4, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(4, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
fcntl(4, F_GETFL)                       = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(4, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
getpid()                                = 875700
epoll_create1(EPOLL_CLOEXEC)            = 5
epoll_ctl(5, EPOLL_CTL_ADD, 4, {0, {u32=587543984, u64=94781925832112}}) = 0
epoll_ctl(5, EPOLL_CTL_MOD, 4, {EPOLLIN, {u32=587543984, u64=94781925832112}}) = 0
getpid()                                = 875700
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f9e6c067000
mprotect(0x7f9e6c068000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone(child_stack=0x7f9e6c866a70, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|
CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[875701], tls=0x7f9e6c867640, child_tidptr=0x7f9e6c867910) = 875701
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
eventfd2(0, EFD_CLOEXEC)                = 6
fcntl(6, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(6, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
fcntl(6, F_GETFL)                       = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(6, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
getpid()                                = 875700
epoll_create1(EPOLL_CLOEXEC)            = 7
epoll_ctl(7, EPOLL_CTL_ADD, 6, {0, {u32=587544176, u64=94781925832304}}) = 0
epoll_ctl(7, EPOLL_CTL_MOD, 6, {EPOLLIN, {u32=587544176, u64=94781925832304}}) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f9e6b866000
mprotect(0x7f9e6b867000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone(child_stack=0x7f9e6c065a70, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|
CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[875702], tls=0x7f9e6c066640, child_tidptr=0x7f9e6c066910) = 875702
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
eventfd2(0, EFD_CLOEXEC)                = 8
fcntl(8, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(8, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
fcntl(8, F_GETFL)                       = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(8, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
getpid()                                = 875700
eventfd2(0, EFD_CLOEXEC)                = 9
fcntl(9, F_GETFL)                       = 0x2 (flags O_RDWR)
fcntl(9, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
fcntl(9, F_GETFL)                       = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(9, F_SETFL, O_RDWR|O_NONBLOCK)    = 0
getpid()                                = 875700
getpid()                                = 875700
```

```
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
unlink("@socket_flow_0.ipc")            = -1 ENOENT (Нет такого файла или каталога)
socket(AF_UNIX, SOCK_STREAM|SOCK_CLOEXEC, 0) = 10
bind(10, {sa_family=AF_UNIX, sun_path=@"socket_flow_0.ipc"}, 20) = 0
listen(10, 100)                         = 0
getsockname(10, {sa_family=AF_UNIX, sun_path=@"socket_flow_0.ipc"}, [128->20]) = 0
getpid()                                = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 875700
write(8, "\1\0\0\0\0\0\0\0", 8)         = 8
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f9e6b065000
mprotect(0x7f9e6b066000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone(child_stack=0x7f9e6b864a70, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|
CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[875703], tls=0x7f9e6b865640, child_tidptr=0x7f9e6b865910) = 875703
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
write(1, "> ", 2> )                     = 2
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
read(0, create 1
"create 1\n", 1024)             = 9
stat("/home/sakost/university/2 course/os/6-8 lab/cmake-build-debug/6_8_lab", {st_mode=S_IFREG|0755,
st_size=1797944, ...}) = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f9e6c86b310) = 876105
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, 0)     = 0 (Timeout)
brk(0x5634230a4000)                       = 0x5634230a4000
getpid()                                = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 875700
write(9, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, -1)    = 1 ([{fd=9, revents=POLLIN}])
getpid()                                = 875700
read(9, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, -1)    = 1 ([{fd=9, revents=POLLIN}])
getpid()                                = 875700
read(9, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, 0)     = 0 (Timeout)
futex(0x7fff90bae118, FUTEX_WAKE_PRIVATE, 1) = 1
write(1, "Ok: 876105\n", 11Ok: 876105
)           = 11
write(1, "> ", 2> )                     = 2
read(0, create 2
"create 2\n", 1024)             = 9
getpid()                                = 875700
poll([{fd=8, events=POLLIN}], 1, 0)     = 1 ([{fd=8, revents=POLLIN}])
getpid()                                = 875700
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
```

```
getpid()                                = 875700
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, -1)    = 1 ([{fd=9, revents=POLLIN}])
getpid()                                = 875700
read(9, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, 0)     = 0 (Timeout)
stat("/home/sakost/university/2 course/os/6-8 lab/cmake-build-debug/6_8_lab", {st_mode=S_IFREG|0755,
st_size=1797944, ...}) = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f9e6c86b310) = 876177
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 875700
write(9, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, -1)    = 1 ([{fd=9, revents=POLLIN}])
getpid()                                = 875700
read(9, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, -1)    = 1 ([{fd=9, revents=POLLIN}])
getpid()                                = 875700
read(9, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, 0)     = 0 (Timeout)
futex(0x7fff90bae118, FUTEX_WAKE_PRIVATE, 1) = 1
write(1, "Ok: 876177\n", 11Ok: 876177
)           = 11
write(1, "> ", 2> )                      = 2
read(0, create 3
"create 3\n", 1024)             = 9
getpid()                                = 875700
poll([{fd=8, events=POLLIN}], 1, 0)     = 1 ([{fd=8, revents=POLLIN}])
getpid()                                = 875700
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 875700
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, -1)    = 1 ([{fd=9, revents=POLLIN}])
getpid()                                = 875700
read(9, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
poll([{fd=8, events=POLLIN}], 1, 0)     = 1 ([{fd=8, revents=POLLIN}])
getpid()                                = 875700
```

```
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 875700
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, -1)    = 1 ([{fd=9, revents=POLLIN}])
getpid()                                = 875700
read(9, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, -1)    = 1 ([{fd=9, revents=POLLIN}])
getpid()                                = 875700
read(9, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, -1)    = 1 ([{fd=9, revents=POLLIN}])
getpid()                                = 875700
read(9, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, 0)     = 0 (Timeout)
futex(0x7fff90bae118, FUTEX_WAKE_PRIVATE, 1) = 1
write(1, "Ok: 876196\n", 11Ok: 876196
)           = 11
write(1, "> ", 2> )                      = 2
read(0, exec 3 3 1 2 3
"exec 3 3 1 2 3\n", 1024)       = 15
getpid()                                = 875700
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, -1)    = 1 ([{fd=9, revents=POLLIN}])
getpid()                                = 875700
read(9, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, -1)    = 1 ([{fd=9, revents=POLLIN}])
getpid()                                = 875700
read(9, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 875700
poll([{fd=9, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
```

```
write(1, "> ", 2> )                     = 2
read(0, Ok: 6
remove 2
"remove 2\n", 1024)            = 9
getpid()                               = 875700
poll([{fd=8, events=POLLIN}], 1, 0)    = 0 (Timeout)
getpid()                               = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
getpid()                               = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
getpid()                               = 875700
poll([{fd=9, events=POLLIN}], 1, -1)   = 1 ([{fd=9, revents=POLLIN}])
getpid()                               = 875700
read(9, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                               = 875700
poll([{fd=9, events=POLLIN}], 1, 0)    = 0 (Timeout)
wait4(0, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 876177
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=876177, si_uid=1000, si_status=0,
si_utime=1, si_stime=0} ---
write(1, "Ok\n", 3Ok
)                  = 3
write(1, "> ", 2> )                     = 2
read(0, ping 1
"ping 1\n", 1024)              = 7
getpid()                               = 875700
poll([{fd=8, events=POLLIN}], 1, 0)    = 1 ([{fd=8, revents=POLLIN}])
getpid()                               = 875700
read(8, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                               = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
getpid()                               = 875700
poll([{fd=8, events=POLLIN}], 1, 0)    = 1 ([{fd=8, revents=POLLIN}])
getpid()                               = 875700
read(8, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                               = 875700
poll([{fd=8, events=POLLIN}], 1, 0)    = 0 (Timeout)
getpid()                               = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
getpid()                               = 875700
poll([{fd=9, events=POLLIN}], 1, -1)   = 1 ([{fd=9, revents=POLLIN}])
getpid()                               = 875700
read(9, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                               = 875700
poll([{fd=9, events=POLLIN}], 1, 0)    = 0 (Timeout)
write(1, "Ok: 1\n", 6Ok: 1
)                = 6
write(1, "> ", 2> )                     = 2
read(0, "", 1024)                      = 0
futex(0x7f9e6b865910, FUTEX_WAIT, 875703, NULL) = 0
getpid()                               = 875700
poll([{fd=8, events=POLLIN}], 1, 0)    = 0 (Timeout)
getpid()                               = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)        = 8
poll([{fd=8, events=POLLIN}], 1, 0)    = 0 (Timeout)
getpid()                               = 875700
```

```
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
getpid()                                = 875700
write(4, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 875700
getpid()                                = 875700
write(9, "\1\0\0\0\0\0\0\0", 8)         = 8
getpid()                                = 875700
poll([{fd=3, events=POLLIN}], 1, -1)    = 1 ([{fd=3, revents=POLLIN}])
getpid()                                = 875700
read(3, "\1\0\0\0\0\0\0\0", 8)          = 8
getpid()                                = 875700
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=876105, si_uid=1000, si_status=0,
si_utime=1, si_stime=1} ---
close(7)                                = 0
close(6)                                = 0
close(5)                                = 0
close(4)                                = 0
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

5 Исходный код

```
┌─sakost@sakost-pc ~/university/2 course/os/6-8 lab ‹master*›
└─$ cat logger.hpp
#ifndef _LOGGER_HPP_
#define _LOGGER_HPP_

#include <iostream>
#include <sstream>


enum loglevel_e
   {logNOTHING, logERROR, logWARNING, logINFO, logDEBUG, logDEBUG1, logDEBUG2, logDEBUG3,
logDEBUG4};

class logIt
{
public:
   static std::string levelToString(loglevel_e level){
      switch (level)
      {
      case logNOTHING:
         return "";
      case logERROR:
         return "ERROR";
      case logWARNING:
         return "WARNING";
      case logINFO:
         return "INFO";
      case logDEBUG:
         return "DEBUG";
```

```
        case logDEBUG1:
            return "DEBUG1";
        case logDEBUG2:
            return "DEBUG2";
        case logDEBUG3:
            return "DEBUG3";
        case logDEBUG4:
            return "DEBUG4";
        default:
            return std::to_string(level);
        }
    }

    logIt(loglevel_e _loglevel = logERROR) {
        this->_log_level = _loglevel;
        if(_log_level == logNOTHING) return;
        _buffer << levelToString(_loglevel) << "(pid: " << std::to_string(::getpid()) << ")" <<  " :"
            << std::string(
                _log_level > logDEBUG
                ? (_log_level - logDEBUG) * 4
                : 1
                , ' ');
    }

    template <typename T>
    logIt & operator<<(T const & value)
    {
        if(_log_level == logNOTHING) return *this;
        _buffer << value;
        return *this;
    }

    ~logIt()
    {
        _buffer << std::endl;
        // This is atomic according to the POSIX standard
        // http://www.gnu.org/s/libc/manual/html_node/Streams-and-Threads.html
        std::cerr << _buffer.str();
    }

private:
    loglevel_e _log_level;
    std::ostringstream _buffer;
};

extern loglevel_e loglevel;

#define log(level) \
if (level > loglevel) ; \
else logIt(level)

#define log_(level, addon) \
if(level > loglevel);      \
else logIt(level) << "[" << addon << "]"
```

```
#endif
 ┌─sakost@sakost-pc ~/university/2 course/os/6-8 lab ‹master*›
 └─$ cat main.cpp
#include <lab.pb.h>

#include <thread>
#include <wait.h>
#include <cinttypes>
#include <vector>
#include <cstdlib>
#include <string>
#include <filesystem>
#include <iostream>

#ifndef _WIN32

#include <unistd.h>

#else
#error "Windows is not supported"
#endif

#define ZMQ_BUILD_DRAFT_API

#include <zmq.hpp>
#include <zmq_addon.hpp>
#include "logger.hpp"

#define UNUSED(x) (void)(x)

using node_id_t = int64_t;

namespace fs = std::filesystem;

const node_id_t default_parent_nid = 0;


using namespace std;

loglevel_e loglevel = logERROR;

const int WAIT_TIME = 3000; // in milliseconds

class node_exception : exception {
public:
    explicit node_exception(Execute_ErrorType errorType) : exception() {
        m_error_type = errorType;
    }

    node_exception(Execute_ErrorType errorType, const string &msg) : node_exception(errorType) {
        m_error_type = errorType;
        this->msg = msg;
    }

    [[nodiscard]] const char *what() const _GLIBCXX_TXN_SAFE_DYN _GLIBCXX_NOTHROW override {
```

```
        switch (m_error_type) {
            case Execute_ErrorType_CUSTOM:
            default:
                string ans = "Some unhandled error";
                if (!this->msg.empty()) {
                    ans = msg;
                }
                char *buf = new char[ans.size() + 1];
                memcpy(buf, ans.data(), sizeof(char) * (ans.size() + 1));
                return buf;
        }
    }

    Execute_ErrorType m_error_type;
protected:
    string msg;
};

static fs::path init_exe_path() noexcept {
#ifdef _WIN32
    wchar_t path[MAX_PATH] = { 0 };
    GetModuleFileNameW(NULL, path, MAX_PATH);
    return path;
#else
    char result[PATH_MAX];
    ssize_t count = readlink("/proc/self/exe", result, PATH_MAX);
    return std::string(result, (count > 0) ? count : 0);
#endif
}

static fs::path exepath = init_exe_path();

inline fs::path getexepath() {
    return exepath;
}

inline pid_t create_child(const string &path_to_exec, const string &nid, const string &parent) {
    if (!fs::exists(path_to_exec)) {
        throw runtime_error("path not exists");
    }
    pid_t pid = fork();
    if (pid > 0) return pid;
    if (pid < 0) throw runtime_error("fork error");
    execl(path_to_exec.c_str(), path_to_exec.c_str(), nid.c_str(), parent.c_str());
    throw runtime_error("execl error");
}

class logger_d {
public:
    explicit logger_d(const string &mes, loglevel_e lvl, const string &addon) {
        msg = mes;
        this->addon = addon;
        this->lvl = lvl;
        log_(lvl, addon) << msg;
    }
```

```
    explicit logger_d(const string &&mes, loglevel_e lvl, const string &addon) {
        msg = mes;
        this->lvl = lvl;
        this->addon = addon;
        log_(lvl, addon) << msg;
    }

    ~logger_d() {
        log_(lvl, addon) << "Ok(" << msg << ')';
    }

private:
    string msg, addon;
    loglevel_e lvl;
};

class Node {
public:
    static const size_t tree_base = 2; // count of children of each node

    enum class S_IDXS : int {
        FLOW,
        CALLBACK,
        SUBSCRIBE,
        UPSTREAM,
    };

    static string get_address(const string &str) {
        return "ipc://@" + str + ".ipc";
    }

    static string get_pub_sub_address(const string &str) {
        return get_address("socket_flow_" + str);
    }

    static string get_pub_sub_address(const string &&str) {
        return get_address("socket_flow_" + str);
    }

    static string get_pub_sub_address(const node_id_t &nodeId) {
        return get_pub_sub_address(::to_string(nodeId));
    }

    static string get_upstream_callback_address(const string &str) {
        return get_address("socket_callback_" + str);
    }

    static string get_upstream_callback_address(const string &&str) {
        return get_address("socket_callback_" + str);
    }

    static string get_upstream_callback_address(const node_id_t parentId, const node_id_t nodeId) {
        return get_upstream_callback_address(::to_string(parentId) + "_" + ::to_string(nodeId));
    }
```

```
    Node(node_id_t nodeId, node_id_t parentId, zmq::context_t *ctx) {
        m_ctx = ctx;
        m_node_id = nodeId;
        m_parent_id = parentId;
        sockets.emplace_back(*ctx, zmq::socket_type::pub); // flow
        sockets.emplace_back(*ctx, zmq::socket_type::pull); // callback
//        get(S_IDXS::CALLBACK).set(zmq::sockopt::rcvtimeo, WAIT_TIME);

        bind_flow();
    }

    void bind_flow() {
        logger_d ll("binding flow socket...", logDEBUG3, ::to_string(m_node_id));
        log_(logDEBUG4, ::to_string(m_node_id)) << "address to flow bind: " <<
get_pub_sub_address(m_node_id);
        get(S_IDXS::FLOW).bind(get_pub_sub_address(m_node_id));
    }

    void connect_callback(node_id_t nodeId) {
        logger_d ll("connecting callback socket...", logDEBUG3, ::to_string(m_node_id));
        log_(logDEBUG4, ::to_string(m_node_id)) << "node id to callback connect: " << nodeId << "; parent id: "
                                                << m_node_id;
        get(S_IDXS::CALLBACK).connect(get_upstream_callback_address(m_node_id, nodeId));
    }

    class exit_exception : runtime_error {
    public:
        exit_exception() : runtime_error("exiting") {}
    };


    virtual void loop() = 0;

    size_t children_count = 0;
protected:
    virtual Execute wait_child() {
        logger_d ll("waiting for a child answer...", logDEBUG2, ::to_string(m_node_id));
        zmq::message_t msg;
        Execute proto_msg;
        auto rc = get(S_IDXS::CALLBACK).recv(msg, zmq::recv_flags::none);
        if (!rc) {
            log_(logWARNING, m_node_id) << "timeout";
            proto_msg.set_client_id(-1);
            return proto_msg;
        }
        if (!proto_msg.ParseFromString(msg.to_string())) {
            log_(logERROR, ::to_string(m_node_id)) << "invalid format";
            return Execute();
        }
        Execute_ErrorType type = proto_msg.error_type();
        if (proto_msg.msg() == Execute_MessageType_ERROR && type != Execute_ErrorType_NOT_FOR_ME &&
            type != Execute_ErrorType_OK) {
            log_(logERROR, m_node_id) << string(proto_msg.error_message());
```

```
            throw node_exception(type, proto_msg.error_message());
        }
        return proto_msg;
    }

    zmq::socket_t &get(S_IDXS idx) {
        return sockets[(size_t) idx];
    }

    void send_flow(zmq::message_t &msg) {
        logger_d ll("sending flow message...", logDEBUG3, ::to_string(m_node_id));
        UNUSED(get(S_IDXS::FLOW).send(zmq::buffer(""), zmq::send_flags::sndmore));
        UNUSED(get(S_IDXS::FLOW).send(msg, zmq::send_flags::none));
    }

    void send_flow(const Execute &proto_msg) {
        string raw_msg = proto_msg.SerializeAsString();
        zmq::message_t msg(raw_msg);
        send_flow(msg);
    }

    void add_pid(node_id_t nid, pid_t pid) {
        pids[nid] = pid;
    }

    void delete_pid(node_id_t nid) {
        int status;
        log_(logINFO, m_node_id) << "waiting for " << nid << "(pid: " << pids[nid] << ")";
        waitpid(pids[nid], &status, 0);
        pids.erase(nid);
    }


    virtual void process_callback_message(const Execute &input_msg) = 0;

    map<node_id_t, pid_t> pids;
    vector<zmq::socket_t> sockets;
    node_id_t m_node_id, m_parent_id;
    zmq::context_t *m_ctx;
};

class CalcNode : Node {
public:
    CalcNode(node_id_t node_id, node_id_t parent_id, zmq::context_t *ctx) : Node(node_id, parent_id, ctx) {
        sockets.emplace_back(*ctx, zmq::socket_type::sub); // subscribe
        sockets.emplace_back(*ctx, zmq::socket_type::push); // upstream

        log_(logDEBUG1, ::to_string(m_node_id)) << "address to subscribe connect: " <<
get_pub_sub_address(m_parent_id);
        connect_subscribe();
        log_(logDEBUG1, ::to_string(m_node_id)) << "setting subscribe opt to \"\"";
        get(S_IDXS::SUBSCRIBE).set(zmq::sockopt::subscribe, "");
        bind_upstream();
        log_(logDEBUG1, ::to_string(m_node_id)) << "all sockets are connected and bound(at CalcNode)";
```

```
        Execute proto_msg;
        proto_msg.set_msg(Execute_MessageType_CREATE);
        proto_msg.set_client_id(m_node_id);
        send_upstream(proto_msg);
    }


    void loop() override {
        zmq::active_poller_t poller;
        poller.add(get(S_IDXS::SUBSCRIBE), zmq::event_flags::pollin, [&](zmq::event_flags ef) {
            Execute proto_msg;
            if (handle_msg(proto_msg, S_IDXS::SUBSCRIBE)) {
                process_subscribe_message(proto_msg);
            }
        });
        poller.add(get(S_IDXS::CALLBACK), zmq::event_flags::pollin, [&](zmq::event_flags ef) {
            Execute proto_msg;
            if (handle_msg(proto_msg, S_IDXS::CALLBACK)) {
                process_callback_message(proto_msg);
            }
        });

        static const chrono::milliseconds timeout(-1);
        while (true) {
            auto n = poller.wait(timeout);
            if (!n) {
                break;
            }
        }
    }

private:
    bool handle_msg(Execute &proto_msg, S_IDXS sock_type) {
        zmq::message_t msg;
        auto res = get(sock_type).recv(msg, zmq::recv_flags::none);
        if (!res) {
            log_(logERROR, ::to_string(m_node_id)) << "something went wrong...";
            return false;
        }
        if (sock_type == S_IDXS::SUBSCRIBE) {
            res = get(sock_type).recv(msg, zmq::recv_flags::none);
            if (!res) {
                log_(logERROR, ::to_string(m_node_id)) << "something went wrong...";
                return false;
            }
        }
        bool parsed = proto_msg.ParseFromString(msg.to_string());
        if (!parsed) {
            log_(logERROR, ::to_string(m_node_id)) << "wrong message format";
            return false;
        }
        return true;
    }

    void process_subscribe_message(const Execute &input_msg) {
```

```
        log_(logDEBUG2, ::to_string(m_node_id)) << "processing message from subscribe socket...";
        switch (input_msg.msg()) {
            case Execute_MessageType_CREATE:
                create_command(input_msg);
                break;
            case Execute_MessageType_PING:
                ping_command(input_msg);
                break;
            case Execute_MessageType_EXEC:
                exec_command(input_msg);
                break;
            case Execute_MessageType_REMOVE:
                remove_command(input_msg);
                break;
            case Execute_MessageType_GET_MIN_DEPTH:
                get_min_depth_command(input_msg);
                break;
            case Execute_MessageType_EXIT_ALL:
                exit_all_command(input_msg);
                break;
            case Execute_MessageType_ERROR:
                error_command(input_msg);
                break;
            default:
                return;
        }
    }

    void process_callback_message(const Execute &input_msg) override {
//        log_(logWARNING, ::to_string(m_node_id)) << "not expected to be got there";
        logger_d ll("processing message from callback socket(just resend it to up)...", logDEBUG3,
                    ::to_string(m_node_id));
        send_upstream(input_msg);
    }

    void create_command(const Execute &input_msg) {
        if (input_msg.client_id() != m_node_id) {
            if (children_count > 0) {
                send_flow(input_msg);
                Execute proto_msg;
                proto_msg.set_msg(Execute_MessageType_ERROR);
                proto_msg.set_error_type(Execute_ErrorType_NOT_FOR_ME);
                for (size_t i = 0; i < children_count; ++i) {
                    logger_d ll("waiting for a child answer...", logDEBUG2, ::to_string(m_node_id));
                    Execute msg = wait_child();
                    if (msg.client_id() == -1 && pids.count(input_msg.client_id()) > 0) {
                        log_(logERROR, m_node_id) << "child not available(pid: " << pids[input_msg.client_id()]
                                                  << "; nid: " << input_msg.client_id() << ")";
                        delete_pid(input_msg.client_id());
                    }
                    if (msg.msg() == Execute_MessageType_CREATE && msg.error_type() == Execute_ErrorType_OK)
{
                        proto_msg.set_msg(Execute_MessageType_CREATE);
                        proto_msg.set_error_type(Execute_ErrorType_OK);
                        proto_msg.set_client_id(msg.client_id());
```

```
                proto_msg.add_args(msg.args(0));
                break;
            }
        }
        send_upstream(proto_msg);
    } else {
        Execute proto_msg;
        proto_msg.set_client_id(m_node_id);
        proto_msg.set_msg(Execute_MessageType_ERROR);
        proto_msg.set_error_type(Execute_ErrorType_NOT_FOR_ME);
        send_upstream(proto_msg);
    }
    return;
}
node_id_t nid = input_msg.args(0);
log_(logDEBUG, ::to_string(m_node_id)) << "I'm going to be a parent for " << nid;
if (children_count < tree_base) {
    log_(logDEBUG2, m_node_id) << "children count: " << children_count;
    children_count++;
    pid_t pid = create_child(getexepath(), ::to_string(nid), ::to_string(m_node_id));
    add_pid(nid, pid);
    connect_callback(nid);
    log_(logDEBUG, ::to_string(m_node_id)) << "Waiting for a child";
    wait_child();

    Execute proto_msg;
    proto_msg.set_msg(Execute_MessageType_CREATE);
    proto_msg.set_error_type(Execute_ErrorType_OK);
    proto_msg.set_client_id(nid);
    proto_msg.add_args(pid);
    send_upstream(proto_msg);
} else {
    Execute proto_msg;
    proto_msg.set_client_id(m_node_id);
    proto_msg.set_msg(Execute_MessageType_ERROR);
    proto_msg.set_error_type(Execute_ErrorType_CUSTOM);
    proto_msg.set_error_message("This node has already had all children");
    send_upstream(proto_msg);
}
}

void ping_command(const Execute &input_msg) {
if (input_msg.client_id() != m_node_id) {
    bool sent = false;
    logger_d ll("I was NOT pinged(pinged not me)", logDEBUG2, ::to_string(m_node_id));
    if (children_count > 0) {
        bool my_child = pids.count(input_msg.client_id()) > 0;

        send_flow(input_msg);
        Execute proto_msg;
        proto_msg.set_msg(Execute_MessageType_ERROR);
        proto_msg.set_error_type(Execute_ErrorType_NOT_FOR_ME);
        log_(logDEBUG2, m_node_id) << "children count: " << children_count;
        for (size_t i = 0; i < children_count; ++i) {
            Execute msg;
```

```
            try {
                logger_d ll("waiting for a child answer...", logDEBUG2, ::to_string(m_node_id));
                msg = wait_child();
            } catch (node_exception &err) {
                continue;
            }
            if (msg.client_id() == -1 && pids.count(input_msg.client_id()) > 0 &&
                proto_msg.error_type() != Execute_ErrorType_OK) {
                proto_msg.set_msg(Execute_MessageType_ERROR);
                proto_msg.set_error_type(Execute_ErrorType_NOT_AVAILABLE);
                proto_msg.set_client_id(input_msg.client_id());
            }
            if (msg.msg() == Execute_MessageType_PING && msg.error_type() == Execute_ErrorType_OK) {
                proto_msg.set_msg(Execute_MessageType_PING);
                proto_msg.set_error_type(Execute_ErrorType_OK);
                proto_msg.set_client_id(msg.client_id());
                sent = true;
                send_upstream(msg);
            }
        }
        if (my_child) {
            // node is not available
            if (proto_msg.error_type() != Execute_ErrorType_OK &&
                proto_msg.error_type() != Execute_ErrorType_NOT_FOR_ME) {
                log_(logINFO, m_node_id) << "deleting node with nid: " << input_msg.client_id();
                children_count--;
                delete_pid(input_msg.client_id());
            }
        }
        if(!sent) {
            send_upstream(proto_msg);
        }
    } else {
        Execute proto_msg;
        proto_msg.set_client_id(m_node_id);
        proto_msg.set_msg(Execute_MessageType_ERROR);
        proto_msg.set_error_type(Execute_ErrorType_NOT_FOR_ME);
        send_upstream(proto_msg);
    }
    return;
  }
  log_(logDEBUG, ::to_string(m_node_id)) << "I was pinged";
  Execute proto_msg(input_msg);
  proto_msg.set_error_type(Execute_ErrorType_OK);
  proto_msg.set_msg(Execute_MessageType::Execute_MessageType_PING);
  proto_msg.set_client_id(m_node_id);
  send_upstream(proto_msg);
}

void exec_command(const Execute &input_msg) {
  if (input_msg.client_id() != m_node_id) {
    send_flow(input_msg);
    return;
  }
  // proceed command
```

```
    int64_t ans(0);
    for (const auto &el: input_msg.args()) {
        ans += el;
    }
    // send result
    Execute proto_msg;
    proto_msg.set_msg(Execute_MessageType_EXEC);
    proto_msg.set_error_type(Execute_ErrorType_OK);
    proto_msg.set_client_id(m_node_id);
    proto_msg.add_args(ans);
    send_upstream(proto_msg);
}

void get_min_depth_command(const Execute &input_msg) {
    if (children_count == tree_base) {
        Execute proto_msg(input_msg);
        proto_msg.set_args(0, proto_msg.args(0) + 1);
        send_flow(proto_msg);

        size_t min_depth = numeric_limits<size_t>::max();
        node_id_t nid = -1;
        for (int i = 0; i < children_count; ++i) {
            zmq::message_t msg;
            auto res = get(S_IDXS::CALLBACK).recv(msg, zmq::recv_flags::none);
            if (!res) {
                log_(logERROR, m_node_id) << "no received messages";
                return;
            }
            string raw_msg = msg.to_string();

            if (!proto_msg.ParseFromString(raw_msg)) {
                log_(logERROR, m_node_id) << "wrong format";
                return;
            }

            if (proto_msg.args(0) < min_depth) {
                min_depth = proto_msg.args(0);
                nid = proto_msg.client_id();
            }

            proto_msg.Clear();
        }
        assert(nid != -1 && "Some wrong behavior");
        proto_msg.set_msg(Execute_MessageType_GET_MIN_DEPTH);
        proto_msg.set_error_type(Execute_ErrorType_OK);
        proto_msg.set_client_id(nid);
        proto_msg.add_args(min_depth);
        send_upstream(proto_msg);
    } else {
        Execute proto_msg(input_msg);
        proto_msg.set_client_id(m_node_id);
        send_upstream(proto_msg);
    }
}
```

```
void remove_command(const Execute &input_msg) {
    if (input_msg.client_id() == m_node_id) {
        logger_d ll("remove command processing...", logDEBUG2, ::to_string(m_node_id));
        Execute proto_msg(input_msg);
        proto_msg.set_client_id(m_parent_id);
        proto_msg.set_msg(Execute_MessageType_REMOVE);
        proto_msg.set_error_type(Execute_ErrorType_OK);
        send_upstream(proto_msg);

        zmq::poller_t<> poller;
        poller.add(get(S_IDXS::UPSTREAM), zmq::event_flags::pollout);
        vector<zmq::poller_event<>> events(1);
        const chrono::milliseconds timeout(-1);
        poller.wait_all(events, timeout);

        proto_msg.Clear();
        proto_msg.set_client_id(m_node_id);
        proto_msg.set_msg(Execute_MessageType_EXIT_ALL);
        exit_all_command(proto_msg);
        return;
    }

    if (children_count > 0) {
        send_flow(input_msg);
        Execute msg;
        for (size_t i = 0; i < children_count; ++i) {
            logger_d ll("waiting for a child answer...", logDEBUG2, ::to_string(m_node_id));
            Execute temp_msg;
            temp_msg = wait_child();
            if (temp_msg.error_type() == Execute_ErrorType_OK) {
                msg = temp_msg;
            }
        }
        if (msg.error_type() == Execute_ErrorType_OK) {
            if (msg.client_id() == m_node_id) {
                children_count--;
                delete_pid(input_msg.client_id());
            }
            send_upstream(msg);
            return;
        }
    }
    Execute proto_msg(input_msg);
    proto_msg.set_msg(Execute_MessageType_ERROR);
    proto_msg.set_error_type(Execute_ErrorType_NOT_FOR_ME);
    send_upstream(proto_msg);
}

void exit_all_command(const Execute &input_msg) {
    if (children_count > 0) {
        send_flow(input_msg);
    }
    zmq::poller_t<> poller;
    poller.add(get(S_IDXS::FLOW), zmq::event_flags::pollout);
    const chrono::milliseconds timeout(-1);
```

```
        vector<zmq::poller_event<>> events(1);
        poller.wait_all(events, timeout);
        throw exit_exception();
    }

    void error_command(const Execute &input_msg) {
        send_upstream(input_msg);
    }

    void bind_upstream() {
        logger_d ll(string("binding upstream socket..."), logDEBUG3, ::to_string(m_node_id));
        log_(logDEBUG4, ::to_string(m_node_id)) << "address to bind upstream: "
                                        << get_upstream_callback_address(m_parent_id, m_node_id);
        get(S_IDXS::UPSTREAM).bind(get_upstream_callback_address(m_parent_id, m_node_id));
    }

    void connect_subscribe() {
        logger_d ll(string("connecting subscribe socket..."), logDEBUG3, ::to_string(m_node_id));
        log_(logDEBUG4, ::to_string(m_node_id)) << "address to connect subscribe: " <<
get_pub_sub_address(m_parent_id);
        get(S_IDXS::SUBSCRIBE).connect(get_pub_sub_address(m_parent_id));
    }


    void send_upstream(zmq::message_t &msg) {
        logger_d ll(string("sending upstream message..."), logDEBUG3, ::to_string(m_node_id));
        UNUSED(get(S_IDXS::UPSTREAM).send(msg, zmq::send_flags::none));
    }

    void send_upstream(const Execute &proto_msg) {
        string raw_msg = proto_msg.SerializeAsString();
        zmq::message_t msg(raw_msg);
        send_upstream(msg);
    }
};

class HeadNode : Node {
public:
    HeadNode(node_id_t nodeId, node_id_t parentId, zmq::context_t *ctx) : Node(nodeId, parentId, ctx) {
        thr = thread([this]() {
            this->loop();
        });
    }

    ~HeadNode() {
        running = false;
        thr.join();
        exit_all_command();
    }

    pid_t create_node(node_id_t nid) {
        if (nid == m_node_id) return -1;
        if (ping_command(nid)) {
            return -1;
        }
```

```
    logger_d ll("creating new node with " + ::to_string(nid) + " nid", logDEBUG, ::to_string(m_node_id));
    if (children_count < tree_base) {
        children_count++;

        lock_guard<mutex> lk(m_mut);
        pid_t pid = create_child(getexepath(), ::to_string(nid), ::to_string(m_node_id));
        connect_callback(nid);

        logger_d ll("waiting for a child answer...", logDEBUG2, ::to_string(m_node_id));
        wait_child();
        return pid;
    }

    auto depth = get_min_depth();

    assert(depth.first != node_id_t(-1) && "Count of actual children is zero, but correspond variable is not zero");

    log_(logDEBUG1, ::to_string(m_node_id)) << "got min depth " << depth.first << " from node with id "
                                            << depth.second;

    Execute proto_msg;
    proto_msg.set_msg(Execute_MessageType::Execute_MessageType_CREATE);
    proto_msg.set_client_id(depth.second);
    proto_msg.add_args(nid);

    lock_guard<mutex> lk(m_mut);
    send_flow(proto_msg);

    pid_t pid = 0;
    for (int i = 0; i < children_count; ++i) {
        auto msg = wait_child();
        if (msg.error_type() == Execute_ErrorType_OK) {
            pid = msg.args(0);
        } else if (msg.error_type() != Execute_ErrorType_NOT_FOR_ME) {
            log_(logWARNING, ::to_string(m_node_id)) << "some error occurred: " << msg.error_message();
        } else {
            log_(logDEBUG, ::to_string(m_node_id)) << "not for " << msg.client_id();
        }
    }
    return pid;
}

pair<size_t, node_id_t> get_min_depth() {
    logger_d ll("getting minimal depth...", logDEBUG2, ::to_string(m_node_id));
    Execute proto_msg;
    proto_msg.set_client_id(default_parent_nid);
    proto_msg.add_args(1);
    proto_msg.set_msg(::Execute_MessageType_GET_MIN_DEPTH);

    lock_guard<mutex> lk(m_mut);
    send_flow(proto_msg);

    size_t min_depth = -1;
    node_id_t nid = m_node_id;
    for (size_t i(0); i < sockets.size(); ++i) {
```

```
        logger_d ll("waiting for a child answer...", logDEBUG2, ::to_string(m_node_id));
        proto_msg = wait_child();

        if (proto_msg.client_id() == -1) {
            continue;
        }

        if (proto_msg.args(0) < min_depth) {
            min_depth = proto_msg.args(0);
            nid = proto_msg.client_id();
        }
    }
    return {min_depth, nid};
}

void loop() override {
    zmq::poller_t<> poller;
    poller.add(get(S_IDXS::CALLBACK), zmq::event_flags::pollin);

    const chrono::milliseconds timeout(0);
    vector<zmq::poller_event<>> events(1);
    while (running) {
        this_thread::sleep_for(chrono::milliseconds(100));
        m_mut.lock();
        zmq::message_t msg;
        const auto nin = poller.wait_all(events, timeout);
        if (!nin) {
            m_mut.unlock();
            log_(logDEBUG4, m_node_id) << "nothing to process";
            continue;
        }
        UNUSED(events[0].socket.recv(msg, zmq::recv_flags::none));
        m_mut.unlock();

        Execute proto_msg;
        bool parsed = proto_msg.ParseFromString(msg.to_string());
        if (!parsed) {
            log_(logERROR, m_node_id) << "wrong message format";
            return;
        }
        process_callback_message(proto_msg);
    }
}

bool remove_command(node_id_t nid) {
    if (children_count > 0) {
        Execute proto_msg;
        proto_msg.set_client_id(nid);
        proto_msg.set_msg(Execute_MessageType_REMOVE);

        lock_guard<mutex> lk(m_mut);
        send_flow(proto_msg);
        for (int i = 0; i < children_count; ++i) {
            logger_d ll("waiting for a child answer...", logDEBUG2, ::to_string(m_node_id));
            Execute msg = wait_child();
```

```
            if (msg.error_type() == Execute_ErrorType_OK) {
                int ind((int) children_count - i - 1);
                for (int j = 0; j < ind; ++j) {
                    logger_d ll("waiting for a child answer...", logDEBUG2, ::to_string(m_node_id));
                    wait_child();
                }
                if (msg.client_id() == m_node_id) {
                    delete_pid(nid);
                    children_count--;
                }
                return true;
            }
        }

    }
    return false;
}

bool ping_command(node_id_t nid) {
    if (children_count > 0) {
        bool my_child = pids.count(nid) > 0;

        Execute proto_msg;
        proto_msg.set_client_id(nid);
        proto_msg.set_msg(Execute_MessageType_PING);
        {
            lock_guard<mutex> lk(m_mut);
            send_flow(proto_msg);
            bool res = false;
            for (size_t i = 0; i < children_count; ++i) {
                logger_d ll("waiting for a child answer...", logDEBUG2, ::to_string(m_node_id));
                Execute msg = wait_child();
                if (msg.client_id() == -1) {
                    if (my_child) {
                        delete_pid(nid);
                        children_count--;
                    }
                    return res;
                }
                if (msg.msg() == Execute_MessageType_PING && msg.error_type() == Execute_ErrorType_OK) {
                    res = true;
                }
            }
            return res;
        }
    }
    return false;
}

bool exec_command(node_id_t nid, const vector<int64_t> &args) {
    if (!ping_command(nid)) {
        return false;
    }

    Execute proto_msg;
```

```
        proto_msg.set_msg(Execute_MessageType_EXEC);
        proto_msg.set_client_id(nid);
        for (const auto &el: args) {
            proto_msg.add_args(el);
        }
        send_flow(proto_msg);
        return true;
    }

    void exit_all_command() {
        Execute proto_msg;
        proto_msg.set_msg(Execute_MessageType_EXIT_ALL);
        send_flow(proto_msg);

        zmq::poller_t<> poller;
        poller.add(get(S_IDXS::FLOW), zmq::event_flags::pollout);

        const chrono::milliseconds timeout(-1);
        vector<zmq::poller_event<>> events(1);
        poller.wait_all(events, timeout);
    }


private:
    void process_callback_message(const Execute &input_msg) override {
        if (input_msg.msg() == Execute_MessageType_EXEC) {
            cout << "Ok: " << input_msg.args(0) << endl;
        } else if (input_msg.msg() == Execute_MessageType_ERROR) {
            cout << "Error: " << input_msg.error_message() << endl;
        } else {
            log_(logWARNING, m_node_id) << "Unexpected callback message received";
        }
    }

    thread thr;
    mutex m_mut;
    volatile bool running = true;
};

void print_versions(ostream &out) {
    auto zver = zmq::version();
    out << "Using 0MQ version " << get<0>(zver) << '.' << get<1>(zver) << '.' << get<2>(zver) << endl;
    out << "Using Protobuf version " <<
google::protobuf::internal::VersionString(GOOGLE_PROTOBUF_VERSION) << endl;
}


int main(int argc, char **argv) { // node_id parent_id
    GOOGLE_PROTOBUF_VERIFY_VERSION;

    zmq::context_t ctx;

    if (argc > 1) {
        string node_id_s(argv[1]), parent_id_s(argv[2]);
```

```
    int64_t node_id = stoi(node_id_s), parent_id = stoi(parent_id_s);

    log_(logDEBUG4, node_id) << "node id: " << node_id << " parent id: " << parent_id;

    CalcNode node(node_id, parent_id, &ctx);
    try {
        log_(logDEBUG1, node_id) << "going to loop...";
        node.loop();
    } catch (Node::exit_exception &err) {
        log_(logINFO, node_id) << "exiting...";
    }
} else {
    print_versions(cout);

    HeadNode head(default_parent_nid, -1, &ctx);

    string cmd;
    cout << "> " << flush;
    while (cin >> cmd) {
        int nid;
        if (!(cin >> nid)) {
            break;
        }
        if (cmd == "create") {
            pid_t pid;
            try {
                pid = head.create_node(nid);
            } catch (node_exception &err) {
                cout << "Error: " << err.what() << endl;
                continue;
            }
            if (pid <= 0) {
                cout << "Error: node already exists" << endl;
            } else {
                cout << "Ok: " << pid << endl;
            }
        } else if (cmd == "ping") {
            bool res;
            try {
                res = head.ping_command(nid);
            } catch (node_exception &err) {
                cout << "Error: " << err.what() << endl;
                continue;
            }
            if (!res) {
                cout << "Ok: 0" << endl;
            } else {
                cout << "Ok: 1" << endl;
            }
        } else if (cmd == "remove") {
            bool res;
            try {
                res = head.remove_command(nid);
            } catch (node_exception &err) {
                cout << "Error: " << err.what() << endl;
```

```
                continue;
            }
            if (!res) {
                cout << "Error: node is unavailable" << endl;
            } else {
                cout << "Ok" << endl;
            }
        } else if (cmd == "exec") {
            size_t n;
            if (!(cin >> n)) {
                break;
            }
            vector<int64_t> args(n);
            for (size_t i = 0; i < n; ++i) {
                cin >> args[i];
            }
            if (!head.exec_command(nid, args)) {
                cout << "Error: node is unavailable" << endl;
            }
        }
        cout << "> " << flush;
    }
    google::protobuf::ShutdownProtobufLibrary();
  }

  return 0;
}
 ┌─sakost@sakost-pc ~/university/2 course/os/6-8 lab ‹master*›
 └─$ cat lab.proto
syntax = "proto3";

message Execute{
  int64 client_id = 1;
  enum MessageType{
    PING=0;
    CREATE=1;
    EXEC=2;
    REMOVE=3;
    GET_MIN_DEPTH = 4;
    EXIT_ALL = 5;
    ERROR=10;
  }
  MessageType msg = 2;

  repeated sint64 args = 3;

  enum ErrorType{
    NOT_FOR_ME=0;
    OK=1;
    NOT_AVAILABLE=2;
    CUSTOM=10;
  }
  ErrorType error_type = 14;
  string error_message = 15;
}
```

```
┌─sakost@sakost-pc ~/university/2 course/os/6-8 lab ‹master*›
└─$ cat CMakeLists.txt
cmake_minimum_required(VERSION 3.17)
project(6_8_lab)

set(CMAKE_CXX_STANDARD 17)

find_package(Protobuf REQUIRED)

include_directories(${Protobuf_INCLUDE_DIRS})
include_directories(${CMAKE_CURRENT_BINARY_DIR})
protobuf_generate_cpp(PROTO_SRCS PROTO_HDRS lab.proto)

find_package(cppzmq)

add_executable(6_8_lab main.cpp ${PROTO_SRCS})
target_link_libraries(6_8_lab ${Protobuf_LIBRARIES} cppzmq)
```

6 Вывод

Во время выполнения данной лабораторной работы я получил много новых навыков по работе с очередями, в частности, с библиотекой 0MQ(a.k.a. zeromq). На примере данной библиотеки я реализовал некоторую топологию вычислительного кластера, взаимодействующего посредством асинхронных очередей.

Моя топология являлась жестко сбалансированным бинарным деревом, в котором узлы являлись вычислительными узлами кластера. Достичь того, что дерево является жестко сбалансированным, мне позволила дополнительная операция поиска узла с минимальной высотой. Команда, которая выполнялась на узлах, являлась асинхронной операцией и производила сложение чисел в массиве.

Помимо данной операции в кластере была реализована операция проверки доступности узлов по уникальному идентефикатору(присвоеному пользователем на этапе создания узла). Данная операция крайне необходима практически всем другим операциям, т.к. она позволяет без лишних действий проверять узел на доступность.

Операция создания узла являлась одной из самых сложных, поскольку в ней требовалось производить множество проверок и на каждом этапе могла возникнуть та или иная ошибка в обработке данных.

Также операция удаления была не из самых простых, поскольку при разработке программы я столкнулся с тем, что дочерние узлы оказывались процессами-зомби, которые необходимо было ожидать(вызывать wait) в родительском процессе, что являлось некоторым ограничением, которое я обошел благодаря обработчикам сигналов.

ZeroMQ – одна из библиотек, которая реализует технологию асинхронных очередей, что, безусловно, является одной из самых часто используемых технологий на практике, поскольку она позволяет обмениваться сообщениями, не блокируя процесс и не вдаваясь в подробности реализации ОС, тем самым обеспечивая кроссплатформенность. Также хочу отметить C++ обертку над C API данной библиотеки – cppzmq. Данная обертка помогла мне не вдаваться в подробности реализации библиотеки и не производить проверку на ошибки при каждом вызове той

или иной функции. Также данная обертка позволила использовать многие приемущества языка C++11, такие как лямбда-функции. Мне очень понравилось работать с данной библиотекой, поскольку существует несколько подробных ресурсов, описывающих работу с ней, а также данная библиотека не требует сторонних сервисов, поскольку работает в отдельном потоке основной программы.