Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №6
по курсу «Операционные системы»
III Семестр

Вариант 29

Студент:	Короткевич Л. В.
Группа:	М80-208Б-19
Преподаватель:	Миронов Е.С
Оценка:	
Дата:	

Москва 2020

1. Постановка задачи. Вариант 29.

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 узле

Вариант 29: топология – бинарное дерево, команда – словарь, 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.

Программа состоит из двух файлов, представляющих управляющий и вычислительные узлы, а также из двух библиотек: первая, server.hpp, описывает взаимодействие между узлами; вторая, tree.hpp, описывает бинарное дерево.

Работа программы обуславливается следующей логистикой: управляющий узел обрабатывает поступающие запросы, после чего пересылает их дочерним узлам (или выводит сообщение об ошибке). Затем по мере получения команды дочерними узлами, они посылают ее дальше в один из низлежащих узлов, из которого затем возвращается сообщение об ошибке или успехе, которое потом посылается вверх по дереву.

Проверка узлов на доступность зиждется на топологии дерева: удаление определенной ноды влечет рекурсивное уничтожение ее детей. Если же узел оказался недоступен, то по истечении времени (SNDTIMEO) будет получено сообщение о его состоянии, которое затем отправится вверх по дереву до самого управляющего узла.

3. Тестирование

Тест №1: проверка общей работоспособности программы, всех операций.

```
leoub@leoub-VirtualBox:~/MEGAsync/ver_src6_final$ ./main
> create 1
OK:91521
> create 2
OK:91525
> create 3
OK:91529
> pingall
Received nodes: 1 2 3
Nodes list: 1 2 3
> remove 2
OK
> pingall
Received nodes: 1
Nodes list: 1
> ping 1
OK:1
> ping 2
Error: Not found
> exec 1 + test 1
OK:1
> exec 1 ? test
OK:1: 1
> remove 1
OK
> ping 1
Error: Not found
> pingall
Error: Tree is empty
> exit
```

Тест №2: удаление большого кол-ва последовательных узлов.

```
leoub@leoub-VirtualBox:~/MEGAsync/ver_src6_final$ ./main
> create 1
OK:91547
> create 2
OK:91552
> create 3
OK:91556
> create 4
OK:91560
> create 5
OK:91564
> create 6
OK:91568
> pingall
Received nodes: 1 2 3 4 5 6
```

Nodes list: 1 2 3 4 5 6
> remove 3
OK
> pingall
Received nodes: 1 2
Nodes list: 1 2
> ping 3
Error: Not found
> ping 2
OK:1
> exit

Тест №3: удаление корня.
leoub@leoub-VirtualBox:~/MEGAsync/ver_src6_final$ ./main
> create 1
OK:91658
> ping 1
OK:1
> remove 1
OK
> ping 1
Error: Not found
> pingall
Error: Tree is empty
> exit

strace:
execve("./main", ["./main"], 0x7ffe14ef55c0 /* 61 vars */) = 0
brk(NULL) = 0x56127b7c5000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffca8d6a350) = -1 EINVAL (Invalid argument)
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=68833, ...}) = 0
mmap(NULL, 68833, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f465024b000
close(3) = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libzmq.so.5", 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`z\1\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=675776, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f4650249000
mmap(NULL, 678128, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f46501a3000
mmap(0x7f46501b9000, 430080, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x16000) = 0x7f46501b9000
mmap(0x7f4650222000, 126976, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7f000) = 0x7f4650222000
mmap(0x7f4650241000, 32768, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9d000) = 0x7f4650241000
close(3) = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\240\341\t\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=1952928, ...}) = 0
mmap(NULL, 1968128, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464ffc2000
mprotect(0x7f4650058000, 1286144, PROT_NONE) = 0
mmap(0x7f4650058000, 983040, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x96000) = 0x7f4650058000
mmap(0x7f4650148000, 299008, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x186000) = 0x7f4650148000

```
mmap(0x7f4650192000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1cf000) = 0x7f4650192000
mmap(0x7f46501a0000, 10240, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f46501a0000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\3405\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=104984, ...}) = 0
mmap(NULL, 107592, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464ffa7000
mmap(0x7f464ffaa000, 73728, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f464ffaa000
mmap(0x7f464ffbc000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x15000) = 0x7f464ffbc000
mmap(0x7f464ffc0000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18000) = 0x7f464ffc0000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\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\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 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
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\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 68, 880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464fdb5000
mprotect(0x7f464fdda000, 1847296, PROT_NONE) = 0
mmap(0x7f464fdda000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7f464fdda000
mmap(0x7f464ff52000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0x7f464ff52000
mmap(0x7f464ff9d000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f464ff9d000
mmap(0x7f464ffa3000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f464ffa3000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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\200\302\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=355016, ...}) = 0
mmap(NULL, 357384, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464fd5d000
mmap(0x7f464fd69000, 229376, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xc000) = 0x7f464fd69000
mmap(0x7f464fda1000, 73728, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x44000) = 0x7f464fda1000
mmap(0x7f464fdb3000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x55000) = 0x7f464fdb3000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpgm-5.2.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\240L\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=302056, ...}) = 0
mmap(NULL, 321584, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464fd0e000
mmap(0x7f464fd12000, 163840, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) = 0x7f464fd12000
mmap(0x7f464fd3a000, 118784, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2c000) = 0x7f464fd3a000
mmap(0x7f464fd57000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x48000) = 0x7f464fd57000
mmap(0x7f464fd59000, 14384, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f464fd59000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libnorm.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\257\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=690344, ...}) = 0
```

```
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f464fd0c000
mmap(NULL, 1420000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464fbb1000
mmap(0x7f464fbbb000, 421888, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xa000) = 0x7f464fbbb000
mmap(0x7f464fc22000, 217088, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x71000) =
0x7f464fc22000
mmap(0x7f464fc57000, 16384, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xa5000) = 0x7f464fc57000
mmap(0x7f464fc5b000, 723680, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f464fc5b000
close(3)                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgssapi_krb5.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\0P\321\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=309712, ...}) = 0
mmap(NULL, 312128, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464fb64000
mmap(0x7f464fb6f000, 204800, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xb000) = 0x7f464fb6f000
mmap(0x7f464fba1000, 49152, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3d000) =
0x7f464fba1000
mmap(0x7f464fbad000, 16384, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x48000) = 0x7f464fbad000
close(3)                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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\0\220\201\0\0\0\0\0\0"..., 832) = 832
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O\305\3743\364B\2216\244\224\306@\261\23\327o"..., 68, 824) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=157224, ...}) = 0
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O\305\3743\364B\2216\244\224\306@\261\23\327o"..., 68, 824) = 68
mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464fb41000
mmap(0x7f464fb48000, 69632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x7000) = 0x7f464fb48000
mmap(0x7f464fb59000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18000) =
0x7f464fb59000
mmap(0x7f464fb5e000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1c000) = 0x7f464fb5e000
mmap(0x7f464fb60000, 13432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f464fb60000
close(3)                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\300\363\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=1369352, ...}) = 0
mmap(NULL, 1368336, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464f9f2000
mmap(0x7f464fa01000, 684032, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xf000) = 0x7f464fa01000
mmap(0x7f464faa8000, 618496, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xb6000) =
0x7f464faa8000
mmap(0x7f464fb3f000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x14c000) = 0x7f464fb3f000
close(3)                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libkrb5.so.3", 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 ?\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=902016, ...}) = 0
mmap(NULL, 904640, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464f915000
mprotect(0x7f464f937000, 700416, PROT_NONE) = 0
mmap(0x7f464f937000, 397312, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x22000) = 0x7f464f937000
mmap(0x7f464f998000, 299008, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x83000) =
0x7f464f998000
mmap(0x7f464f9e2000, 65536, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xcc000) = 0x7f464f9e2000
close(3)                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libk5crypto.so.3", 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\240D\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=191040, ...}) = 0
```

```
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f464f913000
mmap(NULL, 196696, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464f8e2000
mprotect(0x7f464f8e6000, 172032, PROT_NONE) = 0
mmap(0x7f464f8e6000, 114688, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x4000) = 0x7f464f8e6000
mmap(0x7f464f902000, 53248, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x20000) =
0x7f464f902000
mmap(0x7f464f910000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2d000) = 0x7f464f910000
mmap(0x7f464f912000, 88, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1,
0) = 0x7f464f912000
close(3)                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libcom_err.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\200$\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=22600, ...}) = 0
mmap(NULL, 24744, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464f8db000
mmap(0x7f464f8dd000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2000) = 0x7f464f8dd000
mmap(0x7f464f8df000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) =
0x7f464f8df000
mmap(0x7f464f8e0000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x4000) = 0x7f464f8e0000
close(3)                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libkrb5support.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\3605\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=56096, ...}) = 0
mmap(NULL, 58344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464f8cc000
mmap(0x7f464f8cf000, 28672, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x3000) = 0x7f464f8cf000
mmap(0x7f464f8d6000, 12288, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xa000) =
0x7f464f8d6000
mmap(0x7f464f8d9000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xc000) = 0x7f464f8d9000
close(3)                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libkeyutils.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"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=22600, ...}) = 0
mmap(NULL, 24592, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464f8c5000
mmap(0x7f464f8c7000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2000) = 0x7f464f8c7000
mmap(0x7f464f8c9000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x4000) =
0x7f464f8c9000
mmap(0x7f464f8ca000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x4000) = 0x7f464f8ca000
close(3)                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libresolv.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 G\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=101320, ...}) = 0
mmap(NULL, 113280, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464f8a9000
mprotect(0x7f464f8ad000, 81920, PROT_NONE) = 0
mmap(0x7f464f8ad000, 65536, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x4000) = 0x7f464f8ad000
mmap(0x7f464f8bd000, 12288, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x14000) =
0x7f464f8bd000
mmap(0x7f464f8c1000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x17000) = 0x7f464f8c1000
mmap(0x7f464f8c3000, 6784, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -
1, 0) = 0x7f464f8c3000
close(3)                        = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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 \22\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=18816, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f464f8a7000
```

```
mmap(NULL, 20752, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464f8a1000
mmap(0x7f464f8a2000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1000) = 0x7f464f8a2000
mmap(0x7f464f8a4000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) =
0x7f464f8a4000
mmap(0x7f464f8a5000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x3000) = 0x7f464f8a5000
close(3)                        = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f464f89f000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f464f89d000
arch_prctl(ARCH_SET_FS, 0x7f464f8a0600) = 0
mprotect(0x7f464ff9d000, 12288, PROT_READ) = 0
mprotect(0x7f464f8a5000, 4096, PROT_READ) = 0
mprotect(0x7f464f8c1000, 4096, PROT_READ) = 0
mprotect(0x7f464f8ca000, 4096, PROT_READ) = 0
mprotect(0x7f464f8d9000, 4096, PROT_READ) = 0
mprotect(0x7f464fb5e000, 4096, PROT_READ) = 0
mprotect(0x7f464f8e0000, 4096, PROT_READ) = 0
mprotect(0x7f464f910000, 4096, PROT_READ) = 0
mprotect(0x7f464f9e2000, 57344, PROT_READ) = 0
mprotect(0x7f464fb3f000, 4096, PROT_READ) = 0
mprotect(0x7f464fbad000, 8192, PROT_READ) = 0
mprotect(0x7f464ffc0000, 4096, PROT_READ) = 0
mprotect(0x7f4650192000, 45056, PROT_READ) = 0
mprotect(0x7f464fc57000, 12288, PROT_READ) = 0
mprotect(0x7f464fd57000, 4096, PROT_READ) = 0
mprotect(0x7f464fdb3000, 4096, PROT_READ) = 0
mprotect(0x7f4650241000, 28672, PROT_READ) = 0
mprotect(0x56127b0b1000, 4096, PROT_READ) = 0
mprotect(0x7f4650289000, 4096, PROT_READ) = 0
munmap(0x7f465024b000, 68833)           = 0
set_tid_address(0x7f464f8a08d0)         = 5080
set_robust_list(0x7f464f8a08e0, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7f464fb48bf0, sa_mask=[], sa_flags=SA_RESTORER|SA_SIGINFO,
sa_restorer=0x7f464fb563c0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7f464fb48c90, sa_mask=[], sa_flags=SA_RESTORER|SA_RESTART|
SA_SIGINFO, sa_restorer=0x7f464fb563c0}, 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)                               = 0x56127b7c5000
brk(0x56127b7e6000)                     = 0x56127b7e6000
futex(0x7f46501a06bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f46501a06c8, FUTEX_WAKE_PRIVATE, 2147483647) = 0
openat(AT_FDCWD, "/sys/devices/system/cpu/online", O_RDONLY|O_CLOEXEC) = 3
read(3, "0\n", 8192)                    = 2
close(3)                                = 0
openat(AT_FDCWD, "/sys/devices/system/cpu", O_RDONLY|O_NONBLOCK|O_CLOEXEC|O_DIRECTORY) = 3
fstat(3, {st_mode=S_IFDIR|0755, st_size=0, ...}) = 0
getdents64(3, /* 17 entries */, 32768)  = 520
getdents64(3, /* 0 entries */, 32768)   = 0
close(3)                                = 0
getpid()                                = 5080
sched_getaffinity(5080, 128, [0])       = 8
openat(AT_FDCWD, "/etc/nsswitch.conf", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=542, ...}) = 0
read(3, "# /etc/nsswitch.conf\n#\n# Example"..., 4096) = 542
read(3, "", 4096)                       = 0
close(3)                                = 0
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=68833, ...}) = 0
mmap(NULL, 68833, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f465024b000
close(3)                                = 0
```

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/tls/x86_64/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/lib/x86_64-linux-gnu/tls/x86_64/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/tls/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/lib/x86_64-linux-gnu/tls/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/tls/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/lib/x86_64-linux-gnu/tls/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/tls/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/lib/x86_64-linux-gnu/tls", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/x86_64/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/lib/x86_64-linux-gnu/x86_64/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/lib/x86_64-linux-gnu/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/lib/x86_64-linux-gnu/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/lib/x86_64-linux-gnu", {st_mode=S_IFDIR|0755, st_size=69632, ...}) = 0
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/tls/x86_64/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/usr/lib/x86_64-linux-gnu/tls/x86_64/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/tls/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/usr/lib/x86_64-linux-gnu/tls/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/tls/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/usr/lib/x86_64-linux-gnu/tls/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/tls/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/usr/lib/x86_64-linux-gnu/tls", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/x86_64/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/usr/lib/x86_64-linux-gnu/x86_64/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/usr/lib/x86_64-linux-gnu/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/usr/lib/x86_64-linux-gnu/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/x86_64-linux-gnu/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/usr/lib/x86_64-linux-gnu", {st_mode=S_IFDIR|0755, st_size=69632, ...}) = 0
openat(AT_FDCWD, "/lib/tls/x86_64/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/lib/tls/x86_64/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/tls/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/lib/tls/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/tls/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/lib/tls/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/tls/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/lib/tls", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/lib/x86_64/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)

```
openat(AT_FDCWD, "/lib/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
stat("/lib/x86_64", 0x7ffca8d67120)     = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
stat("/lib/x86_64", 0x7ffca8d67120)     = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/lib/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/lib", {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
openat(AT_FDCWD, "/usr/lib/tls/x86_64/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No
such file or directory)
stat("/usr/lib/tls/x86_64/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/tls/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
stat("/usr/lib/tls/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/tls/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
stat("/usr/lib/tls/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/tls/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
stat("/usr/lib/tls", 0x7ffca8d67120)    = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/x86_64/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such
file or directory)
stat("/usr/lib/x86_64/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
stat("/usr/lib/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/x86_64/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or
directory)
stat("/usr/lib/x86_64", 0x7ffca8d67120) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/lib/libnss_db.so.2", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
stat("/usr/lib", {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
munmap(0x7f465024b000, 68833)           = 0
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=68833, ...}) = 0
mmap(NULL, 68833, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f465024b000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libnss_files.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\3005\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=51832, ...}) = 0
mmap(NULL, 79672, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f464f889000
mmap(0x7f464f88c000, 28672, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x3000) = 0x7f464f88c000
mmap(0x7f464f893000, 8192, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xa000) =
0x7f464f893000
mmap(0x7f464f895000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xb000) = 0x7f464f895000
mmap(0x7f464f897000, 22328, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f464f897000
close(3)                                = 0
mprotect(0x7f464f895000, 4096, PROT_READ) = 0
munmap(0x7f465024b000, 68833)           = 0
openat(AT_FDCWD, "/etc/protocols", O_RDONLY|O_CLOEXEC) = 3
lseek(3, 0, SEEK_CUR)                   = 0
fstat(3, {st_mode=S_IFREG|0644, st_size=2932, ...}) = 0
read(3, "# Internet (IP) protocols\n#\n# Up"..., 4096) = 2932
lseek(3, 0, SEEK_CUR)                   = 2932
read(3, "", 4096)                       = 0
close(3)                                = 0
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
```

```
getrandom("\x6a\x60\x7f\x78\xa4\x7d\x85\x3d\x90\x4a\xcf\x22\x34\x7a\xef\x33", 16, 0) = 16
getrandom("\x8c\xc6\xf3\xdf\x9e\x54\x10\x55\x9c\x9e\x8b\x08\xf8\xa7\x50\x81", 16, 0) = 16
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
epoll_create1(EPOLL_CLOEXEC)            = 5
epoll_ctl(5, EPOLL_CTL_ADD, 4, {0, {u32=2071821664, u64=94637381221728}}) = 0
epoll_ctl(5, EPOLL_CTL_MOD, 4, {EPOLLIN, {u32=2071821664, u64=94637381221728}}) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f464f088000
mprotect(0x7f464f089000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f464f887d30, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|
CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[5081], tls=0x7f464f888700, child_tidptr=0x7f464f8889d0) = 5081
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
epoll_create1(EPOLL_CLOEXEC)            = 7
epoll_ctl(7, EPOLL_CTL_ADD, 6, {0, {u32=2071839328, u64=94637381239392}}) = 0
epoll_ctl(7, EPOLL_CTL_MOD, 6, {EPOLLIN, {u32=2071839328, u64=94637381239392}}) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f464e887000
mprotect(0x7f464e888000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f464f086d30, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|
CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[5082], tls=0x7f464f087700, child_tidptr=0x7f464f0879d0) = 5082
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
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
socket(AF_NETLINK, SOCK_RAW|SOCK_CLOEXEC, NETLINK_ROUTE) = 9
bind(9, {sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, 12) = 0
getsockname(9, {sa_family=AF_NETLINK, nl_pid=5080, nl_groups=00000000}, [12]) = 0
sendto(9, {{len=20, type=RTM_GETLINK, flags=NLM_F_REQUEST|NLM_F_DUMP, seq=1608927735, pid=0},
{ifi_family=AF_UNSPEC, ...}}, 20, 0, {sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, 12) = 20
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, msg_namelen=12,
msg_iov=[{iov_base=[{{len=1316, type=RTM_NEWLINK, flags=NLM_F_MULTI, seq=1608927735, pid=5080},
{ifi_family=AF_UNSPEC, ifi_type=ARPHRD_LOOPBACK, ifi_index=if_nametoindex("lo"), ifi_flags=IFF_UP|
IFF_LOOPBACK|IFF_RUNNING|IFF_LOWER_UP, ifi_change=0}, [{{nla_len=7, nla_type=IFLA_IFNAME}, "lo"},
{{nla_len=8, nla_type=IFLA_TXQLEN}, 1000}, {{nla_len=5, nla_type=IFLA_OPERSTATE}, 0}, {{nla_len=5,
nla_type=IFLA_LINKMODE}, 0}, {{nla_len=8, nla_type=IFLA_MTU}, 65536}, {{nla_len=8,
nla_type=IFLA_MIN_MTU}, 0}, {{nla_len=8, nla_type=IFLA_MAX_MTU}, 0}, {{nla_len=8,
nla_type=IFLA_GROUP}, 0}, {{nla_len=8, nla_type=IFLA_PROMISCUITY}, 0}, {{nla_len=8,
nla_type=IFLA_NUM_TX_QUEUES}, 1}, {{nla_len=8, nla_type=IFLA_GSO_MAX_SEGS}, 65535}, {{nla_len=8,
nla_type=IFLA_GSO_MAX_SIZE}, 65536}, {{nla_len=8, nla_type=IFLA_NUM_RX_QUEUES}, 1}, {{nla_len=5,
nla_type=IFLA_CARRIER}, 1}, {{nla_len=12, nla_type=IFLA_QDISC}, "noqueue"}, {{nla_len=8,
nla_type=IFLA_CARRIER_CHANGES}, 0}, {{nla_len=5, nla_type=IFLA_PROTO_DOWN}, 0}, {{nla_len=8,
nla_type=IFLA_CARRIER_UP_COUNT}, 0}, {{nla_len=8, nla_type=IFLA_CARRIER_DOWN_COUNT}, 0},
{{nla_len=36, nla_type=IFLA_MAP}, {mem_start=0, mem_end=0, base_addr=0, irq=0, dma=0, port=0}},
{{nla_len=10, nla_type=IFLA_ADDRESS}, "\x00\x00\x00\x00\x00\x00"}, {{nla_len=10,
nla_type=IFLA_BROADCAST}, "\x00\x00\x00\x00\x00\x00"}, {{nla_len=196, nla_type=IFLA_STATS64},
{rx_packets=168132, tx_packets=168132, rx_bytes=8439716, tx_bytes=8439716, rx_errors=0, tx_errors=0,
rx_dropped=0, tx_dropped=0, multicast=0, collisions=0, rx_length_errors=0, rx_over_errors=0, rx_crc_errors=0,
rx_frame_errors=0, rx_fifo_errors=0, rx_missed_errors=0, tx_aborted_errors=0, tx_carrier_errors=0, tx_fifo_errors=0,
tx_heartbeat_errors=0, tx_window_errors=0, rx_compressed=0, tx_compressed=0, rx_nohandler=0}}, {{nla_len=100,
nla_type=IFLA_STATS}, {rx_packets=168132, tx_packets=168132, rx_bytes=8439716, tx_bytes=8439716,
rx_errors=0, tx_errors=0, rx_dropped=0, tx_dropped=0, multicast=0, collisions=0, rx_length_errors=0,
```

rx_over_errors=0, rx_crc_errors=0, rx_frame_errors=0, rx_fifo_errors=0, rx_missed_errors=0, tx_aborted_errors=0,
tx_carrier_errors=0, tx_fifo_errors=0, tx_heartbeat_errors=0, tx_window_errors=0, rx_compressed=0,
tx_compressed=0, rx_nohandler=0}}, {{nla_len=12, nla_type=IFLA_XDP}, {{nla_len=5,
nla_type=IFLA_XDP_ATTACHED}, XDP_ATTACHED_NONE}}, {{nla_len=760, nla_type=IFLA_AF_SPEC},
[{{nla_len=136, nla_type=AF_INET}, {{nla_len=132, nla_type=IFLA_INET_CONF},
[[IPV4_DEVCONF_FORWARDING-1] = 0, [IPV4_DEVCONF_MC_FORWARDING-1] = 0,
[IPV4_DEVCONF_PROXY_ARP-1] = 0, [IPV4_DEVCONF_ACCEPT_REDIRECTS-1] = 1,
[IPV4_DEVCONF_SECURE_REDIRECTS-1] = 1, [IPV4_DEVCONF_SEND_REDIRECTS-1] = 1,
[IPV4_DEVCONF_SHARED_MEDIA-1] = 1, [IPV4_DEVCONF_RP_FILTER-1] = 0,
[IPV4_DEVCONF_ACCEPT_SOURCE_ROUTE-1] = 1, [IPV4_DEVCONF_BOOTP_RELAY-1] = 0,
[IPV4_DEVCONF_LOG_MARTIANS-1] = 0, [IPV4_DEVCONF_TAG-1] = 0, [IPV4_DEVCONF_ARPFILTER-1] =
0, [IPV4_DEVCONF_MEDIUM_ID-1] = 0, [IPV4_DEVCONF_NOXFRM-1] = 1, [IPV4_DEVCONF_NOPOLICY-1]
= 1, [IPV4_DEVCONF_FORCE_IGMP_VERSION-1] = 0, [IPV4_DEVCONF_ARP_ANNOUNCE-1] = 0,
[IPV4_DEVCONF_ARP_IGNORE-1] = 0, [IPV4_DEVCONF_PROMOTE_SECONDARIES-1] = 1,
[IPV4_DEVCONF_ARP_ACCEPT-1] = 0, [IPV4_DEVCONF_ARP_NOTIFY-1] = 0,
[IPV4_DEVCONF_ACCEPT_LOCAL-1] = 0, [IPV4_DEVCONF_SRC_VMARK-1] = 0,
[IPV4_DEVCONF_PROXY_ARP_PVLAN-1] = 0, [IPV4_DEVCONF_ROUTE_LOCALNET-1] = 0,
[IPV4_DEVCONF_IGMPV2_UNSOLICITED_REPORT_INTERVAL-1] = 10000,
[IPV4_DEVCONF_IGMPV3_UNSOLICITED_REPORT_INTERVAL-1] = 1000,
[IPV4_DEVCONF_IGNORE_ROUTES_WITH_LINKDOWN-1] = 0,
[IPV4_DEVCONF_DROP_UNICAST_IN_L2_MULTICAST-1] = 0,
[IPV4_DEVCONF_DROP_GRATUITOUS_ARP-1] = 0, [IPV4_DEVCONF_BC_FORWARDING-1] = 0]}},
{{nla_len=620, nla_type=AF_INET6}, [{{nla_len=8, nla_type=IFLA_INET6_FLAGS}, IF_READY}, {{nla_len=20,
nla_type=IFLA_INET6_CACHEINFO}, {max_reasm_len=65535, tstamp=160, reachable_time=42812,
retrans_time=1000}}, {{nla_len=208, nla_type=IFLA_INET6_CONF}, [[DEVCONF_FORWARDING] = 0,
[DEVCONF_HOPLIMIT] = 64, [DEVCONF_MTU6] = 65536, [DEVCONF_ACCEPT_RA] = 1,
[DEVCONF_ACCEPT_REDIRECTS] = 1, [DEVCONF_AUTOCONF] = 1, [DEVCONF_DAD_TRANSMITS] = 1,
[DEVCONF_RTR_SOLICITS] = -1, [DEVCONF_RTR_SOLICIT_INTERVAL] = 4000,
[DEVCONF_RTR_SOLICIT_DELAY] = 1000, [DEVCONF_USE_TEMPADDR] = -1,
[DEVCONF_TEMP_VALID_LFT] = 604800, [DEVCONF_TEMP_PREFERED_LFT] = 86400,
[DEVCONF_REGEN_MAX_RETRY] = 3, [DEVCONF_MAX_DESYNC_FACTOR] = 600,
[DEVCONF_MAX_ADDRESSES] = 16, [DEVCONF_FORCE_MLD_VERSION] = 0,
[DEVCONF_ACCEPT_RA_DEFRTR] = 1, [DEVCONF_ACCEPT_RA_PINFO] = 1,
[DEVCONF_ACCEPT_RA_RTR_PREF] = 1, [DEVCONF_RTR_PROBE_INTERVAL] = 60000,
[DEVCONF_ACCEPT_RA_RT_INFO_MAX_PLEN] = 0, [DEVCONF_PROXY_NDP] = 0,
[DEVCONF_OPTIMISTIC_DAD] = 0, [DEVCONF_ACCEPT_SOURCE_ROUTE] = 0,
[DEVCONF_MC_FORWARDING] = 0, [DEVCONF_DISABLE_IPV6] = 0, [DEVCONF_ACCEPT_DAD] = -1,
[DEVCONF_FORCE_TLLAO] = 0, [DEVCONF_NDISC_NOTIFY] = 0,
[DEVCONF_MLDV1_UNSOLICITED_REPORT_INTERVAL] = 10000,
[DEVCONF_MLDV2_UNSOLICITED_REPORT_INTERVAL] = 1000, ...]}, {{nla_len=300,
nla_type=IFLA_INET6_STATS}, [[IPSTATS_MIB_NUM] = 37, [IPSTATS_MIB_INPKTS] = 4,
[IPSTATS_MIB_INOCTETS] = 292, [IPSTATS_MIB_INDELIVERS] = 4,
[IPSTATS_MIB_OUTFORWDATAGRAMS] = 0, [IPSTATS_MIB_OUTPKTS] = 4, [IPSTATS_MIB_OUTOCTETS]
= 292, [IPSTATS_MIB_INHDRERRORS] = 0, [IPSTATS_MIB_INTOOBIGERRORS] = 0,
[IPSTATS_MIB_INNOROUTES] = 0, [IPSTATS_MIB_INADDRERRORS] = 0,
[IPSTATS_MIB_INUNKNOWNPROTOS] = 0, [IPSTATS_MIB_INTRUNCATEDPKTS] = 0,
[IPSTATS_MIB_INDISCARDS] = 0, [IPSTATS_MIB_OUTDISCARDS] = 0, [IPSTATS_MIB_OUTNOROUTES] =
0, [IPSTATS_MIB_REASMTIMEOUT] = 0, [IPSTATS_MIB_REASMREQDS] = 0, [IPSTATS_MIB_REASMOKS] =
0, [IPSTATS_MIB_REASMFAILS] = 0, [IPSTATS_MIB_FRAGOKS] = 0, [IPSTATS_MIB_FRAGFAILS] = 0,
[IPSTATS_MIB_FRAGCREATES] = 0, [IPSTATS_MIB_INMCASTPKTS] = 0, [IPSTATS_MIB_OUTMCASTPKTS]
= 2, [IPSTATS_MIB_INBCASTPKTS] = 0, [IPSTATS_MIB_OUTBCASTPKTS] = 0,
[IPSTATS_MIB_INMCASTOCTETS] = 0, [IPSTATS_MIB_OUTMCASTOCTETS] = 152,
[IPSTATS_MIB_INBCASTOCTETS] = 0, [IPSTATS_MIB_OUTBCASTOCTETS] = 0,
[IPSTATS_MIB_CSUMERRORS] = 0, ...]}, {{nla_len=52, nla_type=IFLA_INET6_ICMP6STATS},
[[ICMP6_MIB_NUM] = 6, [ICMP6_MIB_INMSGS] = 2, [ICMP6_MIB_INERRORS] = 0,
[ICMP6_MIB_OUTMSGS] = 2, [ICMP6_MIB_OUTERRORS] = 0, [ICMP6_MIB_CSUMERRORS] = 0]},
{{nla_len=20, nla_type=IFLA_INET6_TOKEN}, inet_pton(AF_INET6, "::")}, {{nla_len=5,
nla_type=IFLA_INET6_ADDR_GEN_MODE}, IN6_ADDR_GEN_MODE_EUI64}]}]}]}, {{len=1324,
type=RTM_NEWLINK, flags=NLM_F_MULTI, seq=1608927735, pid=5080}, {ifi_family=AF_UNSPEC,
ifi_type=ARPHRD_ETHER, ifi_index=if_nametoindex("enp0s3"), ifi_flags=IFF_UP|IFF_BROADCAST|
IFF_RUNNING|IFF_MULTICAST|IFF_LOWER_UP, ifi_change=0}, [{{nla_len=11, nla_type=IFLA_IFNAME},
"enp0s3"}, {{nla_len=8, nla_type=IFLA_TXQLEN}, 1000}, {{nla_len=5, nla_type=IFLA_OPERSTATE}, 6},
{{nla_len=5, nla_type=IFLA_LINKMODE}, 0}, {{nla_len=8, nla_type=IFLA_MTU}, 1500}, {{nla_len=8,

nla_type=IFLA_MIN_MTU}, 46}, {{nla_len=8, nla_type=IFLA_MAX_MTU}, 16110}, {{nla_len=8,
nla_type=IFLA_GROUP}, 0}, {{nla_len=8, nla_type=IFLA_PROMISCUITY}, 0}, {{nla_len=8,
nla_type=IFLA_NUM_TX_QUEUES}, 1}, {{nla_len=8, nla_type=IFLA_GSO_MAX_SEGS}, 65535}, {{nla_len=8,
nla_type=IFLA_GSO_MAX_SIZE}, 65536}, {{nla_len=8, nla_type=IFLA_NUM_RX_QUEUES}, 1}, {{nla_len=5,
nla_type=IFLA_CARRIER}, 1}, {{nla_len=13, nla_type=IFLA_QDISC}, "fq_codel"}, {{nla_len=8,
nla_type=IFLA_CARRIER_CHANGES}, 8}, {{nla_len=5, nla_type=IFLA_PROTO_DOWN}, 0}, {{nla_len=8,
nla_type=IFLA_CARRIER_UP_COUNT}, 4}, {{nla_len=8, nla_type=IFLA_CARRIER_DOWN_COUNT}, 4},
{{nla_len=36, nla_type=IFLA_MAP}, {mem_start=0, mem_end=0, base_addr=0, irq=0, dma=0, port=0}},
{{nla_len=10, nla_type=IFLA_ADDRESS}, "\x08\x00\x27\x7c\x08\x1d"}, {{nla_len=10,
nla_type=IFLA_BROADCAST}, "\xff\xff\xff\xff\xff\xff"}, {{nla_len=196, nla_type=IFLA_STATS64},
{rx_packets=6364, tx_packets=3302, rx_bytes=5692447, tx_bytes=654659, rx_errors=0, tx_errors=0, rx_dropped=0,
tx_dropped=0, multicast=0, collisions=0, rx_length_errors=0, rx_over_errors=0, rx_crc_errors=0, rx_frame_errors=0,
rx_fifo_errors=0, rx_missed_errors=0, tx_aborted_errors=0, tx_carrier_errors=0, tx_fifo_errors=0,
tx_heartbeat_errors=0, tx_window_errors=0, rx_compressed=0, tx_compressed=0, rx_nohandler=0}}, {{nla_len=100,
nla_type=IFLA_STATS}, {rx_packets=6364, tx_packets=3302, rx_bytes=5692447, tx_bytes=654659, rx_errors=0,
tx_errors=0, rx_dropped=0, tx_dropped=0, multicast=0, collisions=0, rx_length_errors=0, rx_over_errors=0,
rx_crc_errors=0, rx_frame_errors=0, rx_fifo_errors=0, rx_missed_errors=0, tx_aborted_errors=0, tx_carrier_errors=0,
tx_fifo_errors=0, tx_heartbeat_errors=0, tx_window_errors=0, rx_compressed=0, tx_compressed=0, rx_nohandler=0}},
{{nla_len=12, nla_type=IFLA_XDP}, {{nla_len=5, nla_type=IFLA_XDP_ATTACHED},
XDP_ATTACHED_NONE}}, {{nla_len=760, nla_type=IFLA_AF_SPEC}, [{{nla_len=136, nla_type=AF_INET},
{{nla_len=132, nla_type=IFLA_INET_CONF}, [[IPV4_DEVCONF_FORWARDING-1] = 0,
[IPV4_DEVCONF_MC_FORWARDING-1] = 0, [IPV4_DEVCONF_PROXY_ARP-1] = 0,
[IPV4_DEVCONF_ACCEPT_REDIRECTS-1] = 1, [IPV4_DEVCONF_SECURE_REDIRECTS-1] = 1,
[IPV4_DEVCONF_SEND_REDIRECTS-1] = 1, [IPV4_DEVCONF_SHARED_MEDIA-1] = 1,
[IPV4_DEVCONF_RP_FILTER-1] = 2, [IPV4_DEVCONF_ACCEPT_SOURCE_ROUTE-1] = 1,
[IPV4_DEVCONF_BOOTP_RELAY-1] = 0, [IPV4_DEVCONF_LOG_MARTIANS-1] = 0, [IPV4_DEVCONF_TAG-
1] = 0, [IPV4_DEVCONF_ARPFILTER-1] = 0, [IPV4_DEVCONF_MEDIUM_ID-1] = 0,
[IPV4_DEVCONF_NOXFRM-1] = 0, [IPV4_DEVCONF_NOPOLICY-1] = 0,
[IPV4_DEVCONF_FORCE_IGMP_VERSION-1] = 0, [IPV4_DEVCONF_ARP_ANNOUNCE-1] = 0,
[IPV4_DEVCONF_ARP_IGNORE-1] = 0, [IPV4_DEVCONF_PROMOTE_SECONDARIES-1] = 1,
[IPV4_DEVCONF_ARP_ACCEPT-1] = 0, [IPV4_DEVCONF_ARP_NOTIFY-1] = 0,
[IPV4_DEVCONF_ACCEPT_LOCAL-1] = 0, [IPV4_DEVCONF_SRC_VMARK-1] = 0,
[IPV4_DEVCONF_PROXY_ARP_PVLAN-1] = 0, [IPV4_DEVCONF_ROUTE_LOCALNET-1] = 0,
[IPV4_DEVCONF_IGMPV2_UNSOLICITED_REPORT_INTERVAL-1] = 10000,
[IPV4_DEVCONF_IGMPV3_UNSOLICITED_REPORT_INTERVAL-1]
 = 1000, [IPV4_DEVCONF_IGNORE_ROUTES_WITH_LINKDOWN-1] = 0,
[IPV4_DEVCONF_DROP_UNICAST_IN_L2_MULTICAST-1] = 0,
[IPV4_DEVCONF_DROP_GRATUITOUS_ARP-1] = 0, [IPV4_DEVCONF_BC_FORWARDING-1] = 0]}},
{{nla_len=620, nla_type=AF_INET6}, [{{nla_len=8, nla_type=IFLA_INET6_FLAGS}, IF_READY}, {{nla_len=20,
nla_type=IFLA_INET6_CACHEINFO}, {max_reasm_len=65535, tstamp=778, reachable_time=23112,
retrans_time=1000}}, {{nla_len=208, nla_type=IFLA_INET6_CONF}, [[DEVCONF_FORWARDING] = 0,
[DEVCONF_HOPLIMIT] = 64, [DEVCONF_MTU6] = 1500, [DEVCONF_ACCEPT_RA] = 0,
[DEVCONF_ACCEPT_REDIRECTS] = 1, [DEVCONF_AUTOCONF] = 1, [DEVCONF_DAD_TRANSMITS] = 1,
[DEVCONF_RTR_SOLICITS] = -1, [DEVCONF_RTR_SOLICIT_INTERVAL] = 4000,
[DEVCONF_RTR_SOLICIT_DELAY] = 1000, [DEVCONF_USE_TEMPADDR] = 2,
[DEVCONF_TEMP_VALID_LFT] = 604800, [DEVCONF_TEMP_PREFERED_LFT] = 86400,
[DEVCONF_REGEN_MAX_RETRY] = 3, [DEVCONF_MAX_DESYNC_FACTOR] = 600,
[DEVCONF_MAX_ADDRESSES] = 16, [DEVCONF_FORCE_MLD_VERSION] = 0,
[DEVCONF_ACCEPT_RA_DEFRTR] = 1, [DEVCONF_ACCEPT_RA_PINFO] = 1,
[DEVCONF_ACCEPT_RA_RTR_PREF] = 1, [DEVCONF_RTR_PROBE_INTERVAL] = 60000,
[DEVCONF_ACCEPT_RA_RT_INFO_MAX_PLEN] = 0, [DEVCONF_PROXY_NDP] = 0,
[DEVCONF_OPTIMISTIC_DAD] = 0, [DEVCONF_ACCEPT_SOURCE_ROUTE] = 0,
[DEVCONF_MC_FORWARDING] = 0, [DEVCONF_DISABLE_IPV6] = 0, [DEVCONF_ACCEPT_DAD] = 1,
[DEVCONF_FORCE_TLLAO] = 0, [DEVCONF_NDISC_NOTIFY] = 0,
[DEVCONF_MLDV1_UNSOLICITED_REPORT_INTERVAL] = 10000,
[DEVCONF_MLDV2_UNSOLICITED_REPORT_INTERVAL] = 1000, ...]}, {{nla_len=300,
nla_type=IFLA_INET6_STATS}, [[IPSTATS_MIB_NUM] = 37, [IPSTATS_MIB_INPKTS] = 39,
[IPSTATS_MIB_INOCTETS] = 3724, [IPSTATS_MIB_INDELIVERS] = 39,
[IPSTATS_MIB_OUTFORWDATAGRAMS] = 0, [IPSTATS_MIB_OUTPKTS] = 55, [IPSTATS_MIB_OUTOCTETS]
= 5092, [IPSTATS_MIB_INHDRERRORS] = 0, [IPSTATS_MIB_INTOOBIGERRORS] = 0,
[IPSTATS_MIB_INNOROUTES] = 0, [IPSTATS_MIB_INADDRERRORS] = 0,
[IPSTATS_MIB_INUNKNOWNPROTOS] = 0, [IPSTATS_MIB_INTRUNCATEDPKTS] = 0,
[IPSTATS_MIB_INDISCARDS] = 0, [IPSTATS_MIB_OUTDISCARDS] = 0, [IPSTATS_MIB_OUTNOROUTES] =

```
0, [IPSTATS_MIB_REASMTIMEOUT] = 0, [IPSTATS_MIB_REASMREQDS] = 0, [IPSTATS_MIB_REASMOKS] =
0, [IPSTATS_MIB_REASMFAILS] = 0, [IPSTATS_MIB_FRAGOKS] = 0, [IPSTATS_MIB_FRAGFAILS] = 0,
[IPSTATS_MIB_FRAGCREATES] = 0, [IPSTATS_MIB_INMCASTPKTS] = 39,
[IPSTATS_MIB_OUTMCASTPKTS] = 55, [IPSTATS_MIB_INBCASTPKTS] = 0,
[IPSTATS_MIB_OUTBCASTPKTS] = 0, [IPSTATS_MIB_INMCASTOCTETS] = 3724,
[IPSTATS_MIB_OUTMCASTOCTETS] = 5092, [IPSTATS_MIB_INBCASTOCTETS] = 0,
[IPSTATS_MIB_OUTBCASTOCTETS] = 0, [IPSTATS_MIB_CSUMERRORS] = 0, ...]}, {{nla_len=52,
nla_type=IFLA_INET6_ICMP6STATS}, [[ICMP6_MIB_NUM] = 6, [ICMP6_MIB_INMSGS] = 0,
[ICMP6_MIB_INERRORS] = 0, [ICMP6_MIB_OUTMSGS] = 16, [ICMP6_MIB_OUTERRORS] = 0,
[ICMP6_MIB_CSUMERRORS] = 0]}, {{nla_len=20, nla_type=IFLA_INET6_TOKEN}, inet_pton(AF_INET6, "::")},
{{nla_len=5, nla_type=IFLA_INET6_ADDR_GEN_MODE}, IN6_ADDR_GEN_MODE_NONE}]}]}]},
iov_len=4096}], msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 2640
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, msg_namelen=12,
msg_iov=[{iov_base={{len=20, type=NLMSG_DONE, flags=NLM_F_MULTI, seq=1608927735, pid=5080}, 0},
iov_len=4096}], msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 20
sendto(9, {{len=20, type=RTM_GETADDR, flags=NLM_F_REQUEST|NLM_F_DUMP, seq=1608927736, pid=0},
{ifa_family=AF_UNSPEC, ...}}, 20, 0, {sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, 12) = 20
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, msg_namelen=12,
msg_iov=[{iov_base=[{{len=76, type=RTM_NEWADDR, flags=NLM_F_MULTI, seq=1608927736, pid=5080},
{ifa_family=AF_INET, ifa_prefixlen=8, ifa_flags=IFA_F_PERMANENT, ifa_scope=RT_SCOPE_HOST,
ifa_index=if_nametoindex("lo")}, [{{nla_len=8, nla_type=IFA_ADDRESS}, inet_addr("127.0.0.1")}, {{nla_len=8,
nla_type=IFA_LOCAL}, inet_addr("127.0.0.1")}, {{nla_len=7, nla_type=IFA_LABEL}, "lo"}, {{nla_len=8,
nla_type=IFA_FLAGS}, IFA_F_PERMANENT}, {{nla_len=20, nla_type=IFA_CACHEINFO},
{ifa_prefered=4294967295, ifa_valid=4294967295, cstamp=160, tstamp=160}}]}, {{len=88, type=RTM_NEWADDR,
flags=NLM_F_MULTI, seq=1608927736, pid=5080}, {ifa_family=AF_INET, ifa_prefixlen=24, ifa_flags=0,
ifa_scope=RT_SCOPE_UNIVERSE, ifa_index=if_nametoindex("enp0s3")}, [{{nla_len=8,
nla_type=IFA_ADDRESS}, inet_addr("10.0.2.15")}, {{nla_len=8, nla_type=IFA_LOCAL}, inet_addr("10.0.2.15")},
{{nla_len=8, nla_type=IFA_BROADCAST}, inet_addr("10.0.2.255")}, {{nla_len=11, nla_type=IFA_LABEL},
"enp0s3"}, {{nla_len=8, nla_type=IFA_FLAGS}, IFA_F_NOPREFIXROUTE}, {{nla_len=20,
nla_type=IFA_CACHEINFO}, {ifa_prefered=85829, ifa_valid=85829, cstamp=780, tstamp=1566634}}]}],
iov_len=4096}], msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 164
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, msg_namelen=12,
msg_iov=[{iov_base=[{{len=72, type=RTM_NEWADDR, flags=NLM_F_MULTI, seq=1608927736, pid=5080},
{ifa_family=AF_INET6, ifa_prefixlen=128, ifa_flags=IFA_F_PERMANENT, ifa_scope=RT_SCOPE_HOST,
ifa_index=if_nametoindex("lo")}, [{{nla_len=20, nla_type=IFA_ADDRESS}, inet_pton(AF_INET6, "::1")},
{{nla_len=20, nla_type=IFA_CACHEINFO}, {ifa_prefered=4294967295, ifa_valid=4294967295, cstamp=160,
tstamp=160}}, {{nla_len=8, nla_type=IFA_FLAGS}, IFA_F_PERMANENT}]}, {{len=72, type=RTM_NEWADDR,
flags=NLM_F_MULTI, seq=1608927736, pid=5080}, {ifa_family=AF_INET6, ifa_prefixlen=64,
ifa_flags=IFA_F_PERMANENT, ifa_scope=RT_SCOPE_LINK, ifa_index=if_nametoindex("enp0s3")},
[{{nla_len=20, nla_type=IFA_ADDRESS}, inet_pton(AF_INET6, "fe80::18ac:705b:f24b:491c")}, {{nla_len=20,
nla_type=IFA_CACHEINFO}, {ifa_prefered=4294967295, ifa_valid=4294967295, cstamp=779, tstamp=928}},
{{nla_len=8, nla_type=IFA_FLAGS}, IFA_F_PERMANENT|IFA_F_NOPREFIXROUTE}]}], iov_len=4096}],
msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 144
recvmsg(9, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, msg_namelen=12,
msg_iov=[{iov_base={{len=20, type=NLMSG_DONE, flags=NLM_F_MULTI, seq=1608927736, pid=5080}, 0},
iov_len=4096}], msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 20
close(9)                                = 0
socket(AF_INET, SOCK_STREAM|SOCK_CLOEXEC, IPPROTO_TCP) = 9
setsockopt(9, SOL_SOCKET, SO_REUSEADDR, [1], 4) = 0
bind(9, {sa_family=AF_INET, sin_port=htons(8080), sin_addr=inet_addr("127.0.0.1")}, 16) = 0
listen(9, 100)                          = 0
getsockname(9, {sa_family=AF_INET, sin_port=htons(8080), sin_addr=inet_addr("127.0.0.1")}, [128->16]) = 0
getsockname(9, {sa_family=AF_INET, sin_port=htons(8080), sin_addr=inet_addr("127.0.0.1")}, [128->16]) = 0
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
write(8, "\1\0\0\0\0\0\0\0", 8)         = 8
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(1, "> ", 2)                       = 2
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
read(0, "create 1\n", 1024)             = 9
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f464f8a08d0) = 5083
poll([{fd=8, events=POLLIN}], 1, 0)     = 1 ([{fd=8, revents=POLLIN}])
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
```

```
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
poll([{fd=8, events=POLLIN}], 1, 2000)  = 1 ([{fd=8, revents=POLLIN}])
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
poll([{fd=8, events=POLLIN}], 1, -1)    = 1 ([{fd=8, revents=POLLIN}])
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
poll([{fd=8, events=POLLIN}], 1, -1)    = 1 ([{fd=8, revents=POLLIN}])
read(8, "\1\0\0\0\0\0\0\0", 8)          = 8
poll([{fd=8, events=POLLIN}], 1, 0)     = 0 (Timeout)
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
write(1, "OK:5083\n", 8)                = 8
write(1, "> ", 2)                       = 2
read(0, "remove 1\n", 1024)             = 9
kill(5083, SIGTERM)                     = 0
kill(5083, SIGKILL)                     = 0
write(1, "OK\n", 3)                     = 3
write(1, "> ", 2)                       = 2
read(0, 0x56127b7ddf10, 1024)           = ? ERESTARTSYS (To be restarted if SA_RESTART is set)
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_KILLED, si_pid=5083, si_uid=1000, si_status=SIGTERM,
si_utime=0, si_stime=0} ---
read(0, "exit\n", 1024)                 = 5
write(4, "\1\0\0\0\0\0\0\0", 8)         = 8
write(4, "\1\0\0\0\0\0\0\0", 8)         = 8
poll([{fd=3, events=POLLIN}], 1, -1)    = 1 ([{fd=3, revents=POLLIN}])
read(3, "\1\0\0\0\0\0\0\0", 8)          = 8
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
close(7)                                = 0
close(6)                                = 0
close(5)                                = 0
close(4)                                = 0
close(3)                                = 0
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
+++ exited with 0 +++
```

4. *Листинг программы*

```
[leo@pc src]$ cat main.cpp
#include "server.hpp"
#include "tree.hpp"
#include <algorithm>
#include <csignal>
#include <iostream>
#include <set>
#include <string>
#include <unistd.h>
#include <vector>
#include <zmq.hpp>

int main()
{
    zmq::context_t context(1);
    zmq::socket_t mainSocket(context, ZMQ_REQ);

    // ZMQ_SNDTIMEO: Maximum time before a send operation returns with EAGAIN
    mainSocket.setsockopt(ZMQ_SNDTIMEO, 2000);
    // ZMQ_LINGER: linger period determines how long pending messages which have yet to be sent to a
peer shall linger in memory after a socket is closed
    int linger = 0;
    mainSocket.setsockopt(ZMQ_LINGER, &linger, sizeof(linger));
    int port = bindSocket(mainSocket);

    Tree tree;

    int childPid = 0;
    int childId = 0;
    int createNodeId;

    int id;
    char excmd;
```

```
std::string word;
int val;

std::string sendingMsg;
std::string receivedMsg;

std::string cmd;
while (std::cout << "> " && std::cin >> cmd) {
    if (cmd == "create") {
        std::cin >> createNodeId;
        if (childPid == 0) {
            childPid = fork();
            if (childPid == -1) {
                std::cout << "Unable to create first worker node\n";
                childPid = 0;
                exit(1);
            }
            else if (childPid == 0) {
                createNode(createNodeId, port);
            }
            else {
                childId = createNodeId;
                sendMessage(mainSocket, "pid");
                receivedMsg = receiveMessage(mainSocket);
            }
        }
        else {
            std::ostringstream sendingMsgStream;
            sendingMsgStream << "create " << createNodeId;
            sendMessage(mainSocket, sendingMsgStream.str());
            receivedMsg = receiveMessage(mainSocket);
        }

        if (receivedMsg.substr(0, 2) == "OK") {
            tree.insert(createNodeId);
        }

        std::cout << receivedMsg << "\n";
    }
    else if (cmd == "remove") {
        if (childPid == 0) {
            std::cout << "Error: Not found\n";
            continue;
        }
        std::cin >> createNodeId;
        if (createNodeId == childId) {
            kill(childPid, SIGTERM);
            kill(childPid, SIGKILL);
            childId = childPid = 0;
            std::cout << "OK\n";
            tree.erase(createNodeId);
            continue;
        }
        sendingMsg = "remove " + std::to_string(createNodeId);
        sendMessage(mainSocket, sendingMsg);
        receivedMsg = receiveMessage(mainSocket);
        if (receivedMsg.substr(0, 2) == "OK")
            tree.erase(createNodeId);
        std::cout << receivedMsg << "\n";
    }
    else if (cmd == "exec") {
        std::cin >> id >> excmd >> word;
        if (excmd == '+')
            std::cin >> val;

        sendingMsg = "exec " + std::to_string(id);
        sendMessage(mainSocket, sendingMsg);

        receivedMsg = receiveMessage(mainSocket);
        if (receivedMsg == "Node is available") {
            if (excmd == '+') {
                tree.dictInsertWord(id, word, val);
                std::cout << "OK:" << id << std::endl;
            }
            if (excmd == '?') {
                std::cout << "OK:" << id << ": ";
                tree.dictGetWord(id, word);
            }
        }
        else {
```

```
                std::cout << receivedMsg << std::endl;
            }
        }
        else if (cmd == "ping") {
            std::cin >> id;
            std::vector<int> nodesList = tree.getNodesList();
            bool nodeExists = binary_search(nodesList.begin(), nodesList.end(), id);
            if (nodeExists == 0) {
                std::cout << "Error: Not found\n";
            }
            else {
                sendMessage(mainSocket, "exec " + std::to_string(id));
                receivedMsg = receiveMessage(mainSocket);
                std::istringstream is;
                if (receivedMsg.substr(0, 5) == "Error")
                    std::cout << "OK:0\n";
                else
                    std::cout << "OK:1\n";
            }
        }
        else if (cmd == "pingall") {
            std::vector<int> nodesList = tree.getNodesList();
            if (nodesList.empty()) {std::cout << "Error: Tree is empty\n"; continue;}

            sendMessage(mainSocket, "pingall");
            receivedMsg = receiveMessage(mainSocket);
            std::istringstream is;
            if (receivedMsg.substr(0, 5) == "Error")
                is = std::istringstream("");
            else
                is = std::istringstream(receivedMsg);

            std::set<int> receivedNodes;
            int rec_id;
            while (is >> rec_id) {
                receivedNodes.insert(rec_id);
            }

            std::cout << "Received nodes: ";
            for (const int &i : receivedNodes)
                std::cout << i << " ";

            std::cout << "\nNodes list:     ";
            for (const int &i : nodesList)
                std::cout << i << " ";

            std::cout << "\n";
        }
        else if (cmd == "exit") {
            break;
        }
        else {
            std::cout << "Unknown command\n";
        }
    }
    return 0;
}
[leo@pc src]$ cat child.cpp
#include "server.hpp"
#include <csignal>
#include <string>
#include <unistd.h>

int main(int argc, char **argv)
{
    // Ð°Ð¹Ð¸ Ð¸ Ð½Ð¾Ð¼ÐµÑ€ Ð¿Ð¾Ñ€Ñ‚Ð°, Ð° Ð¾-Ñ€Ñ‹ Ð½ÑƒÐ¶Ð½Ñ Ð¿Ð¾Ð´ÐºÐ»ÑŽÑ‡Ð¸Ñ‚ÑŒÑ
    int id = std::stoi(argv[1]);
    int parentPort = std::stoi(argv[2]);

    // Ð¿Ð¾Ð´ÐºÐ»ÑŽÑ‡ÐµÐ½Ð¸Ðµ
    zmq::context_t context(2);
    zmq::socket_t parentSocket(context, ZMQ_REP);

    parentSocket.connect(getPortName(parentPort));

    zmq::socket_t leftSocket(context, ZMQ_REQ);
    zmq::socket_t rightSocket(context, ZMQ_REQ);

    int linger = 0;
    leftSocket.setsockopt(ZMQ_SNDTIMEO, 2000);
```

```
    leftSocket.setsockopt(ZMQ_LINGER, &linger, sizeof(linger));
    rightSocket.setsockopt(ZMQ_SNDTIMEO, 2000);
    rightSocket.setsockopt(ZMQ_LINGER, &linger, sizeof(linger));

    int leftPort = bindSocket(leftSocket);
    int rightPort = bindSocket(rightSocket);

    // Ð²Ñ▫Ð¿Ð¾Ð¼Ð³Ð°Ñ,ÐµÐ»ÑŒÐ½Ñ‹Ðµ Ð¿ÐµÑ€ÐµÐ¼ÐµÐ½Ð½Ñ‹Ðµ
    int leftPid = 0;
    int rightPid = 0;
    int leftId = 0;
    int rightId = 0;

    std::string request;
    std::string cmd;

    while (true) {
        request = receiveMessage(parentSocket);
        std::istringstream cmdStream(request);
        cmdStream >> cmd;
        if (cmd == "id") {
            printf("debug\n");
            std::string parentString = "OK:" + std::to_string(id);
            sendMessage(parentSocket, parentString);
        }
        else if (cmd == "pid") {
            std::string parentString = "OK:" + std::to_string(getpid());
            sendMessage(parentSocket, parentString);
        }
        else if (cmd == "create") {
            int idToCreate;
            cmdStream >> idToCreate;
            // ÑƒÐ¿Ñ€Ð²Ð»Ñ▫ÑŽÑˆÐ¸Ð¹ Ñ▫Ð¾Ð±Ñ‰Ð°Ð» Ð½Ð¾Ð²Ñ‹Ð¹ id Ð½Ð¾Ð²Ð¾Ð³Ð¾ ÑƒÐ·Ð»Ð° Ð¸Ð¿Ð¾Ñ€Ñ‚,, Ðº ÐºÐ¾Ñ€Ð½ÐµÐ²Ð¾Ð¼Ñƒ ÐµÐ³Ð¾ Ð½Ð°Ð´Ð¾ Ð¿Ð¾Ð´ÐºÐ»ÑŽÑ‡Ð¸Ñ,ÑŒ
            if (idToCreate == id) {
                // ÐµÑ▫Ð¸ id Ñ€Ð°Ð²Ð½Ð½ Ð´Ð°Ð½Ð½Ð¾Ð¼Ñƒ, Ð·Ð½Ð°Ñ‡Ð¸Ñ, ÑƒÐ·Ð»Ñ ÑƒÐ¶Ðµ Ñ▫ÑƒÑ‰ÐµÑƒÑ‰ÐµÑ‚Ð²ÑƒÐµÑ‚
                std::string msgString = "Error: Already exists";
                sendMessage(parentSocket, msgString);
            }
            else if (idToCreate < id) {
                if (leftPid == 0) {
                    leftPid = fork();
                    if (leftPid == -1) {
                        sendMessage(parentSocket, "Error: fork fails");
                        leftPid = 0;
                    }
                    else if (leftPid == 0) {
                        createNode(idToCreate, leftPort);
                    }
                    else {
                        leftId = idToCreate;
                        sendMessage(leftSocket, "pid");
                        sendMessage(parentSocket, receiveMessage(leftSocket));
                    }
                }
                else {
                    sendMessage(leftSocket, request);
                    sendMessage(parentSocket, receiveMessage(leftSocket));
                }
            }
            else {
                if (rightPid == 0) {
                    rightPid = fork();
                    if (rightPid == -1) {
                        sendMessage(parentSocket, "Error: fork fails");
                        rightPid = 0;
                    }
                    else if (rightPid == 0) {
                        createNode(idToCreate, rightPort);
                    }
                    else {
                        rightId = idToCreate;
                        sendMessage(rightSocket, "pid");
                        sendMessage(parentSocket, receiveMessage(rightSocket));
                    }
                }
                else {
                    sendMessage(rightSocket, request);
                    sendMessage(parentSocket, receiveMessage(rightSocket));
```

```
            }
        }
    }
    else if (cmd == "remove") {
        int idToDelete;
        cmdStream >> idToDelete;
        if (idToDelete < id) {
            if (leftId == 0) {
                sendMessage(parentSocket, "Error: Node not found");
            }
            else if (leftId == idToDelete) {
                sendMessage(leftSocket, "recursiveKilling");
                receiveMessage(leftSocket);
                kill(leftPid, SIGTERM);
                kill(leftPid, SIGKILL);
                leftId = 0;
                leftPid = 0;
                sendMessage(parentSocket, "OK");
            }
            else {
                sendMessage(leftSocket, request);
                sendMessage(parentSocket, receiveMessage(leftSocket));
            }
        }
        else {
            if (rightId == 0) {
                sendMessage(parentSocket, "Error: Node not found");
            }
            else if (rightId == idToDelete) {
                sendMessage(rightSocket, "recursiveKilling");
                receiveMessage(rightSocket);
                kill(rightPid, SIGTERM);
                kill(rightPid, SIGKILL);
                rightId = 0;
                rightPid = 0;
                sendMessage(parentSocket, "OK");
            }
            else {
                sendMessage(rightSocket, request);
                sendMessage(parentSocket, receiveMessage(rightSocket));
            }
        }
    }
    else if (cmd == "exec") {
        int execNodeId;
        cmdStream >> execNodeId;
        if (execNodeId == id) {
            std::string receiveMessage = "Node is available";
            sendMessage(parentSocket, receiveMessage);
        }
        else if (execNodeId < id) {
            if (leftPid == 0) {
                std::string receiveMessage = "Error:" + std::to_string(execNodeId) + ": Not
found";
                sendMessage(parentSocket, receiveMessage);
            }
            else {
                sendMessage(leftSocket, request);
                sendMessage(parentSocket, receiveMessage(leftSocket));
            }
        }
        else {
            if (rightPid == 0) {
                std::string receiveMessage = "Error:" + std::to_string(execNodeId) + ": Not
found";
                sendMessage(parentSocket, receiveMessage);
            }
            else {
                sendMessage(rightSocket, request);
                sendMessage(parentSocket, receiveMessage(rightSocket));
            }
        }
    }
    else if (cmd == "pingall") {
        std::ostringstream res;
        std::string leftRes;
        std::string rightRes;
        res << id << " ";
        if (leftPid != 0) {
            sendMessage(leftSocket, "pingall");
```

```
                leftRes = receiveMessage(leftSocket);
            }
            if (rightPid != 0) {
                sendMessage(rightSocket, "pingall");
                rightRes = receiveMessage(rightSocket);
            }
            if (!leftRes.empty() && leftRes.substr(0, 5) != "Error") {
                res << leftRes << " ";
            }
            if (!rightRes.empty() && rightRes.substr(0, 5) != "Error") {
                res << rightRes << " ";
            }
            sendMessage(parentSocket, res.str());
        }
        else if (cmd == "recursiveKilling") {
            if (leftPid == 0 && rightPid == 0) {
                sendMessage(parentSocket, "OK");
            }
            else {
                if (leftPid != 0) {
                    sendMessage(leftSocket, "recursiveKilling");
                    receiveMessage(leftSocket);
                    kill(leftPid, SIGTERM);
                    kill(leftPid, SIGKILL);
                }
                if (rightPid != 0) {
                    sendMessage(rightSocket, "recursiveKilling");
                    receiveMessage(rightSocket);
                    kill(rightPid, SIGTERM);
                    kill(rightPid, SIGKILL);
                }
                sendMessage(parentSocket, "OK");
            }
        }
        if (parentPort == 0) {
            break;
        }
    }
}[leo@pc src]$ cat server.hpp
#pragma once

#include <cstdlib>
#include <string>
#include <unistd.h>
#include <zmq.hpp>

// send message to the particular socket
bool sendMessage(zmq::socket_t &socket, const std::string &message_string)
{
    // message size init
    zmq::message_t message(message_string.size());
    // message content init
    memcpy(message.data(), message_string.c_str(), message_string.size());
    return socket.send(message);
}

std::string receiveMessage(zmq::socket_t &socket)
{
    zmq::message_t message;
    int recResult;
    // receiving message from socket
    try {
        recResult = (int)socket.recv(&message);
        if (recResult < 0) {
            perror("socket.recv()");
            exit(1);
        }
    }
    catch (...) {
        recResult = 0;
    }
    // transform to string
    std::string recieved_message((char *)message.data(), message.size());
    if (recieved_message.empty() || !recResult) {
        return "Error: Node is not available";
    }
    return recieved_message;
}

std::string getPortName(int port)
```

```
{
    return "tcp://127.0.0.1:" + std::to_string(port);
}
int bindSocket(zmq::socket_t &socket)
{
    int port = 8080;
    // create endpoint and bind it to the socket
    while (true) {
        try {
            socket.bind(getPortName(port));
            break;
        }
        catch (...) {
            port++;
        }
    }
    return port;
}

void createNode(int id, int port)
{
    // new node process
    execl("./child", "child", std::to_string(id).c_str(), std::to_string(port).c_str(), NULL);
}[leo@pc src]$ cat tree.hpp
#pragma once

#include <iostream>
#include <vector>
#include <unordered_map>

class Tree {
private:
    struct Node;

public:
    Tree() = default;

    ~Tree()
    {
        deleteTree(root);
    }

    bool find(const int &id)
    {
        Node *temp = root;
        while (temp != nullptr) {
            if (temp->id == id)
                break;
            if (id > temp->id)
                temp = temp->right;
            if (id < temp->id)
                temp = temp->left;
        }
        return temp != nullptr;
    }

    void insert(int id)
    {
        if (root == nullptr) {
            root = new Node(id);
            return;
        }
        Node *temp = root;
        while (temp != nullptr) {
            if (id == temp->id)
                break;
            if (id < temp->id) {
                if (temp->left == nullptr) {
                    temp->left = new Node(id);
                    break;
                }
                temp = temp->left;
            }
            if (id > temp->id) {
                if (temp->right == nullptr) {
                    temp->right = new Node(id);
                    break;
                }
                temp = temp->right;
            }
```

```
        }
    }

    void erase(int id)
    {
        Node *prev_id = nullptr;
        Node *temp = root;
        while (temp != nullptr) {
            if (id == temp->id) {
                if (prev_id == nullptr) {
                    root = nullptr;
                }
                else {
                    if (prev_id->left == temp)
                        prev_id->left = nullptr;
                    else
                        prev_id->right = nullptr;
                }
                deleteTree(temp);
            }
            else if (id < temp->id) {
                prev_id = temp;
                temp = temp->left;
            }
            else if (id > temp->id) {
                prev_id = temp;
                temp = temp->right;
            }
        }
    }

    std::vector<int> getNodesList() const
    {
        std::vector<int> result;
        getNodesList(root, result);
        return result;
    }

    void dictInsertWord(int id, std::string word, int value)
    {
        Node *node = getNodeById(root, id);
        node->dictionary[word] = value;
    }

    void dictGetWord(int id, std::string word)
    {
        Node *node = getNodeById(root, id);
        if (node->dictionary.find(word) == node->dictionary.end())
            std::cout << "'" << word << "' not found" << '\n';
        else
            std::cout << node->dictionary[word] << '\n';
    }

private:
    struct Node {
        Node(int id) : id(id) {}
        int id = 0;
        Node *left = nullptr;
        Node *right = nullptr;
        std::unordered_map<std::string, int> dictionary;
    };

    Node *root = nullptr;

    Node *getNodeById(Node *root, int id)
    {
        if (root == nullptr || root->id == id) {
            return root;
        }

        if (root->id < id) {
            return getNodeById(root->right, id);
        }

        return getNodeById(root->left, id);
    }

    void getNodesList(Node *node, std::vector<int> &v) const
    {
        if (node == nullptr)
```

```
            return;
        getNodesList(node->left, v);
        v.push_back(node->id);
        getNodesList(node->right, v);
    }

    void deleteTree(Node *node)
    {
        if (node == nullptr)
            return;
        deleteTree(node->left);
        deleteTree(node->right);
        delete node;
    }
};
```

5. Выводы

По мере выполнения данной лабораторной работы я освоил большое количество навыков: в первую очередь, научился работать с очередями сообщений. На примере библиотеки ZeroMQ я описал взаимодействие между двумя разными программами с помощью неё.

Также я описал бинарное дерево с поддержкой различных операций, в том числе — хранение в узлах дерева словаря. Это было нетрудно, благо, в STL существует контейнер unordered_map, позволяющий производить все необходимые для работы со словарём операции быстро и эффективно.

Помимо всего этого, удалось реализовать два вида проверки доступности узлов — ping id и pingall. Первый было поставлено реализовать по заданию, а второй — для удобного тестирования программы.

ZeroMQ — прекрасная технология, мне понравилось с ней работать, ибо её API кроток и понятен. Функции описываются на довольно-таки «высокоуровневом» языке. Плюс ко всему есть «обертка», более высокоуровневая библиотека cppzmq, сокращающая объем кода в несколько раз. Также радует, что библиотека является кроссплатформенной.