**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ  
ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)**

**ЛАБОРАТОРНАЯ РАБОТА №6-8**

**по курсу операционные системы I семестр, 2021/22  
уч. год**

Студент: Белоусов Егор Владимирович

Группа: М8О-207Б-20

Преподаватель: Миронов Е. С.

Вариант: 6

Дата:

Оценка:

**Задание**  
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:

Вариант 6. Вычислительные узлы находятся в идеально сбалансированном дереве. Каждый следующий узел добавляется в самое наименьшее поддерево.

Команда timer для каждого вычислительного узла.

Команда проверки pingall, проверяет, какие узлы живы, а какие – нет.

**Описание решения**

В каждом вычислительном узле находится своя очередь сообщений. Когда управляющий узел получает команду сделать что-либо, он отправляет команду вниз, если принимающий узел, оказывается не тем, кто нужен, иначе отправляет ответ, но уже по-другому порту. Так как вечно ждать ответа мы не можем, для каждого сообщения установлен timeout в 2 секунды.

**Пример работы**

*Enter a command!*

*create 1*

*Enter a command!*

*Ok: 1142*

*create 2*

*Enter a command!*

*Ok:1169*

*create 3*

*Enter a command!*

*Ok:1232*

*exec 3 time*

*Enter a command!*

*Ok:3:0*

*exec 3 start*

*Enter a command!*

*Ok:3*

*exec 3 time*

*Enter a command!*

*Ok:3:5124*

*exec 3 stop*

*Enter a command!*

*Ok:3*

*exec 3 time*

*Enter a command!*

*Ok:3:12381*

*exec 3 time*

*Enter a command!*

*Ok:3:12381*

*pingall*

*Enter a command!*

*Ok:-1*

**Strace**

*egorb@LAPTOP-TIFMO0IA:/mnt/d/os/lab6$ strace ./main*

*execve("./main", ["./main"], 0x7ffc056601f0 /\* 29 vars \*/) = 0*

*brk(NULL) = 0x5572d7c74000*

*arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fff0948af10) = -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=45305, ...}) = 0*

*mmap(NULL, 45305, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f5c17d49000*

*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\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) = 0x7f5c17d47000*

*mmap(NULL, 678128, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5c17ca1000*

*mmap(0x7f5c17cb7000, 430080, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x16000) = 0x7f5c17cb7000*

*mmap(0x7f5c17d20000, 126976, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x7f000) = 0x7f5c17d20000*

*mmap(0x7f5c17d3f000, 32768, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x9d000) = 0x7f5c17d3f000*

*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`\341\t\0\0\0\0\0"..., 832) = 832*

*fstat(3, {st\_mode=S\_IFREG|0644, st\_size=1956992, ...}) = 0*

*mmap(NULL, 1972224, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5c17abf000*

*mprotect(0x7f5c17b55000, 1290240, PROT\_NONE) = 0*

*mmap(0x7f5c17b55000, 987136, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x96000) = 0x7f5c17b55000*

*mmap(0x7f5c17c46000, 299008, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x187000) = 0x7f5c17c46000*

*mmap(0x7f5c17c90000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d0000) = 0x7f5c17c90000*

*mmap(0x7f5c17c9e000, 10240, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f5c17c9e000*

*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) = 0x7f5c17aa4000*

*mmap(0x7f5c17aa7000, 73728, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f5c17aa7000*

*mmap(0x7f5c17ab9000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x15000) = 0x7f5c17ab9000*

*mmap(0x7f5c17abd000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x18000) = 0x7f5c17abd000*

*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\0\345Ga\367\265T\320\374\301V)Yf]\223\337"..., 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\0\345Ga\367\265T\320\374\301V)Yf]\223\337"..., 68, 824) = 68*

*mmap(NULL, 140408, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5c17a81000*

*mmap(0x7f5c17a88000, 69632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x7000) = 0x7f5c17a88000*

*mmap(0x7f5c17a99000, 20480, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x18000) = 0x7f5c17a99000*

*mmap(0x7f5c17a9e000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1c000) = 0x7f5c17a9e000*

*mmap(0x7f5c17aa0000, 13432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f5c17aa0000*

*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\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 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\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68*

*mmap(NULL, 2036952, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5c1788f000*

*mprotect(0x7f5c178b4000, 1847296, PROT\_NONE) = 0*

*mmap(0x7f5c178b4000, 1540096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25000) = 0x7f5c178b4000*

*mmap(0x7f5c17a2c000, 303104, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19d000) = 0x7f5c17a2c000*

*mmap(0x7f5c17a77000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f5c17a77000*

*mmap(0x7f5c17a7d000, 13528, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f5c17a7d000*

*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) = 0x7f5c17837000*

*mmap(0x7f5c17843000, 229376, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xc000) = 0x7f5c17843000*

*mmap(0x7f5c1787b000, 73728, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x44000) = 0x7f5c1787b000*

*mmap(0x7f5c1788d000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x55000) = 0x7f5c1788d000*

*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, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f5c17835000*

*mmap(NULL, 321584, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5c177e6000*

*mmap(0x7f5c177ea000, 163840, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f5c177ea000*

*mmap(0x7f5c17812000, 118784, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2c000) = 0x7f5c17812000*

*mmap(0x7f5c1782f000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x48000) = 0x7f5c1782f000*

*mmap(0x7f5c17831000, 14384, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f5c17831000*

*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\0\257\0\0\0\0\0\0"..., 832) = 832*

*fstat(3, {st\_mode=S\_IFREG|0644, st\_size=690344, ...}) = 0*

*mmap(NULL, 1420000, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5c1768b000*

*mmap(0x7f5c17695000, 421888, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xa000) = 0x7f5c17695000*

*mmap(0x7f5c176fc000, 217088, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x71000) = 0x7f5c176fc000*

*mmap(0x7f5c17731000, 16384, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xa5000) = 0x7f5c17731000*

*mmap(0x7f5c17735000, 723680, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f5c17735000*

*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) = 0x7f5c1763e000*

*mmap(0x7f5c17649000, 204800, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xb000) = 0x7f5c17649000*

*mmap(0x7f5c1767b000, 49152, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3d000) = 0x7f5c1767b000*

*mmap(0x7f5c17687000, 16384, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x48000) = 0x7f5c17687000*

*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) = 0x7f5c174ef000*

*mmap(0x7f5c174fe000, 684032, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xf000) = 0x7f5c174fe000*

*mmap(0x7f5c175a5000, 618496, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xb6000) = 0x7f5c175a5000*

*mmap(0x7f5c1763c000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x14c000) = 0x7f5c1763c000*

*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) = 0x7f5c17412000*

*mprotect(0x7f5c17434000, 700416, PROT\_NONE) = 0*

*mmap(0x7f5c17434000, 397312, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x22000) = 0x7f5c17434000*

*mmap(0x7f5c17495000, 299008, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x83000) = 0x7f5c17495000*

*mmap(0x7f5c174df000, 65536, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xcc000) = 0x7f5c174df000*

*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) = 0x7f5c17410000*

*mmap(NULL, 196696, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5c173df000*

*mprotect(0x7f5c173e3000, 172032, PROT\_NONE) = 0*

*mmap(0x7f5c173e3000, 114688, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f5c173e3000*

*mmap(0x7f5c173ff000, 53248, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x20000) = 0x7f5c173ff000*

*mmap(0x7f5c1740d000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2d000) = 0x7f5c1740d000*

*mmap(0x7f5c1740f000, 88, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f5c1740f000*

*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) = 0x7f5c173d8000*

*mmap(0x7f5c173da000, 8192, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2000) = 0x7f5c173da000*

*mmap(0x7f5c173dc000, 4096, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f5c173dc000*

*mmap(0x7f5c173dd000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f5c173dd000*

*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) = 0x7f5c173c9000*

*mmap(0x7f5c173cc000, 28672, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f5c173cc000*

*mmap(0x7f5c173d3000, 12288, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xa000) = 0x7f5c173d3000*

*mmap(0x7f5c173d6000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xc000) = 0x7f5c173d6000*

*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) = 0x7f5c173c2000*

*mmap(0x7f5c173c4000, 8192, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2000) = 0x7f5c173c4000*

*mmap(0x7f5c173c6000, 4096, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f5c173c6000*

*mmap(0x7f5c173c7000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f5c173c7000*

*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) = 0x7f5c173a6000*

*mprotect(0x7f5c173aa000, 81920, PROT\_NONE) = 0*

*mmap(0x7f5c173aa000, 65536, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f5c173aa000*

*mmap(0x7f5c173ba000, 12288, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x14000) = 0x7f5c173ba000*

*mmap(0x7f5c173be000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x17000) = 0x7f5c173be000*

*mmap(0x7f5c173c0000, 6784, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f5c173c0000*

*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) = 0x7f5c173a4000*

*mmap(NULL, 20752, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5c1739e000*

*mmap(0x7f5c1739f000, 8192, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1000) = 0x7f5c1739f000*

*mmap(0x7f5c173a1000, 4096, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f5c173a1000*

*mmap(0x7f5c173a2000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f5c173a2000*

*close(3) = 0*

*mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f5c1739c000*

*arch\_prctl(ARCH\_SET\_FS, 0x7f5c1739d180) = 0*

*mprotect(0x7f5c17a77000, 12288, PROT\_READ) = 0*

*mprotect(0x7f5c173a2000, 4096, PROT\_READ) = 0*

*mprotect(0x7f5c173be000, 4096, PROT\_READ) = 0*

*mprotect(0x7f5c173c7000, 4096, PROT\_READ) = 0*

*mprotect(0x7f5c173d6000, 4096, PROT\_READ) = 0*

*mprotect(0x7f5c17a9e000, 4096, PROT\_READ) = 0*

*mprotect(0x7f5c173dd000, 4096, PROT\_READ) = 0*

*mprotect(0x7f5c1740d000, 4096, PROT\_READ) = 0*

*mprotect(0x7f5c174df000, 57344, PROT\_READ) = 0*

*mprotect(0x7f5c1763c000, 4096, PROT\_READ) = 0*

*mprotect(0x7f5c17687000, 8192, PROT\_READ) = 0*

*mprotect(0x7f5c17abd000, 4096, PROT\_READ) = 0*

*mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f5c1739a000*

*mprotect(0x7f5c17c90000, 45056, PROT\_READ) = 0*

*mprotect(0x7f5c17731000, 12288, PROT\_READ) = 0*

*mprotect(0x7f5c1782f000, 4096, PROT\_READ) = 0*

*mprotect(0x7f5c1788d000, 4096, PROT\_READ) = 0*

*mprotect(0x7f5c17d3f000, 28672, PROT\_READ) = 0*

*mprotect(0x5572d7913000, 4096, PROT\_READ) = 0*

*mprotect(0x7f5c17d82000, 4096, PROT\_READ) = 0*

*munmap(0x7f5c17d49000, 45305) = 0*

*set\_tid\_address(0x7f5c1739d450) = 1782*

*set\_robust\_list(0x7f5c1739d460, 24) = 0*

*rt\_sigaction(SIGRTMIN, {sa\_handler=0x7f5c17a88bf0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_SIGINFO, sa\_restorer=0x7f5c17a963c0}, NULL, 8) = 0*

*rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f5c17a88c90, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f5c17a963c0}, 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) = 0x5572d7c74000*

*brk(0x5572d7c95000) = 0x5572d7c95000*

*futex(0x7f5c17c9e6bc, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0*

*futex(0x7f5c17c9e6c8, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0*

*fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x3), ...}) = 0*

*write(1, "Enter a command!\n", 17Enter a command!*

*) = 17*

*fstat(0, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x3), ...}) = 0*

*read(0, create 2*

*"create 2\n", 1024) = 9*

*mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f5c16b99000*

*mprotect(0x7f5c16b9a000, 8388608, PROT\_READ|PROT\_WRITE) = 0*

*clone(child\_stack=0x7f5c17398d30, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[1786], tls=0x7f5c17399700, child\_tidptr=0x7f5c173999d0) = 1786*

*write(1, "Ok: Enter a command!\n", 21Ok: Enter a command!*

*) = 21*

*futex(0x7f5c17a7d4c0, FUTEX\_WAKE\_PRIVATE, 11787*

*) = 1*

*read(0, create 1*

*"create 1\n", 1024) = 9*

*clone(child\_stack=0x7f5c17398d30, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[1829], tls=0x7f5c17399700, child\_tidptr=0x7f5c173999d0) = 1829*

*write(1, "Enter a command!\n", 17Enter a command!*

*) = 17*

*read(0, Ok:1853*

*pingall*

*"pingall\n", 1024) = 8*

*clone(child\_stack=0x7f5c17398d30, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[1907], tls=0x7f5c17399700, child\_tidptr=0x7f5c173999d0) = 1907*

*write(1, "Enter a command!\n", 17Enter a command!*

*) = 17*

*read(0, Ok:-1*

*exec 2 time*

*"exec 2 time\n", 1024) = 12*

*clone(child\_stack=0x7f5c17398d30, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[1970], tls=0x7f5c17399700, child\_tidptr=0x7f5c173999d0) = 1970*

*write(1, "Enter a command!\n", 17Enter a command!*

*) = 17*

*read(0, Ok:2:0*

*exec 2 start*

*"exec 2 start\n", 1024) = 13*

*clone(child\_stack=0x7f5c17398d30, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[2024], tls=0x7f5c17399700, child\_tidptr=0x7f5c173999d0) = 2024*

*write(1, "Enter a command!\n", 17Enter a command!*

*) = 17*

*read(0, Ok:2*

*0x5572d7c86560, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)*

*--- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---*

*read(0,*

**Листинг**

**message\_data.h**

#ifndef MESSAGE\_DATA

#define MESSAGE\_DATA

#include "string"

#include <chrono>

#include <time.h>

using namespace std;

int timeToWait = 1000 \* 100 \* 3;

int timeout = 2;

long long now(){

    return chrono::steady\_clock::now().time\_since\_epoch().count();

}

enum MessageType {

    CREATE,

    PING\_ALL,

    TIMER\_GET,

    TIMER\_START,

    TIMER\_STOP,

    GET\_DEPTH,

    TRASH

};

enum ForwardType{

    FEEDBACK,

    QUESTION

};

struct MessageData{

    int needId;

    int messageNumber;

    int initialPort;

    long long result;

    int bestDepth;

    int newID;

    int where;

    MessageType msgType;

    ForwardType type;

};

#endif // MESSAGE\_DATA

**main.cpp**

#include "iostream"

#include "string"

#include "unistd.h"

#include "./message\_data.h"

#include <sys/wait.h>

#include "memory.h"

#include <zmq.h>

#include "map"

#include "chrono"

#include "node.h"

#include <random>

#include "thread"

#include "algorithm"

#include "mutex"

using namespace std;

map<int, int> doesExist;

int initialNode = -1;

mutex mtx;

char \*getStringPid() {

    int id\_pid = getpid();

    char \*str\_pid = const\_cast<char \*>(to\_string(id\_pid).c\_str());

    return str\_pid;

}

void sendMessage(MessageData md){

    if(md.msgType != CREATE && initialNode == -1){

        cout << "Error:" << md.needId << " can't be accessed!\n";

        return;

    }

    if(md.msgType == CREATE){

        mtx.lock();

    }

    if(md.msgType == CREATE && initialNode == -1){

        int pid = fork();

        if(!pid){

            execl("./child", "./child", "-1", to\_string(md.newID).c\_str(), NULL);

            cout << "Bad fork!\n";

        }

        doesExist[md.newID] = 1;

        initialNode = pid \* 10;

        cout << "Ok: " << pid << endl;

        mtx.unlock();

        return;

    }

    map<int, int> have;

    md.initialPort = getpid() + md.messageNumber + 1000;

    md.result = -1;

    void\* context = zmq\_ctx\_new();

    void\* senderSocket = zmq\_socket(context, ZMQ\_PUSH);

    string senderTCP = "tcp://127.0.0.1:" + to\_string(initialNode);

    zmq\_bind(senderSocket, senderTCP.c\_str());

    string objectTCP = "tcp://127.0.0.1:" + to\_string(md.initialPort);

    void\* objectSocket = zmq\_socket(context, ZMQ\_PULL);

    zmq\_connect(objectSocket, objectTCP.c\_str());

    usleep(timeToWait);

    if(md.msgType == CREATE){

        vector<pair<int,int>> bestDepth;

        md.needId = -1;

        MessageData MsgDepth;

        MsgDepth.msgType = GET\_DEPTH;

        MsgDepth.type = QUESTION;

        MsgDepth.messageNumber = md.messageNumber;

        MsgDepth.initialPort = md.initialPort;

        zmq\_msg\_t msg;

        zmq\_msg\_init\_size(&msg, sizeof(MessageData));

        memcpy(zmq\_msg\_data(&msg), &MsgDepth, sizeof(MessageData));

        int snd = zmq\_msg\_send(&msg, senderSocket, ZMQ\_DONTWAIT);

        zmq\_msg\_close(&msg);

        usleep(timeToWait);

        long long wt = now();

        while(1){

            long long tm = now();

            if((tm - wt)/(long long)(1e9) >= timeout)break;

            zmq\_msg\_t ms;

            zmq\_msg\_init(&ms);

            int status = zmq\_msg\_recv(&ms, objectSocket, ZMQ\_DONTWAIT);

            if(status < 0){

                zmq\_msg\_close(&ms);

                continue;

            }

            MessageData \*m = (MessageData \*)zmq\_msg\_data(&ms);

            bestDepth.push\_back({m->result, m->where});

            zmq\_msg\_close(&ms);

        }

        if(bestDepth.empty()){

            md.needId = -1;

        }

        else{

            sort(bestDepth.begin(),bestDepth.end());

            md.needId = bestDepth[0].second;

        }

    }

    if(md.needId == -1 && md.msgType == CREATE){

        int pid = fork();

        if(!pid){

            execl("./child", "./child", "-1", to\_string(md.newID).c\_str(), NULL);

            cout << "Bad fork!\n";

        }

        doesExist[md.newID] = 1;

        initialNode = pid \* 10;

        cout << "Ok: " << pid << endl;

        zmq\_close(objectSocket);

        zmq\_close(senderSocket);

        zmq\_ctx\_destroy(context);

        mtx.unlock();

        return;

    }

    zmq\_msg\_t message;

    zmq\_msg\_init\_size(&message, sizeof(MessageData));

    memcpy(zmq\_msg\_data(&message), &md, sizeof(MessageData));

    int snd = zmq\_msg\_send(&message, senderSocket, ZMQ\_DONTWAIT);

    zmq\_msg\_close(&message);

    long long startWaiting = now();

    bool ok = false;

    if(md.msgType == PING\_ALL){

        ok = true;

    }

    while(1){

        long long tm = now();

        if((tm - startWaiting)/(long long)(1e9) >= timeout){

            break;

        }

        zmq\_msg\_t msg;

        zmq\_msg\_init(&msg);

        int status = zmq\_msg\_recv(&msg, objectSocket, ZMQ\_DONTWAIT);

        if(status >= 0){

            ok = true;

            MessageData\* it =(MessageData\*)zmq\_msg\_data(&msg);

            md = \*it;

            if(md.msgType == PING\_ALL){

                have[it->where] = 1;

            }

            zmq\_msg\_close(&msg);

            continue;

        }

        zmq\_msg\_close(&msg);

    }

    if(!ok){

        cout << "Error:" << md.needId << " can't be accessed!" << endl;

        zmq\_close(objectSocket);

        zmq\_close(senderSocket);

        zmq\_ctx\_destroy(context);

        return;

    }

    vector<int> bad;

    switch (md.msgType)

    {

        case CREATE:

        doesExist[md.newID] = 1;

        cout << "Ok:" << md.result << endl;

        mtx.unlock();

        break;

        case PING\_ALL:

        for(auto &p : doesExist){

            if(have.count(p.first) == 0){

                bad.push\_back(p.first);

            }

        }

        cout << "Ok:";

        if(bad.empty()){

            cout << "-1" << endl;

        }

        else {

            for(int i = 0; i < bad.size(); ++i){

                cout << bad[i];

                if(i + 1 < bad.size())cout << ";";

            }

            cout << endl;

        }

        break;

        case TIMER\_START:

        cout << "Ok:" << md.needId << endl;

        break;

        case TIMER\_STOP:

        cout << "Ok:" << md.needId << endl;

        break;

        case TIMER\_GET:

        cout << "Ok:" << md.needId << ":" << md.result << endl;

        break;

        default:

        cout << "Error:something strange happened" << endl;

        break;

    }

    zmq\_close(objectSocket);

    zmq\_close(senderSocket);

    zmq\_ctx\_destroy(context);

}

int main(int argc, char \*argv[]) {

    string cmd;

    cout << "Enter a command!" << endl;

    int cnt = 0;

    while (cin >> cmd) {

        if(cmd == "create"){

            int id, parent;

            cin >> id;

            MessageData msg;

            msg.needId = parent;

            msg.newID = id;

            msg.msgType = CREATE;

            msg.type = QUESTION;

            msg.messageNumber = cnt;

            thread th(sendMessage, msg);

            th.detach();

        }

        else if (cmd == "exec"){

            int id;

            string type;

            cin >> id >> type;

            MessageData msg;

            msg.needId = id;

            msg.type = QUESTION;

            msg.newID = -1;

            msg.messageNumber = cnt;

            if(type == "time"){

                msg.msgType = TIMER\_GET;

            }

            else if(type == "start"){

                msg.msgType = TIMER\_START;

            }

            else if(type == "stop"){

                msg.msgType = TIMER\_STOP;

            }

            else{

                cout << "Wrong request!" << endl;

                continue;

            }

            thread th(sendMessage, msg);

            th.detach();

        }

        else if(cmd == "pingall"){

            MessageData msg;

            msg.needId = 0;

            msg.messageNumber = cnt;

            msg.newID = -1;

            msg.type = QUESTION;

            msg.msgType = PING\_ALL;

            thread th(sendMessage, msg);

            th.detach();

        }

        ++cnt;

        cout << "Enter a command!" << endl;

    }

    int status;

    if (wait(&status) == -1) {

        cout << "An error occured!\n";

    }

    return 0;

}

**child.cpp**

#include "iostream"

#include "string"

#include "unistd.h"

#include "./message\_data.h"

#include <sys/wait.h>

#include "memory.h"

#include <zmq.h>

#include "chrono"

#include <random>

#include "thread"

#include "mutex"

using namespace std;

int parent;

int pd;

int id;

int leftPid = -1;

int rightPid = -1;

long long timer;

bool isTimerActive;

long long lastActive;

mutex mtx;

 void stop(){

        if(isTimerActive){

            timer += now() - lastActive;

        }

        lastActive = now();

        isTimerActive = false;

    }

void start(){

    if(isTimerActive)return;

    isTimerActive = true;

    lastActive = now();

}

void update(){

    if(isTimerActive){

        timer += now() - lastActive;

    }

    lastActive = now();

}

void sendMessage(MessageData md, int where = -1){

    mtx.lock();

    if(md.type == FEEDBACK){

        if(parent == -1){

            void\* context = zmq\_ctx\_new();

            void\* senderSocker = zmq\_socket(context, ZMQ\_PUSH);

            string senderTCP = "tcp://127.0.0.1:" + to\_string(md.initialPort);

            zmq\_bind(senderSocker, senderTCP.c\_str());

            usleep(timeToWait);

            zmq\_msg\_t message;

            usleep(timeToWait);

            zmq\_msg\_init\_size(&message, sizeof(MessageData));

            memcpy(zmq\_msg\_data(&message), &md, sizeof(MessageData));

            zmq\_msg\_send(&message, senderSocker, ZMQ\_DONTWAIT);

            zmq\_close(senderSocker);

            zmq\_ctx\_destroy(context);

        }

        else{

            void \* context = zmq\_ctx\_new();

            void\* senderSocker = zmq\_socket(context, ZMQ\_PUSH);

            string senderTCP = "tcp://127.0.0.1:" + to\_string(parent);

            zmq\_bind(senderSocker, senderTCP.c\_str());

            usleep(timeToWait);

            zmq\_msg\_t message;

            zmq\_msg\_init\_size(&message, sizeof(MessageData));

            memcpy(zmq\_msg\_data(&message), &md, sizeof(MessageData));

            zmq\_msg\_send(&message, senderSocker, ZMQ\_DONTWAIT);

            zmq\_close(senderSocker);

            zmq\_ctx\_destroy(context);

        }

    }

    else{

        void \* context = zmq\_ctx\_new();

        void\* senderSocker = zmq\_socket(context, ZMQ\_PUSH);

        string senderTCP = "tcp://127.0.0.1:" + to\_string(where);

        zmq\_bind(senderSocker, senderTCP.c\_str());

        usleep(timeToWait);

        zmq\_msg\_t message;

        zmq\_msg\_init\_size(&message, sizeof(MessageData));

        memcpy(zmq\_msg\_data(&message), &md, sizeof(MessageData));

        zmq\_msg\_send(&message, senderSocker, ZMQ\_DONTWAIT);

        zmq\_close(senderSocker);

        zmq\_ctx\_destroy(context);

    }

    mtx.unlock();

}

int main(int argc, char \*argv[]){

    void \* context = zmq\_ctx\_new();

    void \* senderSocket = zmq\_socket(context, ZMQ\_PULL);

    lastActive = now();

    pd = getpid() \* 10;

    parent = stoi(argv[1]);

    id = stoi(argv[2]);

    string senderTCP = "tcp://127.0.0.1:" + to\_string(pd);

    zmq\_connect(senderSocket, senderTCP.c\_str());

    senderTCP = "tcp://127.0.0.1:" + to\_string(pd + 1);

    zmq\_connect(senderSocket, senderTCP.c\_str());

    senderTCP = "tcp://127.0.0.1:" + to\_string(pd + 2);

    zmq\_connect(senderSocket, senderTCP.c\_str());

    usleep(timeToWait);

    while(1){

        zmq\_msg\_t msg;

        zmq\_msg\_init(&msg);

        int status = zmq\_msg\_recv(&msg, senderSocket, ZMQ\_DONTWAIT);

        if(status < 0){

            continue;

        }

        MessageData \*m = (MessageData \*) zmq\_msg\_data(&msg);

        MessageData md = \*m;

        zmq\_msg\_close(&msg);

        if((md.needId == id || md.msgType == GET\_DEPTH || md.msgType == PING\_ALL) && md.type == QUESTION){

            if(md.msgType == CREATE){

                int pid;

                if(leftPid == -1){

                    pid = fork();

                    if(!pid){

                        execl("./child", "./child", to\_string(pd + 1).c\_str(), to\_string(m->newID).c\_str(), NULL);

                        cout << "Bad fork!\n";

                    }

                    leftPid = pid;

                }

                else{

                    pid = fork();

                    if(!pid){

                        execl("./child", "./child", to\_string(pd + 2).c\_str(), to\_string(m->newID).c\_str(), NULL);

                        cout << "Bad fork!\n";

                    }

                    rightPid = pid;

                }

                md.result = pid;

            }

            else if(md.msgType == TIMER\_START){

                start();

            }

            else if(md.msgType == TIMER\_STOP){

                stop();

            }

            else if(md.msgType == TIMER\_GET){

                update();

                md.result = timer / (long long)(1e6);

            }

            bool ok = true;

            if(md.msgType == GET\_DEPTH){

                md.result++;

                if(leftPid != -1 && rightPid != -1){

                    ok = false;

                }

            }

            if(ok){

            MessageData mdCopy = md;

            mdCopy.type = FEEDBACK;

            mdCopy.where = id;

            thread th(sendMessage, mdCopy, -1);

            th.detach();}

        }

        if(md.needId != id){

            if(md.type == FEEDBACK){

                thread th(sendMessage, md, -1);

                th.detach();

            }

            else{

                if(leftPid != -1){

                    thread th(sendMessage, md, leftPid \* 10);

                    th.detach();

                }

                if(rightPid != -1){

                    thread th(sendMessage, md, rightPid \* 10);

                    th.detach();

                }

            }

        }

    }

    zmq\_close(senderSocket);

    zmq\_ctx\_destroy(context);

    return 0;

}

**Вывод**

В данной лабораторной работе я познакомился с очередями сообщений и реализовал распределенную систему, используя ZeroMQ. Во время выполнения работы, я столкнулся с многими трудностями: для начала нужно было выбрать наиболее оптимальный способ связи между узлами, сперва я хотел использовать Publisher и Subscriber, но затем я перешел на более надежный и быстрый метод Push и Pull. Также были проблемы с тем, что узлы не успевали передавать сообщения, эта проблема была решена с помощью создания новых потоков для каждого сообщения.