Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
Московский авиационный институт
(национальный исследовательский университет)

Факультет № 8
«Информационные технологии и прикладная математика»
Кафедра №806
«Вычислительная математика и программирование»

Лабораторные работы №1 - №6

по курсу

“Операционные системы”

Работу выполнил студент 2 курса
группы 8О-206Б-17
Семенов Д.В.

Преподаватель: Миронов Е.С.
Дата:
Оценка:
Подпись:

Москва-2018

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная математика и программирование»
Факультет «Прикладная математика и физика»
Дисциплина «Операционные системы»

Лабораторная работа № 1

Группа М80-206Б-17
Студент: Семенов Денис Владиславович
Преподаватель: Миронов Евгений Сергеевич

Москва 2018

Лабораторная работа №1

Цель работы

Приобретение практических навыков диагностики работы программного обеспечения.

Задание

При выполнении последующих лабораторных работ необходимо продемонстрировать ключевые системные вызовы, которые в них используются и то, что их использование соответствует варианту ЛР.

Средство диагностики: strace

Код

2 Лабораторная

```
denis@denis-Extensa-2510G:~/labs_3sem/labs_os$ strace -f ./prog wow.txt
execve("./prog", ["./prog", "wow.txt"], [/* 78 vars */]) = 0
brk(NULL)                               = 0x1c04000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=111615, ...}) = 0
mmap(NULL, 111615, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f9528574000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\t\2\0\0\0\0\0"..., 832) =
832
...
pipe([4, 5])                            = 0
clone(strace: Process 5879 attached
child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
SIGCHLD, child_tidptr=0x7f95285729d0) = 5879
[pid  5878] fstat(0,  <unfinished ...>
[pid  5879] read(4,  <unfinished ...>
[pid  5878] <... fstat resumed> {st_mode=S_IFCHR|0620, st_rdev=makedev(136,
4), ...}) = 0
```

```
[pid  5878] brk(NULL)                    = 0x1c04000
[pid  5878] brk(0x1c25000)               = 0x1c25000
[pid  5878] read(0, ls -al
"ls -al\n", 1024)   = 7
[pid  5878] write(5, "ls\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"...,
2048) = 2048
[pid  5879] <... read resumed>
"ls\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"..., 2048) = 2048
[pid  5878] read(0,  <unfinished ...>
[pid  5879] clone(strace: Process 5880 attached
child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
SIGCHLD, child_tidptr=0x7f95285729d0) = 5880
[pid  5879] wait4(-1,  <unfinished ...>
[pid  5880] dup2(3, 1)                   = 1
[pid  5880] execve("/home/denis/bin/ls", ["ls", "-al"], [/* 78 vars */]) = -1 ENOENT
(No such file or directory)
...

[pid  5880] open("/lib/x86_64-linux-gnu/libselinux.so.1", O_RDONLY|
O_CLOEXEC) = 6
[pid  5880] read(6,
"\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\260Z\0\0\0\0\0\0"..., 832) = 832
...
[pid  5880] lseek(6, -936, SEEK_CUR)    = 608
[pid  5880] read(6, "TZif2\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\21\0\0\0\21\0\0\0\0"...,
4096) = 936
[pid  5880] close(6)                     = 0
[pid  5880] stat("/etc/localtime", {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
[pid  5880] stat("/etc/localtime", {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
[pid  5880] stat("/etc/localtime", {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
[pid  5880] stat("/etc/localtime", {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
[pid  5880] stat("/etc/localtime", {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
[pid  5880] stat("/etc/localtime", {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
[pid  5880] stat("/etc/localtime", {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
[pid  5880] stat("/etc/localtime", {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
[pid  5880] stat("/etc/localtime", {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
[pid  5880] stat("/etc/localtime", {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
[pid  5880] stat("/etc/localtime", {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
[pid  5880] stat("/etc/localtime", {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
[pid  5880] stat("/etc/localtime", {st_mode=S_IFREG|0644, st_size=1544, ...}) = 0
[pid  5880] write(1, "total 100\ndrwxrwxr-x 8 denis den"..., 789) = 789
[pid  5880] close(1)                     = 0
[pid  5880] close(2)                     = 0
[pid  5880] exit_group(0)                = ?
```

```
[pid  5880] +++ exited with 0 +++
[pid  5879] <... wait4 resumed> [{WIFEXITED(s) && WEXITSTATUS(s) == 0}],
0, NULL) = 5880
[pid  5879] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED,
si_pid=5880, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
```

```
denis@denis-Extensa-2510G:~/labs_3sem/labs_os$ cat wow.txt
total 100
drwxrwxr-x 8 denis denis  4096 дек 26 13:36 .
drwxrwxr-x 8 denis denis  4096 дек 10 23:00 ..
-rw-rw-r-- 1 denis denis  3600 окт 22 14:33 1lab.cpp
drwxrwxr-x 2 denis denis  4096 ноя 19 15:55 3lab
drwxrwxr-x 2 denis denis  4096 ноя 19 14:56 4lab
drwxrwxr-x 3 denis denis  4096 дек  3 16:11 5lab
drwxrwxr-x 4 denis denis  4096 дек 17 15:31 6lab
-rw-rw-r-- 1 denis denis   448 окт 21 23:28 help.c
drwxrwxr-x 2 denis denis  4096 дек 22 15:57 kp
-rw-rw-r-- 1 denis denis    85 дек 26 13:36 .~lock.Без имени 1.odt#
-rwxrwxr-x 1 denis denis 16696 окт 22 14:20 prog
drwxrwxr-x 2 denis denis  4096 окт 21 19:14 .vscode
-rw------- 1 denis denis   789 дек 26 13:41 wow.txt
-rw-rw-r-- 1 denis denis 31929 дек 26 13:36 Без имени 1.odt
```

3 Лабораторная

```
12569 clone(child_stack=0x7fbfa21e4fb0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7fbfa21e59d0, tls=0x7fbfa21e5700, child_tidptr=0x7fbfa21e59d0)
= 12570
12569 clone(child_stack=0x7fbfa19e3fb0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7fbfa19e49d0, tls=0x7fbfa19e4700, child_tidptr=0x7fbfa19e49d0)
= 12571
12569 clone(child_stack=0x7fbfa11e2fb0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7fbfa11e39d0, tls=0x7fbfa11e3700, child_tidptr=0x7fbfa11e39d0)
= 12572
12569 clone(child_stack=0x7fbfa09e1fb0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
```

```
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7fbfa09e29d0, tls=0x7fbfa09e2700, child_tidptr=0x7fbfa09e29d0)
= 12573
12571 clone(child_stack=0x7fbf93ffefb0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7fbf93fff9d0, tls=0x7fbf93fff700, child_tidptr=0x7fbf93fff9d0) =
12574
12573 +++ exited with 0 +++
12574 clone(child_stack=0x7fbfa09e1fb0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7fbfa09e29d0, tls=0x7fbfa09e2700, child_tidptr=0x7fbfa09e29d0)
= 12575
12575 +++ exited with 0 +++
```

4 Лабораторная

```
denis@denis-Extensa-2510G:~/labs_3sem/labs_os/4lab$ strace ./a.out wow.txt
execve("./a.out", ["./a.out", "wow.txt"], [/* 78 vars */]) = 0
brk(NULL)                               = 0x980000
...
open("wow.txt", O_WRONLY|O_CREAT, 0600) = 3
open("record", O_RDWR|O_CREAT, 0600)    = 4
fallocate(4, 0, 0, 2048)                = 0
mmap(NULL, 2048, PROT_READ|PROT_WRITE, MAP_SHARED, 4, 0) =
0x7fc496db1000
rt_sigprocmask(SIG_BLOCK, [USR1], NULL, 8) = 0
clone(child_stack=0, flags=CLONE_CHILD_CLEARTID|
CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7fc496d939d0) = 6200
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 4), ...}) = 0
brk(NULL)                               = 0x980000
brk(0x9a1000)                           = 0x9a1000
read(0, ls -al
"ls -al\n", 1024)               = 7
msync(0x7fc496db1000, 2048, MS_ASYNC)   = 0
kill(6200, SIGUSR1)                     = 0
read(0, exit
"exit\n", 1024)                 = 5
msync(0x7fc496db1000, 2048, MS_ASYNC)   = 0
kill(6200, SIGUSR1)                     = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

5 Лабораторная

```
denis@denis-Extensa-2510G:~/labs_3sem/labs_os/5lab$ strace ./a.out
execve("./a.out", ["./a.out"], [/* 78 vars */]) = 0
brk(NULL)                               = 0x1f4f000
...
brk(NULL)                               = 0x1f4f000
brk(0x1f70000)                           = 0x1f70000
open("/home/denis/labs_3sem/labs_os/5lab/libqueue.so", O_RDONLY|
O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20\7\0\0\0\0\0\0"..., 832) =
832
fstat(3, {st_mode=S_IFREG|0775, st_size=8408, ...}) = 0
mmap(NULL, 2101312, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_DENYWRITE, 3, 0) = 0x7f44a41cf000
mprotect(0x7f44a41d0000, 2093056, PROT_NONE) = 0
mmap(0x7f44a43cf000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0x7f44a43cf000
close(3)                                = 0
mprotect(0x7f44a43cf000, 4096, PROT_READ) = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 4), ...}) = 0
write(1, "9.500000\n", 99.500000
)             = 9
munmap(0x7f44a41cf000, 2101312)         = 0
exit_group(0)                           = ?
+++ exited with 0 +++

denis@denis-Extensa-2510G:~/labs_3sem/labs_os/5lab$ strace ./prog
execve("./prog", ["./prog"], [/* 78 vars */]) = 0
brk(NULL)                               = 0xeea000
...
open("./libqueue.so", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20\7\0\0\0\0\0\0"..., 832) =
832
...
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=111615, ...}) = 0
mmap(NULL, 111615, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff1cc18c000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\t\2\0\0\0\0\0"..., 832) =
832
fstat(3, {st_mode=S_IFREG|0755, st_size=1868984, ...}) = 0
mmap(NULL, 3971488, PROT_READ|PROT_EXEC, MAP_PRIVATE|
```

```
MAP_DENYWRITE, 3, 0) = 0x7ff1cb9b8000
mprotect(0x7ff1cbb78000, 2097152, PROT_NONE) = 0
mmap(0x7ff1cbd78000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x1c0000) = 0x7ff1cbd78000
mmap(0x7ff1cbd7e000, 14752, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ff1cbd7e000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS, -1, 0) = 0x7ff1cc18b000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS, -1, 0) = 0x7ff1cc18a000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS, -1, 0) = 0x7ff1cc189000
arch_prctl(ARCH_SET_FS, 0x7ff1cc18a700) = 0
mprotect(0x7ff1cbd78000, 16384, PROT_READ) = 0
mprotect(0x7ff1cbf82000, 4096, PROT_READ) = 0
mprotect(0x600000, 4096, PROT_READ)     = 0
mprotect(0x7ff1cc1a9000, 4096, PROT_READ) = 0
munmap(0x7ff1cc18c000, 111615)          = 0
brk(NULL)                               = 0xeea000
brk(0xf0b000)                           = 0xf0b000
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 4), ...}) = 0
write(1, "9.500000\n", 99.500000
)             = 9
write(1, "8.500000\n", 98.500000
)             = 9
write(1, "9.500000\n", 99.500000
)             = 9
exit_group(0)                           = ?
+++ exited with 0 +++
```

6 Лабораторная

```
denis@denis-Extensa-2510G:~/labs_3sem/labs_os/6lab$ strace ./progclient
execve("./progclient", ["./progclient"], [/* 78 vars */]) = 0
brk(NULL)                               = 0x11cd000
...
futex(0x7f727d5041ac, FUTEX_WAKE_PRIVATE, 2147483647) = 0
futex(0x7f727d5041b8, FUTEX_WAKE_PRIVATE, 2147483647) = 0
eventfd2(0, 0)                          = 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
```

```
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 4), ...}) = 0
write(1, "Client Starting\342\200\246.\n", 20Client Starting….
) = 20
eventfd2(0, 0)                          = 4
fcntl(4, F_GETFL)                       = 0x2 (flags O_RDWR)
...
mmap(NULL, 8392704, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f727c465000
mprotect(0x7f727c465000, 4096, PROT_NONE) = 0
clone(child_stack=0x7f727cc64fb0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tidptr=0x7f727cc659d0, tls=0x7f727cc65700, child_tidptr=0x7f727cc659d0)
= 6339
...
write(1, "Enter your id: ", 15Enter your id: )        = 15
read(0, 3
"3\n", 1024)                    = 2
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)
write(6, "\1\0\0\0\0\0\0\0", 8)         = 8
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, "User added\n", 11User added
)           = 1
```

Выводы

Выполняя лабораторную работу, я научился «заглядывать» внутрь моей программы. Strace — замечательная утилита для отслеживания системных вызовов. До того, как я узнал об этой программе, я даже не догадывался, что существует столько системных вызовов. Она очень удобна в использовании и имеет множество полезных флагов.

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная математика и программирование»
Факультет «Прикладная математика и физика»
Дисциплина «Операционные системы»

Лабораторная работа № 2

Группа М80-206Б-17
Студент: Семенов Денис Владиславович
Преподаватель: Миронов Евгений Сергеевич

Москва 2018

Лабораторная работа №2

Цель работы

Приобретение практических навыков в:

- Управление процессами в ОС
- Обеспечение обмена данных между процессами посредством каналов

Задание

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe).

Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Вариант задания

На вход программе подается команда интерпретатора команд и имя файла. Программа должна перенаправить стандартный ввод команды с этого файла и вывести результат команды в стандартный выходной поток. Использование операций write и printf запрещено.

Код

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <errno.h>
```

```
#include <string.h>
#include <stdbool.h>
#include <fcntl.h>
/*
В родиетльском процессе мы считываем команды и преоразуем,
в дочернем записываем все в файл
*/
int main(int argc, char* argv[]) {
int status;
pid_t pid;
int rv;

int file_pipes[2];
char buffer[2][1024];

int fd = open(argv[1], O_CREAT|O_WRONLY, S_IWUSR|S_IRUSR);

if (argc == 1) {
fprintf(stderr, "No file detected.\n");
exit(1); // ошибка в подготовке
}

if (pipe(file_pipes) != 0) {
perror("Pipes: ");
exit(1); // ошибка в подготовке
}

switch (pid = fork()) {
case -1:
{
perror("cannot fork \n");
exit(1); // ошибка в подготовке
}
case 0:
{
//процесс - ребенок
while (true) {
read(file_pipes[0], buffer, sizeof(buffer));
if (strcmp(buffer[0], "exit") == 0) break;
pid_t exec_pid;
int exec_back;
switch (exec_pid = fork()) {
case -1:
{
perror("cannot fork\n");
exit(1); // ошибка в подготовке
}
case 0:
{
dup2(fd, STDOUT_FILENO);
if (buffer[1][0] != '\0'){
execlp(buffer[0],buffer[0], buffer[1], (char*)NULL);
```

```
} else {
execlp(buffer[0],buffer[0], (char*)NULL);
}
dup2(STDOUT_FILENO, fd);
exit(exec_back);
}
default:
{
wait(&exec_back);
fflush(stdout);
}

} // end switch
} // end while
wait(&rv);
}
default:
{
// процесс - родитель
// происходит преобразование в команду и параметры

char buff[1024];
char argums[2][1024]; // до 4 аргументов

do {
bool done = false;
int swords = 0; // количество слов
int schars= 0; //количество букв в слове
int achars = 0; // всего букв в строке

for (int i = 0; i < 5; i++){
for (int j = 0; j < 1024; j++){
argums[i][j] = '\0';
}
}

fgets(buff, sizeof (buff), stdin);
char c = buff[achars];
while(!done){
while(c != ' ' && c != '\n' && c != EOF) {
argums[swords][schars] = c;
schars++;
achars++;
c = buff[achars];
}
swords++;
schars = 0;
achars++;
c = buff[achars];
if (buff[achars] == '\0') done = true;
}
argums[5][0] = (char)NULL;
```

```
write(file_pipes[1], argums, sizeof(argums));
} while(strcmp(argums[0],"exit") != 0);

exit(rv);
}
}
close(fd);
return 0;

}
```

Выводы

Во время выполнения лабораторной работы у меня возникла проблема: каким образом передавать команды и как их исполнять? Первая решилась довольно легко. Так как нужно было выбрать между пайпом и сигналами, то выбор, очевидно, пал на пайп. Решил преобразовывать команду с ключами в двумерный массив [5][1024], с помощью которого можно передать команду и 4 параметра. Осталась еще одна проблема: как выполнять полученные команды? После прочтения нескольких мануалов и нескольких часов раздумий выбор пал на системный вызов execlp. Причем у него есть особенность. Вот описание: int execlp(const char *file, const char *arg, ...); «Параметр const char *arg и аналогичные записи в функциях execl, execlp, и execle подразумевают параметры arg0, arg1, ..., argn. Все вместе они описывают один или нескольких указателей на строки, заканчивающиеся NULL, которые представляют собой список параметров, доступных исполняемой программе. Первый параметр, по соглашению, должен указывать на имя, ассоциированное с файлом, который надо исполнить. Список параметров должен заканчиваться NULL.» Именно поэтому пришлось писать, на первый взгляд, настолько странную конструкцию execlp(buffer[0], buffer[0], buffer[1], (char*)NULL);

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная математика и программирование»
Факультет «Прикладная математика и физика»
Дисциплина «Операционные системы»

Лабораторная работа № 3

Группа М80-206Б-17
Студент: Семенов Денис Владиславович
Преподаватель: Миронов Евгений Сергеевич

Москва 2018

Лабораторная работа №3

Цель работы

Целью является приобретение практических навыков в:

- Управление потоками в ОС
- Обеспечение синхронизации между потоками

Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). При создании необходимо предусмотреть ключи, которые позволяли бы задать максимальное количество потоков, используемое программой. При возможности необходимо использовать максимальное количество возможных потоков. Ограничение потоков может быть задано или ключом запуска вашей программы, или алгоритмом.

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

Вариант задания

Отсортировать массив чисел при помощи битонической сортировки.

Код

```
#include <algorithm>
#include <iostream>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <stdbool.h>
#include <pthread.h>
#include <semaphore.h>
#include <string.h>
#include <fstream>
#include <vector>

using namespace std;
```

```
unsigned long long nProcess = 1;
sem_t semaphore;

typedef struct {
vector<long long>::iterator l;
vector<long long>::iterator r;
bool inv;
} Data;


void* bitseqsort(void *thdata) {

Data *data = new Data;
data->l = ((Data*)(thdata))->l;
data->r = ((Data*)(thdata))->r;
data->inv = ((Data*)(thdata))->inv;

if (data->r - data->l < 1)
return NULL;
   vector<long long>::iterator m = data->l + (data->r - data->l) / 2 ;

   for (vector<long long>::iterator i = data->l, j = m+1; i <= m && j <= data->r; i++, j++) {
      if (!data->inv){
if (*i > *j){
swap(*i, *j);
}
} else {
if (*i < *j){
swap(*i, *j);
}
}
   }

Data *putl = new Data;
putl->l = data->l;
putl->r = m;
putl->inv = data->inv;

Data *putr = new Data;
putr->l = m+1;
putr->r = data->r;
putr->inv = data->inv;

int curProc;
sem_getvalue(&semaphore, &curProc);
//cout << "Sema: " << curProc << endl;

if(curProc == 0){
bitseqsort(putl);
bitseqsort(putr);
} else {
```

```
pthread_t thread_pid;
pthread_attr_t attr;
pthread_attr_init(&attr);
pthread_attr_setscope(&attr, PTHREAD_SCOPE_PROCESS);



sem_wait(&semaphore);

if(pthread_create(&thread_pid, NULL, bitseqsort, putl)) {
cout << "ERROR creating thread" << endl;
exit(1);
}
bitseqsort(putr);
if(pthread_join(thread_pid, NULL)) {
cout << "ERROR joining thread" << endl;
exit(1);
}
sem_post(&semaphore);
}

delete data;
delete putl;
delete putr;
return NULL;
}

void* makebitonic(void* thdata);

void* make_n_sort(void *thdata) {
Data *data = new Data;
data->l = ((Data*)(thdata))->l;
data->r = ((Data*)(thdata))->r;
data->inv = ((Data*)(thdata))->inv;

makebitonic(data);
   bitseqsort(data);
delete data;
return NULL;
}



// делает послдовательность битонной
// дозаполняем ее элементами больше максимального, не влияет на ассимптотику
void* makebitonic(void* thdata) {
Data *data = new Data;
data->l = ((Data*)(thdata))->l;
data->r = ((Data*)(thdata))->r;
data->inv = ((Data*)(thdata))->inv;



   if (data->r - data->l <= 1) {
return NULL;
```

```
}

// находим центр
    vector<long long>::iterator m = data->l + (data->r - data->l) / 2;

Data *putL = new Data;
putL->l = data->l;
putL->r = m;
putL->inv = 0;
Data *putR = new Data;
putR->l = m+1;
putR->r = data->r;
putR->inv = 1;
int curProc;
sem_getvalue(&semaphore, &curProc);
if(curProc == 0){
make_n_sort(putL);
make_n_sort(putR);
} else {

pthread_t thread_pid;
pthread_attr_t attr;
pthread_attr_init(&attr);
pthread_attr_setscope(&attr,PTHREAD_SCOPE_PROCESS);

sem_wait(&semaphore); // отнимает, если может
if(pthread_create(&thread_pid, NULL, make_n_sort, putL)) {
cout << "ERROR creating thread" << endl;
exit(1);
}



make_n_sort(putR);

if(pthread_join(thread_pid, NULL)) {
cout << "ERROR joining thread" << endl;
exit(1);
}

sem_post(&semaphore);
}

delete data;
delete putL;
delete putR;
return NULL;
}

void bitonicsort(vector<long long> &a) {

    vector<long long>::iterator max;
max = std::max_element(a.begin(), a.end());
```

```
long long nmax = *max + 1;
long long n = 1;
   while (n < a.end() - a.begin()) n *= 2;

unsigned long long N = n;

   while (a.size() < N) a.push_back(nmax); // дозаполнение элементов

Data *put = new Data;
put->l = a.begin();
put->r = a.end() - 1;
put->inv = 0;

   makebitonic(put);
   bitseqsort(put);

// чистим
// расчет на сортировку по возрастанию
while(*(a.end()-1) == ((nmax))){
a.pop_back();
}

delete put;
}

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

// проверка входных данных
if (argc <= 1) {
fprintf(stderr, "No file detected.\n");
exit(1);
} else if(argc == 2 && strcmp(argv[1], "-p") == 0) {
fprintf(stderr, "Enter number of threads.\n");
exit(1);
} else if(argc == 3) {
fprintf(stderr, "No file detected.\n");
exit(1);
}

// составление кол-ва потоков и открытие файла
nProcess = 0;
int fil = 1; // определяет положение файла
if(strcmp(argv[1], "-p") == 0){
nProcess = atoi(argv[2]);
fil = 3;
}

ifstream fin(argv[fil]);
if (!fin.is_open()){
fprintf(stderr, "Cannot open file.\n");
exit(2);
}
```

```
long long kol= 0;
fin >> kol;
vector<long long> a(kol);
for(long long i = 0; i < kol; i++)
fin >> a[i];
sem_init(&semaphore, 0, nProcess-1);
bitonicsort(a);
sem_destroy(&semaphore);

ofstream fout("out.txt");
if (!fout.is_open()){
fprintf(stderr, "Cannot open file.\n");
exit(2);
}
for(unsigned int i = 0; i < a.size(); i++){
fout << a[i] << " ";
if ( i%10 == 0 ){
fout << endl;
}
}
cout << endl;

return 0;
}
```

Выводы

В лабораторной работе номер 3 были всего две сложные вещи: правильно написать битоническую сортировку и организовать одновременную работу 8 потоков. Так как поток позволяет запускать только функции вида void* threadFunc(void* thread_data){}, то пришлось каждый раз преобразовывать аргументы в нормальный вид конструкциями типа Data *data = new Data; data->l = ((Data*)(thdata))→l.

И только потом запускать поток, используя локальные скопированные данные. Для шаблона запуска потока был выбран вариант запуска одной половины в потоке. Это оказался наиболее надежным и читабельным (относительно кода) вариантом. Также, для наглядности и простоты кода было принято решение ввести итераторы и осуществлять все операции именно через них (речь идет о swap и max).

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная математика и программирование»
Факультет «Прикладная математика и физика»
Дисциплина «Операционные системы»

Лабораторная работа № 4

Группа М80-206Б-17
Студент: Семенов Денис Владиславович
Преподаватель: Миронов Евгений Сергеевич

Москва 2018

Лабораторная работа №4

Цель работы

Приобретение практических навыков в:

- Освоение принципов работы с файловыми системами
- Обеспечение обмена данных между процессами посредством технологии «File mapping»

Задание

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы (memory-mapped files). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Вариант задания:

Программа позволяющая перенаправлять стандартный вывод. На вход программе подается имя выходного файла. Далее программа должна принимать различные команды интерпретатора команд и весь их вывод перенаправлять в файл.

Код

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <errno.h>
#include <string.h>
#include <stdbool.h>
```

```
#include <fcntl.h>
#include <sys/mman.h>
#include <signal.h>

/*
В родиетльском процессе мы считываем команды и преоразуем,
в дочернем записываем все в файл
*/

typedef struct{
char buffer[2][1024];
} RECORD;


int main(int argc, char* argv[]) {
if (argc == 1) {
fprintf(stderr, "No file detected.\n");
exit(1); // ошибка в подготовке
}

int status;
pid_t pid;
int rv;

RECORD *mapped;

int fd = open(argv[1], O_CREAT|O_WRONLY, S_IWUSR|S_IRUSR);

int f = open("record", O_RDWR | O_CREAT, S_IWUSR|S_IRUSR);

posix_fallocate(f, 0, sizeof(RECORD));

mapped = (RECORD*)mmap(0, sizeof(RECORD), PROT_READ | PROT_WRITE,
MAP_SHARED, f, 0);

sigset_t set;
int sig;
int *sigptr = &sig;
int ret_val;
sigemptyset(&set);
sigaddset(&set, SIGUSR1);
sigprocmask(SIG_BLOCK, &set, NULL);

switch (pid = fork()) {
case -1:
{
perror("cannot fork \n");
exit(1); // ошибка в подготовке
}
case 0:
{
//процесс - ребенок
```

```
while (true) {
// wait signal
ret_val = sigwait(&set, sigptr);

if(ret_val == -1){
perror("sigwait failed\n");
} else {
if(*sigptr == SIGUSR1){
if (strcmp((*mapped).buffer[0], "exit") == 0) break;
pid_t exec_pid;
int exec_back;
switch (exec_pid = fork()) {
case -1:
{
perror("cannot fork\n");
exit(1); // ошибка в подготовке
}
case 0:
{
dup2(fd, STDOUT_FILENO);
if ((*mapped).buffer[1][0] != '\0'){
execlp((*mapped).buffer[0],(*mapped).buffer[0], (*mapped).buffer[1], (char*)NULL);
} else {
execlp((*mapped).buffer[0],(*mapped).buffer[0], (char*)NULL);
}
dup2(STDOUT_FILENO, fd);
exit(exec_back);
}
default:
{
wait(&exec_back);
fflush(stdout);
}

} // end switch
}
else
printf("sigwait returned with sig: %d\n", *sigptr);
}
} // end while
wait(&rv);
}
default:
{
// процесс - родитель
// происходит преобразование в команду и параметры

char buff[1024];
char argums[2][1024]; // до 4 аргументов

do {
bool done = false;
```

```
int swords = 0; // количество слов
int schars= 0; //количество букв в слове
int achars = 0; // всего букв в строке

for (int i = 0; i < 5; i++){
for (int j = 0; j < 1024; j++){
argums[i][j] = '\0';
}
}

fgets(buff, sizeof (buff), stdin);
char c = buff[achars];
while(!done){
while(c != ' ' && c != '\n' && c != EOF) {
argums[swords][schars] = c;
schars++;
achars++;
c = buff[achars];
}
swords++;
schars = 0;
achars++;
c = buff[achars];
if (buff[achars] == '\0') done = true;
}
argums[5][0] = (char)NULL;

for(int i = 0 ; i < 2; i++){
for(int j = 0; j < 1024; j++){
mapped->buffer[i][j] = argums[i][j];
}
}

msync((void*)mapped, sizeof(RECORD), MS_ASYNC);
status = kill(pid, SIGUSR1 );
if ( status < 0)
perror("kill failed");

} while(strcmp(argums[0],"exit") != 0);

exit(rv);
}
}
munmap((void*)mapped, sizeof(RECORD));
close(fd);
return 0;

}
```

Выводы

Так как четвертая работа представляла из себя немного измененную третью, то не составило особого труда ее написать. Нужно было всего лишь заменить передачу параметров не через pipe, а через mmap. Используя mmap, мы проецируем наш файл в память и можем работать с ним быстрее, чем с диском. File mapping может быть полезен когда необходим доступ к случайным областям очень большого файла, выигрыш происходит за счет отсутствия последовательного считывания. Но эта технология не избавляет нас от конкуренции за ресурсами. Общую память нужно синхронизировать при попытки обращения к ней.

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная математика и программирование»
Факультет «Прикладная математика и физика»
Дисциплина «Операционные системы»

Лабораторная работа № 5

Группа М80-206Б-17
Студент: Семенов Денис Владиславович
Преподаватель: Миронов Евгений Сергеевич

Москва 2018

Лабораторная работа №5

Цель работы

Целью является приобретение практических навыков в:

- Создание динамических библиотек
- Создание программ, которые используют функции динамических библиотек

Задание

Требуется создать динамическую библиотеку, которая реализует определенный функционал. Далее использовать данную библиотеку 2-мя способами: во время компиляции (на этапе «линковки»/linking) и во время исполнения программы, подгрузив библиотеку в память с помощью системных вызовов. В конечном итоге, программа должна состоять из следующих частей: динамическая библиотека, реализующая заданных вариантом интерфейс; тестовая программа, которая используют библиотеку, используя знания полученные на этапе компиляции; тестовая программа, которая использует библиотеку, используя только местоположение динамической библиотеки и ее интерфейс. Провести анализ между обоими типами использования библиотеки.

Структура данных, с которой должна обеспечивать работу библиотека:

Работа с очередью

Тип данных, используемый структурой:

Вещественный 64-битный

Код

```
#include <stdlib.h>
#include <stdio.h>
#include "queue.h"
```

```
int main(){
queue *q = create();
push(q, 9.5);
push(q, 8.5);
print(q);
printf("%f\n",pop(q) );
destroy(q);
}
```

```
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include "queue.h"
#include <dlfcn.h>



int main(){
void *dl_handle = dlopen("/home/denis/labs_3sem/labs_os/5lab/libqueue.so", RTLD_LAZY );

if (!dl_handle) {
printf( "!!! %s\n", dlerror() );
return 0;
}
queue *(*func_create)(void) = dlsym( dl_handle, "create" );
char *error = dlerror();
if (error != NULL) {
printf( "!!! %s\n", error );
return 0;
}

bool (*func_push)(queue *q, const double var) = dlsym( dl_handle, "push" );
error = dlerror();
if (error != NULL) {
printf( "!!! %s\n", error );
return 0;
}

void (*func_print)(const queue *q) = dlsym( dl_handle, "print" );
error = dlerror();
if (error != NULL) {
printf( "!!! %s\n", error );
return 0;
}



queue *q = (*func_create)();
(*func_push)(q, 9.5);
```

```
//push(q, 9.5);
//push(q, 8.5);
(*func_print)(q);
//print(q);
//printf("%f\n",pop(q) );
//destroy(q);

dlclose(dl_handle);
return 0;
}
```

```
prog: main.o libqueue.so
    gcc -o prog main.o -L. -lqueue -Wl,-rpath,. #передать опцию с аргументом, добавляет текущий каталог в поиск
    # L - где искать библиотеку

main.o: main.c
    gcc -c main.c

libqueue.so: queue.o
    gcc -shared -o libqueue.so queue.o

queue.o: queue.c
    gcc -c -fPIC queue.c #позиционно независимый код, плавающие адреса

clean:
    rm -f *.o *.so binary
```

Выводы

Статическая линковка проста и удобна, и мы практически всегда ее используем, в отличие от динамической. Динамические библиотеки оказали хоть и очень полезными, но крайне неудобными в использовании. Каждую используемую функцию приходится брать вручную из библиотеки. Например, queue*(*func_create)(void) = dlsum(dl_handle, «create»);

Но благодаря этому подходу исполняемы файлы оказываются куда меньших размеров и используется намного меньше оперативной памяти. Это отлично подходит для больших и тяжелых библиотек, для небольших проектов (и требующих уникальных библиотек) этот метод не очень хорош.

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная математика и программирование»
Факультет «Прикладная математика и физика»
Дисциплина «Операционные системы»

Лабораторная работа № 6

Группа М80-206Б-17
Студент: Семенов Денис Владиславович
Преподаватель: Миронов Евгений Сергеевич

Москва 2018

Лабораторная работа №6

Цель работы

Целью является приобретение практических навыков в управлении серверами сообщений

Задание

Реализовать клиент-серверную систему по асинхронной обработке запросов. Необходимо составить программы сервера и клиента. При запуске сервер и клиент должны быть настраиваемы, то есть должна быть возможность поднятия на одной ЭВМ нескольких серверов по обработке данных и нескольких клиентов, которые к ним относятся. Все общение между процессами сервера и клиентов должно осуществляться через сервер сообщений. Серверное приложение – банк. Клиентское приложение клиент банка. Клиент может отправить какую-то денежную сумму в банк на хранение. Клиент также может запросить из банка произвольную сумму. Клиенты могут посылать суммы на счета других клиентов. Запросить собственный счет. При снятии должна производиться проверка на то, что у клиента достаточно денег для снятия денежных средств. Идентификатор клиента задается во время запуска клиентского приложения, как и адрес банка. Считать, что идентификаторы при запуске клиентов будут уникальными.

Сервер сообщений: ZeroMQ

Внутреннее хранилище сервера:

Бинарное дерево, где ключом является идентификатор клиента

Тип ключа клиента:

Целое число 32 бита

Код

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

void help_page(){
    printf("\nc - credit account\n");
    printf("d - debet account\n");
    printf("\n");
    printf("TO ask for money print \? (d/c)\n");
    printf("To send money print \"+ (amount of money) (c/d) (user id to sent)\"\n");
    printf("To enrich your account print \"+ (amunt of money) (c/d)\"\n");
    printf("To get modey print \"- (amount of modey) (c/d)\"\n");
}


char *s_recv (void *socket) {
char buffer [256];
int size = zmq_recv (socket, buffer, 255, 0);
if (size == -1)
return NULL;
if (size > 255)
size = 255;
buffer [size] = '\0';
return strndup(buffer, sizeof(buffer)-1); //in *nix */
//return strdup (buffer);
}

int main (int argc, char const *argv[])
{
    void* context = zmq_ctx_new();
    printf("Client Starting….\n");

    void* request = zmq_socket(context, ZMQ_REQ);
    zmq_connect(request, "tcp://localhost:4040");

    char initbuf[256];
    char *d = initbuf;
    int count = 0;
    int id = 0;
```

```
char idchar[10] = {'\0'};

do{
    initbuf[0] = '!';
    initbuf[1] = ' ';

    for(int i = 2; i < 256; i++){
        initbuf[i] = '\0';
    }

    printf("Enter your id: ");
    scanf("%s",idchar);

    id = atoi(idchar);
    for(int i = 0; i < 260; i++){
initbuf[i+2] = idchar[i];
    }
    zmq_send(request, initbuf, 256, 0);

    d = s_recv(request);
    printf("%s\n", d);

} while (!strcmp(d, "Such id is already in system"));


//zmq_send(request, buff, )
getchar();

for(;;)
{
    printf("Enter new command: ");
    char buffer[256] = {'\0'};
    fgets(buffer, 256, stdin);
    if(buffer[0] == '\0'){
        return 0;
    }

    if (!strcmp(buffer, "help\n")) {
        help_page();
    } else if(buffer[0] == '?'){
        char newbuffer[256] = {'\0'};
        newbuffer[0] = buffer[0];
        newbuffer[1] = buffer[1];
        newbuffer[2] = buffer[2];
```

```
        newbuffer[3] = ' ';
        for(int i = 4; i < 14; i++){
            newbuffer[i] = idchar[i-4];
        }
        zmq_send(request, newbuffer, 256, 0);
        zmq_recv(request, newbuffer, 256, 0);
        printf("%s\n", newbuffer);

    } else if(buffer[0] == '-'){

        int i = 0;
        while(buffer[i] != '\n'){
            i++;
        }
        buffer[i] = ' ';
        i++;
        for( int j = 0; j < 10; i++, j++){
            buffer[i] = idchar[j];
        }



        zmq_send(request, buffer, 256, 0);
        zmq_recv(request, buffer, 256, 0);
        printf("%s\n", buffer);
    } else if(buffer[0] == '+'){
        char sep[10] = " /\n";
        char send_buffer[256];
        for(int i = 0; i < 256; i++){
            send_buffer[i] = buffer[i];
        }
        char *d = buffer;
        char *nstr = strtok(d, sep);
        nstr = strtok(NULL, sep); // modey
        nstr = strtok(NULL, sep); // c/d
        nstr = strtok(NULL, sep); // id user
        int i = 0;
        if(nstr == NULL){
            while(send_buffer[i++] != '\0');
            send_buffer[i-2] = ' ';
            i--;
            for(int j = 0; j < 10; j++){
                send_buffer[i + j] = idchar[j];
            }
        }
```

```
            zmq_send(request, send_buffer, 256, 0);
            zmq_recv(request, send_buffer, 256, 0);
            printf("%s\n", send_buffer);

        } else {
            zmq_send(request, buffer, 256, 0);
            zmq_recv(request, buffer, 256, 0);
            printf("%s\n", buffer);
            sleep(1);
            count++;
        }

    }
    // We never get here though.
    zmq_close(request);
    zmq_ctx_destroy(context);

    return 0;
}
```

```
#include "btree.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <assert.h>
#include <zmq.h>

#define MIN_ACC -1000

char *s_recv (void *socket) {
char buffer [256];
int size = zmq_recv (socket, buffer, 255, 0);
if (size == -1)
return NULL;
if (size > 255)
size = 255;
buffer [size] = '\0';
return strndup(buffer, sizeof(buffer)-1); //in *nix */
//return strdup (buffer);
}

int main(){
void* context = zmq_ctx_new();
    void* respond = zmq_socket(context, ZMQ_REP);
```

```
   int rc = zmq_bind(respond, "tcp://*:4040");
   assert (rc == 0);

   printf("Starting…\n");

   tnode *tree = NULL;

for(;;)
   {
      char buffer[256];
      char *d = buffer;

      d = s_recv(respond);


      printf("Recieved %s\n", d);
      if(strcmp("", d)){

         char sep[10] = " /";

         char *nstr;
         nstr = strtok(d, sep);

         // getting operation
         if(nstr == NULL){
            printf("OK\n");
         } else
         if (!strcmp(nstr, "+")){
            nstr = strtok(NULL, sep);

            //getting sum
            char *ptr;
            long sum_to_append = strtol(nstr, &ptr, 10);
            printf("Summ to be added: %ld\n", sum_to_append);

            // getting account
            nstr = strtok(NULL, sep);
            if (!strcmp(nstr, "d")) {
               printf("Account to add: d");
               nstr = strtok(NULL, sep);
               int userid = atoi(nstr);
               tnode **d = find_node(&tree, userid);
               if(d == NULL){
                  zmq_send(respond, "No user with such id", 20, 0);
               } else {
                  (*d)->debet_acc += sum_to_append;
                  zmq_send(respond, "Added", 5, 0);
               }

            } else if(!strcmp(nstr, "c")) {
               printf("Account to add: c");
               nstr = strtok(NULL, sep);
```

```
            int userid = atoi(nstr);
            tnode **d = find_node(&tree, userid);
            if(d == NULL){
                zmq_send(respond, "No user with such id", 20, 0);
            } else {
                (*d)->credit_acc += sum_to_append;
                zmq_send(respond, "Added", 5, 0);
            }
        }else {
            zmq_send(respond, "No such account (chech d/c)", 27, 0);
        }

        printf("OK\n");
    } else if(!strcmp(nstr, "!")) {
        nstr = strtok(NULL, sep);
        int idinsert = atoi(nstr);
        tnode **d = find_node(&tree, idinsert);
        if (d == NULL){
            add(&tree, idinsert);
            zmq_send(respond, "User added", 10, 0);
            printf("User added\n");
        } else {
            zmq_send(respond, "Such id is already in system", 28 ,0);
            printf("Already in system\n");
        }
        printf("OK\n");
    } else if(!strcmp(nstr, "?")) {
        nstr = strtok(NULL, sep);
        if (!strcmp(nstr, "d")){
            nstr = strtok(NULL, sep);
            int idinsert = atoi(nstr);
            printf("id is: %d\n", idinsert);
            tnode **d = find_node(&tree, idinsert);

            const int n = snprintf(NULL, 0, "There is %ld on your d account", (*d)->debet_acc);
            assert(n > 0);
            char buf[n+1];
            int c = snprintf(buf, n+1, "There is %ld on your d account", (*d)->debet_acc);
            assert(buf[n] == '\0');
            assert(c == n);

            zmq_send(respond, buf, n, 0);
            printf("OK, sent\n");
        } else if (!strcmp(nstr, "c")) {
            nstr = strtok(NULL, sep);
            int idinsert = atoi(nstr);
            tnode **d = find_node(&tree, idinsert);

            const int n = snprintf(NULL, 0, "There is %ld on your c account", (*d)->credit_acc);
            assert(n > 0);
            char buf[n+1];
            int c = snprintf(buf, n+1, "There is %ld on your c account", (*d)->credit_acc);
```

```
            assert(buf[n] == '\0');
            assert(c == n);

            zmq_send(respond, buf, n, 0);
        } else {
            zmq_send(respond, "No such account (chech d/c)", 27, 0);
        }
    } else if (!strcmp(nstr, "-")) {
        nstr = strtok(NULL, sep);
        char *ptr;
        long sum_to_minus = strtol(nstr, &ptr, 10);
        nstr = strtok(NULL, sep);

        if (!strcmp("d", nstr)) {
            nstr = strtok(NULL, sep);
            int idinsert = atoi(nstr);
            printf("id is: %d\n", idinsert);
            tnode **d = find_node(&tree, idinsert);
            if(d == NULL){
                zmq_send(respond, "No user with such id", 20, 0);
            } else {
                if ((*d)->debet_acc == MIN_ACC){
                    zmq_send(respond, "Reached the minimum value of the acc", 36, 0);
                } else {
                    if (((*d)->debet_acc - sum_to_minus) < MIN_ACC){
                        zmq_send(respond, "Impossible to do, limit will be reached", 39, 0);
                    } else {
                        (*d)->debet_acc -= sum_to_minus;
                        zmq_send(respond, "Done!!!!!!!!!!", 4, 0);
                    }
                }

            }
        } else if (!strcmp("c", nstr)) {
            nstr = strtok(NULL, sep);
            int idinsert = atoi(nstr);
            printf("id is: %d\n", idinsert);
            tnode **d = find_node(&tree, idinsert);
            if(d == NULL){
                zmq_send(respond, "No user with such id", 20, 0);
            } else {
                if((*d)->credit_acc - sum_to_minus < 0){
                    long ostatok = (*d)->credit_acc - sum_to_minus; // otricatelinoye
                    if ((*d)->debet_acc == MIN_ACC){
                        zmq_send(respond, "Reached the minimum value of the acc", 36, 0);
                    } else {
                        if (((*d)->debet_acc + ostatok) < MIN_ACC){
                            zmq_send(respond, "Impossible to do, limit will be reached", 39, 0);
                        } else {
                            (*d)->debet_acc += ostatok;
                            (*d)->credit_acc = 0;
                            zmq_send(respond, "Done!!!!!!!", 4, 0);
```

```
                        }
                    }
                } else {
                    (*d)->credit_acc = 0;
                    zmq_send(respond, "Done!!!!!!!!!11", 4, 0);
                }

            }
        } else {
            zmq_send(respond, "No such account (chech d/c)", 27, 0);
        }

    }
  }


 }
 zmq_close(respond);
 zmq_ctx_destroy(context);

 return 0;
}
```

Выводы

Сервера сообщений оказались намного более удобным методом связи двух программ (процессов \ потоков) за счет своей абстракции. Тебе не нужно задумываться над реализацией, ты просто соединяешься и обмениваешься сообщениями. И можно выбрать любой шаблон передачи данных, все они досконально продуманны разработчиками. В своей лабораторной я пользовался только REQ и REP, их хватило для реализации клиент-серверного приложения банка. Приятной особенностью было узнать, что доставка сообщения гарантируется, даже после обрыва соединения. Написание лабораторной работы дало мне много практических навыков, которые я (уже) реализовал в курсовой работе.

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
Московский авиационный институт
(национальный исследовательский университет)

Факультет № 8
«Информационные технологии и прикладная математика»
Кафедра №806
«Вычислительная математика и программирование»

Курсовой проект

по курсу

“Операционные системы”

Работу выполнил студент 2 курса
группы 8О-206Б-17
Семенов Д.В.

Преподаватель: Миронов Е.С.
Дата:
Оценка:
Подпись:

Москва-2018

Курсовой проект

Цель курсового проекта

Приобретение практических навыков в использовании знаний, полученных в течении курса

Проведение исследования в выбранной предметной области

Задание

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.

Вариант

Консоль-серверная игра шашки. В игре должна быть поддержка нескольких игр сразу (если на сервер зашло 7 человек, то 6 игроков начали игру, а один игрок ждет своего соперника)
При выходе одного игрока - ему засчитывается поражение.
Должен вестись рейтинг всех игроков, которые играли и считаться их "win rate".
При разрыве соединения с сервером клиенты дожидаются пока сервер восстановит связь.

Код

```
import zmq
import sys
import time

board = [
[' ', 'o', ' ', 'o', ' ', 'o', ' ', 'o'],
['o', ' ', 'o', ' ', 'o', ' ', 'o', ' '],
[' ', 'o', ' ', 'o', ' ', 'o', ' ', 'o'],
[' ', ' ', '', ' ', ' ', ' ', ' ', ' '],
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
['*', ' ', '*', ' ', '*', ' ', '*', ' '],
[' ', '*', ' ', '*', ' ', '*', ' ', '*'],
['*', ' ', '*', ' ', '*', ' ', '*', ' ']]
```

```
is_waiting_opponent = False
is_waiting_move = False
iid = -1
color = ""
op_color =""

def change_board(list_cmd, board, color):
    i = 3
    while i < len(list_cmd):
        int1 = list_cmd[i-3]
        int2 = list_cmd[i-2]
        int3 = list_cmd[i-1]
        int4 = list_cmd[i]
        i += 2

    board[int1][int2] = ' '
    if abs(int1 - int3) == 2:
        check1 = 0
        check2 = 0
        if int3 > int1:
            check1 = int1 + 1
        else:
            check1 = int1 - 1
        if int4 > int2:
            check2 = int2 + 1
        else:
            check2 = int2 - 1
    board[check1][check2] = ' '

    if color == "w":
        board[list_cmd[-2]][list_cmd[-1]] = 'o'
    else:
        board[list_cmd[-2]][list_cmd[-1]] = '*'

def print_help():
    print("to make move type 'A3 B4'.")

def print_board(board):
    print("-------------------------------------------")
    print("... A .. B .. C .. D .. E .. F .. G .. H ...")
    print("-------------------------------------------")
    i = 1
    for box in board:
        print(i, "| ", end = "")
        i = i + 1
        for item in box:
            print(item, " | ", end='')
            print(i-1)
     print("-------------------------------------------")
```

```
    print("... A .. B .. C .. D .. E .. F .. G .. H ...")
    print("-----------------------------------------------")

def break_game():
    message = "-" + str(iid) + name
    socket.send_string(message)
    message = socket.recv()
    messageu = str(message, 'utf-8')
    print(messageu)
    exit()

def checknmove_cmd(list_cmd, flag, board, color, is_me):


    for item in list_cmd:
        if item < 0 or item > 7:
            print("Wrong command!")
            return False
        if color == "w" and is_me and (board[list_cmd[0]][list_cmd[1]] != 'o' and
board[list_cmd[0]][list_cmd[1]] != 'O'):
            print("Not your fishka!")
            return False
        elif color == "b" and is_me and (board[list_cmd[0]][list_cmd[1]] != '*' and
board[list_cmd[0]][list_cmd[1]] != '@'):
            print("Not your fishka!")
            return False

        i = 2
        while i < len(list_cmd):
            check1 = list_cmd[i]
            i += 1
            check2 = list_cmd[i]
            i += 1

        if board[check1][check2] != ' ':
            print("Wrong destination!")
            is_norm_cmd = False
            return False

        i = 3

    if flag == "one":
        int1 = list_cmd[i-3]
        int2 = list_cmd[i-2]
        int3 = list_cmd[i-1]
        int4 = list_cmd[i]


        if abs(int1 - int3) != abs(int2 - int4):
            print("Dont you know the rules?")
```

```
            return False
        if abs(int1 - int3) > 2:
            print("You cant move so far!")
            return False

        if abs(int1 - int3) == 2:
            check1 = 0
            check2 = 0
            if int3 > int1:
                check1 = int1 + 1
            else:
                check1 = int1 - 1
            if int4 > int2:
                check2 = int2 + 1
            else:
                check2 = int2 - 1
            if color == "w" and is_me:
                if board[check1][check2] != '*' and board[check1][check2] != '@':
                        print("You dont kill enemy!!!")
                        return False
            elif color == "b" and is_me:
                if board[check1][check2] != 'o' and board[check1][check2] != 'O':
                        print("You dont kill enemy!!!")
                        return False

        board[check1][check2] = ' '

# checking vector
    elif board[int1][int2] == 'o': # down
                if int1 >= int3:
            print("Prikaz 227: NI SHAGU NAZAD!")
            return False
    elif board[int1][int2] == 'O':
    if int1 <= int3:
print("Prikaz 228: NI SHAGU VPERED!")
return False
elif board[int1][int2] == '*': # up
if int1 <= int3:
print("Prikaz 227: NI SHAGU NAZAD!")
return False
elif board[int1][int2] == '@':
if int1 >= int3:
print("Prikaz 228: NI SHAGU VPERED!")
return False

if board[int1][int2] == 'o':
if int3 == 7:
board[int3][int4] = 'O'
else:
board[int3][int4] = 'o'
```

```
elif board[int1][int2] == '*':
if int3 == 0:
board[int3][int4] = '@'
else:
board[int3][int4] = '*'
elif board[int1][int2] == 'O':
if int3 == 0:
board[int3][int4] = 'o'
else:
board[int3][int4] = 'O'
elif board[int1][int2] == '@':
if int3 == 7:
board[int3][int4] = '*'
else:
board[int3][int4] = '@'

board[int1][int2] = ' '



else:

while i < len(list_cmd) - 1:
int1 = list_cmd[i-3]
int2 = list_cmd[i-2]
int3 = list_cmd[i-1]
int4 = list_cmd[i]
i += 4
if abs(int1 - int3) != abs(int2 - int4):
print("Dont you know the rules?")
return False
if abs(int1 - int3) > 2:
print("You cant move so far!")
return False

if abs(int1 - int3) < 2:
print("Wrong move!")
return False

check1 = 0
check2 = 0
if int3 > int1:
check1 = int1 + 1
else:
check1 = int1 - 1
if int4 > int2:
check2 = int2 + 1
else:
check2 = int2 - 1
if color == "w" and is_me:
if board[check1][check2] != '*' or board[check1][check2] != '@':
```

```
print("You dont kill f enemy!!!")
return False
elif color == "b" and is_me:
if board[check1][check2] != 'o' or board[check1][check2] != 'O':
print("You dont kill f enemy!!!")
return False

if board[int1][int2] == 'o':
if int3 == 7:
board[int3][int4] = 'O'
else:
board[int3][int4] = 'o'
elif board[int1][int2] == '*':
if int3 == 0:
board[int3][int4] = '@'
else:
board[int3][int4] = '*'
elif board[int1][int2] == 'O':
if int3 == 0:
board[int3][int4] = 'o'
else:
board[int3][int4] = 'O'
elif board[int1][int2] == '@':
if int3 == 7:
board[int3][int4] = '*'
else:
board[int3][int4] = '@'

board[int1][int2] = ' '

return True



def change_list(change):
i = 0
while i < len(change):
change[i], change[i+1] = change[i+1], change[i]
i = i + 2

newchange = []
newchange.append(change[0])
i = 1
while i < len(change):
newchange.append(change[i])
newchange.append(change[i])
i = i + 1
change = newchange
if __name__ == "__main__":
```

```
port = "5556"
context = zmq.Context()
print("Connecting to server...")
socket = context.socket(zmq.REQ)
socket.connect ("tcp://localhost:%s" % port)

message = "%"
socket.send_string(message)
message = socket.recv()
messageu = str(message, 'utf-8')
print(messageu)


# processing of name
name = ""
while name == "":
name = input("Please, enter your Name: ")
if name == "":
print("Name NOT PUSTOE!!!")

message = "+" + name
socket.send_string(message)


while True:
message = socket.recv()
messageu = str(message, 'utf-8')
#print("Received reply {}".format(message))
# main loop
if messageu == "Sorry, but this name already taken.":
print(messageu)
name = input("Please, Enter new name: ")
message = "+" + name
socket.send_string(message)


elif messageu == "Waiting opponent...":
color = "w"
op_color = "b"
is_waiting_move = False
if not is_waiting_opponent:
is_waiting_opponent = True
print("Waiting opponent.")
time.sleep(1)
message = "?" + name
socket.send_string(message)
```

```
elif messageu[:27] == "Waiting for opponents move.":
if len(messageu) > 27:
iid = messageu[27]
color = messageu[28]
if color == "w":
op_color = "b"
else:
op_color = "w"
print("Your color is Black")

is_waiting_opponent = False
if not is_waiting_move:
is_waiting_move = True
print("Waiting for opponents move.")
time.sleep(1)
message = "*" + str(iid) + name
socket.send_string(message)


elif messageu[0] == "i":

iid = int(messageu[1])
op_cmd = messageu[2:]
op_list_cmd = []

if len(messageu) == 2:
if color == "b":
print("Your color is Black!")
else:
print("Your color is White!")

else:
i = 0
while i < len(op_cmd):
#print(i)
#print(len(op_cmd))
op_list_cmd.append(int(op_cmd[i]))
i = i + 1
op_list_cmd.append(int(op_cmd[i]))
i = i + 1
#print(op_list_cmd)
if len(op_list_cmd) > 4:
checknmove_cmd(op_list_cmd, "many", board, op_color, False)
else:
checknmove_cmd(op_list_cmd, "one", board, op_color, False)

is_waiting_opponent = False
is_waiting_move = False
print_help()
print_board(board)
```

```
is_norm_cmd = False
list_cmd = []

while not is_norm_cmd:
list_cmd = []
cmd = input("Your turn to move: ")
if cmd == "exit":
break_game()

i = 0
while i < len(cmd):
list_cmd.append(ord(cmd[i]) - 65)
i+=1
list_cmd.append(int(cmd[i]) - 1)
i+=2

change_list(list_cmd)

#print(list_cmd)
if len(list_cmd) > 4:
is_norm_cmd = checknmove_cmd(list_cmd, "many", board, color, True)
else:
is_norm_cmd = checknmove_cmd(list_cmd, "one", board, color, True)
print_board(board)
cmd = "!" + str(iid)
for item in list_cmd:
cmd = cmd + str(item)

#print(cmd)
socket.send_string(cmd)
elif messageu == "Wrong command":
print(messageu)
elif messageu[:3] == "You":
print(messageu)
exit()

else:
print("F")
exit()
```

Вывод

Шашки оказалась очень сложной игрой для реализации (если использовать все правила игры) и, как оказалось, очень удобной для использования через модель клиент-клиент. Доска для игры хранится на стороне игроков, тем самым мы разгружаем сервер по памяти. Также мы разгружаем сервер передачей контроля за ходом на сторону сервера. Это очень важные шаги, так как сервер всего один и поэтому для быстрого взаимодействия важно максимально разгрузить его. У сервера есть небольшой минус: так как я не реализовал резервное копирование, то если он упадет, будет утерян весь прогресс игроков (винрейт).