Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовой проект по курсу**

**«Операционные системы»**

**Клиент-серверная система для передачи мгновенных сообщений**

Студент: Бутырев Д. В.

Группа: М80-206Б-20

Преподаватель: Соколов Андрей Алексеевич

Дата: \_\_\_\_\_\_\_\_\_\_\_

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

1. **Цель курсового проекта**

1. Приобретение практических навыков в использовании знаний, полученных в течении курса

2. Проведение исследования в выбранной предметной области

1. **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.

1. **Постановка задачи**

Необходимо написать 3-и программы. Далее будем обозначать эти программы A, B, C. Программа A принимает из стандартного потока ввода строки, а далее их отправляет программе С. Отправка строк должна производится построчно. Программа C печатает в стандартный вывод, полученную строку от программы A. После получения программа C отправляет программе А сообщение о том, что строка получена. До тех пор пока программа А не примет «сообщение о получение строки» от программы С, она не может отправлять следующую строку программе С. Программа B пишет в стандартный вывод количество отправленных символов программой А и количество принятых символов программой С. Данную информацию программа B получает от программ A и C соответственно. Способ организация межпроцессорного взаимодействия выбирает студент.

1. **Листинг программы**

A.cpp:

#include <iostream>

#include <string>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/mman.h>

#include <unistd.h>

#include <fcntl.h>

#include <cstdio>

#include <cstdlib>

#include <errno.h>

#include "interconnection.hpp"

//1024 byte - string; 1 byte - A status or not; 4 byte - B from A; 4 byte - C from A

int main() {

int fd = open( "./.file", O\_RDWR | O\_CREAT | O\_TRUNC , S\_IRWXU );

write( fd, "", sizeof( "" ) );

memory\* mem = ( memory\* )mmap( 0, sizeof( memory ), PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0 );

close( fd );

if( !mem ) return 2;

mem->status |= A\_READY | A\_TURN;

mem->AtoB = 0;

mem->CtoB = 0;

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

while( ( mem->status & ALL\_READY ) != ALL\_READY ) {

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

}

bool all\_good = true;

while( true ) {

while( !( mem->status & A\_TURN ) ) {

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

if( mem->status & SOMEONE\_CLOSED ) {

all\_good = false;

break;

}

}

if( !all\_good ) break;

char ch;

int count = 0;

std::putc( '>', stdout );

while( ( ( ch = std::getc( stdin ) ) != '\n' ) && ( ch != EOF ) && ( count < 1024 ) ) {

mem->line[ count++ ] = ch;

}

if( ch == EOF ) break;

mem->line[ count ] = '\0';

mem->AtoB = count + 1; //all chars = count + \0

mem->status &= ~A\_TURN;

mem->status |= C\_TURN;

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

}

mem->status |= A\_CLOSED;

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

munmap( &mem, sizeof( memory ) );

return 0;

}

B.cpp:

#include <iostream>

#include <string>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/mman.h>

#include <fcntl.h>

#include <cstdio>

#include <cstdlib>

#include <unistd.h>

#include <errno.h>

#include "interconnection.hpp"

//1024 byte - string; 1 byte - A wait or not; 4 byte - B from A; 4 byte - C from A

int main() {

int fd = open( "./.file", O\_RDWR /\* | O\_CREAT | O\_TRUNC \*/, S\_IRWXU );

write( fd, "", sizeof( "" ) );

memory\* mem = reinterpret\_cast< memory\* >( mmap( 0, sizeof( memory ), PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0 ) );

close( fd );

if( !mem ) return 2;

mem->status = mem->status | B\_READY;

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

while( ( mem->status & ALL\_READY ) != ALL\_READY ) {

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

}

bool all\_good = true;

while( true ) {

while( !( mem->status & B\_TURN ) ) {

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

if( mem->status & SOMEONE\_CLOSED ) {

all\_good = false;

break;

}

}

if( !all\_good ) break;

std::cout << "B: A sent " << mem->AtoB << " signs\n";

mem->AtoB = 0;

std::cout << "B: C got " << mem->CtoB << " signs\n";

mem->CtoB = 0;

mem->status &= ~B\_TURN;

mem->status |= A\_TURN;

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

}

mem->status |= B\_CLOSED;

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

munmap( &mem, sizeof( memory ) );

return 0;

}

C.cpp:

#include <iostream>

#include <string>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/mman.h>

#include <fcntl.h>

#include <cstdio>

#include <cstdlib>

#include <unistd.h>

#include <errno.h>

//1024 byte - string; 1 byte - A wait or not; 4 byte - B from A; 4 byte - C from A

#include "interconnection.hpp"

int main() {

int fd = open( "./.file", O\_RDWR /\* | O\_CREAT | O\_TRUNC \*/, S\_IRWXU );

write( fd, "", sizeof( "" ) );

memory\* mem = reinterpret\_cast< memory\* >( mmap( 0, sizeof( memory ), PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0 ) );

close( fd );

if( !mem ) return 2;

for( int i = 0; i < sizeof( memory ); ++i ) {

( int )( ( ( char\* )mem )[ i ] );

}

mem->status = mem->status | C\_READY;

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

while( ( mem->status & ALL\_READY ) != ALL\_READY ) {

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

}

bool all\_good = true;

while( true ) {

while( !( mem->status & C\_TURN ) ) {

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

if( mem->status & SOMEONE\_CLOSED ) {

all\_good = false;

break;

}

}

if( !all\_good ) break;

std::cout << "C: " << mem->line << '\n';

int count = 0;

while( mem->line[ count++ ] && ( count < 1024 ) );

mem->CtoB = count;

mem->status &= ~C\_TURN;

mem->status |= B\_TURN;

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

}

mem->status |= C\_CLOSED;

if( msync( mem, sizeof( memory ), MS\_INVALIDATE ) ) {

std::cerr << errno << '\n';

return errno;

}

munmap( &mem, sizeof( memory ) );

return 0;

}

interconnection.hpp:

#ifndef INTERCONNECTION\_H\_

#define INTERCONNECTION\_H\_

enum interconnection : unsigned {

A\_READY = 1u,

B\_READY = 1u << 1,

C\_READY = 1u << 2,

ALL\_READY = A\_READY | B\_READY | C\_READY,

A\_CLOSED = 1u << 3,

B\_CLOSED = 1u << 4,

C\_CLOSED = 1u << 5,

SOMEONE\_CLOSED = A\_CLOSED | B\_CLOSED | C\_CLOSED,

A\_TURN = 1u << 6,

B\_TURN = 1u << 7,

C\_TURN = 1u << 8,

};

struct memory {

char line[ 1024 ];

unsigned status;

int AtoB;

int CtoB;

};

#endif // INTERCONNECTION\_H\_

makefile:

##

# Project Title

#

# @file

# @version 0.1

compile: A B C

A: A.cpp

g++ -g -Wall -pedantic -o A A.cpp

B: B.cpp

g++ -g -Wall -pedantic -o B B.cpp

C: C.cpp

g++ -g -Wall -pedantic -o C C.cpp

clean:

rm A B C .file

# end

1. **Пример работы программы**

steep@gg:~/education\_disk/instit/os/kp/scr$ ./A &

[2] 38962

steep@gg:~/education\_disk/instit/os/kp/scr$ ./B &

[3] 38965

steep@gg:~/education\_disk/instit/os/kp/scr$ ./C &

[4] 38966

steep@gg:~/education\_disk/instit/os/kp/scr$ >fg ./A

./A

aaaaa

C: aaaaa

B: A sent 6 signs

B: C got 6 signs

>bbbbbbb

C: bbbbbbb

B: A sent 8 signs

B: C got 8 signs

>444444

C: 444444

B: A sent 7 signs

B: C got 7 signs

>exit

C: exit

B: A sent 5 signs

B: C got 5 signs

>[4]- Завершён ./C

steep@gg:~/education\_disk/instit/os/kp/scr$

[3]+ Завершён ./B

1. **Выводы**

Можно достаточно просто реализовать межпроцессное взаимодействие на одном компьютере с помощью shared memory.