МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Н.Э. БАУМАНА

ОТЧЕТ  
по лабораторным работам  
по курсу «Разработка программных систем»

Выполнил: студент группы РК6-113  
Минаев М.И.  
Преподаватель: Федорук В.Г.

Москва  
2014 год

# Лабораторная работа по многопроцессорному программированию

**Вариант 12**

**Текст задания на лабораторную работу**

Разработать программу, реализующую обработку текстовых файлов и функционирующую в рамках 3-х процессов.  
Корневой процесс является управляющим и принимает в качестве аргументов имена 2-х файлов. В начале своей работы он порождает 2 процесса, передавая каждому по одному имени файла.  
Первый порожденный процесс осуществляет побайтное считывание файла и вывод с небольшой задержкой прочитанных байт (по одному на строке) в верхнем регистре на стандартный вывод.  
Второй порожденный процесс осуществляет побайтное считывание файла и вывод с небольшой задержкой прочитанных байт (по одному на строке) в нижнем регистре на стандартный вывод.  
Управляющий процесс считывает со стандартного ввода строки, содержащие имена новых текстовых файлов и заставляет порожденные процессы переходить на обработку новых файлов по следующей схеме: второй порожденный процесс переходит к обработке файла, ранее обрабатываемого первым процессом; первый порожденный процесс начинает обрабатывать файл, имя которого ему передает управляющий процесс.

**Описание структуры программы и реализованных способов взаимодействия процессов.**

file\_name1  
file\_name2

Корневой процесс

file\_name2

file\_name1

2 порожденный процесс

1 порожденный процесс

После того как корневой процесс прочитает новое имя файла, он пошлет сигнал 1му порожденному процессу, который поменяет свой файл. Затем 1ый порожденный процесс пошлет сигнал 2му порожденному процессу который должен изменить свой файл для вывода.

file\_name3 from stdin

Корневой процесс

1 порожденный процесс

file\_name3

file\_name1

2 порожденный процесс

**Описание основных используемых структур данных**

Для обмена именами файлов между процессоров используется pipe. Для посылки сообщения 1 порожденному процессу используется сигнал SIGUSR1. После получения этого сигнала 1 порожденный процесс читает имя файла из пайпа, записывает свое старое имя файла в другой пайп для отправки 2 порожденному процессу, и отправляет сигнал 2 порожденному процессу, затем начинает обработку нового файла. Для посылки сообщения 2 порожденному процессу используется сигнал SIGUSR2. После получения этого сигнала 2 порожденный читает имя файла и приступает к его обработке.

**Примеры результатов работы программы**

**./a.out text1.txt text2.txt**

**3179 3180**

**2 process pid is 3180**

**1 process pid is 3179**

**z**

**A**

**x**

**B**

**c**

**C**

**v**

**A**

**z**

**B**

**x**

**C**

**v**

**text1.txt**

**A**

**a**

**b**

**B**

**c**

**C**

**a**

**A**

**b**

**B**

**c**

**C**

**Текст программы с исчерпывающими комментариями**

|  |
| --- |
| #include <stdio.h> |
|  | #include <stdlib.h> |
|  | #include <unistd.h> |
|  | #include <sys/types.h> |
|  | #include <sys/uio.h> |
|  | #include <fcntl.h> |
|  | #include <string.h> |
|  | #include <signal.h> |
|  | #include <sys/mman.h> |
|  | #define SWITCHED 1 |
|  | #define BUF\_SIZE 64 |
|  | #define FILE\_NAME\_SIZE 256 |
|  | int fd1, fd2; |
|  | int pipe1[2], pipe2[2]; |
|  | pid\_t pid1; |
|  | pid\_t pid2; |
|  | static pid\_t \*pid2\_t; |
|  | char file1[FILE\_NAME\_SIZE]; |
|  |  |
|  | void switch\_files(int sig\_type) |
|  | { |
|  | if (sig\_type == SIGUSR1) |
|  | { |
|  | write(pipe2[1], file1, strlen(file1)+1); |
|  | if (switch\_file(pipe1[0], &fd1, 1) == SWITCHED) |
|  | { |
|  | kill(\*pid2\_t, SIGUSR2); |
|  | } |
|  |  |
|  | } |
|  | if (sig\_type == SIGUSR2) |
|  | { |
|  | switch\_file(pipe2[0], &fd2, 2); |
|  | } |
|  | } |
|  |  |
|  | int switch\_file(int checking\_desc, int \*switching\_desc, int p\_idx) |
|  | { |
|  | char buf[FILE\_NAME\_SIZE]; |
|  | if (read(checking\_desc, buf, FILE\_NAME\_SIZE) > 0) |
|  | { |
|  | int openning = open(buf, O\_RDONLY); |
|  | if (openning > 0) |
|  | { |
|  | if (p\_idx == 1) |
|  | { |
|  | strncpy(file1, buf, FILE\_NAME\_SIZE); |
|  | //printf("file1 %s\n", file1); |
|  | } |
|  | close(\*switching\_desc); |
|  | \*switching\_desc = openning; |
|  | } |
|  | else |
|  | { |
|  | perror(buf); |
|  | return -1; |
|  | } |
|  | return SWITCHED; |
|  | } |
|  | return 0; |
|  | } |
|  |  |
|  | void terminate(int sig\_type) |
|  | { |
|  | \_exit(0); |
|  | } |
|  | int main(int argc, char \*\*argv) |
|  | { |
|  | if (argc != 3) |
|  | { |
|  | fprintf(stderr, "give 2 args\n"); |
|  | exit(-1); |
|  | } |
|  | pid2\_t = mmap(NULL, sizeof(\*pid2\_t), PROT\_READ | PROT\_WRITE, MAP\_SHARED | MAP\_ANONYMOUS, -1, 0); |
|  | pid\_t lpid1, lpid2; |
|  | char file2[FILE\_NAME\_SIZE]; |
|  | strncpy(file2, argv[2], FILE\_NAME\_SIZE); |
|  | char input[256]; |
|  | strncpy(file1, argv[1], FILE\_NAME\_SIZE); |
|  | pipe(pipe1); |
|  | pipe(pipe2); |
|  |  |
|  | signal(SIGINT, terminate); |
|  |  |
|  | if ((lpid1 = fork()) == 0) |
|  | { |
|  | int read\_bytes = 0; |
|  | char c; |
|  | printf("1 process pid is %d\n", getpid()); |
|  | signal(SIGUSR1, switch\_files); |
|  | fd1 = open(file1, O\_RDONLY); |
|  | if (fd1 < 0) |
|  | { |
|  | fprintf(stderr, "error opening file %s\n", file1); |
|  | return -1; |
|  | } |
|  | while(1) |
|  | { |
|  | read\_bytes = read(fd1, &c, 1); |
|  | if (read\_bytes > 0) |
|  | { |
|  | fprintf(stdout, "%c\n", toupper(c)); |
|  | } |
|  | usleep(500000); |
|  | } |
|  | close(fd1); |
|  | printf("fd1 closed\n"); |
|  | return 0; |
|  | } else { |
|  | pid1 = lpid1; |
|  | } |
|  |  |
|  | if ((lpid2 = fork()) == 0) |
|  | { |
|  | int read\_bytes = 0; |
|  | char c; |
|  | printf("2 process pid is %d\n", getpid()); |
|  | signal(SIGUSR2, switch\_files); |
|  | fd2 = open(file2, O\_RDONLY); |
|  | if (fd2 < 0) |
|  | { |
|  | fprintf(stderr, "error opening file %s\n", file2); |
|  | return -1; |
|  | } |
|  | while(1) |
|  | { |
|  | read\_bytes = read(fd2, &c, 1); |
|  | if (read\_bytes > 0) |
|  | { |
|  | fprintf(stdout, "\t%c\n", tolower(c)); |
|  | } |
|  | usleep(500000); |
|  | } |
|  | close(fd2); |
|  | printf("fd2 closed\n"); |
|  | return 0; |
|  | } else { |
|  | \*pid2\_t = lpid2; |
|  | pid2 = lpid2; |
|  | } |
|  |  |
|  | printf("%d %d\n", pid1, pid2); |
|  | while(scanf("%s", input) > 0) |
|  | { |
|  | write(pipe1[1], input, strlen(input) + 1); |
|  | kill(pid1, SIGUSR1); |
|  | } |
|  |  |
|  | waitpid(pid1, 0, 0); |
|  | waitpid(pid2, 0, 0); |
|  | printf("exit\n"); |
|  | } |

# Лабораторная работа по многопоточному программированию

**Вариант 13**

**Текст задания на лабораторную работу**

Разработать многопотоковый вариант программы моделирования распространения электрических сигналов в двухмерной прямоугольной сетке RC-элементов (одномерный аналог которых представлен на рис. выше). Метод формирования математической модели - узловой. Метод численного интегрирования - **явный** Эйлера. Внешнее воздействие - источники тока и напряжения. Количество потоков, временной интервал моделирования и количество элементов в сетке - параметры программы. Программа должна демонстрировать [ускорение](http://fedoruk.comcor.ru/Dev_bach/mthread_lab.html#par_effect) по сравнению с последовательным вариантом. Предусмотреть визуализацию результатов посредством утилиты gnuplot.

**Описание структуры программы и реализованных способов взаимодействия потоков управления**

При старте запускается N потоков, каждый поток получает диапазон индексов для обработки. На каждом шаге вычисления поток выполняет необходимые вычисления и ждет завершения вычислений в других потоках, с помощью pthread\_barrier\_wait. После того как все потоки закончат вычисления начинается этап копирования результатов. После завершения копирования поток ждет завершения копирования всех остальных потоков, с помощью pthread\_barrier\_wait. Дальше начинается следующая итерация.

**Блок-схема программы и пояснения к ней**

args

Главный процесс  
Запускает N потоков

…

N

1

2

N-1

Вычисления в потоке N

Вычисления в потокеN-1

Вычисления в потоке 2

Вычисления в потоке 1

Ожидание выполнения вычислений всех потоков

Копирование

Копирование

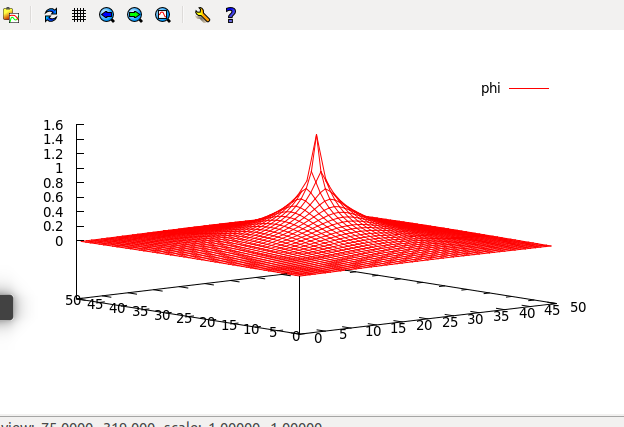
Копирование

Копирование

Следующая итерация

Ожидание выполнения вычислений всех потоков

**Примеры результатов работы программы**



**Текст программы с исчерпывающими комментариями**

#include <iostream>

#include <vector>

#include <getopt.h>

#include <stdlib.h>

#include <cmath>

#include <pthread.h>

#include <unistd.h>

#include <time.h>

#include <utility>

#include <fstream>

#include <sstream>

#ifndef NOGNUPLOT

std::string data\_file\_name = "data.out";

std::ofstream gnuplot("cmd", std::ofstream::trunc | std::ofstream::out);

std::ofstream out(data\_file\_name.c\_str(), std::ofstream::out | std::ofstream::trunc);

std::vector<std::pair<int,int> > gnuplot\_points;

#endif

typedef std::vector<std::vector<double> > matrix;

typedef std::vector<double> row;

std::vector<std::vector<double> > phi, I;

std::vector<int> condition;

static int THREADS, X\_SIZE, Y\_SIZE;

pthread\_barrier\_t bar, syn, val;

pthread\_mutex\_t lock;

static double R = 1, C = 1e-3, dt, T;

long long clock\_time() {

return time(NULL);

}

void print\_matrix(const matrix &v) {

int size = v.size();

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

for(int j = 0; j < v[i].size(); j++) {

std::cout << v[i][j] << " ";

}

std::cout << std::endl;

}

}

void print\_matrix(const row &v) {

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

std::cout << v[i] << std::endl;

}

}

row solve\_progon(const matrix &A,const row &B) {

row P, Q, ans;

int N = B.size();

P.resize(N);

Q.resize(N);

ans.resize(N);

P[0] = - A[0][1] / A[0][0];

Q[0] = B[0]/A[0][0];

//Pr9moy xod

for(int i = 1; i < N; i++) {

P[i] = A[i][i + 1]/(-A[i][i] - A[i][i - 1]\*P[i-1]);

Q[i] = (A[i][i - 1] \* Q[i - 1] - B[i]) / (-A[i][i] - A[i][i-1] \* P[i - 1]);

}

//Obratniy xod

ans[N - 1] = (A[N-1][N-2] \* Q[N-2] - B[N-1]) / (-A[N-1][N-1] - A[N-1][N-2] \* P[N-2]);

for(int i = N - 2; i >= 0; --i) {

ans[i] = P[i] \* ans[i+1] + Q[i];

}

return ans;

}

row check\_solution(const matrix &A , const row &X, const row &B) {

row diff;

diff.resize(B.size());

int size = B.size();

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

double temp = 0;

for(int j = 0; j < size; j++) {

temp += A[i][j] \* X[j];

}

diff[i] = temp - B[i];

}

return diff;

}

row solve(const row &left,const row &mid,const row &right, int j) {

int size = left.size();

double pt, bt;

row B(size);

matrix A(size, row(size));

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

if(i - 1 >=0)

A[i][i-1] = 1;

A[i][i] = -2;

if(i + 1 < A.size())

A[i][i+1] = 1;

if(i == 0)

pt = 0;

else

pt = mid[i-1];

if (i == A.size() - 1)

bt = 0;

else

bt = mid[i+1];

B[i] = (-left[i] + 2 \* mid[i] - right[i]) \* dt /(R \* C) + (pt - 2 \* mid[i] + bt) + I[j][i] \* dt/C;//(left[i] - 2 \* mid[i] + right[i]) \* (dt / (R \* C) - 1) + I[j][i] \* dt / C ;

}

row s = solve\_progon(A, B);

return s;

}

void \* calc(void \*thread) {

long t = (long) thread;

int start\_index = t \* (X\_SIZE / THREADS);

int end\_index = (t != THREADS - 1)?(t + 1) \* (X\_SIZE / THREADS) - 1: X\_SIZE - 1;

matrix local(phi.size()), next(phi.size(),row(phi[0].size()));

row zeros;

for (int i = 0; i < Y\_SIZE; i++) {

zeros.push\_back(0);

}

double cur\_time = 0, a = 0;

int x, y;

int counter = 0;

unsigned long long iters = 0;

std::cout << "I am thread " << t << ". My start index: " << start\_index << ". My end index: " << end\_index << std::endl;

row left\_t, mid\_t, right\_t;

unsigned long long start = clock\_time();

double mid, right, left, top, bot;

while (cur\_time < T) {

//for(int i = 0; i < 3; i ++) {

pthread\_barrier\_wait(&syn);

#ifndef NOGNUPLOT

if (t == 0) {

if (cur\_time > (double) T / 4)

I[25][25] = 0;

if (cur\_time - a > 1e-2) {

a = cur\_time;

std::stringstream convert;

convert << counter;

std::string num = convert.str();

std::ofstream file((data\_file\_name + num).c\_str(), std::ofstream::out|std::ofstream::trunc);

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

for(int j = 0; j < phi[i].size(); j++) {

file << i << " " << j << " " << phi[i][j] << std::endl;

}

}

file.close();

counter ++;

gnuplot << "splot \"" << data\_file\_name + num << "\"" << "with lines title \"phi\"\n";

gnuplot << "pause 0.1" << std::endl;

}

}

/\*out << cur\_time << " ";

for (int i = 0; i < gnuplot\_points.size(); i++) {

x = gnuplot\_points[i].first;

y = gnuplot\_points[i].second;

if (x >= 0 && x < X\_SIZE && y >=0 && y < Y\_SIZE)

out << phi[x][y] << " ";

}

}

}

out << std::endl;\*/

#endif

cur\_time += dt;

/\*for (int i = start\_index; i <= end\_index; i ++) {

iters ++;

mid\_t = phi[i];

if (i == start\_index && start\_index == 0) {

left\_t = zeros;

right\_t = phi[i+1];

}

else if (i == end\_index && end\_index == X\_SIZE - 1) {

left\_t = phi[i - 1];

right\_t = zeros;

}

else {

left\_t = phi[i - 1];

right\_t = phi[i + 1];

}\*/

for(int i = start\_index; i <= end\_index; i++){

for(int j = 0; j < phi[i].size(); j++){

mid = phi[i][j];

top = j == 0 ? 0:phi[i][j-1];

bot = j == Y\_SIZE - 1 ? 0 : phi[i][j+1];

left = i == start\_index && start\_index ==0 ? 0 : phi[i-1][j];

right = i == end\_index && end\_index == X\_SIZE - 1 ? 0 :phi[i+1][j];

next[i][j] = mid + (top+ bot + left + right - 4 \* mid) \* dt / (R \* C) + I[i][j] \* dt /C;

}

}

//next[i] = solve(left\_t, mid\_t, right\_t, i);

//}

pthread\_barrier\_wait(&bar);

for (int i = start\_index; i <=end\_index; i++) {

phi[i] = next[i];

}

}

pthread\_exit(NULL);

}

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

int time\_to\_modulate;

int num\_threads;

int x\_grid, y\_grid;

const char\* short\_options = "tpxy";

int c;

T = atoi(argv[4]);

dt = 1e-6;

pthread\_attr\_t attr;

pthread\_attr\_init(&attr);

pthread\_attr\_setdetachstate(&attr, PTHREAD\_CREATE\_JOINABLE);

x\_grid = atoi(argv[1]);

y\_grid = atoi(argv[2]);

num\_threads = atoi(argv[3]);

#ifndef NOGNUPLOT

/\*int print\_points = atoi(argv[5]);

std::cout << "Enter " << print\_points << " nodes to print in format <m> <n>" << std::endl;

int x, y;

for(int i = 0; i < print\_points; i++) {

std::cin >> x >> y;

gnuplot\_points.push\_back(std::make\_pair(x, y));

}\*/

#endif

THREADS = num\_threads;

time\_to\_modulate = 2;

X\_SIZE = x\_grid;

Y\_SIZE = y\_grid;

phi.resize(x\_grid);

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

phi[i].resize(y\_grid);

}

I.resize(x\_grid);

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

I[i].resize(y\_grid);

for(int j = 0; j < y\_grid; j++)

I[i][j] = 0;//rand() % 6 - 3;

}

I[25][25] = 2;

#ifndef NOGNUPLOT

gnuplot << "set size ratio 1\n";

gnuplot << "set hidden3d\n";

gnuplot << "set dgrid 50,50\n";

gnuplot << "set zrange [-3: 3]\n";

//int res = system("gnuplot -persist cmd");

#endif

// I[1][1] = 1;

// I[0][1] = -1;

pthread\_barrier\_init(&bar, NULL, THREADS);

pthread\_barrier\_init(&val, NULL, THREADS);

pthread\_barrier\_init(&syn, NULL, THREADS);

pthread\_mutex\_init(&lock, NULL);

pthread\_t \*threads = new pthread\_t[THREADS];

unsigned long long start = clock\_time();

for (long i = 0; i < THREADS; i++) {

if (pthread\_create(&threads[i], NULL, calc, (void \*)i) != 0) {

std::cout << "Can't create thread " << i << std::endl;

}

}

for (int i = 0; i < THREADS; i++) {

pthread\_join(threads[i], NULL);

}

std::cout << "It takes " << (clock\_time() - start) << std::endl;

return 0;

}

# . Лабораторная работа №3. «Сетевое программирование»

**Задание (Вариант №8)**

Разработать программу-"получатель" электрической почты по протоколу POP-3 из почтового ящика на POP3-сервере. Программа должна обеспечивать минимальную функциональность UNIX-утилиты mail (без аргументов). Описание протокола POP-3 можно найти в Internet (например, www.citforum.ru), другой путь - использование сканеров сети типа tcpdump или snoop.

**Описание протокола POP-3.**

**POP3** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Post Office Protocol Version 3 — [протокол](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) почтового отделения, версия 3) — стандартный [Интернет-протокол прикладного уровня](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B_%D0%BF%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F), используемый [клиентами электронной почты](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%87%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) для получения почты с удаленного [сервера](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%87%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) по [TCP/IP](https://ru.wikipedia.org/wiki/TCP/IP)-соединению.

POP и [IMAP](https://ru.wikipedia.org/wiki/IMAP) (Internet Message Access Protocol) — наиболее распространенные [Интернет](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82)-протоколы для извлечения почты. Практически все современные клиенты и серверы электронной почты поддерживают оба стандарта. Протокол POP был разработан в нескольких версиях, нынешним стандартом является третья версия (POP3).

POP поддерживает простые требования «загрузи-и-удали» для доступа к удаленным почтовым ящикам. Хотя большая часть POP-клиентов предоставляет возможность оставить почту на сервере после загрузки, использующие POP клиенты обычно соединяются, извлекают все письма, сохраняют их на пользовательском компьютере как новые сообщения, удаляют их с сервера, после чего разъединяются

Доступные сообщения клиента фиксируются при открытии почтового ящика POP-сессией и определяются количеством сообщений для сессии, или, по желанию, с помощью уникального идентификатора, присваиваемого сообщению POP-сервером. Этот уникальный идентификатор является постоянным и уникальным для почтового ящика и позволяет клиенту получить доступ к одному и тому же сообщению в разных POP-сессиях. Почта извлекается и помечается для удаления с помощью номера сообщения. При выходе клиента из сессии помеченные сообщения удаляются из почтового ящика.

В протоколе POP3 предусмотрено 3 состояния сеанса:

[**Авторизация**](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F)

Клиент проходит процедуру [Аутентификации](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F).

[**Транзакция**](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))

Клиент получает информацию о состоянии почтового ящика, принимает и удаляет почту.

**Обновление**

Сервер удаляет выбранные письма и закрывает соединение.

**Описание основных структур данных**.

В программе используется структура sockaddr\_in которая имеет следующее описание

struct sockaddr\_in{

short sin\_family; Семейство адреса.

unsigned short sin\_port; Порт IP-адресов.

struct in\_addr sin\_addr; IP-адрес.

char sin\_zero[8]; Заполнение, что бы сделать структуру одного размера.

};

Для получения IP адреса по имени хоста написана функция get\_ip\_by\_hostname, в ней используются структуры struct in\_addr и struct hostent. Структура in\_addr представляет собой адрес интернета. Структура hostent используется функциями, чтобы хранить информацию о хосте: его имя, тип, IP адрес, и т.д.

**Блок-схема прогаммы**.

На каждом шаге предусмотрено при неудачном действии предусмотрен вывод диагностического сообщения об ошибке, в блок схеме упрощённо это отражает условный оператор перед последним блоком.



**Пример результатов работы программы.**

В качестве аргументов командной строки программа принимает: адрес почтового ящика, пароль для доступа к нему и адрес pop сервера. К примеру запуск программы может выглядеть следующим образом:

./pop3c minaev.mike@qip.ru passwd pop.qip.ru

На что программа ответит сообщением о удачном или неудачном подключении к серверу и удалось ли, используя пароль, получить доступ к почтовому ящику. Удачный запуск выглядит так:

Succsessfully get ip 80.68.248.170

Success login

<list> - to print list

<msg> <n> - get <n>th message

<exit> - exit

list

+OK

1 1615

2 2210

3 1627

4 2210

5 19524

6 2223

7 3093

8 2448

9 1611

10 1979

11 3844

12 3093

13 1154

14 29720

15 32093

16 33003

17 254535

18 16852

.

Exiting

msg 2

+OK Message follows

Received: from [62.141.94.163] (HELO test.mk.pochta.ru)

by node8-3.ks.pochta.ru with QIP.RU LMTP

for minaev.mike@qip.ru;

Thu, 22 Aug 2013 17:15:04 +0400

Received-SPF: pass (test.mk.pochta.ru: domain of pochta.ru designates 62.141.94.163 as permitted sender) client-ip=62.141.94.163; envelope-from=no-reply@pochta.ru; helo=pochta.ru;

X-QIP-Domain: pochta.ru

X-QIP-Sender: 62.141.94.163

Received: from test.mk.pochta.ru (pochta.ru [62.141.94.163] #r)

by test.mk.pochta.ru with QIP.RU SMTP

envelope from no-reply@pochta.ru

for minaev.mike@qip.ru; Thu, 22 Aug 2013 17:15:04 +0400

Received: from [62.141.94.155] (HELO mx5.ks.pochta.ru)

by node8-2.ks.pochta.ru with QIP.RU LMTP

for minaev.mike@qip.ru;

Thu, 22 Aug 2013 16:58:43 +0400

X-QIP-Domain: yandex.ru

X-QIP-Sender: 95.108.130.119

Received: from forward15.mail.yandex.net (forward15.mail.yandex.net [95.108.130.119] #t)

by mx5.ks.pochta.ru with QIP.RU SMTP

envelope from mike111133@yandex.ru

for minaev.mike@qip.ru; Thu, 22 Aug 2013 16:58:43 +0400

Received: from web18f.yandex.ru (web18f.yandex.ru [95.108.131.158])

by forward15.mail.yandex.net (Yandex) with ESMTP id EDCCB9E1ECE

for <minaev.mike@qip.ru>; Thu, 22 Aug 2013 16:58:42 +0400 (MSK)

Received: from 127.0.0.1 (localhost.localdomain [127.0.0.1])

by web18f.yandex.ru (Yandex) with ESMTP id 8AD521270006;

Thu, 22 Aug 2013 16:58:42 +0400 (MSK)

DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=yandex.ru; s=mail;

t=1377176322; bh=qoobdEabRDz0fOXh0opu+1esLIIa0wqk1cEdL0iNf6I=;

h=From:To:Subject:Date;

b=bZGSDEkFib5OJ490076Wfg3jEhh75sKvGKPwFnaW4H9bepBcRKCm8lXwADvOnEkzP

+gLRk3FFpn36GYg3YOS4KwQ5BR9iBYkdw1GB8j5Oi441juOfoxMAQN/Cn7SGuXLC9b

nSX2RNLuVUyoUNnw6Ej0f2mqw4mSHWp1eUwOUMY8=

Received: from ws-80-68-252-11.rbc.ru (ws-80-68-252-11.rbc.ru [80.68.252.11]) by web18f.yandex.ru with HTTP;

Thu, 22 Aug 2013 16:58:42 +0400

From: =?koi8-r?B?7TEg7TE=?= <mike111133@yandex.ru>

To: minaev.mike@qip.ru

Subject: hi

MIME-Version: 1.0

Message-Id: <227891377176322@web18f.yandex.ru>

X-Mailer: Yamail [ http://yandex.ru ] 5.0

Date: Thu, 22 Aug 2013 16:58:42 +0400

Content-Transfer-Encoding: 7bit

Content-Type: text/plain

hello my friends

.

ExitingПосле слова лист, выведены номер сообщения и соответствующий размер сообщения в байтах.

Программа подразумевает возможность вывода количества сообщений(list), отображение содержимого n-го сообщения (msg n) и удаление n-го сообщения (del n).

**Текст программы.**

#include <stdio.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <netdb.h>

#include <unistd.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

#include <sys/ioctl.h>

#define PORT 110

int get\_ip\_by\_hostname(char \*ip, char \*server) {

struct hostent \*he;

struct in\_addr \*\*addr\_list;

if ((he = gethostbyname(server)) == NULL) {

perror("GETHOSTBYNAME");

return 1;

}

addr\_list = (struct in\_addr \*\*) he->h\_addr\_list;

if (addr\_list[0] == NULL) {

perror("Addr list empty");

return 1;

}

else {

strcpy(ip, inet\_ntoa(\*addr\_list[0]));

printf("Succsessfully get ip %s\n", ip);

return 0;

}

}

void handle\_socket(int sockfd) {

int len;

int n;

char buf[1024];

n = recv(sockfd, buf, sizeof(buf) - 1, 0);

while (n > 0) {

buf[n] = '\0';

printf("%s", buf);

if (strstr(buf, "\r\n.\r\n") != NULL) {

break;

}

n = recv(sockfd, buf, sizeof(buf) - 1, 0);

}

printf("Exiting\n");

}

void print\_list(int sockfd) {

write(sockfd, "LIST\n", 5);

}

void disconnect(int sockfd) {

write(sockfd, "QUIT\n", 5);

}

void print\_mail(int sockfd, char \*num) {

write(sockfd, "RETR ", 5);

write(sockfd, num, strlen(num) - 1);

write(sockfd, "\n", 1);

}

void print\_help(char \*msg) {

printf("%s\n", msg);

printf("./pop3c <username> <password> <server>\n");

exit(0);

}

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

if (argc != 4)

print\_help("Wrong number of arguments");

struct sockaddr\_in serv\_addr;

int sockfd = 0;

char buf[1024];

char cmd[1024];

char ip[32];

int n, k;

if (get\_ip\_by\_hostname(ip, argv[3]) != 0) {

perror("Error by getting ip");

print\_help("Bad server");

}

if ((sockfd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

perror("Can't create socket");

return -1;

}

memset(&serv\_addr, 0, sizeof(serv\_addr));

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

if(inet\_pton(AF\_INET, ip, &serv\_addr.sin\_addr) <= 0) {

perror("Inet\_Pton error");

return -1;

}

if (connect(sockfd, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0) {

perror("Connect error");

return -1;

}

//wrte(sockfd, "\n", 1);

int state = 0;

char \*token;

while ((n = read(sockfd, buf, sizeof(buf))) > 0) {

if (strstr(buf, "+OK") != NULL) {

state++;

} else{

if (state == 1) {

print\_help("Bad username or password");

}

if (state == 2) {

print\_help("Bad username or password");

}

}

if (state == 1) {

write(sockfd, "USER ", 5);

write(sockfd, argv[1], strlen(argv[1]));

write(sockfd, "\n", 1);

//Write password

continue;

}

if (state == 2){

write(sockfd, "PASS ",5);

write(sockfd, argv[2], strlen(argv[2]));

write(sockfd, "\n", 1);

continue;

}

if (state == 3)

break;

}

printf("Success login\n\t<list> - to print list\n\t<msg> <n> - get <n>th message\n\t<exit> - exit\n");

while ((k = read(0, cmd, sizeof(cmd))) > 0) {

if (strstr(cmd, "list") != NULL) {

print\_list(sockfd);

handle\_socket(sockfd);

} else if (strstr(cmd, "msg") != NULL) {

token = strtok(cmd, " \t");

char\* msg = strtok(NULL," \t");

print\_mail(sockfd, msg);

handle\_socket(sockfd);

} else if (strstr(cmd, "exit") != NULL) {

disconnect(sockfd);

break;

}

}

close(sockfd);

}

# Лабораторная работа «Программирование средствами MPI».

**Текст задания на лабораторную работу**

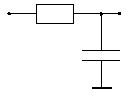
Вариант 6.

Разработать средствами MPI параллельную программу моделирования распространения электрических сигналов в линейной цепочке RC-элементов (см. рис. ниже). Метод формирования математической модели - узловой. Метод численного интегрирования - **явный** Эйлера. Внешнее воздействие - источники тока и напряжения. Количество элементов в сетке, временной интервал моделирования и количество параллельных процессов - параметры программы. Программа должна демонстрировать [ускорение](http://fedoruk.comcor.ru/Dev_bach/mpi_lab.html#par_effect) по сравнению с последовательным вариантом. Предусмотреть визуализацию результатов посредством утилиты gnuplot.

**Описание структуры программы и способов взаимодействия между процессами.**

Для обмена информацией между процессами используются функции MPI. Для отправки широковещательного сообщения используется вызов MPI\_Bcast((void \*)msg, 2, MPI\_INT, 0, MPI\_COMM\_WORLD), для отправки блоков данных используется MPI\_Scatter((void \*)A, (N+1)\*size, MPI\_DOUBLE,(void \*)a, (N+1)\*size, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD), для сбора данных используется функция MPI\_Gather((void \*)a,(N+1)\*size,MPI\_DOUBLE, (void \*)A,(N+1)\*size, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD).

**Описание алгоритма работы**



Узловой метод для формирования математической модели электрической системы использует уравнение второго закона Кирхгофа - сумма токов в узле равна нулю. Это уравнение для любого i-ого узла в цепочке (кроме крайних) имеет вид

IRл - IRп - IC =0, где

IRл=(Ui-1-Ui)/R - электрический ток через "левое" сопротивление;

IRп=(Ui-Ui+1)/R - электрический ток через "правое" сопротивление;

IC=C\*dUi/dt - электрический ток через ёмкость;

Ui - электрический потенциал узла с номером i.

Явный метод Эйлера для численного интегрирования ОДУ подразумевает аппроксимацию производных по времени "разностями вперёд", поэтому дискретизированное уравнение баланса токов в узле принимает следующий вид:

(Uni-1-Uni)/R -  (Uni-Uni+1)/R -  C\*(Un+1i-Uni)/ht, где

ht - величина шага численного интегрирования.

Из него легко выразить единственную неизвестную величину

Un+1i=(Uni-1-2\*Uni +Uni+1)\*ht/(R\*C)-Uni

Что, в свою очередь, дает возможность просто организовать вычислительный процесс в виде "цикл в цикле" (без деталей, связанных с крайними узлами):

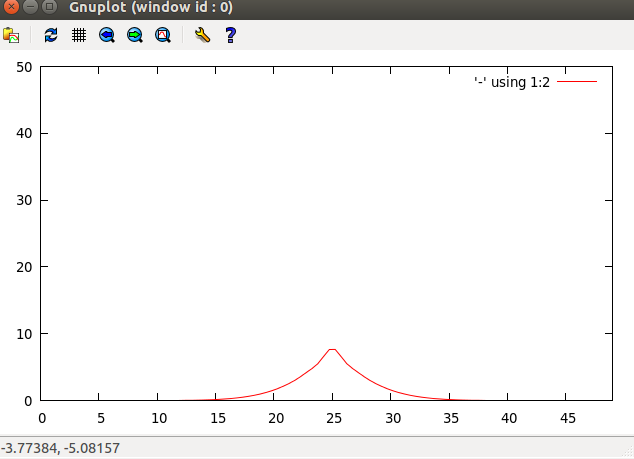
for (n=0; n<N; n++)

  for (i=0; i<M; i++)

    Un+1i = ...

Напомним, что значения U0i известны из начальных условий. Здесь N=Tкон/ht

**Пример работы программы**



**Исходный код программы**

#include <stdio.h>

#include <pthread.h>

#include <stdlib.h>

#include <string.h>

#include <semaphore.h>

#include <sys/time.h>

#include <mpi.h> //mpi

#define \_REENTRANT //mpi

#define C 2.0

#define R 2.0

//#define I 5.0

#define T2C 10.0 //capasity - time relation

#define CUR 2.0 // currency in one of nodes

#define CUR\_NOD 0// number of node with curr!=0

double \*volt\_cur;//mass inside time step

double \*volt\_last;//mass inside last time step

double \*volt\_last\_hard;//mass inside time step

double \*flow;//mass of currencies

pthread\_t\* part\_by\_thread;

int threads\_num;

int nodes\_len;

int nodes\_wid;

int nodes\_num;

int time\_num;

int nodes\_per\_thread;

double ht;

pthread\_barrier\_t bp;

//timing

struct timeval tv1,tv2,dtv;

struct timezone tz;

void time\_start() { gettimeofday(&tv1, &tz); }

long time\_stop()

{ gettimeofday(&tv2, &tz);

dtv.tv\_sec= tv2.tv\_sec -tv1.tv\_sec;

dtv.tv\_usec=tv2.tv\_usec-tv1.tv\_usec;

if(dtv.tv\_usec<0) { dtv.tv\_sec--; dtv.tv\_usec+=1000000; }

return dtv.tv\_sec\*1000+dtv.tv\_usec/1000;

}

void visualiser(void)

{

int i,j;

FILE \*fp;

fp = fopen ("gnulab.txt","a");

fprintf(fp, "plot '-' using 1:2 smooth csplines\n");

//for(j=0;j<2;j++)

for(i=0; i<nodes\_num; i++)

{ fprintf(fp, "%d ", i);

fprintf(fp, "%f\n", volt\_last[i]);

if(!((i+1)%nodes\_wid))

{

fprintf(fp, "\n");

}

}

fprintf(fp, "e\npause 0.01\n\n" );

fclose(fp);

}

int getFirstById(int id)

{

return id\*nodes\_per\_thread;

}

int getLastById(pthread\_t id)

{

if(id==threads\_num-1)

return nodes\_num;

return id\*nodes\_per\_thread + nodes\_per\_thread;

}

char whatIs(int i)

{

if(!i)

{

//leftside

return 1;

}

else if (i==nodes\_num-1)

{

//rightside

return 2;

}

else

//inside

return 3;

}

void compute(int first, int last, int myrank)

{

int t,i,u,timer=0;

MPI\_Status stat;

double \* volt\_cur\_mini = malloc(sizeof(double));

volt\_cur\_mini = (double \*)calloc(last-first, sizeof(double));

for(t=0; t < time\_num; t++)

{

//if(t==0)time\_start();//////////

for(i=first; i<last; i++)

{

char what = whatIs(i);

switch(what)

{

case 1: //leftside

volt\_cur[i] = volt\_last[i] +

(ht/C)\*(flow[i] + (1.0/R)\*

(-2\*volt\_last[i] + volt\_last[i+1]));

break;

case 2: //rightside

volt\_cur[i] = volt\_last[i] +

(ht/C)\*(flow[i] + (1.0/R)\*

(-2\*volt\_last[i] + volt\_last[i-1]));

break;

case 3: //inside

volt\_cur[i] = volt\_last[i] +

(ht/C)\*(flow[i] + (1.0/R)\*

(-2.0\*volt\_last[i] + volt\_last[i+1]+

volt\_last[i-1]));

}

}

//barrier !!!!!

MPI\_Barrier(MPI\_COMM\_WORLD);

//swapping

//for(i = 0; i<(last-first); volt\_cur\_mini[i] = volt\_cur[i+first],i++);

for(i = first; i<(last); volt\_last[i] = volt\_cur[i],i++) {

}

MPI\_Barrier(MPI\_COMM\_WORLD);

if(threads\_num!=1)

{

if(myrank==threads\_num-1)

{//rightside

MPI\_Send((void\*)&(volt\_last[first]),1,

MPI\_DOUBLE,myrank-1,1,MPI\_COMM\_WORLD);

MPI\_Recv((void\*)&(volt\_last[first-1]),1,

MPI\_DOUBLE,myrank-1,MPI\_ANY\_TAG,

MPI\_COMM\_WORLD,&stat);

}

else if(!myrank)

{//leftside

MPI\_Send((void\*)&(volt\_last[last-1]),1,

MPI\_DOUBLE,myrank+1,1,MPI\_COMM\_WORLD);

MPI\_Recv((void\*)&(volt\_last[last]),1,

MPI\_DOUBLE,myrank+1,MPI\_ANY\_TAG,

MPI\_COMM\_WORLD,&stat);

}

else

{

MPI\_Send((void\*)&(volt\_last[last-1]),1,

MPI\_DOUBLE,myrank+1,1,MPI\_COMM\_WORLD);

MPI\_Send((void\*)&(volt\_last[first]),1,

MPI\_DOUBLE,myrank-1,1,MPI\_COMM\_WORLD);

MPI\_Recv((void\*)&(volt\_last[last]),1,

MPI\_DOUBLE,myrank+1,MPI\_ANY\_TAG,

MPI\_COMM\_WORLD,&stat);

//volt\_last[last]=\*volt\_cur\_mini;

MPI\_Recv((void\*)&(volt\_last[first-1]),1,

MPI\_DOUBLE,myrank-1,MPI\_ANY\_TAG,

MPI\_COMM\_WORLD,&stat);

//volt\_last[first-1]=\*volt\_cur\_mini;

}

}

// MPI\_Bcast((void \*)volt\_last, nodes\_num, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD);

//visualise volt\_last

MPI\_Barrier(MPI\_COMM\_WORLD);

//barrier !!!!!

//if(t==0)fprintf(stderr,"Time: %ld, myrank is %d\n", time\_stop(),myrank);

MPI\_Gather((void \*)&(volt\_last[first]), last-first, MPI\_DOUBLE,

(void \*)&(volt\_last[0]), last-first, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD);

if (!myrank) {visualiser();}

}

}

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

{

nodes\_len = 1;

nodes\_wid = atoi(argv[1]);

//threads\_num = atoi(argv[2]);

int m\_time = atoi(argv[2]);

ht = (C/T2C) \* 1e-1;

int myrank, total, i;

MPI\_Init(&argc,&argv);

MPI\_Comm\_size (MPI\_COMM\_WORLD, &total);

MPI\_Comm\_rank (MPI\_COMM\_WORLD, &myrank);

printf ("Total=%d, rank=%d\n", total, myrank);

threads\_num = total;

nodes\_num = nodes\_wid\*nodes\_len;

if(nodes\_num%threads\_num)

nodes\_per\_thread = (int)((double)nodes\_num/(double)threads\_num)+1;

else

nodes\_per\_thread = (int)((double)nodes\_num/(double)threads\_num);

time\_num = (int)(m\_time/ht);

volt\_cur = (double \*)calloc(nodes\_num, sizeof(double));

for(i = 0; i<nodes\_num; volt\_cur[i++] = 0.0);

volt\_last = (double \*)calloc(nodes\_num, sizeof(double));

for(i = 0; i<nodes\_num; volt\_last[i++] = 0.0);

//volt\_last[0]=100;

//volt\_last[nodes\_num-1]=50;

//allocate currency memory

flow = (double \*)calloc(nodes\_wid, sizeof(double));

for(i = 0; i < nodes\_wid; flow[i++] = 0.0);

flow[nodes\_wid / 2]=CUR;

if(!myrank)

{

//super hard array

volt\_last\_hard = (double \*)calloc(nodes\_num, sizeof(double));

for(i = 0; i<nodes\_num; volt\_last\_hard[i++] = 0.0);

//time steps number

if(threads\_num>nodes\_num)

{

perror("threads num can't be grater then nodes num");

exit(2);

}

//allocate voltage memory

//prepare file for gnuplt

FILE \*fp;

fp = fopen ("gnulab.txt","w");

fprintf(fp, "set cbrange [0:20]\n");

fprintf(fp, "set cblabel \"Score\"\n");

fprintf(fp, "set xrange [0:%d]\n", nodes\_wid-1);

fprintf(fp, "set yrange [0:%d]\n", 50);

fprintf(fp, "set view map\n");

//fprintf(fp, "set size ratio 1\n"); ?????????????????????????

fclose(fp);

visualiser();

//pthread\_barrier\_init(&bp, NULL, threads\_num);

//create threads and pass they to function

time\_start();

}

//MPI\_Barrier(MPI\_COMM\_WORLD);

compute(getFirstById(myrank),getLastById(myrank),myrank);

if(!myrank)

{

fprintf(stderr,"Time: %ld\n", time\_stop());

system("gnuplot -persist \"gnulab.txt\""); //Ñ‡Ñ‚Ð¾Ð±Ñ‹ Ð¿Ñ€Ð¾Ð²ÐµÑ€Ð¸Ñ‚ÑŒ Ñ€Ð°Ð±Ð¾Ñ‚Ñƒ

}

// pthread\_barrier\_destroy(&bp);

MPI\_Finalize();

exit(0);

}