



Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет имени
Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Робототехника и комплексная автоматизация»

КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине «Разработка программных систем»

Студент:	Новокшанов Евгений Андреевич
Группа:	РК6-66Б
Тип задания:	лабораторная работа
Тема:	Сетевое программирование
Вариант:	6

Студент

подпись, дата

Новокшанов Е.А.
Фамилия, И.О.

Преподаватель

подпись, дата

Козов А.В.
Фамилия, И.О.

Москва, 2023

Содержание

Задание	3
Описание структуры программы и реализованных способов взаимодействия процессов	4
Описание основных используемых структур данных	6
Блок-схема	7
Примеры работы программы	10
Текст программы	12

Задание

Разработать "чрезвычайно упрощенный клиент" протокола http. Программа- "браузер" принимает в качестве аргумента командной строки URL-адрес web-сервера, обращается по прикладному протоколу http к серверу, получает содержимое страницы и выводит его в текстовом виде на стандартный вывод. Кроме того выводятся в виде списка все ссылки, имеющиеся на странице. Пользователь-человек имеет возможность указать номер ссылки и заставить "браузер" обратиться по ней. Для поиска в тексте страницы ссылок рекомендуется использовать пару функций `regstr` и `regex`.

Описание структуры программы и реализованных способов взаимодействия процессов

Программа создаёт и связывает сокет для соединения по стеку протоколов TCP/IP через клиентский порт 1235. URL-адрес, полученный в качестве аргумента разделяется на две части: доменное имя хоста и URL- путь. По доменному имени определяется адрес сервера и посылается запрос на соединение. В случае удачного соединения программа посылает на сервер http-запрос и ожидает ответа. Запрос формируется в символьном массиве `sendbuf` по следующему шаблону:

- GET pwd HTTP/1.1;
- host: hostname;
- pwd – URL-путь;
- hostname - доменное имя.

Получение страницы осуществляется в цикле `while` с помощью функции **`recv()`**. Необходимость цикла объясняется тем, что TCP соединение не гарантирует, что вся запрошенная страница придёт на сокет в одно время, а функция **`recv()`** считывает из сокета все символы, пришедшие на момент её вызова. Полученная страница записывается в буфер **`recvbuf`**, размер которого задаётся через **`#define`**. Переполнение буфера невозможно, так как третий аргумент функции **`recv()`** задаёт максимальное число считываемых символов. Если страница превышает заданный размер, то отобразится лишь её часть. Содержимое полученной страницы выводится на экран вместе с заголовком. При помощи пары функций **`regcmp`** и **`regex`** программа ищет ссылки в тексте страницы. Ссылки хранятся в двумерном массиве.

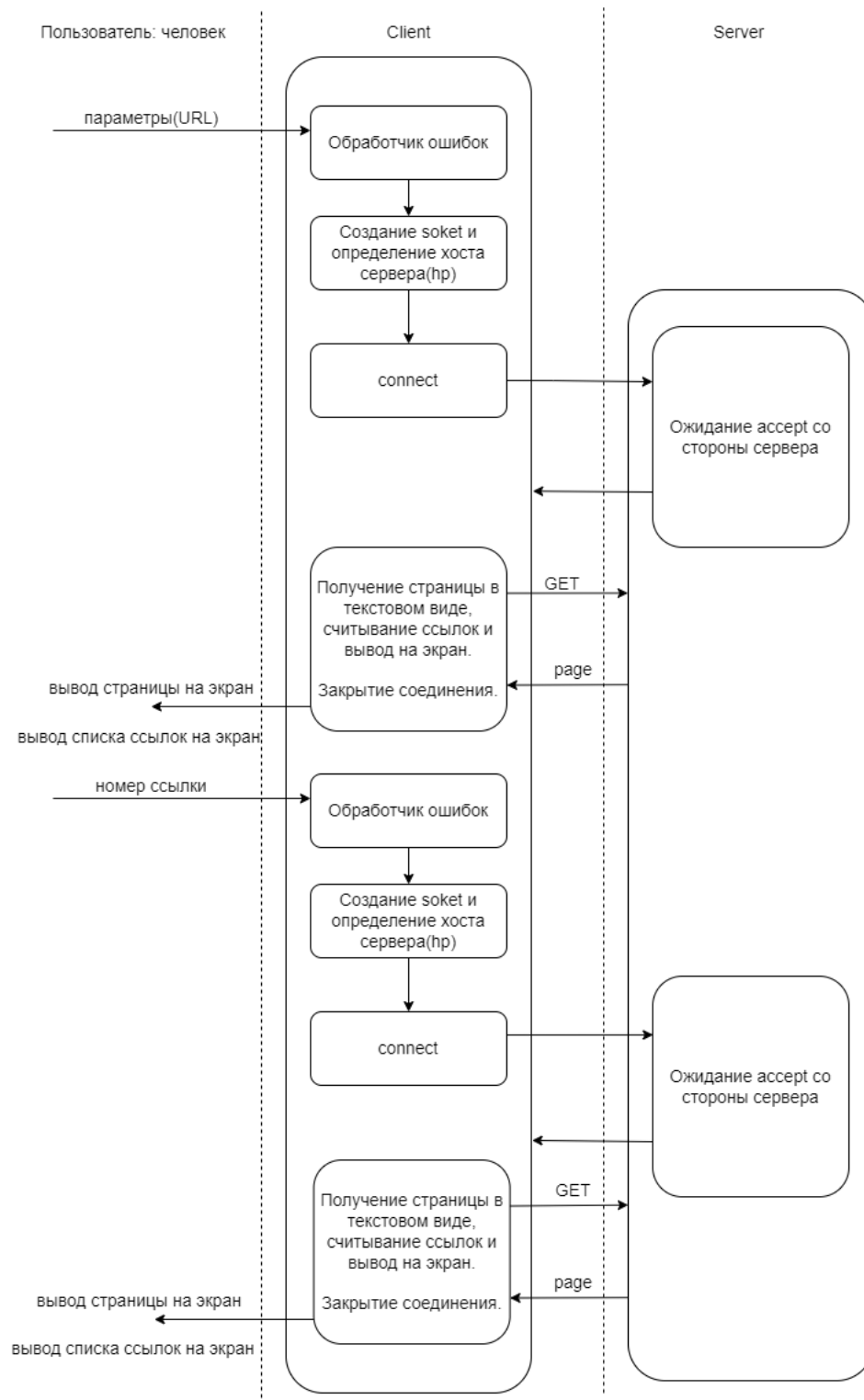


Рис. 1. UML-диаграмма взаимодействия клиента и сервера

Описание основных используемых структур данных

`char links[MAX_LINKS][LINK_LENGTH]`, где `MAX_LINKS = 50` и `LINK_LENGTH = 256` – константы, заданные через `#define`.

Блок-схема

На следующих рисунках представлена блок-схема программы.

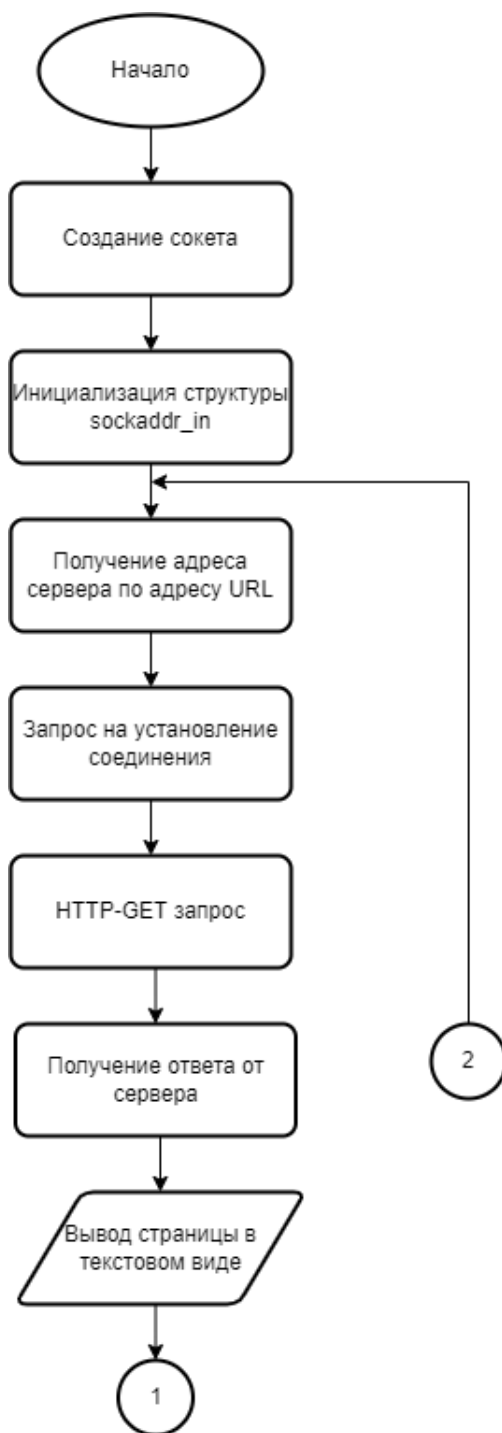


Рис. 2. Блок-схема работы программы 1

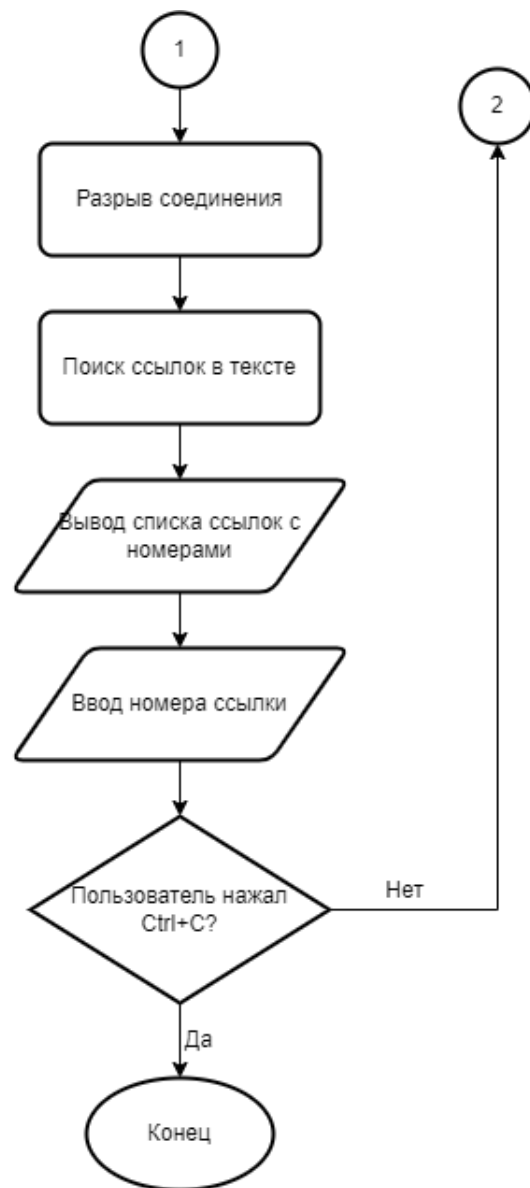


Рис. 3. Блок-схема работы программы 2

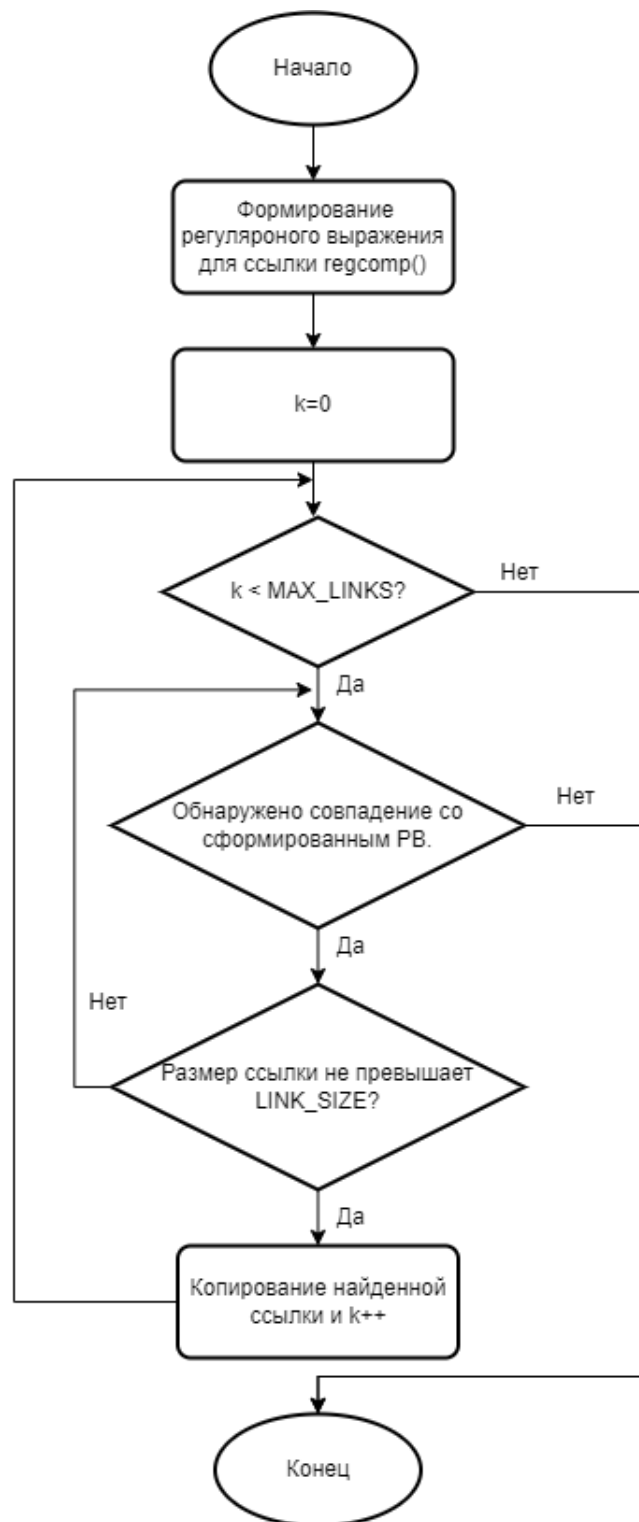


Рис. 4. Блок-схема функции поиска ссылок в тексте

Примеры работы программы

На рисунках 5, 6, 7 приведен пример работы программы: вывод страницы в текстовом виде, вывод списка ссылок и переход по ссылке.

```
HTTP/1.1 200 OK
Date: Wed, 29 May 2019 12:19:19 GMT
Server: Apache/1.3.37 (Unix) PHP/4.4.7 rus/PL30.22
Last-Modified: Tue, 12 Apr 2016 09:09:43 GMT
ETag: "babf-b52-570cbb57"
Accept-Ranges: bytes
Content-Length: 2898
Connection: close
Content-Type: text/html

<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=koI8-r">
  <meta name="GENERATOR" content="Mozilla/4.7 [en] (X11; I; SunOS 5.8 sun4u) [Netscape]">
</head>
<body>

<CENTER><a href="exams+labs.html">Расписание экзаменов, консультаций,
лабораторных работ.
</a></CENTER>
<BR>

<HR>
<a href="Dev_bach/index.html">Учебно-методические материалы к курсу
"Разработка программных систем" для бакалавров</a>
```

Рис. 5. Пример работы программы №1: создание ПМ

```
links on the page:
1 fedoruk.comcor.ru/exams+labs.html
2 fedoruk.comcor.ru/Dev_bach/index.html
3 fedoruk.comcor.ru/AI_mag/index.html
4 fedoruk.comcor.ru/diploma_faq.html
5 fedoruk.comcor.ru/Ovchinnikov2012.doc
6 fedoruk.comcor.ru/programing.html
7 fedoruk.comcor.ru/non_canon.html
8 fedoruk.comcor.ru/unix-debug.html
9 fedoruk.comcor.ru/gdb_briefly.html
10 fedoruk.comcor.ru/Ling/simplecompiler.html
11 fedoruk.comcor.ru/lingware.html
12 fedoruk.comcor.ru/SQL/sql_tutor.html
13 fedoruk.comcor.ru/VHDLbook/index.html
14 fedoruk.comcor.ru/VHDLlab/lab_vhdl.html
15 fedoruk.comcor.ru/knirs.html
16 fedoruk.comcor.ru/practice_blank.html
17 fedoruk.comcor.ru/anketa_Ovchinnikov.xls
18 fedoruk.comcor.ru/archives/electric-7.00.tar.gz
19 fedoruk.comcor.ru/archives/electric-8.01.jar
20 fedoruk.comcor.ru/archives/alliance-5.0-20060218.tar.gz
21 fedoruk.comcor.ru/archives/pa9.zip
22 fedoruk.comcor.ru/archives/PA9j6.jar
23 fedoruk.comcor.ru/archives/jdk1.1.zip
24 fedoruk.comcor.ru/archives/pa8.new.tgz
25 fedoruk.comcor.ru/archives/vs2015.2.com_rus.iso
26 fedoruk.comcor.ru/archives/mmapr.zip
27 fedoruk.comcor.ru/cad_tick.pdf
=====
choose link by number (1-27)
1
```

Рис. 6. Пример работы программы: вывод страницы в текстовом виде

```

27 fedoruk.comcor.ru/cad_tick.pdf
=====
choose link by number (1-27)
1
=====
fedoruk.comcor.ru/exams+labs.html
HTTP/1.1 200 OK
Date: Wed, 29 May 2019 12:20:42 GMT
Server: Apache/1.3.37 (Unix) PHP/4.4.7 rus/PL30.22
Last-Modified: Tue, 28 May 2019 12:57:21 GMT
ETag: "bbcb-11c0-5ced3031"
Accept-Ranges: bytes
Content-Length: 4544
Connection: close
Content-Type: text/html

<HTML>
<HEAD>
    <META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html; charset=koI8-r">
    <TITLE>Расписание экзаменов, консультаций, лабораторных работ</TITLE>
</HEAD>
<BODY>
<CENTER>Расписание экзаменов, консультаций, лабораторных работ</CENTER>
<BR><BR>
<TABLE BORDER=1 ALIGN=CENTER>

<TR><TH>Дата</TH><TH>Время</TH><TH>Вид</TH><TH>Группа</TH><TH>Место</TH></TR>
<TR><TD>07.11 ср.</TD><TD>09:00</TD><TD>лаб. раб.</TD><TD>ПК6-11,12м</TD><TD>ауд. 404э</TD></TR>
<TR><TD>14.11 ср.</TD><TD>09:00</TD><TD>лаб. раб.</TD><TD>ПК6-11,12м</TD><TD>ауд. 404э</TD></TR>
<TR><TD>21.11 ср.</TD><TD>09:00</TD><TD>лаб. раб.</TD><TD>ПК6-11,12м</TD><TD>ауд. 404э</TD></TR>
<TR><TD>28.11 ср.</TD><TD>09:00</TD><TD>лаб. раб.</TD><TD>ПК6-11,12м</TD><TD>ауд. 404э</TD></TR>
<TR><TD>05.12 ср.</TD><TD>09:00</TD><TD>лаб. раб.</TD><TD>ПК6-11,12м</TD><TD>ауд. 404э</TD></TR>

```

Рис. 7. Пример работы программы: вывод списка ссылок

Текст программы

Ниже в листинге 1 представлен текст программы.

Листинг 1. Листинг программы

```
1
2 #include <sys/types.h>
3 #include <sys/socket.h>
4 #include <netinet/in.h>
5 #include <netdb.h>
6 #include <memory.h>
7 #include <stdlib.h>
8 #include <stdio.h>
9 #include <unistd.h>
10 #include <string.h>
11 #include <regex.h>
12
13 //задание ограничений на количество ссылок и размер ссылки
14 #define SRV_PORT 80
15 #define CLNT_PORT 1235
16 #define BUF_SIZE 8192
17 #define MAX_LINKS 50
18 #define LINK_LENGTH 256
19 #define LINK_HALF 128
20
21 int done = 1;
22 char new_link[LINK_LENGTH];
23 int create_tcp_socket()//функция создания сокета
24 {
25     int sock;
26     if ((sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0) {//проверка
27         perror("Can't create TCP socket");//вывод сообщения об ошибке
28         exit(1);
29     }
30     return sock;
31 }
32 void next_step(int argc, char* argv[]) {
33     if (new_link[0] != ' ') {
34         strcpy(argv[1], new_link);
35         printf("%s\n", argv[1]);
36     }
37     int s;
38     int from_len;
39     char recvbuf[BUF_SIZE];
40     char sendbuf[BUF_SIZE];
```

```

41 struct hostent* hp;
42 struct sockaddr_in clnt_sin, srv_sin;
43 char* pwd_beg;
44 char pwd[LINK_HALF];
45 char hostname[LINK_HALF];
46 int i = 0;
47 int k = 0;
48 int length = 0;
49 regex_t regex;
50 regmatch_t link;
51 char* begin;
52 char links[MAX_LINKS][LINK_LENGTH];
53 if (argc != 2) {
54     fprintf(stderr, "usage: URL address\n"); //вывод сообщения об ошибке
55     exit(1);
56 }
57 s = create_tcp_socket();
58 memset((char*)&clnt_sin, '\0', sizeof(clnt_sin)); //очистить структуру
59 clnt_sin.sin_family = AF_INET; //инициализация структуры
60 clnt_sin.sin_addr.s_addr = INADDR_ANY;
61 clnt_sin.sin_port = htons(CLNT_PORT);
62
63 if ((pwd_beg = strchr(strchr(argv[1], '.'), '/')) != NULL) { //разделение URL адреса
64     strcpy(pwd, pwd_beg);
65     *(pwd_beg) = '\0';
66     strcpy(hostname, argv[1]);
67 }
68 memset((char*)&srv_sin, '\0', sizeof(srv_sin));
69 if ((hp = gethostbyname(hostname)) == NULL) {
70     fprintf(stderr, "invalid URL\n");
71     close(s);
72     exit(4);
73 } //адрес хоста по имени
74 srv_sin.sin_family = AF_INET;
75 memcpy((char*)&srv_sin.sin_addr, hp->h_addr, hp->h_length);
76 srv_sin.sin_port = htons(SRV_PORT);
77 if ((connect(s, (struct sockaddr*)&srv_sin, sizeof(srv_sin))) < 0) {
78     fprintf(stderr, "connection failed\n");
79     close(s);
80     exit(5);
81 } //запрос на соединение
82 sprintf(sendbuf, "GET %s HTTP/1.1\nhost:%s\nconnection:close\n\n", pwd,
83     hostname); //отправление http-запроса
84 write(s, sendbuf, strlen(sendbuf));
85 int offset = 0; //получение страницы в текстовом виде
86 while ((from_len = recv(s, recvbuf + offset, BUF_SIZE - offset, 0)) != 0) {

```

```

87     offset += from_len;
88 }
89 write(1, recvbuf, offset);
90 write(1, "\n", 1);
91 regcomp(&regex, "href=", 0); //регулярное выражение
92 begin = recvbuf;
93 printf("=====\n");
94 write(1, "\nlinks on the page:\n\n", 20);
95 int n = 0;
96 n = strlen(pwd);
97 while (pwd[n] != '/') {
98     n--;
99 }
100 memset(pwd + n + 1, '\0', sizeof(pwd) - n - 1);
101
102 char num[4];
103 while ((regexexec(&regex, begin, 1, &link, 0) != REG_NOMATCH) //формирование
104     регулярного выражения
105     && (k < MAX_LINKS)) {
106     length = link.rm_eo - link.rm_so;
107     if (length > LINK_LENGTH)
108         continue;
109     sprintf(num, "%d", k + 1);
110     write(1, num, strlen(num));
111
112     write(1, hostname, sizeof(hostname));
113     write(1, pwd, sizeof(pwd));
114     write(1, begin + link.rm_so + 6, length - 7);
115     write(1, "\n", 1);
116     memset(links[k], '\0', sizeof(links[k]));
117     strncpy(links[k], begin + link.rm_so + 6, length - 7);
118     k++; // увеличение счетчика ссылок
119     begin += link.rm_eo; //переход к следующей ссылке
120 }
121 if (k == 0) {
122     printf("no links on this page\n"); //вывод сообщения о том, что ссылок не найдено
123     done = 0;
124     //exit(6);
125 }
126 printf("=====\n");
127 fprintf(stdout, "\nchoose link by number (1-%d)\n", k);
128 scanf("%d", &k); //считывание номера ссылки для перехода
129 printf("=====\n");
130 k--;
131 char sample[8];
132 strncpy(sample, links[k], 7);

```

```
132 sample[7] = '\0';
133 char abs_prefix[] = "http://\0"; //формирование следующего URL
134 if (strcmp(sample, abs_prefix) == 0) {
135     strcpy(new_link, links[k] + 7);
136 }
137 else {
138     strcpy(new_link, hostname);
139     strcat(new_link, pwd);
140     strcat(new_link, links[k]);
141     //printf(
142 shutdown(s, 2); //закрытие соединения и сокета close(s); int main(int argc, char* argv[])
    new_link[0] = ' '; while (1) if (done) next_step(argc, argv); exit(0);
```
