

SimpleWebServer 程序设计报告

一. 程序文件结构说明

程序由四个文件组成: main.c net.c net.h makefile

main.c: main.c 文件中对输入的命令进行处理, 并设置相应的 flag

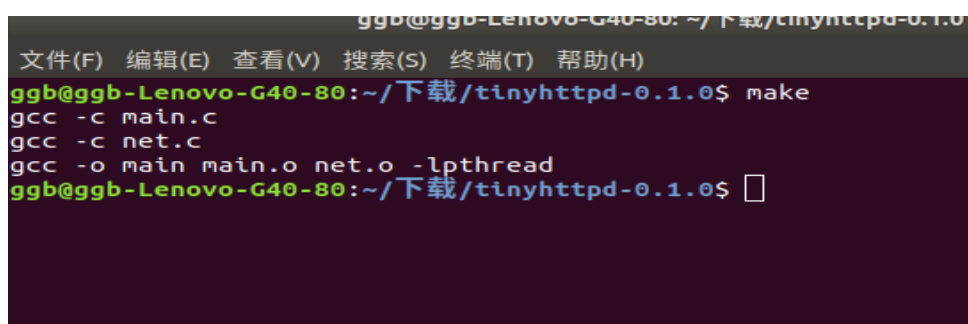
net.c: net.c 中根据输入的命令, 进行设置端口, 监听 IP 等操作

net.h: 包含运行所需要的库文件

makefile: 将文件联系, 编译生成可执行文件

二. 程序测试与使用

打开 Linux 终端, 输入: make, 程序执行 makefile 文件, 生成对应的可执行文件 main



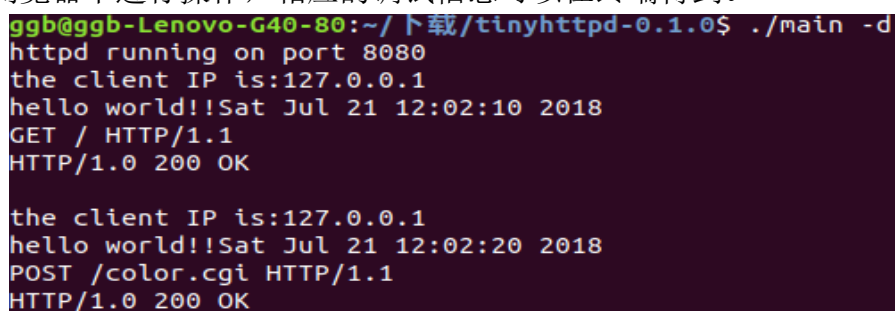
```
ggb@ggb-Lenovo-G40-80: ~/下载/tinyhttpd-0.1.0
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
ggb@ggb-Lenovo-G40-80:~/下载/tinyhttpd-0.1.0$ make
gcc -c main.c
gcc -c net.c
gcc -o main main.o net.o -lpthread
ggb@ggb-Lenovo-G40-80:~/下载/tinyhttpd-0.1.0$
```

注: 默认的端口为 8080, 若不输入监听的 IP, 则监听所有。

2.1. 测试-d 命令

在终端输入: ./main -d

程序进入 debug 模式, 在浏览器中输入: localhost:8080, 程序只建立一个连接, 在浏览器中进行操作, 相应的调试信息可以在终端得到。



```
ggb@ggb-Lenovo-G40-80:~/下载/tinyhttpd-0.1.0$ ./main -d
httpd running on port 8080
the client IP is:127.0.0.1
hello world!!Sat Jul 21 12:02:10 2018
GET / HTTP/1.1
HTTP/1.0 200 OK

the client IP is:127.0.0.1
hello world!!Sat Jul 21 12:02:20 2018
POST /color.cgi HTTP/1.1
HTTP/1.0 200 OK
```

2.2.测试-h 命令

在终端输入: `./main -h`

程序打印出服务器的使用说明，并退出。

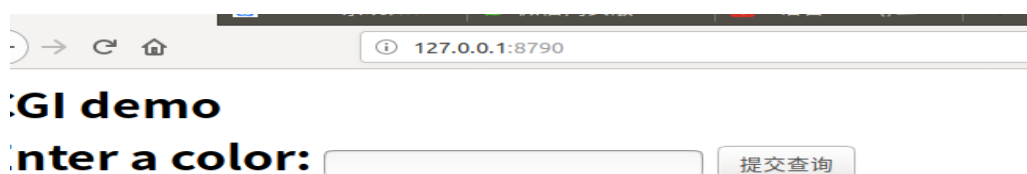
```
ggb@ggb-Lenovo-G40-80:~/下载/tinyhttpd-0.1.0$ ./main -h
-c dir Allow execution of CGIs from the given directory. See CGIs for details
-d Enter debugging mode. That is, do not daemonize, only accept one connection at a time and enable logging to stdout.
-h Print a short usage summary and exit.
-i address Bind to the given IPv4 or IPv6 address. If not provided, sws will listen on all IPv4 and IPv6 addresses on this host.
-l file Log all requests to the given file. See LOGGING for details.
-p port Listen on the given port. If not provided, sws will listen on port 8080.
ggb@ggb-Lenovo-G40-80:~/下载/tinyhttpd-0.1.0$
```

2.3.测试 -i -p 命令

在终端输入: `./main -p8790 -i127.0.0.1`

程序绑定端口 8790，监听 IP 127.0.0.1，在浏览器中输入: `localhost:8080`，无响应，没有连接。输入 `127.0.0.1:8080`，进入操作界面，输入颜色 `blue`，进入一个蓝色界面。

```
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
ggb@ggb-Lenovo-G40-80:~/下载/tinyhttpd-0.1.0$ make
gcc -c main.c
gcc -c net.c
gcc -o main main.o net.o -lpthread
ggb@ggb-Lenovo-G40-80:~/下载/tinyhttpd-0.1.0$ ./main -p8790 -i127.0.0.1
127.0.0.1httpd running on port 8790
```





2.4.测试 -l 命令

在终端输入: `./main -llog1.txt -p8890`

服务器绑定端口 8890，在浏览器中输入: `localhost:8890`，进去 index 界面，输入颜色 blue，进入一个蓝色界面。

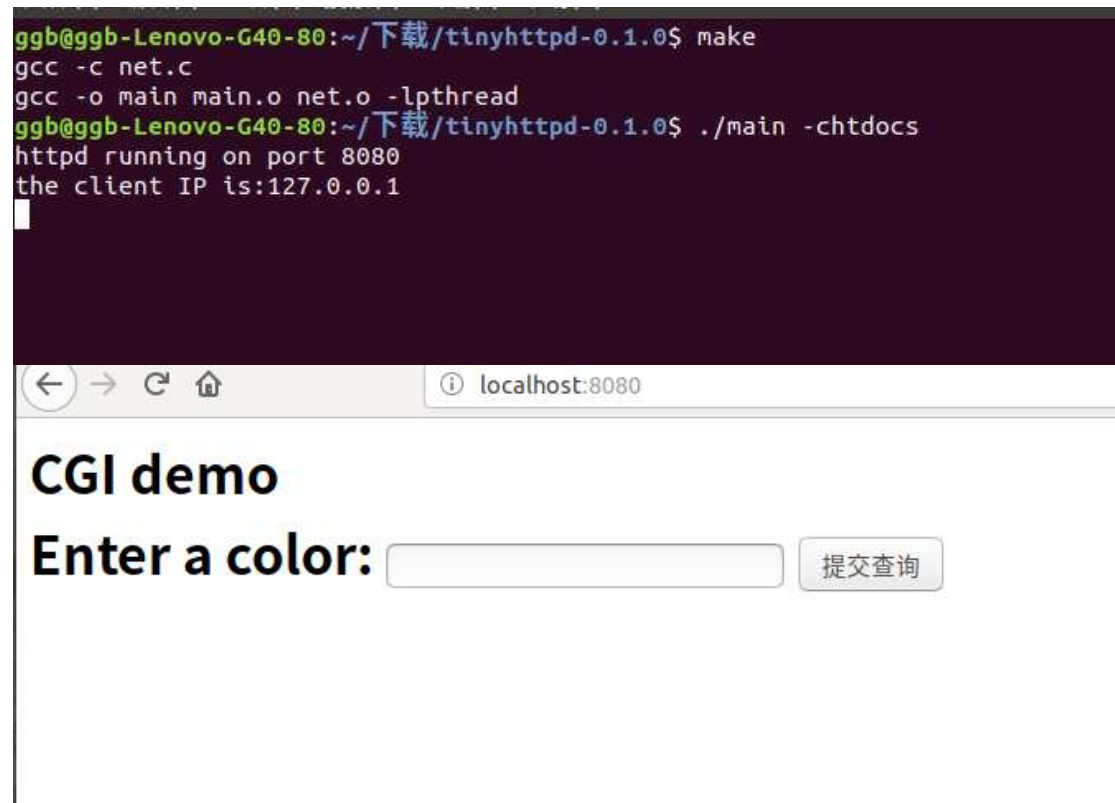
```
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
ggb@ggb-Lenovo-G40-80:~/下载/tinyhttpd-0.1.0$ ./main -llog1.txt
bind: Address already in use
ggb@ggb-Lenovo-G40-80:~/下载/tinyhttpd-0.1.0$ ./main -llog1.txt -p8890
httpd running on port 8890
the client IP is:127.0.0.1
信息写入日志入log
the client IP is:127.0.0.1
信息写入日志入log
the client IP is:127.0.0.1
信息写入日志入log
```

```
127.0.0.1Sat Jul 21 12:04:32 2018
GET / HTTP/1.1
HTTP/1.0 200 OK
127.0.0.1Sat Jul 21 12:04:32 2018
GET /favicon.ico HTTP/1.1
HTTP/1.0 404 NOT FOUND
127.0.0.1Sat Jul 21 12:04:39 2018
POST /color.cgi HTTP/1.1
HTTP/1.0 200 OK
```

2.5.测试 -c 命令

在终端输入`./main -chtdocs`

服务器绑定到默认端口 8080，在浏览器进入 localhost:8080,程序会默认执行路径下的 index.html 文件。



三、具体代码实现说明

3.1.-c 指令实现

本 web 服务器的实现是基于进程创建的。为实现 -c 命令，需添加 dir 参数给 accept_request 函数。因此创建了 arg_struct 结构体用来存储套接口信息以及 dir 信息，并通过进程创建函数 pthread_create 函数将结构体信息送入 accept_request 函数中。此外，只需对 url 内的 /cgi-bin 前缀进行判断即可。

```
struct arg_struct {
    char dir[255];
    int socket_id;
};
```

```
if (pthread_create(&newthread, NULL, accept_request, (void *)&args) != 0)
    perror("pthread_create");
```

```

int k1=0;

if(url[0]!='\0')
{
    int k2=0;
    char tool[8];
    for(; k2<=7; k2++)
    {
        tool[k2]=url[k2];
    }
    if(strcmp(tool,"/cgi-bin")==0)
    {
        while(url[k1+k2]!='\0')
        {
            url[k1]=url[k1+k2];
            k1++;
        }
        url[k1]='\0';
        //printf("5555..url of cgi_dir: %s\n",url);
        //fflush(stdout);
    }
}

/*****分隔符*****/

if(args->dir[0]!='\0')
{
    sprintf(path, "htdocs%s", url);
    //printf("3333.:%s\n",path);
    //fflush(stdout);
}
else{
    sprintf(path, "%s%s", args->dir,url);
    //printf("4444.:%s\n",path);
    //fflush(stdout);
}

```

3.2.-h 指令实现

在程序中接收-h 命令符，执行 print_usage 函数，然后直接调用 exit(),退出程序

```

void print_usage()
{
    printf("-c dir Allow execution of CGIs from the given directory. See    CGIs for details.\n");
    printf("-d Enter debugging mode. That is, do not daemonize, only accept one connection at a time\n");
    printf("-h Print a short usage summary and exit.\n");
    printf("-i address Bind to the given IPv4 or IPv6 address. If not provided, sws will listen on all\n");
    printf("-l file Log all requests to the given file. See LOGGING for details.\n");
    printf("-p port Listen on the given port. If not provided, sws will listen on port 8080.\n");
}

```

3.3.-i 指令实现

在命令行传入 IP，设置相应的 flag，将 address 传入 startup 函数，监听相应 IP 地址。对 ipv4 和 ipv6 做不同的处理，ipv4 以 “.” 分割开，ipv6 以 “:” 分隔开。

```

int k=0;
printf("%s",address);
while(address[k]!='\0')
{
    k++;
    if(address[k]=='.')
        {tag=0;break;}
    else if(address[k]==':')
        {tag=1;break;}
}

```

```

.....
if(tag==1)
{
    httpd = socket(PF_INET6, SOCK_STREAM, 0);
    if (httpd == -1)
        error_die("socket");
    name.sin_family = AF_INET6;
    inet_pton(AF_INET6,address, (void *)& name.sin_addr.s_addr);
}
//ipv4
else if(tag==0)
{
    httpd = socket(PF_INET, SOCK_STREAM, 0);
    if (httpd == -1)
        error_die("socket");
    name.sin_family = AF_INET;
    name.sin_addr.s_addr = inet_addr(address);
}
else
{
    httpd = socket(PF_INET, SOCK_STREAM, 0);
    if (httpd == -1)
        error_die("socket");
    name.sin_family = AF_INET;
    name.sin_addr.s_addr = htonl(INADDR_ANY);
}

```

3.4.-p 实现说明

如果没有传入绑定的端口，则默认是 8080；传入端口，则绑定该端口。

```

//port修改
server_sock = startup(&port,tag,address);
printf("httpd running on port %d\n", port);

```

3.5.-l 实现说明

在函数中设置全局变量，文件流 FILE* logfp，获取本地时间写入 log 文件中，并将 request 以及 response 信息写入 log 文件。

```

extern FILE* logfp;
extern int islog;
extern int isdebug;

```

```

/*****/

time_t    now;           //实例化time_t结构
struct    tm    *timenow;    //实例化tm结构指针
time(&now);
timenow    =    localtime(&now);
strcpy(time1,asctime(timenow));
if(islog)
{
    fwrite(time1,strlen(time1),1,logfp);
    fflush(logfp);
    printf("信息写入日志入log\n");
}
/localtime函数把从time取得的时间now换算成你电脑中的时间(就是你设置的地区)
}
if(isdebug)
{
    printf("%s",time1);
}

```

3.6.-d 实现说明

根据传入的命令，如果-d 和-l 同时出现，则进行判断，只进行 debug，不写 log 文件。程序把写入 log 的信息直接写到控制台。