C 程 序 设 计
C Programming

4

WEBBENCH阅读

阅读课程

厦門大學
XIAMEN UNIVERSITY

信息学院
（国家示范性软件学院）
School of Informatics

黄 炜
博士，副教授
Dr. Wei Huang

```c
/* $Id: socket.c 1.1 1995/01/01 07:11:14 cthuang Exp $
 *
 * This module has been modified by Radim Kolar for OS/2 emx
 */

/***********************************************************************
  module:       socket.c
  program:      popclient
  SCCS ID:      @(#)socket.c    1.5  4/1/94
  programmer:   Virginia Tech Computing Center
  compiler:     DEC RISC C compiler (Ultrix 4.1)
  environment:  DEC Ultrix 4.3
  description:  UNIX sockets code.
 ***********************************************************************/

#include <sys/types.h>
#include <sys/socket.h>
#include <fcntl.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/time.h>
```

```c
#include <string.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>

int Socket(const char *host, int clientPort)
{
    int sock;
    unsigned long inaddr;
    struct sockaddr_in ad;
    struct hostent *hp;

    memset(&ad, 0, sizeof(ad));
    ad.sin_family = AF_INET;

    inaddr = inet_addr(host);
    if (inaddr != INADDR_NONE)
        memcpy(&ad.sin_addr, &inaddr, sizeof(inaddr));
    else
    {
        hp = gethostbyname(host);
        if (hp == NULL)
            return -1;
        memcpy(&ad.sin_addr, hp->h_addr, hp->h_length);
    }
    ad.sin_port = htons(clientPort);

    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock < 0)
        return sock;
    if (connect(sock, (struct sockaddr *)&ad, sizeof(ad)) < 0)
        return -1;
    return sock;
}
```

```c
/*
* (C) Radim Kolar 1997-2004
* This is free software, see GNU Public License version 2 for
* details.
*
* Simple forking WWW Server benchmark:
*
* Usage:
*   webbench --help
*
* Return codes:
*    0 - sucess
*    1 - benchmark failed (server is not on-line)
*    2 - bad param
*    3 - internal error, fork failed
*
*/

#include "socket.c"
#include <unistd.h>
#include <sys/param.h>
#include <rpc/types.h>
```

```c
#include <getopt.h>
#include <strings.h>
#include <time.h>
#include <signal.h>

/* values */
volatile int timerexpired=0;
int speed=0;
int failed=0;
int bytes=0;

/* globals */
int http10=1; /* 0 - http/0.9, 1 - http/1.0, 2 - http/1.1 */
/* Allow: GET, HEAD, OPTIONS, TRACE */
#define METHOD_GET 0
#define METHOD_HEAD 1
#define METHOD_OPTIONS 2
#define METHOD_TRACE 3
#define PROGRAM_VERSION "1.5"
int method=METHOD_GET;
int clients=1;
int force=0;
int force_reload=0;
int proxyport=80;
char *proxyhost=NULL;
int benchtime=30;

/* internal */
int mypipe[2];
char host[MAXHOSTNAMELEN];
#define REQUEST_SIZE 2048
char request[REQUEST_SIZE];

static const struct option long_options[]=
{
    {"force",no_argument,&force,1},
    {"reload",no_argument,&force_reload,1},
    {"time",required_argument,NULL,'t'},
    {"help",no_argument,NULL,'?'},
    {"http09",no_argument,NULL,'9'},
    {"http10",no_argument,NULL,'1'},
    {"http11",no_argument,NULL,'2'},
    {"get",no_argument,&method,METHOD_GET},
    {"head",no_argument,&method,METHOD_HEAD},
    {"options",no_argument,&method,METHOD_OPTIONS},
    {"trace",no_argument,&method,METHOD_TRACE},
    {"version",no_argument,NULL,'V'},
    {"proxy",required_argument,NULL,'p'},
    {"clients",required_argument,NULL,'c'},
    {NULL,0,NULL,0}
};

/* prototypes */
static void benchcore(const char* host,const int port, const char *request);
static int bench(void);
static void build_request(const char *url);

static void alarm_handler(int signal)
{
    timerexpired=1;
}

static void usage(void)
{
    fprintf(stderr,
            "webbench [option]... URL\n"
            "   -f|--force               Don't wait for reply from server.\n"
            "   -r|--reload              Send reload request - Pragma: no-
cache.\n"
            "   -t|--time <sec>          Run benchmark for <sec> seconds. 
Default 30.\n"
            "   -p|--proxy <server:port> Use proxy server for request.\n"
            "   -c|--clients <n>         Run <n> HTTP clients at once. 
Default one.\n"
            "   -9|--http09              Use HTTP/0.9 style requests.\n"
            "   -1|--http10              Use HTTP/1.0 protocol.\n"
            "   -2|--http11              Use HTTP/1.1 protocol.\n"
            "   --get                    Use GET request method.\n"
            "   --head                   Use HEAD request method.\n"
            "   --options                Use OPTIONS request method.\n"
            "   --trace                  Use TRACE request method.\n"
            "   -?|-h|--help             This information.\n"
            "   -V|--version             Display program version.\n"
           );
}
```

```
int main(int argc, char *argv[])
{
    int opt=0;
    int options_index=0;
    char *tmp=NULL;

    if(argc==1)
    {
        usage();
        return 2;
    }

    while((opt=getopt_long(argc,argv,"912Vfrt:p:c:?h",long_options,
&options_index))!=EOF )
    {
        switch(opt)
        {
            case  0 : break;
            case 'f': force=1;break;
            case 'r': force_reload=1;break;
            case '9': http10=0;break;
            case '1': http10=1;break;
            case '2': http10=2;break;
            case 'V': printf(PROGRAM_VERSION"\n");exit(0);
            case 't': benchtime=atoi(optarg);break;
            case 'p':
            /* proxy server parsing server:port */
            tmp=strrchr(optarg,':');
            proxyhost=optarg;
            if(tmp==NULL)
            {
                break;
            }
            if(tmp==optarg)
            {
                fprintf(stderr,"Error in option --proxy %s: Missing 
hostname.\n",optarg);
                return 2;
            }
            if(tmp==optarg+strlen(optarg)-1)
            {
                fprintf(stderr,"Error in option --proxy %s Port number is 
missing.\n",optarg);
                return 2;
            }
            *tmp='\0';
            proxyport=atoi(tmp+1);break;
            case ':':
            case 'h':
            case '?': usage();return 2;break;
            case 'c': clients=atoi(optarg);break;
        }
    }

    if(optind==argc) {
        fprintf(stderr,"webbench: Missing URL!\n");
        usage();
        return 2;
    }

    if(clients==0) clients=1;
    if(benchtime==0) benchtime=30;

    /* Copyright */
    fprintf(stderr,"Webbench - Simple Web Benchmark "PROGRAM_VERSION"\n"
          "Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.\n"
            );

    build_request(argv[optind]);

    // print request info ,do it in function build_request
    /*printf("Benchmarking: ");

    switch(method)
    {
        case METHOD_GET:
        default:
        printf("GET");break;
        case METHOD_OPTIONS:
        printf("OPTIONS");break;
        case METHOD_HEAD:
        printf("HEAD");break;
        case METHOD_TRACE:
        printf("TRACE");break;
    }

    printf(" %s",argv[optind]);
```

```
switch(http10)
{
    case 0: printf(" (using HTTP/0.9)");break;
    case 2: printf(" (using HTTP/1.1)");break;
}

printf("\n");
*/

printf("Runing info: ");

if(clients==1)
    printf("1 client");
else
    printf("%d clients",clients);

printf(", running %d sec", benchtime);

if(force) printf(", early socket close");
if(proxyhost!=NULL) printf(", via proxy server %s:%d",proxyhost,proxyport);
if(force_reload) printf(", forcing reload");
```

```c
    printf(".\n");

    return bench();
}


void build_request(const char *url)
{
    char tmp[10];
    int i;

    //bzero(host,MAXHOSTNAMELEN);
    //bzero(request,REQUEST_SIZE);
    memset(host,0,MAXHOSTNAMELEN);
    memset(request,0,REQUEST_SIZE);

    if(force_reload && proxyhost!=NULL && http10<1) http10=1;
    if(method==METHOD_HEAD && http10<1) http10=1;
    if(method==METHOD_OPTIONS && http10<2) http10=2;
    if(method==METHOD_TRACE && http10<2) http10=2;
```

```
switch(method)
{
    default:
    case METHOD_GET: strcpy(request,"GET");break;
    case METHOD_HEAD: strcpy(request,"HEAD");break;
    case METHOD_OPTIONS: strcpy(request,"OPTIONS");break;
    case METHOD_TRACE: strcpy(request,"TRACE");break;
}

strcat(request," ");

if(NULL==strstr(url,"://"))
{
    fprintf(stderr, "\n%s: is not a valid URL.\n",url);
    exit(2);
}
if(strlen(url)>1500)
{
    fprintf(stderr,"URL is too long.\n");
    exit(2);
}
```

```
    if (0!=strncasecmp("http://",url,7))
    {
        fprintf(stderr,"\nOnly HTTP protocol is directly supported, set --
proxy for others.\n");
        exit(2);
    }

    /* protocol/host delimiter */
    i=strstr(url,"://")-url+3;

    if(strchr(url+i,'/')==NULL) {
        fprintf(stderr,"\nInvalid URL syntax - hostname don't ends with
'/'.\n");
        exit(2);
    }

    if(proxyhost==NULL)
    {
        /* get port from hostname */
        if(index(url+i,':')!=NULL && index(url+i,':')<index(url+i,'/'))
        {
            strncpy(host,url+i,strchr(url+i,':')-url-i);
            //bzero(tmp,10);
            memset(tmp,0,10);
            strncpy(tmp,index(url+i,':')+1,strchr(url+i,'/')-
index(url+i,':')-1);
            /* printf("tmp=%s\n",tmp); */
            proxyport=atoi(tmp);
            if(proxyport==0) proxyport=80;
        }
        else
        {
            strncpy(host,url+i,strcspn(url+i,"/"));
        }
        // printf("Host=%s\n",host);
        strcat(request+strlen(request),url+i+strcspn(url+i,"/"));
    }
    else
    {
        // printf("ProxyHost=%s\nProxyPort=%d\n",proxyhost,proxyport);
        strcat(request,url);
    }

    if(http10==1)
```

```
        strcat(request," HTTP/1.0");
    else if (http10==2)
        strcat(request," HTTP/1.1");

    strcat(request,"\r\n");

    if(http10>0)
        strcat(request,"User-Agent: WebBench "PROGRAM_VERSION"\r\n");
    if(proxyhost==NULL && http10>0)
    {
        strcat(request,"Host: ");
        strcat(request,host);
        strcat(request,"\r\n");
    }

    if(force_reload && proxyhost!=NULL)
    {
        strcat(request,"Pragma: no-cache\r\n");
    }

    if(http10>1)
        strcat(request,"Connection: close\r\n");
```

```c
    /* add empty line at end */
    if(http10>0) strcat(request,"\r\n");

    printf("\nRequest:\n%s\n",request);
}


/* vraci system rc error kod */
static int bench(void)
{
    int i,j,k;
    pid_t pid=0;
    FILE *f;

    /* check avaibility of target server */
    i=Socket(proxyhost==NULL?host:proxyhost,proxyport);
    if(i<0) {
        fprintf(stderr,"\nConnect to server failed. Aborting benchmark.\n");
        return 1;
    }
    close(i);
```

```
/* create pipe */
if(pipe(mypipe))
{
    perror("pipe failed.");
    return 3;
}

/* not needed, since we have alarm() in childrens */
/* wait 4 next system clock tick */
/*
cas=time(NULL);
while(time(NULL)==cas)
sched_yield();
*/

/* fork childs */
for(i=0;i<clients;i++)
{
    pid=fork();
    if(pid <= (pid_t) 0)
    {
        /* child process or error*/
        sleep(1); /* make childs faster */
        break;
    }
}

if( pid < (pid_t) 0)
{
    fprintf(stderr,"problems forking worker no. %d\n",i);
    perror("fork failed.");
    return 3;
}

if(pid == (pid_t) 0)
{
    /* I am a child */
    if(proxyhost==NULL)
        benchcore(host,proxyport,request);
    else
        benchcore(proxyhost,proxyport,request);

    /* write results to pipe */
    f=fdopen(mypipe[1],"w");
    if(f==NULL)
    {
        perror("open pipe for writing failed.");
        return 3;
    }
    /* fprintf(stderr,"Child - %d %d\n",speed,failed); */
    fprintf(f,"%d %d %d\n",speed,failed,bytes);
    fclose(f);

    return 0;
}
else
{
    f=fdopen(mypipe[0],"r");
    if(f==NULL)
    {
        perror("open pipe for reading failed.");
        return 3;
    }

    setvbuf(f,NULL,_IONBF,0);
    speed=0;
    failed=0;
    bytes=0;

    while(1)
    {
        pid=fscanf(f,"%d %d %d",&i,&j,&k);
        if(pid<2)
        {
            fprintf(stderr,"Some of our childrens died.\n");
            break;
        }

        speed+=i;
        failed+=j;
        bytes+=k;

        /* fprintf(stderr,"*Knock* %d %d read=%d\n",speed,failed,pid);
*/
        if(--clients==0) break;
    }
    fclose(f);

    printf("\nSpeed=%d pages/min, %d bytes/sec.\nRequests: %d susceed, 
%d failed.\n",
        (int)((speed+failed)/(benchtime/60.0f)),
        (int)(bytes/(float)benchtime),
        speed,
        failed);
}

return i;
}


void benchcore(const char *host,const int port,const char *req)
{
int rlen;
char buf[1500];
int s,i;
struct sigaction sa;
```

```
/* setup alarm signal handler */
sa.sa_handler=alarm_handler;
sa.sa_flags=0;
if(sigaction(SIGALRM,&sa,NULL))
    exit(3);

alarm(benchtime); // after benchtime,then exit

rlen=strlen(req);
nexttry:while(1)
{
    if(timerexpired)
    {
        if(failed>0)
        {
            /* fprintf(stderr,"Correcting failed by signal\n"); */
            failed--;
        }
        return;
    }

    s=Socket(host,port);
    if(s<0) { failed++;continue;}
    if(rlen!=write(s,req,rlen)) {failed++;close(s);continue;}
    if(http10==0)
    if(shutdown(s,1)) { failed++;close(s);continue;}
    if(force==0)
    {
        /* read all available data from socket */
        while(1)
        {
            if(timerexpired) break;
            i=read(s,buf,1500);
            /* fprintf(stderr,"%d\n",i); */
            if(i<0)
            {
                failed++;
                close(s);
                goto nexttry;
            }
            else
            if(i==0) break;
            else
            bytes+=i;
            }
        }
        if(close(s)) {failed++;continue;}
        speed++;
    }
}
```

```
CFLAGS?=-Wall -ggdb -W -O
CC?=gcc
LIBS?=
LDFLAGS?=
PREFIX?=/usr/local/webbench
VERSION=1.5
TMPDIR=/tmp/webbench-$(VERSION)

all:   webbench tags

tags:  *.c
-ctags *.c

install: webbench
install -d $(DESTDIR)$(PREFIX)/bin
install -s webbench $(DESTDIR)$(PREFIX)/bin
ln -sf $(DESTDIR)$(PREFIX)/bin/webbench $(DESTDIR)/usr/local/bin/webbench

install -d $(DESTDIR)/usr/local/man/man1
install -d $(DESTDIR)$(PREFIX)/man/man1
install -m 644 webbench.1 $(DESTDIR)$(PREFIX)/man/man1
ln -sf $(DESTDIR)$(PREFIX)/man/man1/webbench.1
```

```
$(DESTDIR)/usr/local/man/man1/webbench.1

install -d $(DESTDIR)$(PREFIX)/share/doc/webbench
install -m 644 debian/copyright $(DESTDIR)$(PREFIX)/share/doc/webbench
install -m 644 debian/changelog $(DESTDIR)$(PREFIX)/share/doc/webbench

webbench: webbench.o Makefile
$(CC) $(CFLAGS) $(LDFLAGS) -o webbench webbench.o $(LIBS)

clean:
-rm -f *.o webbench *~ core *.core tags

tar:   clean
-debian/rules clean
rm -rf $(TMPDIR)
install -d $(TMPDIR)
cp -p Makefile webbench.c socket.c webbench.1 $(TMPDIR)
install -d $(TMPDIR)/debian
-cp -p debian/* $(TMPDIR)/debian
ln -sf debian/copyright $(TMPDIR)/COPYRIGHT
ln -sf debian/changelog $(TMPDIR)/ChangeLog
```

```
-cd $(TMPDIR) && cd .. && tar cozf webbench-$(VERSION).tar.gz webbench-
$(VERSION)

webbench.o:webbench.c socket.c Makefile

.PHONY: clean install all tar
```

C 程 序 设 计
C Programming

4

谢谢观看

阅读课程

厦門大學
XIAMEN UNIVERSITY

信 息 学 院
（国家示范性软件学院）
School of Informatics

黄 煒
博士，副教授
Dr. Wei Huang