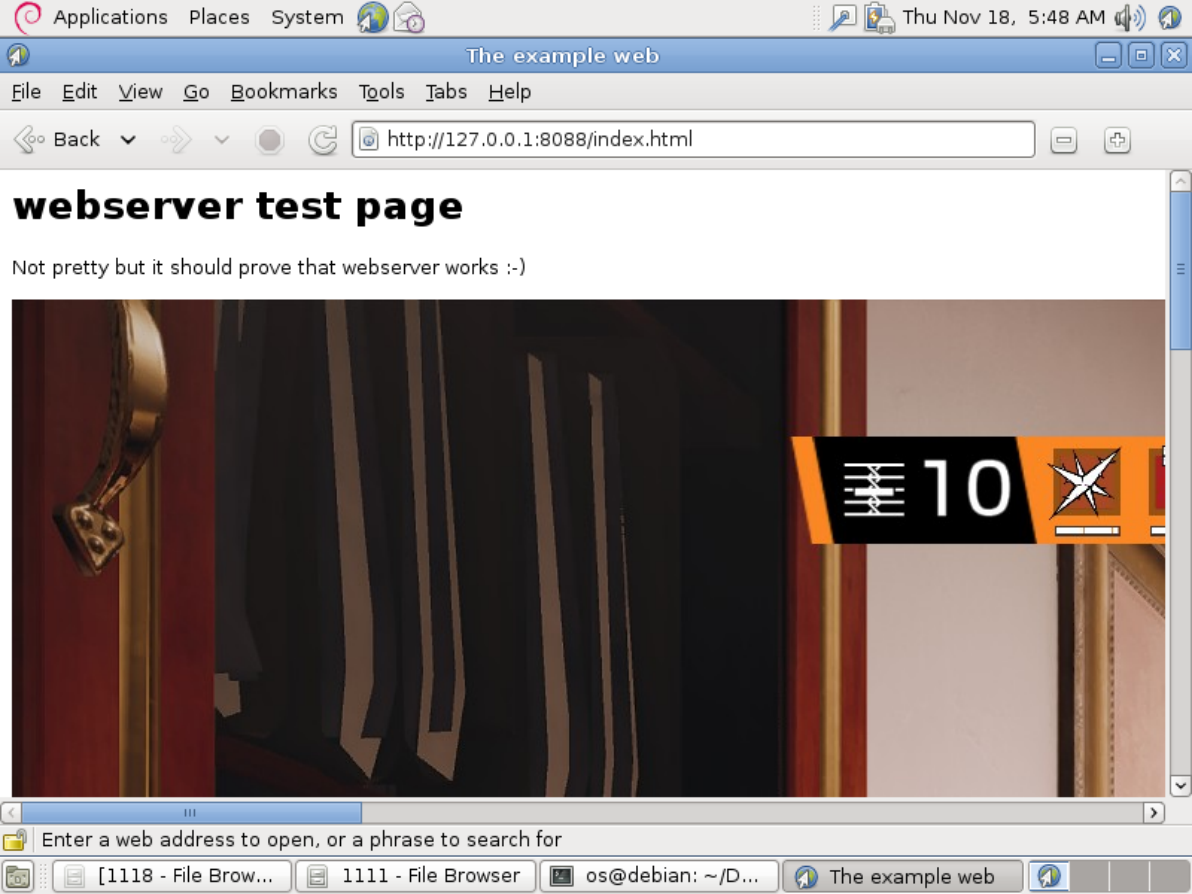
2021.11.11 OS 课设——web服务器初步实现

题目1、2：成功执行程序webserver.c，如图



题目三：



Logbuffer用time库中的struct tm \*p来获取当前年月日时分秒。并显示在webserver.log里面。



图中date是时间信息，是\*p变量得出的结果

至于为什么请求一次网页，webserver接收到很多次请求，因为一次访问本程序设置的有三个请求信息，第一次返回html，第二次返回jpg，第三次返回.ico。从webserver.log中可以看出这三个返回信息。另外，由于文件夹中没有ico，所以在ico的返回日志中写的是not found。

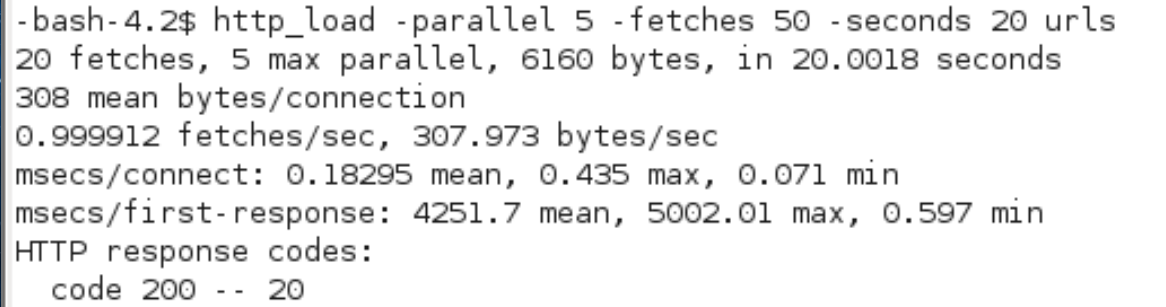
加快html网页速度的技术：1、页面减肥：删除不必要的空格、注释；2、减少文件数量；3、减少域名查询；4、缓存重新使用刷新；5、减少inline JavaScript的数量。6、指定图像和表格的大小。

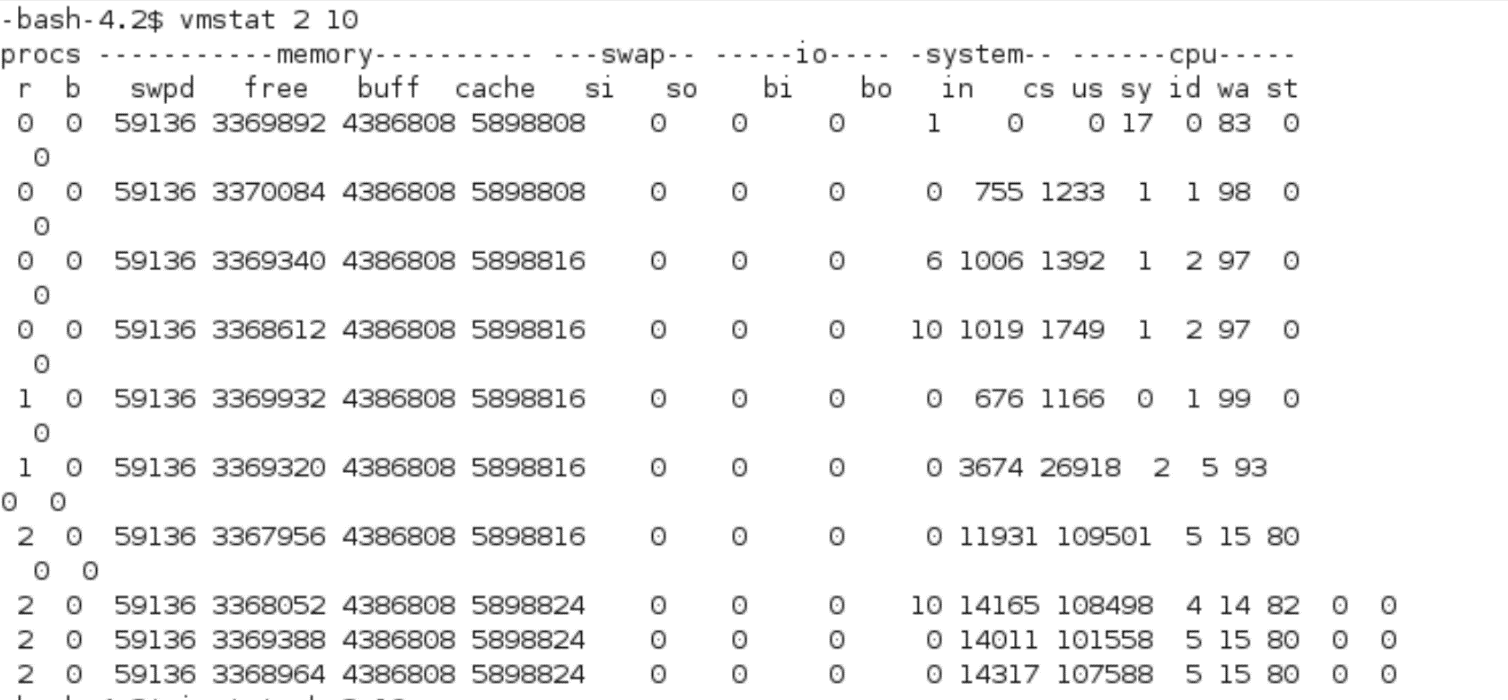


第四题，因为每次刷新浏览器要先加载网页的html文件，再从访问中获取jpg图片，就是第二题中显示的图片，第三次返回ico文件。所以会有延迟。

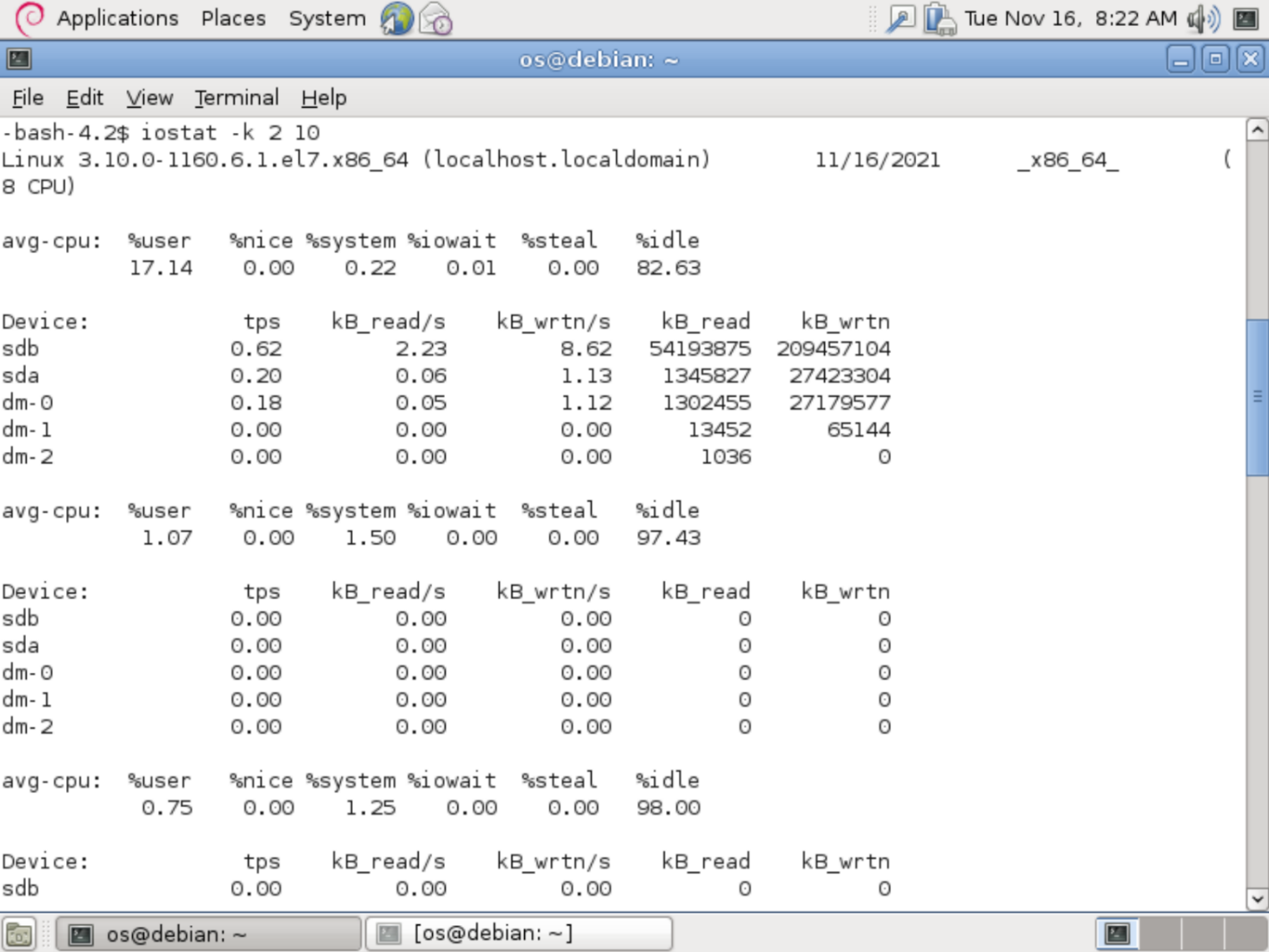


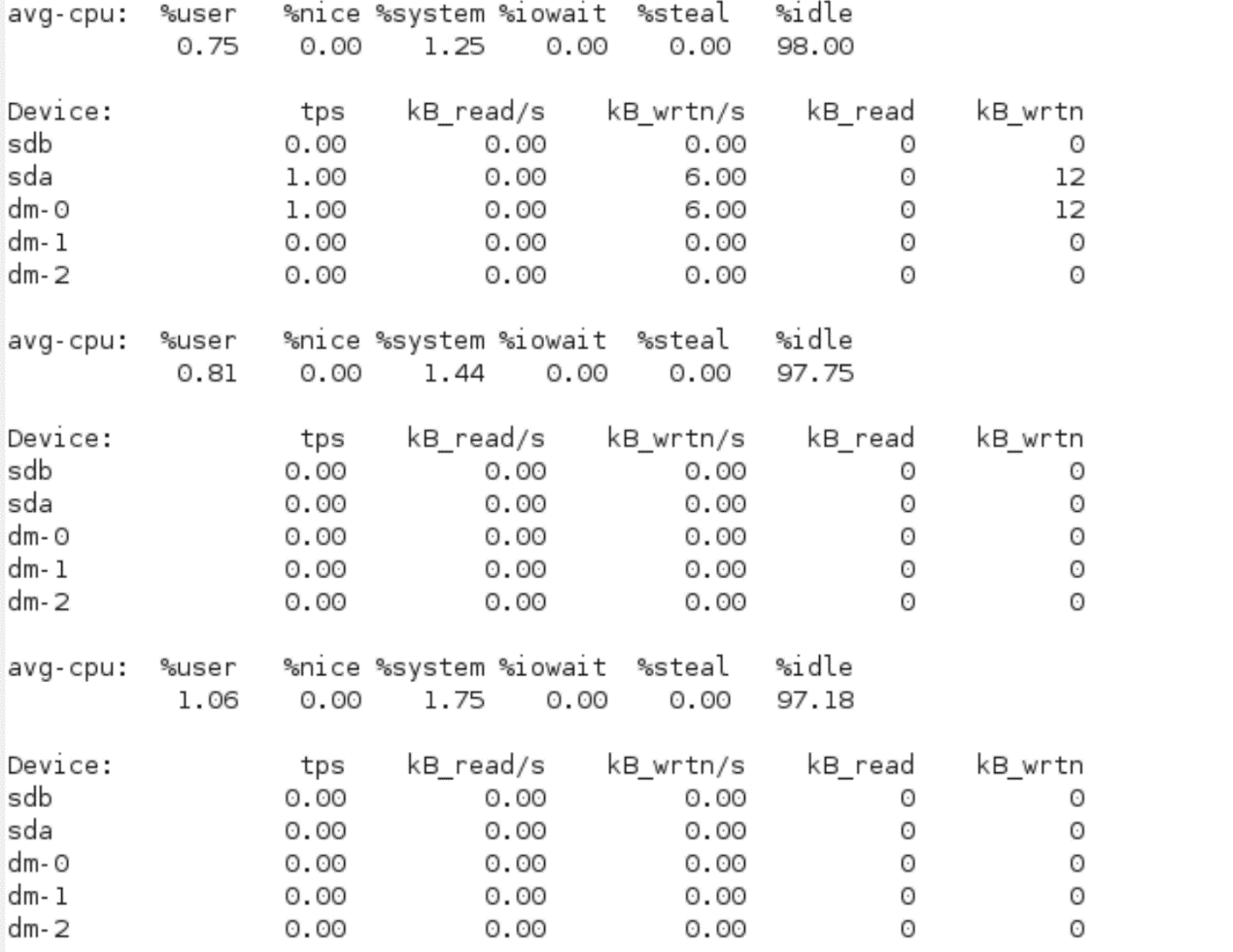


第五题，执行http\_load,在20.0018秒内最大启动了5个客户端，完成20次抓取，共传输6160个字节，没有完成50次网页抓取。每次连接平均传输308个字节。每秒平均完成0.999912次网页传输，每秒传输307.973字节的数据。建立请求连接的平均时间为0.18295毫秒，最大时间是0.435ms，最小0.071ms。每个连接从发出http请求的第一个响应平均时间是4251.7ms，最大5002.01ms，最小0.597ms。响应代码为200的有20个。



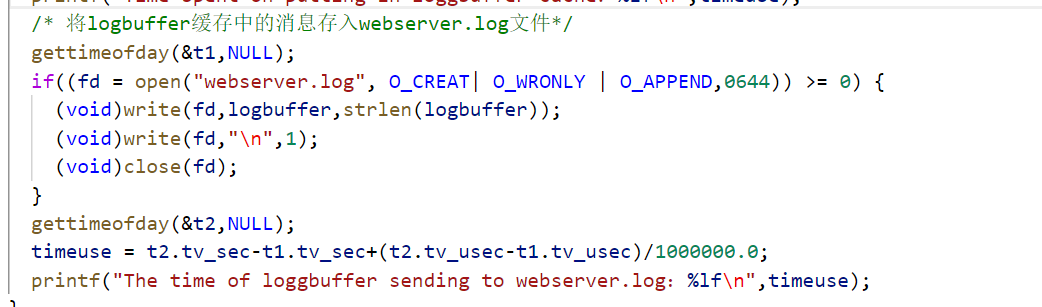
Vmstat显示，刚开始有0个任务，后来加至一个，再后来变两个；但一直没有进程阻塞。虚拟内存一直使用59136byte。空闲物理内存一直保持在3369320-3370084之间，缓冲区大小依次增加8byte。从磁盘读写的数据一直是0，I/O设备读取的一直是0，写入有时会有数据写入。CPU中断次数和上下文交换次数随着任务数量增多而增加，因为是并发操作。而不管几个任务，CPU大部分时间是空闲的，只有少量时间在执行用户进程和系统进程。I/O没有等待时间，也没有从虚拟机偷时间。

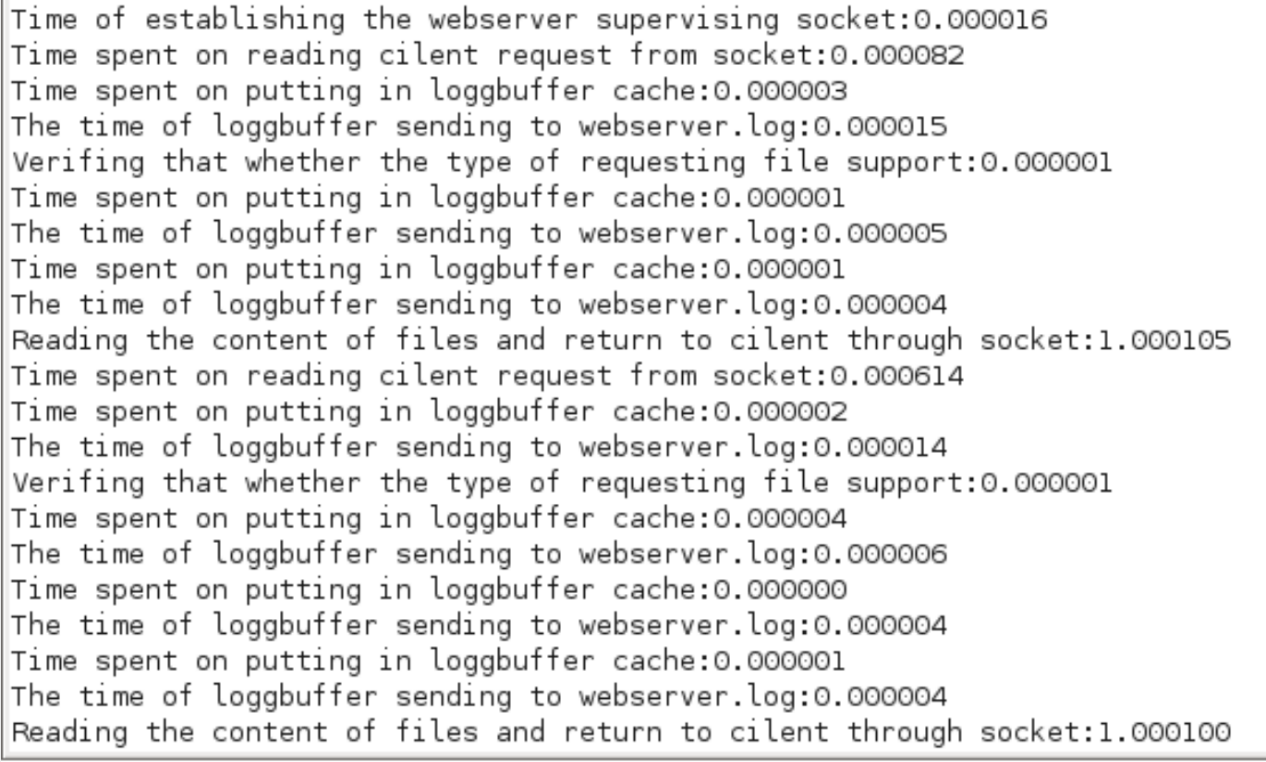


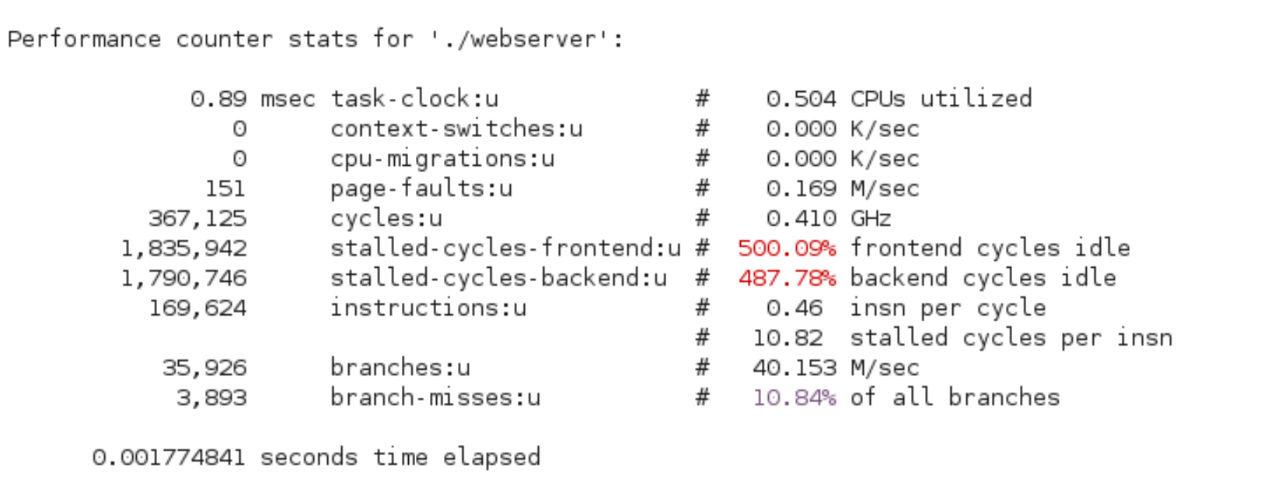


Iostat可知只有在前两秒时候，I/O有请求信息，但是运行webserver没有从磁盘内读取信息，但有时候需要往磁盘内写入。

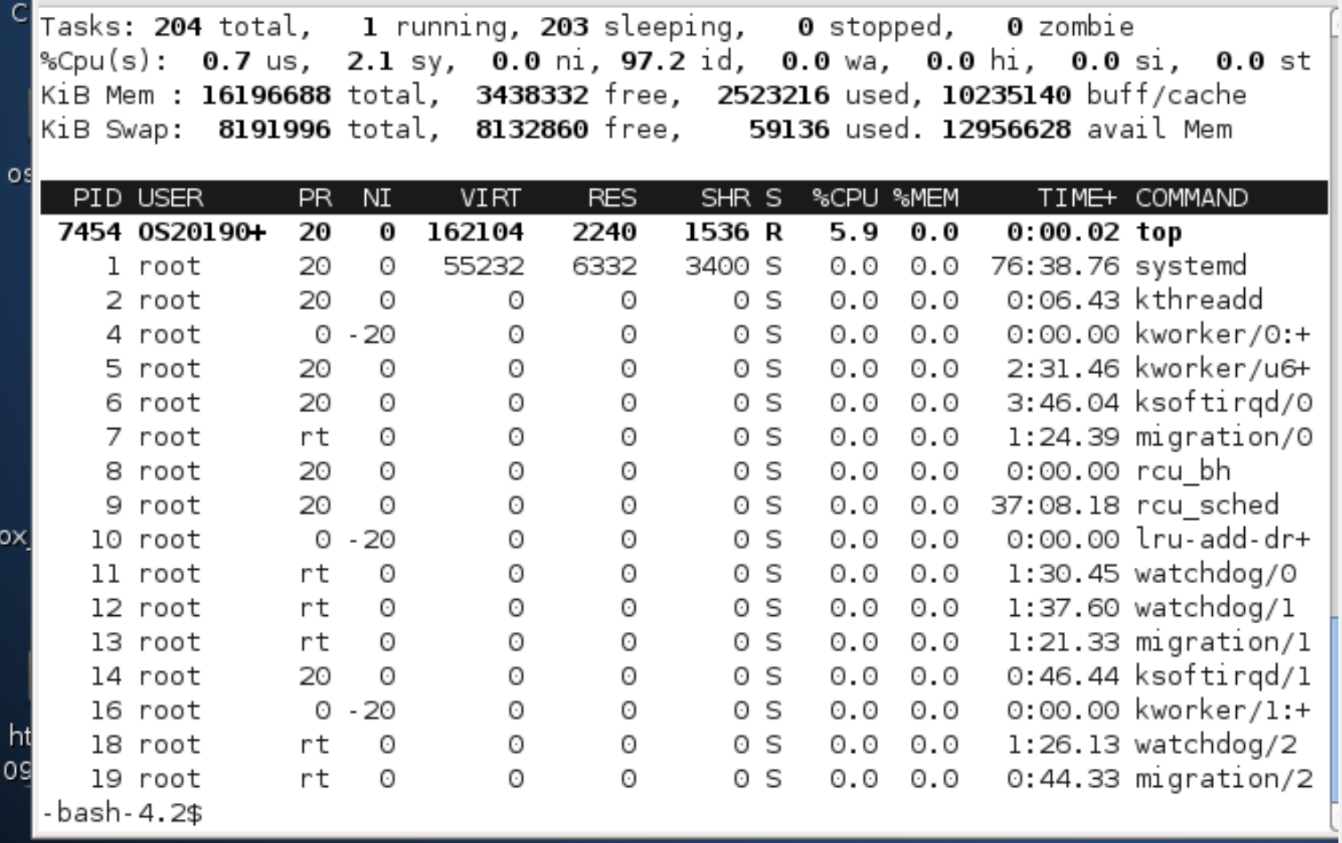
第六题：

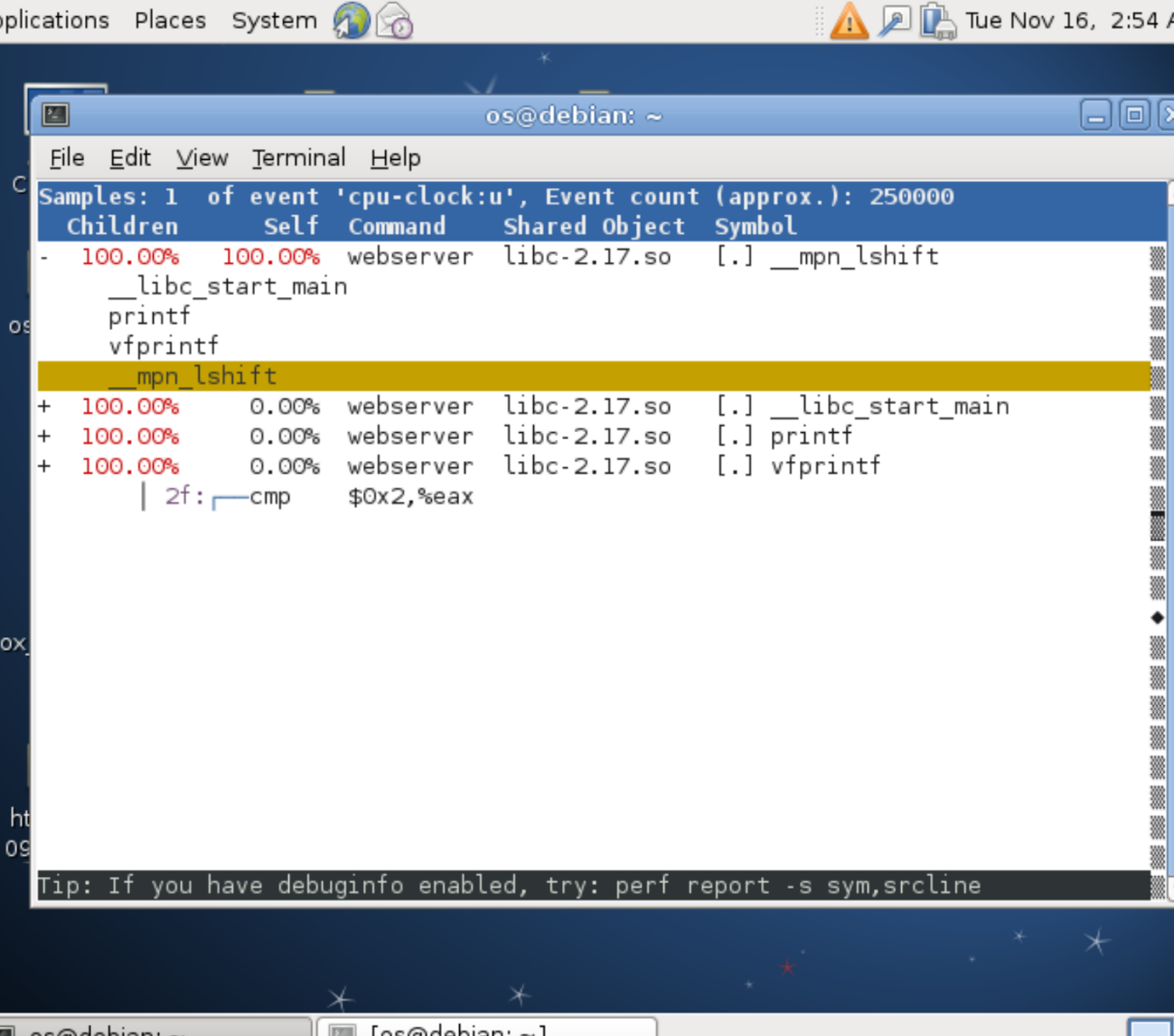
我通过gettimeofday方法获取程序的某一部分开始的时间点，和结束的时间点，做差得到该段程序运行的时间。

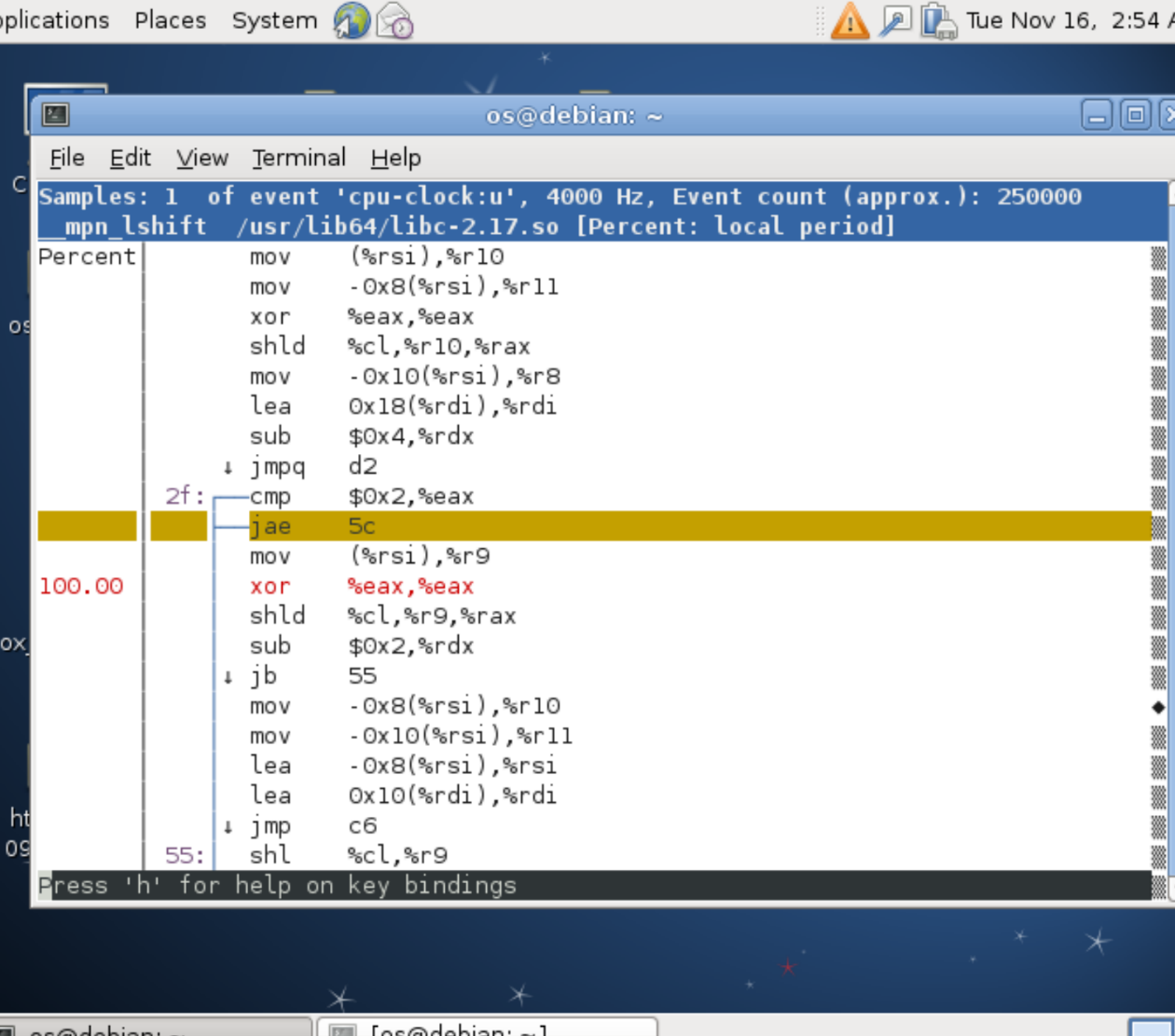
看输出可知，读取文件内容并通过socket返回给客户端耗时最长，时间大约在1s左右，其次是通过socket端口读取用户请求。



CPU利用率为0.504，没有上下交换进程，运行指令没有迁移CPU，缺页151次，CPU逻辑时钟运行367125次，有169624条机器指令，35926个分支，分支预测失败3893次。







通过perf record和report可知，cpu所有的时间都耗费在\_mpn\_lshift函数，打开它的annotate ,看到里面很多的循环跳转语句，说明大量时间耗费在这里。

读取文件内容并通过socket返回给客户端耗时最长，其次是通过socket端口读取用户请求。

第七题：

原因：读取用户信息的时间很长，cpu有大量闲置时间，有两个任务执行时cpu频繁切换执行任务，费时间。且调用了虚拟内存，服务器物理内存容量太小，CPU和虚拟内存数据交互更麻烦，费时间。解决方法：优化web函数中读取用户请求信息的方法，减小时间复杂度，优化系统的CPU调度，减少闲置时间；扩大物理内存，尽量不要用虚拟内存。

——2021.11.16

附 源代码：

/\*Server Code\*/

/\* webserver.c\*/

/\*The  following  main  code  from  https://github.com/ankushagarwal/nweb\*,  but  they  are  modified

slightly\*/

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <fcntl.h>

#include <signal.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <time.h>

#define VERSION 23

#define BUFSIZE 8096

#define ERROR      42

#define LOG        44

#define FORBIDDEN 403

#define NOTFOUND  404

#ifndef SIGCLD

#define SIGCLD SIGCHLD

#endif

/\*struct timeval

{

  long tv\_sec;

  long tv\_usec;

};\*/

struct {

  char \*ext;

  char \*filetype;

} extensions [] = {

  {"gif", "image/gif" },

  {"jpg", "image/jpg" },

  {"jpeg","image/jpeg"},

  {"png", "image/png" },

  {"ico", "image/ico" },

  {"zip", "image/zip" },

  {"gz",  "image/gz"  },

  {"tar", "image/tar" },

  {"htm", "text/html" },

  {"html","text/html" },

  {0,0} };

/\* 日志函数，将运行过程中的提示信息记录到webserver.log 文件中\*/

void logger(int type, char \*s1, char \*s2, int socket\_fd)

{

  int fd ;

  struct timeval t1,t2;

  double timeuse;

  char logbuffer[BUFSIZE\*2];

  time\_t timep;

  struct tm \*p;

  time(&timep);

  p = gmtime(&timep);

  /\*根据消息类型，将消息放入logbuffer缓存，或直接将消息通过socket通道返回给客户端\*/

  gettimeofday(&t1,NULL);

  switch (type) {

  case ERROR:

    (void)sprintf(logbuffer,"date:%d\\%d\\%d %d:%d:%d\nERROR: %s:%s Errno=%d exiting pid=%d",(1900+p->tm\_year),(1+p->tm\_mon),p->tm\_mday,p->tm\_hour+8,p->tm\_min,p->tm\_sec,s1, s2, errno,getpid());

    break;

  case FORBIDDEN:

    (void)write(socket\_fd, "HTTP/1.1 403 Forbidden\nContent-Length: 185\nConnection: close\nContent-Type: text/html\n\n<html><head>\n<title>403 Forbidden</title>\n</head><body>\n<h1>Forbidden</h1>\n The requested URL, file type or operation is not allowed on this simple static file webserver.\n</body></html>\n",271);

    (void)sprintf(logbuffer,"date:%d\\%d\\%d %d:%d:%d\nFORBIDDEN: %s:%s",(1900+p->tm\_year),(1+p->tm\_mon),p->tm\_mday,p->tm\_hour+8,p->tm\_min,p->tm\_sec,s1, s2);

    break;

  case NOTFOUND:

    (void)write(socket\_fd,  "HTTP/1.1  404  Not  Found\nContent-Length:  136\nConnection: close\nContent-Type:  text/html\n\n<html><head>\n<title>404  Not Found</title>\n</head><body>\n<h1>Not  Found</h1>\nThe  requested  URL  was  not  found  on  this server.\n</body></html>\n",224);

    (void)sprintf(logbuffer,"date:%d\\%d\\%d %d:%d:%d\nNOT FOUND: %s:%s",(1900+p->tm\_year),(1+p->tm\_mon),p->tm\_mday,p->tm\_hour+8,p->tm\_min,p->tm\_sec,s1, s2);

    break;

  case LOG:

    (void)sprintf(logbuffer,"date:%d\\%d\\%d %d:%d:%d\nINFO: %s:%s:%d",(1900+p->tm\_year),(1+p->tm\_mon),p->tm\_mday,p->tm\_hour+8,p->tm\_min,p->tm\_sec,s1, s2,socket\_fd);

    break;

  }

  gettimeofday(&t2,NULL);

  timeuse = t2.tv\_sec-t1.tv\_sec+(t2.tv\_usec-t1.tv\_usec)/1000000.0;

  printf("Time spent on putting in loggbuffer cache:%lf\n",timeuse);

  /\* 将logbuffer缓存中的消息存入webserver.log文件\*/

  gettimeofday(&t1,NULL);

  if((fd = open("webserver.log", O\_CREAT| O\_WRONLY | O\_APPEND,0644)) >= 0) {

    (void)write(fd,logbuffer,strlen(logbuffer));

    (void)write(fd,"\n",1);

    (void)close(fd);

  }

  gettimeofday(&t2,NULL);

  timeuse = t2.tv\_sec-t1.tv\_sec+(t2.tv\_usec-t1.tv\_usec)/1000000.0;

  printf("The time of loggbuffer sending to webserver.log:%lf\n",timeuse);

}

void web(int fd, int hit)

{

  struct timeval t1,t2;

  double timeuse;

  int j, file\_fd, buflen;

  long i, ret, len;

  char \* fstr;

  static char buffer[BUFSIZE+1]; /\* 设置静态缓冲区 \*/

  gettimeofday(&t1,NULL);

  ret =read(fd,buffer,BUFSIZE);   /\* 从连接通道中读取客户端的请求消息 \*/

  gettimeofday(&t2,NULL);

  timeuse = t2.tv\_sec-t1.tv\_sec+(t2.tv\_usec-t1.tv\_usec)/1000000.0;

  printf("Time spent on reading cilent request from socket:%lf\n",timeuse);

  if(ret == 0 || ret == -1) {  //如果读取客户端消息失败，则向客户端发送HTTP失败响应信息

    logger(FORBIDDEN,"failed to read browser request","",fd);

  }

  if(ret > 0 && ret < BUFSIZE)  /\* 设置有效字符串，即将字符串尾部表示为0 \*/

    buffer[ret]=0;

  else buffer[0]=0;

  for(i=0;i<ret;i++)  /\* 移除消息字符串中的“CF”和“LF”字符\*/

    if(buffer[i] == '\r' || buffer[i] == '\n')

      buffer[i]='\*';

  logger(LOG,"request",buffer,hit);

 /\*判断客户端HTTP请求消息是否为GET类型，如果不是则给出相应的响应消息\*/

  if( strncmp(buffer,"GET ",4) && strncmp(buffer,"get ",4) ) {

    logger(FORBIDDEN,"Only simple GET operation supported",buffer,fd);

  }

  for(i=4;i<BUFSIZE;i++) { /\* null terminate after the second space to ignore extra stuff \*/

    if(buffer[i] == ' ') { /\* string is "GET URL " +lots of other stuff \*/

      buffer[i] = 0;

      break;

    }

  }

  for(j=0;j<i-1;j++)   /\* 在消息中检测路径，不允许路径中出现“.” \*/

    if(buffer[j] == '.' && buffer[j+1] == '.') {

      logger(FORBIDDEN,"Parent directory (..) path names not supported",buffer,fd);

    }

  if( !strncmp(&buffer[0],"GET /\0",6) || !strncmp(&buffer[0],"get /\0",6) )

   /\* 如果请求消息中没有包含有效的文件名，则使用默认的文件名index.html \*/

    (void)strcpy(buffer,"GET /index.html");

  /\* 根据预定义在extensions中的文件类型，检查请求的文件类型是否本服务器支持 \*/

  gettimeofday(&t1,NULL);

  buflen=strlen(buffer);

  fstr = (char \*)0;

  for(i=0;extensions[i].ext != 0;i++) {

    len = strlen(extensions[i].ext);

    if( !strncmp(&buffer[buflen-len], extensions[i].ext, len)) {

      fstr =extensions[i].filetype;

      break;

    }

  }

  if(fstr == 0) logger(FORBIDDEN,"file extension type not supported",buffer,fd);

  gettimeofday(&t2,NULL);

  timeuse = t2.tv\_sec-t1.tv\_sec+(t2.tv\_usec-t1.tv\_usec)/1000000.0;

  printf("Verifing that whether the type of requesting file support:%lf\n",timeuse);

  if(( file\_fd = open(&buffer[5],O\_RDONLY)) == -1) {  /\* 打开指定的文件名\*/

    logger(NOTFOUND, "failed to open file",&buffer[5],fd);

  }

  logger(LOG,"SEND",&buffer[5],hit);

  len = (long)lseek(file\_fd, (off\_t)0, SEEK\_END); /\* 通过lseek 获取文件长度\*/

   (void)lseek(file\_fd, (off\_t)0, SEEK\_SET); /\* 将文件指针移到文件首位置\*/

  (void)sprintf(buffer,"HTTP/1.1  200  OK\nServer:  nweb/%d.0\nContent-Length:  %ld\nConnection: close\nContent-Type: %s\n\n", VERSION, len, fstr); /\* Header + a blank line \*/

  logger(LOG,"Header",buffer,hit);

  (void)write(fd,buffer,strlen(buffer));

  /\* 不停地从文件里读取文件内容，并通过socket通道向客户端返回文件内容\*/

  gettimeofday(&t1,NULL);

  while (  (ret = read(file\_fd, buffer, BUFSIZE)) > 0 ) {

    (void)write(fd,buffer,ret);

  }

  sleep(1);  /\* sleep的作用是防止消息未发出，已经将此socket通道关闭\*/

  close(fd);

  gettimeofday(&t2,NULL);

  timeuse = t2.tv\_sec-t1.tv\_sec+(t2.tv\_usec-t1.tv\_usec)/1000000.0;

  printf("Reading the content of files and return to cilent through socket:%lf\n",timeuse);

}

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

{

  struct timeval t1,t2;

  double timeuse;

  int i, port, listenfd, socketfd, hit;

  socklen\_t length;

  static struct sockaddr\_in cli\_addr; /\* static = initialised to zeros \*/

  static struct sockaddr\_in serv\_addr; /\* static = initialised to zeros \*/

  /\*解析命令参数\*/

  gettimeofday(&t1,NULL);

  if( argc < 3  || argc > 3 || !strcmp(argv[1], "-?") ) {

    (void)printf("hint: nweb Port-Number Top-Directory\t\tversion %d\n\n"

  "\tnweb is a small and very safe mini web server\n"

  "\tnweb only servers out file/web pages with extensions named below\n"

  "\t and only from the named directory or its sub-directories.\n"

  "\tThere is no fancy features = safe and secure.\n\n"

  "\tExample:webserver 8181 /home/nwebdir &\n\n"

  "\tOnly Supports:", VERSION);

  gettimeofday(&t2,NULL);

  timeuse = t2.tv\_sec-t1.tv\_sec+(t2.tv\_usec-t1.tv\_usec)/1000000.0;

  printf("Analysing the parameters of command:%lf\n",timeuse);

    for(i=0;extensions[i].ext != 0;i++)

      (void)printf(" %s",extensions[i].ext);

    (void)printf("\n\tNot Supported: URLs including \"..\", Java, Javascript, CGI\n"

  "\tNot Supported: directories / /etc /bin /lib /tmp /usr /dev /sbin \n"

  "\tNo warranty given or implied\n\tNigel Griffiths nag@uk.ibm.com\n"  );

    exit(0);

  }

  if( !strncmp(argv[2],"/"   ,2 ) || !strncmp(argv[2],"/etc", 5 ) ||

      !strncmp(argv[2],"/bin",5 ) || !strncmp(argv[2],"/lib", 5 ) ||

      !strncmp(argv[2],"/tmp",5 ) || !strncmp(argv[2],"/usr", 5 ) ||

      !strncmp(argv[2],"/dev",5 ) || !strncmp(argv[2],"/sbin",6) ){

    (void)printf("ERROR: Bad top directory %s, see nweb -?\n",argv[2]);

    exit(3);

  }

  if(chdir(argv[2]) == -1){

    (void)printf("ERROR: Can't Change to directory %s\n",argv[2]);

    exit(4);

  }

  /\* 建立服务端侦听socket\*/

  gettimeofday(&t1,NULL);

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

    logger(ERROR, "system call","socket",0);

  port = atoi(argv[1]);

  if(port < 0 || port >60000)

    logger(ERROR,"Invalid port number (try 1->60000)",argv[1],0);

  serv\_addr.sin\_family = AF\_INET;

  serv\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

  serv\_addr.sin\_port = htons(port);

  if(bind(listenfd, (struct sockaddr \*)&serv\_addr,sizeof(serv\_addr)) <0)

    logger(ERROR,"system call","bind",0);

  if( listen(listenfd,64) <0)

    logger(ERROR,"system call","listen",0);

  gettimeofday(&t2,NULL);

  timeuse = t2.tv\_sec-t1.tv\_sec+(t2.tv\_usec-t1.tv\_usec)/1000000.0;

  printf("Time of establishing the webserver supervising socket:%lf\n",timeuse);

  for(hit=1; hit<=4;hit++) {

    length = sizeof(cli\_addr);

    if((socketfd = accept(listenfd, (struct sockaddr \*)&cli\_addr, &length)) < 0)

      logger(ERROR,"system call","accept",0);

    web(socketfd,hit);

    printf("current pid is:%d\n",getpid());

  }

}