컴퓨터 네트워크

- TermProject – FTP -

제 출 일	2017년 12월 08일
분 반	00반
담당교수	김기일
학 과	컴퓨터공학과
학 번	201302423
이 름	신종욱

1. 주요코드 및 분석

Server의 upload

```
int file_upload(int sockfd, char *filename)
{
    int fd;//디스크립터 저장용 변수
    int readn;
    int writen;
    char buf[MAXLINE];
    struct sockaddr_in addr;
    int addrlen;
    char tstr[22];//현재시각을 리턴해서 char배열로 저장하기위한 배열
    char sizech[6];//파일사이즈 저장할 배열
    int size;//사이즈를 추출하고 저장할 변수
    int ret;
    FILE *fp;//사이즈를 추출할때 쓰는 FILE 변수
    int count;
    char countch[5];//현재의 리스트파일을 읽고 count를 확인
    if( (fd = open(filename, O_WRONLY|O_CREAT|O_TRUNC)) == -1)
    {//읽어온 파일명이 없다면 생성하고 이미 있다면 지우고 다시 재생성한다.
        return -1;
    }
    memset(buf, 0x00, MAXLINE);
    while((readn = recv(sockfd, buf, MAXLINE, 0)) > 0)//다 읽을때까지 읽는다.
    {
        writen = write(fd, buf, readn);//읽은 양 만큼 작성한다
        if(writen < 0) break;
        memset(buf, 0x00, MAXLINE);
    }
    close(fd);//파일 업로드가 끝난후 파일을 닫는다
    fp = fopen(filename, "r");//size 측정을 위해 다시 fopen으로 연다
    fseek(fp, 0, SEEK_END);
    size = ftell(fp); //fseek를 이용하여서 파일크기를 측정한다
    sprintf(sizech,"%d",size);//출력을 위해 char배열에 저장
    fclose(fp);//사용이 끝났음으로 파일을 닫는다

    ret = open("./list.txt", O_WRONLY|O_CREAT|O_APPEND);//리스트 파일이 없으면 새로열고 있으면 계쏙 추가하는 식이다.
    count=fline_cnt("./list.txt");//현재 리스트의 줄수를 세서 리턴
    count = count/5;//5줄마다 한 파일씩 작성되기때문에 5로 나누면 count 1개이다.
```

이부분 까지 client에서 보낸 버퍼를 recv로 받고 write하여서 파일을 쓰는 과정이다.
그리고 리스트문서를 작성하기위해서 준비하는 단계이다.

```
    whattime(tstr);//현재시각을 리턴하여서 저장
    write(ret,"name = ",7);
    write(ret,filename,strlen(filename));//파일명을 입력
    write(ret,"\nip = ",6);//ip주소 입력
    write(ret,inet_ntoa(addr.sin_addr),strlen(inet_ntoa(addr.sin_addr)));
    write(ret,"\nup_date = ",11);
    write(ret,tstr,strlen(tstr));
    write(ret,"\ncount = ",9);
    write(ret,countch,strlen(countch));
    write(ret,"\nsize = ",8);//앞서 저장한 시각 카운트 사이즈 입력
    write(ret,sizech,strlen(sizech));
    write(ret,"\n",1);//출력에 필요한 양식으로 list.txt에 출력을 한다

    addrlen = sizeof(addr);
    getpeername(sockfd, (struct sockaddr *)&addr, &addrlen);
    printf("File Upload %s\n", inet_ntoa(addr.sin_addr));
    return 1;
```

현재 ret는 list.txt의 위치를 저장하고 있고
리스트에 저장할 현재시각을 저장하고 name ip 시각 count와 size를 차례대로 입력한다.

Server의 download

```
int file_download(int sockfd, char *filename)
{//다운로드 요청이왔을경우에는 서버의 입장에선 클라이언트로 보내야한다
    int fd;
    int readn;
    int sendn;
    char buf[MAXLINE];
    if( (fd = open(filename, O_RDONLY)) == -1)
    {
        printf("open error\n");
        return -1;
    }//원하는 파일이 있는지 확인후
    memset(buf, 0x00, MAXLINE);
    while( (readn = read(fd, buf, MAXLINE)) > 0)
    {
        sendn = send(sockfd, buf, readn, 0);
        if(sendn < 0) break;
        memset(buf, 0x00, MAXLINE);
    }//파일을 읽으면서 보낸다
    if (sendn == -1) {
    printf("send error\n");
    return -1;}
    return 1;
}
```

다운로드 요청은 요청된 파일이름이 있는지 확인 후 파일을 읽으면서 send해주면된다.

server의 list

```
int file_list(int sockfd)
{//리스트를 요청할땐 list.txt에 있는 데이터를 클라이언트로 보내면된다.
    //다운로드와 같이 전송하는것은 같지만 클라이언트에서 파일로 저장할지 바로 출력할지 선택하면된다.
    char *ErrMsg;
    int ret;
    int readn;
    int sendn;
    char buf[MAXLINE];
    ret = open("./list.txt", O_RDONLY);
    if(ret == -1)
    {
        printf("Open Error\n");
        return -1;
    }
    printf("Open Success\n");
    while( 0 < ( readn = read( ret, buf, MAXLINE-1)))    //
    {
      sendn = send(sockfd, buf, readn, 0);//읽으면서 보낸다.
      if(sendn < 0) break;
      memset(buf, 0x00, MAXLINE);
    }
    return 1;
}
```

리스트도 마찬가지로 똑같이 파일내부를 읽고 send해주면된다.

client의 download

```
int download(int sockfd, char *file)
{//다운로드를 요청할경우
    struct Cquery query;
    int fd;
    int readn, writen;
    char buf[MAXLINE];
    if( (fd = open(file, O_WRONLY|O_CREAT|O_TRUNC)) == -1 )
    {//파일이 없다면 생성하고 있더라면 덮어쓰기를 하여 파일을 생성한다.
        printf("file make error");
        return -1;
    }
    memset(buf, 0x00, MAXLINE);
    memset(&query, 0x00, sizeof(query));
    query.command = htonl(Q_DOWNLOAD);
    sprintf(query.fname, "%s", file);//sprintf로 char배열로 옮겨준다
    if(send(sockfd, (void *)&query, sizeof(query), 0) <=0)
    {//커리 요청 메세지를 보낸다
        return -1;
    }
    while((readn = recv(sockfd, buf, MAXLINE, 0)) > 0)
    {
        writen = write(fd, buf, readn);//서버로부터 받은 버퍼를 파일에 작성한다.
        if(writen != readn)
        {
            return -1;
        }
        memset(buf, 0x00, MAXLINE);
    }
    close(fd);
    return 1;
}
```

다운로드시에는 파일을 덮어쓰기형태로 일단 파일을 생성후에 서버에서 보내는 파일을 읽으면서 파일에 작성하는 식이다.

client의 upload

```
int upload(int sockfd, char *file)
{
    struct Cquery query;
    int fd;
    int readn;
    int sendn;
    char buf[MAXLINE];
    if( (fd = open(file, O_RDONLY)) == -1 ) {
        return -1;
    }//파일명을 읽고 디스크립퍼를 저장한다
    memset(&query, 0x00, sizeof(query));
    query.command = htonl(Q_UPLOAD);
    sprintf(query.fname, "%s", file);
    if(send(sockfd, (void *)&query, sizeof(query), 0) <=0) {
    return -1;
    }
    while((readn = read(fd, buf, MAXLINE)) > 0)
    {//파일을 읽을게 없을때 까지 읽고
        sendn = send(sockfd, buf, readn, 0);//읽은 버퍼를 보내준다.
        if(sendn != readn)
        {
            printf("Upload Error\n");
        return -1;
        }
    }
    close(fd);
    return 1;
}
```

업로드일땐 읽으면서 server에게 send해주면된다.

client의 list

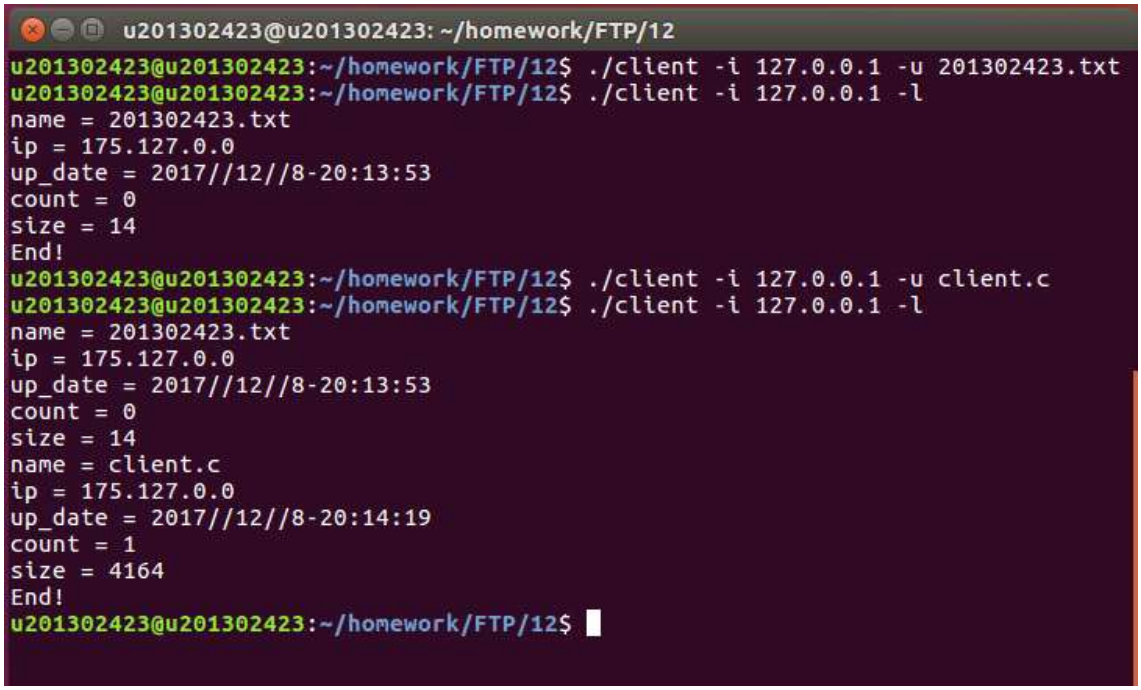
```
int get_list(int sockfd)
{
    struct Cquery query;
    char buf[MAXLINE];
    int len;
    int readn;
    memset(&query, 0x00, sizeof(query));
    query.command = htonl(Q_LIST);
    if(send(sockfd, (void *)&query, sizeof(query), 0) <=0 )
    {//list요청 메세지를 보냄
        printf("List Send Error\n");
        return -1;
    }
    memset(buf, 0x00, MAXLINE);
    while(1)
    {
        len = recv(sockfd, buf, MAXLINE, 0);
        if(len <= 0) break;
        printf("%s", buf);
        memset(buf, 0x00, MAXLINE);
    }//받은 버퍼를 바로 출력해준다.
    printf("End!\n");
}
```

list 요청을한후 돌아오는 버퍼를 저장할필요가 없으니 write없이 바로바로 출력해준다

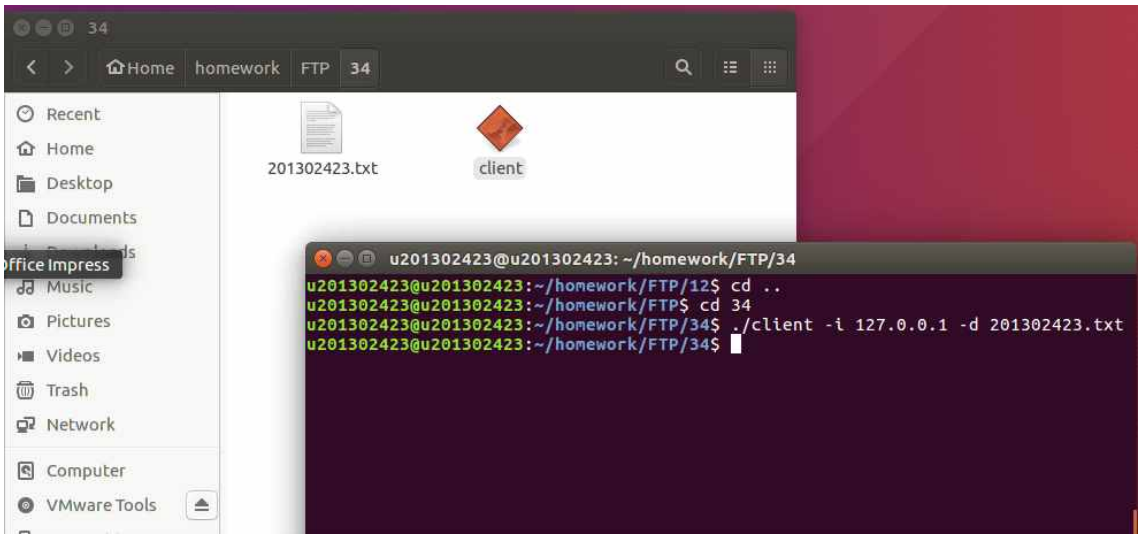
이외 기타코드는 소스코드에 주석으로 처리하였습니다.

2. 실행 결과

실행환경 FTP폴더에서 server파일을 실행해서 서버 실행
12폴더와 34 폴더에 client 파일을 각각 저장
1. 12폴더 client에서 업로드후 리스트 확인



2. 34폴더 이동후 올라온 파일 다운로드 후 파일 생성 확인



3.느낀점

이전에 했던 소켓프로그래밍에서 file open read가 추가된 형태였다. 커리 메시지는 인터넷에서 정보를 얻어서 구현하였다. 이번과제로 리눅스에서 사용하는 전용 c함수들을 많이 알게되었다.