Software Practice2

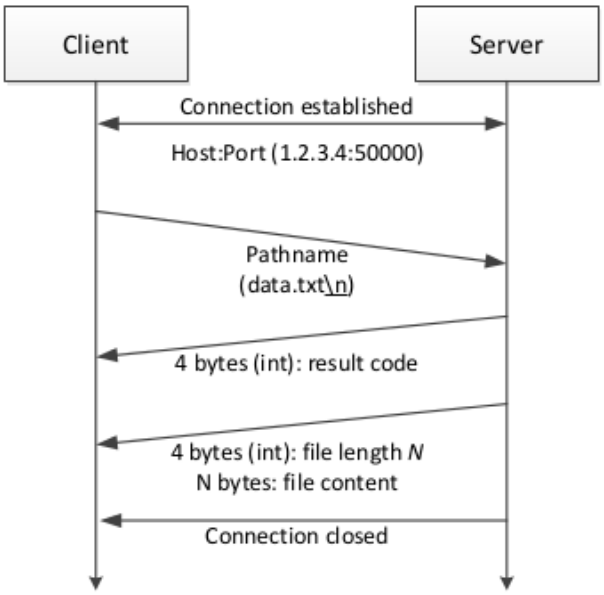
Programming Assignment #4:

File transfer over network

2014313303, 홍태하

1. 전체적인 흐름.
2. Server.c
3. Client.c
4. 실행 화면

Protocol



1. 전체적인 흐름

코드의 전체적인 흐름은 위의 Protocol을 따른다. 먼저 인풋으로 Host와 포트, pathname을 한번에 받기 때문에 그것을 분리하여 각각 저장한다. 그 후 서버와 클라이언트에서 각각 그 호스트와 포트번호로 소켓을 열고 연결한다. 연결이 되면 클라이언트 쪽에서 Pathname을 보내고 서버에서 그 pathname에 위치한 파일을 찾아서 오픈하는데 오픈의 결과에 따라서 result code를 클라이언트 쪽에 보낸다. 그리고 오픈한 파일의 내용과 길이를 보내면 클라이언트 쪽에서는 그것을 받아서 같은 이름의 파일로 저장한다.

2. Server.c

```
if(argv[1] == '\0'){
    port = 10001;
}else{
    port = stoi(argv[1]);
}
```

서버를 열 때 argument를 받을 수도 있고 안 받을 수도 있는데 안 받는 경우에는 포트 번호를 10001로 한다.

```
    if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        return -1;
    }
    bzero((char *)&saddr, sizeof(saddr));
    saddr.sin_family = AF_INET;
    saddr.sin_addr.s_addr = htonl(INADDR_ANY);
    saddr.sin_port = htons(port);
    if (bind(listenfd, (struct sockaddr *)&saddr, sizeof(saddr)) < 0) {
        return -1;
    }
    if (listen(listenfd, 5) < 0) {
        return -1;
    }
    while (1) {
        caddrlen = sizeof(caddr);
        if ((connfd = accept(listenfd, (struct sockaddr *)&caddr, &caddrlen)) < 0
) {
            return -1;
        }
        h = gethostbyaddr((const char *)&caddr.sin_addr.s_addr, sizeof(caddr.sin_
addr.s_addr), AF_INET);
```

그 후 서버를 소켓을 이용하여 열어준다. 각 과정에서 에러가 생길 경우 -1을 리턴하고 종료해준다.

서버를 열고 클라이언트와 연결이 되면 클라이언트가 보낸 pathname의 파일을 보내는 과정을 진행한다. 먼저 병렬적으로 처리하기 위해 자식 프로세스를 생성하여 자식 프로세스에서 클라이언트의 요청 하나를 처리하고 파일 전송을 완료한 후 프로세스를 종료시킨다. 그 후 부모 프로세스는 자식 프로세스를 정리한다.

```
read(connfd, pathname, MAXLINE);
pathname[strlen(pathname)-1]='\0';
```

클라이언트에서 pathname을 받는다. Pathname의 끝에 ₩n이 들어있으므로 그 부분을 null로 바꿔준다.

```
fd = open(pathname, O_RDONLY);
if(fd == -1){
    if(errno == ENOENT){
        result = 1;
        write(connfd,&result,4);
        close(connfd);
        exit(0);
    }else if(errno == EPERM){
        result = 2;
        write(connfd,&result,4);
        close(connfd);
        exit(0);
    }
}else{
    result = 0;
    write(connfd,&result,4);
}
```

받은 pathname을 오픈해준다. 그 후 그 파일이 존재하지 않는 에러가 뜨면 1을 클라이언트로 보내주고 파일을 사용할 권한이 없는 에러가 뜨면 2를 클라이언트로 보내준다. 정상적이라면 0을 클라이언트로 보내준다.

```
while((n=read(fd,buf,MAXLINE))>0){
    write(connfd,&n,4);
    write(connfd,buf,n);
}
```

Result code를 보내주고 오픈한 파일을 읽어서 그 파일을 읽은 길이와 내용을 클라이언트로 보내준다.

보내주는 과정을 마친 후에는 파일과 소켓, 자식 프로세스 등의 자원을 해제해 준다.

3. Client.c

```
input_len = read(STDIN_FILENO,buf,MAXLINE);
buf[input_len] = '\0';
if(!strcmp(buf,"exit\n")){
    exit(EXIT_SUCCESS);
}
```

일단 클라이언트를 에러가 없다면 계속 돌리기 위해 코드 전체적

으로 while문을 씌워준다. 클라이언트는 argument없이 바로 실행하는데 그 후에 인풋을 입력받는다. 그래서 인풋을 받아서 buf에 저장하고 인풋의 길이를 input_len에 저장한다. 인풋에 exit이 들어오면 클라이언트를 종료해준다.

```
for(i=0;i<input_len;i++){
    if(buf[i] == '.'){
        dot_count++;
    }
    if(buf[i] == ':'){
        buf[i] = '\0';
        if(!strcmp(buf,"localhost")){
            host = local;
        }else if(dot_count == 3){
            host = buf;
        }else{
            err_flag=1;
            break;
        }
        port_start = i+1;
        col_flag=1;
    }
    if(buf[i] == '/' && col_flag == 1){
        buf[i] = '\0';
        if(i == port_start){
            port = 10001;
        }else{
            port = stoi(buf+port_start);
        }
        pathname = buf+i+1;
        s_flag = 1;
        buf[i] = '/';
        break;
    }else if(buf[i] == '/' && col_flag == 0){
        buf[i] = '\0';
        if(!strcmp(buf,"localhost")){
            host = local;
        }else if(dot_count == 3){
            host = buf;
        }else{
            err_flag = 1;
            break;
        }
        port = 10001;
        pathname = buf+i+1;
        s_flag = 1;
        buf[i] = '/';
        break;
    }
}
```

input_len만큼 포문을 돌리면서 인풋을 host, port, pathname으로 자른다.

:을 찾으면 그 앞의 내용이localhost인지 본 후 맞으면 host에 127.0.0.1을 넣어준다(local이라는 변수에 127.0.0.1이 들어있다.). 만약 아니면 .이 3개인지 본 후 그 내용을 호스트에 넣어주고 그 것도 아니면 에러를 체크하는 플래그를 1로 바꿔준 후 포문을 나간다. :다음 자리의 인덱스를 port_start에 저장하고 col_flag를 1로 바꿔준다.

/가 나올 때에는 그 전에 :이 나왔었는지 아니었는지 두가지로 나눠서 판별한다. /가 나오고 :가 나왔었으면 port_start가 /가 나온 인덱스와 같다면 :/이렇게 중간에 숫자가 들어있지 않게 된 것이므로 기본 포트인 10001을 넣어주고 그 것이 아니면 port_start의 인덱스 다음의 숫자를 port에 넣어준다. 그 후 pathname은 /다음의 스트링을 넣어주고 s_flag를 1로 바꿔주고 포문을 종료한다.

/가 나오고 :이 안 나온 경우 /앞의 내용이 localhost이면 127.0.0.1을 host에 넣어주고 아니면 .이 3개 나온지 판별한 후 그것을 host에 넣어준다. 그 경우도 아니면 에러이므로

err_flag를 1로 바꿔준 후 포문을 나간다. /가 나왔는데 :이 안 나왔으면 포트 번호를 따로 안 넣어준 것이므로 포트는 10001로 하고 /다음의 내용을 pathname에 넣어준다. s_flag를 1로 바꿔주고 포문을 종료한다. /나 :에 null을 넣은 것은 host, port, pathname에 넣기 위해서 인데 나중에 file이름을 따로 다시 자르기 위해서 /자리에는 다시 /을 넣어준다.

```
dot_count = 0;
col_flag = 0;
port_start = 0;
if(err_flag==1 || s_flag == 0){
    err_flag = 0;
    s_flag = 0;
    continue;
}
```

포문을 종료한 후 변수들을 초기화 해주고 err_flag가 1인 경우나 /가 안 쓰인 경우에는 인풋 에러이므로 그 인풋을 무시하기 위해서 continue해준다.

```
if ((cfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    continue;
}
if ((h = gethostbyname(host)) == NULL) {
    continue;
}
bzero((char *)&saddr, sizeof(saddr));
saddr.sin_family = AF_INET;
bcopy((char *)h->h_addr, (char *)&saddr.sin_addr.s_addr, h->h_length);
saddr.sin_port = htons(port);
if (connect(cfd,(struct sockaddr *)&saddr,sizeof(saddr)) < 0) {
    write(STDOUT_FILENO,"E_CONN, ",8);
    write(STDOUT_FILENO,pathname,strlen(pathname));
    continue;
}
```

그 후 인풋에서 자른 호스트와 포트로 소켓을 열고 서버와 연결을 한다. connect에서 에러가 있을 경우에는 E_CONN, pathname을 출력해준다. 다른 경우에 에러가 있으면 그냥 다시 인풋을 받는다.

```
write(cfd, pathname,MAXLINE);

for(i=input_len-2;i>=0;i--){
    if(buf[i]=='/'){
        file = buf+i+1;
        break;
    }
}
```

소켓을 통해 서버로 ₩n이 포함된 pathname을 보낸다. 그리고 pathname에 포함 된 file이름으로 file을 만들어야 하기 때문에 pathname에서 file이름을 따로 뽑아낸다.

```
read(cfd,result,4);
if(result[0]==0){
    write(STDOUT_FILENO,"OK, ",4);
    write(STDOUT_FILENO,pathname,strlen(pathname));
}else if(result[0]==1){
    write(STDOUT_FILENO,"E_FILE, ",8);
    write(STDOUT_FILENO,pathname,strlen(pathname));
    continue;
}else if(result[0]==2){
    write(STDOUT_FILENO,"E_PERM, ",8);
    write(STDOUT_FILENO,pathname,strlen(pathname));
    continue;
}
```

서버에서 파일을 열고 result code를 보내오면 그것을 받는다. 그 값에 따라서 정상적일 경우 OK, pathname을 출력하고 다음 코드를 진행하고 파일이 없는 경우 E_FILE, pathname을 출력하고 다시 인풋을 받고 파일에 대한 권한이 없는 경우 E_PERM, pathname을 출력하고 다시 인풋을 받는다.

```
file[strlen(file)-1] = '\0';
fd = open(file, O_WRONLY | O_TRUNC | O_CREAT, 0666);

read(cfd,&file_len,4);
while((n = read(cfd,buf2,file_len))>0){
    write(fd, buf2,n);
    read(cfd,&file_len,4);
}
```

File의 끝에 ₩n이 들어간 부분에 null을 넣어주고 그 파일에 쓰기 위해 열어준다. 서버에서 보내온 읽은 길이와 내용을 받아서 file에 써준다. 그 후에 파일과 소켓을 닫아주고 다시 인풋을 받는 과정을 반복한다.

4. 실행화면

1.2.3.4

1.2.3.4:10001

/data.txt

:10001/data.txt → 이 4가지 인풋은 전부 무시한다.

localhost:1234/awoiefj → 포트 번호가 다르므로 connect가 안 된다.

localhost/wjeofi → wjeofi라는 이름의 파일을 찾을 수 없다.

localhost/Server.c → Server.c라는 이름의 파일을 찾아서 받아온다.

exit → 종료