

시스템 프로그래밍 실습

FTP 3-2

Class : D
Professor : 최상호 교수님
Student ID : 2020202090
Name : 최민석

Introduction

이번 Assignment3-2에서는 서버와 클라이언트 간 데이터가 전달될 때, 명령어를 전달하는 컨트롤 커넥션과 데이터를 전송하는 데이터 커넥션을 분리한 형태의 서버-클라이언트 구조를 구현해볼 것이다. 컨트롤 커넥션에서는 명령어와 ACK 신호를 주고받으며 데이터 커넥션은 서버가 클라이언트에 접속해 처리 결과를 전송하고 바로 close 한다.

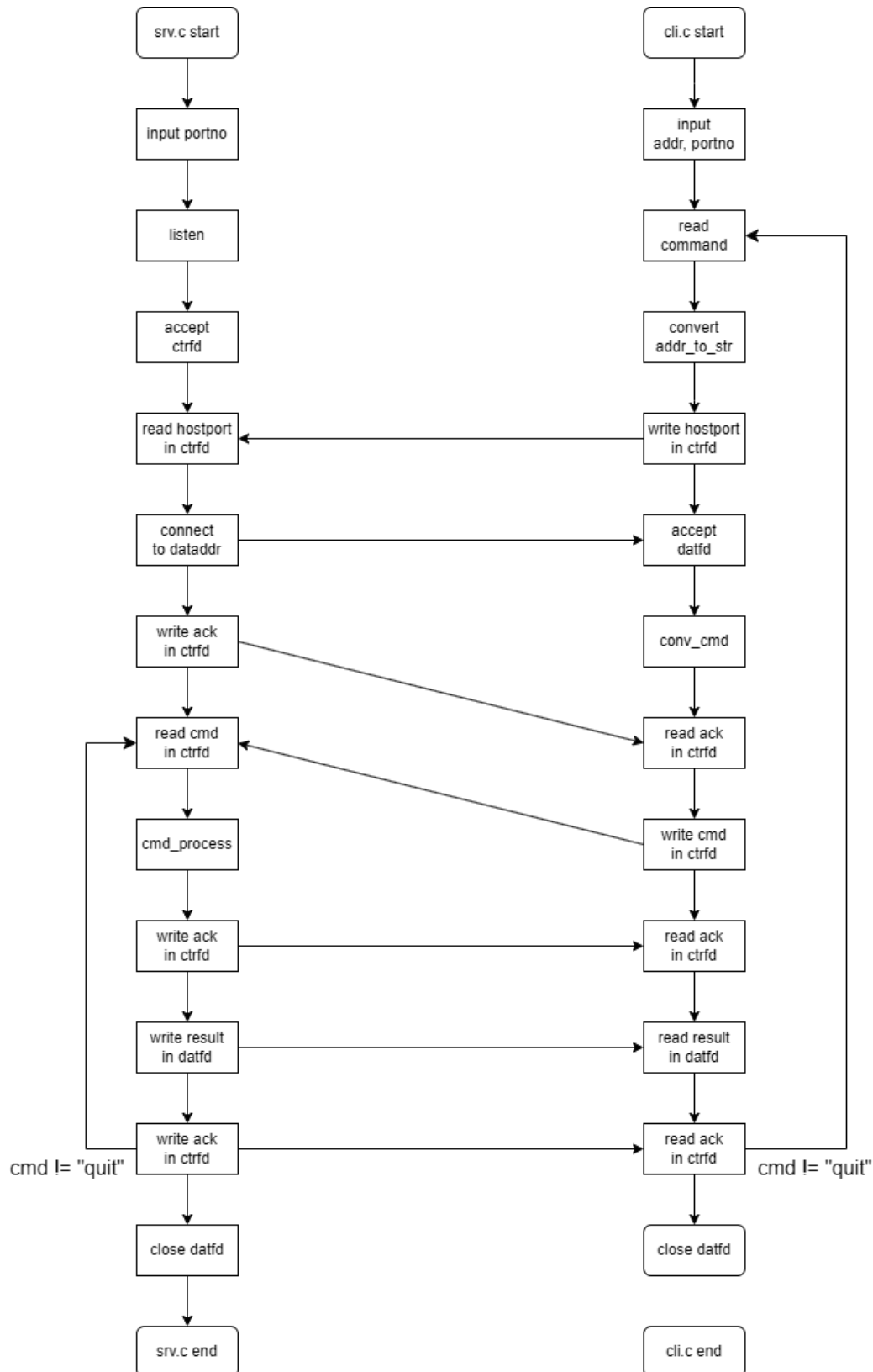
서버에 클라이언트가 접속하면 클라이언트는 자신의 IP 주소 및 10001~30000 사이의 임의의 포트 번호를 생성하고 이를 주어진 스키텔론 코드의 `convert_addr_to_str` 함수를 이용해 concatenation 한다. 이 때, IP 주소 32 비트와 포트 번호 16 비트는 각 8 비트마다 끊어서 콤마로 구분하는 comma-decimal 형태로 변환한다.

명령어를 입력 받으면 컨트롤 커넥션으로 명령어를 보내기 전 `convert_addr_to_str` 함수로 생성한 클라이언트의 IP 주소와 포트 정보를 전송하고 서버에서 이를 받아 클라이언트로 접속에 성공했다면 포트 명령어 성공 ACK 신호를 보낸다. 클라이언트에서 ACK 신호를 받으면 명령어를 전송하고 서버에서 받아 처리한다.

명령어 처리에 성공했으면 ACK 신호를 보내고 클라이언트로 명령어 처리 결과를 데이터 커넥션으로 전송한다. 이후 데이터 커넥션은 close 한다.

위처럼 컨트롤 커넥션과 데이터 커넥션을 분리해서 서버와 클라이언트가 FTP 명령어를 처리할 수 있는 형태의 시스템을 구현하고 테스트하는 것이 이번 Assignment3-2의 목표이다.

Flow chart



컨트롤 커넥션으로 명령어와 ack 를, 데이터 커넥션으로 명령어의 처리 결과를 전송한다.

Pseudo code

cli.c

define ctrfd, datfd, tempfd

define string cmd, rcv

set ctraddr, dataddr, tempaddr

connect(ctrfd, ctraddr)

loop {

 read(stdin, cmd)

 cmd = conv_cmd(cmd)

 define string hostport = convert_addr_to_str(client.addr)

 write(ctrfd, hostport)

 read(ctrfd, ack)

 bind(datfd, dataddr)

 listen(datfd)

 tempfd = accept(datfd)

 write(ctrfd, cmd)

 read(ctrfd, ack)

 read(tempfd, rcv)

 print(rcv)

 read(ctrfd, ack)

 close(datfd)

}

functon convert_addr_to_str(addr, &port) {

 define string addr

 define port = randint(10001, 30000)

 define porthi = port/256

 define portlo = port%256

 sscanf(addr, "%u.%u.%u.%u", temp[0], temp[1], temp[2], temp[3])

 snprintf(addr, "%u, %u, %u, %u, %u, %u", temp[0], temp[1], temp[2], temp[3],
 porthi, portlo)

 return addr

srv.c

```
define ctrfd, clifd, datfd
```

```
define buf, cmd, rst
```

```
set ctraddr, cliaddr, dataddr
```

```
bind(ctrfd, ctraddr)
```

```
listen(ctrfd)
```

```
loop {
```

```
    clifd = accept(ctrfd)
```

```
    loop {
```

```
        read(clifd, hostport)
```

```
        host_ip = convert_str_to_addr(hostport, &port)
```

```
        write(clifd, ack)
```

```
        connect(datfd, dataddr)
```

```
        read(clifd, cmd)
```

```
        print(cmd)
```

```
        cmd_process(cmd, rst)
```

```
        write(clifd, ack)
```

```
        write(datfd, rst)
```

```
        write(clifd, ack)
```

```
    }
```

```
}
```

```
function convert_str_to_addr(hostport, &port) {
```

```
    define addr
```

```
    define ip_bytes[4]
```

```
    define porthi, portlo
```

```
    sscanf(hostport, "%u, %u, %u, %u, %u, %u, ip_bytes, porthi, portlo)
```

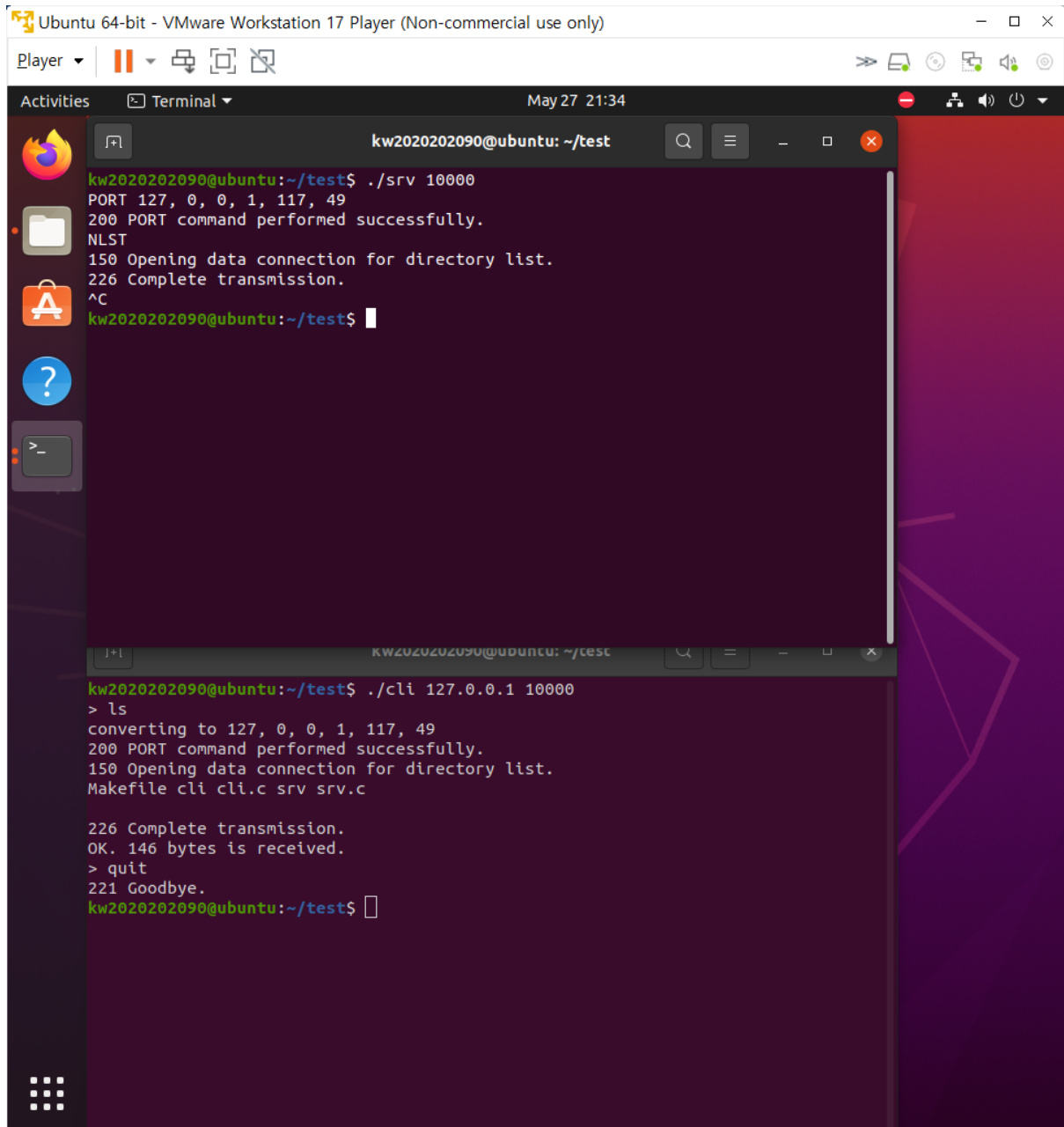
```
    addr = (ip_bytes[0] << 24) | (ip_bytes[1] << 16) | (ip_bytes[2] << 8) | (ip_bytes[3])
```

```
    port = (porthi << 8) | portlo
```

```
    return addr
```

```
}
```

결과화면



```
kw2020202090@ubuntu: ~/test$ ./srv 10000
PORT 127, 0, 0, 1, 117, 49
200 PORT command performed successfully.
NLST
150 Opening data connection for directory list.
226 Complete transmission.
^C
kw2020202090@ubuntu:~/test$

kw2020202090@ubuntu:~/test$ ./cli 127.0.0.1 10000
> ls
converting to 127, 0, 0, 1, 117, 49
200 PORT command performed successfully.
150 Opening data connection for directory list.
Makefile cli cli.c srv srv.c

226 Complete transmission.
OK. 146 bytes is received.
> quit
221 Goodbye.
kw2020202090@ubuntu:~/test$
```

2024-1_SPLab_FTP_Assginment3_2_v2 의 실행 예제를 참고하였다. 클라이언트에서 명령어를 전송하면 서버로 클라이언트의 주소와 포트번호를 전송하고, 서버에서 해당 정보로 클라이언트에 연결한 후 명령어를 받아 처리하고 데이터 커넥션으로 결과를 전송한다. 시퀀스의 단계 별로 ACK 신호를 생성해 주고받는다.

quit 를 입력하면 클라이언트에서 명령어를 바로 전송하고 ACK 를 받는 quick quit sequence 를 진행하여 실행 예제와 맞추었다.

고찰

hostport 를 생성할 때, IP 주소와 포트를 어떻게 concatenate 하는 이유를 잘 알지 못하지만 일단 과제 제안서에 있는 대로 구현하였다. 원래는 8 비트마다 콤마로 구분한 형태의 comma decimal 형태로 전송하지만, 차라리 IP 주소 자체와 포트를 콜론으로 구분하여 전송했다면 parse 하기 좀 더 편한 형태가 되었을 것이라고 생각한다.

컨트롤 커넥션으로는 명령어와 ACK 신호를 주고받고, 데이터 커넥션으로는 오직 명령어의 처리 결과를 전송받고, 한 번 사용한 다음 close 한다. 서버에서의 read write 개수와 클라이언트의 read write 개수 자체는 같지만, 읽는 순서에 따라 한 번씩 밀린다면 서로 상호작용할 수 없는 상태가 되므로 이에 주의해서 시퀀스를 고려하며 코드를 작성하였다.

사용자에게서 명령어를 입력 받을 때, read 와 printf 를 혼용하여 사용할 경우 순서가 꼬여서 read 앞에 있는 printf 가 나중에 호출될 수도 있다는 것을 알게 되었다. 호출 속도에 따른 차이로 생각하고, 똑같이 시스템 콜인 write 를 이용해 터미널에 출력하였더니 순서가 뒤바뀌는 문제를 해결할 수 있었다.

컨트롤 커넥션과 데이터 커넥션을 분리하여 관리하도록 설계하였는데, 이를 조율하기 위해 컨트롤 커넥션으로 계속 ACK 와 같은 handshake 신호를 주고받는다. 데이터 커넥션으로 데이터를 받을 땐 ACK 이후에만 전송, 수신하면 되므로 FTP 시퀀스를 고려할 때 도움이 될 수도 있다고 생각한다.

다음 Assignment3-3 에선 지금까지 구현한 모든 것들을 종합하고 로그 작성까지 할 수 있도록 코드를 추가하면 FTP 서버를 구현할 수 있을 것이라고 생각한다.

Reference