
Cloud-Service

using Unix Network Programming

최종보고서

2016707079 하상천

목 차

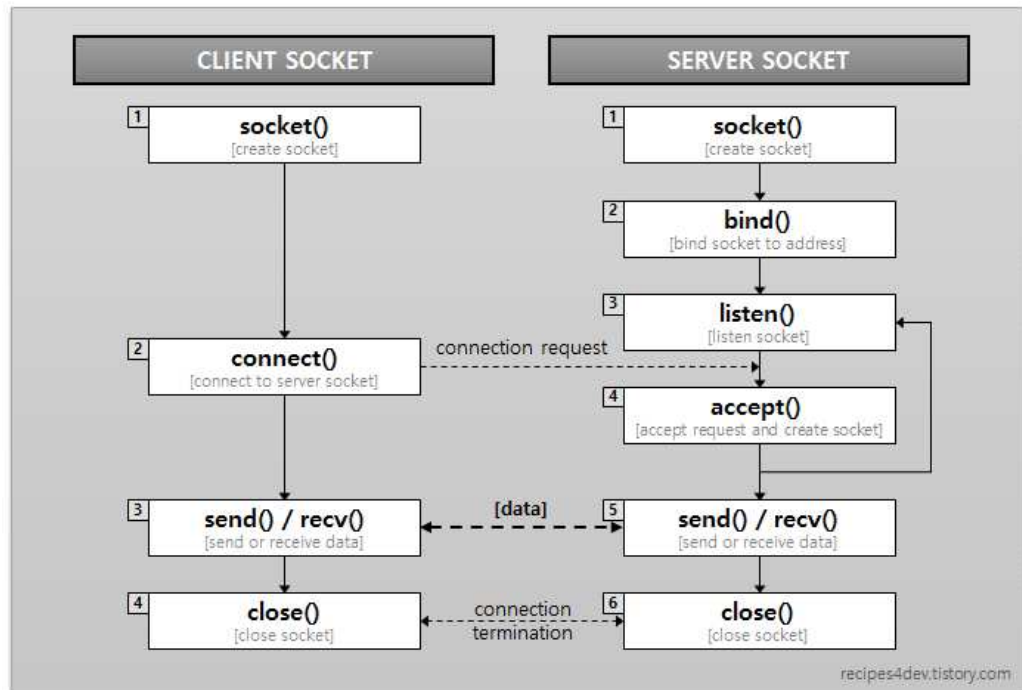
1. 프로젝트 선정 및 목표
 2. 관련이론 및 제반사항
 3. 설계 및 구현 사항
 4. 고찰
 5. 참고문헌
-

1. 프로젝트 선정 및 목표

이번 프로젝트에서는 수업시간에 배운 TCP/IP 소켓 통신을 직접 구현해 보고 싶어서 Unix Network Programming을 이용한 Cloud-Service를 만들게 되었다. 로그인과 회원가입 과정을 통해 각각의 디렉토리에 파일을 upload, download 하고 여러 클라이언트가 동시에 서버와 통신하여 Cloud-Service를 이용 하는 것을 목표로 하였다.

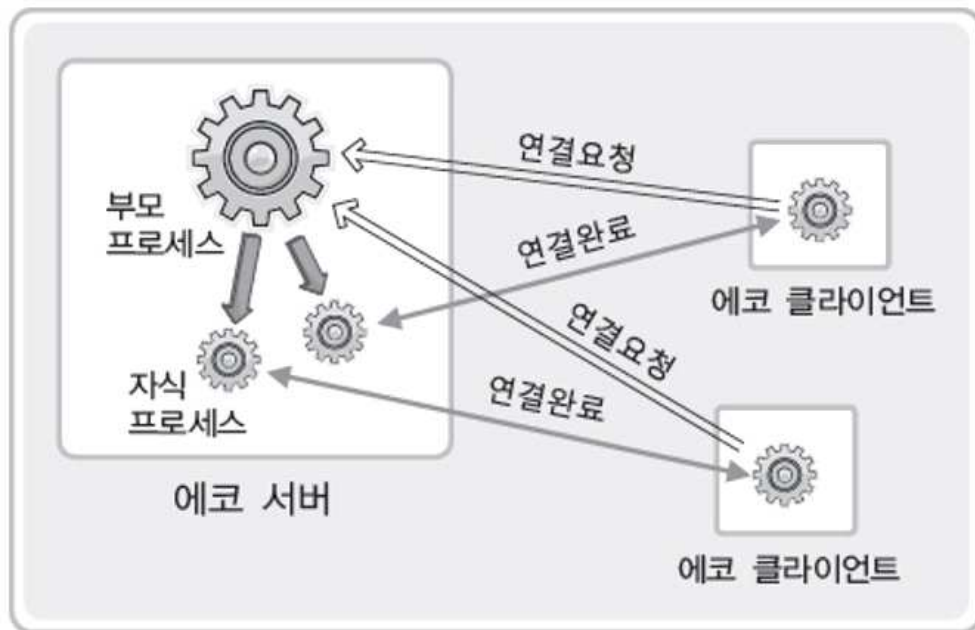
2. 관련이론 및 제반사항

(1) TCP/IP Socket Programming



클라이언트 소켓은 소켓을 생성한 다음, 서버 측에 연결(connect)을 요청한다. 그리고 서버 소켓에서 연결이 받아들여지면 데이터를 송수신(send/recv)하고, 모든 처리가 완료되면 소켓을 닫는다(close). 서버 소켓도 소켓을 생성하고, 서버가 사용할 IP 주소를 포트 번호를 생성한 소켓에 결합(bind)시킨다. 그런 다음 클라이언트로부터 연결 요청이 수신되는지 주시(listen)하고, 요청이 수신되면 요청을 받아들여(accept) 데이터 통신을 위한 소켓을 생성한다. 새로운 소켓을 통해 연결이 수립(ESTABLISHED)되면, 클라이언트와 마찬가지로 데이터를 송수신(send/recv)할 수 있고, 마지막으로 데이터 송수신이 완료되면 소켓을 닫는다(close).

(2) 멀티 프로세스



한 번에 하나의 클라이언트에게만 서비스를 제공하는 것이 아니라, 여러 클라이언트에게 서비스를 제공하기 위해 멀티프로세스 방식을 사용한다.

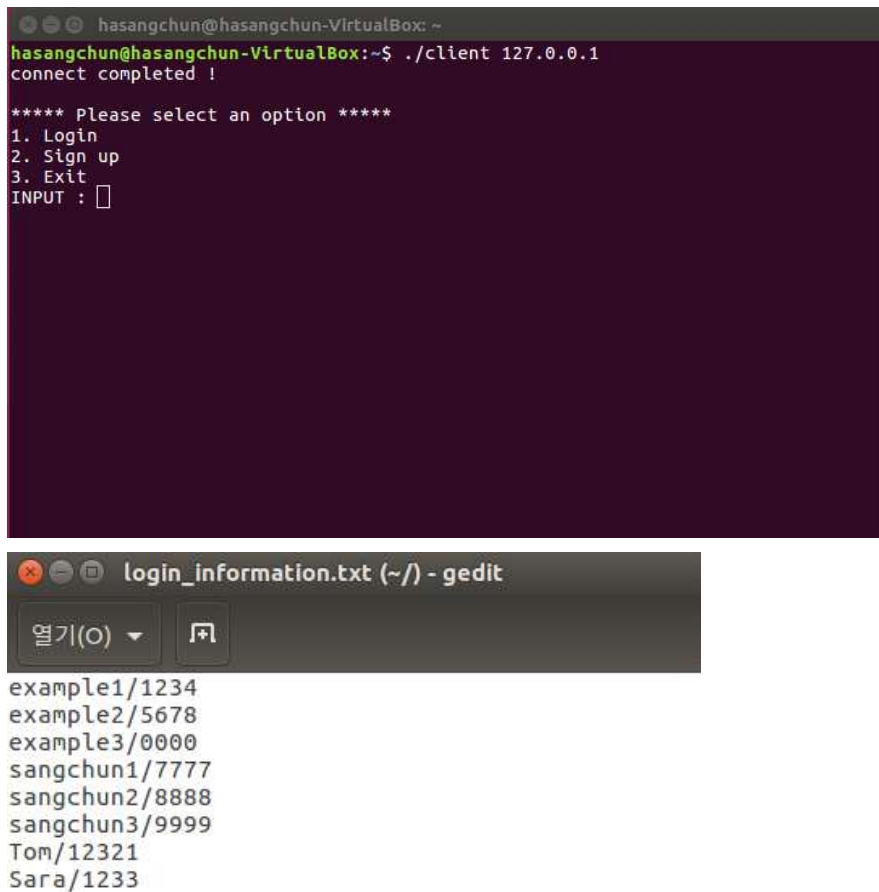
멀티프로세스를 구현하기 위한 방법 중 가장 구현하기 쉬운

Thread-per-Connection 방식을 사용한다.

단순히 어떠한 클라이언트에게서 연결 요청이 오게 되면 `fork()` 함수를 이용하여 자식 프로세스를 생성한 뒤, 해당 프로세스와 클라이언트를 Connect 해주는 방식이다.

3. 설계 및 구현 사항

(1) 회원가입 & 로그인



```
hasangchun@hasangchun-VirtualBox: ~  
hasangchun@hasangchun-VirtualBox:~$ ./client 127.0.0.1  
connect completed !  
  
***** Please select an option *****  
1. Login  
2. Sign up  
3. Exit  
INPUT : 
```



```
login_information.txt (~/) - gedit  
열기(O)   
example1/1234  
example2/5678  
example3/0000  
sangchun1/7777  
sangchun2/8888  
sangchun3/9999  
Tom/12321  
Sara/1233
```

클라이언트가 Sign up을 선택하여 회원가입을 진행하면, 그 정보들을 서버 측에 저장된 login_information.txt 파일에 ID/Password 형식으로 저장시킨다. 클라이언트가 로그인을 시도할 때는 /을 기준으로 나눠서 로그인 성공 여부를 판단한다.

(2) Upload & Download



유저 아이디 별로 서버 측에서 폴더를 생성하고, 해당 유저 ID가 업로드시 해당 폴더 내에서 파일을 관리한다. 그리고 텍스트, 이미지, 오디오, 영상 등 모든 파일을 업로드, 다운로드 할 수 있도록 구현했다.

(3) 서버 로그

```
hasangchun@hasangchun-VirtualBox: ~  
hasangchun@hasangchun-VirtualBox:~$ ./server  
Waiting the clients...  
Client IP : 127.0.0.1  
process 2045 : allocated !!!!!  
  
Client IP : 127.0.0.1  
process 2049 : allocated !!!!!  
  
process 2049 : sign up is successful !  
process 2049 : login success !  
process 2045 : login fail !  
process 2045 : login success !  
sangchun1 : Directory already existed !  
sangchun1 : Uploading the file ...  
ID : sangchun1  
File name : test1.jpg  
Uploading is completed !  
  
hasangchun@hasangchun-VirtualBox: ~  
Sara : Directory is created !  
Sara : Uploading the file ...  
ID : Sara  
File name : 무민.jpg  
Uploading is completed !  
Sara : Uploading the file ...  
ID : Sara  
File name : Son.mp4  
Uploading is completed !  
sangchun1 : Downloading the file ...  
ID : sangchun1  
File name : Krong.jpeg  
Downloading is completed !  
process 2045 : terminated !!!!!  
process 2049 : terminated !!!!!
```

서버 측에서 어떤 프로세스가 사용 중이고, 어떤 프로세스에 어떤 ID가 연결이 되었고, 현재 어느 서비스를 이용하고 있는 중인지를 한눈에 파악하기 위해 서버 로그 기능을 구현했다.

4. 고찰

이번 프로젝트는 UNIX Network Programming을 사용하여 Cloud-Service를 구현하는 것이었다. 프로젝트를 진행하면서 가상머신과 리눅스 환경을 처음 경험해보았는데 윈도우와 달라서 적응하는데 시간이 조금 걸렸다. 특히, 리눅스는 User interface가 CLI(Command-Line Interface)인 것이다. 처음에 클라이언트가 회원가입을 하면 서버 측에서 정보를 받아 login_information.txt 파일에 ID/Password 형식으로 저장한다. 그리고 클라이언트가 로그인을 시도하면 서버 측에서 login_information.txt 파일을 열어 ID와 Password를 /으로 나눠서 로그인 성공 여부를 판단한다. 또한 유저 아이디 별로 서버 측에서 폴더를 생성하고, 해당 유저 아이디가 업로드시 해당 폴더 내에서 파일을 관리하도록 구현했다. 그리고 텍스트, 이미지, 오디오, 영상 등 모든 파일을 업로드, 다운로드 할 수 있도록 구현했다. 여러 클라이언트가 동시에 접속하여 upload, download 할 수 있도록 구현했기 때문에 어떤 프로세스가 할당 되었고, 어떤 ID가 어떤 서비스를 이용하고 있는 중인지를 한눈에 파악하고 싶어서 서버 로그를 만들었다. 작년 겨울방학에 자바와 데이터베이스를 연동하여 클라이언트가 음료를 주문하면 서버 측에서 정보를 받아 음료의 레시피를 알려주는 프로젝트를 진행했었다. 그때 소켓통신을 해보았지만, c언어로 해보는 것은 처음이었고 멀티 프로세스를 이용한 것도 처음이라 조금 막막하였지만 인터넷 검색과 책으로 공부를 하면서 서버측과 클라이언트측 UI(User Interface)를 적고 하나하나 상황을 따져가면서 코딩을 하다보니까 최종적으로 구현이 된 것 같다. 다음에 소켓통신 프로젝트를 진행할 때에도, 서버측과 클라이언트측을 따로 적어서 하나하나 확인하면서 코딩을 하면 좋을 것 같다.

5. 참고문헌

Unix Network Programming by W.Richard Stevens

<https://recipes4dev.tistory.com/153>

<http://blog.naver.com/PostView.nhn?blogId=s2kiess&logNo=220157649224>
