컴퓨터네트워크

실습 #09 문제 및 보고서

이름	황명원
학번	20185309
소속 학과/대학	콘텐츠 it 전공/정보과학대학
분반	01 (담당교수: 박찬영)

<주의사항>

- 개별 과제 입니다. (팀으로 진행하는 과제가 아니며, 모든 학생이 보고서를 제출해야 함)
- 각각의 문제 바로 아래에 답을 작성 후 제출해 주세요.
    - 소스코드/스크립트 등을 작성한 경우, 해당 파일의 이름도 적어주세요.
- SmartLEAD 제출 데드라인:
    - 다음 실습시간 전날 (5/11) 23:55 까지 (2 주간 진행하는 과제 입니다)
    - 데드라인을 지나서 제출하면 0 점
    - 주말/휴일/학교행사 등으로 인한 데드라인 연장 없음
    - 부정행위 적발 시, 원본(보여준 사람)과 복사본(베낀 사람) 모두 0 점 처리함
- SmartLEAD 에 아래의 파일을 제출해 주세요
    - 보고서(PDF 파일로 변환 후 제출을 권장하나, WORD 로 제출해도 됨)
    - 보고서 파일명에 이름과 학번을 입력해 주세요.
    - 소스코드, 스크립트, Makefile 등을 작성해야 하는 경우, 모든 파일 제출(또는 본 문서에 소스코드 화면 캡처해서 붙여넣기)

<개요>

이번 과제는 소켓 프로그래밍을 통한 통신 프로그램을 구현하는 내용으로 구성되어 있습니다.

** 주의: 전체 소스코드를 압축하여 첨부파일로 제출하세요.

<실습 과제>

[Q 1] AF_UNIX 도메인 STREAM 소켓 프로그래밍 [배점: 20]

AF_UNIX/SOCK_STREAM 타입의 소켓을 사용하는 서버-클라이언트 프로그램을 작성하는 문제입니다. 소켓 파일 경로는 ./sock_addr 를 사용하세요.

간단한 단방향 메시지 전송 프로그램을 작성하는 구현하는 문제입니다. 서버-클라이언트간 연결이 설정되면(=클라이언트의 connect 요청에 서버가 accept 로 반응하면)…

- 클라이언트 프로그램은 사용자 터미널로부터 전달받은 문자열을 서버로 전송하고
- 서버 프로그램은 클라이언트로부터 받은 메시지를 터미널에 출력하는

과정을 반복합니다. 클라이언트에서 \quit 이라는 메지시를 입력하면 클라이언트와 서버 프로그램 모두 종료합니다. 주의: 서버 프로그램을 먼저 실행하고, 다음으로 클라이언트 프로그램을 실행하세요. 클라이언트에서 사용자가 \quit 메시지를 입력하면, 서버로 해당 메시지를 전달하고 난 다음에 클라이언트를 종료하세요.

- (문제 1) 서버와 클라이언트 프로그램을 서로 다른 터미널에서 구동하고, 그 상태에서 세 번째 터미널을 실행하세요. 서버-클라이언트 프로그램이 저장된 디렉토리로 이동한 뒤 $ls 명령을 입력하세요. 터미널 결과를 캡처하여 아래에 첨부하세요. 주의: $ls 명령의 결과에 sock_addr 이라는 파일이 나타나야 합니다.
- (문제 2) 서버-클라이언트 프로그램을 구동하고, 클라이언트는 1) hello world 메시지를 보내고, 다음으로 2) nice to meet you 메시지를 보내고, 마지막으로 3) \quit 메시지를 전송합니다. 서버측 터미널 출력 결과를 캡처하여 아래에 첨부하세요.

답변 1)

```
┌──(kali㉿kali)-[~/week-9]
└─$ ls
client  client.c  server  server.c  sock_addr

┌──(kali㉿kali)-[~/week-9]
└─$ nano client.c

┌──(kali㉿kali)-[~/week-9]
└─$ cat sock_addr
cat: sock_addr: No such device or address
```

답변 2)

```
┌──(kali㉿kali)-[~/week-9]
└─$ ./server
Client: hello world
Client: nice to meet you
Client: \quit
```

[Q 2] AF_INET 도메인 STREAM 소켓 프로그래밍 : 동기형 1 대 1 채팅 [배점: 40]

AF_INET/SOCK_STREAM 타입의 서버-클라이언트 소켓 프로그램을 작성하는 문제입니다. 이번에는 서버와 클라이언트가 1:1 로 순서에 맞춰서 채팅하는 프로그램을 작성하세요.
동기화된 양방향 메시지 전송 프로그램을 작성하는 구현하는 문제입니다. 서버-클라이언트간 연결이 설정되면(=클라이언트의 accept 요청에 서버가 accept 로 반응하면) 아래의 동작을 순서대로 반복합니다.

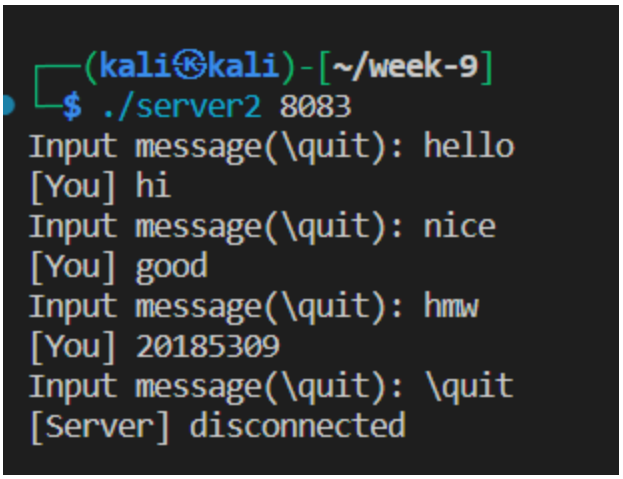
1. [클라이언트] 사용자로부터 전달받은 문자열을 서버로 전송 (사용자는 터미널에서 문자열 입력)
2. [서버] 클라이언트로부터 받은 메시지를 터미널에 출력
3. [서버] 사용자 터미널로부터 전달받은 문자열을 클라이언트로 전송
4. [클라이언트] 프로그램은 서버로부터 받은 메시지를 터미널에 출력
5. 위의 1~4 과정을 반복 (서버와 클라이언트는 순서에 맞게 채팅 메시지를 입력함)

클라이언트 또는 서버에서 \quit 이라는 메지시를 입력하면 클라이언트와 서버 프로그램 모두 종료합니다. 상대방으로부터 전달받은 메시지를 터미널에 출력할 때, [You] 라는 문자열을 먼저

출력하고, 다음으로 상대방의 메시지를 출력하세요. 예를 들어, 클라이언트가 `hello` 라는 메시지를 전송하면, 서버는 `[You] hello` 라고 출력해야 합니다.
주의: 서버 프로그램을 먼저 실행하고, 다음으로 클라이언트 프로그램을 실행하세요. 클라이언트 >> 서버 >> 클라이언트 >> 서버 … 순으로 메시지를 전송해야 합니다 (= 동기형 1:1 채팅).

문제) 클라이언트와 서버 각각 3 번씩 메시지를 입력하도록 하고, 서버와 클라이언트의 터미널을 모두 캡처해서 아래에 첨부하세요.

답변 (서버의 터미널 화면 캡처)



답변 (클라이언트의 터미널 화면 캡처)

```
┌──(kali㉿kali)-[~/week-9]
└─$ ./client2 127.0.0.1 8083
Connected to server...
Input message (\quit): hi
[You] hello
Input message (\quit): good
[You] nice
Input message (\quit): 20185309
[You] hmw
Input message (\quit): \quit
[You] \quit
[Client] disconnected
```

[Q 3] AF_INET 도메인 STREAM 소켓 : 멀티 서비스 [배점: 40]

이번에는 서버 프로그램과 클라이언트 프로그램을 서로 다른 디렉토리에 저장해야 합니다. 디렉토리를 아래와 같이 구성하세요.

- MultiService/server
- MultiService/client

'server' 디렉토리 아래에 server.c 소스코드를 생성하여 코딩하고, SmartLEAD 에 첨부된 Book.txt, HallymUniv.jpg 파일을 서버 쪽 디렉토리에 저장하세요. 'client' 디렉토리 아래에 client.c 소스코드를 생성하여 코딩하세요.

클라이언트가 서버에 접속하면, 다음과 같은 메시지를 서버로부터 전달받습니다.

```
[Service List]
1. Get Current Time
2. Download File
3. Echo Server
Enter:
```

클라이언트는 메시지를 터미널에 출력하고, 사용자 입력을 기다립니다.

- 사용자가 1 을 입력하면
  - \service 1 이라는 메시지가 서버로 전달됩니다.
  - 서버는 \service 1 이라는 메시지를 받으면, 현재 시간을 문자열 형태로 클라이언트에게 전달합니다 (현재 시간을 문자열 형태로 얻는 코드는 첨부된 get_localtime.c 파일을 참고하세요). 클라이언트는 전달받은 메시지를 터미널에 출력합니다.

- 서버는 다음으로 [Service List] 메뉴 메시지를 클라이언트에게 전달합니다. 클라이언트는 메뉴 메시지를 터미널에 출력하고 사용자 입력을 기다립니다.

■ 사용자가 2 를 입력하면

- \service 2 라는 메시지가 서버로 전달됩니다.
- 서버는 \service 2 라는 메시지를 받으면, 아래의 메시지를 클라이언트에게 전달합니다.

```
[Available File List]
1. Book.txt
2. HallymUniv.jpg
3. Go back
Enter:
```

- 클라이언트는 메시지를 터미널에 출력하고 사용자 입력을 기다립니다.
- 사용자가
  - ◆ 1 또는 2 를 입력하여 서버에 전달하면, 클라이언트는 해당 파일을 서버로부터 다운받습니다. 파일 전송이 완료되면 서버는 [Service List] 메뉴 메시지를 클라이언트에게 전달합니다. 클라이언트는 메뉴 메시지를 터미널에 출력하고 사용자 입력을 기다립니다.
  - ◆ 3 을 입력하여 서버에 전달하면, 서버는 [Service List] 메뉴 메시지를 클라이언트에게 전달합니다. 클라이언트는 메뉴 메시지를 터미널에 출력하고 사용자 입력을 기다립니다.

■ 사용자가 3 을 입력하면

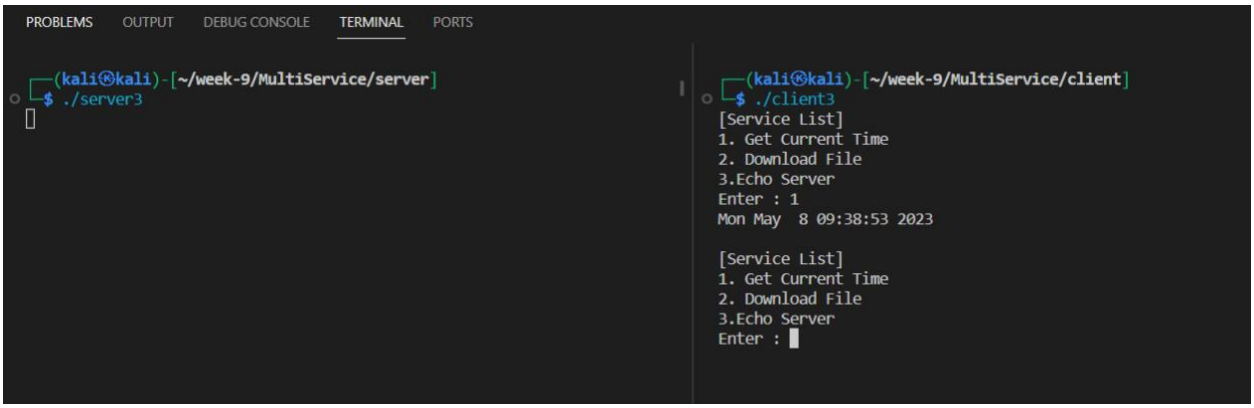
- 서버는 ECHO SERVER 로 동작합니다. 즉, 클라이언트가 메시지를 입력하면, 해당 메시지는 서버로 전달되고, 동일한 메시지가 다시 클라이언트로 전달됩니다. 클라이언트는 서버로부터 수신한 메시지는 [You] … 이런 식으로 터미널에 출력합니다.
  - ◆ 사용자가 \quit 이라고 입력하면 ECHO SERVER 는 중지되고, 서버는 [Service List] 메뉴 메시지를 클라이언트에게 전달합니다. 클라이언트는 메뉴 메시지를 터미널에 출력하고 사용자 입력을 기다립니다.

■ (문제 1) 클라이언트가 1 번 서비스를 사용하게 하고, 터미널 출력 화면을 캡처하여 아래에 첨부하세요.

- (문제 2) 클라이언트가 2 번 서비스를 사용하게 하고, 텍스트 파일과 이미지 파일을 모두 다운받은 후, 클라이언트 터미널 출력 화면을 캡처하여 아래에 첨부하세요. 클라이언트 터미널에서, 프로그램 구동 전 `$ls` 결과, 그리고 구동 후 `$ls` 결과를 캡처하여 아래에 첨부하세요. 프로그램 구동 후에는 `$ls` 명령 입력 시, 다운받은 파일이 조회되어야 합니다.
- (문제 3) 클라이언트가 3 번 서비스를 사용하게 하고, 클라이언트가 3 번 메시지를 전송하도록 하세요. 터미널 출력 화면을 캡처하여 아래에 첨부하세요.

[참고] 터미널 화면 캡처 시, 출력된 문자열이 너무 많다면, 최근의 출력문만 캡처해도 괜찮습니다.

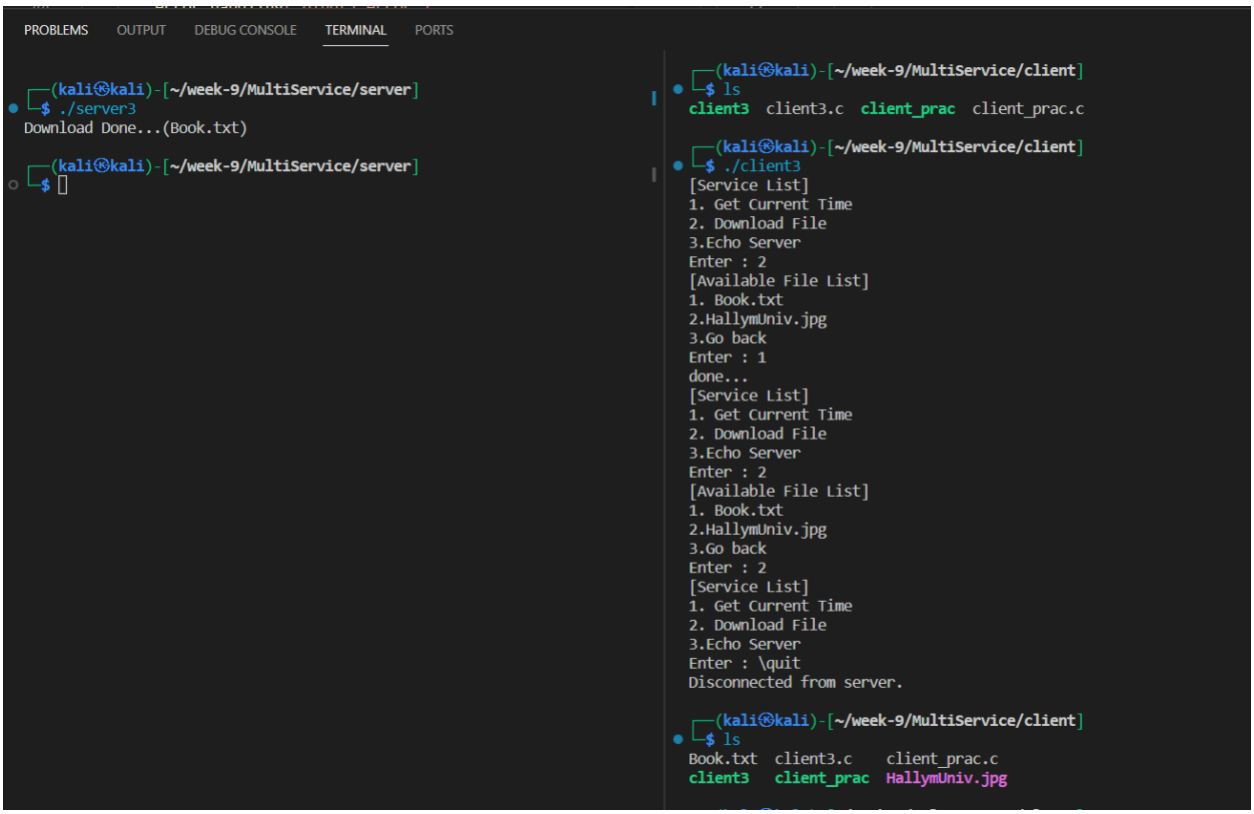
답변 1)



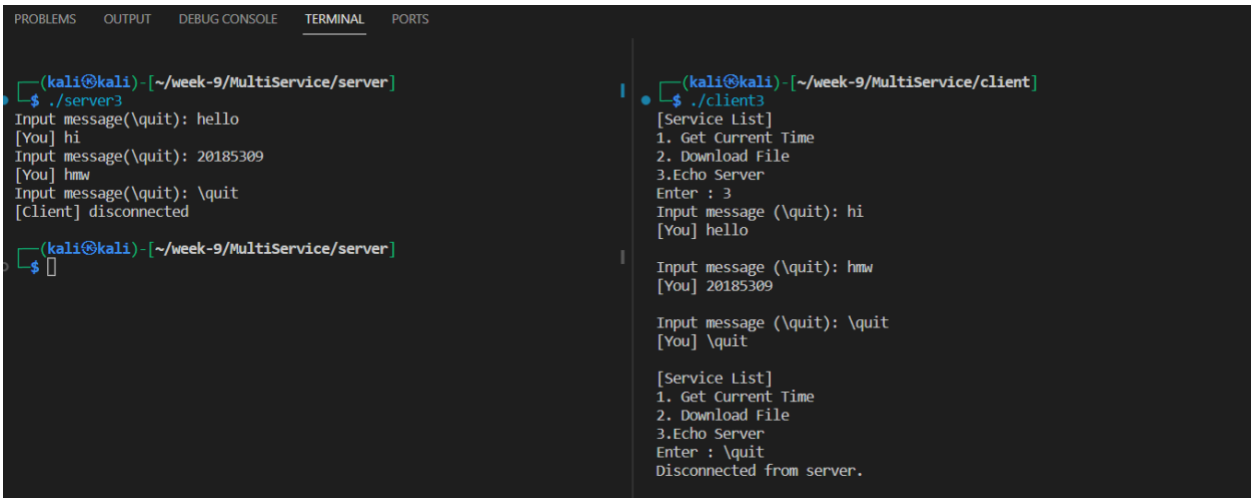
답변 2)

다운받기 전:

다운 받은 후:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
(kali㉿kali)-[~/week-9/MultiService/server]
$ ./server3
Download Done...(Book.txt)
(kali㉿kali)-[~/week-9/MultiService/server]
$
(kali㉿kali)-[~/week-9/MultiService/client]
$ ls
client3 client3.c client_prac client_prac.c
(kali㉿kali)-[~/week-9/MultiService/client]
$ ./client3
[Service List]
1. Get Current Time
2. Download File
3.Echo Server
Enter : 2
[Available File List]
1. Book.txt
2.HallymUniv.jpg
3.Go back
Enter : 1
done...
[Service List]
1. Get Current Time
2. Download File
3.Echo Server
Enter : 2
[Available File List]
1. Book.txt
2.HallymUniv.jpg
3.Go back
Enter : 2
[Service List]
1. Get Current Time
2. Download File
3.Echo Server
Enter : \quit
Disconnected from server.
(kali㉿kali)-[~/week-9/MultiService/client]
$ ls
Book.txt client3.c client_prac.c
client3 client_prac HallymUniv.jpg

답변 3)


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
(kali㉿kali)-[~/week-9/MultiService/server]
$ ./server3
Input message(\quit): hello
[You] hi
Input message(\quit): 20185309
[You] hmw
Input message(\quit): \quit
[Client] disconnected
(kali㉿kali)-[~/week-9/MultiService/server]
$
(kali㉿kali)-[~/week-9/MultiService/client]
$ ./client3
[Service List]
1. Get Current Time
2. Download File
3.Echo Server
Enter : 3
Input message (\quit): hi
[You] hello
Input message (\quit): hmw
[You] 20185309
Input message (\quit): \quit
[You] \quit
[Service List]
1. Get Current Time
2. Download File
3.Echo Server
Enter : \quit
Disconnected from server.

끝! 수고하셨습니다 ☺