[예제 1] 반복 서버: Server (1/2)

C n1.c × C n2.c

hw12 > C n1.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define BUFFSIZE 4096
#define SERVERPORT 7799

int main(void) {
    int i, s_sock, c_sock;
    struct sockaddr_in server_addr, client_addr;
    socklen_t c_addr_size;
    char buf[BUFFSIZE] = {0};
    char hello[] = "Hello~I am Server!\n";

    s_sock = socket(AF_INET, SOCK_STREAM, 0);

    int option = 1;          // SO_REUSEADDR 의 옵션 값을 TRUE 로
    setsockopt(s_sock, SOL_SOCKET, SO_REUSEADDR, &option, sizeof(option));

    bzero(&server_addr, sizeof(server_addr));

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(SERVERPORT);
    server_addr.sin_addr.s_addr = inet_addr("10.0.0.249");

    if (bind(s_sock, (struct sockaddr *) &server_addr, sizeof(server_addr)) == -1) {
        perror("[S] Can't bind a socket");
        exit(1);
    }

    listen(s_sock,1);
    c_addr_size = sizeof(struct sockaddr);

```

```
ubuntu@41983:~/hw12$ ./n1
[S] Can't bind a socket: Address already in use
```

```
ubuntu@41983:~/hw12$ netstat -al | grep 7799
tcp        0      0 41983:7799              41983:38752             TIME_WAIT
tcp        0      0 41983:7799              41983:38756             TIME_WAIT
tcp        0      0 41983:38754             41983:7799              TIME_WAIT
```

- 프로그램 종료 후, 다시 실행시켰을 때, bind() 에서 “already in use” 오류가 나는 것을 무시하기 위해 사용 (참고)

[예제 1] 반복 서버: Server (2/2)

```
35      for(i=0; i<3; i++) {
36          printf("[S] waiting for a client..#%02d\n", i);
37          c_sock = accept(s_sock, (struct sockaddr *) &client_addr, &c_addr_size);
38          if (c_sock == -1) {
39              perror("[S] Can't accept a connection");
40              exit(1);
41          }
42
43          printf("[S] Connected: client IP addr=%s port=%d\n", inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));
44
45          //1. say hello to client
46          if (send(c_sock, hello, sizeof(hello)+1, 0) == -1) {
47              perror("[S] Can't send message");
48              exit(1);
49          }
50
51          printf("[S] I said Hello to Client!\n");
52
53          //2. recv msg from client
54          if (recv(c_sock, buf, BUFFSIZE, 0) == -1) {
55              perror("[S] Can't receive message");
56              exit(1);
57          }
58
59          printf("[S] Client says: %s\n", buf);
60
61          close(c_sock);
62      }
63      close(s_sock);
64
65      return 0;
66  }
```

[예제 1] 반복 서버: Client (1/2)

hw12 > C n2.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define BUFFSIZE 4096
#define SERVERPORT 7799

int main(void) {
    int c_sock;
    struct sockaddr_in server_addr, client_addr;
    socklen_t c_addr_size;
    char buf[BUFFSIZE] = {0};
    char hello[] = "Hi~I am Client!!\n";

    c_sock = socket(AF_INET, SOCK_STREAM, 0);

    bzero(&server_addr, sizeof(server_addr));

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(SERVERPORT);
    server_addr.sin_addr.s_addr = inet_addr("10.0.0.249");


    printf("[C] Connecting...\n");

    if (connect(c_sock, (struct sockaddr *) &server_addr, sizeof(server_addr)) == -1) {
        perror("[C] Can't connect to a Server");
        exit(1);
    }

    printf("[C] Connected!\n");
```

[예제 1] 반복 서버: Client (2/2)

hw12 > C n2.c

```c
    printf("[C] Connecting...\n");

    if (connect(c_sock, (struct sockaddr *) &server_addr, sizeof(server_addr)) == -1) {
        perror("[C] Can't connect to a Server");
        exit(1);
    }

    printf("[C] Connected!\n");

    //1. recv msg from server (maybe it's "hello")
    if (recv(c_sock, buf, BUFFSIZE, 0) == -1) {
        perror("[C] Can't receive message");
        exit(1);
    }

    printf("[C] Server says: %s\n", buf);

    //2. say hi to server

    if (send(c_sock, hello, sizeof(hello)+1, 0) == -1) {
        perror("[C] Can't send message");
        exit(1);
    }

    printf("[C] I said Hi to Server!!\n");

    //printf("[C] I am going to sleep...\n");
    //sleep(10);

    close(c_sock);

    return 0;
}
```

[예제 1] 정상 실행 예

```
ubuntu@41983:~/hw12$ ./n1
[S] waiting for a client..#00
[S] Connected: client IP addr=10.0.0.249 port=38970
[S] I said Hello to Client!
[S] Client says: Hi~I am Client!!

[S] waiting for a client..#01
[S] Connected: client IP addr=10.0.0.249 port=38972
[S] I said Hello to Client!
[S] Client says: Hi~I am Client!!

[S] waiting for a client..#02
[S] Connected: client IP addr=10.0.0.249 port=38974
[S] I said Hello to Client!
[S] Client says: Hi~I am Client!!

ubuntu@41983:~/hw12$
```

```
ubuntu@41983:~/hw12$ ./n2 & ./n2 & ./n2
[1] 3010698
[2] 3010699
[C] Connecting...
[C] Connected!
[C] Connecting...
[C] Connected!
[C] Connecting...
[C] Server says: Hello~I am Server!

[C] I said Hi to Server!!
[C] Connected!
[C] Server says: Hello~I am Server!

[C] I said Hi to Server!!
[C] Server says: Hello~I am Server!

[C] I said Hi to Server!!
[1]-  Done                    ./n2
[2]+  Done                    ./n2
ubuntu@41983:~/hw12$
```

[예제 1] 비정상 실행 예

- 클라이언트 소스 수정
  - Send() 하지 않고, 10초간 sleep()
  - 맨 첫 클라이언트만 수정한 코드로 실행
- 결과
  - 다른 클라이언트들은 연결 수립은 됨
    - 포트가 열려있으므로, 커널이 수락한 것
  - 그러나 recv() 에서 계속 대기하여야 함
    - 서버 프로세스가 recv()에서 대기하느라, send()를 할 수 없으므로
  - 10초 후, 첫 번째 클라이언트가 소켓을 닫음으로써 연결이 종료되고,
    이로 인해 recv() 가 취소되어 다음 동작을 수행 가능함

```
45      //2. say hi to server
46      /*
47      if (send(c_sock, hello, sizeof(hello)+1, 0) == -1) {
48          perror("[C] Can't send message");
49          exit(1);
50      }
51
52      printf("[C] I said Hi to Server!!\n");
53      */
54
55      printf("[C] I am going to sleep...\n");
56      sleep(10);
57
58      close(c_sock);
```

```
ubuntu@41983:~/hw12$ ./n1
[S] waiting for a client..#00
[S] Connected: client IP addr=10.0.0.249 port=39018
[S] I said Hello to Client!
```

```
ubuntu@41983:~/hw12$ ./n2_s & ./n2 & ./n2
[1] 3013779
[2] 3013780
[C] Connecting...
[C] Connected!
[C] Server says: Hello~I am Server!

[C] I am going to sleep...
[C] Connecting...
[C] Connected!
[C] Connecting...
[C] Connected!
```

Process-based Parallel Socket Programming

- 프로세스 기반 동시 동작 서버
  - Door-keeper process
    - 기존 서버 코드와 같이 서버 소켓을 열고 bind(), listen() 수행
    - 새로운 연결이 수립되면, child process를 만들어, 아래 service 동작을 수행하게 함
    - 서버는 fork() 이후, 즉각 다시 accept()에서 대기. 이를 3회 반복함
    - 종료 전, 생성된 child 개수만큼 wait()를 수행하여 모든 종료 상태값을 출력 후, 종료
  - Service process
    - 기존과 같이 send(), recv() 수행하고, 소켓을 닫고, 종료
- 개인과제 12-1
  - 뒤의 12-2와 함께 제출
  - 파일 명: hw12/p-server.c, hw12/p-client.c
  - 앞의 비정상 실행 예제와 같이, 중간에 sleep()하는 클라이언트와 일반 클라이언트를 혼합하여 실행한 후, 결과를 캡처하여 보고서에 기재

[예제 2] 동시 동작 서버 수행 예제

```
ubuntu@41983:~/hw12$ ./server
[S] waiting for a client..#00
[S] Connected: client IP addr=10.0.0.249 port=42254
[S] waiting for a client..#01
[SS] Service: I am your child! pid=3052751
[SS] I said Hello to Client!
[S] Connected: client IP addr=10.0.0.249 port=42256
[S] waiting for a client..#02
[SS] Service: I am your child! pid=3052752
[SS] I said Hello to Client!
[SS] Client says: Hi~I am Client!!

[S] Connected: client IP addr=10.0.0.249 port=42258
[S] Child #00 is finished with 0
[SS] Service: I am your child! pid=3052753
[SS] I said Hello to Client!
[SS] Client says: Hi~I am Client!!

[S] Child #01 is finished with 0
[SS] Client says:
[S] Child #02 is finished with 0
ubuntu@41983:~/hw12$
```

```
ubuntu@41983:~/hw12$ ./n2_s & ./n2 & ./n2
[2] 3052748
[3] 3052749
[C] Connecting...
[C] Connected!
[C] Server says: Hello~I am Server!

[C] I am going to sleep...
[C] Connecting...
[C] Connected!
[C] Server says: Hello~I am Server!

[C] I said Hi to Server!!
[1]   Done                    ./n2_s
[C] Connecting...
[C] Connected!
[C] Server says: Hello~I am Server!

[C] I said Hi to Server!!
[3]+  Done                    ./n2
ubuntu@41983:~/hw12$
```

개인 과제 12-2: Multi-threaded File Transfer Service

- 프로그램 2개 작성: server.c and client.c
  - 개인 과제 11과 동일한 내용의 파일 전송 서버-클라이언트 프로그램
  - 쓰레드를 사용하여 여러 전송 요청을 동시에 처리하여야 함
    - 최대 10개
  - 여러 전송 요청이 동시에 처리됨을 확인할 수 있도록 실험을 구성하고, 결과를 기재
  - 추가 점수 (최대 50%)
    - 12-1 의 프로세스를 이용한 예제와 비교하여, 성능, 메모리 사용량 등을 비교
- 주의 사항
  - /home/ubuntu/hw12 에서 위 파일명으로 새로운 파일 생성하여 작업할 것
- 제출 기한
  - 12/16 (수) 23:59 (기한 이후 제출 불가)
  - 12-1, 12-2 과제를 함께 압축하여 LMS 제출
    - 동작 설명과 결과가 포함된 간단한 보고서(PDF!!!) & 소스 파일들