HW2

2021115744_권구태

1. 소켓 통신을 이용한 주소 변환 프로그램

[클라이언트**]**

- dotted-decimal 형태의 주소(192.168.0.1)를 화면에서 입력받고, BUF_SIZE 크기의 배열에 저장한 다음 해당 주소를 서버로 전송

```
printf("Input dotted-decimal address: ");
        fgets(address, BUF_SIZE, stdin);
        // 입력 시 정확히 BUF_SIZE - 1 크기만큼 입력하면 해당 문자열의 인덱스에 유효한 값이 있는데, 이를 NULL로 바꿔버림. 이를 방지하기 위함.
        size_t len = strlen(address);
        if (len > 0 && address[len - 1] == '\n') {
            address[len - 1] = '\0';
        }

        // 서버 주소 전송
        write(sock, address, sizeof(address));
```

- “quit” 를 입력하며, 서버로 “quit” 메시지 전송 후 클라이언트 종료

```
        if (strcmp(address, "quit") == 0) {
            break;
        }

        read_len=read(sock, message, BUF_SIZE);
        if (read_len == 0) {  // 서버가 연결을 종료한 경우
            printf("Server disconnected.\n");
            break;
        } else if (read_len == -1) {
            error_handling("read() error!");
        }
```

- 서버에게 quit를 전송하고 나서 위의 if 문을 바로 실행하는게 아니라, read 함수로 서버에서 보내는 데이터를 먼저 기다림. 그래서 서버에서는 quit가 실행되면 close하니까 이를 반영해서 0이 오면 while문 종료하면 됨.
  if (strcmp(address, "quit") == 0) break;

[서버]

```
   char SuccessMessage[]="[Rx] Address conversion success\n";
   char FailMessage[] = "[Rx] Address conversio fail: Format
error.\n";
   char SuccessMessageInServer[]="[Tx] Address conversion
success\n";
   char FailMessageInServer[] = "[Tx] Address conversio fail:
Format error.\n";
   char FinishMessage[] = "quit received and exit program!\n";
   char address[BUF_SIZE];
```

- 전송된 dotted-decimal 주소를 inet_aton() 및 inet_ntoa() 함수를 호출하여 변환하고 그 결과가 정상인 경우에 서버쪽 화면에 출력

```
                // 문자열 주소 -> 숫자로 변환 성공
                printf("inet aton: %s -> %#x\n", address,
addr_inet.sin_addr.s_addr);

                // 32비트 숫자를 다시 문자열로 변환
                temp = addr_inet.sin_addr.s_addr;
                str_ptr = inet_ntoa(addr_inet.sin_addr);
                printf("inet ntoa: %#x -> %s\n", temp, str_ptr);
```

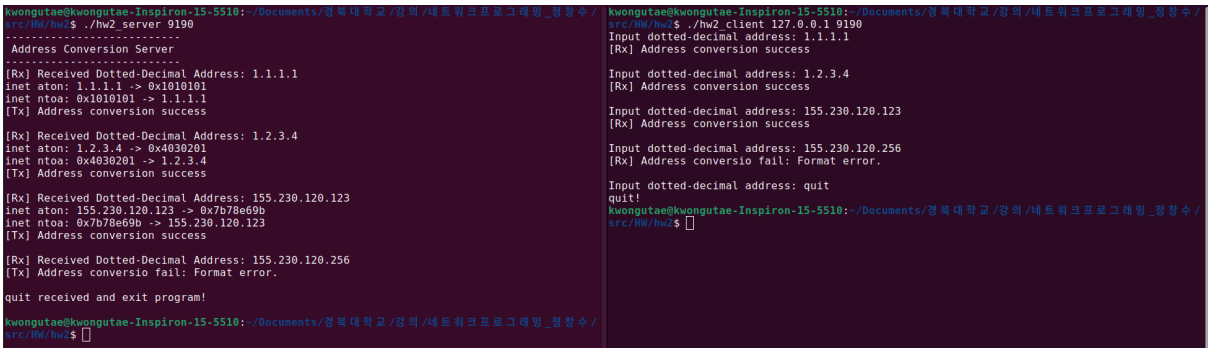
- 주소가 정상인 경우: 서버는 변환된 주소를 출력하고 “Address conversion success” 문자열을 화면 출력 및 클라이언트로 전송

```
                printf("%s\n", SuccessMessageInServer);
                // 주소 변환 성공했다는 메세지 전공
                write(clnt_sock, SuccessMessage,
sizeof(SuccessMessage));
```

- 잘못된 주소인 경우: “Address conversion fail: Format error” 문자열을 화면에 출력하고 해당 문자열을 클라이언트로 전송

```
            // 문자열로 받은 주소를 32비트 숫자로 변환
            if (!inet_aton(address, &addr_inet.sin_addr)) {
                // 잘못된 주소를 입력했을 시, 오류 메세지 전송
                printf("%s\n", FailMessageInServer);
                write(clnt_sock, FailMessage,
sizeof(FailMessage));
```

[실행 결과**]**



[새롭게 알게된 점 및 주의사항**]**

1. 서버에게 quit를 전송하고 나서 위의 if 문을 바로 실행하는게 아니라, read 함수로 서버에서 보내는 데이터를 먼저 기다림. 그래서 서버에서는 quit가 실행되면 close하니까 이를 반영해서 0이 오면 while문 종료하면 됨.
   if (strcmp(address, "quit") == 0) break;
2. 과제 초반엔 inet_addr함수가 10진수 형태로된 문자열을 32비트 정수형으로 변한하여 서버에까지 전달해주는줄 알았음(즉 소켓에 넣어주는줄 알았음). 하지만 그게 아니라 서버와 클라이언트 간의 데이터는 소켓을 통해서만 이루어지고, 해당 소켓에 쓰고 읽는 방식으로만(read, write 등) 이루어짐.
3. ```
   inet_aton(address, &addr_inet.sin_addr)
   ```
   inet_aton 함수 자체가 두번째 인자를 왜 addr_inet.sin_addr.s_addr의 값을 한번에 넣지 않는가? -> 이는 int inet_aton(const char *cp, struct in_addr *inp); 함수 프로토타입을 보면, 포인터 형식으로 받기때문에, struct in_addr의 주소값을 전달해야 함. 즉 구조체의 주소를 넘겨줘야 한다는 말임. 따라서 addr_inet.sin_addr.s_addr의 값을 바로 넣으면 해당 값을 uint32_t 형식이라 컴파일 에러 발생.
4. inet_ntoa도 비슷한 원리
   ```
   str_ptr = inet_ntoa(addr_inet.sin_addr);
   ```
   char *inet_ntoa(struct in_addr in); 프로토타입을 보면 인자는 구조체이고, 반환 형식이 포인터이다. 따라서 받는 변수도 포인터 변수여야 함.
5. %#x는 32비트 unsigned int 형식
   ```