

## Operation System – COSE341

### 실습 2 과제 레포트 – IPC – Assignment 2

고려대학교 컴퓨터학과 2017320108

고재영

개발 환경 : Oracle VM VirtualBox, Linux, Ubuntu 18.04.5, Visual Studio 2019

과제 만기일 : 2022/05/09 10:29 AM

유의 사항 : 가독성을 위해 그림판으로 편집한, Visual studio 2019로 작성된 .c 파일 코드 전문과 작성된 코드에서 특정 영역들에 대한 여러가지 설명으로 구성되어 있다.

제출 날짜 : 2022.05.07

#### [0] - 과제 2. Pipe

##### ● 목표: pipe 사용하기

Client가 server에 정수 값을 보내면, server가 받은 값을 제공하여 client에 반환해주는 코드를 작성

##### 1. 아래 기능을 구현할 것

assignment\_2/client.c

- init receive\_fd and send\_fd
- send msg and receive msg

assignment\_2/server.c

- make pipes and init fds
- read msg
- write msg

##### 2. 추가한 코드의 역할과 필요성, 작동 방식에 대해 작성하여 PDF로 제출

## [1] assignment\_2/client.c

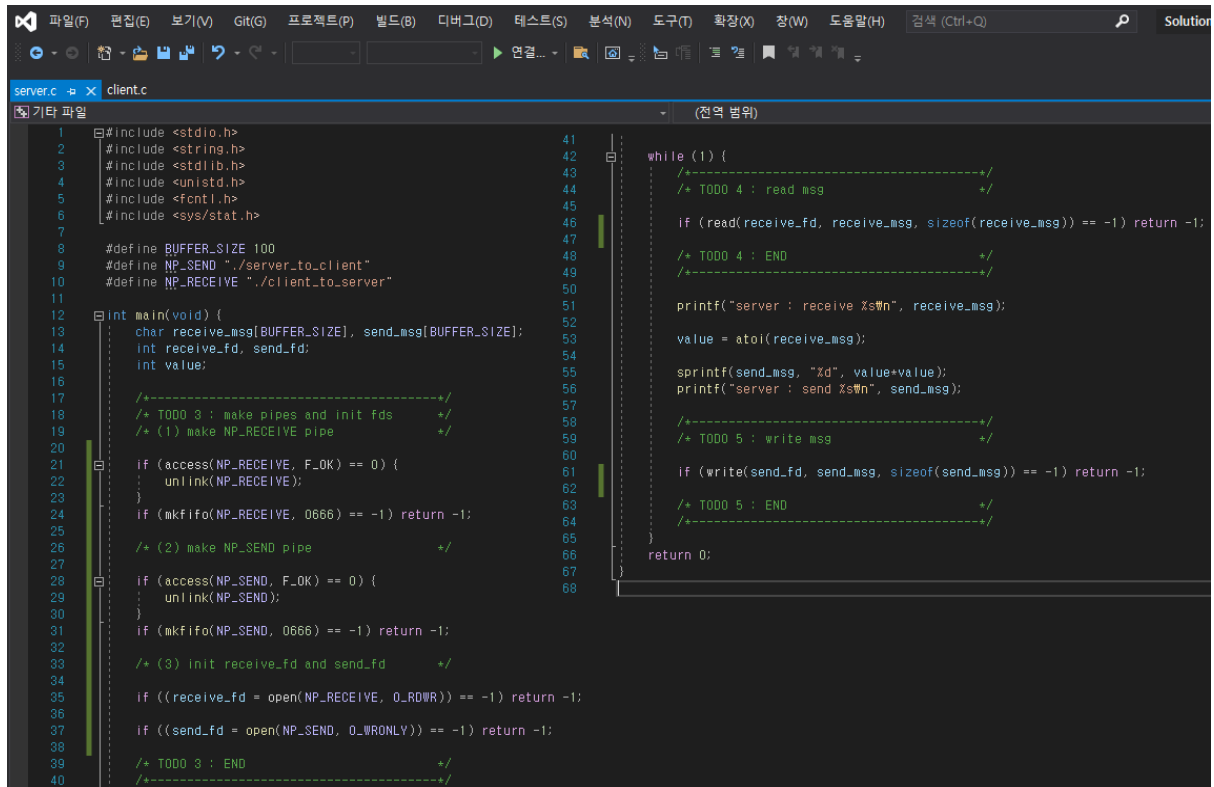
```
파일(F)  편집(E)  보기(V)  Git(G)  프로젝트(P)  빌드(B)  디버그(D)  테스트(S)  분석(N)  도구(T)
확장(X)  창(W)  도움말(H)

server.c  client.c  ✕

기타 파일  (전역 범위)  main

1  #include <stdio.h>
2  #include <string.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <sys/stat.h>
6
7  #define BUFFER_SIZE 100
8  #define NP_RECEIVE "./server_to_client"
9  #define NP_SEND "./client_to_server"
10
11 int main(void) {
12     char receive_msg[BUFFER_SIZE], send_msg[BUFFER_SIZE];
13     int receive_fd, send_fd;
14     /*-----*/
15     /* TODO 1 : init receive_fd and send_fd */
16
17     if ((receive_fd = open(NP_RECEIVE, O_RDONLY)) == -1) return -1;
18
19     if ((send_fd = open(NP_SEND, O_WRONLY)) == -1) return -1;
20
21     /* TODO 1 : END */
22     /*-----*/
23
24     for (int i=12; i<16; i++) {
25         printf("client : send %d\n", i);
26         sprintf(send_msg, "%d", i);
27         /*-----*/
28         /* TODO 2 : send msg and receive msg */
29
30         if (write(send_fd, send_msg, sizeof(send_msg)) == -1) return -1;
31
32         if (read(receive_fd, receive_msg, sizeof(receive_msg)) == -1) return -1;
33
34         /* TODO 2 : END */
35         /*-----*/
36
37         printf("client : receive %s\n\n", receive_msg);
38
39         usleep(500*1000);
40     }
41     return 0;
42 }
43
```

## [2] assignment\_2/server.c



```
server.c  client.c
기타 파일
(전역 범위)

1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <unistd.h>
5  #include <fcntl.h>
6  #include <sys/stat.h>
7
8  #define BUFFER_SIZE 100
9  #define NP_SEND "./server_to_client"
10 #define NP_RECEIVE "./client_to_server"
11
12 int main(void) {
13     char receive_msg[BUFFER_SIZE], send_msg[BUFFER_SIZE];
14     int receive_fd, send_fd;
15     int value;
16
17     /*-----*/
18     /* TODO 3 : make pipes and init fds */
19     /* (1) make NP_RECEIVE pipe */
20
21     if (access(NP_RECEIVE, F_OK) == 0) {
22         unlink(NP_RECEIVE);
23     }
24     if (mkfifo(NP_RECEIVE, 0666) == -1) return -1;
25
26     /* (2) make NP_SEND pipe */
27
28     if (access(NP_SEND, F_OK) == 0) {
29         unlink(NP_SEND);
30     }
31     if (mkfifo(NP_SEND, 0666) == -1) return -1;
32
33     /* (3) init receive_fd and send_fd */
34
35     if ((receive_fd = open(NP_RECEIVE, O_RDONLY)) == -1) return -1;
36
37     if ((send_fd = open(NP_SEND, O_WRONLY)) == -1) return -1;
38
39     /* TODO 3 : END */
40     /*-----*/
41
42     while (1) {
43         /* TODO 4 : read msg */
44
45         if (read(receive_fd, receive_msg, sizeof(receive_msg)) == -1) return -1;
46
47         /* TODO 4 : END */
48         /*-----*/
49
50         printf("server : receive %s\n", receive_msg);
51
52         value = atoi(receive_msg);
53         sprintf(send_msg, "%d", value+value);
54         printf("server : send %s\n", send_msg);
55
56         /* TODO 5 : write msg */
57
58         if (write(send_fd, send_msg, sizeof(send_msg)) == -1) return -1;
59
60         /* TODO 5 : END */
61         /*-----*/
62     }
63     return 0;
64 }
```

### [3] 코드 설명

Named pipe를 통한 message passing을 하기 위해서, 먼저 pipe를 생성하고 pipe file descriptor에 대한 초기화가 필요하다. Server.c에서 TODO 3 영역의 line 21 ~ 24가 receive를 위한 pipe, 그리고 line 27 ~ 31이 send를 위한 pipe를 생성하는 코드다. line 21 ~ 24를 설명하자면, 먼저 기존의 pipe가 있는지 검사를 하기위해, access하여, access 반환값이 0이 된다면 기존 생성된 pipe가 존재한다는 의미이기 때문에 unlink 함수를 통해 제거해준다. 그 이후, mkfifo 함수를 통해 접근 권한이 0666이며, define으로 정의된 `./server_to_client` 란 pathname을 가리키는 NP\_SEND란 이름의 선입선출 형태의 fifo pipe를 생성한다. line 27 ~ 31은 이와 동일한 원리이다.

위와 같이 두 pipe를 생성했으므로, 이를 사용하고자 client와 server 둘 다 pipe file descriptor에 대한 초기화를 해야 한다. 이 때, receive용 pipe의 file descriptor와 send용 pipe의 file descriptor가 server에서 초기화를 할 때, client에서 또한 동일하게 open함으로 synchronization 즉 동기화한다. Client.c의 line 17 ~ 19와 server.c의 line 35 ~ 37가 바로 이에 해당한다. Receive를 위한 pipe file descriptor인 receive\_fd는 NP\_RECEIVE란 pipe에 읽기/쓰기용으로 열며, send를 위한 pipe file descriptor인 send\_fd는 NP\_SEND란 pipe에 read only 읽기 전용으로 열린다. 이는 send할 때에는 전송에만 목적을 두어 read only로 제한하지만, receive한 것에 대해선 해당 메시지를 가지고 추가 작업을 수행하기 때문에 (이 예제에선 제공시키기) write도 가능하도록 하는 것이다.

이제 client.c와 server.c의 나머지 부분을 보면, client가 send를 한다고 출력 후 `write(send_fd, send_msg, sizeof(send_msg))` 통해 send\_msg안에 담긴 해당 값을 message passing 한다. 그럼 server는 `read(receive_fd, receive_msg, sizeof(receive_msg))` 통해 passing된 메시지를 읽는다. 곧 바로 그 메시지를 받았다고 출력한 뒤, value란 변수를 이용하여 받은 값을 제공해주고 client에게 `write(send_fd, send_msg, sizeof(send_msg))` 통해 제공된 값을 message passing한다. 그러면 client가 `read(receive_fd, receive_msg, sizeof(receive_msg))` 통해서 passing된 메시지를 보고 제공된 값을 받았다고 출력한다. 따라서 위와 같은 코드로 작성한 것이다.