OS01 6주차 과제 - 201702081

- FIFO & Semaphore

1. 코드 이미지 및 설명

(1) Server

```c
int main()
{

    int fd;
    int cnt = 0;            // 턴 수
    char buf[BUF_SIZE];

    const char *msg = "ping\n";
    sem_t *p_sem;
    int score = 100;

    /* TODO */

    // FIFO : 보낸 내용을 기록할 파일을 지정 (없으면 생성)
    if(-1 == mkfifo(FIFO_PATH, 0600))
        printf("FIFO ERROR \n");
    else printf("FIFO OK \n");

    // 세마포어 초기화
    if((p_sem = sem_open("my_semaphore", O_CREAT, 0600, 1)) == SEM_FAILED) {
        perror("SEM_OPEN_ERR \n");
        exit(1);
    }
    else printf("SEM_OPEN OK \n");
```

➔ mkfifo를 통해 두 프로세스 간 매개 파일을 만들어주고, 권한은 read, write만 가능하도록 지정한다.

이후 다른 프로세스와의 실행 순서 제어를 위해 my_semaphore 라는 이름의 네임드 세마포어를 만들어준다. 이 때 초기값은 1로 설정해줌으로 상호 배제를 만족하도록 하였다.

```c
while(cnt < TURN) {

    // 보냄
    if(-1 == (fd = open(FIFO_PATH, O_WRONLY)))
        printf("OPEN ERR \n");
    printf("Your turn! : ");
    memset(buf, 0, BUF_SIZE);
    fgets(buf, BUF_SIZE, stdin);
    printf("%s \n", buf);
    write(fd, buf, strlen(buf));
    close(fd);
    if(strcmp(buf, msg) != 0) {
        score -= 20;
        printf("wrong! -20 \n");
    }
    cnt++;
    close(fd);
    sem_post(p_sem);


    // 받음
    sem_wait(p_sem);
    if(-1 == (fd = open(FIFO_PATH, O_RDONLY)))
        printf("OPEN ERR \n");
    memset(buf, 0, BUF_SIZE);
    read(fd, buf, BUF_SIZE);
    printf("[opponent] %s \n", buf);
    close(fd);

}


printf("Done! Your score: %d\n", score);
return 0;
```

➔ Server에서는 먼저 송신을 시작하게 된다. stdin으로부터 입력을 받아 buf 에 저장하고, 그것을 FIFO 파일에다 쓴다. 틀린 메시지일 경우에는 점수를 차감하고 메시지를 출력한다.

이후에는 sem_post를 통하여 wait 중인 Client를 깨워준 뒤 자신은 sleep 상태가 된다.

➔ Client로부터의 sem_post로 인해 깨어나면, FIFO 파일의 내용을 읽어 상대방의 메시지를 출력한다.

➔ Buffer에다 무언가를 쓰기 전에, memset을 통하여 초기화함으로써 이전 내용들의 누적을 막아준다.

(2) Client

```c
int main()
{

    int fd;
    int cnt = 0;            // 턴 수
    char buf[BUF_SIZE];

    const char *msg = "pong\n";
    sem_t *p_sem;
    int score = 100;

    /* TODO */

    // 세마포어 초기화
    if((p_sem = sem_open("my_semaphore", O_CREAT, 0600, 1)) == SEM_FAILED) {
        perror("SEM_OPEN_ERR");
        exit(1);
    }
    else printf("SEM_OPEN OK \n");
```

➔ 전반부는 Server와 동일하지만, mkfifo를 통해 FIFO 파일을 만드는 과정은 하지 않는다.

```c
while(cnt < TURN) {

    // 받음
    sem_wait(p_sem);
    if(-1 == (fd = open(FIFO_PATH, O_RDONLY)))
        printf("OPEN ERR \n");
    memset(buf, 0, BUF_SIZE);
    read(fd, buf, BUF_SIZE);
    printf("[opponent] %s \n", buf);
    close(fd);
```

```
    // 보냄
    if(-1 == (fd = open(FIFO_PATH, O_WRONLY)))
        printf("OPEN ERR \n");
    printf("Your turn! : ");
    memset(buf, 0, BUF_SIZE);
    fgets(buf, BUF_SIZE, stdin);
    printf("%s \n", buf);
    write(fd, buf, strlen(buf));
    close(fd);
    if(strcmp(buf, msg) != 0) {
        score -= 20;
        printf("wrong! -20 \n");
    }
    cnt++;
    close(fd);
    sem_post(p_sem);

}


printf("Done! Your score: %d\n", score);
return 0;
```

➔ 후반부는 Server에서 보내는 부분과 받는 부분이 순서만 역전되었다.

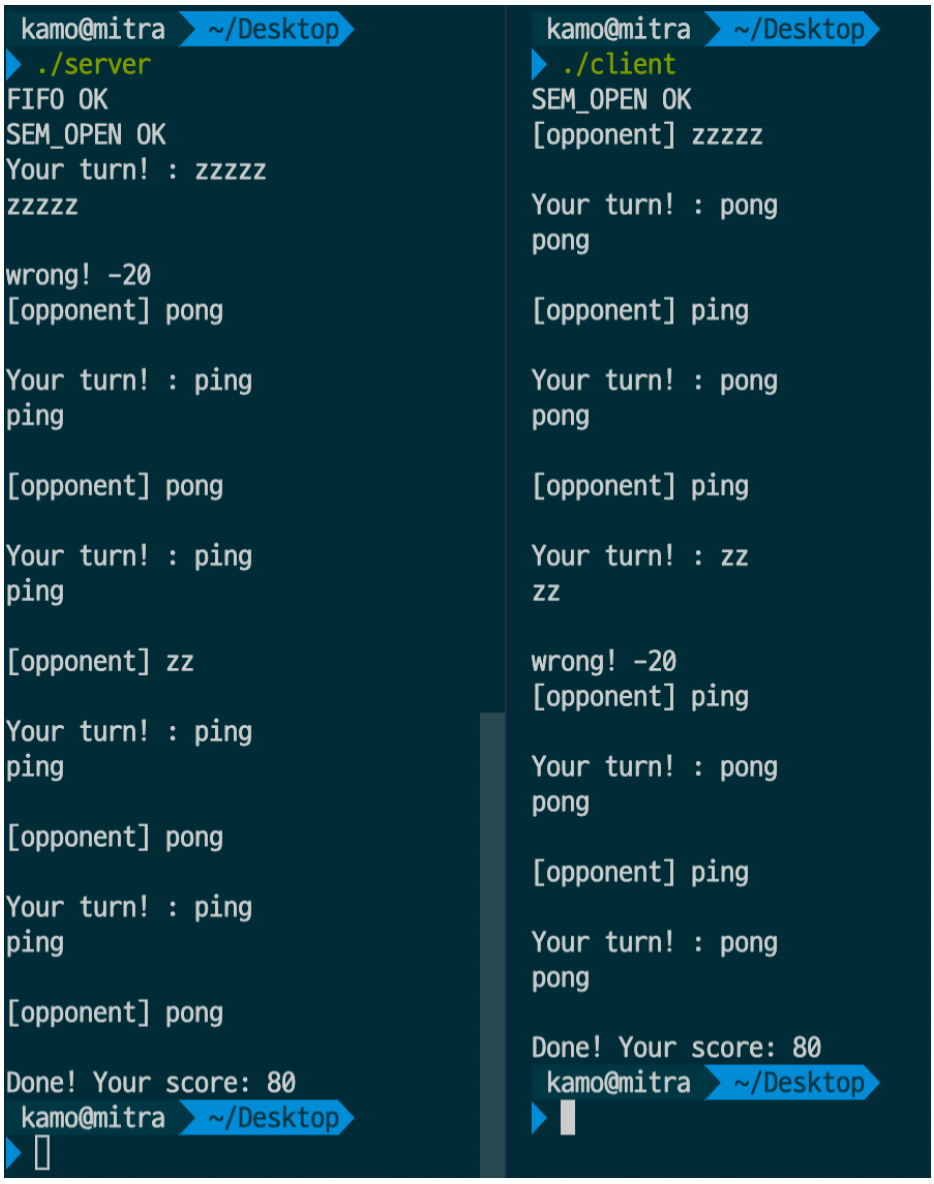
➔ 처음엔 sem_wait를 통하여 Server가 깨워주길 기다린 뒤, 깨어나면 FIFO 파일에 저장된 메시지를 읽고 출력한다.

➔ 그 다음에는 바로 메시지를 입력받아 FIFO 파일에다 쓰고(전송), wait중인 server를 깨워준다

2. 예상 결과 및 실행 결과

➔ 예상 결과 : 자신이 입력한 내용이 FIFO 파일을 통하여 상대에게 전달되어 출력됨. 한쪽이 입력할 때에는 입력을 받지 않음

➔ 실행 결과



둘다 한번씩 틀리니까 20점씩 깎여서 80점씩 나왔다 -> 점수계산 잘됨

상대방이 입력한 내용이 FIFO 파일을 통하여 잘 전달되었음을 확인할 수 있다.

한쪽이 입력할 때에는 다른 쪽의 입력이 활성화되지 않았다.

- 느낀 점

  FIFO open이 안되고 자꾸 프로그램이 뻗어서 삽질을 엄청 했는데 재부팅하니까 해결돼서 허망했다