설계4 피드백 바탕으로 다시 구현

- 교수님께서 다시 수정해서 내면 점수 올려주신다고해서 다시 짠 코드

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <unistd.h>
#include <dirent.h>
#include <string.h>
#include <time.h>
#include <ftw.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/mman.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <sys/sem.h>
#include <sys/shm.h>

#define BUFSIZE 512

static struct sigaction act;
int mode,pos,cnt;
pid_t pid;
int p[2][2];    // p[0]이 control, p[1]이 data

void catchInt(int signo){
    int temp;
    scanf("%d",&temp);

    // 복구 신호왔을 때위치 업데이트 하고  control pipe에 넣음
    pos += temp;
    write(p[0][1], &pos, sizeof(int));

    // 자식 프로세스 종료
    exit(0);
}

void create_Process(){
    if((pid = fork()) == 0){
        // 자식 프로세스 SIG_INT 신호 처리
        act.sa_handler = catchInt;
        sigaction(SIGINT,&act,NULL);
    }
}

void r_init(){
    int i;

    // 파이프 생성
    for(i=0; i<2;i++){
```

```
        pipe(p[i]);
    }

    // 부모 프로세스 SIG_INT 신호 무시
    act.sa_handler = SIG_IGN;
    sigaction(SIGINT, &act, NULL);

    // 초기 자식 프로세스 생성
    create_Process();
}

void r_scanf(char *input, int *data){
    if(pid == 0){
        // 자식 프로세스
        scanf(input,data);

        // Data pipe에 입력 받은 값 쌓기
        write(p[1][1],data,sizeof(int));

        // 자식 프로세스가 얼마나 입력 받았는지 세기
        pos++;
    } else{
        // 부모프로세스

        if(mode == 0){

            // 정상 모드, 자식에서 입력받고 처리하는 중, blocking
            read(p[0][0],&pos,sizeof(int));

            if(pos >= 0){
                // 복구 모드
                mode = 1;
            }
            else {
                // 자식 프로세스 정상 종료
                mode = 2;
            }

            // 종료한 자식 프로세스 기다리기
            waitpid(pid,0,0);
        }

        // 복구 모드
        read(p[1][0],data,sizeof(int));

        cnt++;

        if(cnt == pos && mode == 1){
            int temp;

            // 복구가 끝나면 정상 모드로 변경
            mode = 0;

            // Data pipe Non block 설정
            fcntl(p[1][0], F_SETFL,O_NONBLOCK);

            // data 비우기
            while(read(p[1][0],&temp,sizeof(int)) > 0);
```

```
            // 새로운 자식 프로세스 생성
            create_Process();
        }
    }
}

void r_printf(char *str, int data){
    if(pid == 0){
        // 자식 프로세스 출력
        printf(str,data);
    }
}

void r_close(){
    if(pid == 0){
        pos = -1;
        // 자식 프로세스 정상종료 됨을 부모에게 알려줌
        write(p[0][1], &pos, sizeof(int));
        exit(0);
    }
}
```