UNIX 프로그래밍 실습
(2019/11/12)

1. fifo를 이용하여 통신하는 두 개의 프로그램을 작성 합니다. 프로그램 A는 외부 입력으로 정수를 입력 받아 프로그램 B에 전달합니다. 프로그램 B는 전달 받은 정수에 +8을 한 뒤 프로그램 A에 돌려줍니다. 프로그램 A는 돌려받은 정수 값을 출력 합니다. -1이 입력되면 두 프로그램은 종료 합니다.

```
int main(void){
        char f[2][3]={"f1", "f2"};
        int i, in, fd[2];

        // “f1”과 “f2” open

        for (;;){
                // “f1”에서 정수 읽기
                  printf("%d₩n", in);
                if (in==-1)
                        exit(0);
                in=in+8;
                // “f2”로 정수 보내기
        }

        return 0;
}
```

```
int main(void){
        char f[2][3]={"f1", "f2"};
        int i, in, fd[2];

        // FIFO 2개 만들기
        // “f1”과 “f2” open

        for (;;){
                scanf("%d", &in);
                // “f1”으로 정수 보내기
                if (in==-1)
                        exit(0);
                // “f2”에서 정수 읽기
                printf("%d₩n", in);
        }

        return 0;
}
```

2. server process는 세 개의 client process들과 데이터를 주고받기 위한 fifo를 만듭니다. 각 client는 미리 정해진 이름의 FIFO로 접속하여, 표준 입력으로 입력된 정수를 server process에게 전송 합니다. server process는 client process로부터 전송된 정수 값에 +8을 한 후, 해당 client에게 다시 보냅니다. client process는 돌려받은 정수 값을 표준 출력으로 출력 합니다. client process는 정수 데이터의 입/출력 작업을 5회 반복 한 후 종료 합니다. client process로 부터의 입력을 blocking으로 기다리기 위해 select 문장을 사용 합니다.

```
int main(int argc, char **argv){
        char f[6][3]={"f1", "f2", "f3", "f4", "f5", "f6"};
        int i, in, k, fd[2];

        k=atoi(argv[1]);
        // 필요한 FIFO open

        for (i=0; i<5; i++){
                scanf("%d", &in);
                // 정수 보내기
                // 정수 받기
                printf("%d₩n", in);
        }

        exit(0);
}
```

```
int main(void){
        char f[6][3]={"f1", "f2", "f3", "f4", "f5", "f6"};
        fd_set set, master;
        int i, j, k, in, fd[6];
        struct timeval t;

        for (i=0;i<6;i++)
                // FIFO 만들기

        for (i=0;i<3;i++){
                // FIFO open
        }

        // master에 적절한 file descriptor 설정

        // timer 설정
        while (                                                    ) {
                for (i=0; i<3; i++){
                        if ( // message 도착 확인){
                                if ( // message read & 확인){
                                        in=in+8;
                                        // message 보내기
                                }
                        }
                }
                // 추가로 필요한 코드는 적절한 위치에 추가...
        }
        // timer의 동작 확인 ...

        exit(0);
}
```

3. 네 개의 프로세스가 동기화를 하며 자신의 프로세스 id를 5회 출력하는 프로그램을 작성 합니다. 이 프로그램은 main() 함수의 arguments로 동기화에 참여하는 전체 프로세스 중 자신의 출력 순서를 입력 받습니다. 프로그램이 시작되면, 순서대로 자신의 프로세스 id를 출력 합니다. 동기화 작업은 fifo를 사용하여 수행 합니다.