Spring 2019 Operating Systems
Lab 04: Basic C Training IV

0. 과제 수행 전 유의사항
프로그래밍은 자기 손으로 작성해야 한다. 아이디어를 서로 공유하고 도움을 받는 것은 좋으나, 다른 사람의 코드를 그대로 가져다 쓰는 것은 자신에게도 손해이고 다른 사람에게 피해를 끼치는 결과를 낳게되므로 허용하지 않는다. 과제 체크 시 두 사람의 코드가 유사하면, 두 사람 모두에게 과제 점수를 부여할 수 없으니 유의바란다.

1. 실습의 목표
구조체에 대하여 실습한다.

2. 작성해야할 파일
elevator.h, main.h, main.c, ops.h, ops.c, move.h, move.c, request.c, Makefile

[elevator.h]
```
1
2 #define TOP_FLOOR     15
3 #define DELAY_FACTOR  50000
4
5 typedef enum {OUT, IN} BUTTON_STATUS;
6 typedef enum {DOWN, STOP, UP} DIRECTION_STATUS;
7 typedef struct
8 {
9     int current_floor;
10    BUTTON_STATUS *buttons;
11 } ELEVATOR;
12
```

[main.h]
```
1
2 #include <stdio.h>
3
4 #include "elevator.h"
5
6 extern void init_elevator(ELEVATOR *);
7 extern void run_elevator(ELEVATOR *);
8 extern void term_elevator(ELEVATOR *);
9
```

[main.c]
```
1
2 #include "main.h"
3
4 int
5 main(void)
6 {
7     static ELEVATOR elevator;
8
9     init_elevator(&elevator);
10    run_elevator(&elevator);
11    term_elevator(&elevator);
12
13    return 0;
14 }
15
```

[ops.h]

```
1
2 extern void move(ELEVATOR *);
3
```

[ops.c]

```
1
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <ctype.h>
5
6  #include "elevator.h"
7  #include "ops.h"
8
9  void
10 init_elevator(ELEVATOR *elev)
11 {
12     elev->buttons = calloc(TOP_FLOOR+1, sizeof(BUTTON_STATUS));
13     for (int i = 0; i <= TOP_FLOOR; i++) {
14         elev->buttons[i] = OUT;
15     }
16     elev->current_floor = 1;
17
18     return;
19 }
20
21 void
22 run_elevator(ELEVATOR *elev)
23 {
24     char buffer[81];
25     int floor;
26     char *p_str_in;
27
28     printf("\n\nThis elevator goes from basement \"0\" to floor \"%d\"", TOP_FLOOR);
29     printf("\nType floors & press return to start.  If no new floors,");
30     printf("\npress return key.  To quit, key EOF (Ctrl+C)");
31     printf("\n\nPlease enter floors: ");
32
33     while (fgets(buffer, 81, stdin)) {
34         p_str_in = buffer;
35         while (*p_str_in != '\n') {
36             while (*p_str_in == ' ') {
37                 p_str_in++;
38             }
39             if (!isdigit(*p_str_in)) {
40                 printf("\aInvalid floor %c\n", *p_str_in);
41                 p_str_in++;
42             } else {
43                 sscanf(p_str_in, "%d", &floor);
44                 if (floor == elev->current_floor) {
45                     printf("\n\aAlready on floor %d.", elev->current_floor);
46                 } else {
47                     if (floor < 0 || floor > TOP_FLOOR) {
48                         printf("\n\a%d invalid floor", floor);
49                     } else {
50                         elev->buttons[floor] = IN;
51                     }
52                 }
53                 while (isdigit(*p_str_in)) {
54                     p_str_in++;
55                 }
56             }
57         }
58         move(elev);
59         printf("\n\nPlease enter floors: ");
60     }
61
62     return;
63 }
64
65 void
66 term_elevator(ELEVATOR *elev)
67 {
68     free(elev->buttons);
69     return;
70 }
71
```

[move.h]

```
1
2 static void move_up(ELEVATOR *);
3 static void move_down(ELEVATOR *);
4 static void time_pass(int);
5
6 extern bool any_up_request(ELEVATOR *);
7 extern bool any_down_request(ELEVATOR *);
8
```

[move.c]

```
1
2 #include <stdio.h>
3 #include <stdbool.h>
4
5 #include "elevator.h"
6 #include "move.h"
7
8 void
9 move(ELEVATOR *elev)
10 {
11     static DIRECTION_STATUS direction = STOP;
12     bool any_up, any_down;
13
14     any_up = any_up_request(elev);
15     any_down = any_down_request(elev);
16
17     if (direction == UP) {
18         if (!any_up && any_down) {
19             direction = DOWN;
20         } else {
21             if (!any_up && !any_down) {
22                 direction = STOP;
23             }
24         }
25     } else if (direction == DOWN) {
26         if (!any_down && any_up) {
27             direction = UP;
28         } else {
29             if (!any_down && !any_up) {
30                 direction = STOP;
31             }
32         }
33     } else if (direction == STOP) {
34         if (any_up) {
35             direction = UP;
36         } else {
37             if (any_down) {
38                 direction = DOWN;
39             }
40         }
41     }
42
43     if (direction == UP) {
44         move_up(elev);
45     } else {
46         if (direction == DOWN) {
47             move_down(elev);
48         } else {
49             printf("\n***** NO BUTTON PRESSED *****");
50         }
51     }
52
53     return;
54 }
55
56 static void
57 move_up(ELEVATOR *elev)
58 {
59     printf("\nThe door is being closed ... We are going up.\n");
60
61     (elev->current_floor)++;
62     while (elev->buttons[elev->current_floor] != IN) {
63         time_pass(10000);
64         printf("Passing floor %d\n", elev->current_floor);
65         time_pass(10000);
66         (elev->current_floor)++;
67     }
68
69     elev->buttons[elev->current_floor] = OUT;
70     time_pass(10000);
71     printf("The door is being opened ...\n");
72     printf("\n ***** FLOOR %d ***** \n", elev->current_floor);
```

```
    time_pass(10000);

    return;
}

static void
move_down(ELEVATOR *elev)
{
    printf("\nThe door is being closed ... We are going down.\n");

    (elev->current_floor)--;
    while (elev->buttons[elev->current_floor] != IN) {
        time_pass(10000);
        printf("Passing floor %d\n", elev->current_floor);
        time_pass(10000);
        (elev->current_floor)--;
    }

    elev->buttons[elev->current_floor] = OUT;
    time_pass(10000);
    printf("The door is being opened ...\n");
    printf("\n ***** FLOOR %d ***** \n", elev->current_floor);
    time_pass(10000);

    return;
}

static void
time_pass(int time)
{
    for (long i = 0; i < (time *DELAY_FACTOR); i++);
    return;
}

```

[request.c]

```

#include <stdio.h>
#include <stdbool.h>

#include "elevator.h"

bool
any_up_request(ELEVATOR *elev)
{
    bool is_any = false;

    for (int check = elev->current_floor;
            check <= TOP_FLOOR && !is_any;
            check++) {
        is_any = (elev->buttons[check] == IN);
    }

    return is_any;
}

bool
any_down_request(ELEVATOR *elev)
{
    bool is_any = false;

    for (int check = elev->current_floor;
            check >= 0;
            check--) {
        is_any = is_any || (elev->buttons[check] == IN);
    }
}

```

[Makefile]

```
1
2  .SUFFIXES : .c .o
3  CC = gcc
4  CFLAGS = -g -c
5
6  OBJS = main.o ops.o move.o request.o
7  SRCS = $(OBJS:.o=.c)
8  TARGET = elev
9
10 all : $(TARGET)
11
12 $(TARGET) : $(OBJS)
13     $(CC) -o $(TARGET) $(OBJS)
14
15 clean :
16     $(RM) $(OBJS)
17
18 dep :
19     gccmakedep $(SRCS)
20
```

3. 컴파일

예제로 작성된 Makefile 은 위와 같다. 정확한 의존성이 나타나있지 않기 때문에 다음 명령으로 의존성을 만들어준다. 참고로 위에서 TARGET 키워드는 다음과 같이 수정되어야 할 것이다. lab04_학번

$ make dep

명령어를 수행하고 다시 Makefile 을 열어본다면, 많은 라인들이 추가되었을 것이다. 이후 make를 수행하여 아래와 같은 결과를 보이면 성공이다.

```
kwp@A1409-OSLAB-01:~/.toy/lab4/task$ make
gcc -g -c    -c -o main.o main.c
gcc -g -c    -c -o ops.o ops.c
gcc -g -c    -c -o move.o move.c
gcc -g -c    -c -o request.o request.c
gcc -o elev main.o ops.o move.o request.o
```

4. 출력내용 확인

본 과제는 엘레베이터가 움직이는 것과 같은 느낌을 주는 프로그램이다. 아래와 같은 출력을 보이면 된다.

```
kwp@A1409-OSLAB-01:~/.toy/lab4/task$ ./elev


This elevator goes from basement "0" to floor "15"
Type floors & press return to start.  If no new floors,
press return key.  To quit, key EOF (Ctrl+C)

Please enter floors: 5

The door is being closed ... We are going up.
Passing floor 2
Passing floor 3
Passing floor 4
The door is being opened ...

 ***** FLOOR 5 *****


Please enter floors: 0

The door is being closed ... We are going down.
Passing floor 4
Passing floor 3
Passing floor 2
Passing floor 1
The door is being opened ...

 ***** FLOOR 0 *****


Please enter floors:

***** NO BUTTON PRESSED *****

Please enter floors: ^C
```

5. 제출

위 파일을 코딩하고 테스트 한 다음, ls를 입력하여 자신의 파일 목록을 캡쳐한 후 추가하고 실행결과와 주석이 달린 코드를 워드 파일로 추합하여 제출한다.

코드로 적혀있는 내용을 단순히 한글로 옮긴 주석에는 점수를 부여할 수 없으니 참고하기 바란다.

자신의 디렉토리 목록과 실행과정은 스크린샷을 이용하도록 하고, 소스코드는 자신의 것을 Copy하여 첨부한다.

모든 내용은 MS워드나 Pages를 이용하여 한 개의 파일로 작성한다.