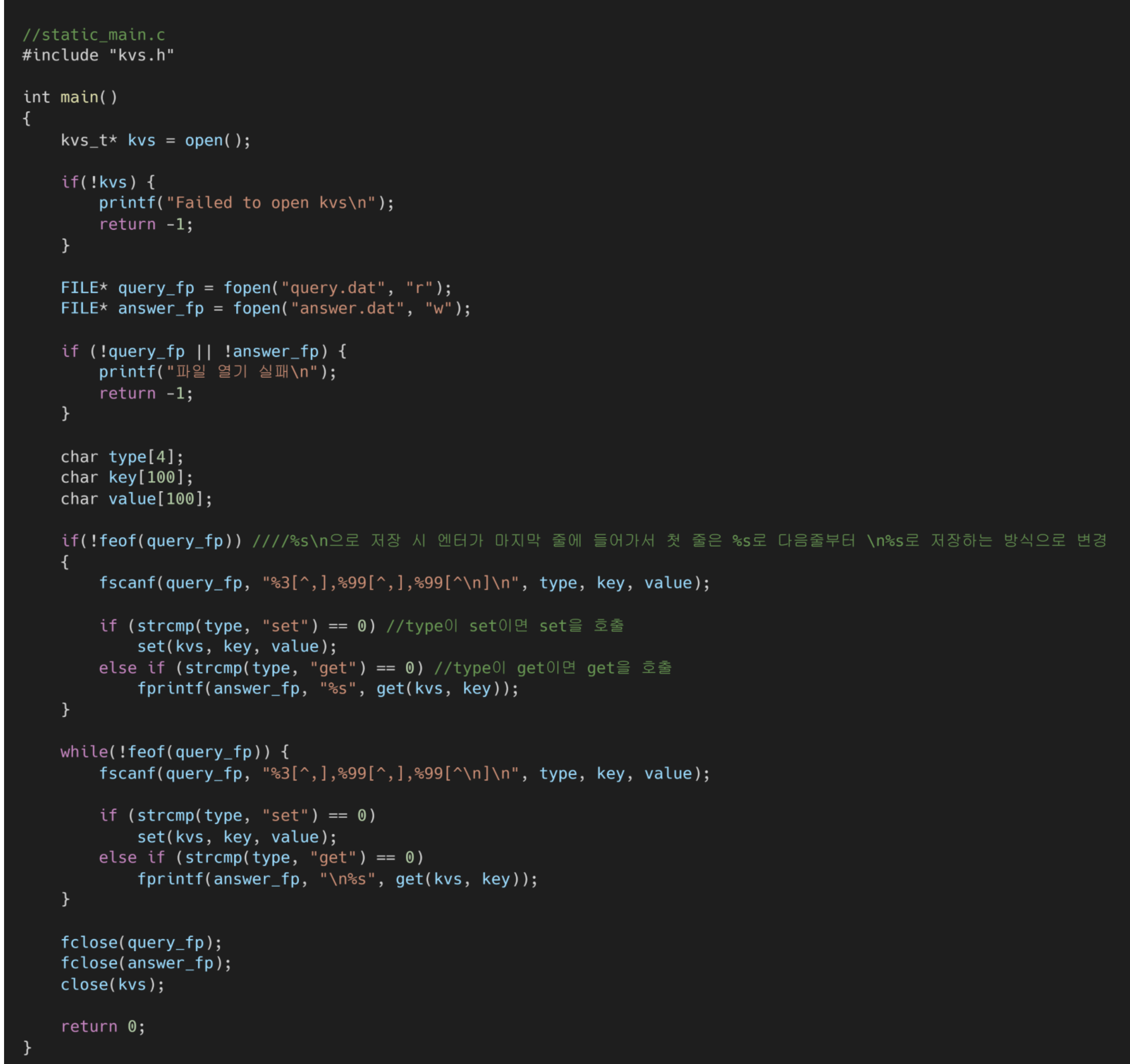
# static_main.c

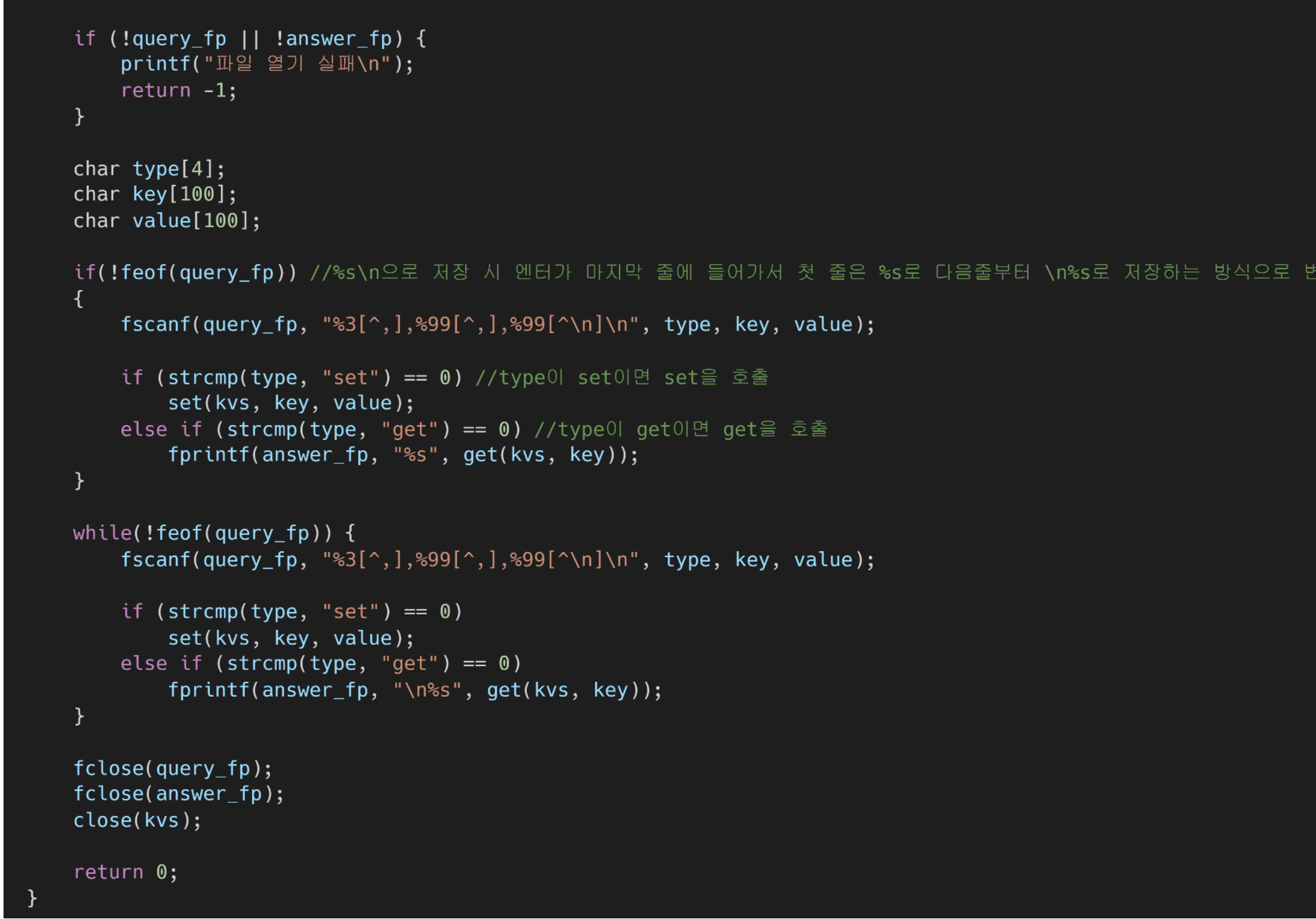


```c
//static_main.c
#include "kvs.h"

int main()
{
    kvs_t* kvs = open();

    if(!kvs) {
        printf("Failed to open kvs\n");
        return -1;
    }

    FILE* query_fp = fopen("query.dat", "r");
    FILE* answer_fp = fopen("answer.dat", "w");

    if (!query_fp || !answer_fp) {
        printf("파일 열기 실패\n");
        return -1;
    }

    char type[4];
    char key[100];
    char value[100];

    if(!feof(query_fp)) ////%s\n으로 저장 시 엔터가 마지막 줄에 들어가서 첫 줄은 %s로 다음줄부터 \n%s로 저장하는 방식으로 변경
    {
        fscanf(query_fp, "%3[^,],%99[^,],%99[^\n]\n", type, key, value);

        if (strcmp(type, "set") == 0) //type이 set이면 set을 호출
            set(kvs, key, value);
        else if (strcmp(type, "get") == 0) //type이 get이면 get을 호출
            fprintf(answer_fp, "%s", get(kvs, key));
    }

    while(!feof(query_fp)) {
        fscanf(query_fp, "%3[^,],%99[^,],%99[^\n]\n", type, key, value);

        if (strcmp(type, "set") == 0)
            set(kvs, key, value);
        else if (strcmp(type, "get") == 0)
            fprintf(answer_fp, "\n%s", get(kvs, key));
    }

    fclose(query_fp);
    fclose(answer_fp);
    close(kvs);

    return 0;
}
```

## dynamic_main.c

```c
//dynamic_main.c
#include "kvs.h"
#include <dlfcn.h>

int main() {
    void* handle;
    char* error;

    handle = dlopen("libkvs.so", RTLD_LAZY);

    if (!handle) {
        fprintf(stderr, "%s\n", dlerror());
        exit(1);
    }

    kvs_t* (*open)();
    void (*close)(kvs_t*);
    void (*set)(kvs_t*, const char*, const char*);
    const char* (*get)(kvs_t*, const char*);

    open = (kvs_t* (*)()) dlsym(handle, "open");
    close = (void (*)(kvs_t*)) dlsym(handle, "close");
    set = (void (*)(kvs_t*, const char*, const char*)) dlsym(handle, "set");
    get = (const char* (*)(kvs_t*, const char*)) dlsym(handle, "get");

    if ((error = dlerror()) != NULL) {
        fprintf(stderr, "%s\n", error);
        exit(1);
    }

    kvs_t* kvs = open();

    FILE* query_fp = fopen("query.dat", "r");
    FILE* answer_fp = fopen("answer.dat", "w");
```

dynamic_main.c

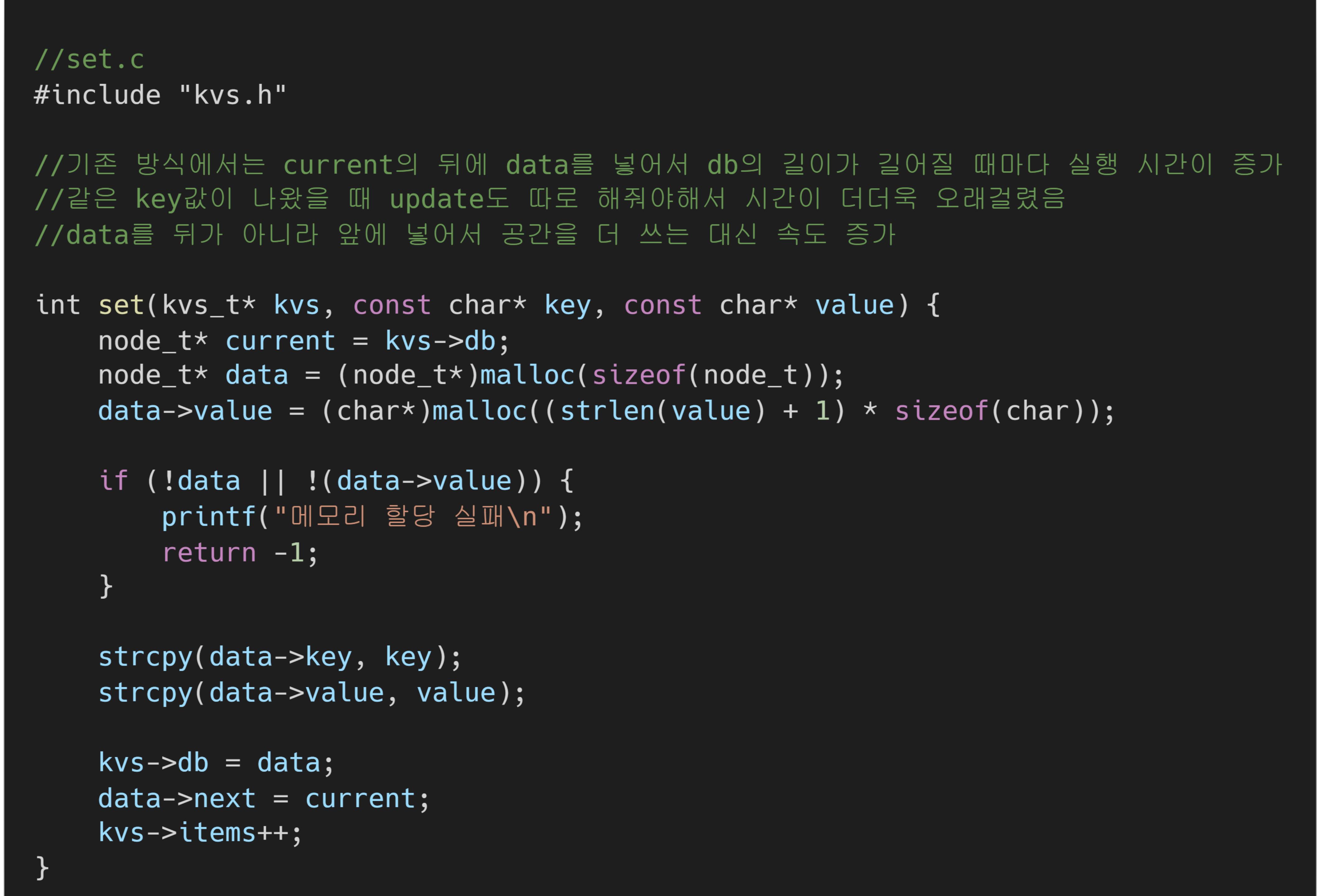


```c
    if (!query_fp || !answer_fp) {
        printf("파일 열기 실패\n");
        return -1;
    }

    char type[4];
    char key[100];
    char value[100];

    if(!feof(query_fp)) //%s\n으로 저장 시 엔터가 마지막 줄에 들어가서 첫 줄은 %s로 다음줄부터 \n%s로 저장하는 방식으로
    {
        fscanf(query_fp, "%3[^,],%99[^,],%99[^\n]\n", type, key, value);

        if (strcmp(type, "set") == 0) //type이 set이면 set을 호출
            set(kvs, key, value);
        else if (strcmp(type, "get") == 0) //type이 get이면 get을 호출
            fprintf(answer_fp, "%s", get(kvs, key));
    }

    while(!feof(query_fp)) {
        fscanf(query_fp, "%3[^,],%99[^,],%99[^\n]\n", type, key, value);

        if (strcmp(type, "set") == 0)
            set(kvs, key, value);
        else if (strcmp(type, "get") == 0)
            fprintf(answer_fp, "\n%s", get(kvs, key));
    }

    fclose(query_fp);
    fclose(answer_fp);
    close(kvs);

    return 0;
}
```

# open.c

```c
//open.c
#include "kvs.h"

kvs_t* open()
{
    kvs_t* kvs = (kvs_t*) malloc (sizeof(kvs_t));

    if(kvs)
        kvs->items = 0;
    printf("%d\n", kvs->items);

    return kvs;
}
```

## set.c



```
//set.c
#include "kvs.h"

//기존 방식에서는 current의 뒤에 data를 넣어서 db의 길이가 길어질 때마다 실행 시간이 증가
//같은 key값이 나왔을 때 update도 따로 해줘야해서 시간이 더더욱 오래걸렸음
//data를 뒤가 아니라 앞에 넣어서 공간을 더 쓰는 대신 속도 증가

int set(kvs_t* kvs, const char* key, const char* value) {
    node_t* current = kvs->db;
    node_t* data = (node_t*)malloc(sizeof(node_t));
    data->value = (char*)malloc((strlen(value) + 1) * sizeof(char));

    if (!data || !(data->value)) {
        printf("메모리 할당 실패\n");
        return -1;
    }

    strcpy(data->key, key);
    strcpy(data->value, value);

    kvs->db = data;
    data->next = current;
    kvs->items++;
}
```

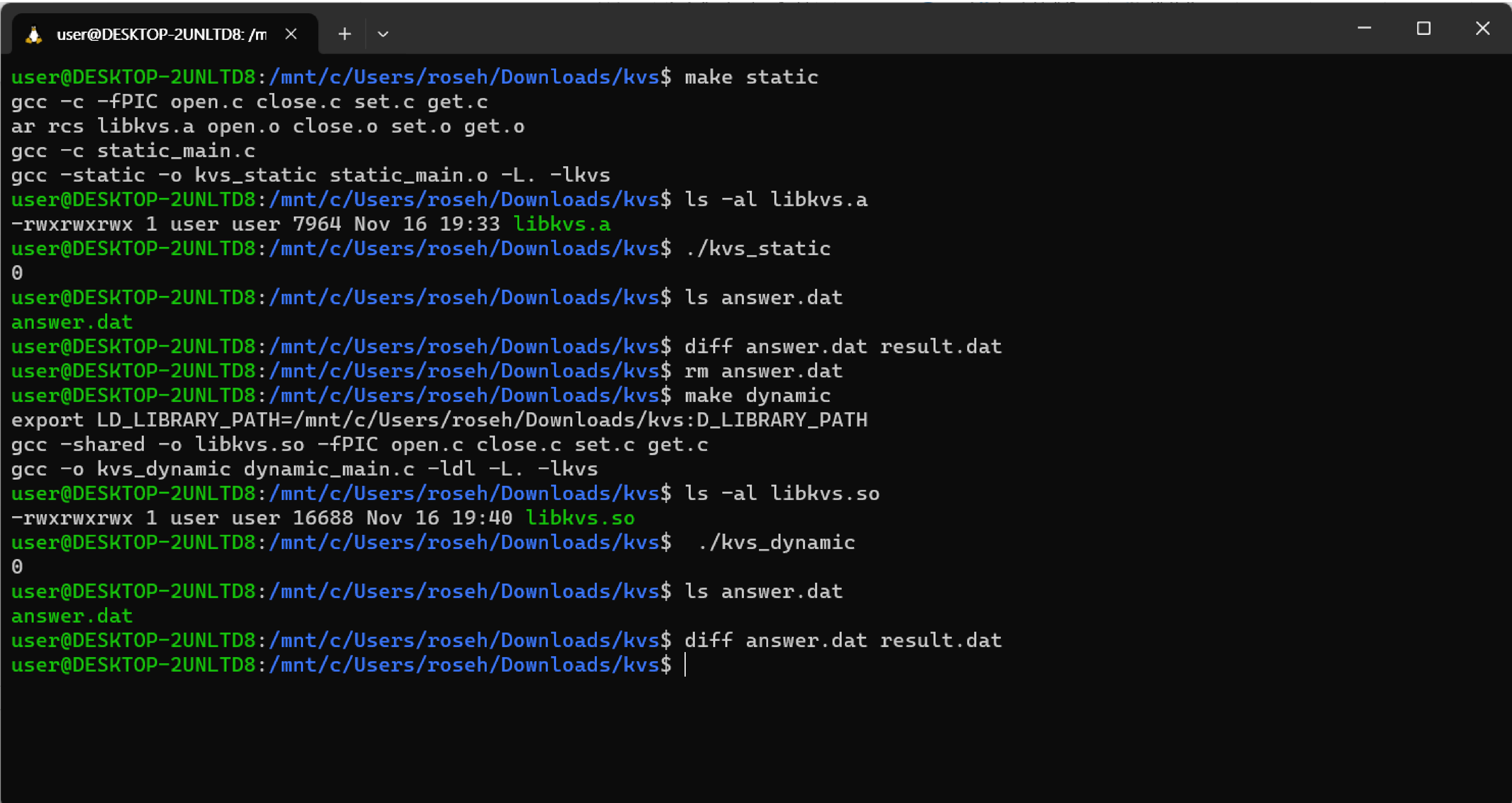
## get.c

```c
//get.c
#include "kvs.h"

char* get(kvs_t* kvs, const char* key)
{
    node_t* current = kvs->db;

    while(1) {
        if (current==NULL) //key를 찾지 못하고 맨 끝에 도달하면 -1을 리턴
            return "-1";
        else if (strcmp(current->key, key) == 0)
            break;
        current = current->next; //key값을 찾을 때 까지 db안을 탐색
    }

    char* value = (char*)malloc((strlen(current->value) + 1) * sizeof(char));

    if(!value){
        printf("Failed to malloc\n");
        return NULL;
    }

    strcpy(value, current->value); //value에 값을 복사해서 return
    return value;
}
```

close.c

```c
//close.c
#include "kvs.h"

int close(kvs_t* kvs) {
    node_t* current = kvs->db;

    while (current != NULL) { //노드를 끝까지 탐색하면 current에 할당된 메모리를 해제
        node_t* next = current->next;
        free(current->value);
        free(current);
        current = next;
    }
    kvs->db = NULL;
    kvs->items = 0;
    return 0;
}
```

## 실행화면



user@DESKTOP-2UNLTD8: /m

```
user@DESKTOP-2UNLTD8:/mnt/c/Users/roseh/Downloads/kvs$ make static
gcc -c -fPIC open.c close.c set.c get.c
ar rcs libkvs.a open.o close.o set.o get.o
gcc -c static_main.c
gcc -static -o kvs_static static_main.o -L. -lkvs
user@DESKTOP-2UNLTD8:/mnt/c/Users/roseh/Downloads/kvs$ ls -al libkvs.a
-rwxrwxrwx 1 user user 7964 Nov 16 19:33 libkvs.a
user@DESKTOP-2UNLTD8:/mnt/c/Users/roseh/Downloads/kvs$ ./kvs_static
0
user@DESKTOP-2UNLTD8:/mnt/c/Users/roseh/Downloads/kvs$ ls answer.dat
answer.dat
user@DESKTOP-2UNLTD8:/mnt/c/Users/roseh/Downloads/kvs$ diff answer.dat result.dat
user@DESKTOP-2UNLTD8:/mnt/c/Users/roseh/Downloads/kvs$ rm answer.dat
user@DESKTOP-2UNLTD8:/mnt/c/Users/roseh/Downloads/kvs$ make dynamic
export LD_LIBRARY_PATH=/mnt/c/Users/roseh/Downloads/kvs:D_LIBRARY_PATH
gcc -shared -o libkvs.so -fPIC open.c close.c set.c get.c
gcc -o kvs_dynamic dynamic_main.c -ldl -L. -lkvs
user@DESKTOP-2UNLTD8:/mnt/c/Users/roseh/Downloads/kvs$ ls -al libkvs.so
-rwxrwxrwx 1 user user 16688 Nov 16 19:40 libkvs.so
user@DESKTOP-2UNLTD8:/mnt/c/Users/roseh/Downloads/kvs$  ./kvs_dynamic
0
user@DESKTOP-2UNLTD8:/mnt/c/Users/roseh/Downloads/kvs$ ls answer.dat
answer.dat
user@DESKTOP-2UNLTD8:/mnt/c/Users/roseh/Downloads/kvs$ diff answer.dat result.dat
user@DESKTOP-2UNLTD8:/mnt/c/Users/roseh/Downloads/kvs$
```

## Makefile

```makefile
all:

static: static_lib
	gcc -c static_main.c
	gcc -static -o kvs_static static_main.o -L. -lkvs

static_lib: open.c close.c set.c get.c
	gcc -c -fPIC open.c close.c set.c get.c
	ar rcs libkvs.a open.o close.o set.o get.o

dynamic: dynamic_lib
	gcc -o kvs_dynamic dynamic_main.c -ldl -L. -lkvs

dynamic_lib:
	export LD_LIBRARY_PATH := $(shell pwd):$(LD_LIBRARY_PATH)
	gcc -shared -o libkvs.so -fPIC open.c close.c set.c get.c

clean:
	rm -f *.o kvs_static kvs_dynamic *.a *.so answer.dat
```