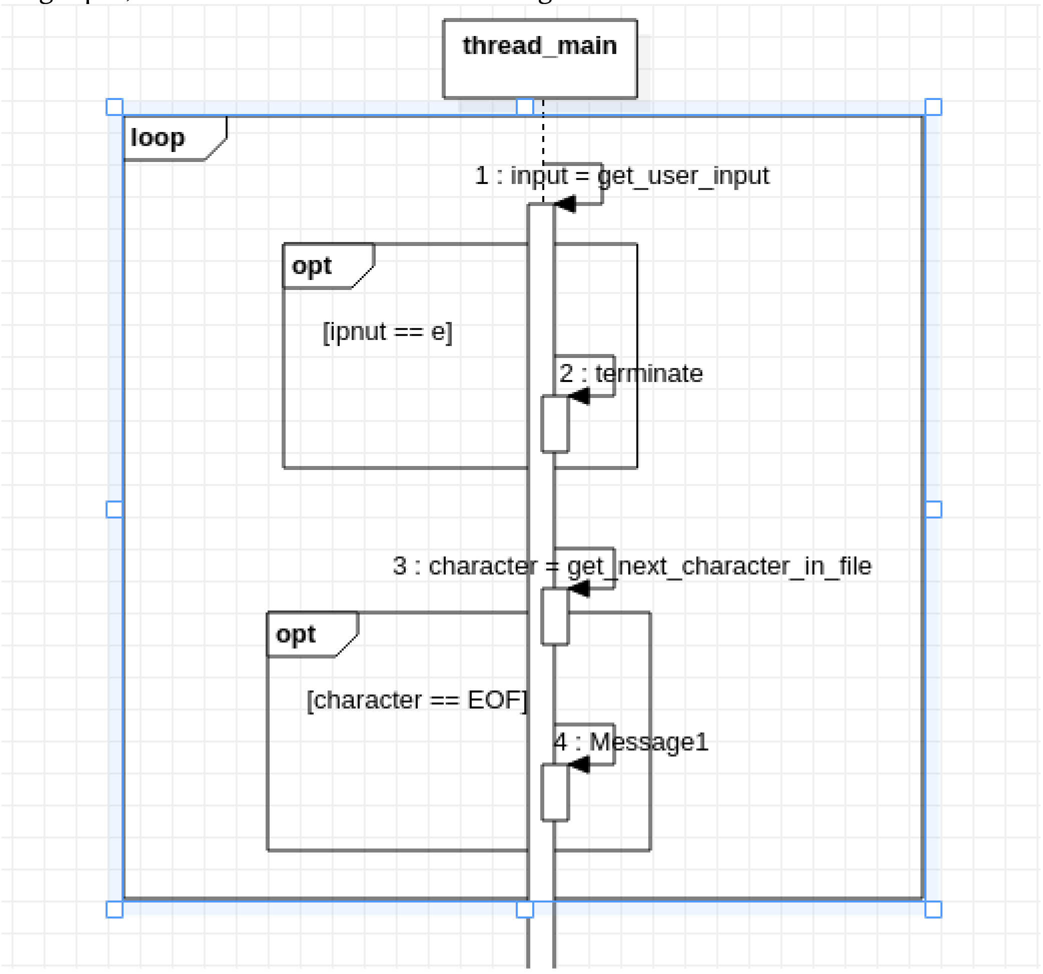
Create Responsive User Interaction With Threads

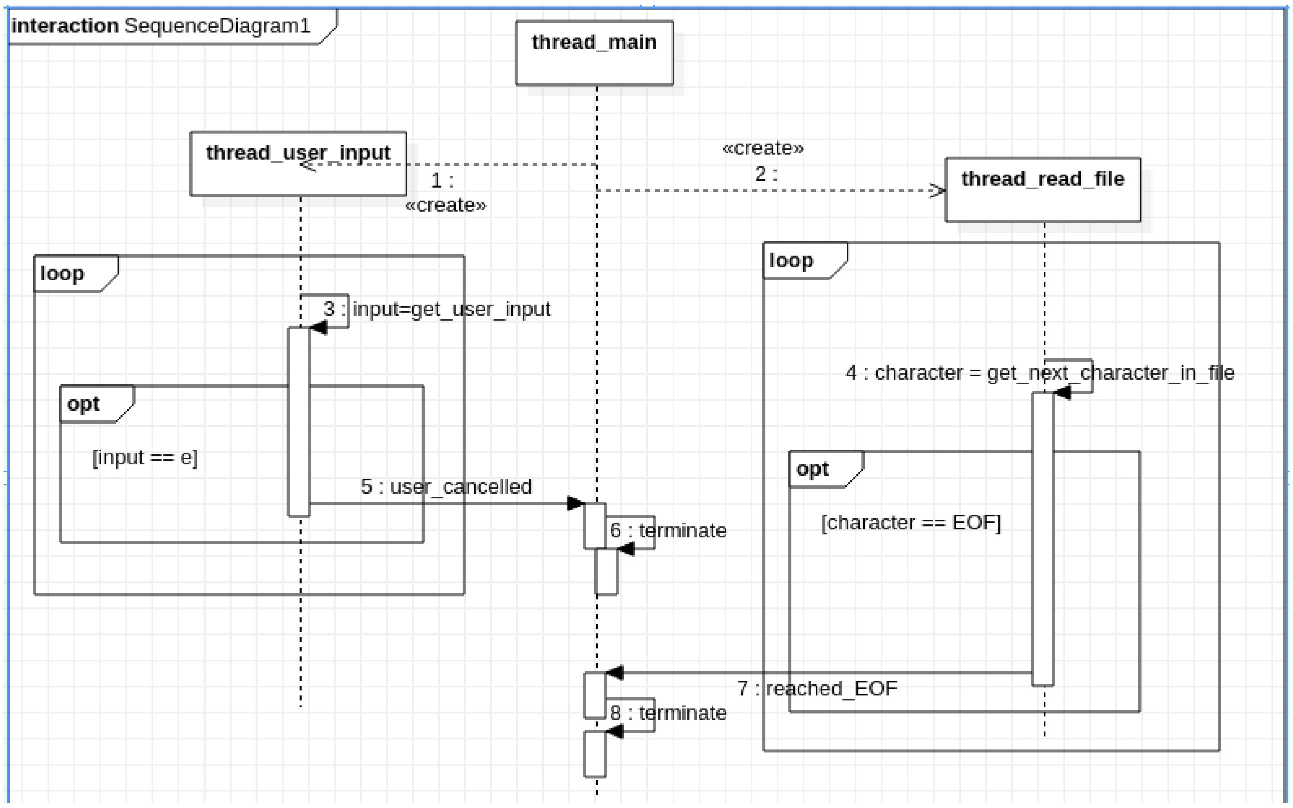
Handle User Input
&
Load File
Concurrently

What we are going to build?
A program that handle user input and read characters in file one by one.
While handling input, if user press 'e', we terminate program.
While reading file, if EOF is readed, we terminate program.

Single Thread Approach
If we get input and read file a single thread.
After handling input, user must wait a while for reading file task to continue.



Multiple Threads Approach
Placing these operations in a separate threads.

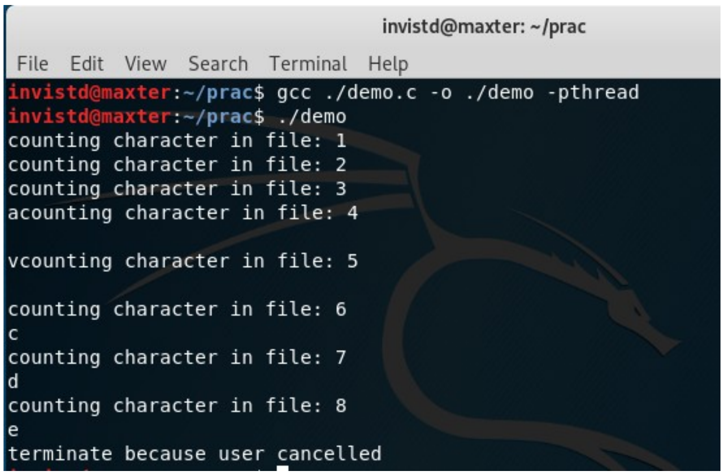


Code
```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>

pthread_mutex_t mtx = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;

int is_user_cancalled = 0;
int is_reach_EOF = 0;

void* handle_user_input(void* arg){
    while (1) {
        int c = getchar();

        if (c == 'e'){
            pthread_cond_signal(&cond);
            is_user_cancalled = 1;
            return NULL;
        }
    }
}

void cleanup_handler_read_file(void* arg){
    FILE* fp = (FILE*)arg;

    fclose(fp);
}
```

```
void* read_file(void* arg){
    FILE* fp = fopen("./test.txt", "r");

    pthread_cleanup_push(cleanup_handler_read_file, fp);

    char c;
    int count = 0;
    while ( (c = getc(fp)) != EOF){
        sleep(1);
        count ++;
        printf("counting character in file: %d\n", count);
    }

    pthread_cleanup_pop(1);

    is_reach_EOF = 1;
    pthread_cond_signal(&cond);
}

int main(){
    pthread_t thread_input, thread_file;

    pthread_create(&thread_input, NULL, handle_user_input, NULL);
    pthread_create(&thread_file, NULL, read_file, NULL);

    pthread_mutex_lock(&mtx);
    pthread_cond_wait(&cond, &mtx);
    pthread_mutex_unlock(&mtx);

    if (is_user_cancalled) {
        printf("terminate because user cancelled\n");
        pthread_cancel(thread_file);
    }

    if (is_reach_EOF){
        printf("terminate because EOF is reached\n");
        pthread_cancel(thread_input);
    }

    return 0;
}
```

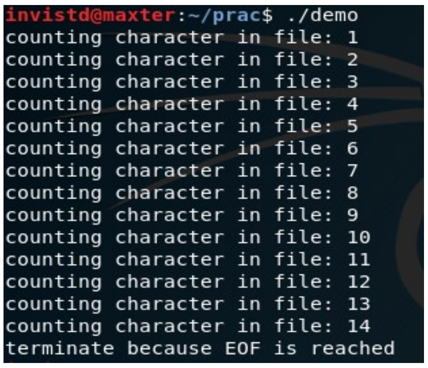
Result

Case 01:
User press ‘e’ to terminate application



```
invistd@maxter:~/prac$ gcc ./demo.c -o ./demo -pthread
invistd@maxter:~/prac$ ./demo
counting character in file: 1
counting character in file: 2
counting character in file: 3
acounting character in file: 4

vcounting character in file: 5

counting character in file: 6
c
counting character in file: 7
d
counting character in file: 8
e
terminate because user cancelled
```

Case 02:
Application is terminated because EOF is reached while reading file.



```
invistd@maxter:~/prac$ ./demo
counting character in file: 1
counting character in file: 2
counting character in file: 3
counting character in file: 4
counting character in file: 5
counting character in file: 6
counting character in file: 7
counting character in file: 8
counting character in file: 9
counting character in file: 10
counting character in file: 11
counting character in file: 12
counting character in file: 13
counting character in file: 14
terminate because EOF is reached
```