

Forum:[Hack Forums](#)

Thread: [C - Read pairs of integers from file](#)

Golden Kappa - December 26, 2020, 9:57 am

Hello guys, first off let me tell you I'm not experienced at all with C, so bare with me please. I'm helping a friend with one of his school C exercises, but we're stuck in one part, so I'm hoping some of you more advanced programmers can help us out. The exercise is basically making a program that can read a configuration file containing a map with the location of a series of mines, and it should provide the user with an interface to make changes to the map, like adding new mines or blow up an existing mine. Each position on the map is represented by 2 coordinates x and y. These coordinates are integers and they can assume values in the range [0, 24]. Each of the map's coordinates may be empty (without a mine) or have a mine, and each mine can only assume 2 states, either armed or off. The program should start by presenting the following menu to the user:

```
+-----  
read <filename>  - read input file  
show            - show the mine map  
trigger <x> <y>  - trigger mine at <x> <y>  
plant <x> <y>    - place armed mine at <x> <y>  
export <filename> - save file with current map  
quit           - exit program  
sos           - show menu  
+-----
```

The program then waits for user to input one of the commands from above. Now, the command we're having trouble with is the "read <filename>" one. When the user enters the read command, the program should read the file with the name "filename" and build the corresponding map. The input file will be a text-encoded file that should contain a set of pairs of coordinates that represent the locations of the mines in armed state. Mines in an off state or empty spaces are not represented in the file. Each coordinate is represented by a pair of integer values (X Y) separated by a blank space, X being the row number and Y the column number. There is no restriction regarding line breaks within the file or the order of each pair of coordinates within the file.

Example of input file:

```
Code 0 1  
0 2  
0 3  
5 9 6 4 6 5  
0 4  
0 5  
  
1 4 2 3 2 6 2 9  
3 4  
3 9 4 9  
5 2  
5 3  
  
5 4
```

6 9 7 6

7 2

8 3

8 6

The file must be read until the end of the file is detected. For each X coordinate represented in the file, there must be a Y coordinate. If this is not the case, it means that the file is poorly formatted. In this case, the program should show stderr stream this message: "File is corrupted".

With all that said, let me show you how far we've gotten with the piece of code we have for the read command:

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

```
#include <string.h>
```

```
#define SIZE 128
```

```
int main(void)
```

```
{
```

```
    char line[SIZE], file_name[SIZE], command[SIZE];
```

```
    int x[SIZE], y[SIZE], i = 0;
```

```
    while (1)
```

```
    {
```

```
        fgets(line, SIZE, stdin);
```

```
        if (!strncmp(line, "read", 4))
```

```
        {
```

```
            sscanf(line, "%s %s", command, file_name);
```

```
            FILE *fp = fopen(file_name, "r");
```

```
            while (fscanf(fp, "%d %d", &x[i], &y[i]) == 2)
```

```
            {
```

```
                printf("%d %d ", x[i], y[i]);
```

```
            }
```

```
            if (fp == NULL)
```

```
            {
```

```
                fputs("Couldn't open file.\n", stderr);
```

```
            }
```

```
        }
```

```
    }
```

```
}
```

We're not even sure if that's a good solution, but it is indeed working when the file is correctly formatted, as it prints out every pair of coordinates. What we don't know how to do is when the file is corrupted. Let's imagine a corrupted file like this:

```
Code 0 1
```

```
0 2
```

```
0 3
```

```
5 9 6 4 6 5
```

```
0 4
```

```
0 5
```

1 4 2 3 2 6 2
3 4
3 9 4 9
5
5

5 4
6 9 7 6
7 2
8 3
8 6

How are we supposed to get the "File is corrupted" message with an input file like the above? We have 3 single numbers, how can we catch them?

guardian - December 26, 2020, 5:38 pm

I am pretty tired, so I hope I got everything right

Have a look at [this](#)

By splitting each line and using a whitespace as the delimiter, also putting them all into an array afterwards, you could catch each line that only contains a single number, by simply checking whether the array contains more than one value, if the values are in the right format etc

Please let me know, if I got something wrong, as I said, I am pretty tired haha

cghhhhasd - December 27, 2020, 3:46 am

Quote:There is no restriction regarding line breaks within the file or the order of each pair of coordinates within the file.

Quote:How are we supposed to get the "File is corrupted" message with an input file like the above? We have 3 single numbers, how can we catch them?

This is your same file, re-formatted:

Code 0 1
0 2
0 3
5 9
6 4
6 5
0 4
0 5
1 4
2 3
2 6
2 3
4 3
9 4
9 5
5 5
4 6
9 7
6 7
2 8
3 8

6

Where are the 3 single numbers? There are 21 valid pairs and 1 invalid pair (Because it has no Y coordinate).

What I did in my code was I read the entire text file into memory, went through each one of its bytes and checked if it is between 0x30 and 0x39 (These are the numbers from 0 to 9 in the ASCII table). If it is, then I increase the number count. If the number count is even, it means we reached a new pair ("0 1 0 2 0 3 5" = "X Y X Y X Y X"), so we store the current number into X, otherwise it means we finished a pair so we will store it in Y.

At the end I check if the total number of numbers in the text file is uneven/odd, which means there is an extra number without a pair, given the requirements of how the program should work you mentioned in your post.

Here is the output of my code with that list above:

Code STATUS: 2 // (CORRUPT_COORDINATE_FILE)
FOUND: 21

0 1
0 2
0 3
5 9
6 4
6 5
0 4
0 5
1 4
2 3
2 6
2 3
4 3
9 4
9 5
5 5
4 6
9 7
6 7
2 8
3 8

My code, written for Windows:
Code #include <Windows.h>

```
typedef struct {  
    // ONLY 4 BITS USED FOR EACH COORDINATE. MAXIMUM NUMBER IT CAN HANDLE IS 15  
    // THIS LOWERS THE STRUCT SIZE BY HALF  
    unsigned int X : 0b1111;  
    unsigned int Y : 0b1111;  
} COORDINATE;  
  
#define IS_ANSI_INTEGER(X) (((X)>=(0x30))&&((X)<=(0x39)))  
#define TO_INTEGER(X) ((X)-(0x30))  
#define OTHER_ERROR 0
```

```
#define ALL_OK 1
#define CORRUPT_COORDINATE_FILE 2
```

```
unsigned char GET_COORDINATES(const unsigned short* FILE_PATH, COORDINATE**
FOUND_COORDINATES, unsigned int* NUMBER_OF_COORDINATES_FOUND) {
    if (!FILE_PATH || !*FILE_PATH || !FOUND_COORDINATES ||
!NUMBER_OF_COORDINATES_FOUND) {
        return OTHER_ERROR;
    }
    // DOES FILE EXIST?
    if (INVALID_FILE_ATTRIBUTES == GetFileAttributesW(FILE_PATH)) {
        return OTHER_ERROR;
    }
    // OPEN FILE
    void* FILE = CreateFileW(FILE_PATH, FILE_READ_ACCESS, FILE_SHARE_READ, 0,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
    if (!FILE) {
        return OTHER_ERROR;
    }
    // GET THE SIZE OF FILE IN BYTES
    unsigned char STATUS = OTHER_ERROR;
    unsigned int SIZE = GetFileSize(FILE, 0);
    if (!SIZE) {
        goto ABORT_WITH_FH;
    }
    // ALLOCATE MEMORY TO STORE ENTIRE FILE
    void* MEM = VirtualAlloc(0, SIZE, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
    if (!MEM) {
        goto ABORT_WITH_FH;
    }
    // READ ENTIRE FILE INTO MEMORY
    DWORD READ = 0;
    if (!ReadFile(FILE, MEM, SIZE, &READ, 0)) {
        goto ABORT_WITH_FH_MEM;
    }
    unsigned int INITIAL_COORDINATE_CAPACITY = 10;
    // IF SOME OTHER COORDINATES WERE PREVIOUSLY ALLOCATED, FREE THEM
    if (*FOUND_COORDINATES) {
        HeapFree(GetProcessHeap(), 0, *FOUND_COORDINATES);
        *FOUND_COORDINATES = 0;
    }
    // ALLOCATE HEAP MEMORY FOR 10 COORDINATES TO BEGIN WITH
    *FOUND_COORDINATES = HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY,
sizeof(COORDINATE) * INITIAL_COORDINATE_CAPACITY);
    if (!*FOUND_COORDINATES) {
        goto ABORT_WITH_FH_MEM;
    }
    COORDINATE CURRENT = { 0 };
    unsigned int NUMBER_COUNT = 0;
    // GO TROUGH EACH BYTE OF THE FILE
    for (unsigned char I = 0; I < SIZE; ++I) {
        // GET THE BYTE AT THE CURRENT INDEX OF THE FILE
        unsigned char C = ((unsigned char *)MEM)[I];
```

```

// IS THE CURRENT BYTE AN ASCII NUMBER?
if (IS_ANSI_INTEGER(C)) {
    // INCREASE THE TOTAL NUMBER OF NUMBERS FOUND
    NUMBER_COUNT += 1;
    // IF THE NUMBER COUNT IS EVEN (MEANING WE FINISHED ANOTHER PAIR OF XY
COORDINATES)
    if (!(NUMBER_COUNT % 2)) {
        // CONVERT THE CURRENT ASCII BYTE TO AN ACTUAL NUMBER
        CURRENT.Y = TO_INTEGER(C);
        // IF WE ARE ABOUT TO GO OVER THE CAPACITY OF OUR COORDINATE LIST,
ALLOCATE MEMORY FOR 10 MORE COORDINATES
        if (INITIAL_COORDINATE_CAPACITY == (NUMBER_COUNT / 2)) {
            INITIAL_COORDINATE_CAPACITY += 10;
            void* TEMP = HeapReAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY,
*FOUND_COORDINATES, sizeof(COORDINATE) * INITIAL_COORDINATE_CAPACITY);
            if (!TEMP) {
                goto ABORT_WITH_FH_MEM_HEAP;
            }
            *FOUND_COORDINATES = (COORDINATE*)TEMP;
        }
        // SAVE THE CURRENT COORDINATE
        *((COORDINATE*)((unsigned int)*FOUND_COORDINATES +
(*NUMBER_OF_COORDINATES_FOUND * sizeof(COORDINATE))) = CURRENT;
        // INCREASE THE NUMBER OF FOUND COORDINATES
        *NUMBER_OF_COORDINATES_FOUND += 1;
    }
    // THE NUMBER COUNT IS UNEVEN/ODD, MEANS WE REACHED A NEW PAIR OF XY
COORDINATES
    else {
        CURRENT.X = TO_INTEGER(C);
    }
}
}

// IF THE TOTAL NUMBER COUNT IS UNEVEN/ODD, IT MEANS THERE WERE EXTRA
NUMBERS IN THE FILE
if (NUMBER_COUNT % 2) {
    STATUS = CORRUPT_COORDINATE_FILE;
    // CLOSE FILE, FREE MEMORY ALLOCATED FOR FILE DATA
    goto ABORT_WITH_FH_MEM;
}
STATUS = ALL_OK;
// CLOSE FILE, FREE MEMORY ALLOCATED FOR FILE DATA
goto ABORT_WITH_FH_MEM;
// FREE ALLOCATED HEAP MEMORY FOR COORDINATES
ABORT_WITH_FH_MEM_HEAP:
    HeapFree(GetProcessHeap(), 0, *FOUND_COORDINATES);
    // FREE MEMORY ALLOCATED FOR FILE DATA
ABORT_WITH_FH_MEM:
    VirtualFree(MEM, 0, MEM_RELEASE);
    // CLOSE FILE
ABORT_WITH_FH:
    CloseHandle(FILE);
    return STATUS;

```

```

}

int main()
{
    COORDINATE* C = 0;
    unsigned int N = 0;
    unsigned char STATUS = GET_COORDINATES(L"C:\\Users\\User\\Desktop\\a.txt", &C, &N);
    printf("STATUS: %X\nFOUND: %d\n\n", STATUS, N);
    for (unsigned int I = 0; I < N; ++I) {
        printf("%d %d\n", C[I].X, C[I].Y);
    }
    getchar();
    return 0;
}

```

saltycracker - December 27, 2020, 9:23 pm

(Dec 26, 2020 10:57 AM)Yawnerd Wrote: [\[CITE\]](#) How are we supposed to get the "File is corrupted" message with an input file like the above? We have 3 single numbers, how can we catch them?

Pro-tip.. When you post a question, post just the question. We don't all the other info. We just need the stuff for the question.

Example:

You have a ascii text file with pairs of numbers. Give example of text file.

You need to read pairs of numbers. Note what is a valid pair.

You need to indicate an error if single numbers(non-pairs) occur. Note what a non-pair number is.

To answer your question.. scanf family of functions return(quote from my manpages)

Quote:On success, these functions return the number of input items successfully matched and assigned; this can be fewer than provided for, or even zero, in the event of an early matching failure.

The value EOF is returned if the end of input is reached before either the first successful conversion or a matching failure occurs. EOF is also returned if a read error occurs, in which case the error indicator for the stream (see `ferror(3)`) is set, and `errno` is set to indicate the error.