# C Review, PT II (+READMEs, File Issues)

*Note: C Review DOES NOT mean I will actively go over 1310 topics. I will simply be doing programs in C at the level you are expected to be at after taking 1310*

**Before we code, a couple of notes:**

## README files:

See sample README file
It should let the user know how to run your program
You should turn in READMEs with your HWs so the TA knows how to run your program

- Omega? VM?
- Inputs it can handle?
- etc

## Possible issues reading files:

- **EOL conversions (end of line)**
  - UNIX (like Linux) uses LF, Windows uses CRLF, Mac uses CR
    - CR-carriage return
      - \r, 13 on the ASCII table
    - LF-line feed
      - \n, 10 on the ASCII table
    - CRLF
      - \r\n
  - You may run into issues when reading files due to different endings on the line
  - See below for one way to handle this
    - We will convert the file to Unix EOL (\n for everything-no \r)
  - *Note: If you are a computer science major, you will learn more about different operating systems in future courses*
    - *Even if you are not a computer science major, you can still look it up ☺*

Notice below an issue when reading the file (for example, the file input “running into each other”):

```
C computer$ ./a.out
Welcome to Maggiano's
Pizza 1!
Pizza 2!
Pizza 3!
Large
Red Sauce
Parmesan
Veggies
25.50

Medium
White Sauce
Parmesan
Meat
18.25

Small
Red Sauce
Veggiesn Notice here
14.92
```

pizzastore1.txt

```
Maggiano's
Large
Red Sauce
Parmesan
Veggies
25.50
Medium
White Sauce
Parmesan
Meat
18.25
Small
Red Sauce
Parmesan
Veggies
14.92
```

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

void read_file(char filename[], char pizza_store[3][5][40])
{
   FILE* fp=fopen(filename,"r");
   char line[40];
   int pizza_counter=0;
   int i=0,j=0;
   char letter;

   if(!fp)
   {
    printf("File didn't open.");
   }

   else
   {

    fgets(line,40,fp);
    printf("Welcome to %s",line);

    while(i<3)
    {
     printf("Pizza %d!\n",(i+1));
     for(j=0;j<5;j++)
     {
      fgets(pizza_store[i][j],40,fp); /*it looks like the issue is here*/
      printf("%s", pizza_store[i][j]);
     }
     i++;
    }
    fclose(fp);
   }
}

int main(int argc, char **argv)
{
 char pizza_store[3][5][40];
 read_file("pizzastore1.txt", pizza_store);
}
```

Notice how we can fix this by doing the following:

```
C computer$ cat pizzastore1.txt |tr '\r' '\n' | tr -s '\n' > updatedfile.txt
C computer$ gcc practice.c
C computer$ ./a.out
Welcome to Maggiano's
Pizza 1!
Large
Red Sauce
Parmesan
Veggies
25.50
Pizza 2!
Medium
White Sauce
Parmesan
Meat
18.25
Pizza 3!
Small
Red Sauce
Parmesan
Veggies
```

What this line is doing is basically taking the file pizzastore1.c and creating a new file (without the EOL issues mentioned above). It gets rid of the \r and \n and only has \n:

```
cat pizzastore1.txt |tr '\r' '\n' | tr -s '\n' > updatedfile.txt
```

- tr (translate) is a command, just like cat and ls (the official name for all these commands is *command line utility*)

- tr is being used to modify the lines of the file. See more about it (you can also look up additional resources):
  - *https://www.geeksforgeeks.org/tr-command-in-unix-linux-with-examples/*

- At the end, a new file is output with these updates (updatedfile.txt in the above example). The original file is still intact

You can actually see before and after (I added a print statement). Note I'm using a Mac so on a Windows, you might have different output):

*(not using the line above)*

```
Welcome to Maggiano's
Pizza 1!
Large
n here: 5
Red Sauce
n here: 9
Parmesan
n here: 8
Veggies
n here: 7
25.50
n here: 5
Pizza 2!
Medium
n here: 6
White Sauce
n here: 11
Parmesan
n here: 8
Meat
n here: 4
18.25
n here: 5
Pizza 3!
Small
n here: 5
Red Sauce
n here: 9
Veggiesn
r here: 8 r here-issues here
n here: 16
14.92
```

*(using the line above-updated file)*

```
Welcome to Maggiano's
Pizza 1!
Large
n here: 5
Red Sauce
n here: 9
Parmesan
n here: 8
Veggies
n here: 7
25.50
n here: 5
Pizza 2!
Medium
n here: 6
White Sauce
n here: 11
Parmesan
n here: 8
Meat
n here: 4
18.25
n here: 5
Pizza 3!
Small
n here: 5
Red Sauce
n here: 9
Parmesan
n here: 8
Veggies
n here: 7  no r here
```

*(added print statement)*

```
while(k<strlen(pizza_store[i][j]))
{
        if(pizza_store[i][j][k]=='\r')
        {
          printf("r here: %d\n",k);
        }

        if(pizza_store[i][j][k]=='\n')
        {
          printf("n here: %d\n",k);
        }
        k++;
}
```

Note 1: I will show you different ways to read in file info:

```
while(!feof(fp))
{
    fgets(line,40,fp);
    printf("%s", line);
}
```

This goes until we hit the end of the file. Each iteration takes a line from the file after checking if we hit the end of the file

```
while(fgets(line,40,fp))
{
        printf("%s", line);
}
```

*this can also be written:*
*fgets(line,40,fp)!=NULL*

This keeps taking lines from the file until have no more lines to take. Note that we are taking lines from the file AND checking with fgets (unlike the separate actions shown with feof)

```
fscanf(fp, "%s",  line)

while(fscanf("%s, line)!=EOF)
{
        printf("%s", line);
 }
```

Note 2: to get specific lines of info, we can strategically place fgets (and fscanf):

```
while(!feof(fp))
{
   fgets(line,40,fp);
   fgets(line,40,fp);
}
```

*the first fgets gets one line, the second fgets gets a second line… so with every iteration we are getting two lines. You can save the values wherever you want.*

```
while(fgets(line,40,fp))
{
    fgets(line,40,fp);
}
```

*this also gets two lines*

Note you can strategically use fgets to get file lines in all kinds of ways…for example, checking if a line starts with a specific first letter or only printing out every third line etc. You can also do something similar with fscanf.

# Program 1:

Create a program that reads the total number of people on a train.

```
Computers-MacBook-Air:C computer$ cat train2.txt |tr '\r' '\n' | tr -s '\n' > updatedtrain.txt
Computers-MacBook-Air:C computer$ gcc practice.c
Computers-MacBook-Air:C computer$ ./a.out updatedtrain.txt output.txt
Exiting...
```

Input file (originally train2.txt, cleaned it up and created updatedtrain.txt-notice to our eyes it still looks the same…we clean it up for the computer):

updatedtrain.txt

```
First
5
Business
30
Economy
50
```

train2.txt

```
First
5
Business
30
Economy
50
```

Output file:

output.txt

```
--Total people on this train:
1. 5
2. 30
3. 50
Total=85
```

*Note: the two ways I get input from files in class are using fgets (you saw that last time) and using fscanf (you'll see that below)*

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

void get_user_input(char message[], char answer[])
{
 printf("%s",message);
 scanf("%s",answer);
}

/*this function returns 0 if the file does not open and 1 if it does*/
int get_train_info(char filename[], int all_people[])
{
 char line[3][40];
 FILE* fp=fopen(filename,"r");

 if(!fp)
 {
  printf("File not opened.\n");
  return 0;
 }
```

```
  else
  {
   fscanf(fp,"%s %d %s %d %s %d ",line[0],&all_people[0],line[1],&all_people[1],line[2],&all_people[2]);
   fclose(fp);
   return 1;
  }
}

/*this function outputs a file in the format given above*/
void write_file(char filename[],int all_people[])
{
  int i,total=0;
  FILE* fp=fopen(filename,"w");
  fprintf(fp,"--Total people on this train: \n");

  for(i=0;i<3;i++)
  {
   fprintf(fp,"%d. %d\n",i+1,all_people[i]);
   total+=all_people[i];
  }

  fprintf(fp,"Total=%d",total);
}


int main(int argc, char** argv)
{

  int all_people[3];
  int n=get_train_info(argv[1], all_people); /*For right now, I just want you to have working knowledge of argv…we will learn more about what it actually is when we start learning about pointers*/

  if(n)
  {
   write_file(argv[2],all_people); /*this will output to the current folder you are running the program from.  If you want, you could specify another location using a path-something like "/Users/Computer/Desktop/Teach/Examples/output.txt"*/

C computer$ ./a.out updatedtrain.txt /Users/Computer/Desktop/output.txt

  }

  printf("Exiting...\n");
}
```

# Program 2:

Your city has a specific lotto scheme where if you guess the winning numbers in order, you win a billion dollars.  Winning numbers are kept in a file and if you successfully guess the numbers in order you get added to a list of winners.

```
(base) Computers-MacBook-Air:C computer$ ./a.out winningnumbers1.txt winnerlist.txt
exit or play? play
Alright! Ready to play?  Enter 6 numbers.
1. Enter a number: 45
```

```
---Match! lotto number: 45, your pick: 45.

2. Enter a number: 89
---NOT a match! lotto number: 12 your pick: 89.

Sorry, can't win them all.  Have a good day though!

exit or play? play
Alright! Ready to play?  Enter 6 numbers.
1. Enter a number: 45
---Match! lotto number: 45, your pick: 45.

2. Enter a number: 12
---Match! lotto number: 12, your pick: 12.

3. Enter a number: 1
---Match! lotto number: 1, your pick: 1.

4. Enter a number: 33
---Match! lotto number: 33, your pick: 33.

5. Enter a number: 8
---Match! lotto number: 8, your pick: 8.

6. Enter a number: 19
---Match! lotto number: 19, your pick: 19.

Wow! You won a billion dollars! Congrats!!! :)
Enter your name to be added to the winners list: Felisha
All winners so far:
1. Felisha
****1 billion spent by the lotto commission! Play for your chance to win today!***

exit or play? exit
Thanks for playing!  Bye!
(base) Computers-MacBook-Air:C computer$ ./a.out winningnumbers1.txt winnerlist.txt
exit or play? play
Alright! Ready to play?  Enter 6 numbers.
1. Enter a number: 45
---Match! lotto number: 45, your pick: 45.

2. Enter a number: 12
---Match! lotto number: 12, your pick: 12.

3. Enter a number: 1
---Match! lotto number: 1, your pick: 1.

4. Enter a number: 33
---Match! lotto number: 33, your pick: 33.

5. Enter a number: 8
---Match! lotto number: 8, your pick: 8.

6. Enter a number: 19
---Match! lotto number: 19, your pick: 19.

Wow! You won a billion dollars! Congrats!!! :)
Enter your name to be added to the winners list: Petey
All winners so far:
1. Felisha
2. Petey
****2 billion spent by the lotto commission! Play for your chance to win today!***

exit or play? exit
Thanks for playing!  Bye!
```

*Note 1: I used the same file of winning numbers for both runs above.  I could have changed the winning number file for the second run (meaning I would have to guess different numbers) and the program would still work...in fact, it would be more realistic since the second lotto run should be a different set of numbers*

*Note 2: Notice the third argument (the winners list file name) is the same in both runs. Since I am opening in the append mode, I'm simply adding any new name to same file. For the first run, the file doesn't exist- append mode creates the file if it doesn't exist (first run) and just adds to it if it does (any run afterwards using the same file name). This is different from write mode where opening the same file would overwrite any previous file info.*

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#define BUFFER 40

int get_winning_numbers(char filename[], int winning_nums[])
{
 FILE* fp=fopen(filename,"r");
 char line[BUFFER];
 int i=0;

 if(!fp)
 {
  printf("File not opened.\n");
  return 0;
 }

 else
 {
  while(fgets(line,BUFFER,fp)) /*see note 1*/
  {
   fgets(line,BUFFER,fp); /*see note 2*/
   winning_nums[i]=atoi(line);
   i++;
  }
  fclose(fp);  /*close the file connection*/
  return 1;
 }
}


/*return 0 means win, return 1 means lose*/
int play_lotto(int winning_nums[])
{
 char line[BUFFER];
 int i, wrong=0;
 printf("Alright! Ready to play?  Enter 6 numbers. \n");

 for(i=0;i<6 && !wrong;i++)
 {
  printf("%d. Enter a number: ", (i+1));
  fgets(line,BUFFER,stdin); /*I'm using fgets for the whole program-no scanf*/
  if(atoi(line)!=winning_nums[i])
```

```
   {
      printf("---NOT a match! lotto number: %d your pick: %d.\n\n",winning_nums[i],atoi(line));
      wrong=1;
   }
   else
   {
      printf("---Match! lotto number: %d, your pick: %d.\n\n",winning_nums[i],atoi(line));
   }

  }

  if(!wrong)
  {
   printf("Wow! You won a billion dollars! Congrats!!! :)\n");
  }

  else
  {
   printf("Sorry, can't win them all.  Have a good day though!\n\n");
  }

  return wrong;
}

void winners_list(char filename[])
{
  int i=1,total_billions=0;
  char line[BUFFER];
  FILE* fp=fopen(filename,"a+");

  printf("Enter your name to be added to the winners list: ");
  fgets(line,BUFFER,stdin);
  fprintf(fp,"%s",line);

  rewind(fp); /*go back to the beginning of the file since we are currently at the end …can also use FSEEK/SEEKSET…see next program for notes on this*/

  printf("All winners so far:\n");
   while(fgets(line,BUFFER,fp)) /*see note 1*/
   {
    printf("%d. %s",i,line);
    i++;
    total_billions++; /*keeping track of the number of winners to know how many total billions spent*/
   }

   fclose(fp);
   printf("****%d billion spent by the lotto commission! Play for your chance to win today!***\n\n",
total_billions);

}
```

```
int main(int argc, char** argv)
{
  int continue_game=1;
  char answer[BUFFER];
  int winning_nums[6];
  int check=get_winning_numbers(argv[1], winning_nums);

  if(check)
  {
    while(continue_game)
    {
      printf("exit or play? ");
      fgets(answer,BUFFER,stdin); /*im using fgets for the whole program*/

      if(!strcmp(answer,"play\n")) /*not checking for capital letters…also including \n since fgets add it…we could have also cut it off*/
      {
        check=play_lotto(winning_nums);

        if(!check) /*winner means we returned 0, so !0 is true*/
        {
          winners_list(argv[2]);
        }

      }

      else if(!strcmp(answer,"exit\n"))
      {
        printf("Thanks for playing!  Bye!\n");
        continue_game=0;
      }

      else
      {
        printf("Unknown response.\n");
      }

    }


  }

  else
  {
    printf("...Exiting...\n");
  }
}
```

```
(base) Computers-Air:C computer$ gcc practice.c
(base) Computers-Air:C computer$ ./a.out

Enter list name:
wordlist

***All words currently:

Do you want to add add a word? y or n
y
Add word to the list: dog
Adding: dog

***All words currently:
dog

Exiting program...
(base) Computers-Air:C computer$ gcc practice.c
(base) Computers-Air:C computer$ ./a.out

Enter list name:
wordlist

***All words currently:
dog

Do you want to add add a word? y or n
y
Add word to the list: cat
Adding: cat

***All words currently:
dog
cat

Exiting program...
```

```c
#include <stdio.h>
#include <string.h>
#define SIZE 40


void add(FILE *fp)
{
  char word[100];

  printf("Add word to the list: ");
  scanf("%s", word),
  printf("Adding: %s\n", word);
  fprintf(fp, " %s", word);
}

void print_words(FILE *fp)
{
  printf("\n***All words currently:\n");

  char line[100];

  while(fscanf(fp," %s", line)!=EOF)
  {
    printf("%s\n", line);
```

```
  }

  //can also do
  // while (fgets(line, 100, fp))
  // {
  //  printf("%s", line);
  // }
}


int main (int argc, char **argv)
{
  char line[100];
  printf("\nEnter list name:\n");
  scanf("%s", line);

  strcat(line, ".txt"); /*putting together list name and .txt to make a file name-
every time, we are opening the file name*/

  FILE* fp=fopen(line, "a+");

  if(!fp)
  {
    printf("No file opened.\n");
  }
  else
  {
    fseek(fp, 0, SEEK_SET); /*start file pointer at beginning of file so we can print
the whole song (you can also use rewind like I did below)*/

    print_words(fp);
    printf("\nDo you want to add add a word? y or n\n");
    scanf("%s", line);

    if(!strcmp(line, "y")) /*add, can also compare line[0]=='y'*/
    {
      add(fp); /*file pointer is already pointing at end since we printed out all
above (and fp moved along to the end to do this), so we are not overwriting*/
    }

    rewind(fp); /*instead of  fseek(fp, 0, SEEK_SET) you can use either*/
    print_words(fp);
    printf("\nExiting program...\n");
    fclose(fp);
  }
}
```