RLE Compression
Project 3

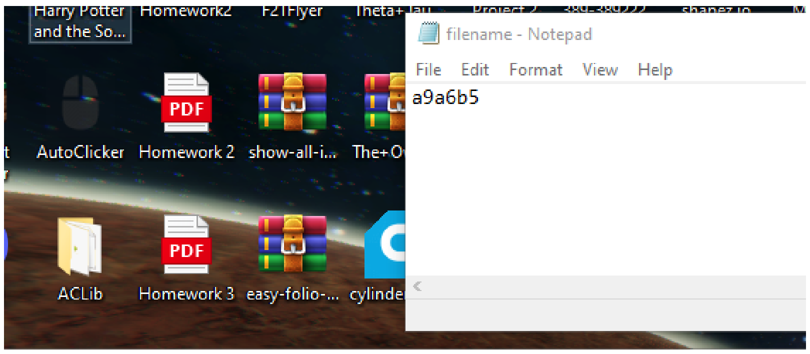
By
Alenn Wright

Overview

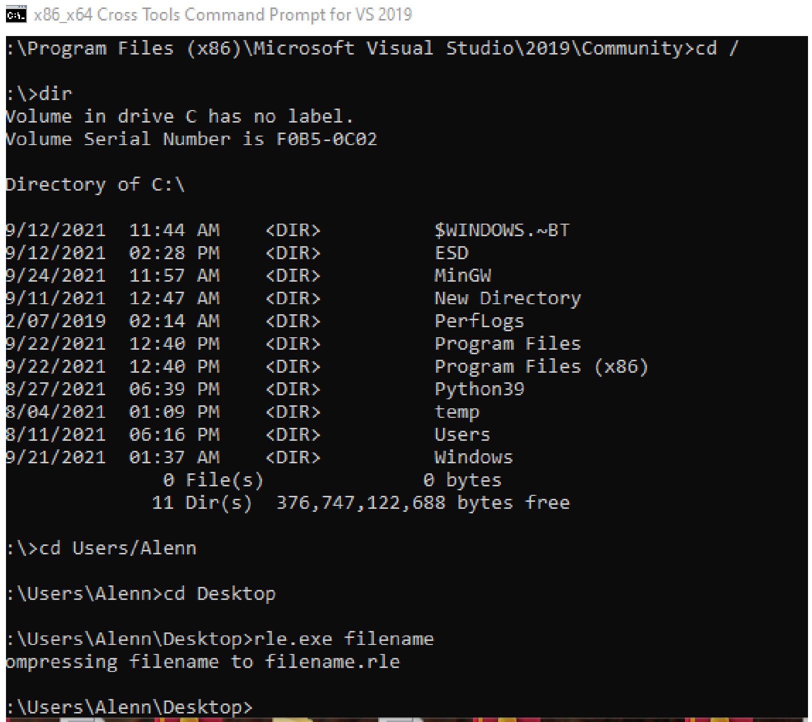
For this project we are tasked with making a rle compression tool. We are given a skeleton code that will compress a file by taking an uncompressed file and taking the characters and printing the number of instances after it, that it occurs and outputs it into a file that is the same name but an rle extension. The other half of this project is to decompress the rle files into a file of the same name but with an extension of plain. The decompressed file will ideally be the same as the original file.

Sample out

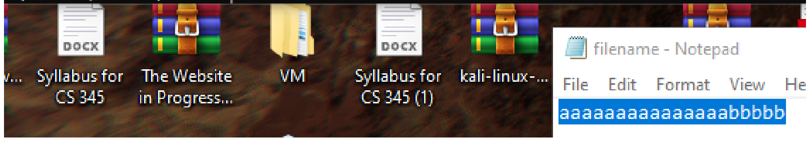
What the compression input should look like is “aaaaaaaaaaaaaabbbbbb” and output should be “a9a5b6”. Then the decompression should reverse the compression.
The input file I used was just named “filename” containing “aaaaaaaaaaaaaaabbbbb”.
Outputting “a9a6b5”.



```
a9a6b5
```

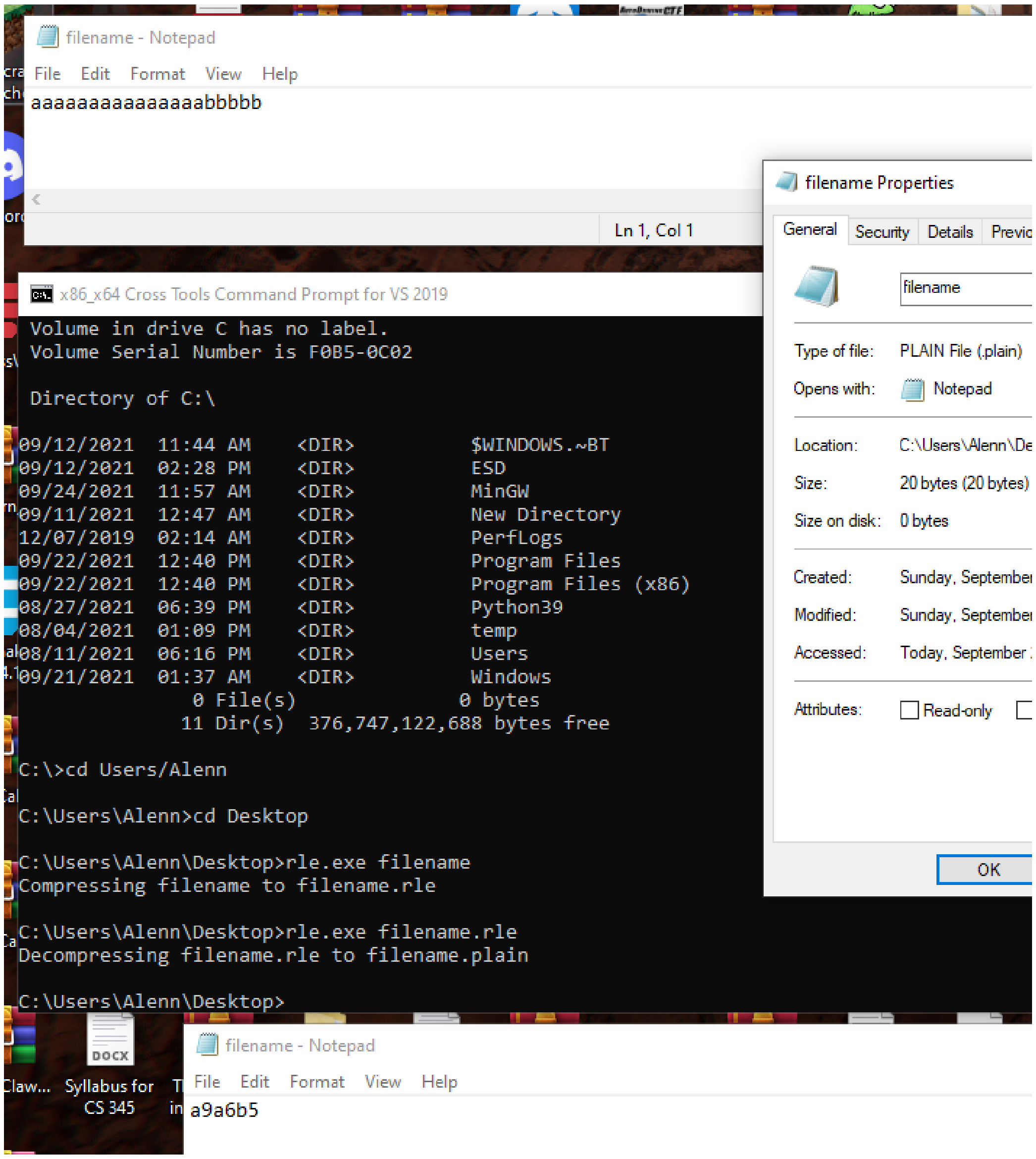


```
:\Program Files (x86)\Microsoft Visual Studio\2019\Community>cd /

:\>dir
Volume in drive C has no label.
Volume Serial Number is F0B5-0C02

Directory of C:\

9/12/2021  11:44 AM    <DIR>          $WINDOWS.~BT
9/12/2021  02:28 PM    <DIR>          ESD
9/24/2021  11:57 AM    <DIR>          MinGW
9/11/2021  12:47 AM    <DIR>          New Directory
2/07/2019  02:14 AM    <DIR>          PerfLogs
9/22/2021  12:40 PM    <DIR>          Program Files
9/22/2021  12:40 PM    <DIR>          Program Files (x86)
8/27/2021  06:39 PM    <DIR>          Python39
8/04/2021  01:09 PM    <DIR>          temp
8/11/2021  06:16 PM    <DIR>          Users
9/21/2021  01:37 AM    <DIR>          Windows
              0 File(s)              0 bytes
             11 Dir(s)  376,747,122,688 bytes free

:\>cd Users/Alenn

:\Users\Alenn>cd Desktop

:\Users\Alenn\Desktop>rle.exe filename
ompressing filename to filename.rle

:\Users\Alenn\Desktop>
```



```
aaaaaaaaaaaaaaabbbbb
```

Then the decompression I pushed the output file into it to decompress it.


filename - Notepad
File Edit Format View Help
Ln 1, Col 1
filename Properties
General Security Details Previo
OK
x86_x64 Cross Tools Command Prompt for VS 2019
Syllabus for CS 345

```
aaaaaaaaaaaaaaabbbbb
```

Type of file: PLAIN File (.plain)
Opens with: Notepad
Location: C:\Users\Alenn\De
Size: 20 bytes (20 bytes)
Size on disk: 0 bytes
Created: Sunday, September
Modified: Sunday, September
Accessed: Today, September
Attributes: Read-only

```
 Volume in drive C has no label.
 Volume Serial Number is F0B5-0C02

 Directory of C:\

09/12/2021  11:44 AM    <DIR>          $WINDOWS.~BT
09/12/2021  02:28 PM    <DIR>          ESD
09/24/2021  11:57 AM    <DIR>          MinGW
09/11/2021  12:47 AM    <DIR>          New Directory
12/07/2019  02:14 AM    <DIR>          PerfLogs
09/22/2021  12:40 PM    <DIR>          Program Files
09/22/2021  12:40 PM    <DIR>          Program Files (x86)
08/27/2021  06:39 PM    <DIR>          Python39
08/04/2021  01:09 PM    <DIR>          temp
08/11/2021  06:16 PM    <DIR>          Users
09/21/2021  01:37 AM    <DIR>          Windows
               0 File(s)              0 bytes
              11 Dir(s)  376,747,122,688 bytes free

C:\>cd Users/Alenn

C:\Users\Alenn>cd Desktop

C:\Users\Alenn\Desktop>rle.exe filename
Compressing filename to filename.rle

C:\Users\Alenn\Desktop>rle.exe filename.rle
Decompressing filename.rle to filename.plain

C:\Users\Alenn\Desktop>
```

```
a9a6b5
```

Source code

The source code below is based on the skeleton code supplied by us, I just had to code the compress and decompress functions.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void compress(char* data, int count, FILE* outfile);
void decompress(char* data, int count, FILE* outfile);
char* readFileData(char* filename, int* count_ptr);
int
```

```
main(int num_args, char* arg_values[])
{
   if (num_args != 2)
   {
      printf("Usage: rle filename (produces filename.rle)\n");
      printf(" rle filename.rle (produces filename.plain)\n");
      exit(1);
   }
   char* input_filename = arg_values[1];
   // read the file data into an array
   int count;
   char* data = readFileData(input_filename, &count);
   // Call compress() or decompress().
   FILE* outfile;
   int len = strlen(input_filename);
   if (len < 4 || strcmp(input_filename + (len - 4), ".rle") != 0)
   {
      char output_filename[80];
      strcpy(output_filename, input_filename);
      strcat(output_filename, ".rle");
      printf("Compressing %s to %s\n", input_filename, output_filename);
      outfile = fopen(output_filename, "wb");
      compress(data, count, outfile);
   }
   else
   {
      char output_filename[80];
      strncpy(output_filename, input_filename, len - 4);
      output_filename[len - 4] = 0;
      strcat(output_filename, ".plain");
      printf("Decompressing %s to %s\n", input_filename, output_filename);
      outfile = fopen(output_filename, "wb");
      decompress(data, count, outfile);
   }
   // Close the output file to ensure data is saved.
   fclose(outfile);
   // Free the array we allocated
   delete data;
   return 0;
}

void
compress(char* data, int count, FILE* outfile)
{
    // TODO: compress the data instead of just writing it out to the file
    // uses html ascii
    char previous;
    int intchar_offset = 48;
    int counter = 0;
    for (int i = 0; i < count; ++i)
    {
        counter += 1;
        if(data[i] == data[i+1])
        {
            //continue counting
        }
        else
        {
            while(counter > 9)
            {
                putc(data[i], outfile);
                putc((char)9+intchar_offset, outfile);
                counter -= 9;
            }
            putc(data[i], outfile);
            putc((char)counter+intchar_offset, outfile);
            counter = 0;
        }
    }
}

void
decompress(char* data, int count, FILE* outfile)
{
    // TODO: decompress the data instead of just writing it out to the file
    int intchar_offset = 48;
    for (int i = 0; i < count; ++i)
    {
        for (int c = 0; c < ((int)data[i + 1])-intchar_offset; ++c)
        {
            putc(data[i], outfile);
        }
        i += 1;
    }
}

char*
readFileData(char* filename, int* count_ptr)
```

```
{
   // Returns a pointer to an array storing the file data.
   // Sets the variable pointed to by 'count' to contain the file size.
   // Exits the program if the filename doesn't exist.
   FILE* infile = fopen(filename, "rb");
   if (!infile)
   {
      printf("No such file \"%s\"!\n", filename);
      exit(1);
   }
   // Get file size by going to the end of the file, getting the
   // position, and then going back to the start of the file.
   fseek(infile, 0, SEEK_END);
   int count = ftell(infile);
   fseek(infile, 0, SEEK_SET);
   // read the data from the file
   char* data = new char[count];
   fread(data, 1, count, infile);
   fclose(infile);
   *count_ptr = count;
   return data;
}
```

Conclusion

During this project, I had a lot of issues with my C compiler and found that my version of windows had an issue with the compiler so I had to upgrade a compatible version and it ran then. Additionally it was a fun experience since I usually code in linux making scripts that I call on to run, however, it's the first time I have run a script with a file condition in line.