Project 2 :

Unix Utilities

Summary

The program my-cat.c is a simple program that reads a file as specified by the user and prints its contents.

The program my-grep.c looks through a file, line by line, trying to find a user-specified search term in the line. If a line has the word within it, the line is printed out, otherwise it is not.

The program my-zip.c is a compression tool. When you encounter n characters of the same type in a row, my-zip will turn that into the number n and a single instance of the character. The program my-upzip.c is a decompression tool. It does the reverse of the my-zip tool, taking in a compressed file and writing (to standard output again) the uncompressed results.

My-cat

The my-cat program reads the file and prints out its contents. The following details have been taken into account :

1. The program my-cat can be invoked with one or more files on the command line; it should just print out each file in turn.
2. In all non-error cases, my-cat should exit with status code 0, usually by returning a 0 from main() (or by calling exit(0)).
3. If no files are specified on the command line, my-cat should just exit and return 0.
4. If the program tries to fopen() a file and fails, it should print the exact message "my-cat: cannot open file" (followed by a newline) and exit with status code 1.
    - I tested this with a file named non_existent.txt that didn't exist.
5. If multiple files are specified on the command line, the files should be printed out in order until the end of the file list is reached or an error opening a file is reached (at which point the error message is printed and my-cat exits).
    - I tested this with 2 textfiles and 1 file that didn't exist in 2 orders :
        - file_1 non_existent.txt file_2
        - file_1 file_2 non_existent.txt

```
vagrant@precise32:/vagrant/Linux/project_2$ gcc -o my-cat my-cat.c -Wall -Werror
vagrant@precise32:/vagrant/Linux/project_2$ ./my-cat
vagrant@precise32:/vagrant/Linux/project_2$ ./my-cat non_existent.txt
my-cat: cannot open file
vagrant@precise32:/vagrant/Linux/project_2$ ./my-cat textfile1.txt
We're gonna go our way
To places still unknown
We're gonna show the way
We made it on our own
We do what we wanna do
The message through our music
It ain't even done, it's true
Yeah, our rhythm never stops

vagrant@precise32:/vagrant/Linux/project_2$ ./my-cat textfile1.txt textfile2.txt
We're gonna go our way
To places still unknown
We're gonna show the way
We made it on our own
We do what we wanna do
The message through our music
It ain't even done, it's true
Yeah, our rhythm never stops

Stray Kids still gonna rock
On the Hellevator, yeah, we head to the top (Stray Kids)
Stray Kids, run 'til we're done
Do whatever we want, yeah, we don't give a what (Stray Kids)
Know that this is who we are
There ain't no last step out, oh, we'll never stop
Stray Kids still gonna rock
On the Hellevator, yeah, we head to the top, go on and on


vagrant@precise32:/vagrant/Linux/project_2$ ./my-cat textfile1.txt non_existent.txt textfile2.txt
We're gonna go our way
To places still unknown
We're gonna show the way
We made it on our own
We do what we wanna do
The message through our music
It ain't even done, it's true
Yeah, our rhythm never stops

my-cat: cannot open file
vagrant@precise32:/vagrant/Linux/project_2$ ./my-cat textfile1.txt textfile2.txt non_existent.txt
We're gonna go our way
To places still unknown
We're gonna show the way
We made it on our own
We do what we wanna do
The message through our music
It ain't even done, it's true
Yeah, our rhythm never stops

Stray Kids still gonna rock
On the Hellevator, yeah, we head to the top (Stray Kids)
Stray Kids, run 'til we're done
Do whatever we want, yeah, we don't give a what (Stray Kids)
Know that this is who we are
There ain't no last step out, oh, we'll never stop
Stray Kids still gonna rock
On the Hellevator, yeah, we head to the top, go on and on

my-cat: cannot open file
```

My-grep

The my-grep program looks through a file, line by line, trying to find a user-specified search term in the line. The following details have been taken into account :

1. The program my-grep is always passed a search term and zero or more files to grep through (thus, more than one is possible). It should go through each line and see if the search term is in it; if so, the line should be printed, and if not, the line should be skipped.
2. The matching is case sensitive. Thus, if searching for foo, lines with Foo will not match.
3. Lines can be arbitrarily long (that is, you may see many many characters before you encounter a newline character, \n). my-grep should work as expected even with very long lines. For this, you might want to look into the getline() library call (instead of fgets()), or roll your own.
4. If my-grep is passed no command-line arguments, it should print "my-grep: searchterm [file ...]" (followed by a newline) and exit with status 1.
5. If my-grep encounters a file that it cannot open, it should print "my-grep: cannot open file" (followed by a newline) and exit with status 1.
    - I tested this with a file named non_existent.txt that didn't exist.
6. In all other cases, my-grep should exit with return code 0.
7. If a search term, but no file, is specified, my-grep should work, but instead of reading from a file, my-grep should read from standard input.
    - I tested this with the searchterm foo. The program immediately prints the foo line to the screen when pressed space bar.
8. For simplicity, if passed the empty string as a search string, my-grep can either match NO lines or match ALL lines, both are acceptable.

```
vagrant@precise32:/vagrant/Linux/project_2$ gcc -o my-grep my-grep.c -Wall -Werror
vagrant@precise32:/vagrant/Linux/project_2$ ./my-grep
my-grep: searchterm [file ...]
vagrant@precise32:/vagrant/Linux/project_2$ ./my-grep foo non_existent.txt
my-grep: cannot open file
vagrant@precise32:/vagrant/Linux/project_2$ cat bar.txt
this line has foo in it

so does this foolish line; do you see where?

even this line, which has barfood in it, will be printed.

vagrant@precise32:/vagrant/Linux/project_2$ ./my-grep foo bar.txt
this line has foo in it
so does this foolish line; do you see where?
even this line, which has barfood in it, will be printed.
vagrant@precise32:/vagrant/Linux/project_2$ ./my-grep Foo bar.txt
vagrant@precise32:/vagrant/Linux/project_2$ ./my-grep we textfile1.txt textfile2.txt
We do what we wanna do
On the Hellevator, yeah, we head to the top (Stray Kids)
Stray Kids, run 'til we're done
Do whatever we want, yeah, we don't give a what (Stray Kids)
Know that this is who we are
There ain't no last step out, oh, we'll never stop
On the Hellevator, yeah, we head to the top, go on and on
```

```
vagrant@precise32:/vagrant/Linux/project_2$ ./my-grep foo
It's Christmas Eve Foo
Jack Frost coming for y'all, run run
Christmas EveL foo
Christmas EveL foo
Merry Christmas
Jingle, jingle, jingle all the way
You got me spinning
I don't need no presents foo
I don't need no presents foo
Crossing out X-MAS
Feliz Navidad (ho, ho, ho) Foo
Feliz Navidad (it's Christmas Eve)
```

My-zip & my-unzip

The my-zip program will compress characters of the same type in a row into a single instance of the character. The my-unzip program will do the reverse and write the uncompressed results. The following details have been taken into account :

1. Correct invocation should pass one or more files via the command line to the program; if no files are specified, the program should exit with return code 1 and print "my-zip: file1 [file2 ...]" (followed by a newline) or "my-unzip: file1 [file2 ...]" (followed by a newline) for my-zip and my-unzip respectively.
2. The format of the compressed file must match the description above exactly (a 4-byte integer followed by a character for each run).
3. Do note that if multiple files are passed to *my-zip, they are compressed into a single compressed output, and when unzipped, will turn into a single uncompressed stream of text (thus, the information that multiple files were originally input into my-zip is lost). The same thing holds for my-unzip.

```
vagrant@precise32:/vagrant/Linux/project_2$ gcc -o my-zip my-zip.c -Wall -Werror
vagrant@precise32:/vagrant/Linux/project_2$ gcc -o my-unzip my-unzip.c -Wall -Werror
vagrant@precise32:/vagrant/Linux/project_2$ ./my-zip file.txt > file.z
vagrant@precise32:/vagrant/Linux/project_2$ ./my-unzip file.z
aaaaaaaaaabbbb
vagrant@precise32:/vagrant/Linux/project_2$ ./my-zip
my-zip: file1 [file2 ...]
vagrant@precise32:/vagrant/Linux/project_2$ ./my-unzip
my-unzip: file1 [file2 ...]
vagrant@precise32:/vagrant/Linux/project_2$ ./my-zip non_existent.txt
my-zip: cannot open file
vagrant@precise32:/vagrant/Linux/project_2$ ./my-unzip non_existent.z
my-unzip: cannot open file

vagrant@precise32:/vagrant/Linux/project_2$ cat file1.txt
aaaa
vagrant@precise32:/vagrant/Linux/project_2$ cat file2.txt
bbbb
vagrant@precise32:/vagrant/Linux/project_2$ ./my-zip file1.txt file2.txt > multi.z
vagrant@precise32:/vagrant/Linux/project_2$ ./my-unzip multi.z
aaaa
bbbb
vagrant@precise32:/vagrant/Linux/project_2$ cat mixed.txt
aaaabbbbccdddddde

vagrant@precise32:/vagrant/Linux/project_2$ ./my-zip mixed.txt > mixed.z
vagrant@precise32:/vagrant/Linux/project_2$ ./my-unzip mixed.z
aaaabbbbccdddddde
```

Sources

I used the following sources while doing this project :

- ➔ My code exercises from the course Basics of C-programming
- ➔ https://stackoverflow.com/questions/46211465/c-language-cat-program-that-opens-file-from-user-input
- ➔ https://www.geeksforgeeks.org/build-your-own-cat-command-in-c-for-linux/
- ➔ https://stackoverflow.com/questions/30714247/how-to-grep-a-string-in-a-program
- ➔ https://www.quora.com/How-do-you-write-a-C-program-to-mimic-grep-command-in-Linux
- ➔ https://www.geeksforgeeks.org/cpp-program-to-read-and-print-all-files-from-a-zip-file/
- ➔ https://youtu.be/MENLyuc1tcg?si=rtH8Ef2OtOE-WmTS
- ➔ https://stackoverflow.com/questions/10440113/simple-way-to-unzip-a-zip-file-using-zlib