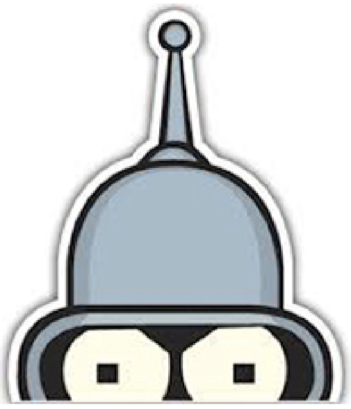
B1 - Elementary Programming in C

B-CPE-110

Antman

Itsy Bitsy Bytes



1.0

Antman

binary name: antman, giantman
language: C
compilation: via Makefile, including re, clean and fclean rules

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

The goal of this project is to compress data. You will write two programs.
The first one will take a file as input and compress it.
The second one will take a compressed file as input and translate it back to its original state.

Turn-in methods

You must have 2 folders at the root of your repository: a folder named **antman** with a binary named **antman** in it, and a folder named **giantman** with binary named **giantman** in it.
Each folder must contain a Makefile compiling the corresponding binary and a Makefile at the root must compile all binaries.

BOTH binaries must be functional for you to be graded on this project. We cannot evaluate one without the other.

Authorized functions

The only system calls allowed are the following ones:
- open
- read
- write
- close
- malloc
- free
- stat

ANTMAN

Your antman binary is the one that will compress files. It will take two parameters. The first one is the path to the file we want compressed. The second one is a number corresponding to the type of file it is (see ‘Lossless Compression’).
Your program will then print a compressed version of the file on the standard output. It should be possible to redirect the compressed data to a file.

```
Terminal
∼/B-CPE-110> cat file.txt
Maiha hi Maiha hou Maiha ha Maiha ha ha
```

```
Terminal
∼/B-CPE-110> ./antman/antman file.txt 1
Maiha@hi@hou@ha@121314144
```

This is an exemple of a possible implementation of a compression algorithm. YOUR ANTMAN OUTPUT CAN LOOK DIFFERENTLY.

GIANTMAN

Your giantman binary is the one that will recover files from their compressed format. It will take two parameters. The first one is the path of a file containing compressed data. The second one is a number corresponding to the type of file it originaly was.
Your program will then print the readable file the compressed data was based of on the standard output.

```
Terminal
~/B-CPE-110> cat file.txt
Maiha hi Maiha hou Maiha ha Maiha ha ha
```

```
Terminal
~/B-CPE-110> ./antman/antman file.txt 1 > compressed.data
```

```
Terminal
~/B-CPE-110> ./giantman/giantman compressed.data 1
Maiha hi Maiha hou Maiha hi Maiha ha ha
```

This is an exemple of how your program will be tested. Again, YOUR ANTMAN OUTPUT CAN LOOK HOWEVER YOU WANT.

Lossless Compression

The type of compression we want is lossless compression. If we execute the following command:

```
Terminal
~/B-CPE-110> ./antman/antman file.txt 1 > compressed.data ; ./giantman/giantman
compressed.data 1 > uncompressed.data
```

the "file.txt" and "uncompressed.data" files should be strictly identical.

We are dealing with 3 different types of input files:

1 corresponds to text files containing song lyrics. They are series of words, spaces, punctuation and new-lines. Other characters can appear but are not common.
2 corresponds to HTML files. They can contain any character.
3 corresponds to P3 PPM images.

We have given you a few exemple files but feel free to add your own.

We are not asking you to judge whether the input file is valid. The fileformat parameter is a strong and reliable hint but your program should attempt compression regardless of the file content.

Think hard about what this means for your program. Those file types will allow for very different techniques and tricks to archieve compression

You only need to read from the given files. Meaning that your programs should work even if you don't have the permissions to write on the files.

YOUR GOALS

MUST

- The antman binary **must** write a compressed version of the file it got as a parameter on the standard output
- The giantman binary **must** write the original version of the compressed file it got as a parameter on the standard output
- The original file printed by giantman **must** be the **exact** same as the one that antman got as parameter
- The programs **must** work with files containing **any** value
- The programs **must** compress files sometimes
- The programs **mustn't** take an unreasonable time to run, even on large files.
- The programs **must** work with files that you only have permission to read from

SHOULD

- The compressed file **shouldn't** be bigger than the original file.
- The programs **should** take into account the given filetype.
- The programs **should** compress files everytime.

COULD

- You **could** define new and/or more precise filetypes and use them
- You **could** push a quantified analysis of your different compression algorithms as a bonus
- You **could** propose new challenges for your peers and organise a competition
- The programs **could** gather informations from the file itself (extension, size) and use them as indicators