Intro to Computer Systems :: Project 0 :: Programming Best Practices

Assignment

Write, document, and package a program to remove white space and comments from a text file. You will use this functionality in several programming assignments in this course. You may program in the language of your choosing. Popular choices include java, python, c, and c++, but you may use any language you are comfortable with.

Packaging, portability, ease of build, and documentation of your program is the main emphasis of this assignment, and this is reflected in the grading rubric:

30% Your program is cleanly packaged and organized
25% There are clear instructions on how to compile and run your program
35% Portability of your program
10% Program functionality

Program Specification

User interface

- From the command line, your executable should take a filename ("filename.in") as input and output should be written to "filename.out".

- "filename" should not be hardcoded but rather taken as an argument. You can assume that it will end with ".in". For example:
  ```
  $ your-executable testfile.in
  ```

- "filename.out" should be written to the directory where "filename.in" resides. Don't assume that "filename".in resides in the same directory as your executable.
  ```
  $ your-executable some-path/testfile.in
  ```

Functionality

1) Remove white space:
   - Strip out all white space from "filename.in" and write everything else to "filename.out"
   - White space in this case means spaces, tabs, and blank lines, but not line returns.

Functionality (continued)

2) Remove comments:
   - If the argument "no-comments" is specified by the user, then remove all comments in addition to the whitespace. For example:
     ```
     $ your-executable testfile.in no-comments
     ```
   - Assume that comments begin with the sequence "//" and end at the line return.
   - If "no-comments" is not specified, don't worry about preserving the whitespace within the comment.

Packaging and Documentation

- Your program's source code and documentation should be zip or tar compressed into a single file, for example:
  `your-name-Project0.tar`
- Upon decompressing your assignment, one should see a `README.txt` and a `src/` directory.
- The `README.txt` should include
  - specific instructions on how to compile your code
  - specific instructions on how to run your code
  - a description of what works and what doesn't work in your code
- The `src/` directory should contain your source code
- Your source code should be documented. *Please document your code as you write it, not as an afterthought.*

Portability

- Assume your code will be compiled and run outside of your build environment. It should not rely on libraries, native compilers, or operating system hooks specific to your build environment.
- You should test that your code compiles and works outside of your build environment, ideally on a different operating system. You can use the lab machines or the machines available via shell access for testing.

Example

Assume that your executable is called `stripWhiteSpace`, and you have a text file named `myProg.in` that looks like this:

```
// Draws a rectangle at the top-left corner of the screen.
// The rectangle is 16 pixels wide and R0 pixels high.

(KBDLOOP)
   @KBD         // loop until key pressed
   D=M
   @KBDLOOP
   D;JEQ

   @50          // setup: rect will be 50 high
   D=A
   @R0
```

After the command `stripWhiteSpace myProg.in`
there should exist a file named `myProg.out` that looks like this:

```
//Drawsarectangleatthetop-leftcornerofthescreen.
//Therectangleis16pixelswideandR0pixelshigh.
(KBDLOOP)
@KBD//loopuntilkeypressed
D=M
@KBDLOOP
D;JEQ
@50//setup:rectwillbe50high
D=A
@R0
```

After the command `stripWhiteSpace no-comments myProg.in`
there should exist a file named `myProg.out` that looks like this:

```
(KBDLOOP)
@KBD
D=M
@KBDLOOP
D;JEQ
@50
D=A
@R0
```