COS 350: Program #1: myexpand [-t number]

Objectives: This first programming assignment is to get used to the tools used to edit and compile C programs and to write a small Unix utility in C.

You will write a simplified version of the Unix utility "expand". Its purpose is to take a text file and expand the TAB characters into the appropriate number of spaces. By default TAB characters expand to as many spaces as needed to reach the next column position at a multiple of 8 characters. Some programs however, such as the Eclipse, use different conventions. Eclipse uses a TAB spacing of 4 columns. This causes a problem when you try to use these files with other programs. For instance viewing the file with **more** will show tab spacing of 8 columns. Expand has a command line argument to specify the desired tabstop spacing.

Your program will act as a filter. That means that it does not directly open or close files, but rather it takes its input from standard input and sends its output to standard output. To fix the problem with an Eclipse file as explained above, you could use the following command:

myexpand -t 4 < file.java

The file redirection part of this command "< file.java" sends contents of file.java into the standard input for myexpand. myexpand also gets the two command line arguments "-t" and "4" which tells it to use tabstops every 4 columns, it reformats the text using spaces and then prints it to standard output. Try out the unix **expand** program to see how it works.

Suggestions:

- For input use: **int getchar(void);**
  This function returns one character at a time from the standard input stream. The type integer is used because there is one additional special value that can be returned, EOF, which is used to indicate the end of the file.

- For outut use: **int putchar(int c);**
  This function sends one character to the standard output stream. Do not send the EOF. When your program exits, the output stream will be closed.

- This is a small program. Focus on writing robust code. Think about any possible problems or errors that might occur, and try to give useful feedback and appropriate behavior. Error messages should not use printf(), but instead be sent to stderr. Use: **fprintf(stderr,** "msg"**)**;

- If in doubt about what the program should do, look at what the unix **expand** program does.

- While testing your code, you might want to expand tabs to a series of periods so that you can see what is happening.

Input file:
You can find a test input file (tabtest.txt) in the course directory: **/usr/class/cos350/prog1**

(OVER)

What to turn in:
For each assignment you will turn in a written report through Brightspace as well as submitting your code electronically from a COS linux computer.

Written report:

- Your report should be a single document. I expect a well organized and formatted report. Please do not use black backgrounds, small fonts, or difficult to read colors like yellow.
- You can use unix tools cat and emacs to prepare your report as a text file. Unix does not care about file extensions but BrightSpace does, so use the extension .txt on your report file. You can also use Microsoft word and submit the .doc or .docx file. If you use any other word processor, export your file as a .pdf because BrightSpace can only display these formats.
- Include in your report:
  1. Your name(s), and if you worked as a team say who did the electronic submit.
  2. A discussion of any incomplete parts, or any known bugs in your code.
  3. A listing of your code. I expect it to be correctly formatted and commented. Please avoid line wrapping.
  4. Sample results from your program. Show the results from 3 sample runs of your program:
     1. myexpand –t 3 < tabtest.txt — *if your program is working correctly, you should be able to read a hidden message*
     2. myexpand < tabtest.txt — *this checks the default case*
     3. myexpand 3 < tabtest.txt — *this checks handling of a command line error*
- To gather the results, I suggest that you use the unix program script. Script captures everything that you type and everything that any programs print to the terminal.
  1. At the unix command line run script
  2. Run your test cases.
  3. When you are done with your testing session type exit or press Ctrl-D.
  4. The transcript of your session is put in the file typescript
  5. You can look at it to see if it worked with more typescript
  6. Unfortunately everything that you type, including backspaces, is recorded. If you typed any special characters, these will sometimes make a mess when viewed or printed. There is a small program called scriptCleaner which is a filter that cleans up a typescript file. Use this command: scriptCleaner typescript > results.txt

Electronic submission:

- Before you submit your program, compile and test it on a linux machine in the lab.
- From a machine in the lab run the program "submit" to submit your files.
- Submit your source code (myexpand.c) and your executable (myexpand) to a directory named: **prog1**
- **Do not in any way combine, compress, zip, or tar your files!**

Grading:

- 40 points: correct operation for standard case
- 10 points: default case and error handing
- Programs not meeting the style guidelines will be returned ungraded.