CSc 453: Programming Assignment 1 (html2txt): Part 2

CSc 453: Programming Assignment 1 (html2txt): Part 2

Start Date: Tue Sept 8, 2015

Due Date: 11:59 PM, Mon Sept 14, 2015

---

1. General

This assignment involves writing a scanner and parser for HTML that enforce some simple grammar rules, e.g., that the **<li>** tag for list items can occur only within lists, or that tags specifying boldface **<b>**... **</b>** and italics **<i>**... **</i>**should be properly nested.

As with part 1 of this assignment, the goal is to get you sufficiently acquainted with **lex** and **yacc** that you can start the main compiler project. For this reason, the grammar rules used in this assignment cover only a very small part of the complete HTML syntax.

2.1. Functionality

Your program should read its input from **stdin**, ensure that the input follows the grammar rules for our subset of HTML, discard all HTML tags, and write the remaining text to **stdout**. Error messages (see below) should be written to **stderr**.

An example is shown here.

I have placed my implementation of this program as an executable **myhtml2txt** in the directory **/home/cs453/fall15/assignments/html2txt-2/**. If this program behaves differently than this specification, then the program is buggy. If you find bugs in this code, please let me know.

2.2. Conflicts

The HTML grammar provided may generate conflicts when you translate it to a YACC input file. If this happens (it may or may not), for this assignment you are not required to remove these conflicts. However, if this happens you should figure out why the conflict(s) are occurring, so that you can determine whether the default action taken by your parser is appropriate.

2.3. Syntax Errors

Your program will be expected to deal with errors in a “reasonable” way. Error messages should be printed to **stderr**. They should be specific and should contain enough information (with at least a line number) to allow the user to locate the problems.

Since we have not yet discussed error recovery, you are not *required* to recover from syntax errors: it is OK for your program to exit after detecting the first syntax error. However, if you choose to implement error recovery, that is OK too.

3. Invoking Your Program

Your executable program will be called **myhtml2txt**. It will read input from **stdin** and write its output to **stdout**. Thus, to translate an HTML file **foo.html** to a text file **bar.txt**, invoke your program as

myhtml2txt < foo.html > bar.txt

4. Turnin

Turn in your files on host **lectura.cs.arizona.edu**. You should turn in all of your source files, as well as a Makefile that supports the following targets:

clean Executing the command ***make clean*** should delete the ***.o** files, as well as the executable **myhtml2txt**, from the current directory.

myhtml2txt Executing the command ***make myhtml2txt*** should create, in the current directory, an executable file **myhtml2txt** that implements your HTML-to-text translator from scratch, by invoking the appropriate tools (**lex/flex**) on the input specifications.

To turn in your files, use the command

> **turnin cs453f15-html2txt-part2** *files*

For more information on the **turnin** command, try **man turnin**. **Note:** The **turnin** command copies the files submitted into another directory. Because of this, programs that compile and execute without problems in your directory may not work correctly once they are turned in, because of problems with relative path names in include files and make files. Such problems are considered to be sloppiness inappropriate in an upper division course, and are liable to be penalized heavily.

The output of your program will be compared with our output using **diff** utility (see **diff**(1)). With the exception of error messages (where the requirements are given above), your output must follow the specification exactly. For this reason it is recommended that you follow the specification, and instructions for turnin, closely.