Advanced Unix Tools	**Homework 3**
CS214 Spring 2002	**Due Date: Friday , March 15, 2002**

Although this week we have been studying *make*, this homework is still mostly about shell scripts. We'll introduce *make* and *makefiles* late, in Part III.

This homework revisit the Web ideas of the first homework. What you'll be implementing is a simple set of scripts to generate a web site, automatically maintaining a table of contents of the site on all the pages in site.

This homework requires some familiarity with HTML. See the web classic

http://archive.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html

for a quick introduction.

Here's a quick overview of the setup I have in mind. You want to create a small web site, say made up of three web pages. You want each of these pages to have some content, but also to have a table of contents allowing you to go to any of the other pages. You do not want to manage this by hand: adding a new page to the site requires you to change the table of contents on all the other pages, a process that is both error-prone, and sheer tedious. Enter automation.

Instead of writing the web pages directly, you create corresponding to every web page two files. The first contains the title of the web page, and the second contains the content of the web page. For example, you may have a file *main.title*:

```
Main page
```

and a file *main.content*:

```
<p>
  This web page only has the following
  sentence. It’s pretty lame.
</p>
```

Notice that the content file contains HTML code, the HTML code that will be used to create the final web page. By convention, we'll assume that all titles go in files with extension *.title*, and all contents go in files with extension *.content* (and the prefix of those files is the same). In the end, the pair *foo.title* and *foo.content* will be used to create a file *foo.html*, the actual web page that will be part of the site. But for now, your web-site-to-be is a set of *.title* files and corresponding *.content* files.

In Part I, you will write a script that takes a bunch of those *.title* files and creates a table of contents in HTML linking to all the (future) *.html* files. Don't worry, I'll give you a template.

In Part II, you will write a script that takes a *.content* file and a table of contents (presumably produced by the script in Part I), and create a web page containing both the table of contents and the contents. This script adds in all the HTML junk that is required to make a real web page.

Presumably, you don't want to invoke all of these scripts by hand when you want to actually create your web site. Since this is a process of deriving files from other files, we will write a makefile to automate the creation of the web site. The idea, of course, is that we may not need to recreate all the *.html* files of a web site when only a single page changes. But some of the changes may affect the table of contents, others don't. A *makefile* will help us write down all the dependencies, and take care of the grungy details for us. In Part III, you'll actually do better than this. You will write a script that takes in a list of the files that will make up the web site, and generates a *makefile* to build and manage the web site using the scripts you wrote.

(For this homework, you can assume that your scripts live in the same directory as the web site you're creating, or simply are accessible through the PATH environment variable.)

Part I: Generating a table of contents

Write a script *gen-toc* taking as input a list of *.title* files and writing to stdout HTML code implementing a table of contents corresponding to those files.

A example may help make this clear. Consider the files *foo.title*:

```
Foo
```

and *bar.title*:

```
Bar
```

along with the file *main.title* given in the introduction. Invoking *gen-toc main.title foo.title bar.title* should write something like this to stdout:

```
<hr>
<h2>Table of Contents:</h2>
<ul>
   <li><a href="main.html">Main page</a></li>
   <li><a href="foo.html">Foo</a></li>
   <li><a href="bar.html">Bar</a></li>
</ul>
<hr>
```

This is just a bunch of HTML code defining a table of contents. You can use the above as a template, or write you own formatting for the table of contents, if you enjoy hacking HTML. As a rule, a file *name.title* with a line *title* in it should produce a link such as:

```
<a href="name.html">title</a>
```

in the table of contents.

Hint: The Unix command *basename* may be handy for this script.

Part II: Generating a page

Write a script *gen-page* that takes file containing the HTML code for a table of content, a *.title* file and a corresponding *.content* file, and writes to stdout HTML code implementing a web page with the given title, the provided table of contents, and the content of the *.content* file.

For example, given the files *main.title* and *main.page* given in the introduction, and the table of contents file *toc* as generated in the previous example, executing *gen-page toc main.title main.content* should produce output that looks like this:

```
<html>
  <head>
    <title>Main page</title>
  </head>
  <body>

  <hr>
  <h2> Table of Contents:</h2>
  <ul>
     <li><a href="main.html">Main page</a></li>
     <li><a href="foo.html">Foo</a></li>
     <li><a href="bar.html">Bar</a></li>
  </ul>
  <hr>

  <p>
    This web page only has the following
    sentence. It’s pretty lame.
  </p>

  </body>
</html>
```

Notice that in this case, the generated output is the concatenation of some header gunk, including the title read from the *.title* file within the *<title>*,*</title>* tags, the table of contents, the contents given by the *.content* file, and some footer gunk. That’s the basic structure that you can use. Of course, you can modify this as you would like to suit your aesthetic needs.

Part III: Generating a makefile

Allright. Up until now, everything was just basic review of shell scripting. You have a set of *.title* and *.content* files, and you want to build your web pages. You first generate the table of contents using *gen-toc*, and then for each *.title* file you generate the corresponding *.html* web page using *gen-page*. If after that you change one of the *.title* file or one of the *.content* file, you will need to recreate some or all of the resulting *.html* files. This can be a pain, not to mention time-consuming, if you have a few hundred of those *.title* files. This is where *make* comes in handy.

Just for practice, write a couple of *.title* and *.content* files, and write a *makefile* that builds all the web pages. In other words, write a *makefile* such that when you execute *make* in the directory, it first builds the table of contents and then builds all the *.html* files. This should be a straightforward *makefile* to write.

If you think about it for a few seconds, you realize that given a set of *.title* and corresponding *.content* files, the *makefile* for building the web pages can be derived rather automatically.

Write a shell script *gen-makefile* that, when executed in a directory, figures out all the *.title* files that are in that directory, and builds a *makefile* to create a table of contents and compile those *.title* files and corresponding *.content* files into *.html* files. This script should not take any arguments. It also doesn't need to write to stdout, it can just directly create the *makefile*.

REMEMBER: Makefiles require a TAB character at the beginning of every line that correspond to a command to be executed by a rule. You'll have to do this when generating the *makefile*.

That's it. Have fun. Submission instructions will be posted to the web site.