CPSC 254
Project #3
Fall 2015

Introduction

You will write a Python script that generates an HTML file full of statistics about your system. Your Python script will process a template HTML file and substitute system information for special keywords in the template file.

Requirements

You will need a Linux installation with a working Python interpreter and network access. The Ubuntu system you set up for the first project will suffice.

Write a Python program that does the following:

1. Takes two command line arguments. The first argument is the path to a template file to read, and the second argument is the name of an output file to write.
2. Read the template file as a string.
3. Replace every keyword in the template string with some text (see below).
4. Write the result into the output file.

The main objective of your script is to find keywords within the template file, and replace them with the output of system commands. Your script must recognize the following keywords:

Template Keyword	Text to insert	Example text
%HOSTNAME%	System hostname	test-laptop
%DATE%	Today's date	03/15/2010
%UPTIME%	Current system uptime	4:21

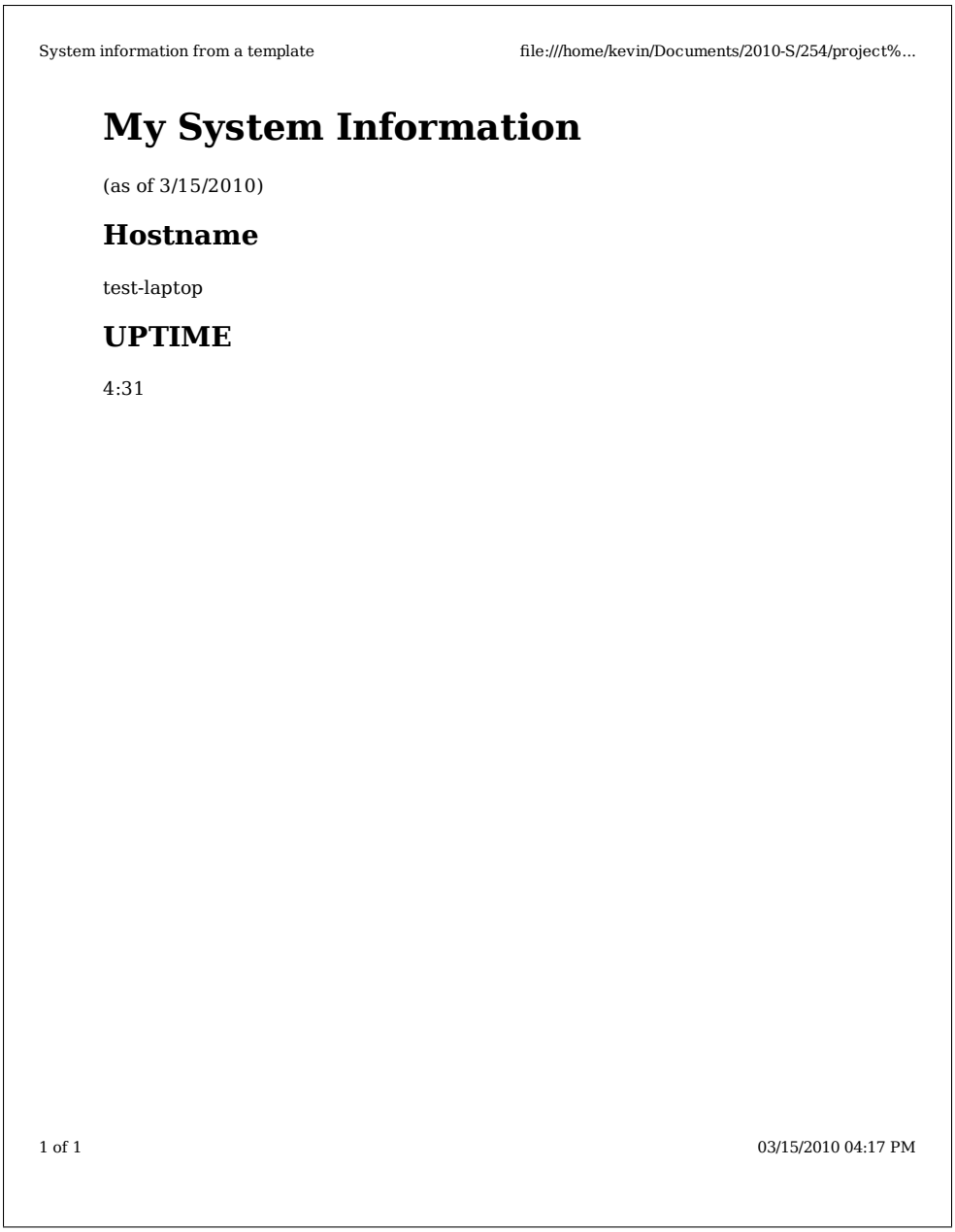
So, given the following template file `test1.html`,

```
<HTML>
<HEAD>
    <TITLE>
    System information from a template
    </TITLE>
</HEAD>

<BODY>
<H1>My System Information</H1>
    (as of %DATE%)
<H2>Hostname</H2>
    %HOSTNAME%
<H2>UPTIME</H2>
    %UPTIME%
</BODY>
</HTML>
```

your script should generate the following output file:

```
<HTML>
<HEAD>
    <TITLE>
    System information from a template
    </TITLE>
</HEAD>

<BODY>
<H1>My System Information</H1>
    (as of 3/15/2010)
<H2>Hostname</H2>
    test-laptop
<H2>UPTIME</H2>
    4:31
</BODY>
</HTML>
```

which is Firefox renders as:



Where to get the substitution text

There are many ways of getting the hostname, date, and uptime in a Python script. The two most straightforward ways I can think of are:

1. use Python API calls to get the information directly

2. use the Python `subprocess` module to run a shell command and capture the command's output

You are free to use whatever approach you prefer, or use a mixture of approaches. For this reason, **I will not be picky about the precise formatting of your template substitutions**. One way of generating a date string might yield "03/15/2010", and another might yield "Mon Mar 15 2010". Both are fine, as is any other reasonable textual description of a calendar date. The same goes for the other two keywords.

Once you have your script working, write at an additional template file called `test2.html` to test that it works.

If you refer to any sources other than the *Python Standard Library* reference, *Dive Into Python* site, or the textbook, make sure to cite them.

Deliverables

When you have your script working, print out the following:

1. Your script's source code
2. The HTML code that your script outputs when given `test1.html`
3. The code from 2. as displayed by Firefox
4. Your HTML source code for `test2.html`
5. The HTML code that your script outputs when given `test2.html`
6. The code from 5. as displayed by Firefox

The first page of your submission should include your name and a title.

Hints

We first covered the idea of writing a script that generates an HTML file in the labs based on the *Writing shell scripts* part of the *Linuxcommand.org* tutorials:

http://linuxcommand.org

I expect that you will need to search the Python API documentation for some helpful modules. That searching process is part of the project. The official Python documentation is at the following address:

http://docs.python.org/library/

If you choose to get the keyword text using Python function calls, you may find the following helpful:

1. the `datetime` module
2. the `socket` module
3. The contents of the `/proc/uptime` file

If you choose to get the text by calling shell commands, have a look at the `subprocess` module, particularly the `Popen.stdout` attribute, as well as the following shell commands: `date`, `hostname`, and `uptime`.

To access the script's command-line arguments, have a look at the `sys.argv` attribute in the `sys` module. To search and replace strings, have a look at the documentation for the `str` built-in type. Reading and writing files works similar to C++ and is handled by `File` objects; `File` is another built-in type.

You may also want to refer to the *Dive Into Python* tutorial:

http://diveintopython.org/