Department of Computer Science
University of Pretoria

Programming Languages
COS 333

Practical 6: PHP and Perl

May 7, 2024

1 Objectives

This practical aims to achieve the following general learning objective:

- To gain and consolidate some experience writing programs in several different imperative scripting languages, namely: PHP and Perl;
- To consolidate a variety of basic concepts related to imperative programming languages and scripting languages, as presented in the prescribed textbook for this course.

2 Plagiarism Policy

Plagiarism is a serious form of academic misconduct. It involves both appropriating someone else's work and passing it off as one's own work afterwards. Thus, you commit plagiarism when you present someone else's written or creative work (words, images, ideas, opinions, discoveries, artwork, music, recordings, computer-generated work, etc.) as your own. Note that using material produced in whole or part by an AI-based tool (such as ChatGPT) also constitutes plagiarism. Only hand in your own original work. Indicate precisely and accurately when you have used information provided by someone else. Referencing must be done in accordance with a recognised system. Indicate whether you have downloaded information from the Internet. For more details, visit the library's website: `http://www.library.up.ac.za/plagiarism/`.

3 Submission Instructions

Upload your practical-related source code files to the appropriate assignment upload slot on the ClickUP course page. For your PHP submission you must implement and submit a single file named `s99999999.php7`, where `99999999` is your student number. For your Perl submission you must implement and submit a single file named `s99999999.pl`, where `99999999` is your student number. Multiple uploads are allowed, but only the last one will be marked. The submission deadline is **Monday, 20 May 2024, at 12:00**.

4 Background Information

For this practical, you will be writing programs in PHP 7 (version 7.2.24 is installed in the Informatorium) and Perl 5 (version 26, subversion 1 is installed in the Informatorium). You will have to compare these languages in terms of their support for different programming language concepts. To do this, you will have to write short programs to demonstrate how each language handles the concept under consideration. These programs will not be long, but will demonstrate the concepts adequately, and should allow you to understand the language's support (or lack of support) for the features in question.

Your PHP 7 and Perl 5 programs should run using their respective command line interpreter interfaces on Linux. Be sure to check the version of the interpreter you are using in each case. This is important because in some cases later versions of language specifications are not backward compatible with earlier versions. Manuals (which include tutorials) for PHP [1] and Perl [2] and are available online. The official documentation for PHP (in HTML format) is also provided as additional documentation on the ClickUP page for the course.

5 Practical Tasks

For this practical, you will have to write programs in PHP 7 and Perl 5 to perform the same processing task on the contents of a text file. Each of your scripts must execute using the command line interpreter for the respective scripting language (even though, for example, PHP programs are typically written for execution on a web server). Information on how to use the respective command line interpreters is available via man pages under Linux.

Each program should receive two user-provided command line parameters. Command line parameters are included after the command to execute the PHP or Perl interpreter, and are separated by spaces. For more information on command line parameters, please refer to the documentation for each language, or other online resources.

The programs first read an ASCII input file. The name of the text file is specified by the first command line parameter. The file can consist of any number of lines. The read lines can consist of any standard ASCII characters, including uppercase and lowercase letters, numbers, symbols, and whitespace characters (including spaces, tabs, linefeeds, and so on).

The file input should be concatenated into a single string. All uppercase characters are converted to lowercase, while all non-alphabetic characters are stripped from the string. For example, if the input file consists of the following text:

```
Madam, here is Adam
Adam, look HERE at 1
```

the converted string would be:

```
madamhereisadamadamlookhereat
```

The program also receives a single word as the second command line parameter, and counts the number of occurrences of this word in the converted string, in reverse order (from right to left). The number of occurrences of the word should be produced as output. For example, if the second parameter is `mad`, the output would be:

```
Matches: 3
```

You may assume that there will be no overlapping matches (i.e. you will not have to handle cases where the input file contains text such as “adada”, and the word to search for is “ada”). A sample input file, named `input.txt`, is provided in the “Additional Files” folder for this practical.

There are, of course, several ways in which the required input reading and search can be performed, and you are free to use whichever approach you prefer. You may use any built-in functionality provided by either language. **You may not, however, use any third party libraries or extensions**. One possible approach to converting the input and extracting a count is to use regular expressions. Also note the following special requirements for the two scripts:

PHP

While PHP programs are typically written for execution on a web server, we will be using the command line interface of PHP 7 to interpret your script. Your script must accept the two command line parameters, as described above.

When the PHP command line interpreter runs your PHP script, it should generate HTML output which can be opened in a web browser. Note that this output must include HTML tags (it can’t just be the text output, even though a Web browser will open a file containing just the output). Also note that this output is generated on the command line, and not as a file output. Check the `php` man page under Linux for information on how to use PHP from the command line.

Perl

We will be using the command line interface of Perl 5 to interpret your script. As for the PHP implementation, your script must accept the command line parameter, as described above.

Your script should not generate HTML output, as the PHP script does. Instead, only output the required result of the program’s execution.

6 Marking

Each of the implementations will count 5 marks for a total of 10 marks. Submit the PHP and Perl implementations to the appropriate assignment upload slots. Do not upload any additional files other than your source code. Both the implementation and the correct execution of the programs will be taken into account. Your program code will be assessed during the practical session in the week of **Monday, 20 May 2024**.

References

[1] Mehdi Achour, Friedhelm Betz, Antony Dovgal, Nuno Lopes, Hannes Magnusson, Georg Richter, Damien Seguy, and Jakub Vrana. PHP manual, 2020. Access at `http://php.net/download-docs.php`.

[2] Perl.org. Perl docs, 2020. Access at `https://www.perl.org/docs.html`.