Edward Young
ey52@columbia.edu
COMSW4115 PLT

The Text Processing Language (TPL)

An Overview

Introduction

Text Processing Language (TPL) is a text-processing language that is designed to provide very simple text file manipulation tools. Text file manipulation programs are very useful when dealing with a large number of text files. Two particular instances come to mind:

- Web Developers who have many text files that are written in HTML, PHP, or some other scripting language.
- Systems Administrators who deal with text based configuration files and log files.

Text files which contain ASCII text are used in almost every aspect of computing. Common examples are HTML web pages, Java source files, and Apache Web Server log files. Many times, a user who has to deal with these files might want to manipulate many of these files at once. Opening and manipulating each of these files can be tedious. The ability to write a simple program using TPL to handle the tedious tasks can be time-saving.

Why Another Language?

TPL is designed with a "right tool for the right job" mentality. It is not intended to replace powerful programming languages such as C, C++, or Java. Instead, it is intended to be another tool in a programmer's toolbox dealing primarily with text file manipulation.

Another text-processing language that is available today is AWK. While AWK is very simple and powerful, some of the syntax may be confusing for those who are just beginning to learn the language. For example, some of the built in variables such as `NR` or `NF` may be difficult to decipher since only two upper case letters are used to describe the variables. Variables with such similar names may be easily confused. TPL uses descriptive variable names such as `NUM_ROWS` or `NUM_FIELDS` in order to make reading the code easier. This requires more typing but the ease of maintenance will increase if the variable names are more descriptive.

Goal

TPL is designed to be a simple language that anyone should be able to learn. The text processing capabilities can enable a programmer to perform normally tedious tasks with a

simple program. Inexperienced programmers can use TPL to write useful programs without having to learn all of the caveats of a larger programming language such as C/C++ or Java. The programming syntax is simple enough for anyone to learn.

Simple

TPL will be modeled after procedural languages such as C/C++. Programs can be anything from "one-liners" to 100 line programs. Those who are familiar with procedural languages will be able to make the transition from writing large programs to small simple programs using TPL. For example, the "Hello World" program in C would normally take 5 lines of code:

```
#include<stdio.h>

main()
{
    printf("Hello World");
}
```

In TPL, the "Hello World" program can be reduced to:

```
print ("Hello World");
```

These two programs are very similar in that they both call a "print" function to print the string "Hello World" to the console. Since TPL is has a specific focus on text processing, some complexity can be taken out. A main() function is not needed since TPL programs will not be compiled. They will simply be run through the TPL interpreter. TPL programs can be simple scripts or somewhat complex programs using many functions.

Anyone who has programmed in C/C++ or Java will be able to learn TPL quickly due to the intuitive syntax and semantics. An inexperienced programmer will also be able to learn TPL quickly.

Procedural

TPL programs are procedural. They focus on writing functions which perform small tasks and then combining these functions together to create programs. This is different from Object Oriented Programming in that the data is not represented as objects. When processing text, such abstraction probably is not needed. If it is, it is better to use a language such as C++ or Java which provide the full benefits of Object Oriented Programming.

Interpreted

TPL programs will need the TPL program to run. The TPL program is a java program that will interpret the TPL program and provide the output. An example of running a TPL program:

```
java TPL nightly_batch.tpl
```

Interpreted programs may not run as fast as compiled programs but speed usually is not an issue with text file manipulation. The main importance is the accuracy of the result provided by the program. Unless the program takes a very long time, the programmer might not care if he or she has to wait 5 minutes for a TPL program to search and replace certain variable names in 100 PHP files. This would probably save the programmer hours of work. The programmer also would not have to worry about user error since the computer performed the search and replace. Some additional benefits of TPL being an interpreted language are that the programs are portable and architecture independent.

Architecture Neutral and Portable

Java is currently available on Windows, Macintosh, Solaris, and Linux. Therefore, the TPL interpreter can be installed on any of these platforms as well. A TPL program that was first written on a Windows machine can also be used on a Linux machine without modification. Since TPL programs are interpreted by a Java program, TPL programs will hold many of the same benefits and advantages as Java programs.

Domain Specific

TPL is meant for only for the manipulation of text files. Programming languages such as C/C++ or Java are meant to be multipurpose. You can write databases, applications, and even computer games with those languages. TPL will not do any of those. It is only meant to provide a powerful tools for text file processing. Here are some examples of situations where a language like TPL can be applied:

- Programmers: They may have text in dozens of source code files that need to be changed. Instead of opening and editing each and every file, they can write a TPL program to search through all of the programs and do the replace automatically.

  An example:

  A PHP web programmer just found out that a column name in a database table has changed. He has 10 files which contain a reference to that column name. Instead of opening and searching each file, he simply writes the following TPL program:

```
$dir = “.”;
$files = getFileList($dir);
for($i=0;$i<count($files);$i++) {
    if(!isDir($files[$i])) {
        // columnA is search text
        // columnB is the replacement text
        replace(“columnA”, “columnB”, $files[$i]);
    }
}
```

- Systems Administrators: Systems administrators can find a text processing language to be very helpful. Often Systems Administrators deal with text file configuration and log files. An administrator can spend a lot of time modifying or reading these files. It can be helpful for an administrator to write a program that can automate some of his or her tasks.

  An example:

  A Linux Systems Administrator wants to get a daily report of errors in the /var/logs/messages log and various other logs such as Apache and Tomcat. He will search for the string "ERROR" and "WARNING" and output the count to a file. He can run the job when he needs to and then check the results. If there are no warnings or errors, he will not have to examine any log files and he would have probably saved himself about half an hour or more by not having to open each individual log.

```
$logs[0] = "/var/log/messages";
$logs[1] = "/var/log/httplog";
$logs[2] = "/var/log/tomcatlog";
$logs[3] = "/var/log/otherlog";

$output_file = "/home/admin/daily_report.txt";

for($i=0;$i<count($logs);$i++) {
    $warning_count =
            string_count("WARNING",$logs[$i]);
    $error_count = string_count("ERROR",$logs[$i]);
    write_line($logs[$i]);
    write_line($output_file, $warning_count);
    write_line($output_file, $error_count);
    write_line();
}
```

Summary

TPL is a good tool for any programmer to have. With its ease of use and portability, the language can be used across many platforms and by many different people. Text processing will be made easier and less of a chore if a well written TPL program is used.