Using Perl to Sort English Placement Test Results

Peter Ilic, Asia University

Abstract: This paper presents a computer program designed to assist in the sorting of first year students at a private university. The program significantly reduces the time needed to sort approximately 1700 students into 21 levels. We explain the basic steps in the sorting processes and relate these to the computer code. An explanation of problems faced in the testing of the program is included. In addition possible improvements to the program are discussed.

The purpose of this program is to partially automate the sorting of students into levels based on their English placement test scores. When done individually these sort routines are very simple to perform manually. However, with a large number of students the task becomes very tedious and time consuming. This program reduces the sort time to a fraction of that needed for a manual sort.

This program was written in the Practical Extraction and Report Language (Perl). For our purpose Perl has several advantages. It was originally designed for manipulating text. Being a high level language the command structure is relatively easy to use. Perl is available under a free use license, so there is no charge for using it. Perl programs are compiled each time they are run, so they are system independent.

Following the Freshman English Placement Test (FEPT), we receive a single file named "data.csv" containing data on all Freshman English (FE) students entering Asia University (AU) and their individual FEPT test results. The original data file contains 110 columns of data separated by commas for approximately 1700 students. A file containing data that is separated by commas is called a comma-separated file or a CSV file. This program will use these commas to distinguish one data field from another.

The first step is to separate the data we require from the 110 columns in the original file. For our purpose we need student number, major, name, gender, listening score, reading score, total score, and FE level. This information is found in columns 0, 1, 3, 4, 105, 106, 107, and 109 respectively. The following code copies only these columns from the original "data.csv" file and saves them in a new file named "data2.txt".

```
open (OrigFile, "data.csv") or die "Can't open read file: $!";
open (TrimData, ">data2.txt") or die "Can't open write file: $!";

while (<OrigFile>) {

      chomp;
      my @fields = split (/,/, $_);
      print TrimData "$fields[0], $fields[1], $fields[3], $fields[4],
                  $fields[105], $fields[106], $fields[107], $fields[109]\n";

}

close TrimData;
close OrigFile;
```

We now have a new file called “data2.txt” that contains eight columns of data separated by commas. We will no longer use the original file “data.csv”. Column nine in “data2.txt” contains the FE level of each student. However, there are always some students that do not have an assigned FE level. At the present time this program does not place these students. So the data for these students will be copied to a separate file and manually inserted into levels later. The following code copies the data for any student that does not have an assigned FE level and saves it in a new CSV file named “no_level.csv”.

```
open (TrimData, "data2.txt") or die "Can't open read file: $!";
open (Data, ">no_level.csv") or die "Can't open write file: $!";

while (<TrimData>) {

      chomp;
      my @fields = split (/,/, $_);
      if ( $fields[7] == '' ) {
            print Data "$fields[0], $fields[1], $fields[2], $fields[3],
                    $fields[4], $fields[5], $fields[6], $fields[7]\n";
      }
}

close Data;
close TrimData;
```

It is important to note that we have only copied the data from the original “data2.txt” file, so the file contents remain unchanged.

In step three we need to separate the students by their area of study. This program recognizes five areas of study: business, economics, law, international relations (IR), and junior collage (JC). With the exception of business each of these areas has been assigned a single three-digit number and this number is stored in column one of “data2.txt”. Business is an exception because it has three different three-digit numbers. This means we will need to do three sorts for business. The following code copies any students with the business identification numbers 122, 123, or 124 and places them in a new CSV file named “bus.txt”.

```
open (TrimData, "data2.txt") or die "Can't open read file: $!";
open (Data, ">bus.txt") or die "Can't open write file: $!";

while (<TrimData>) {

      chomp;
      my @fields = split (/,/, $_);
      if ( $fields[1] == '122' ) {
            print Data "$fields[0]", ",Bus,", "$fields[2], $fields[3],
                  $fields[4], $fields[5], $fields[6], $fields[7]\n";
      }

      if ( $fields[1] == '123' ) {
            print Data "$fields[0]", ",Bus,", "$fields[2], $fields[3],
                  $fields[4], $fields[5], $fields[6], $fields[7]\n";
      }

      if ( $fields[1] == '124' ) {
            print Data "$fields[0]", ",Bus,", "$fields[2], $fields[3],
                  $fields[4], $fields[5], $fields[6], $fields[7]\n";
      }
}

close Data;
close TrimData;
```

Next we copy the data for the economics students that are identified by the number 133. This data is then saved in a new CSV file named “econ.txt” by the following code.

```
open (TrimData, "data2.txt") or die "Can't open read file: $!";
open (Data, ">econ.txt") or die "Can't open write file: $!";

while (<TrimData>) {

      chomp;
      my @fields = split (/,/, $_);
      if ( $fields[1] == '133' ) {
            print Data "$fields[0]", ",Econ,", "$fields[2], $fields[3],
                  $fields[4], $fields[5], $fields[6], $fields[7]\n";
      }
}
close Data;
close TrimData;
```

The code for the other three areas of study is largely unchanged from the economics example above. The only difference is that the sort number in line six and the destination file name in line two will change for each area. Law will be saved as “law.txt”, IR as “ir.txt”, and JC as “jc.txt”. At this stage we have six files that where created from the original “data2.txt” file. Five files containing the students from each of the areas of study and one file containing those students that have not been assigned an FE level. At this point we no longer need the file “data2.txt”.

The fourth and final step consists of separating the students from each area into their FE level. The FE levels for each area of study are stored in column seven of “bus.txt”, “econ.txt”, “law.txt”, “ir.txt”, and “jc.txt”. The following code searches the “bus.txt” file and copies any students that have a “1” in column seven to a new CSV file named “ bus_1.csv”. This file will contain all level-one business students.

```
open (TrimData, "bus.txt") or die "Can't open read file: $!";
open (Data, ">bus_1.csv") or die "Can't open write file: $!";

print Data "Major, Class, Name in Romaji (Family Given), ID, M-F, Lstn, Rdg,
            Total, ACTFL, Comments\n";

while (<TrimData>) {

      chomp;
      my @fields = split (/,/, $_);

      if ( $fields[7] == '1' ) {
            print Data "$fields[1], ", 1,", $fields[2], $fields[0], $fields[3],
                                          $fields[4], $fields[5], $fields[6]\n";
      }
}

close Data;
close TrimData;
```

In addition to the basic sort, this code does three other important jobs. The third line in the code adds a row of headings to each destination file. The headings are, from left to right major, class, student name, student number, gender, listening score, reading score, total score, ACTFL level, and a comments column. For the time being the ACTFL level and comments columns will be blank. Then the eighth line of this code changes the order of the data so that it matches the new column headings. The reason we do this is to make the final destination file easier to read and to create space for the data that will be added later. You can also see that the FE level or column seven was not saved to the destination file since this data is no longer required.

This section of code is repeated for all levels of business. With each level the search term changes from "1" for level one to "2" for level two, and so on to the final level "21". At the same time the destination file name changes from "bus_1.csv" for level one to "bus_2.csv" for level 2, and so on to "bus_21.csv". Once the business students have been sorted into their levels, the same code can be used to sort all the remaining areas of study. The code will be identical to the business sort above with the exception of the destination file name in line two. Where business is saved as "bus_1.csv" through "bus_21.csv", economics will be "econ_1.csv" through "econ_21.csv", and so on for law, IR, and JC.

We ran this program using the results from the 2005 FEPT. It was successful in creating 21 text files for each of the five areas of study and one text file containing the students without an assigned level. However one adjustment was necessary to the column numbers in the first sort routine. The cause of this was an additional comma in the 2005 data that was not in the original test data. The comma was inserted between each student's first and family name. Since this program uses commas to separate data fields this had the effect of adding one to the column count. Once the numbers were adjusted the program preformed without error. There are several ways that this program could be expanded and improved. A section could be added to read a list of instructors' names then place the correct class list into a directory for each instructor. In addition a graphical user interface could be added using the Perl/TK extension to Perl.