Simple Database Manipulation Using MKS Toolkit

Ever need to easily keep track of many pieces of data? It's not uncommon. Will a dedicated database application do the job for you? Sure. Do you need that much power? Probably not. Many simple database tasks can be handled easily with just the tools provided in MKS Toolkit. In fact, since MKS Toolkit includes programming languages like **awk** and **perl**, you can actually create and access quite complex databases. However, we're not going to be using either **awk** or **perl** in creating our simple database. With this restriction, we can see just how powerful many of the other utilities are.

THE DATABASE STRUCTURE

The first step in creating the database is deciding what pieces of data will be stored for each entry. For our purposes, let's say that we want to keep track of a list of friends' names, the city they live in and their birthdays. The name will come first, followed by a tab, then the city followed by a tab, and finally, the birthday. All of this information will be kept in a simple text file that can be created and edited with **vi**. *Figure 1: The Data* shows the contents of such a file.

Let's assume that we have typed this data into a file `database.txt`. We will be using this file throughout this document to demonstrate the database techniques discussed.

Once the data has been entered or if new entries have been added, the `database.txt` file should be sorted. The easiest way to do this is with the **sort** utility:

```
sort database.txt > tempfile
mv tempfile database.txt
```

These two commands sort the database file, store the sorted result in the file named `tempfile`, and then rename `tempfile` with the name `database.txt`.

As an alternative, you could invoke the sort utility from with the **vi** editor itself. For example, if you were editing `database.txt`, you could just enter the following EX command lines:

```
:1,$ ! sort
:w!
```

Regardless of the method used, the sorted database will look like *Figure 2: The Sorted Data*.

While the examples here only use the basic version of the **sort** utility (with no options) which sorts a text file based on the full contents of each line, **sort** is actually capably of complex sorting using multiple fields. For example, you could use it to sort this database by first city, then birthday, and finally by name.

Ed Brown	Austin, TX	January 12
Marcy Davis	Louisville, KY	August 17
Kevin Logan	Toronto, ON	June 1
Tina Monahan	Vancouver, BC	April 22
Steve Baxter	Des Moines, IA	November 9
Deborah Caine	Providence, RI	September 19
Steve Kent	Bangor, ME	December 30
Jerry Curtis	Toronto, ON	July 15
Elizabeth Simms	New York, NY	July 31
Cathy Marx	Los Angeles, CA	February 11
Tina Douglas	Milwaukee, WI	March 28
Angie Kemp	Cleveland, OH	April 2
April Dempsey	Denver, CO	May 10
Charles Simpson	Portland, ME	September 14
Ron Austin	Portland, OR	October 25

Figure 1: The Data

Angie Kemp	Cleveland, OH	April 2
April Dempsey	Denver, CO	May 10
Cathy Marx	Los Angeles, CA	February 11
Charles Simpson	Portland, ME	September 14
Deborah Caine	Providence, RI	September 19
Ed Brown	Austin, TX	January 12
Elizabeth Simms	New York, NY	July 31
Jerry Curtis	Toronto, ON	July 15
Kevin Logan	Toronto, ON	June 1
Marcy Davis	Louisville, KY	August 17
Ron Austin	Portland, OR	October 25
Steve Baxter	Des Moines, IA	November 9
Steve Kent	Bangor, ME	December 30
Tina Douglas	Milwaukee, WI	March 28
Tina Monahan	Vancouver, BC	April 22

Figure 2: The Sorted Data

RETRIEVING DATA

There are two approaches that can be use to search the database and retrieve an entry based on a keyword. The first uses the **look** utility; the second uses the **grep** utility.

With the look Utility

The **look** utility searches a sorted text file for all entries that begin with a given string and displays them. For example:

MKS Toolkit

```
look "Kevin" database.txt
```

searches the `database.txt` file for an entry beginning with "`Kevin`". It displays:

```
Kevin Logan      Toronto, ON      June 1
```

On the other hand:

```
look "Tina" database.txt
```

displays:

```
Tina Douglas     Milwaukee, WI    March 28
Tina Monahan     Vancouver, BC    April 22
```

If you just wanted to display the information for Tina Monahan, you would use:

```
look "Tina Monahan" database.txt
```

There are two important things to remember about using the **look** utility. First, the database needs to be sorted on the first field. Second, **look** is only looking for matches at the beginning of that first field.

With the grep Utility

The alternative to **look** for retrieving data is **grep**. The **grep** utility lets you search for a string of characters that can appear anywhere on a line in a text file and displays the lines that contain matches. With **grep**, you can search for and retrieve data based on any field in the database.

For example, if you wanted to find everyone in the database that was from Toronto, you would use:

```
grep "Toronto" database.txt
```

and would get back the following:

```
Jerry Curtis     Toronto, ON      July 15
Kevin Logan      Toronto, ON      June 1
```

However, you have to remember that **grep** searches for the specified string anywhere in the database not just in select fields. As a result, if you were looking for everyone from Austin and used:

```
grep "Austin" database.txt
```

you would see the following entries displayed:

```
Ed Brown         Austin, TX       January 12
Ron Austin       Portland, OR     October 25
```

As you can see, **grep** found not only the entry for Ed Brown from Austin, TX but also the entry for Ron Austin. Sometimes you can eliminate the undesirable entries from **grep**'s display by using a more detailed search string. In this particular case, specifying "`Austin, TX`" as the search string instead of just "`Austin`" returns only Ed's entry and not Ron's, whereas specifying "`Austin    `" (that is, `Austin` following by a TAB character) will only return Ron's entry.

Displaying Specific Fields

Sometimes you don't want to display all the information in an entry, just specific fields. For example, you may only want to see the name and birthday fields. You can do this with the **cut** utility which lets you specify (with the **-f** option) which fields to display. For example:

```
cut -f1,3 database.txt
```

displays only the first and third fields (that is, the name and birthday fields).

By piping the output from a **look** or **grep** search of the database through **cut**, you can tailor the information that you receive back. For example:

```
look "Steve" database.txt | cut -f 1,3
```

MKS
ToolKit

displays the full names and birthdays of every Steve in the database:

```
Steve Baxter       November 9
Steve Kent         December 30
```

REORDERING THE DATABASE FIELDS

In the previous section, you saw how you can use the `cut` utility to select and display individual fields. You can use this capability along with the ability of the `paste` utility to glue two files together as one to rearrange the order of the database fields—giving a whole new meaning to "cut and paste".

Let's suppose that you want to change the arrangement of your database so that the birthday field (currently field 3) comes first followed by the name field (currently field 1) and finally the city field (currently field 2). First, you would use the `cut` utility to extract field 3 from each entry and put it in a temporary file:

```
cut -f 3 database.txt > tempfile1
```

Next, you would extract fields 2 and 3 and store them in a secondary temporary file:

```
cut -f 1,2 database.txt  > tempfile2
```

Finally, you would use the `paste` utility to paste the two temporary files back together as a new database :

```
paste tempfile1 tempfile2 > new_database.txt
```

The result of this operation is shown in *Figure 3: The Reordered Database.* Like the original database, you must sort `new_database.txt` before using the `look` utility to search it. Using `grep` to retrieve entries does not require it to be sorted.

January 12	Ed Brown	Dallas, TX
August 17	Marcy Davis	Louisville, KY
June 1	Kevin Logan	Toronto, ON
April 22	Tina Monahan	Vancouver, BC
November 9	Steve Baxter	Des Moines, IA
September 19	Deborah Caine	Providence, RI
December 30	Steve Kent	Bangor, ME
July 15	Jerry Curtis	Toronto, ON
July 31	Elizabeth Simms	New York, NY
February 11	Cathy Marx	Los Angeles, CA
March 28	Tina Douglas	Milwaukee, WI
April 2	Angie Kemp	Cleveland, OH
May 10	April Dempsey	Denver, CO
September 14	Charles Simpson	Portland, ME
October 25	Ron Guerin	Portland, OR

Figure 3: The Reordered Database

So, why would you reorder the fields in your database? The primary reason would be that so you can use `look` as opposed to `grep` to retrieve entries. As we saw earlier, `grep` has the problem of potentially matching on fields other than the one desired. Whereas, if you make the field you want to search on the first field in the database, you can easily use `look` to search for matching entries.

MERGING TWO DATABASES

Let's say that you have a second database for these same people that just contains their names and the kind of pie each likes as shown in *Figure 4: The Sorted Pie Database.* In some ways, it seems silly to be maintaining two separate databases for the same group people. Especially, for instance, if you like to send people their favorite pie as a birthday gift each year. In which case, it would be very handy for all the data to be in the same place.

Fortunately, the MKS Toolkit has a very handy utility named `join` designed to do just what we want to do here: merge two databases based on common first fields. As you can see, if you compare our sorted original database as shown in *Figure 2* to the one in *Figure 4,* you can see that the two databases contain all the same names. This means that we can easily combine the two with this command:

Angie Kemp	Lemon Meringue
April Dempsey	Apple
Cathy Marx	Blueberry
Charles Simpson	Pumpkin
Deborah Caine	Cherry
Ed Brown	Mincemeat
Elizabeth Simms	Pecan
Jerry Curtis	Apple
Kevin Logan	Pumpkin
Marcy Davis	Lemon Meringue
Ron Guerin	Rhubarb
Steve Baxter	Blueberry
Steve Kent	Key Lime
Tina Douglas	Apple
Tina Monahan	Boston Cream

Figure 4: The Pie Database

```
join -t" " database.txt piedatabase.txt > new_database.txt
```

MKS
ToolKit

The **-t** option (with the quoted TAB character following it) specifies that **join** should use the TAB character as the field separator for this operation, instead of the default which is the space character. The result of this operation is the combined database (`new_database.txt`) shown in Figure 5*: The Combined Database*.

Angie Kemp	Cleveland, OH	April 2	Lemon Meringue
April Dempsey	Denver, CO	May 10	Apple
Cathy Marx	Los Angeles, CA	February 11	Blueberry
Charles Simpson	Portland, ME	September 14	Pumpkin
Deborah Caine	Providence, RI	September 19	Cherry
Ed Brown	Austin, TX	January 12	Mincemeat
Elizabeth Simms	New York, NY	July 31	Pecan
Jerry Curtis	Toronto, ON	July 15	Apple
Kevin Logan	Toronto, ON	June 1	Pumpkin
Marcy Davis	Louisville, KY	August 17	Lemon Meringue
Ron Austin	Portland, OR	October 25	Rhubarb
Steve Baxter	Des Moines, IA	November 9	Blueberry
Steve Kent	Bangor, ME	December 30	Key Lime
Tina Douglas	Milwaukee, WI	March 28	Apple
Tina Monahan	Vancouver, BC	April 22	Boston Cream

Figure 5: The Combine Database

You can now use **look** and **grep** to retrieve information from this new database. Similarly, you can use **cut** and **paste** to reorder the fields.

CONCLUSIONS

While I suspect that very few, if any, of you have a specific need for creating and managing a database of home towns, birthdays, and pies, the basic techniques can be applied to any number of situations where lists of data need to be searched or manipulated. Additionally, MKS Toolkit contains many other utilities that can be used in conjunction with both simple and complex databases: from using **diff** to compare two text databases to using **db** to send SQL (Structured Query Language) requests to real databases. Finally, as mentioned in the introduction, the **awk** and **perl** programming languages can be used to write scripts which create and manage quite complex databases.

For more information about the MKS Toolkit products please visit http://www.mkssoftware.com and to view the full reference pages for the commands mentioned in this document visit http://www.mkssoftware.com/docs/cmd_index.asp.

MKS Toolkit