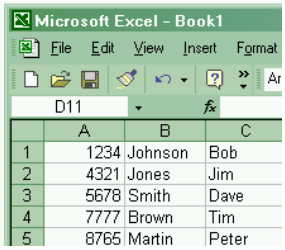
SYST13416
Introduction to Linux
Operating

Advanced commands

DATA, INFORMATION, AND DATABASE TABLES

Objectives

- Manipulate database tables using flat files
- Understand the concepts of data and database
- What is date? What is a database? What is a relational database?
- Unix Database processing: database, record, field, delimiter, default delimiter, key
- Commands: join, sort, cut, paste

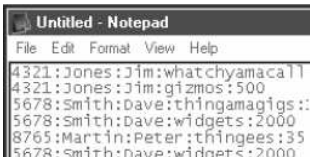


What is data?

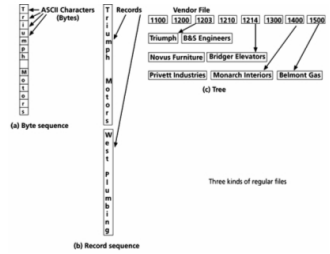
- The word data is the plural of datum, meaning a piece of information, or, "the given". It is also commonly used as a mass noun, where it is treated as singular, e.g., "This is all the data from the experiment". Many consider this usage incorrect, or at least colloquial, and would prefer "These are all the data from the experiment" (each individual measurement or result from the experiment is a single datum). The word datum is taking on a life of its own, however, and it is now most often used to mean a designated base point against which measurements are taken.
- Raw data are numbers, characters, images, or other method of recording, Computers nearly always represent data in binary. in a form which can be assessed by a human or (especially) input into a computer, stored and processed there, or transmitted on some digital channel. Data on its own has no meaning, only when interpreted by some kind of data processing system does it take on meaning and become information.
- People or computers can find patterns in data to perceive information, and information can be used to enhance knowledge. Since knowledge is prerequisite to wisdom, we always want more data and information. But, as modern societies verge on information overload, we especially need better ways to find patterns.

What is database?

- A database is an information set with a regular structure that allows automated searches and updates. There are a wide variety of databases, from simple tables stored in a single file to very large databases with many millions of records, stored in rooms full of disk drives.
- One way of classifying databases is by the programming model associated with the database. Several models have been in wide use for some time. Historically, the hierarchical model was implemented first, then the network model, then the relational model and flat models reached their zeniths.



- The flat (or table) model consists of a single, two-dimensional array of data elements, where all members of a given column are assumed to be similar values, and all members of a row are assumed to be related to one another. For instance, columns for name and password might be used as a part of a system security database. Each row would have the specific password associated with a specific user. Columns of the table often have a type associated with them, defining them as character data, date or time information, integers, or floating point numbers. This model is the basis of the spreadsheet.
- The network model allows multiple tables to be used together though the use of pointers (or references). Some columns contain pointers to different tables instead of data. Thus, the tables are related by references, which can be viewed as a network structure. A particular subset of the network model, the hierarchical model, limits the relationships to a tree structure, instead of the more general directed graph structure implied by the full network model.
- Relational databases also consist of multiple database tables. Unlike the hierarchical and network models, there are no explicit pointers; in theory, columns of any type may be used to create an ad-hoc relationship between two or more tables. Relational databases allow users (or, more often, programmers) to write queries that were not anticipated by the database designer. As a result, relational databases can be used by multiple applications in ways the original designers did not foresee, which is especially important for databases that might be used for decades. This has made relational databases very popular with businesses.
- A database application is a type of computer application dedicated to managing a database. Database applications span a huge variety of needs and purposes, from small user-oriented tools such as an address book, to huge enterprise-wide systems for tasks like accounting.



Relational databases

- Most database comprise of a number of related tables. Why? There are three types of anomalies that we need to worry about with databases: insertion,

Order

Order#	OrderDate	Customer#	CustomerName	CustAddr	CustCity	CustProv	CustPostal	Product#	ProdDesc	QtyOrdered	UnitPrice
123	2001/03/03	C321	Ajax Furniture	23 Oak Street	Burlington	Ont	L9F 3M5	D12 R32 B23	Desk Dresser Bed	42 20 32	112.00 202.00 140.00
124	2001/03/03	C523	Bumble's Beds	46 Soft Place	Oakville	Ont	L8K 2H3	B23	Bed	24	140
125	2001/03/07	C321	Ajax Furniture	23 Oak Street	Burlington	Ont	L9F 3M5	T10	Table	16	90

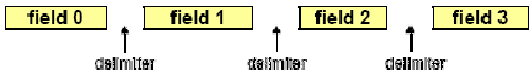
update and deletion. If the database consists of a single table, i.e., like this:

- Notice that Order 123 has multiple values in a single field. This may complicate the manner in which the data is stored and retrieved.
- Also notice the repeated data on Ajax Furniture. Image that instead of three orders there were 3,000 orders, 300 of which were placed by Ajax. Over time, the size of the table would grow and become hard to manage.
- If the description of Product B23 is changed, multiple instances of the description need to be changed (update anomaly).
- A new products cannot be added to the table, until a customer places an order (insertion anomaly).
- If the Product B23 is deleted, customer C523 would also be deleted since this is the only instance of this customer in the database (deletion anomaly).
- To eliminate these anomalies, databases are designed in such a way that they do not contain redundant copies of data and the relationship among the different objects are preserved. For example, the above single table is divided into four tables, each holding data about a single object: orders, customers, and products. The table OrderProduct describes the relationship between orders and products because there exists a many-to-many relationship.

Order(Order#, OrderDate, *Customer#*)
Customer(Customer#, CustomerName, CustAddr, CustCity, CustProv,CustPostal)
OrderProduct(Order#, Product#, QtyOrdered)
Product(Product#, ProdDesc, UnitPrice)

UNIX Database Processing

- **Database**: A database is data set with a regular structure that allows automated searches and updates. There are a wide variety of databases, from simple tables stored in a single file to very large databases with many millions of records, stored in rooms full of disk drives.
- Processing relational database differs from other file processing and uses the following concepts:
  - **Database (table):** An organized collection of files, containing related records
  - The **flat** (or **table**) model consists of a single, two-dimensional array of data elements, where all members of a given column are assumed to be similar values, and all members of a row are assumed to be related to one another. This model is **the basis of the spreadsheet**.
  - **Record (row):** Data made up of a number of separate elements of information,
    name:address:phoneNumber
  - **Field (column):** One element of the record, e.g., address
  - **Delimiter**: The character that separates one field from another. UNIX default



    delimiter is a space or a tab.
  - **Default Delimiter**: Default delimiter between fields is a single space (blank) character, or tab character
  - A **key** is the field or part of a field that the program uses when it acts on data.

- **Commands**
  join, sort, cut, paste, join, pr

Sheridan

SYST13416
Introduction to Linux Operating

Advanced Commands

SIMPLE FILTERS AND TABLE MANIPULATION

Objectives

- Commands: sort, cut, paste, comm, diff

Sort, Cut, and Paste Text Files

Commands and Terms

- Command: **sort -tko**
- Command: **cut -dk**
- Command: **paste**
- Command: **comm**
- Command: **diff**

For each command, examine the manual page.

For each question, note the result and ensure you understand it.

A1) Examine the manual pages for cut, sort, and paste, and note their descriptions and what the various options used in these exercises do for more details.

A2) Sorting files with no options

```
sort music.new
```

Note that the sort is alphabetical, starting with the first column of characters.

A3) Extract the second, fourth, fifth, and sixth column from the file only. Note the file is delimited by colons:

```
cut  -d:  -f2,4-6  music.data
```

A4) Extract the first field:

```
cut  -d:  -f1  music.data
```

A5) Display the third column:

```
cut  -d:  -f3  music.data
```

A6) What are the results of the following commands?

```
cut  -d:  -f2-3  music.data
cut  -d:  -f1,2-3  music.data
```

A7) Sort the file alphabetically starting at the beginning of each line and save the result to a file called music.new:

```
sort   -t:  -o  music.new  music.data
```

A8) Sort the file on the fourth file and redirect the results to file called music.sorted

```
sort   -t:  –k  4  music.data  >  music.sorted
```


A9) Display the files music.new and music.sorted

```
cat  music.new  music.sorted
```

Verify the sort results for each file.

A10) Find the files list1 and list2 that you created earlier. Now try the following commands and note the results:

```
paste  list1  list2
paste  -s  list2
paste  -s  -d'\t\n'  list1
```

A11) Combine files using the cat command:

```
cat  list1  list2  >  list3
```

A12) Combine files using the paste command:

```
paste  list1  list2  >  list4
```

A13) Examine the results:

```
cat list3
```

Then examine the results:

```
cat list4
```

What is the difference between list3 and list4 (if any)?

A14) Using the customer.data you created earlier, execute the following set of commands and note the results after each.

```
cut -d: -f1 customer.data > names
cat names
cut -d: -f6 customer.data > prices
cat prices
cut -d' ' -f2 names | paste - prices > orders
cat orders
```

What results did you observe with names, prices, and orders?

Compare Text Files with comm and diff

A15) List reasons why you or a programmer or an end user would need to compare the content of 2 files.

A16) Sort the file exercise and name the new file exer

A17) Sort the file success and name the new file succ

A18) Compare the files exercise and success using the command comm:

```
comm  exercise  success
comm  –12  exercise  success
comm  –13  exercise  success
comm  –23  exercise  success
```

What did you find? Read the manual page for comm and explain your results.

A19) Compare the sorted files exer and succ using the command comm:

```
comm  exer succ
comm  –12  exer succ
comm  –13  exer succ
comm  –23  exer succ
```

What did you find? How do these results differ from the results of the previous question?

A20) Read the diff manual page and examine the results

```
diff  exercise  success
diff  -ibw  exercise  success
```

How is this useful to a Computer Programmer working with various versions of the same code?

SYST13416
Introduction to Linux Operating

Advanced Commands

WORKING WITH MULTIPLE TABLES

Objectives

- Manipulate database tables using flat files
- Creating reports: Use awk to format output
- Create and incorporate new manual pages
- flat-file database tables (rows, columns, delimiters)
- manipulate flat-file tables with join, cut, paste, sort
- Modify and substitute patterns
- Print and delete lines
- Commands: **join, pr**

Command	Result
-i	ignore case
-t	specify field delimiter (most systems, default is the TAB)
-o format	specify output format in the form a list of positional file.field
-1 field1	specify common field in file1 (default: first field)
-2 field2	specify common field in file2 (default: first field)
-a1	outer join; list all lines from file1 even if they don't have a match
-a2	outer join; list all lines from file2 even if they don't have a match
-a1 –a2	list all lines from both files
-v1	output only those lines from file1 that don't have a match
-v2	output only those lines from file2 that don't have a match
-v1 –v2	output all lines from both files that don't have a match

Related filters: colrm, cut, paste

The join command is used to join data from two database files on common information (inner join).

```
join [-i] [-a1|-v1] [-a2|-v2] [-1 field1] [-2 field2] file1 file2
```

- Create the files names.data and products.data with the given content.
- Examine the manual page for join and pr. You may wish to review the commands sort, cut, and paste as well, since they also may help in manipulating tables.
- Contents of file names.data

```
1234:Johnson:Bob
4321:Jones:Jim
5678:Smith:Dave
7777:Brown:Tim
8765:Martin:Peter
```

- Contents of file products.data

```
4321:whatchymacallits:200
4321:gizmos:500
5678:thingamagigs:150
5678:widgets:2000
8765:thingees:3500
```

- Note that the first field contains the same information in both tables, which can be used as a common field. Run the following set of commands, one at a time, note the results and explain what each does.

```
join –t: names.data products.data
join –t: -o 1.2 2.3 2.2 names.data products.data > file6
join –t: names.data products.data > file7
pr –d –h 'Products Purchased' file6
pr –h 'CUSTOMER REPORT' -e:11 file7
```


Command	Result
-h 'string'	use 'string' as the centered header in the output
-e:[CHAR [WIDTH]]	expand **char**acter to tab **width** (default char: tab, default tab width: 8)
-d	double space the output

Reference the online manual for additional options.

- In the next example, the content of the two files is delimited by the default, tab, (don't need to specify the –t option) and the first field in both files is common:
- Contents of file birthdays

```
Al        May-10-1987
Barbara   Feb-2-1992
Dave      Apr-8-1990
Frances   Oct-15-1991
George    Jan-17-1992
```

- Contents of file gifts

```
Al        books
Barbara   music
Dave      chocolate
Frances   camera
George    money
```

- Run the following set of commands, one at a time, and note the results. Explain what each does.

```
join birthdays gifts
join –a1 birthdays gifts
join –a2 birthdays gifts
join –a1 –a2 birthdays gifts
join –v1 birthdays gifts
join –v2 birthdays gifts
join –v1 –v2 birthdays gifts
join birthdays gifts > shopList
pr –h 'MY SHOPPING LIST FOR BIRTHDAY GIFTS' -e:11 shopList
```

- See Example online (Provinces and Capitals of Canada)

FORMATTING OUTPUT AND REPORTS

Objectives

- Understand and use awk command to manipulate data and run simple awk scripts

Commands and Terms

- Command: **xxx**
-

AWK Programming

1. **awk** is a programming language named after its authors: Alfred Aho, Peter Weinberger and Brian Kernighan.
2. General syntax
   ```
   awk [-Fc] [[-f] awkprogram] [datafiles...]
   ```
3. awk can be used to select all records containing a specified pattern or string. The *string* is referred to as an **awk target**
   ```
   awk '/string/{print}' filename
   ```
4. awk can also display either the complete record to the screen or selected fields.
   ```
   awk '/string/{print $0}'filename
   ```
   - The braces referto an action, in this case print.
   - The target and the action are enclosed in single quotes to prevent unwanted shell interpretation of these metacharacters.
   - $0 denotes all fields
   - If you do not specify the action, the default is print , i.e., to the screen.
5. If the field does not use a tab or space character as a field delimiter,you must use -F followed by the delimiter - e.g.,-F; -F: -F,
   ```
   awk -F, '/string/ {print}'  filenename
   ```
6. Each field is referred by its number; first field is $1, second field is $2, etc.
   ```
   awk-F, '{print $1, $2}'filename
   ```

Formating with awk

- Examine this example. Create a data file called employees, using vi editor.

```
409,John Baker,56000,civil engineering
678,Fred Smith,73000,physics
429,Julia Tanguay,47000,computer science
349,Peggy Bantin,67000,physics
268,Mario Hodgkins,55000,programming
```

- Using the file employees you just created, try the following commands and observe the results. Display all records*:*
  awk –F, '{print}' employees
- Display all records of employees teaching in the physics department:
  awk -F, '/physics/ {print}' employees
- Note that we are declaring the field delimiter as a comma,. What is declared as the target? the action? and the file name?
- Display employee name and department from all records
  awk -F, '{print $2, $4}'employees
- Extract all records of employees teaching in the physics department, and display their names and salary
  awk -F, '/physics/{print $2, $3}' employees
- Ignoring letter case.
  awk -F, '/[Pp]hysics/ {print}' employees
- Selecting lines by field value. Display records where the salaries are: 55000
  awk -F, '$3 == 55000' employees
- *Display records where the salaries are less than 55000*
  awk -F, '$3 <' employees

- Display *records where the salaries are more than 55000*
  awk -F, '$3 > 55000' employees
- Display *records of employees working either in the physics or programming departments:*
  awk -F, '/physics | programming/{print}'employees
- Display *records for employees who make more than $55,000 AND work in the physics department:*
  awk '$3>&&$4=="physics" ' employees
- Display *records for employees who makemore than $55,000 but less than $70,000:*
  awk -F, '$3>&&$3<' employees
- Display *records for employees who make more than $55,000 OR less than $20,000*
  awk -F, '$3 > 55000 || $3 < 20,000' employees

COMPLEX AWK SCRIPTS

- awk is a very powerful data manipulation tool, and commands can become very long and complex. In such cases it is easier and more effective to place these commands into a file, and then use the awk command to execute the statements within the file.Using vi editor, create a file called **awktest.awk** and include the following code:

```
/physics/{
    name= $2
    salary=$3
    print " employee name is:"name
    print " employeesalary is:"salary
}
```

- Save and exit the vi editor. Run using awk:
  awk -F, -f awktest.awk employees

- Next, create a file called physicstest.awk and include code as follows:

```
awk –F, ’
   BEGIN {
    print “The name and salary of each employee is: ”
    }
    {
    name = $2
    salary = $3
    salTotal += salary
    print name, salary
  }
  END {
    print “Total salaries for this depart are ” salTotal
  }
’ employees
```

- Save and execute the program. There is no need this time to use the awk command to run the script, as it is already built into your script. Note that within the script, you did not use –f [filename]. The file name is stated on the last line.
- To format the appearance of a report, use the **printf** command, followed by the format string enclosed in double quotes:

```
printf “%-20s %10.2f\n”, name, salary
```


Contents