McGill University
School of Computer Science

COMP-206

Software Systems

Due: January 29, 2009 on WEB CT at 23:55

Operating Systems

This assignment explores the Unix operating system and how its subsystems are inter-related. We will explore three areas of this system: the command line subsystem, the batch file subsystem and the operating system shell.

Question 1: The Unix Operating System Command Line

In order to help you understand how software systems work together, the goal of the course is to build a rudimentary web store. We will begin this here in question one. Many of the assignments will build on each other, each with the goal of completing a little bit more of that web store.

As in all single user software projects we must first reserve some space on the hard drive to develop our web store application. To do this, we will need to create a directory space. Using the Unix command line build the following directory structure:

- Within your home directory create a directory called PROJECTS.
- Within Projects create the directory STORE.
- Within Store create the directories: ARCHIVE, BACKUP, BUILD1, and DOCUMENTS.
- Within Build1 create the directories: SOURCE, DATA and RUNTIME.
- Within Source create the directories: BATCH, C, HTML, CGI, PERL and PYTHON.

As the course progresses we will create our programs in the Batch, C, HTML, CGI, Python and PERL directories. Our database of products will be located in the DATA directory, and all the executables will be saved in the RUNTIME directory. This directory space is considered to be a part of our development environment. Later, we will have to put this on the web (for real) and this will be done in our deployment environment, PUBLIC_HTML (but that will come later).

The next thing we need to decide is the type of products we would like to sell. This I will leave up to you, but we will need to create some databases. Since this is not a database course and since many of you have probably not programmed databases, our database

will be a "flat file" database (i.e. a text file). You MUST use VI to create two of our primary databases (there will be more databases later):

- The first database is called INVENTORY.TXT. It will contain the following information: Product ID number, product name, quantity in stock, and unit price.
- Our second database is called MEMBERS.TXT. It will contain the following information: User ID, user password, user name, security level (1 = supervisor, 0 = regular user), and purchase history (dollar amount sum).

As the names imply the inventory database will record the products your store sells and how many are in stock. The member database contains the names of all the valid users of your store (like eBay).

A flat file database is organized around records and fields. A field is each piece or type of information stored: like user ID or user password. A record is a group or set of fields. It contains all the information about a single user or product saved in your database. In a flat file, the record is one line and commas separate fields. For example, the member's database could look something like this:

1234, abcd, Joseph Vybihal, 1, 500
9987, ddee, Bill Smith, 0, 20000
1072, ABCdef, Sam Smiley, 0, 100

Each row contains all the information about one user. This is called a record. A comma separates each piece of information about the user in that row: ID first, then password, followed by user name, etc. Pressing the carriage return terminates the record. There is also a carriage return on the last record. This leaves us with a blank line at the end of the file. This is okay.

For question 1 do:

- Create the development environment directory structure (record what you did in a text file called STRUCT.TXT). I cannot confirm if you actually created the directory structure using the command-line. You could have done it using Windows and then pretended. To overcome this, I am asking you to type out the commands you would use, **in-order** in this text file, as if you where at the command-line actually doing it (You should ACTUALLY do it – but the TA will not verify this, only the STRUCT.TXT file you upload to WEB CT).
- Using VI, create the two databases INVENTORY.TXT and MEMBERS.TXT and populate them with data (build and save them in the directory DATA). Then send these text files to WEB CT. Again I cannot confirm if you actually used VI but I suggest you do, so that you can be used to it for exam questions.

Question 2: Operating Systems Shell Environments

Every operating system has a shell. The shell is the part of the operating system that processes the commands from the user and displays the user interface. This could be a

command-line shell or a windowed shell. It is the environment in which your program will run. It is also the environment in which you will work in. Every shell has its advantages and disadvantages for the software and the programmer. Look carefully at the Unix and Windows operating systems (if you have a MAC at home do the comparison between Unix and the MAC). Identify the following:

- List all the shells currently on those two operating systems.
- Select two shells from each operating system and describe
    1. how each shell accepts commands from the user.
    2. how each shell returns errors to the user.
    3. Describe in short how we can customize and personalize these shells,
    4. And state what is advantageous and what is disadvantageous in each shell.

Note: To answer this question you may have to do readings. Check the OS manual that comes with your computer, the Internet and the MAN command in Unix or HELP in DOS.

Upload to Web CT a PDF or TXT file containing you answer. Name this file SHELL.txt or SHELL.pdf.

Question 3: Login Shell Scripts

Some shells provide a scripting environment. The scripting environment is a programming language for the shell. These files are not compiled. The shell interprets them. But they give the programmer an avenue where they can automate operating system procedures since all command-line commands can be part of the script program. In addition they also provide a rudimentary control flow language that actually permits you to do some real programming (in the real sense of the technology). Three forms of scripts exist in Unix: the set-up script, scripts proper and the batch script. Batch scripts are only a special form of the general purpose "scripts proper".

One of the most common forms of script programming in Unix is known as BASH.

Set-up scripts are used to initialize or cleanup the shell environment when you login or logout. Shells are initiated and destroyed when you login and out of the operating system. You are also permitted to switch shells at any time. You have the opportunity to select the shell you want to use and through its setup script you can customize its look and feel, and then clean up your session when you logout (if you need to).

- This assignment question wants you to customize your current Unix login set-up script.

As discussed in class there are a few login script files in Unix depending on the shell you are using. Identify the default shell that is in operation during your login and modify its login script in the following ways:

- Customize the prompt.
- Display the directory you have been sent to
- Display the date and time
- List all the users currently logged on
- Finger your best friend
- Alias the ls -l -a command to the word lsa, and
- Launch your web browser application.

Your login script can process this in any order you like.

Question 4: Batch Processing

Batch processing is a special use of scripts for managing and automating the execution of applications. Historically they were created for large mainframe computers that could only execute one program at a time. Setting up a mainframe to execute a user's application took a lot of time. Therefore there was a lot of wasted time (and money) between executions. Normally a "Job" would run for 0.5 seconds and the set-up time was 10 to 20 minutes. This was not cost effective. The solution was to have a special script that in the morning would be built containing the instructions on how to initialize the run-time environment, run the application and perform any cleanup of files and allocated resources for each Job. Then the mainframe would fly through all these applications setting up the execution environment, printing the results and setting up for the next Job, until all were done.

BASH is the modern batch programming environment. These days, we view a Job as the tasks the user wants the computer to process automatically. The user then waits for the work to complete and verifies the results. Bash files are created as needed by the user, not once at the beginning of the day. Let us try a simple version of this.

Inside the BATCH directory create a bash file called VERIFY. Remember to use CHMOD to make it executable. VERIFY will be used to validate the MEMBERS.TXT flat file. It will perform the following tasks; all the activity will be contained within the single VERIFY.BAT file:

Set-up procedure for Job 1:

- Automatically create a directory in STORE called TESTBED
- Copy MEMBERS.TXT into TESTBED

The application to run for Job 1:

- Sort MEMBERS.TXT into SORTED.TXT (use the Unix command)

Cleanup procedure for Job 1:

- Delete the MEMBERS.TXT file from the TESTBED directory

Set-up procedure for Job 2:

- Automatically create a text file called INVALID.TXT that is initialized with the words: PLEASE ENTER INVALID USERS, using the commands ECHO and redirection

The application to run for Job 2:

- Open the file INVALID.TXT with VI. The user will enter names of users that are not permitted to become members of your store. The user will have to terminate VI manually before the batch file continues execution.

Cleanup procedures for Job 2:

- Nothing

Set-up procedure for Job 3:

- Nothing

The application to run for Job 3:

- GREP SORTED.TXT with INVALID.TXT and save the results in FOUND.TXT
- Use MORE to display SORTED.TXT so that the user can visually verify if there are duplicate users
- Use MORE again to display FOUND.TXT to see if any unauthorized users have registered.
- Copy FOUND.TXT and SORTED.TXT back into the directory DATA

Cleanup Job 3:

- Delete all the files in TESTBED and the directory TESTBED itself.

WHAT TO HAND IN

Everything must be submitted to WEB CT before the due date. Remember that WEB CT permits you to hand in up to two days late but there will be a penalty of 5% each day. After that, the WEB CT submission box will close. Please hand in the following:

- For question 1: STRUCT.TXT, INVENTORY.TXT and MEMBERS.TXT. Please remember that STRUCT.TXT contains the command-line session you used to create the requested directory structure. Include ALL the Unix commands you used, in order.

- For question 2: SHELL.TXT or SHELL.PDF that will contain your essay about Unix and Windows shell comparisons (optionally Unix/Mac). The login script that you are using.
- For question 3: The login file you edited.
- For question 4: VERIFY, SORTED.TXT and FOUND.TXT

HOW IT WILL BE GRADED

The TA will use the following instructions when grading your assignment. TA's are often given additional instructions not presented here but compiled from common student questions or found problems in the assignment or adjustments to the assignment / procedures.

- ASSIGNMENT: is worth a total of 20 points.
- QUESTION 1: is worth 5 points.
  - +3 STRUCT.TXT, proportional to the number of correct commands presented
  - +2 Proper structure of both flat files
- QUESTION 2: is worth 5 points.
  - +3 Proportional grading based on the proper identification of shells and features in SHELL.TXT
  - +2 Proportional grading based on the correct commands used in the login script
- QUESTION 3: is worth 5 points.
  - Graded proportionally.
- QUESTION 4: is worth 5 points.
  - +3 Proportional grading based on the correct commands and setup of the batch file
  - +2 the existence and correct format of the SORTED.TXT and FOUND.TXT files. Maker sure that FOUND.TXT actually found someone.