McGill University

COMP 310 – Operating Systems – ECSE 427

Assignment #1

Due: September 28, 2008 at 23:55 on Web CT

Experimenting with Operating Systems

A good operating system course should give you experience in the theory and in the practical nature of using operating systems. This first assignment attempts to do a little of both. In order to be prepared for this assignment you should make sure that you have access to the labs on the third floor of Trottier.

Question 1: What should be in an Operating System?

At times controversies and even legal proceedings have been put forward over what should be part of an operating system. I would like you to give two arguments. Each argument should be one paragraph in length. The first argument is "for" and the second argument is "against" the following: *Should operating systems be integrated with Internet Browsers and Internet E-Mailers?* Not only argue both sides of the point, but also give justification.

WHAT TO HAND IN: **(5 points)** - Two paragraphs with supporting justifications. One paragraph in support with the italicized question while the other paragraph argues against the question.

Question 2: Clustered Systems

In class we discussed a little about a form of operating system known as a Clustered System (maybe you should review the class notes and the section in the textbook at this time). As an example of a Clustered System the example of the SETI (Search for Extra-Terrestrial life Institute's) big dish analysis software that can be downloaded to your personal computer as a screen saver was introduced. This is not an operating system but it is a cluster driven system and a good introductory problem to study. It lets you think about issues that OS designers need to worry about, in addition to programming. The SETI system has a master program at SETI and a screen saver at the client's side. In screen saver mode the program downloads data from SETI headquarters' satellite dish (using the Internet), performs some fast-furrier calculations and then uploads the results back to SETI (again using the Internet). This is called Symmetric Clustering. *What I would like you to do is describe a possible Master algorithm at the SETI site that would control this Symmetric Clustering arrangement*. You will do this by drawing a flowchart (Simply use boxes and arrows. Use a Circle for databases.). Take into account the following problems that often arise in systems like this: (1) data from the dish needs to be recorded on hard disk, (2) execution must be in parallel or at least pseudo-parallel, (3) which client program has what data, (4) the state of the data (solved, unsolved, assigned to a client for calculation), (5) a home computer that never returns a result or takes a long time to return a result (this is not the same), (6) transmission data loss (either from SETI to client or from client to SETI). Answer all these questions using the flowchart. Since you are handing in an electronic document it is probably best to use Microsoft Excel for the flowchart (or something similar). It has a Drawing Toolbar with built-in flowchart shapes. You can place these shapes on the infinite sized spreadsheet. You can then right-click these shapes to add text. If you do not have access to Excel, then use any other program that can save the image in JPG or PDF format so that the TAs can grade them over the web.

WHAT TO HAND IN: **(5 points)** - A detailed flowchart that outlines an algorithm that has access to a database of inter-stellar radio scans, access to a database of active screen saver programs, plus any other databases you think may be needed. The flowchart will describe how this synchronization can take place between master program and slave screen savers. Therefore, the entire algorithm is not actually important but only the synchronization process and the databases it uses.

NOTE: I have stated that you are to use only boxes, arrows and circles. I do not want your flowchart to be so detailed that it becomes pseudo-code but at the same time there should be enough detail that the algorithm does not need to be guessed at. Again, focus your attention on how we can do the synchronization and how databases can be used. These two areas are where all the points in this question will go to.

Question 3: Building your own OS Shell

In this course we will be building our own personal mini operating system called MyLinux. It will be a simulation of a real operating system. We are going to pattern our OS after the Unix OS. This first part of the project asks you to build the Command Line user interface for the OS. You are required to write the program using the C programming language. Your program must be able to execute on the lab computers on the third floor of the Trottier building. Since the C language is almost standard, you can develop the program on any computer but you must recompile and debug on the 3rd floor lab computers so the TAs can run your programs. If the TAs have trouble running your program they will give you zero. Give yourself more than one day to debug the program.

In class a lecture was given about the Command Line Interpreter. You should probably reference that lecture as you determine how you want to build your application.

What you are basically creating for this assignment is an OS shell, the command line portion of your OS shell, the shell's memory and simple script processing. Your shell is a text-based command-line program. You will call your shell: mysh. Make sure not to cheat and have the real OS do the work for you (unless the question specifically asks you to. Assume that the question wants you to write all the code yourself.).

To distinguish your command-line commands from the real OS commands, your OS command syntax requires that the commands be written all in upper case.

Your shell will have the following components:

- At launch time it will accept parameters from the command line (note that your shell is running in the real Unix environment. What this means is that you are not creating an OS but a shell that will run in an OS. In assignment #2 we will start building an OS).
    - –H will not run the shell but simply return a help screen describing the execution syntax of mysh. This specifies the command-line syntax with and without the -H and -V.
    - –V will put the program in verbose mode. This means that at every prompt the program will tell the user what it is expecting from them. In other words:
        - During the command line prompt an additional message will be displayed with the prompt telling the user what to do (i.e.: "Please enter a command. $")
        - After the execution of a command line command the shell will tell the user what it just did as a confirmation that everything went as it should have (i.e. it will repeat the command just input by the user and prompt the user to confirm that they want to execute that command. The user will have to press Y or y for the command to run.).
- It will return an error code upon exit. This will be useful only with the launch parameters. If the user does not input the correct launch parameters: mysh, mysh -h, mysh -v, mysh -h -v, mysh -v -h, the program terminates without displaying the command-line prompt. It will return an error code and before that will display the contents of the -H information as the error message.
    - A result of zero will indicate no error occurred
    - A result of 1 will indicate that an error occurred
- It will implement a rudimentary command line interpreter
    - Just to be different from DOS and UNIX all you command line commands must be in upper case.
    - Displays prompt and optionally extra comments when in verbose mode. Then wait and read in the user's command.

  - The command EXIT will cause the shell to terminate
  - SET VAR VALUE will put VALUE into the shell memory with the variable name VAR
  - GET VAR will display the VALUE stored in the shell memory for the given name VAR
  - VER will return your name as the programmer of this shell, the date created and version number (make something up)
  - CLR will clear the screen
  - PROMPT VALUE permits the user to customize their prompt with the string in VALUE
  - HELP will display all the commands you defined in your shell with its syntax and short one line explanation
  - VERBOSE ON/OFF will either turn verbose mode on or off regardless of –V. The user must type VERBOSE ON to turn it on or VERBOSE OFF to turn it off. VERBOE ON is -V.
  - Any other command you input will not generate an error, instead your shell should pass that command to the operating system using the "system" function call. In this case these commands could be in lower case.
  - The command SCRIPT FILENAME, where FILENAME is the name of a text file, is the command that will cause the interpretation of a script file. You will need to write a script file using a text editor to test this. Your script language will be composed of all the command-line commands described above (including the SCRIPT command – this will not be recursive. It will simply replace the execution of the current script by another.).
- It will implement a simple shell memory
  - As in class, you will have a choice. Either implement your memory as an array of strings or as a linked list of strings. It is up to you. In either case you have to handle freeing memory and when memory gets full. An appropriate message should be displayed to the user of your shell.
  - The user will not be given the ability to change the size of memory if you implement the array version, at this time.

WHAT TO HAND IN: (**10 Points**)

Hand in to Web CT both the source and executable file(s) for this program. Make sure it is zipped if it is multiple files.

HOW IT WILL BE GRADED

- The assignment has 3 questions that total to 20 points
- All questions are graded proportionally. This means that, if your question is 50% correct then you will get 50% of the grade.
- The questions are graded according to:
  - Did you follow the instructions?
  - Is it correct?
  - Did you hand in what was asked for?
- 5% of the grade is removed for each day late up to 2 days. After 2 days the assignment is not accepted.