ECE 329 HW #1

Write a simplified simulated UNIX shell. Your console-based application should support the following commands and syntax:

- *login name*
  logs in user as *name* (no password required). *root* should be the default user. For simplicity, limit user names to a maximum of 14 characters.
- *whoami*
  prints user name
- *date*
  prints the current date and time, in the standard UNIX format
  Example: Fri Aug 17 14:16:31 EDT 2007
- *pi n* (No, this is not a real UNIX command)
  prints the value of π after the *n*th iteration of computing the following formula:
  $\pi/4 = 1 - 1/3 + 1/5 - 1/7 + \cdots + (-1)^n / (2n+1) + \cdots$
- *exit*

Your shell should ignore extra whitespace typed by the user, and it should print appropriate error messages upon erroneous user input. When in doubt about desired behavior, run an actual UNIX/Linux shell.

For *pi*, be sure to store the value as a double and print 16 digits after the decimal point (which is the accuracy of a double), so that all the relevant digits can be seen. Do not compute π using alternate algorithms – the formula above has been chosen specifically because of its slow convergence, to make the display meaningful even with a large number of iterations.

Only for the *pi* command, if the user types the ampersand (&) at the end of the line (with whitespace before it), then the command should proceed in the background by spawning a separate thread. Multiple background commands will cause multiple threads to run concurrently, with no limit on the number of concurrent commands. Similar to UNIX, when the background command begins it should print a unique number (from a global counter) identifying the job, and when the copy completes it should print 'Done' along with the identifying number and the command itself.

Hint: Be sure to flush the output buffer by calling *fflush()* whenever you call *printf()*. Otherwise the multiple threads will interact in unpredictable ways.

Note that no synchronization between threads is required for this assignment.

The following library routines may be helpful:

- gets and sscanf in <stdio.h>
- strftime in <time.h>
- CString and CTime (MSDN)
- CreateThread (MSDN) – creates a thread
- CloseHandle (MSDN) – cleans up an object (e.g., thread)

- Sleep (MSDN) – causes thread to sleep for a specified period of time, allowing other threads to run

Separately, answer the following problems in Chapter 1 of the textbook (Tanenbaum, *Modern Operating Systems*, 3rd ed.): 1, 2, 7, 10, 14, 20, 22, 23, 25, 28.