Programming Assignment 1 – Developing a Shell

<u>Assigned</u>: Tuesday, 2/14/2005, 9pm
<u>Due</u>: Friday, 2/24/2005, 9pm (per email, as instructed below)

Part 1. Read chapters 1, 8 and 9 of the required textbook by Bruce Molay (2002) *Understanding Unix/Linux Programming: A Guide to Theory and Practice (1st Edition).* Try out (compile and run) all the example codes of chapter 8 and make sure you understand how they work. Sources can be downloaded from the book's companion Web site (link in the CS 446/646 Web page). You might have to modify a few details in the code, depending on the platform and compiler you are using (for example, you might need to add type `void` to the main and other functions, and remove the function declarations from within the main, etc.) This part will not be graded, it is only intended to prepare you for Part 2. Do not turn in a report about Part 1.

Part 2. Complete the "Programming Project One" as described in the required textbook by William Stallings (2004) *Operating Systems: Internals and Design Principles (5th Edition)*, on pages 153 to 156, with the following modifications:

Shell Properties & Code Design

- ✓ Implement only properties 1. (all items from i. to ix.), 2. and 6., as described in the book on pages 153-154. ***Graduate students only (extra credit for the others):*** implement property 5., too. Everyone: do *not* implement properties 3. and 4.

- ✓ To complete this assignment you may reuse the source code from Bruce Molay's book as your base code and customize it, or write your own code. Keep in mind that you must follow Stallings' requirements, not Molay's features. Do not use code from Molay's book that is not strictly necessary for this assignment: there will be a penalty for copying and pasting unnecessary code. As for the original part of the code that you are going to write, do not show, exchange or copy code with anyone, and do not look for a solution on the Web. Plagiarism *will* be detected and severely penalized. This must remain an individual effort.

- ✓ Whether your code is adapted from Molay or is originally yours, please keep clean code organization, layout and coding conventions, these will also be an important part of the grade. A few guidelines (reminders from your freshman year): (a) *Never* duplicate blocks of code: always try to parametrize, factorize and centralize your logic in unique places (functions, classes, etc.). (b) Break up your code into functions: do not keep long and detailed stretches of code; break them up into subparts and put them into functions. (c) Create a good design based on layers, modules and interfaces: write .h (header) files that contain the functions (in C) or public class members (in C++) that other modules are using; if you have general purpose utility functions, hide them away in a utility layer. (d) Use the keywords offered by the language: instead of `if (x==1)` ... `else if (x ==2) else` ..., use a `switch`. (e) Use consistent tab indentation in the layout. (d) Insert plenty of comments (banners at the beginning of modules and functions, one-line remarks, etc.) and empty lines to improve readability, etc., etc.

Project Requirements, Submission & Required Documentation

- ✓ Follow all the requirement and submission guidelines *exactly* as prescribed in Stallings' book on pages 154 to 156. The only exception is that you may ignore any mention of "i/o redirection" or, for the undergraduate students, "background program execution", since you are not asked to implement these features in the present assignment. Notice that Stallings' guidelines include, among other things: appropriately structuring and properly commenting the code (3.), *not* including object code or executables in your submission, only source code and documentation (5.), naming the makefile exactly `makefile` (6.), naming the readme file exactly `readme`, and writing a reasonably detailed UNIX-like readme documentation (2.). Your documentation should describe the problem's background, the program's usage and parameters, and a sample of a typical outcome.

- ✓ Your code *must* compile without any errors or warnings and run properly under Gentoo Linux, the system used in the ECC lab (check their Web site for hours). You may develop and test your programs on your own UNIX machine, but it is your responsibility to ensure that they also work properly on the Gentoo installation of ECC, under the default ECC Lab shell. There will be a heavy penalty if they don't.

- ✓ Place all necessary files and documents in a compressed tarball using tar and gzip (`.tar.gz` or `.tgz`) or, alternatively, a zip file (`.zip`) (please do not use the `bzip2` format). Email this file to the instructor doursat@unr.edu before the deadline above. Do not turn in printouts in any form. Late assignments will be marked down according to the late policy published on the course Web page.

Good luck!