Programming Assignment 3 – The Dining Philosophers Problem

Assigned: Monday, 10/10/2005, 2pm
Due: Monday, 10/24/2005 before 2pm

Part 1. Read chapter 14—sections 14.1 to 14.5 *only*—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 featured in sections 14.1 to 14.5 of chapter 14 and make sure you understand how they work. You are not required to read sections 14.6 and 14.7 for this assignment. Source files can be downloaded from the book's companion Web site (link in the CS 446/646 Web page). 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. Write a program that implements the so-called "Dining Philosophers Problem" (DPP) described in section 6.6 of the required textbook by William Stallings (2004) *Operating Systems: Internals and Design Principles (5th Edition)* on pages 275 to 279, as follows:

Specifications

- ✓ Your program must accept three command-line arguments in this order: (a) the number of philosophers, (b) the number of seconds (possibly decimal) that each philosopher takes to think or eat, and (c) the total number of think-eat cycles that each philosopher will go through (so that the program eventually ends).

- ✓ Each philosopher will be represented by one separate thread. A simple output statement such as `printf` will be sufficient to specify the action of each philosopher when she/he is about to think or eat (this message must contain the philosopher's identification). The thinking and eating times will be simulated using the `sleep` function with the number of seconds given in argument.

Implementation

- ✓ You must implement the *monitor*–like version of the DPP: a detailed template of this solution is presented in Fig. 6.14 of Stallings (do *not* use pure semaphores in this assignment). However, as with Molay's source examples, you must write your program in C/C++ and may only use the POSIX Thread Library `pthread`, without explicit support for monitor objects. Therefore, refer to section 5.4 of Stallings, which explains how synchronization in a monitor actually relies on *condition variables*, and section 14.5 of Molay, which precisely illustrates the use of condition variables with the `pthread` library in a manner equivalent to a monitor.

Project Requirements, Submission & Required Documentation

- ✓ Follow all the requirement and submission guidelines *exactly* as prescribed in Stallings' book on pages 154 to 156, except that you will replace the name of the executable `myshell` with `dpp` and, naturally, any mention of "shell" and shell features with the

present DPP program. I remind you that these guidelines include, among other things: appropriately structuring and properly commenting the code (3.), not including object code or executables in your submission (5.), naming the makefile exactly `makefile` (6.), naming the readme file exactly `readme`, and writing a reasonably detailed readme documentation (2.).

- ✓ Your code *must* compile without 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 code on your own UNIX machine, but it is your responsibility to make sure that it is also going to work properly on the Gentoo installation of ECC, under the default ECC Lab shell. There will be a penalty if it doesn't.

- ✓ Place all the 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 within two weeks, before Monday, 10/24/2005 at 2pm. 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.

- ✓ 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. The only exception is that your code may bear some family resemblance with Molay's source code (such as `twordcount4.c`), as you might decide to use his code as a starting point—but naturally, the end result will perform an entirely different function and, accordingly, should look very different.

- ✓ Please keep clean code organization, layout and coding conventions, as these will also be an important part of the grade.

Thank you and good luck!