**CMSC 412**
**Fall 2010**
**Lab 1**
**September 8, 2010**

This lab is designed to familiarize you with the Windows Research Kernel (WRK) and its development environment.

We've previously given you the target (Windows Server 2003 SP1) and development (Windows XP) virtual machines, and we've helped you set-up the kernel debugger in the discussion sections. You're now going to modify and make a new Windows Kernel.

## Due Date

**This project is due at midnight September 15th** via the CS submit server. We'll be posting the submit file on the forum shortly.

Late projects will only be accepted until September 17th with a 10% deduction for every 24 hrs the project is late.

Partial credit will be given – it behooves you to turn in a project.

## Task 1

This task will ease you into building and debugging new kernels. Part of any real world work is reading other's code and understanding it. Your task is to determine the function or functions where processes are created. Modify this/these function(s) such that a debugging statement is printed it out in the debugger.

**Turn in** the modified C file, and a transcript of your debugger session.

## Task 2

This task is more challenging than the first task, but still not significantly difficult. It is again designed to familiarize your self with Windows kernel debugging.

1. Create a new file "lab1.c" in the appropriate directly under C:\WRK-v1.2\base\ntos, and modify the build files needed to ensure lab1.c is built and linked into wrkx86.exe.
2. Create a new function named *void PsMyEnumerateProcs(void)* which enumerates all of the active processes in the system and prints out the same information as the '!process' command in the debugger for each active process. For example (from page 296 of Windows Internals 4th Edition) the information you need to print for EVERY active process in the system is,

```
lkd> !process

PROCESS 8575f030 SessionId: 0 Cid: 08d0     Peb: 7ffdf000 ParentCid:
0360 DirBase: 1a81b000 ObjectTable: e12bd418 HandleCount: 66. Image:
windbg.exe
```

3. Call your newly created function from the debugger using ‘.call’.

You may use ANY existing code within the kernel to help you complete this task.

**Turn in** lab1.c and all modified files required to build along with a transcript of your debugger output after running your new function.

## Grading

This lab is worth 10% of your total class grade. Grades will be assigned as follows:

1. Correctness: 70%
2. Coding style: 30%

Correctness includes stability, e.g. no blue screens, and displaying all information about all active processes. We will provide a test script for the debugger for testing in the forum later this week.

Coding style is important especially when others need to read and modify your code. Note the preambles for each function in the NT source. Use a similar preamble along with appropriate comments, and you’ll receive full credit for coding style.