Project 3: Context Switch and Non-Preemptive Scheduling  
CSCI 423 Fall 2009

Due: Thursday Oct 1, 11:59PM

Students enrolled in CSCI 423 should complete this project with your partner. Students enrolled in CSCI 501 must complete this project independently. Each student/group must complete this project without the aid of any other student/group or any other source of Xinu code. Discussing high-level OS and Xinu concepts with other members of other groups is allowed provided that the specifics of how to code the project is not discussed.

Always follow the submission directions at the end the this document. Following directions is part of your grade. Any variance from the stated directions may result in zero credit for the project. Note that you can re-submit any project after the due date/time for a 30% penalty.

Context Switch and Non-Preemptive Scheduling:

Your task for this assignment is to modify your Xinu kernel by adding the ability to create and manage processes. When finished, your Xinu kernel will allow multiple processes to take turns executing on the cpu.

The operating system is responsible for knowing about all of the processes that currently exist in the system. Most computers have relatively few cpus used to execute a relatively large number of possesses. The WRT54GL router your operating system uses has only one cpu and Xinu supports multiple processes. Only one process can execute on the cpu at any given time. Any multi-process operating system contains the functionality needed to remove a process from the cpu by pausing it and storing its state, as well as, un-pause a previously paused process and allow it to execute.

The act of removing a process from the cpu and putting another process on the cpu is called a context switch. An operating system’s context switch function typically must be written in assembly language because the low level manipulations it performs are not modeled well in higher-level languages.

For the assignment, you will need to write the C code necessary to create and manage Xinu processes and the assembly code necessary to context switch Xinu processes in your kernel. If you have not worked in Mips assembly language before, there are many helpful resources available online. Despite its low-level nature, a context switch does not require complex instructions. Our context switch can be completed using only arithmetic opcodes, and the load (lw), store (sw), and add (addiu) opcodes.
Objectives: After completing this assignment students will:

- understand the Xinu operating systems process control structure.
- be able to design and implement assembly code for a context switch on a MIPSEL architecture.

Preparation:
Untar the new project files in a fresh working directory:
```
tar xvzf ~ruth/pub/xinu-proj3.tgz
```
Copy your synchronous serial driver file (`kprintf.c`) into the `system/` directory.

Xinu Source Code:
The new source files include two new headers, `include/proc.h` (definitions for Process Control Blocks and a process table), and `include/queue.h` (definition for a process ready queue), as well as trivial updates to `include/kernel.h`.

The new source files also include:
```
- system/initialize.c  Updated initialization for Project 3.
- system/main.c       A "main program" for testing scheduling.
- system/queue.c      An implementation of the queue data structure.
- system/create.c     A partial function for creating a new process.
- system/ctxsw.S      An incomplete assembly routine for switching process contexts.
- system/ready.c      A function for adding a process to the ready queue.
- system/resched.c    The primary scheduling code, equivalent to `yield()`.
- system/getstk.c     A rudimentary function for allocating a stack for a new process.
- compile/Makefile    Updated rules for compiling XINU.
```
The `create()` and `ctxsw()` functions are incomplete and must be filled in. The major locations are marked with "// TODO...". File `system/initialize.c` contains code to test your context switch with three processes, each of which prints a process ID and then yields. Once your context switch and creation functions are working, you will see these three processes take turns running.

2
Submission:

Before submitting your project please type `make clean` from your compile directory. This removes the compiled files leaving only the source files. This is necessary to save space on the server.

To submit your files ON-TIME use the following command:

```
turnin -c csci423 -p proj3 xinu-proj3
```

To submit your files LATE use the following command:

```
turnin -c csci423 -p proj3-late xinu-proj3
```

Any variance from the following requirements may result in **zero credit** for the project.

- Your project MUST be submitted using the `turnin` command on `xinu.cs.olemiss.edu`.
- You MUST include the entire directory `xinu-proj3`.
- Your project MUST compile and run on `xinu.cs.olemiss.edu` using all of the original files with the exception of your `create.c` and `ctxsw.S` files.