SE350 – Project Final Report

Winter-2012

By Nabil Drawil
ndrawil@uwaterloo.ca
Room E5-5109

Deliverables

Project Parts	Requirements	Submissions
✓ RTX Project P1	Memory management (data structure + APIs) Specified processes in the SPECs as well as few testing processes	Source code + Documentation
✓ RTX Project P2	Simplified version of the RTX	Source code + Documentation
➡ RTX Project P3	Final version of the TTX	Source code + Documentation
RTX Project P4	Final project documents	Documentation

Software Design Document

- Requirements
  - A structural description of the design
  - Functional description of all procedures
  - Implementation, testing, and measurement plan

Format

- Main body - not more than 30 pages
- Use standard formatting
  - Title page
  - Table of content
  - List of figures and tables
  - Organized into sections and subsection
- Use figures and tables

Document Outline

- Introduction
- Global Information
- Primitives
- Software Interrupts
- Hardware Interrupts
- System Processes
- User Processes
- Initialization
- Implementation / Test Plan

Introduction

- Basic information and overview: we are building an OS, this is the design doc, etc.
- Overview of your operating system's structure (largely determined by the project description)

Global Information

- List and define the meaning of:
  - Data structures (eg. how you're storing process queues, lists of memory blocks, etc.)
  - Constants (process IDs, states, etc.)
  - Global variables
- Memory map: A diagram showing the LPC1768’s Memory Map, and where in memory various elements of your operating system will reside.

Primitives

- Basic services provided by the microkernel
- Pseudocode
  - No C code!
  - May use the style from ECE-250 / SE-240
  - Should be detailed enough to allow TAs to understand your intended implementation

H/W, S/W Interrupts

- Areas where switch between user and supervisor mode
- Software interrupt section essentially describes the interface by which processes invoke the kernel services
- Describe the path of execution from the interrupt vector to each i-process
- Describe i-processes – pseudocode, no H/W details

System & User Processes

- Describe the purpose of each process, assumptions, requirements, dependencies on other processes
- Describe the format of the messages the process can receive / send
- Pseudocode for each process

Initialization

- Outline the steps the OS will take when it starts execution
- You are required to make certain system parameters tunable: how are these initialized?

Implementation / Test Plan

- WHO did complete WHAT code and WHEN?
- WHEN and WHAT code was integrated?
- WHEN was various functionality working?
- HOW did you test your code?
  - Manual test cases? An automated test suite?
  - Describe your test cases!
- WHERE and HOW was your code stored?
  - Source control? Stored on a central computer?
- HOW would your code be compiled?
  - Dependency graph of your build system? – Makefile
- Development tools