Deverick Simpson & Yuekun Yang

Report due 1:30pm October 17th, 2014

CSE 351- OS Kernels

Project Part 2

Thread Management of Prototype OS

Introduction

To continue our understanding of the OS kernel, the objective of the assignment is to design and implement a thread scheduling subsystem to allow our prototype OS to create, schedule, maintain, and terminate multiple threads at runtime. The thread scheduler will mimic some of the main features used within the pThread library of Linux. Groups are provided a DE2 NIOS board to implement the design. The DE2 board provides an alt\_alarm() function that will be manipulated within the project to enable a scheduling system for the purpose of concurrent threading.

To avoid compatibility between machines, the programming environment will primarily be within the CSE server on campus.

Project Management

To begin the project, partners individually researched the materials related to threads and set up environments on local machines. An attempt at pair programming was made but scheduling conflicts required partners to divide the work.

Strategy for Solution

Trial and Error

Summary

In this project, our goal is that creating 8 threads at run time, scheduling them by using Round-Robin and use interrupt handler to operate them. Since we can’t use pThread library, we have to build our thread creating and joining functions. We also need to consider about how should use TCB table and deal with the context when we change thread. Moreover, how to interact assembly files and our handler file is also another changeling part. This project due November 10, 2014, we start it almost 20 days before due day. In the beginning, we just did some research for this project. 2 or 3 day after that, we got a basic overview of it. Then we started to implement it. We scheduled very well and always worked together. We tried pair program model and separate program model. Usually, we would take a break in each 2 hours in order to have a higher efficiency. After we finished, we push what we changed into Github, and check and try to understand those changes. Deverick implemented the most part of code, and Yuekun wrote the paper. During this period, we got a lot of help from the class and discussed a lot with other students. It helped us understand this project deeper. Our risk anticipation is the debugging part. S we just start to debug in the end and didn’t wrote any test case before this point. Also, since we built this project from zero, we just tried to implement according to our logic. Therefore, there are a lot of errors when we start to debug. Some of them are very small mistakes. However, to some of them, we have to change the design a little bit. It took us a long time to fix them. Of course, during the debugging time, we learned some knowledge that we didn’t think about during the implementation time.

In this project, we used the myThread function, which is given to build our creating and joining function. We also build a TCB structure to store the information of individual thread. Next, in order to store 8 threads and schedule them by using Round-Robin, we built a TCB queue and several functions to operate this queue. Our idea is that dequeue the first thread in this queue when interrupt occurs and global flag is active, and let it run a very short time. Then, put it back to the queue if it could finish at that shot period. In order to disable and enable the interrupt handler, we defined them by using assembly codes.

In this project, we finished all 5 steps. We built 8 threads in the run time, active the alarm for interrupt handler, scheduled them and switch their context properly. Overall, this project is difficult, however, during the struggling time, we understand what we learned in class deeper, such as scheduling and what is the function of an operating system. Since we start it very early, we had enough time to do this homework. We believe we’ve done is very good in our thought. Moreover, we solved it in a proper way. Thus, our solution is not very complicated and complex. The project is the most helpful tool for us. We followed the instruction step and step. It helped us work on the correct direction all the time.

create -> store -> enable the alarm->