CS 452 Assignment 1

Felix Fung (20351771)
Dusan Zelembaba ()

May 21, 2013

1 How To Run

```
> load -b 0x00218000 -h 10.15.167.4 "ARM/f2fung/main.elf"
> go
```

2 Submitted Files

Files listed here can be found under `/u0/f2fung/cs452/a1`

Listing Format:

`filename (md5hash)`
description

2.1 Header Files

`context_switch.h ()`
Function definitions for compiler to use our assembly functions.
`queue.h ()`
Header for our queue implementation.
`scheduler.h()`
Function definitions for scheduler.
`syscall.h ()`
Function definitions for syscalls.
`task.h ()`
Contains Task structure and function definitions.

2.2 Source Files

`context_switch.s ()`
ARM assembly used to switch in and out of tasks and kernel mode.
`kernel.c ()`

The infinite kernel loop. Interrupts are processed here and tasks and scheduled.
`main.c ()`
Beginning of execution, described below.
`queue.c ()`
A simple queue implementation.
`Makefile ()`
Our makefile.
`scheduler.c ()`
Our scheduler.
`syscall.c ()`
System calls.
`task.c ()`
Creating and managing tasks.

2.3 Test Files

`basic_test.c ()`
This is a basic test
`run_tests.h ()`
Header for run_tests.c.
`run_tests.c ()`
Calls all our tests.
`test_helpers.h ()`
Header for test_helpers.c.
`test_helpers.c ()`
Test helpers. Asserts and other things.

3 Program Description

3.1 Main

This is where the kernel's execution begins. We first set a known address ($0x28$) to point to our interrupt handler.

3.2 Data Structures

3.2.1 Task Id Provisioning

We use a queue which is initialized with the universe of available task ids (0-1023). These task ids point into an array of Task structures. We recycle task ids by pushing returned task ids into the back of our queue.

3.2.2 Scheduler

We use 4 queues, one for each priority we define. Queues are implemented by circular arrays.