Project 2: Process Scheduling in Linux

For this project, you will implement a round robin scheduler in the Linux kernel that you can select using a modified system call. You will also add a new system call to set the default timeslice in your new scheduler. **You are allowed to work in groups of 2**. Each group will also submit a short report describing their implementation. Other details are in the handout linked below.

Submission Due: April 26, 11:59pm

PROJECT MATERIALS

1. Project Description

The starter code is available here. You should download the starter code from this web-site and transfer the files to your KVM image.

1. Starter Kernel
2. Kernel Configuration File
3. thread_runner
4. Kernel Build Script
5. Kernel (Re-)install Script
6. .vimrc for Editing the Kernel

GETTING STARTED

You will use the KVM setup described in the lab to build and test your project. You can use the image you created in lab earlier this semester. You will need to manually transfer the files listed above to your KVM.

The kernel configuration file should be located in /configs/kvm-kernel.config. The source code for the thread_runner program is provided (thread_runner). Additionally, we have provided scripts for building (/bin/kvm-kernel-build) and installing (/bin/reinstall-kernel) the kernel for this project. Finally, we are including a customized dot_vimrc file (/root/.vimrc) for editing code with the kernel indentation style. You should review the Lab 8 (Setting up Your KVM Environment) and Lab 10 (Using GDB in Kernel) lab for information on how to build and install kernels on your image.

SUBMISSION

When you have completed your project, you will submit your report as well as a *patch file* with your implementation to the TA's email account. Follow these instructions to generate your patch file:

1. Log into your KVM and create a directory to hold the original kernel we distribute with this assignment and cd into this directory:

```
mjantz@eecs678-kvm:~$ mkdir orig
mjantz@eecs678-kvm:~$ cd orig/
```

2. Copy the kernel tar file we distribute for this assignment to this directory. You can download the original kernel source from within your KVM by doing:

```
mjantz@eecs678-kvm:~/orig$ wget http://www.ittc.ku.edu/~heechul/courses/EECS678/F14/project2/linux-2.6.32.60.tar.gz
```

3. Untar the original kernel source:

```
mjantz@eecs678-kvm:~/orig$ tar xvzf linux-2.6.32.60.tar.gz
```

4. Next, you will need to clean out the custom kernel that you implemented for this assignment. Note, this will remove many built files and may cause a longer build time (10 to 15 minutes) the next time you build your modified kernel:

```
mjantz@eecs678-kvm:~/orig$ cd /home/mjantz/kernel/linux-2.6.32.60/
mjantz@eecs678-kvm:~/kernel/linux-2.6.32.60$ make-kpkg clean
```

5. Now, create the patch file using the diff command. Please name your patch file: $(STUDENT_ID)_scheduling.patch:

```
mjantz@eecs678-kvm:~/kernel/linux-2.6.32.60$ cd /home/mjantz
mjantz@eecs678-kvm:~$ diff -Naur orig/linux-2.6.32.60 kernel/linux-2.6.32.60 > 1334070_scheduling.patch
```

   Examine the generated patch file to make sure that it contains only the differences that you have written into your modified kernel tree and that it is not too large.

6. Finally, copy your patch file to your EECS account so you can email it to me:

```
mjantz@eecs678-kvm:~$ scp 1334070_scheduling.patch cycle2.eecs.ku.edu:~/Desktop/
```

FAQ

1. I do not understand how to use the kernel linked list. How does it work?

   A. For the purposes of this project, you do not need to understand exactly how the linked list API works -- just how to use it. This website gives an overview of how to use the list and also explains how it works.