ECS 150 Programming Project #2

Due date: 11:59 PM, May 18th, 2007

This project will extend your work from Programming Project #1. You will be modifying the FreeBSD ULE scheduler and also adding one more (simple) system call.

Part 0: BEFORE YOU START (VERY IMPORTANT)

Before you start writing any code at all, it is very important that you perform the following steps:

```
cd /usr/src/sys/i386/compile
\rm -r ECS150
cd /usr/src
cp -r sys sys.orig
```

If you named your custom kernel configuration something other than "ECS150" for project 1, substitute the proper name in the commands above.

Part 1: Fix & Update Project 1 Code

Your first step in this assignment will be to get your Project #1 code working 100% correctly. An extended test program has been added to the website; your code should now pass all tests in that program without errors and without kernel crashes.

Furthermore, Project #1 did not specify where inside the kernel `proc` structure your `p_category` and `p_message` fields should be defined. For this project, please define `p_category` in the section of the process structure which is inherited from the parent process and define `p_message` in the section of the process structure that is zeroed upon process creation.

Next, update your `setcat()` system call from project #1 so that it only accepts category values that are either a power of 2, between 1 and 512 (inclusive), or the value 0. Invalid category values should result in -1 being returned to the user process and no change being made to a process' category.

Finally, make sure that all of your system calls are implemented inside the same `.c` file and that a single Makefile is used. Some of you already did this for project #1, but if you used separate `.c` files for each system call, they now need to be merged. Please call your `.c` file `syscalls.c`.

Part 2: Configure Kernel to use ULE Scheduler

FreeBSD supports two different scheduling algorithms. The original scheduler (4.4BSD) was described in lecture and the new scheduler (ULE) was described in discussion section. ULE should be the default for new FreeBSD installations since version 5.2, but you'll need to make sure your system is actually configured this way (the provided

VMWare image is configured to use 4.4BSD rather than ULE). To do this, open the file `/usr/src/sys/i386/conf/ECS150` in your favorite text editor and search for one of the following lines:

```
options   SCHED_4BSD
```

or

```
options   SCHED_ULE
```

If you find the first line, comment it out and add the second. If you find the second, no modifications are required.

Part 3: Add a "scheduling mode" option

Your first modification to the scheduler should be to add an integer value which indicates the "scheduling mode" of the system. This value should be initialized to 0. Add one additional system call to your `syscalls.c` file with the following signature:

```
int set_sched_mode(int mode);
```

This system call should set the kernel's scheduling mode to the value passed as the `mode` parameter and return the previous scheduling mode. If `mode` is a negative number, return -1 and leave the scheduling mode unchanged.

Hint: Declaring the variable inside the ULE scheduler (`/usr/src/sys/kern/sched_ule.c`) will allocate memory for it in the running kernel. However you will also need to add an "extern" declaration to `/usr/src/sys/sys/sched.h` (and include this header in `syscalls.c`) so that the compiler is aware of it. E.g.

```
extern int foo;     /* assuming "int foo" is declared inside ULE */
```

Part 4: Modify process scheduling

Usually any process in the `READY` state is eligible for scheduling, but we will be adding an additional piece of criteria. The scheduling mode of the operating system (which you defined in part 3) is to be used as a bitwise mask of process categories that are *not* eligible to run. Specifically, a process with category C should not be scheduled when the operating system has scheduling mode M if $C\&M \neq 0$ (note that & is the bitwise-and operator, not the && logical-and operator).

> **Example:** We have three processes A, B, and C with process categories 0, 1, and 2 respectively. The system starts in scheduling mode 0, so any of these processes are eligible for execution. If the system switches to scheduling mode 1, processes A and C are still eligible for execution, but process B is not. If the system then changes to scheduling mode 3, only process A will be able to run since $1\&3 = 1 \neq 0$ and $2\&3 = 2 \neq 0$.

If a change in the system scheduling mode makes the currently running process ineligible, that process is allowed to continue execution until the end of its timeslice, or until it blocks.

Part 5: Develop a test program

The final step of this assignment is to develop your own driver program to exercise the scheduler and make sure that it's working as you expect it to. No additional test program will be provided for this assignment. How you write this test program is up to you... there are many possible approaches. Call your test program `driver.c`. Please indicate your reasoning behind various tests in the source code comments.

Submitting Your Assignment

When you are convinced that your assignment works, start by running the following commands:

```
cd /usr/src/sys/i386/compile
\rm -r ECS150
cd /usr/src
diff -ur sys.orig sys > /root/project2-lname.diff
```

Replace the "lname" part with your last name (in lowercase). If you are working with a partner, replace "lname" with both of your last names, separated by a dash (e.g., "/root/project2-smith-jones.diff"). You should now have a text file in `/root/project2-lname.diff` that shows the exact changes you made to the kernel source tree.

Write a README file that gives your name, the name of your partner (if applicable), and a general description of your project. Indicate what parts of your project do or do not work and the general approach you took to modifying the scheduler. Also describe the logic behind your test program.

Transfer `project2.diff`, `syscalls.c`, a Makefile for your system calls, and your README file to a system that you can send email from. Inspect all of the files to make sure they are not corrupt, then email them as attachments to `roper@cs.ucdavis.edu`. Use "ECS 150 Project 2 - LASTNAME" as the subject of your email (replace LASTNAME with your last name). There is no need to tar or zip your files for this assignment; just attach them individually to the email message.

Sources of Information

Some information that will help you with this project will be covered in the May 7th discussion section. You can also find a paper on the design of the ULE scheduler on the course website. Make use of the course newsgroup and instructor/TA office hours if you have additional questions.

Please start this assignment early. Although this may be a "small" assignment in terms of raw number of lines you have to write, the concepts are complex and require critical thinking than the first assignment did.