Section 6: Lab 4

CSE 451 18SP

Announcements

- Lab 4A Due May 8th
- Lab 4B Due May 15th
- Lab 4C Due May 22nd
  - Only needs to pass the tests at the Lab 4C Deadline
  - Other deadlines are just checkpoints (but still turn something in)
- Lab X proposals due May 18th
- We have lecture again on Friday (Guest Lecture!)

Review of processes and switching

- When you have more than 1 cpu, you can run multiple processes at the same time
- Process = Environment, these 2 words mean pretty much the same thing for our purposes
    - There is also “thread” which also means kind of the same thing, but the context of thread can be different
- Processes need to keep track of their own stack and cpu registers when they switch
    - Question: How does JOS accomplish this with its environments?

Lots of new acronyms

- APIC - Advanced Programmable Interrupt Controller
  - A Programmable Interrupt Controller is a device that is used to combine several sources of interrupt onto one or more CPU lines, while allowing priority levels to be assigned to its interrupt outputs. From Wikipedia
- SMP - Symmetric Multiprocessing
- BSP - Bootstrap Processor
- AP - Application Processor
- LAPIC - Local APIC
  - The LAPIC units are responsible for delivering interrupts throughout the system
- MMIO - Memory Mapped I/O

Locks and Fun with Concurrency

- Locks: Used to stop multiple processes from executing the same code
  - In Lab 4, the locks are known as Spinlocks (see spinlock.c/h)
- With concurrency comes the fun issue of: Non-deterministic bugs
  - Where your code may work sometimes, and crash others
- Good Luck!

Lab 4 Part A: Multiprocessor JOS

- Exercises 1-4 are about setting JOS up to be able to use multiple CPUs
- The code for these exercises is not particularly long, it is mainly about understanding what is going on and what you are setting up

Big Kernel Lock

- Though you may have multiple CPUs running multiple user environments, you only every want one environment running the kernel at any time
  - Why is this important?
- JOS uses a single big lock for the kernel, to make sure only one process is executing it at a time
- Exercise 5 is about locking/unlocking in the Big Lock certain places

Cooperative (Round Robin Scheduling)

- Exercise 6 is about setting up cooperative scheduling for environment
- Environments use a system call to yield control of the CPU, allowing the scheduler to let another process run
- Env_state is very important in determining which process to run next. Do not want the case where 2 CPUs are running the same environment
- Question: Why can’t we always use cooperative scheduling? Why do we need to change it in part C?

System Calls for Environment Creation

- Allowing user environments to create new environments
  - Vital to the operation of any OS
- For Exercise 7, system calls are implemented for a very primitive fork wastes a lot of extra memory
  - dumbfork

Park B: Smart Fork (Copy on Write Fork)

- Will go over it next week in section