Lab0 - KOS

Objective:
- Implement basic functionalities for KOS, an operating system for a simulated MIPS machine

Tasks:
- Implement read() & write() system call for reading/writing to console
- Make KOS load and execute a program, with appropriate argc & argv

Reminder: For this lab, you need to use cs170.cs.ucsb.edu for development
simulator.h

- char *kos_argv[]
- void exceptionHandler(ExceptionType which) - implement this
- void interruptHandler(IntType which) - implement this
- int load_user_program(char *filename) - load user programs into memory
- int run_user_code(int registers[]) - give CPU control to the user program and wait for exceptions/interrupts
- char console_read()
- void console_write(ch)
- int examine_registers(int buf[40])
- int noop() - idle the machine and wait for interrupts
- register macros
- system call numbers
demo starter code
Cookbook

The cookbook contains a step-by-step description of a way of implementing this lab. It is not required that you do things this way, but it is in your best interest to follow this set of guidelines.

Make sure you understand what you are doing at each step!

https://sites.cs.ucsb.edu/~rich/class/cs170/labs/kos_start/cook_book.txt
naive solution

- load_user_program()
- run_user_code()
- wait for syscall
- examine_registers()
- repeat:
  - console_read()/write()
  - noop()
  - wait for interrupt
- run_user_code()
The Kthreads Library

The cookbook uses Kthreads, a simple, non-preemptive thread library. It is included in the Makefile.

The Kthreads Library will be properly introduced in lecture in week 4. You can look ahead at the lecture notes. The cookbook will also tell you exactly where you need them.

use: #include "kt.h"

header: /cs/faculty/rich/cs170/include/kt.h

source: /cs/faculty/rich/cs170/src/libkt/
The Kthreads Library: Threading

Functions used in the cookbook:

- `void *kt_fork(void (*func)(void *), void *arg)`: equivalent to `pthread_create`
- `void kt_joinall()`: causes current thread to block until all other threads have either exited or blocked on a semaphore
- `kt_exit()`: equivalent to `pthread_exit`

Other functions useful for debugging:

- `void kt_join(void *kt_id)`
- `void *kt_self()`
- `void kt_sleep(int sec)`
The Kthreads Library: Semaphores

The cookbook also uses semaphores, a type of synchronization primitive. It is the only synchronization primitive provided by Kthreads. It will be introduced in lecture in week 3.

- void make_kt_sem(int initval)
- void kill_kt_sem(kt_sem ksem)
- void P_kt_sem(kt_sem ksem)
- void V_kt_sem(kt_sem ksem)
- int kt_getval(kt_sem s)
Test your code

- tests available at /cs/faculty/rich/cs170/test_execs - both source code and binary
- You can also use Gradescope - the output is much better now
- You can also build your own tests
Start early!

You will have 2 weeks for this lab - which is very short, considering the amount of material.

Lab 2 & 3 will build on this project, so make sure you understand the code, and leave as few bugs as possible. Later labs are worth exponentially more, but if you fail to understand this one, you will have a hard time on Lab 2 & 3.
Lab 0 Grading

We have just begun grading Lab 0. You should be able to see the final test outputs already, but they are not reflective of your Lab 0 grade.

Make sure to add your teammate on Gradescope or notify us about your team if you haven't done so already.
Questions?