-*- mode: org -*-
#+STARTUP: indent

* plan
- context switching solution
- sequence coordination
- xv6's implementation
- in-class exercise

* context switching solution
show solution

* big picture:
- Multiple threads executing in the kernel
- sharing memory, devices, and various data structures.
- locks to protect invariants
  one outstanding disk request
  one scheduler selecting a thread to run
- show context switching pattern in kernel

* sequence coordination:
- how to arrange for threads to wait for each other to do
  wait for disk interrupt to complete
  wait for pipe readers to make space in pipe
  wait for child to exit
  wait for block to use

* straw man solution: spin
waste CPU cycles if need to spin for long time

* beter solution: sleep when waiting
tricky: lost wakeup
- case study: iderw()
- why does sleep take the ide lock as argument?
  comment acquire/relase in sleep/wakeup
  sleep misses wakeup; deadlock
  ideintr() runs before we to sleep

* sleep arranges making going to sleep atomic:
first hold ptable lock
set SLEEP
then release the lock argument
but requires API change: sleep takes a lock argument

* Another example: pipe
what is the race if sleep didn't take p->lock as argument?

* Real kernels also deal with receiving a ctrl-C, e.g., in sleep.
This is messy because a process could sleep somewhere deep in the kernel
A signal forces it out of sleep
But when it comes out of the sleep it is not because the condition it is waiting on is true.
A common approach is use longjmp (unwind the stack), and retry the system call.
xv6 doesn't do this; but handles kill signal when sleeping on pipe by checking

6.828 Operating System Engineering

Fall 2012