xv6©-rev10
(Copyright Frans Kaashoek, Robert Morris, and Russ Cox.)
Scheduler

Carmi Merimovich

Tel-Aviv Academic College

November 20, 2017

Context

```
kinit1(end, P2V(4*1024*1024)); // phys page alloca
kvmalloc();        // kernel page table
   ⋮
seginit();        // set up segments
   ⋮
pinit();           // process table
   ⋮

   ⋮
kinit2(P2V(4*1024*1024), P2V(PHYSTOP)); // must co
userinit();        // first user process
mpmain();
```

Auxiliary context

- The primary processor begins its C code in **main()**.
- The auxiliary processors begins their C code in **mpenter()**.
- The state on entering either **main()** or **mpenter()** is the same.
- There is a separate stack of each processor.

```
static void mpenter(void) {
  switchkvm();
  seginit();
  lapicinit();
  mpmain();
}
```

mycpu()

```
struct cpu* mycpu(void) {
 int apicid, i;

 if (readeflags()&FL_IF)
  panic("mycpu called with interrupts enabled\n");

 apicid = lapicid();
 for (i = 0; i < ncpu; ++i) {
  if (cpus[i].apicid == apicid)
   return &cpus[i];
 }
 panic("unknown apicid\n");
}
```

mpmain()

```
static void mpmain(void) {
 cprintf("cpu%d: starting %d\n", cpuid(), cpuid());
 idtinit(); // load idt register
 xchg(&(mycpu()->started), 1); // tell startothers()
 scheduler(); // start running processes
}
```

myproc()

```
struct proc *myproc(void) {
 struct cpu *c;
 struct proc *p;
 pushcli();
 c = mycpu();
 p = c->proc;
 popcli();
 return p;
}
```

scheduler

```
2758 void scheduler(void) {
      struct proc *p;
      struct cpu *c = mycpu();
      c->proc = 0;
      for(;;) { sti();
       acquire(&ptable.lock);
       for(p = ptable.proc; p < &ptable.proc[NPROC]; p++) {
        if (p->state != RUNNABLE) continue;

        c->proc = p;
        switchuvm(p);
        p->state = RUNNING;
        swtch(&c->scheduler, p->context);
        switchkvm();

        c->proc = 0;
       }
       release(&ptable.lock);
      }
```

scheduler() operation

- For each proc struct p with state RUNNABLE the following is executed:
    - **c**$-\!>$**proc = p**;
    - **switchuvm()**.
    - **swtch()**.
    - **switchkvm()**.
    - **c**$-\!>$**proc=NULL**.