XV6 SCHEDULING

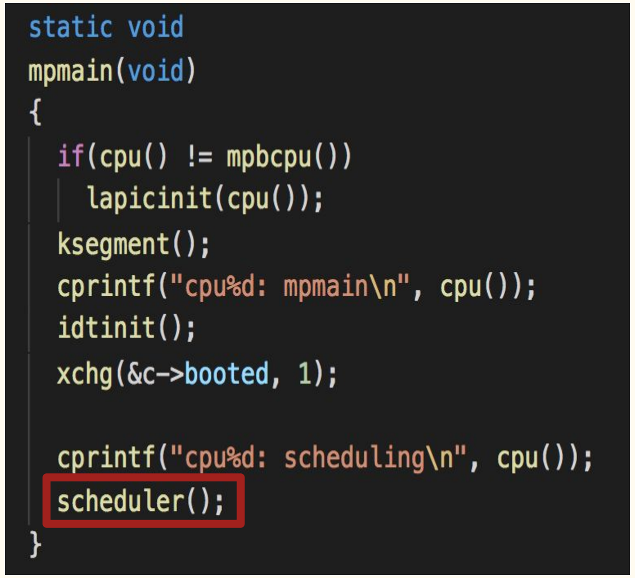
Operating Systems

University of Tehran-Faculty of Computer Engineering

Spring 1400

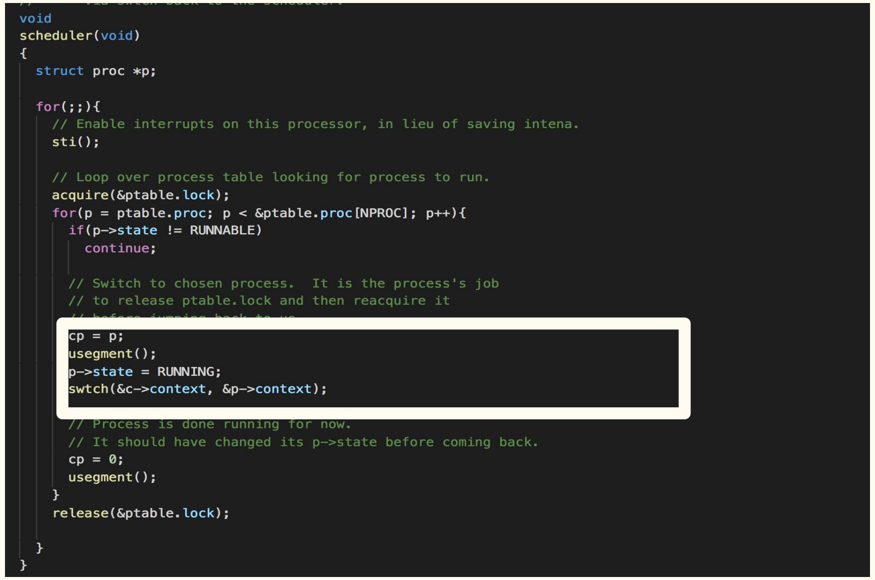
LET'S SEE HOW XV6 DOES SCHEDULING

- Main.c □ scheduler() / Proc.c □ scheduler() □ Round Robin Implementation



```
static void
mpmain(void)
{
  if(cpu() != mpbcpu())
    lapicinit(cpu());
  ksegment();
  cprintf("cpu%d: mpmain\n", cpu());
  idtinit();
  xchg(&c->booted, 1);

  cprintf("cpu%d: scheduling\n", cpu());
  scheduler();
}
```



```
//      via swtch back to the scheduler.
void
scheduler(void)
{
  struct proc *p;

  for(;;){
    // Enable interrupts on this processor, in lieu of saving intena.
    sti();

    // Loop over process table looking for process to run.
    acquire(&ptable.lock);
    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
      if(p->state != RUNNABLE)
        continue;

      // Switch to chosen process.  It is the process's job
      // to release ptable.lock and then reacquire it
      // before jumping back to us.
      cp = p;
      usegment();
      p->state = RUNNING;
      swtch(&c->context, &p->context);

      // Process is done running for now.
      // It should have changed its p->state before coming back.
      cp = 0;
      usegment();
    }
    release(&ptable.lock);

  }
}
```

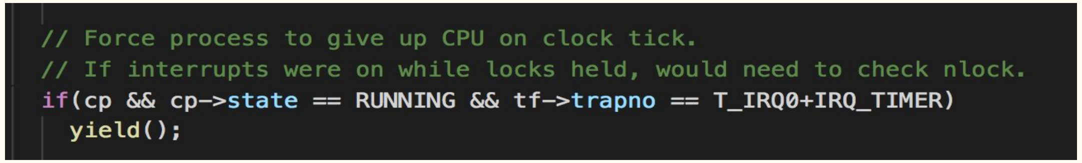
LET’S SEE HOW XV6 DOES SCHEDULING: SCHEDULER() FUNCTION

- swtch(&c->context, &p->context);
  - Makes process ”p” run in next time quantum by substituting context pointers
  - &c->context: pointer to current CPU scheduler context
  - &p->context: pointer to next running process context

- What happens if a process is paused by timer interrupt or is blocked by I/O operation?
- How can we pick another process to run?

LET’S SEE HOW XV6 DOES SCHEDULING: CHOOSING ANOTHER PROCESS

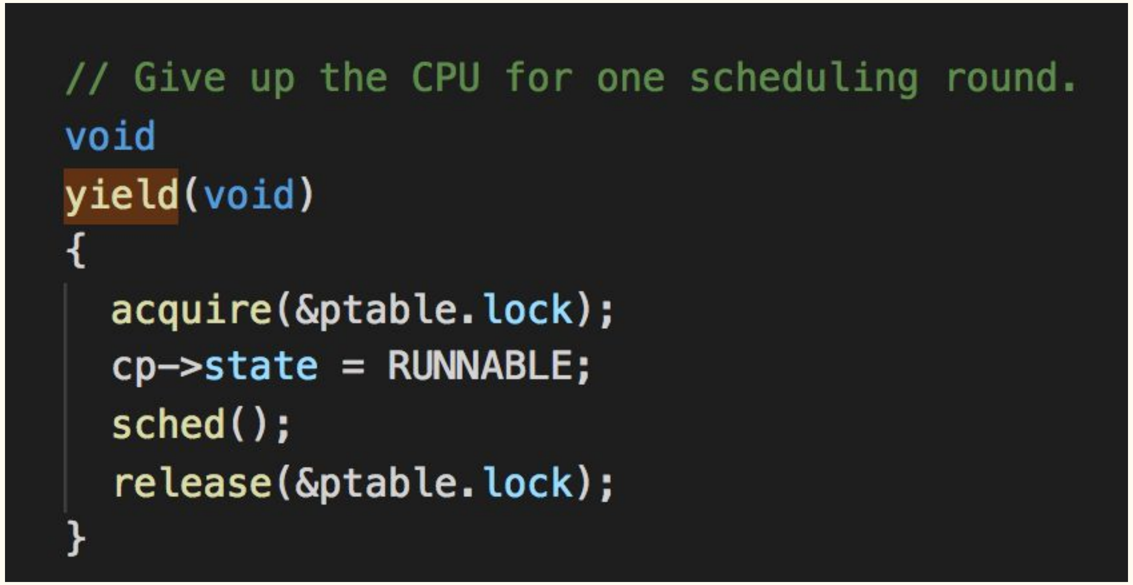
- Assume we have timer interrupt
  - Timer generates interrupt □ Cause syscall to call a trap(implemented in trap.c)
  - Yield function is executed(implemented in proc.c)



```
// Force process to give up CPU on clock tick.
// If interrupts were on while locks held, would need to check nlock.
if(cp && cp->state == RUNNING && tf->trapno == T_IRQ0+IRQ_TIMER)
  yield();
```

Trap.c

This mentioned procedure is implementation of in XV6!



```
// Give up the CPU for one scheduling round.
void
yield(void)
{
  acquire(&ptable.lock);
  cp->state = RUNNABLE;
  sched();
  release(&ptable.lock);
}
```

Proc.c

SCHEDULING MODIFICATIONS

- Round Robin Multi Layer Scheduling
  - 1st level: Round-Robin Scheduling (First priority)
  - 2nd level: Priority Scheduling (Second priority)
  - 3rd level: BJF Scheduling (Third priority)
  - 4rd level: FCFS Scheduling (Forth priority)

ROUND-ROBIN SCHEDULING

- In this algorithm we define a small unit of time, called time quantum or time slice
- CPU scheduler allocate the CPU to each process for a time interval of up to 1 time quantum.
- If process has CPU burst of less than 1 time quantum, it releases the CPU voluntarily and scheduler proceed to next process in the ready queue.
- If CPU burst of currently running process is longer than 1 time quantum, the timer will go off and will cause an interrupt to operating system. process will be put at the tail of ready queue. and CPU scheduler will then select next process in ready queue.

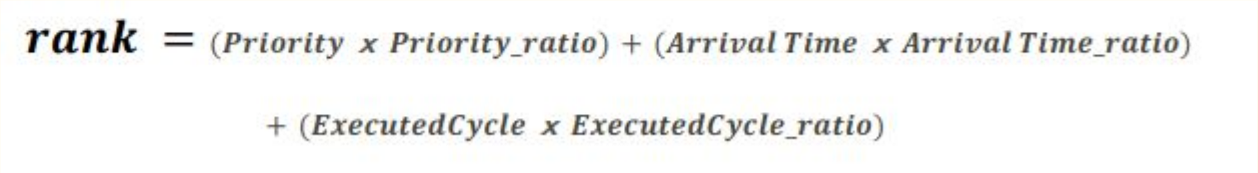
PRIORITY SCHEDULING

- Generating a random priority for each process
- Process with smaller priority number is candidate to be chosen

BJF SCHEDULING

- You need to calculate a process executed cycles

  When a process executes, its executed cycle attributes increases 0.1 in magnitude, and the default value is set to 0
- The lower a process response ratio is, the higher the process chance is to be executed



$$\boldsymbol{rank} = (Priority\ x\ Priority\_ratio) + (Arrival\,Time\ x\ Arrival\,Time\_ratio) + (ExecutedCycle\ x\ ExecutedCycle\_ratio)$$

FCFS SCHEDULING

- Process that arrives earlier, would be chosen first.

AGING

- Every process in system has a level
- If a process do not belongs to 1st level and waits more than 8000 cycle

Change its level, to upper level.

COMPLEMENTARY SYSTEM CALLS

- 1. Change level of scheduling
- 2. Assigning priority to processes
- 3. Change ratios of the BJF in process level
- 4. Listing all processes (helpful for your debugging)

Thank You for your attendance 😊