Operating System

Project #1-2

16.10.04

Project Plan

- 5 projects
  - Install Xv6
  - System call + scheduling
  - Thread-support
  - Virtual memory
  - Concurrency

- Single-handed project

Process State

```
enum procstate { UNUSED, EMBRYO, SLEEPING, RUNNABLE, RUNNING, ZOMBIE };

// Per-process state
struct proc {
  uint sz;                     // Size of process memory (bytes)
  pde_t* pgdir;                // Page table
  char *kstack;                // Bottom of kernel stack for this process
  enum procstate state;        // Process state
  int pid;                     // Process ID
```

- UNUSED : Unused
- EMBRYO : Newly allocated (not ready for running yet)
- SLLEPING : Waiting for I/O, child process, time
- RUNNABLE : Ready to run
- RUNNING : Running on CPU
- ZOMBIE : Exited

Xv6 Process Scheduler

- proc.c

```
void
scheduler(void)
{
  struct proc *p;

  for(;;){
    // Enable interrupts on this processor.
    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.
      proc = p;
      switchuvm(p);
      p->state = RUNNING;
      swtch(&cpu->scheduler, p->context);
      switchkvm();

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

  }
}
```

Xv6 Process Scheduler (cont.)

- proc.c

```
void
sched(void)
{
  int intena;

  if(!holding(&ptable.lock))
    panic("sched ptable.lock");
  if(cpu->ncli != 1)
    panic("sched locks");
  if(proc->state == RUNNING)
    panic("sched running");
  if(readeflags()&FL_IF)
    panic("sched interruptible");
  intena = cpu->intena;
  swtch(&proc->context, cpu->scheduler);
  cpu->intena = intena;
}
```

Xv6 Process Scheduler (cont.)

- swtch.S

```
.globl swtch
swtch:
  movl 4(%esp), %eax
  movl 8(%esp), %edx

  # Save old callee-save registers
  pushl %ebp
  pushl %ebx
  pushl %esi
  pushl %edi

  # Switch stacks
  movl %esp, (%eax)
  movl %edx, %esp

  # Load new callee-save registers
  popl %edi
  popl %esi
  popl %ebx
  popl %ebp
  ret
```

Xv6 Process Scheduler (cont.)

- When?
  - Exiting process (exit() in proc.c)

```
// Jump into the scheduler, never to return.
proc->state = ZOMBIE;
sched();
panic("zombie exit");
```

  - Sleeping process (sleep() in proc.c)

```
// Go to sleep.
proc->chan = chan;
proc->state = SLEEPING;
sched();
```

Xv6 Process Scheduler (cont.)

- When?
  - Yielding CPU
    - trap() in trap.c

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

    - yield() in proc.c

```
void
yield(void)
{
  acquire(&ptable.lock);  //DOC: yieldlock
  proc->state = RUNNABLE;
  sched();
  release(&ptable.lock);
}
```

Project #1-2 – Priority Scheduler

- Implement priority-based scheduler on xv6
  - The lower nice value, higher priority
  - The highest priority process is selected for next running
    - Tiebreak : Round-robin fashion

- Entering scheduler when
  - Exiting process
  - Sleeping process
  - Yielding CPU
    - Timer : Must be removed
    - Called by user

Project #1-2 – Priority Scheduler (cont.)

- Testcases
  - Exiting process – **20p**
  - Waiting process – **20p**
  - Sleeping process – **20p**
  - Yielding CPU – **20p**
  - Changing priority – **20p**

Project #1-2 – Priority Scheduler (cont.)

- Submit a tar.gz file

- Send email to T.A
  - [SSE3044]Project#2-YOURID-YOURNAME.tar.gz
    - ex) [SSE3044]Project#1_2-2016710580-이규선.tar.gz
  - Email address : lgs0409@naver.com
  - Wrong title is not allowed

- Due date
  - 2016-10-09(Sun) PM 23:59
  - Get no point for late submission