EEE3052: Introduction to Operating Systems

Fall 2017

Project #1

Project Plan

- 4 projects
  0) Install Xv6
  1) Process management
     - System call (9/11 ~ 9/17)
     - Scheduling (9/18 ~ TBD)
  2) Virtual memory
  3) Synchronization
  4) File system

Process State

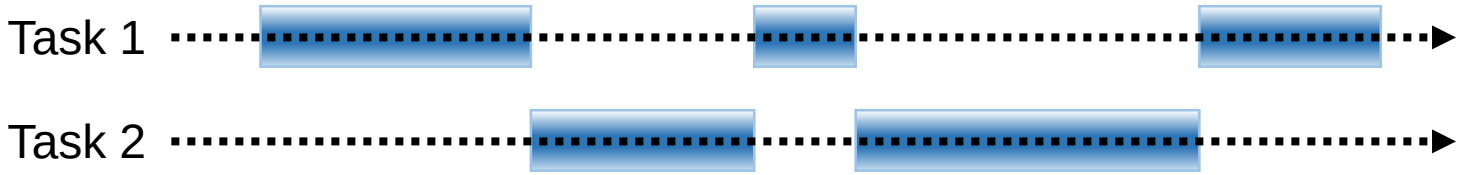
- Represent process situation

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

- 5 state in Xv6
  - EMBRYO: Newly allocated (not ready for running yet)
  - SLEEPING: Waiting for I/O, child process, time
  - RUNNABLE: Ready to run
  - RUNNING: Running on CPU
  - ZOMBIE: Exited but not withdrawn by parent

Process Scheduler

- Time sharing to run more threads than processors



  - Interleaving processes
  - When a process waits for an event (I/O response)
  - Preemption (forced stopping of a task)

- Scheduler determines
  - Which task to run
  - How long it will run

Scheduler in Xv6

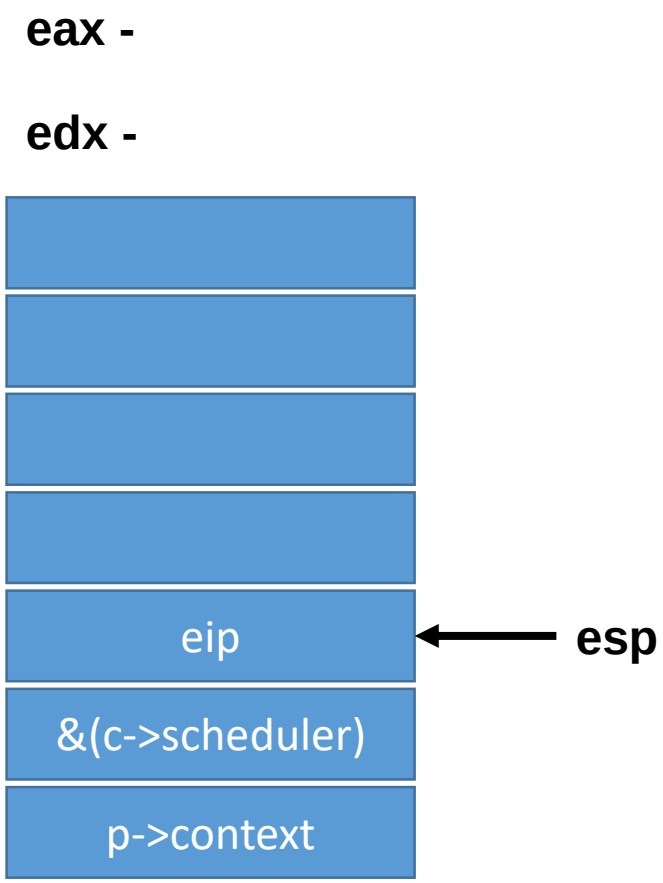
- Round-robin fashion
  - Why it isn't FIFO?

```
void
scheduler(void)
{
  struct proc *p;
  struct cpu *c = mycpu();
  c->proc = 0;

  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.
      c->proc = p;
      switchuvm(p);
      p->state = RUNNING;

      swtch(&(c->scheduler), p->context);
      switchkvm();

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

  }
}
```

Context Switch in Xv6

- Swap process context

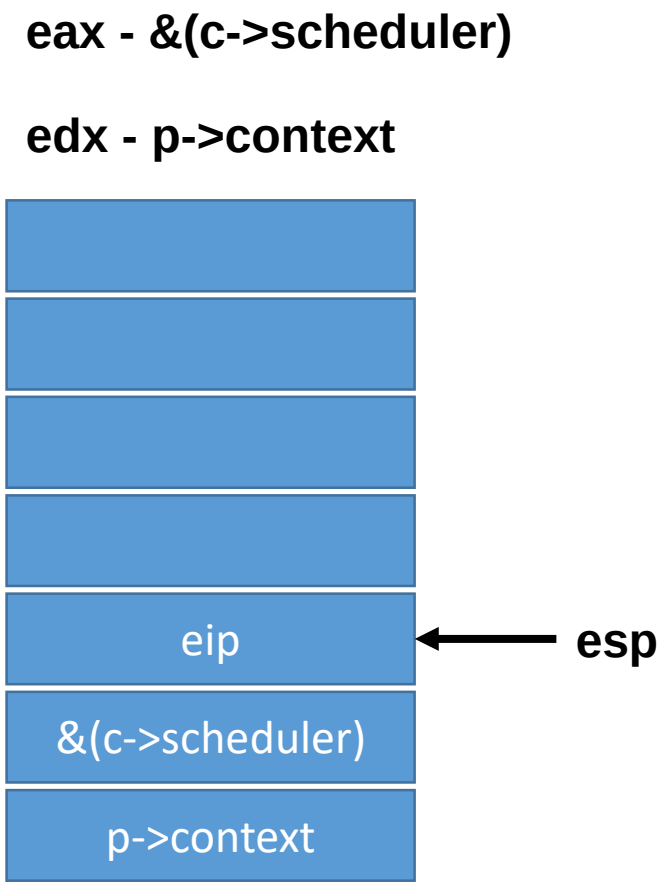
```
.globl swtch
swtch:
  movl 4(%esp), %eax
  movl 8(%esp), %edx

  # 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
```



Context Switch in Xv6 (Cont.)

- Swap process context

```
.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
```



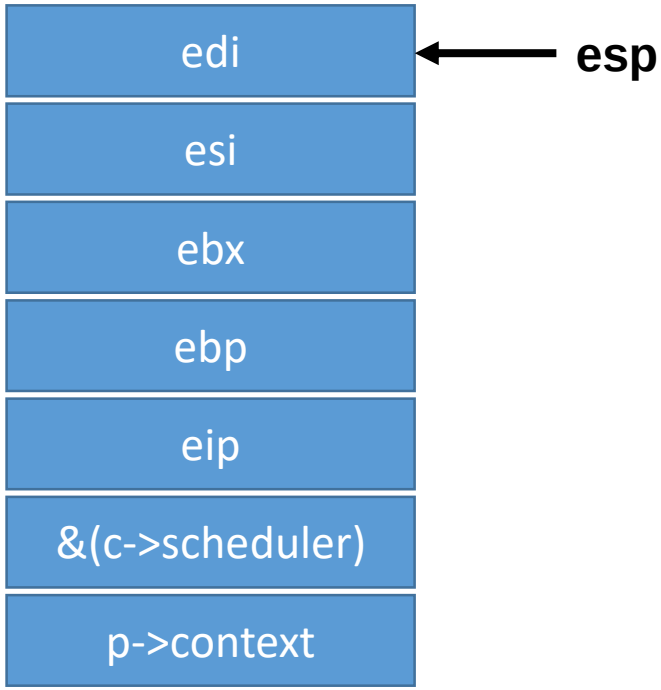
Context Switch in Xv6 (Cont.)

- Swap process context

```
.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
```

eax - &(c->scheduler)

edx - p->context



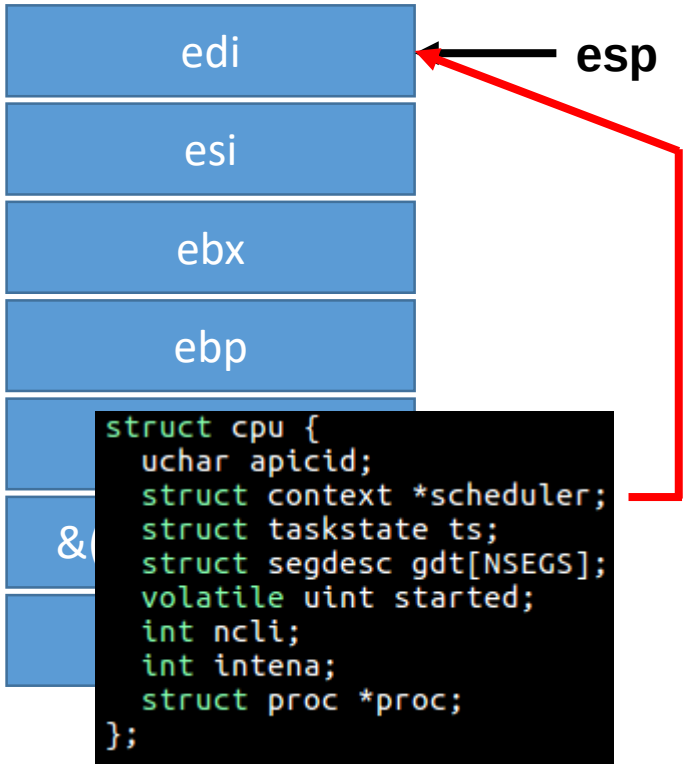
Context Switch in Xv6 (Cont.)

- Swap process context

```
.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
```

eax - &(c->scheduler)

edx - p->context



Context Switch in Xv6 (Cont.)

- Swap process context

```
.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
```

eax - &(c->scheduler)

edx - p->context

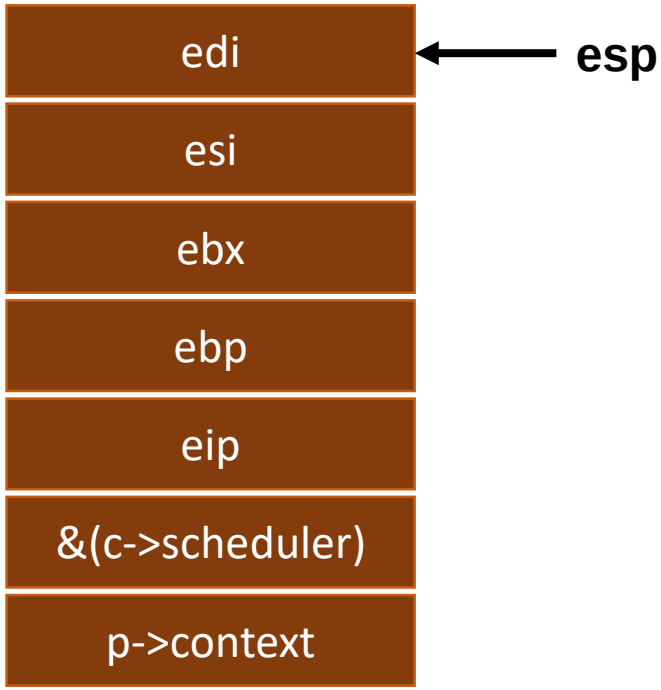
Context Switch in Xv6 (Cont.)

- Swap process context

```
.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
```

eax - &(c->scheduler)

edx - p->context



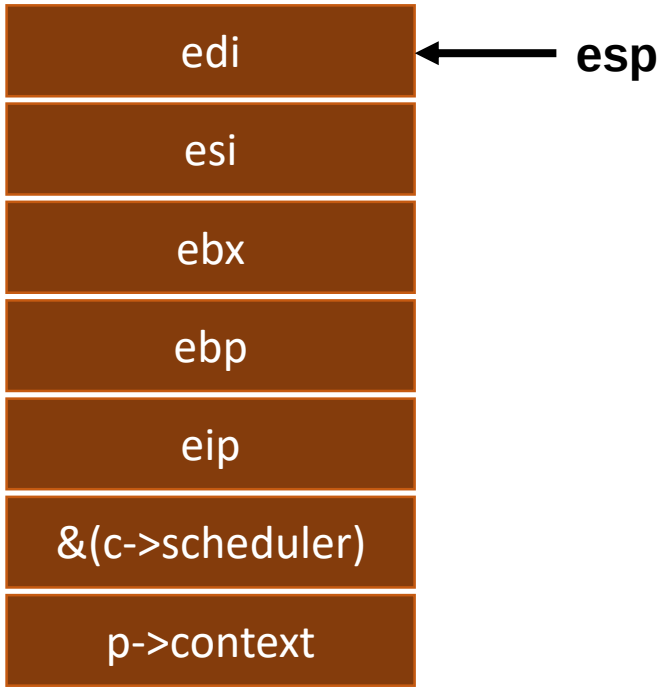
Context Switch in Xv6 (Cont.)

- Swap process context

```
.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
```

eax - &(c->scheduler)

edx - p->context



Enter Scheduler

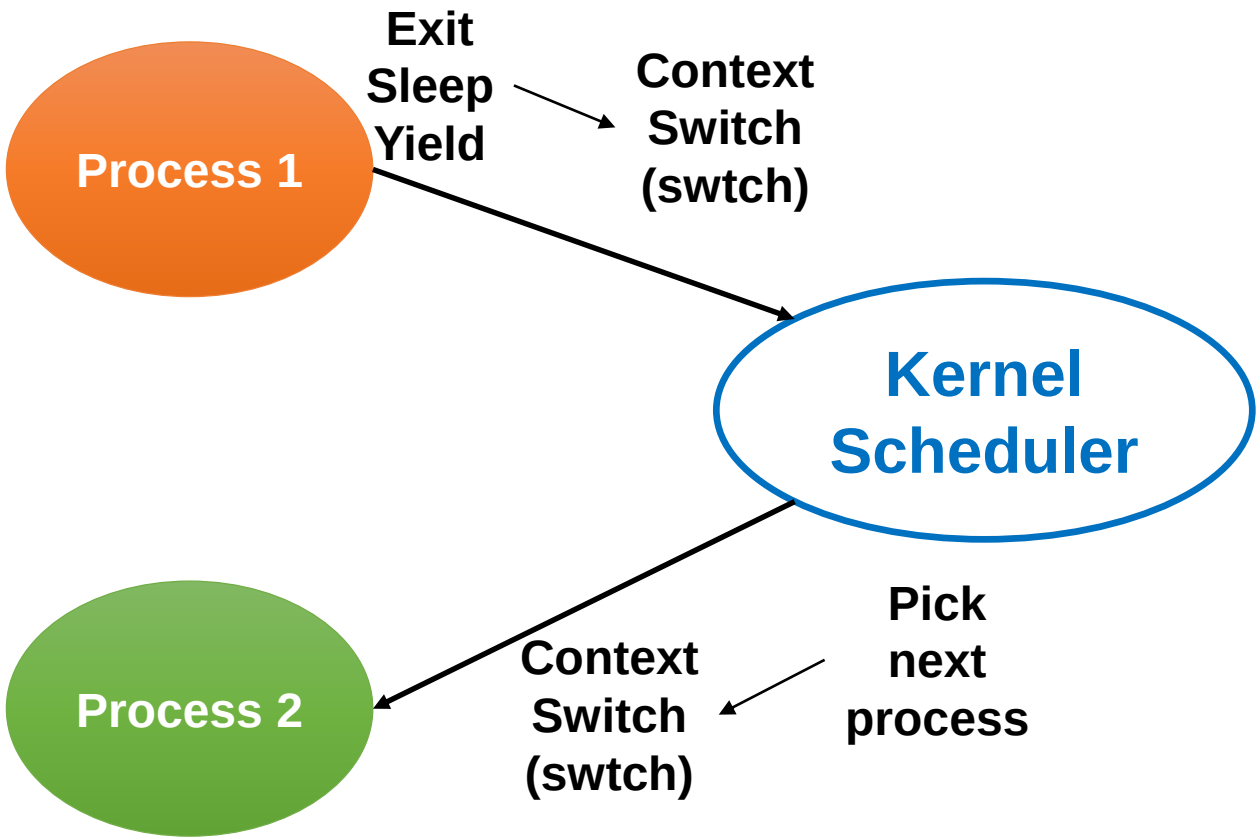
- Process calls scheduler by running sched()

```
void
sched(void)
{
  int intena;
  struct proc *p = myproc();

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

Schedule Entering Point

- Exit function
  - When process exit
- Sleep function
  - When process sleep
- Yield function
  - When process yield CPU to other process

Scheduler Summary



Project #1-2 – Priority Scheduler

- Make system call
  - int getyieldcnt(pid p)
    1. Return process yield count
    2. Process should add yield count at yield
  - void yield(void)

- Implement priority-based scheduler
  - The lower nice value, the higher priority
  - The highest priority process is selected for next run
    - Tiebreak: FIFO fashion

Project #1-2 – Priority Scheduler

- Entering scheduler when
  - Exiting process
  - Sleeping process
  - Yielding CPU
  - Changing priority

Project #1-2 Score

- Total 100 point

- Checkpoint
  - [2-0] (20 point) Call yield system call iteratively
  - [2-1] (10 point) Exit process
  - [2-2] (10 point) Wait process
  - [2-3] (10 point) Sleep process
  - [2-4] (20 point) Yield CPU to other process
  - [2-5] (30 point) Change process priority

Project #1-2 Answers

[2_0]

```
$ test_2_0
PID 4 yield 100 count
```

[2_1]

```
$ test_2_1
##### State 1 #####
##### State 2 #####
PID 16 yield 0 count
##### State 3 #####
PID 18 yield 0 count
##### State 4 #####
PID 17 yield 0 count
zombie! (main function in init.c)
zombie! (main function in init.c)
```

[2_2]

```
$ test_2_2
##### State 1 #####
##### State 2 #####
PID 12 yield 0 count
##### State 3 #####
PID 11 yield 0 count
```

[2_3]

```
$ test_2_3
##### State 1 #####
##### State 2 #####
PID 24 yield 0 count
##### State 3 #####
PID 23 yield 0 count
zombie! (main function in init.c)
```

```
$ test_2_4
##### State 1 #####
##### State 2 #####
##### State 3 #####
##### State 4 #####
##### State 5 #####
##### State 6 #####
##### State 7 #####
##### State 8 #####
##### State 9 #####
##### State 10 #####
##### State 11 #####
##### State 12 #####
##### State 13 #####
##### State 14 #####
##### State 15 #####
##### State 16 #####
##### State 17 #####
##### State 18 #####
##### State 19 #####
##### State 20 #####
##### State 21 #####
##### State 22 #####
##### State 23 #####
##### State 24 #####
##### State 25 #####
##### State 26 #####
##### State 27 #####
##### State 28 #####
##### State 29 #####
##### State 30 #####
PID 31 yield 10 count
PID 32 yield 10 count
PID 33 yield 10 count
zombie! (main function in init.c)
zombie! (main function in init.c)
```

[2_4]

```
$ test_2_5
##### State 1 #####
##### State 2 #####
##### State 3 #####
PID 27 yield 0 count
##### State 4 #####
PID 28 yield 0 count
zombie! (main function in init.c)
```

[2_5]

Makefile

- User program →

```
UPROGS=\
	_cat\
	_echo\
	_forktest\
	_grep\
	_init\
	_kill\
	_ln\
	_ls\
	_mkdir\
	_rm\
	_sh\
	_stressfs\
	_usertests\
	_wc\
	_zombie\
	_minitop\
	_test_2_0\
	_test_2_1\
	_test_2_2\
	_test_2_3\
	_test_2_4\
	_test_2_5\
```

- CPU count →

```
ifndef CPUS
CPUS := 1
endif
QEMUOPTS = -drive file=fs.img,index=1,m
```

Submit

- Send email to jaehyun.song@csl.skku.edu
  - Title: [EEE3052]Project-1_2-studentID-name
  - File name: studentID-1_2.tar.gz

- Due date
  - TBD
  - Penalty 10% of each project score per one day

- TA contact
  - jaehyun.song@csl.skku.edu **(ONLY THIS EMAIL)**