SSE3044 Introduction to Operating Systems
prof. Jinkyu Jeong

Project 1-2. Priority scheduler

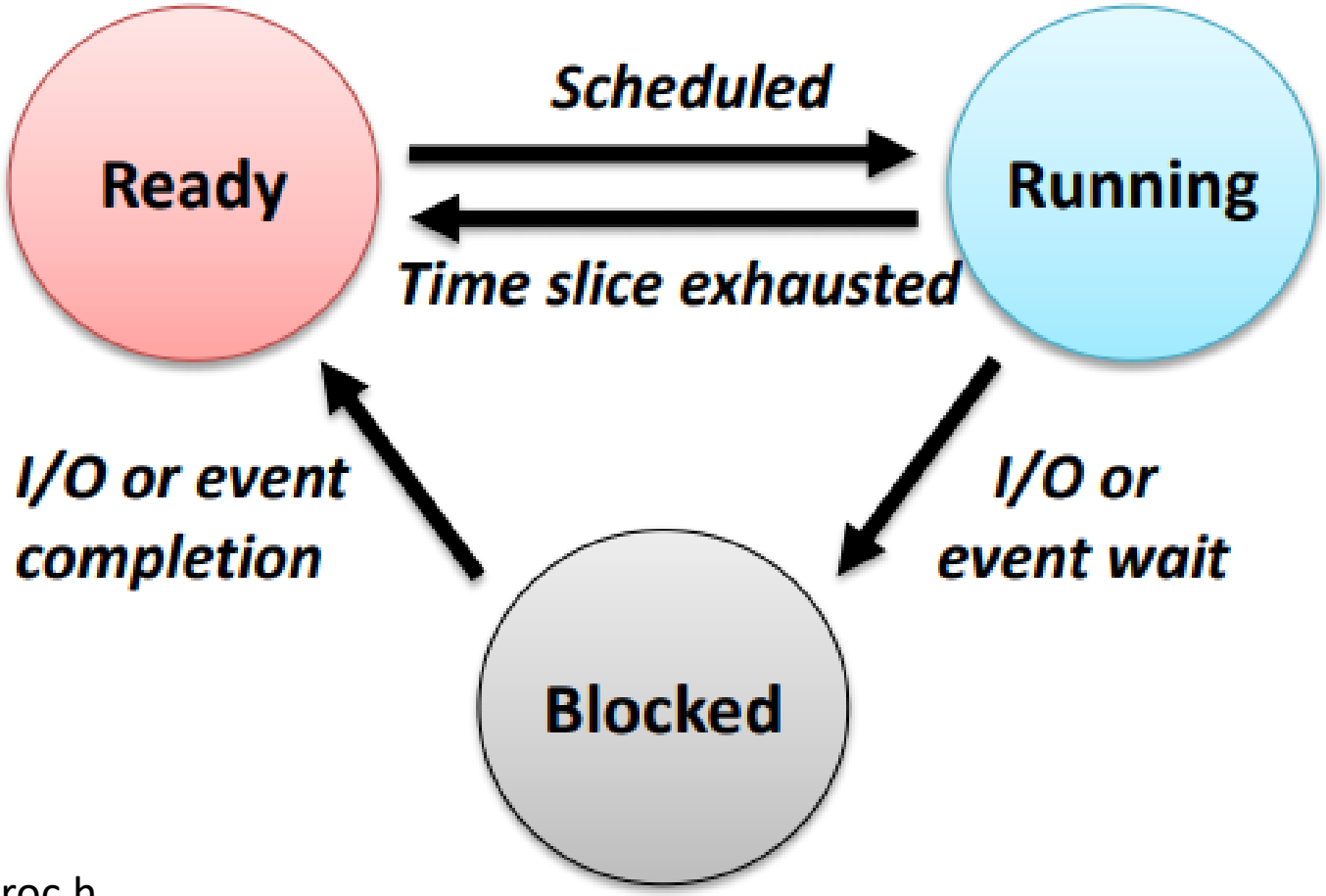
2018.3.27 (Wed.)

TAs
이규선(gyusun.lee@csl.skku.edu) /
안민우(minwoo.ahn@csl.skku.edu)

Project plan

- Total 4 projects
  0) Starting xv6 operating system
  1) Process management
     1) System call
     2) Scheduling (3/28 ~ 4/3)
  2) Virtual memory
  3) Scheduling
  4) Filesystem

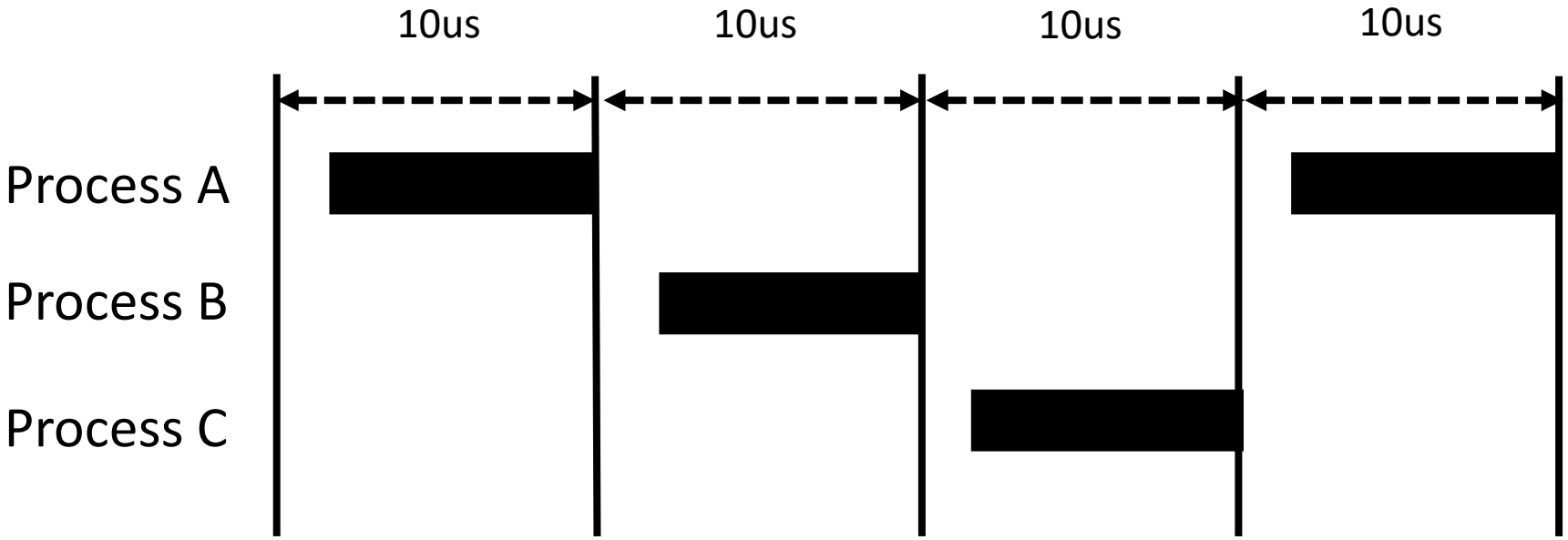
Process states



proc.h

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

Process scheduler (ex. RR)



- Scheduler's decisions
  - When to switch process
  - Which process to switch

xv6 scheduler

Yield at
“timer interrupt”

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

```
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);

  }
}
```

sched() function

- Process enters scheduler by calling 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;
}
```

Entering scheduler

- Inside proc.c,
  - Exit() function
    - When process tries to exit (-> zombie)
  - Sleep() function
    - When process tries to sleep (-> waiting)
  - Yield() function
    - When process yield CPU to other process (-> runnable)

Project #1-2 – Priority scheduler

- Two system calls
  - int getyieldcnt(pid)
    - Return how many yields occurred to process(pid)
    - Should be accumulated until shutdown
  - void yield(void)

- Implement priority-based scheduler
  - Lower nice value means higher priority
  - The highest priority process is decided to preempt CPU
    - For same priority, apply FIFO concept

Project #1-2 – Priority scheduler

- To sum up, process enters scheduler when
  - Exiting process
  - Sleeping process
  - Yielding CPU
  - Change priority (setnice)

- Change number of CPU to emulate to 1

```
ifndef CPUS
CPUS := 2
endif
```

1

Submission

- Compress your code as YourStudentID-1-2.tar.gz
- Send your file to minwoo.ahn@csl.skku.edu
  - Please command $make clean, before submission
- PLEASE DO NOT COPY
  - YOU WILL GET F GRADE IF YOU COPIED
- Due date: 4/3(Tue.), 23:59:59 PM
  - -25% per day for delayed submission

Tips

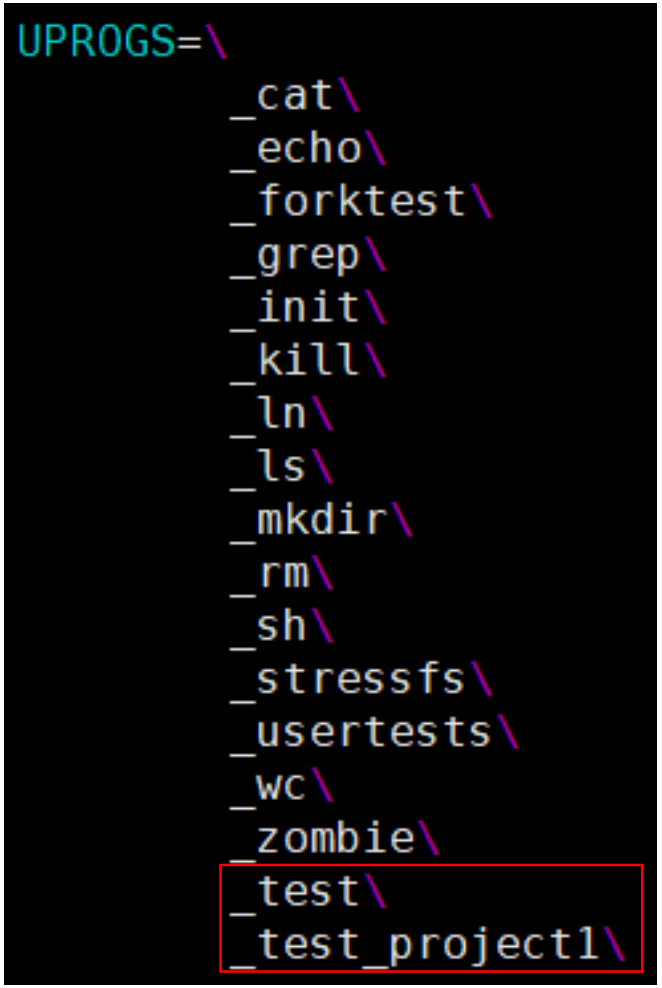
- Reading xv6 commentary will help you a lot
  - http://csl.skku.edu/SSE3044S18/Resources?action=upload&upname=book-rev10.pdf

Questions

- If you have questions, please email to TA
- You can also visit Semiconductor Building #400509
  - Please email TA before visiting

(How to) Add user program

- Write your .c code and add it’s name to “Makefile”



Appendix. ctags & grep

- Install ctags
  - $sudo apt install ctags
- Vim setting for ctags
  - $ctags –R (where you will use)
  - $vi ~/.vimrc
  - Add “set tags=[Location of tag file]/tags”
  - $source ~/.vimrc
- Ctags usage
  - ctrl +] : follow tag
  - ctrl + t : back to last tag
- Grep Usage
  - grep –nR “[string to search]”

Appendix. vi instructions

** $sudo apt install vim

① vi [filename] -> Open file

② i -> keyboard typing mode

③ esc + :w -> save file

④ esc + :q -> exit file (esc + :wq -> save and exit)

⑤ /[string] -> search [string]

⑥ u -> back to last command

⑦ :vs -> open additional file on same session

⑧ dd -> erase one line

⑨ :set mouse=a -> activate mouse

  ① Drag and y -> copy multi-line

  ② p -> paste

* Reference: http://www.antsys.co.kr/data/vi_editor.htm