

SWE3004: Operating Systems  
prof. Euseong Seo

# Project 4. Priority Scheduler

---

2019.5.1 (Wed.)

TAs

김종석(ks77sj@gmail.com) /

최동규(gmj03003@gmail.com)

# Project Plan

---

- Total 7 projects

- 0) Starting xv6 operating system (5%)
- 1) System call (10%)
- 2) Thread (15%)
- 3) Synchronization (15%)
- 4) Scheduling 1(10%)
- 5) Scheduling 2(15%)
- 6) Page fault handler (15%)
- 7) Copy on Write (15%)

# Xv6 Process

---

- Process states (procstate in proc.h)
  - UNUSED: Not used
  - EMBRYO: Newly allocated (not ready for running yet)
  - SLEEPING: Waiting for I/O, child process, or time
  - RUNNABLE: Ready to run
  - RUNNING: Running on CPU
  - ZOMBIE: Exited

# Xv6 Process Scheduler

---

- scheduler() in proc.c
  - Round-robin fashion

```
279 void
280 scheduler(void)
281 {
282     struct proc *p;
283
284     for(;;){
285         // Enable interrupts on this processor.
286         sti();
287
288         // Loop over process table looking for process to run.
289         acquire(&ptable.lock);
290         for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
291             if(p->state != RUNNABLE)
292                 continue;
293
294             // Switch to chosen process. It is the process's job
295             // to release ptable.lock and then reacquire it
296             // before jumping back to us.
297             proc = p;
298             switchvm(p);
299             p->state = RUNNING;
300             swtch(&cpu->scheduler, p->context);
301             switchkvm();
302
303             // Process is done running for now.
304             // It should have changed its p->state before coming back.
305             proc = 0;
306         }
307         release(&ptable.lock);
308
309     }
310 }
```

# Xv6 Entering Scheduler

---

- sched() in proc.c

```
319 void
320 sched(void)
321 {
322     int intena;
323
324     if(!holding(&ptable.lock))
325         panic("sched ptable.lock");
326     if(cpu->ncli != 1)
327         panic("sched locks");
328     if(proc->state == RUNNING)
329         panic("sched running");
330     if(readeflags() & FL_IF)
331         panic("sched interruptible");
332     intena = cpu->intena;
333     swtch(&proc->context, cpu->scheduler);
334     cpu->intena = intena;
335 }
```

# Xv6 Entering Scheduler (Cont'd)

---

- When?

1. Exiting process (exit() in proc.c)

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

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

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

# Xv6 Entering Scheduler (Cont'd)

---

- When?

3. Yielding CPU due to timer interrupt

- trap() in trap.c

```
106     if(proc && proc->state == RUNNING && tf->trapno == T_IRQ0+IRQ_TIMER)
107         yield();
```

- Yield() in proc.c

```
341     acquire(&ptable.lock); //DOC: yieldlock
342     proc->state = RUNNABLE;
343     sched();
344     release(&ptable.lock);
```

# Project 4 - Priority Scheduler

---

- Implement **priority-based scheduler** on xv6
  - The lower nice value, the higher priority
  - The highest priority process is selected for next running
    - **Tiebreak: round-robin fashion**
- Entering scheduler when
  1. Exiting process
  2. Sleeping process
  3. Yielding CPU
  4. **Changing priority**



# Project 4 - Priority Scheduler

---

- You also have to make **2 system calls** that you did in Project 1. (Parameters, return values, etc. are same with previous project)
  - getnice
  - setnice
    - This time, you will also have to consider entering the scheduler
- When you call `fork()`, the child process has the same priority as the parent process.
- You don't need to make or consider thread.

# Template Code

---

- `wget http://csl.skku.edu/uploads/SWE3004S19/xv6-project-4.tar.gz`
- Modifications
  - halt system call
  - Halt xv6 program
  - make tarball
    - Compress your source codes into one .tar.gz file for submission
    - You should enter your ID & project no. on Makefile
  - `CPUS=1`
  - Ignore to yield CPU on clock tick
  - `yield()` system call

# Submission

---

- You need to submit a document.
- Just write how you implemented your code.
- You can use English or Korean.

# Submission

---

- Send your code file (xv6-project-4-studentID.tar.gz) and document file to [ks77sj@gmail.com](mailto:ks77sj@gmail.com)
- Please send a mail with tittle including [SWE3004-P4]
  - Ex) [SWE3004-P4] 2014111111-project4
- **PLEASE DO NOT COPY**
  - **YOU WILL GET F GRADE IF YOU COPIED**
- Due date: 5/8(Wed.), 23:59:59 PM
  - Delays are allowed only one week from the deadline. And there will be up to -40% penalty.

# Questions

---

- If you have questions, please email to TA
  - From this project, you can't ask questions on deadline day
- You can also visit #85533. Please email TA before visiting