대면 강의 계획 확인

- eCampus->운영체제->강의계획서

[주별 강의계획서]

1주차 03-02 ~ 03-08	주별학습목표	CPU Virtualization
	강의내용	Introduction (Ch. 2), Process API (Ch. 4 and 5)
	수업유형	이론
	학습활동	
	강의실	월13-15(녹화강의), 수13-15(녹화강의)
2주차 03-09 ~ 03-15	주별학습목표	CPU Virtualization
	강의내용	Limited Direct Execution (Ch. 6), CPU Scheduling (Ch. 7)
	수업유형	이론
	학습활동	
	강의실	월13-15(녹화강의), 수13-15(녹화강의)
3주차 03-16 ~ 03-22	주별학습목표	CPU Virtualization
	강의내용	Multi-Level Feedback Queue (Ch. 8), Multiprocessor Scheduling (Ch. 10)
	수업유형	이론
	학습활동	
	강의실	월13-15(녹화강의), 수13-15(녹화강의)
4주차 03-23 ~ 03-29	주별학습목표	Memory Virtualization
	강의내용	Assignment #1, Address Spaces (Ch. 13, 14, and 15)
	수업유형	이론
	학습활동	과제 #1
	강의실	월13-15(녹화강의), 수13-15(공B475)

3월 7일 3월 2일

3월 28일 3월 23일

System Software Laboratory

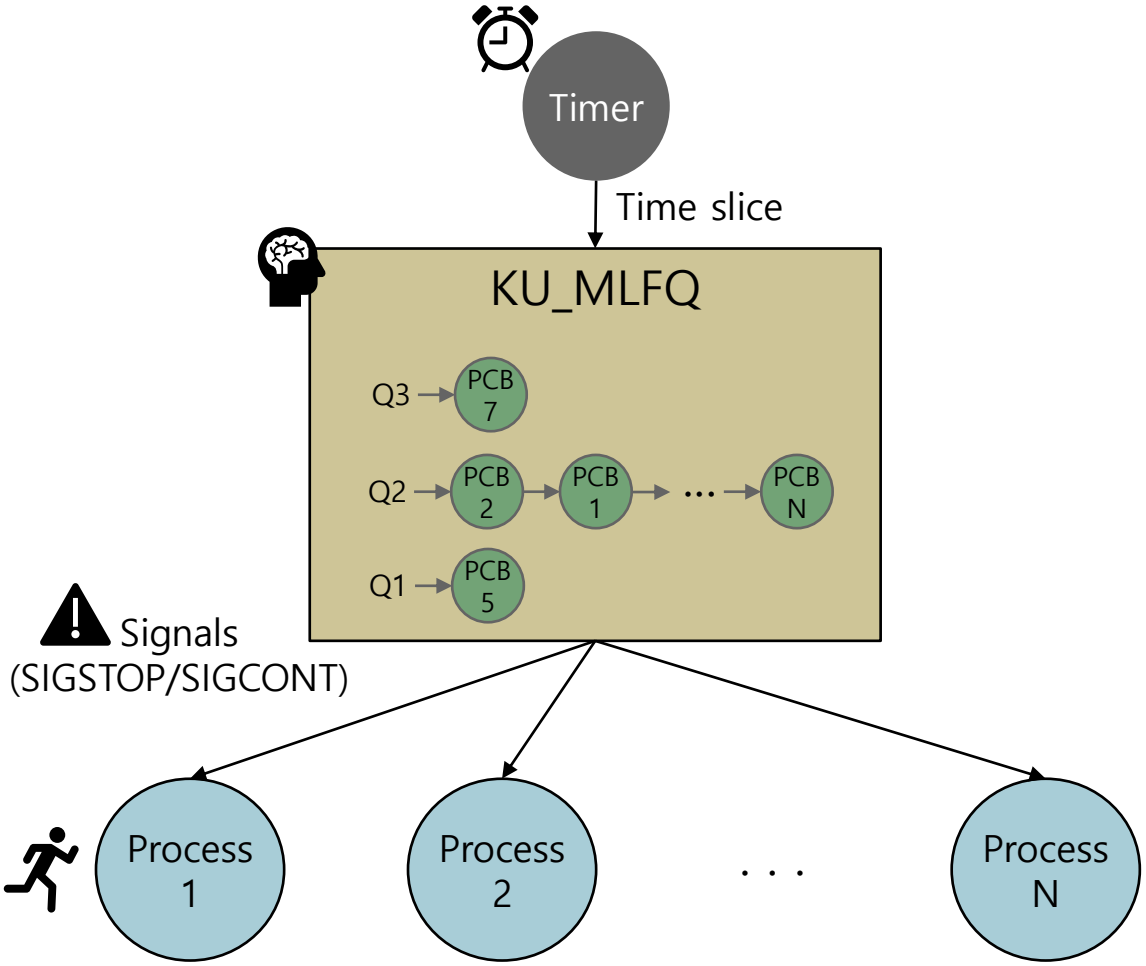
KU KONKUK UNIVERSITY

Operating Systems

Assignment #1: KU_MLFQ

Hyun-Wook Jin
System Software Laboratory
Dept. of Computer Science and Engineering
Konkuk University
jinh@konkuk.ac.kr

KU_MLFQ

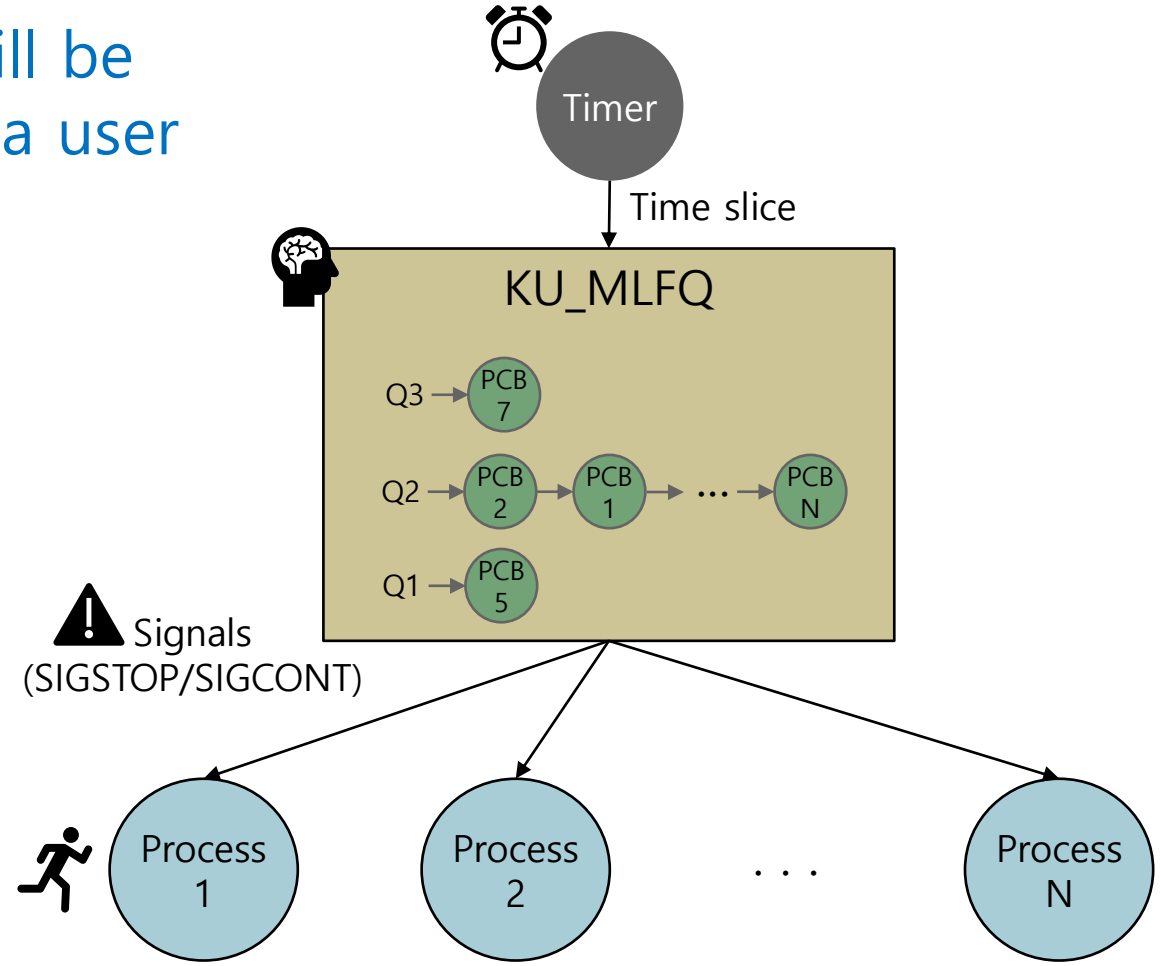

Timer
Time slice
KU_MLFQ
Q3
PCB 7
Q2
PCB 2
PCB 1
...
PCB N
Q1
PCB 5
Signals
(SIGSTOP/SIGCONT)
Process 1
Process 2
. . .
Process N

Scheduling Policy

- Rule 1 (Basic)
  - If Priority(A) > Priority(B), A runs (B doesn't)
- Rule 2 (Basic)
  - If Priority(A) = Priority(B), A & B run in RR
- Rule 3 (Priority adjustment algorithm)
  - When a job enters the system, it is placed at the highest priority
- Rule 4 (Priority adjustment algorithm & Gaming tolerance)
  - Once a job uses up its time allotment at a given level its priority is reduced
- Rule 5 (Priority boost)
  - After some time period *S*, move all the jobs in the system to the topmost queue

User-Level Implementation

- Our scheduler will be implemented as a user process! ;-)



Multi-Level Queues

- Priority
  - 1 (low)
  - 2
  - 3 (high)
- Ready queues
  - Array 
  - Linked list 
  - Red-black tree 

Time Slice

- Interval timer (itimer)
  - `int setitimer(int which, const struct itimerval *new_value, struct itimerval *old_value)`

```
struct itimerval {
    struct timeval it_interval; /* next value */
    struct timeval it_value;    /* current value */
};
```

```
struct timeval {
    time_t      tv_sec;     /* seconds */
    suseconds_t tv_usec;    /* microseconds */
};
```

  - Refer its man page for more details
- Using the POSIX timer is also fine

Scheduling Parameters

- Time slice
  - 1 second
- Gaming tolerance
  - Time allotment: 2 seconds
- *S*
  - 10 seconds

- Number of CPU cores
  - 1

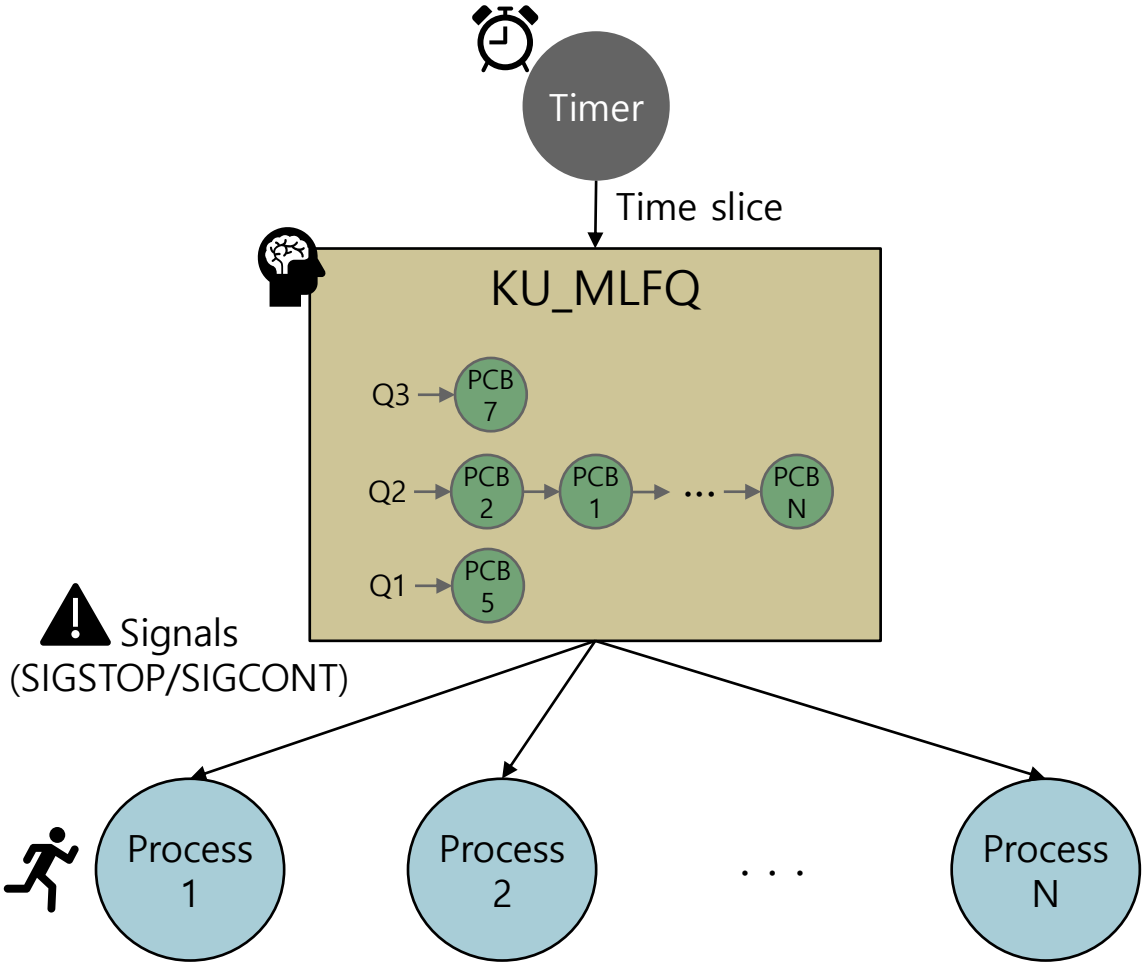
Scheduling Policy

- Rule 1 (Basic)
  - If Priority(A) > Priority(B), A runs (B doesn't)
- Rule 2 (Basic)
  - If Priority(A) = Priority(B), A & B run in RR (time slice: 1 sec.)
- Rule 3 (Priority adjustment algorithm)
  - When a job enters the system, it is placed at the highest priority (3)
- Rule 4 (Priority adjustment algorithm & Gaming tolerance)
  - Once a job uses up its time allotment (2 sec.) at a given level its priority is reduced
- Rule 5 (Priority boost)
  - After some time period *S* (10 sec.), move all the jobs in the system to the topmost queue

Signaling

- System calls for signaling
  - `handler_t *signal(int signum, handler_t *handler)`
  - `int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);`
  - `int kill(pid_t pid, int sig);`
- Signals
  - SIGALRM
  - SIGSTOP: Stop process
  - SIGCONT: Continue if stopped

KU_MLFQ


Timer
Time slice
KU_MLFQ
Q3
PCB 7
Q2
PCB 2
PCB 1
...
PCB N
Q1
PCB 5
Signals
(SIGSTOP/SIGCONT)
Process 1
Process 2
. . .
Process N

Process Creation

- System calls for process management
  - `int fork(void)`
  - `int execl(char *path, char *arg0, char *arg1, …, 0)`
  - Make sure that all applications are successfully initialized before scheduling
    - A simple and dumb solution: `sleep(5)` in the scheduler after forks

Application

- Target application
  - The user program is given in a binary format
    - Stops itself
    - Performs an infinite loop
    - Prints a character passed through argv for every 200ms
    - Will be uploaded on the class board
  - E.g., > `ku_app A`
  - Assume `ku_app` locates in the same directory with the scheduler

Execution Command

- `ku_mlfq n ts`
  - `n`: number of processes
    - The first process prints 'A'
    - The second process prints 'B'
    - …
    - i.e., 1<= n <= 26
  - `ts`: number of time slices to run

Execution Command

- Examples
  - `> ku_mlfq 3 30`
    - Creates three processes that print 'A', 'B' and 'C', respectively
    - Runs for 30 time slices
  - `> ku_mlfq 3 30 >& output.txt`
    - Saves characters (i.e., A~C) into output.txt by using redirection
    - We will also test your codes in this way
      - Do not include any printing statements in your codes

Submission

- Source codes and documents
  - Source files
    - ku_mlfq.h (optional) and ku_mlfq.c
    - Will be compiled and tested on a Linux machine
      - Don't use a special library
  - Document
    - Basic design (around 3 pages)
    - Description for important functions

Function Name	Functionality	
	Parameters	
	Return Value	

Submission

- Submit your homework through eCampus
  - Deadline: 4/6 Wed. Midnight (11:59 pm)
- Cheating, plagiarism, and other anti-intellectual behavior will be dealt with severely