Simplified 리눅스 스케줄러(FIFO, RR)

국민대학교 임베디드 연구실
경 주 현

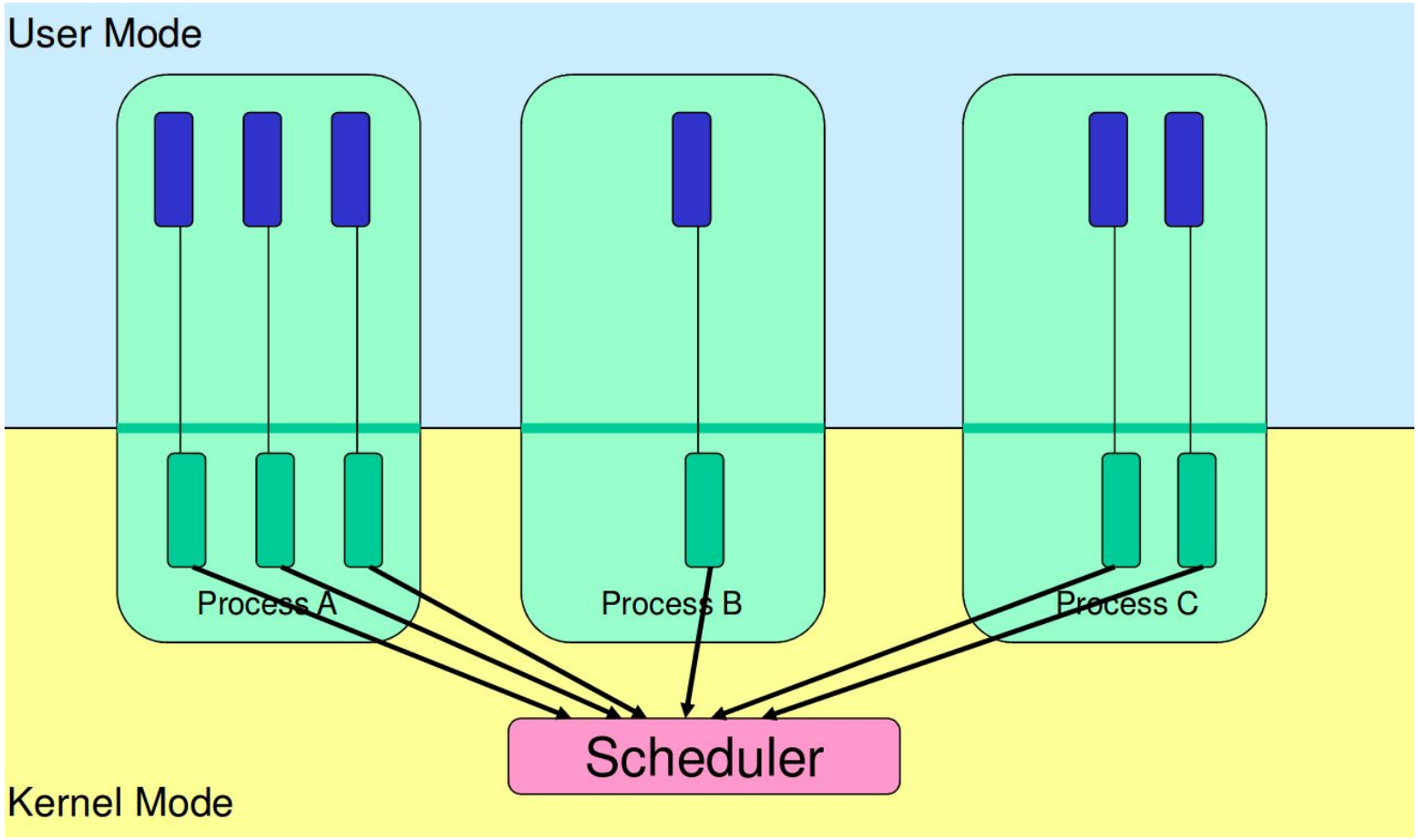
Outline

- **Simplified Linux scheduler(FIFO, RR)**
- **Application : 스케줄러 System call 사용**
- **트레이스 실습**
- **Simplified Linux scheduler(FIFO, RR) 소스**

The objectives of this pt is to learn

- How to add a pluggable CPU scheduler in Linux kernel.
- How to apply different scheduling policies to individual processes.
- Learn the internal working of a CPU scheduler.

Kernel-level thread



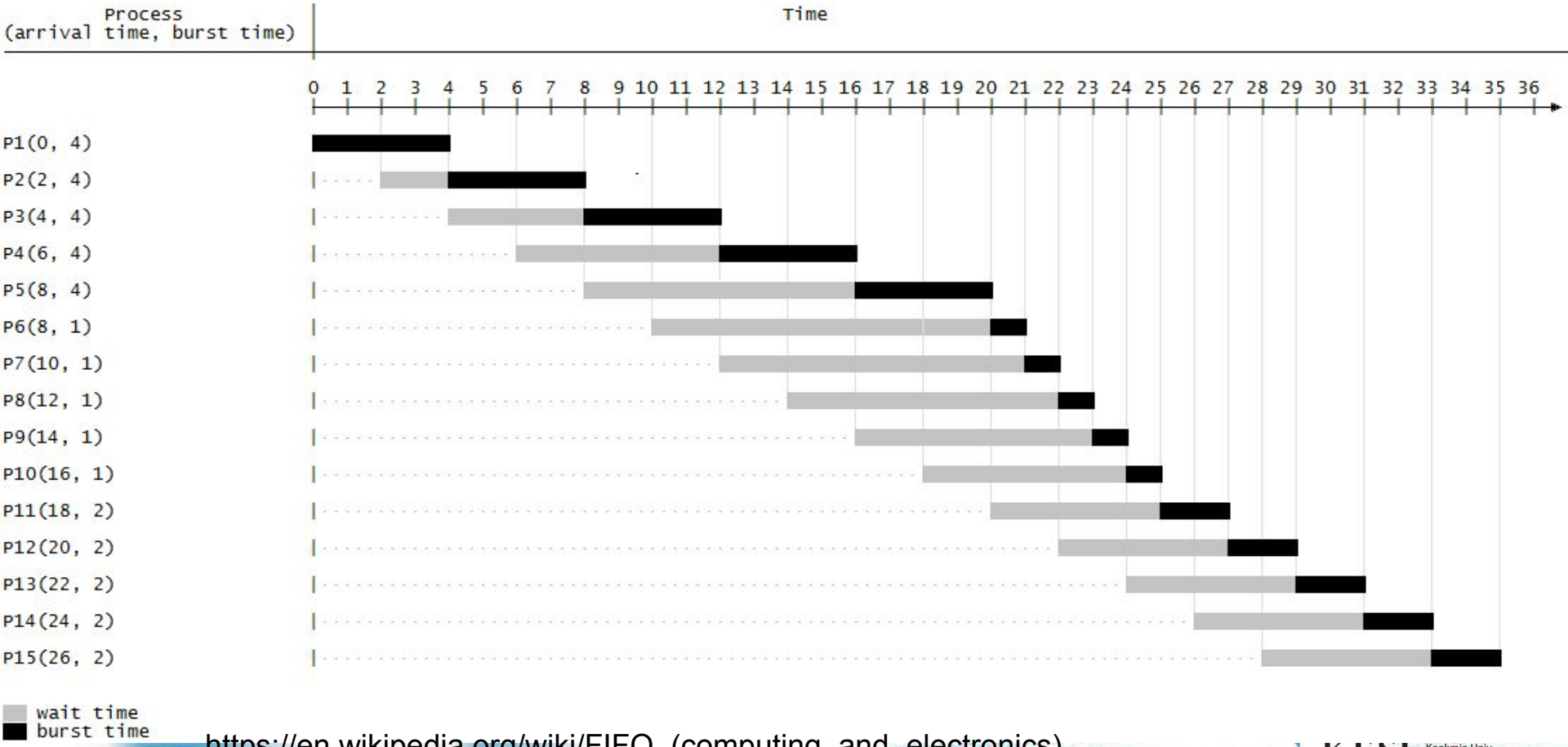
https://cgi.cse.unsw.edu.au/~cs3231/12s1/lectures/SchedulerActivations.pdf

The key decisions made in the scheduler

*"how to determine a thread's **timeslice**? and how to pick the **next thread** to run"*

FIFO

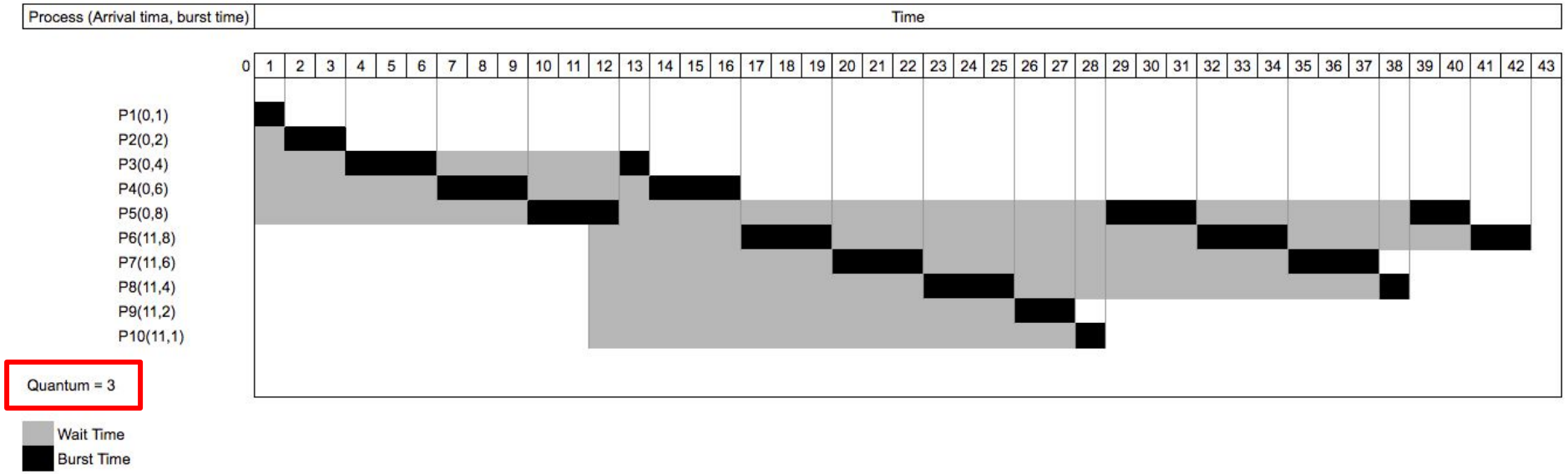
- first in, first out
- Problem : starvation



https://en.wikipedia.org/wiki/FIFO_(computing_and_electronics)

Round Robin

- **Simple, easy to implement, and starvation-free.**



커널 소스 위치

- **/home/lkc/Desktop/linux-linaro-stable**

- **Rootfs 위치**
  - /home/lkc/Desktop/linux-linaro-stable/rootfs/initrd

커널 컴파일

- **cd ~/Desktop/linux-linaro-stable**
- **make -j2 bzImage**

```
lkc@lkc-VM:~/Desktop/linux-linaro-stable$ make -j2 bzImage
  CHK     include/config/kernel.release
  CHK     include/generated/uapi/linux/version.h
  CHK     include/generated/utsrelease.h
  CHK     include/generated/timeconst.h
  CHK     include/generated/bounds.h
  CHK     include/generated/asm-offsets.h
  CALL    scripts/checksyscalls.sh
  CHK     include/generated/compile.h
Kernel: arch/x86/boot/bzImage is ready  (#1)
```

커널 실행

- **cd ~/Desktop/linux-linaro-stable**
- **make qemu**

```
lkc@lkc-VM:~/Desktop/linux-linaro-stable$ pwd
/home/lkc/Desktop/linux-linaro-stable
lkc@lkc-VM:~/Desktop/linux-linaro-stable$ make qemu
../qemu/i386-softmmu/qemu-system-i386 -smp 2 -kernel arch/x86/boot/bzImage  -initrd
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Initializing cgroup subsys cpuacct
[    0.000000] Linux version 4.4.71 (lkc@lkc-VM) (gcc version 4.8.4 (Ubuntu 4.8.4-2u
[    0.000000] x86/fpu: Legacy x87 FPU detected.
[    0.000000] x86/fpu: Using 'lazy' FPU context switches.
[    0.000000] e820: BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
[    0.000000] BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
[    0.000000] BIOS-e820: [mem 0x0000000000100000-0x0000000007fdffff] usable
[    0.000000] BIOS-e820: [mem 0x0000000007fe0000-0x0000000007ffffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000fffc0000-0x00000000ffffffff] reserved
[    0.000000] Notice: NX (Execute Disable) protection missing in CPU!
[    0.000000] SMBIOS 2.8 present.
[    0.000000] e820: last_pfn = 0x7fe0 max_arch_pfn = 0x100000
[    0.000000] MTRR: Disabled
```

User-level application

- **FIFO application**

/home/lkc/Desktop/linux-linaro-stable/rootfs/initrd/root/sched/fifo_app.c

- **RR application**

/home/lkc/Desktop/linux-linaro-stable/rootfs/initrd/root/sched/rr_app.c

FIFO scheduler 트레이스

- **trace-cmd 실행**
- **nfs 폴더로 이동**

/nfs # trace-cmd record -e sched /root/sched/fifo_app

kernelshark

kernelshark 실행 후 분석

RR scheduler 트레이스

- **trace-cmd 실행**
- **nfs 폴더로 이동**

/nfs # trace-cmd record -e sched /root/sched/rr_app

kernelshark

kernelshark 실행 후 분석

Simplified 리눅스 스케줄러

- **kernel/sched/rt.c**
- 기존 리눅스 **real-time scheduler**는 복잡함
- 분석을 위해 단순화 시킴

Key Functions in Scheduling Class

- **enqueue_task**
  - put the task into the run queue
  - increment the nr_running variable
- **dequeue_task**
  - remove the task from the run queue
  - decrement the nr_running variable
- **yield_task**
  - relinquish the CPU
- **check_preempt_curr**
  - check whether the currently running task can be preempted by a new task
- **pick_next_task**
  - choose the most appropriated task
- **load_balance**
  - trigger load balancing code

Simplified 리눅스 스케줄러

- **kernel/sched/rt.c**

enqueue_task_rt()

kernel/sched/rt.c

dequeue_task_rt()

kernel/sched/rt.c

check_preempt_curr_rt()

kernel/sched/rt.c

task_tick_rt()

kernel/sched/rt.c

Next Step.

Energy-aware scheduling: EAS

1. **CFS scheduler - Kernel level**
2. **Load Balancer(Group Scheduling, Bandwidth Control, PELT)**
3. **EAS features**

Reference

- https://pdos.csail.mit.edu/6.828/2016/schedule.html
- http://web.mit.edu/6.033
- http://www.rdrop.com/~paulmck/
- "**Is Parallel Programming Hard, And If So, What Can You Do About It?"**
- Davidlohr Bueso. 2014. Scalability techniques for practical synchronization primitives. *Commun. ACM* 58

http://queue.acm.org/detail.cfm?id=2698990

- **"CPUFreq and The Scheduler Revolution in CPU Power Management", Rafael J. Wysocki**
- **https://sites.google.com/site/embedwiki/oses/linux/pm/pm-qos**
- **https://intl.aliyun.com/forum/read-916**
- **User-level threads : co-routines**

  http://www.gamedevforever.com/291

  https://www.youtube.com/watch?v=YYtzQ355_Co

- **Scheduler Activations**
  - https://cgi.cse.unsw.edu.au/~cs3231/12s1/lectures/SchedulerActivations.pdf
- **https://en.wikipedia.org/wiki/FIFO_(computing_and_electronics)**
- **http://jake.dothome.co.kr/**
- **http://www.linuxjournal.com/magazine/completely-fair-scheduler?page=0,0**
- **https://www2.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/6_CPU_Scheduling.html**
- **"Energy Aware Scheduling", Byungchul Park, LG Electronic**
- **"Update on big.LITTLE scheduling experiments", ARM**
- **"EAS Update" 2015 september ARM**
- **"EAS Overview and Integration Guide", ARM TR**
- **"Drowsy Power Management", Matthew Lentz, SOSP 2015**
- **https://www.slideshare.net/nanik/learning-aosp-android-hardware-abstraction-layer-hal**
- https://www.youtube.com/watch?v=oTGQXqD3CNI
- https://www.youtube.com/watch?v=P80NcKUKpuo
- https://lwn.net/Articles/398470/
- "SCHED_DEADLINE: It's Alive!", ARM, 2017