

임베디드 리눅스 커널 디바이스 드라이버

Imbeded Linux Kernel Device Driver

라즈베리파이 (arm/arm64)

커널 디바이스 드라이버 이해 및 실습

정재준 <rgb13307@naver.com>

커널연구회 <www.kernel.bz>

임베디드 리눅스 커널 디바이스 드라이버

목차

- 커널연구회 (www.kernel.bz) 및 강사 소개
- 라즈베리파이 리눅스 배포본 설치
- 리눅스 커널 소스 빌드 (컴파일) 와 부팅
- 임베디드 리눅스 커널 이해
- 커널 스케줄러 , 프로세스 , 쓰레드 이해
- 커널 모듈과 디바이스 드라이버 이해
- Device Tree 이해
- 커널 모듈 프로그래밍 예제 및 실습
- Makefile 작성 및 설명
- hello, timer, kobject 소스 설명 및 실습
- 라즈베리파이 GPIO, Interrupt 소스 설명 및 실습
- 커널 디바이스 드라이버 실습
- i2c 프로토콜 설명
- i2c device driver (input touchscreen) 소스 설명

1 일차
커널 소스 빌딩
리눅스 커널 이해

2 일차
디바이스드라이버 이해
커널 모듈 코딩 실습

3 일차
GPIO, 인터럽트 실습
I2C 드라이버 실습

임베디드 리눅스 커널 디바이스 드라이버

커널연구회 (www.kernel.bz) 및 강사 소개

커널연구회 소개 :

<https://www.kernel.bz/history>

출판 서적 :

<https://www.kernel.bz/books>

교육 활동 :

<https://www.kernel.bz/edu>

커널소스 분석 블로그 :

<https://www.kernel.bz/kernel>

소스 공유 (github):

<https://www.kernel.bz/doc>



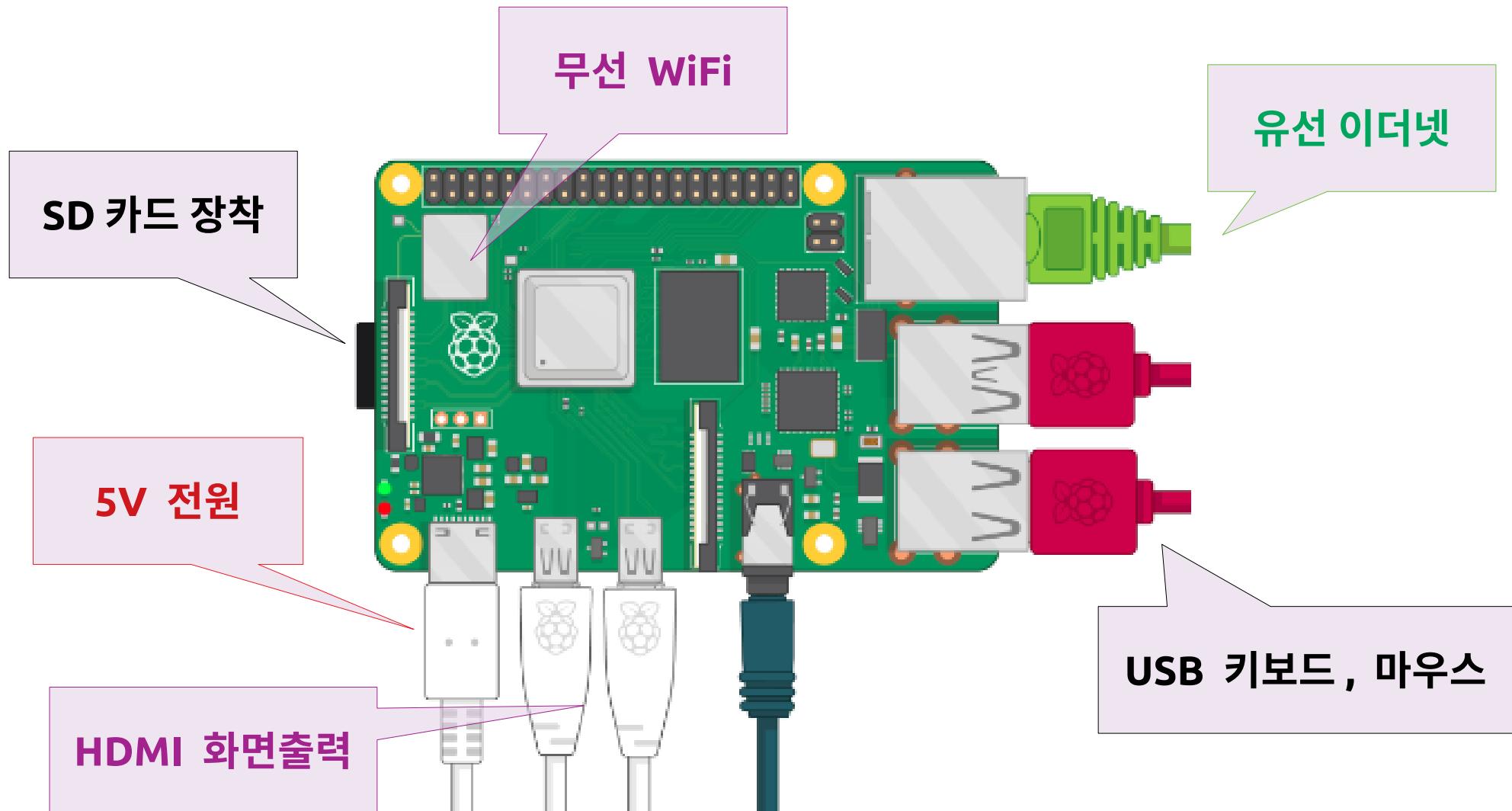
정재준 <rabi3307@naver.com>

커널연구회 <www.kernel.bz>

임베디드 리눅스 (ARM) 배포본 설치

라즈베리파이 리눅스 배포본 설치 (장치연결)

<https://www.raspberrypi.com/software/>



임베디드 리눅스 (ARM) 배포본 설치

라즈베리파이 리눅스 배포본 설치

<https://www.raspberrypi.com/software/>

Install Raspberry Pi OS using Raspberry Pi Imager

Raspberry Pi Imager is the quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card, ready to use with your Raspberry Pi. [Watch our 45-second video](#) to learn how to install an operating system using Raspberry Pi Imager.

Download and install Raspberry Pi Imager to a computer with an SD card reader. Put the SD card you'll use with your Raspberry Pi into the reader and run Raspberry Pi Imager.

[Download for Ubuntu for x86](#)

[Download for Windows](#)

[Download for macOS](#)



\$ sudo apt install rpi-imager

or

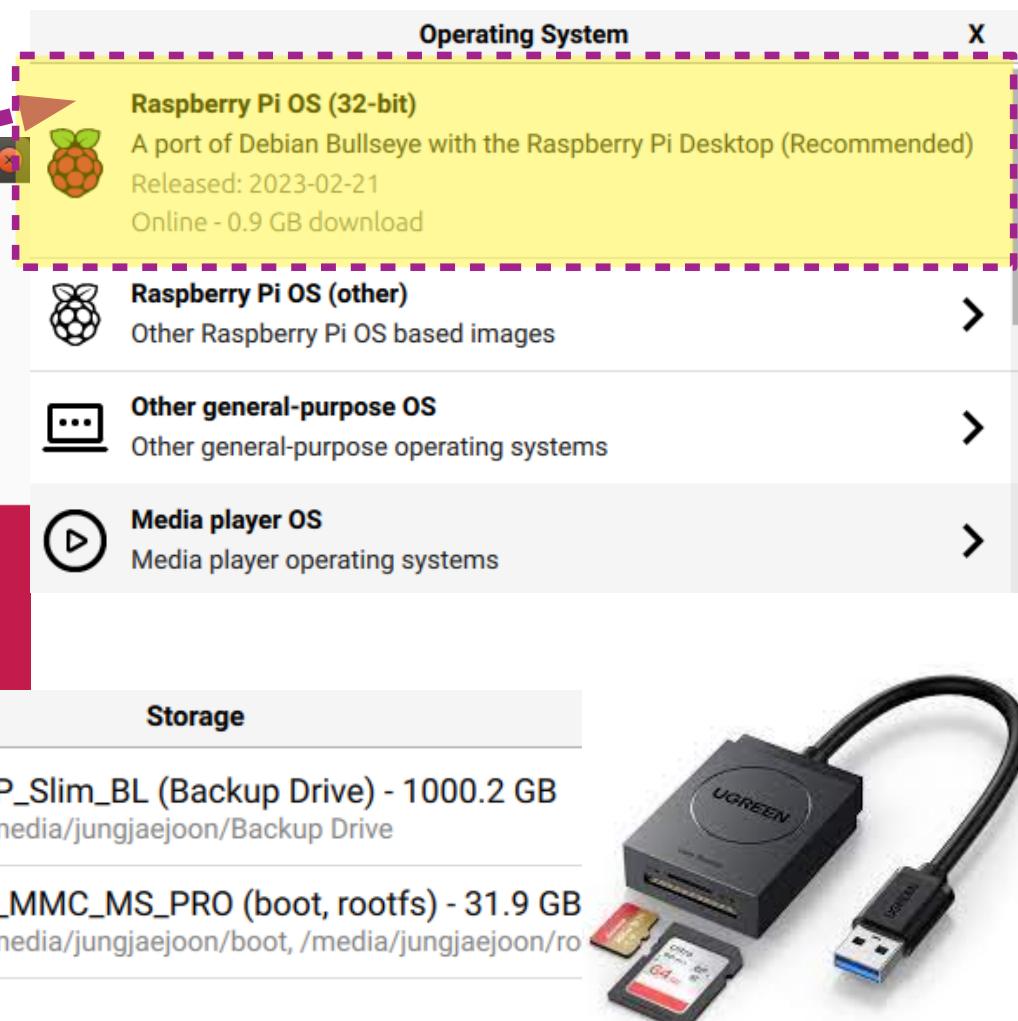
\$ **sudo snap install rpi-imager**

임베디드 리눅스 (ARM) 배포본 설치

라즈베리파이 리눅스 배포본 설치

<https://www.raspberrypi.com/software/>

\$ rpi-imager



임베디드 리눅스 (ARM) 배포본 설치

라즈베리파이 리눅스 배포본 설치

<https://www.raspberrypi.com/software/>



booting 시 설정

호스트명 (컴퓨터 이름)

네트워크 (WiFi)

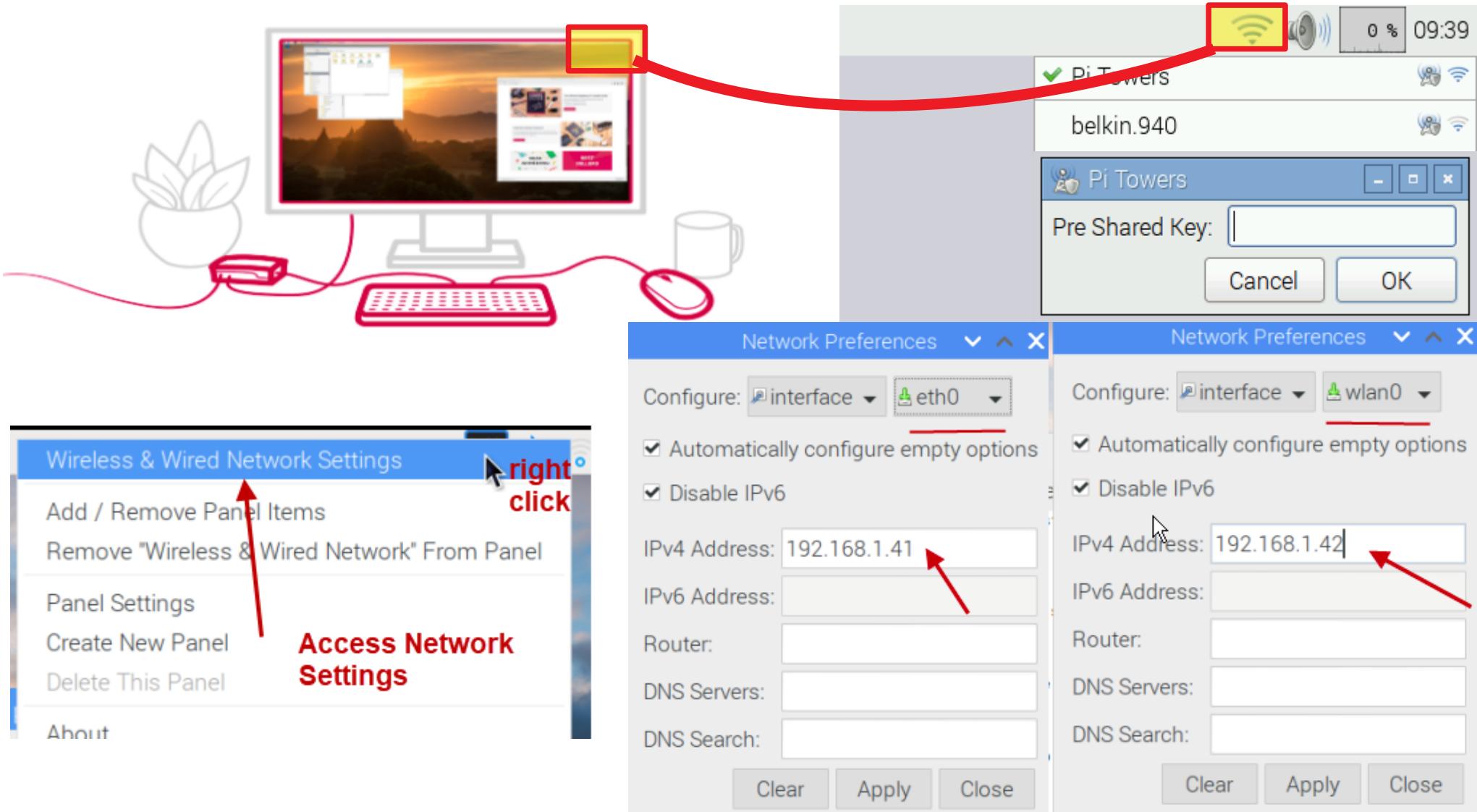
언어 , 키보드

사용자 아이디 , 암호

설치시 프로그램 업데이트는 하지 않습니다 (시간이 많이 걸림)

임베디드 리눅스 (ARM) 배포본 설치 네트워크 설정 및 인터넷 연결

<https://www.raspberrypi.com/documentation/computers/configuration.html#configuring-networking>



임베디드 리눅스 (ARM) 배포본 설치

라즈베리파이 문서 (매뉴얼)

<https://www.raspberrypi.com/documentation/>



Getting started

How to get started with your Raspberry Pi



Raspberry Pi OS

The official Raspberry Pi operating system



Configuration

Configuring your Raspberry Pi's settings



The config.txt file

Low-level settings control



The Linux kernel

How to configure and build a custom kernel for your Raspberry Pi



Remote access

Accessing your Raspberry Pi remotely

임베디드 리눅스 커널 디바이스 드라이버

임베디드 리눅스 커널 빌드와 부팅

임베디드 리눅스 커널 디바이스 드라이버

임베디드 리눅스 커널 빌드와 부팅

정재준 <rgbi3307@naver.com>

커널연구회 <www.kernel.bz>

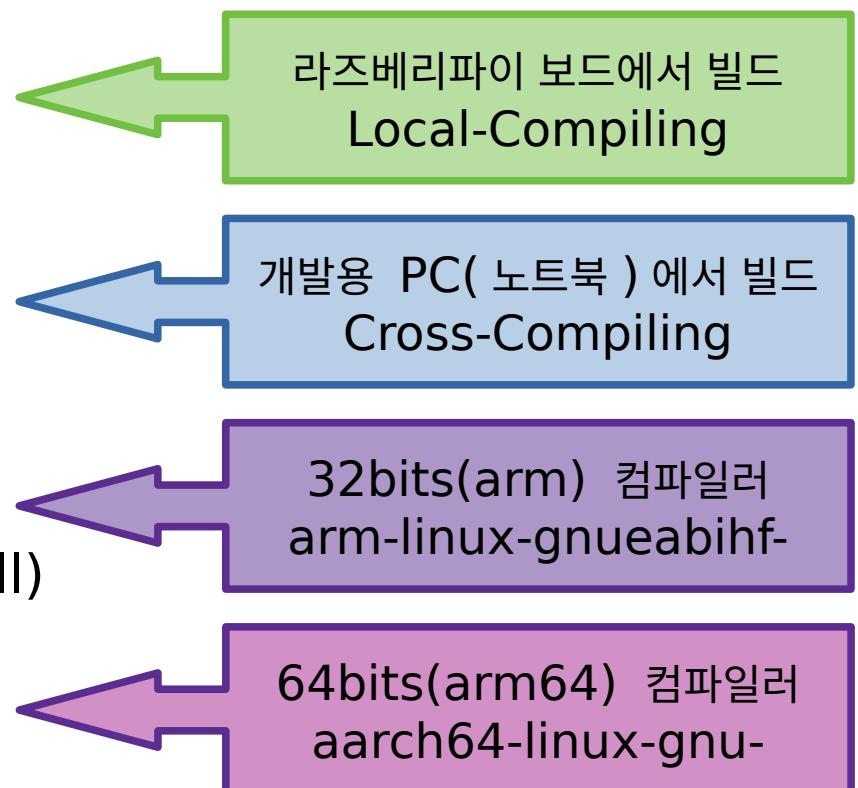
임베디드 리눅스 커널 빌드와 부팅

라즈베리파이 리눅스 커널 소스 빌드 순서

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

리눅스 커널 소스 빌드 (컴파일) 순서

1. 빌드 패키지 (도구) 설치
2. 커널 소스 다운로드
3. 빌드 설정 파일 (.config) 생성
4. 빌드 설정 메뉴 (menuconfig)
5. 빌드 하기 (make)
6. 모듈 빌드 하기 (make modules)
7. 모듈 설치 하기 (make modules_install)
8. 커널 설치 (make install)
9. 재부팅



임베디드 리눅스 커널 빌드와 부팅

라즈베리파이 리눅스 커널 소스 빌드 (Local-Compile)

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

1. 빌드 패키지 (도구) 설치

```
$ sudo apt install git bc bison flex libssl-dev make libncurses-dev
```

2. 커널 소스 다운로드

전체 소스 다운로드 (변경 이력 포함 , 1 시간 이상 소요)

```
$ git clone https://github.com/raspberrypi/linux
```

현재 버전 다운로드 (active branch)

```
$ git clone --depth=1 https://github.com/raspberrypi/linux
```

원하는 버전 (브랜치) 다운로드 (약 20 분 소요)

```
$ git clone --depth=1 --branch rpi-6.6.y https://github.com/raspberrypi/linux
```

임베디드 리눅스 커널 빌드와 부팅

라즈베리파이 리눅스 커널 소스 빌드 (Local-Compile)

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

3. 빌드 설정 파일 (.config) 생성

For Raspberry Pi 1, Zero and Zero W, and Raspberry Pi Compute Module 1 default (32-bit only) build configuration

```
cd linux  
KERNEL=kernel  
make bcmrpi_defconfig
```

For Raspberry Pi 2, 3, 3+ and Zero 2 W, and Raspberry Pi Compute Modules 3 and 3+ default 32-bit build configuration

```
cd linux  
KERNEL=kernel7  
make bcm2709_defconfig
```

라즈베리파이 3 b+, **kernel7**, 32 비트 컴파일 설정파일 (.config) 생성

For Raspberry Pi 4 and 400, and Raspberry Pi Compute Module 4 default 32-bit build configuration

```
cd linux  
KERNEL=kernel7l  
make bcm2711_defconfig
```

For Raspberry Pi 3, 3+, 4, 400 and Zero 2 W, and Compute Modules 3, 3+ and 4 default 64-bit build configuration

```
cd linux  
KERNEL=kernel8  
make bcm2711_defconfig
```

라즈베리파이 3 b+, **kernel8**, 64 비트 컴파일 설정파일 (.config) 생성

임베디드 리눅스 커널 빌드와 부팅

라즈베리파이 리눅스 커널 소스 빌드 (Local-Compile)

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

4. 빌드 (make) 및 설치 (install)

약 2 시간 20 분 소요

For the 32-bit kernel:

```
make -j4 zImage modules dtbs  
sudo make modules_install  
sudo cp arch/arm/boot/dts/broadcom/*.dtb /boot/  
sudo cp arch/arm/boot/dts/overlays/*.dtb* /boot/overlays/  
sudo cp arch/arm/boot/dts/overlays/README /boot/overlays/  
sudo cp arch/arm/boot/zImage /boot/$KERNEL.img
```

32 비트 컴파일 zImage modules dtbs
modules_install --> /lib/modules/kernel-version/

dtb 설치 --> /boot/
zImage 설치 --> /boot/

For the 64-bit kernel:

```
make -j4 Image.gz modules dtbs  
sudo make modules_install  
sudo cp arch/arm64/boot/dts/broadcom/*.dtb /boot/  
sudo cp arch/arm64/boot/dts/overlays/*.dtb* /boot/overlays/  
sudo cp arch/arm64/boot/dts/overlays/README /boot/overlays/  
sudo cp arch/arm64/boot/Image.gz /boot/$KERNEL.img
```

64 비트 컴파일 zImage modules dtbs
modules_install --> /lib/modules/kernel-version/

dtb 설치 --> /boot/
Image 설치 --> /boot/

임베디드 리눅스 커널 빌드와 부팅

라즈베리파이 리눅스 커널 Local-Compile 전체요약

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

```
$ sudo apt install git bc bison flex libssl-dev make lib ncurses-dev libncurses-dev
```

```
$ git clone --depth=1 --branch rpi-6.6.y https://github.com/raspberrypi/linux
```

```
$ cd linux
```

```
$ KERNEL=kernel7
```

```
$ make bcm2709_defconfig
```

라즈베리파이 3 b+, **kernel7**, 32 비트 컴파일 설정파일 (.config) 생성

```
$ make -j4 zImage modules dtbs  
$ sudo make modules_install
```

32 비트 컴파일 zImage modules dtbs
modules_install --> /lib/modules/kernel-version/

```
$ sudo cp arch/arm/boot/dts/broadcom/*.dtb /boot/  
$ sudo cp arch/arm/boot/dts/overlays/*.dtb* /boot/overlays/  
$ sudo cp arch/arm/boot/dts/overlays/README /boot/overlays/  
$ sudo cp /boot/$KERNEL.img /boot/$KERNEL.bak.img  
$ sudo cp arch/arm/boot/zImage /boot/$KERNEL.img
```

dtb 설치 --> /boot/
zImage 설치 --> /boot/

임베디드 리눅스 커널 빌드와 부팅

라즈베리파이 리눅스 커널 소스 빌드 (Cross-Compile)

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

1. 빌드 패키지 (도구) 설치

Install Required Dependencies and Toolchain

```
sudo apt install git bc bison flex libssl-dev make libc6-dev libncurses5-dev
```

크로스 - 컴파일용 패키지 (도구) 설치

Install the 32-bit Toolchain for a 32-bit Kernel

```
sudo apt install crossbuild-essential-armhf
```

32 비트 크로스 컴파일러 설치
arm-linux-gnueabihf-

Install the 64-bit Toolchain for a 64-bit Kernel

```
sudo apt install crossbuild-essential-arm64
```

64 비트 크로스 컴파일러 설치
aarch64-linux-gnu-

임베디드 리눅스 커널 빌드와 부팅

라즈베리파이 리눅스 커널 소스 빌드 (Cross-Compile)

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

2. 커널 소스 다운로드

전체 소스 다운로드 (변경 이력 포함, 1 시간 이상 소요)

```
$ git clone https://github.com/raspberrypi/linux
```

현재 버전 다운로드 (active branch)

```
$ git clone --depth=1 https://github.com/raspberrypi/linux
```

원하는 버전 (브랜치) 다운로드 (약 20 분 소요)

```
$ git clone --depth=1 --branch rpi-6.6.y https://github.com/raspberrypi/linux
```

3. 안정화 (longterm) 버전 선택

```
$ cd linux
```

```
$ git checkout rpi-6.6.y
```

임베디드 리눅스 커널 빌드와 부팅

라즈베리파이 리눅스 커널 소스 빌드 (Cross-Compile)

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

4. 빌드 설정 파일 (.config) 생성 (32 비트)

↳ For Raspberry Pi 1, Zero and Zero W, and Raspberry Pi Compute Module 1:

```
cd linux  
KERNEL=kernel  
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcmrpi_defconfig
```

↳ For Raspberry Pi 2, 3, 3+ and Zero 2 W, and Raspberry Pi Compute Modules 3 and 3+:

```
cd linux  
KERNEL=kernel7  
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2709_defconfig
```

라즈베리파이 3 b+, **kernel7**, 32 비트 컴파일 설정파일 (.config) 생성

↳ For Raspberry Pi 4 and 400, and Raspberry Pi Compute Module 4:

```
cd linux  
KERNEL=kernel7l  
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2711_defconfig
```

임베디드 리눅스 커널 빌드와 부팅

라즈베리파이 리눅스 커널 소스 빌드 (Cross-Compile)

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

4. 빌드 설정 파일 (.config) 생성 (64 비트)

For Raspberry Pi 3, 3+, 4, 400 and Zero 2 W, and Compute Modules 3, 3+ and 4:

```
cd linux  
KERNEL=kernel8  
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- bcm2711_defconfig
```

라즈베리파이 3 b+, **kernel8**, 64 비트 컴파일 설정파일 (.config) 생성

임베디드 리눅스 커널 빌드와 부팅

라즈베리파이 리눅스 커널 소스 빌드 (Cross-Compile)

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

5. 빌드 하기 (make)

약 40 분 소요

For all 32-bit Builds

32 비트 컴파일 zImage modules dtbs

```
make -j$(nproc) ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage modules dtbs
```

For all 64-bit Builds

```
make -j$(nproc) ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- Image modules dtbs
```

64 비트 컴파일 Image modules dtbs

임베디드 리눅스 커널 빌드와 부팅

라즈베리파이 리눅스 커널 소스 빌드 (Cross-Compile)

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

6. 모듈 설치 하기 (make modules_install), 마이크로 SD 카드 마운트

First, use `lsblk` before and after plugging in your SD card to identify it. You should end up with something like:

```
sdb
  sdb1
  sdb2
```

with `sdb1` being the `FAT` filesystem (boot) partition, and `sdb2` being the `ext4` filesystem (root) partition.

Mount these first, adjusting the partition letter as necessary:

```
mkdir mnt
mkdir mnt/fat32
mkdir mnt/ext4
sudo mount /dev/sdb1 mnt/fat32
sudo mount /dev/sdb2 mnt/ext4
```

임베디드 리눅스 커널 빌드와 부팅

라즈베리파이 리눅스 커널 소스 빌드 (Cross-Compile)

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

7. 모듈 설치 하기 (make modules_install)

For 32-bit

```
sudo env PATH=$PATH make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- \
INSTALL_MOD_PATH=mnt/ext4 modules_install
```

32 비트 modules_install --> mnt/ext4/lib/modules/kernel-version/

For 64-bit

```
sudo env PATH=$PATH make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- \
INSTALL_MOD_PATH=mnt/ext4 modules_install
```

64 비트 modules_install --> mnt/ext4/lib/modules/kernel-version/

임베디드 리눅스 커널 빌드와 부팅

라즈베리파이 리눅스 커널 소스 빌드 (Cross-Compile)

https://www.raspberrypi.com/documentation/computers/linux_kernel.html

8. 커널 설치 (install)

For 32-bit

```
sudo cp mnt/fat32/$KERNEL.img mnt/fat32/$KERNEL-backup.img  
sudo cp arch/arm/boot/zImage mnt/fat32/$KERNEL.img  
sudo cp arch/arm/boot/dts/broadcom/*.dtb mnt/fat32/  
sudo cp arch/arm/boot/dts/overlays/*.dtb* mnt/fat32/overlays/  
sudo cp arch/arm/boot/dts/overlays/README mnt/fat32/overlays/  
sudo umount mnt/fat32  
sudo umount mnt/ext4
```

zImage 설치 --> mnt/fat32/boot/**kernel7**.img
dtb 설치 --> mnt/fat32/boot/

For 64-bit

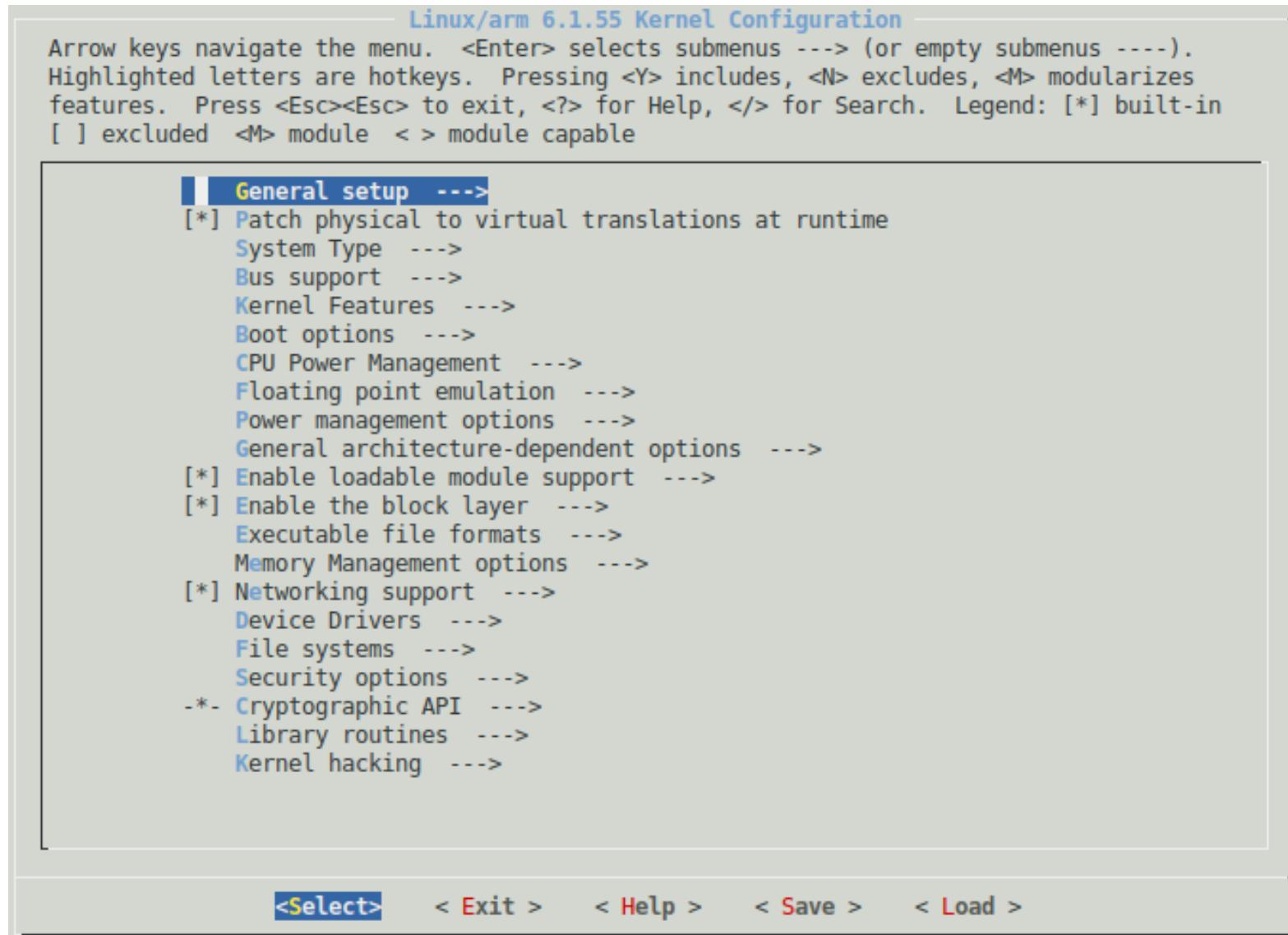
```
sudo cp mnt/fat32/$KERNEL.img mnt/fat32/$KERNEL-backup.img  
sudo cp arch/arm64/boot/Image mnt/fat32/$KERNEL.img  
sudo cp arch/arm64/boot/dts/broadcom/*.dtb mnt/fat32/  
sudo cp arch/arm64/boot/dts/overlays/*.dtb* mnt/fat32/overlays/  
sudo cp arch/arm64/boot/dts/overlays/README mnt/fat32/overlays/  
sudo umount mnt/fat32  
sudo umount mnt/ext4
```

Image 설치 --> mnt/fat32/boot/**kernel8**.img
dtb 설치 --> mnt/fat32/boot/

임베디드 리눅스 커널 빌드와 부팅

리눅스 커널 소스 빌드 환경설정 (make menuconfig)

https://www.raspberrypi.com/documentation/computers/linux_kernel.html



임베디드 리눅스 커널 디바이스 드라이버

임베디드 리눅스 커널 이해

임베디드 리눅스 커널 디바이스 드라이버

임베디드 리눅스 커널 이해

정재준 <rgbi3307@naver.com>

커널연구회 <www.kernel.bz>

임베디드 리눅스 커널 이해

리눅스 커널 소스 참고 사이트

<https://www.kernel.org/>

리눅스 커널 소스 보관소 (Kernel Archives):

<https://www.kernel.org/>

커널소스 GitHub:

<https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git>

리눅스 커널 메일링 리스트 :

<https://lkml.org/>

리눅스 커널 참조 사이트 (키워드 검색):

<https://elixir.bootlin.com/linux/latest/source>

리눅스 커널 다이어그램 :

<https://makelinux.github.io/kernel/diagram/>

<https://makelinux.github.io/kernel/map/>

커널연구회 :

<https://www.kernel.bz>

임베디드 리눅스 커널 이해

리눅스 커널 소스 보관소, 배포 사이트

<https://www.kernel.org/>

Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Release

6.5.6 

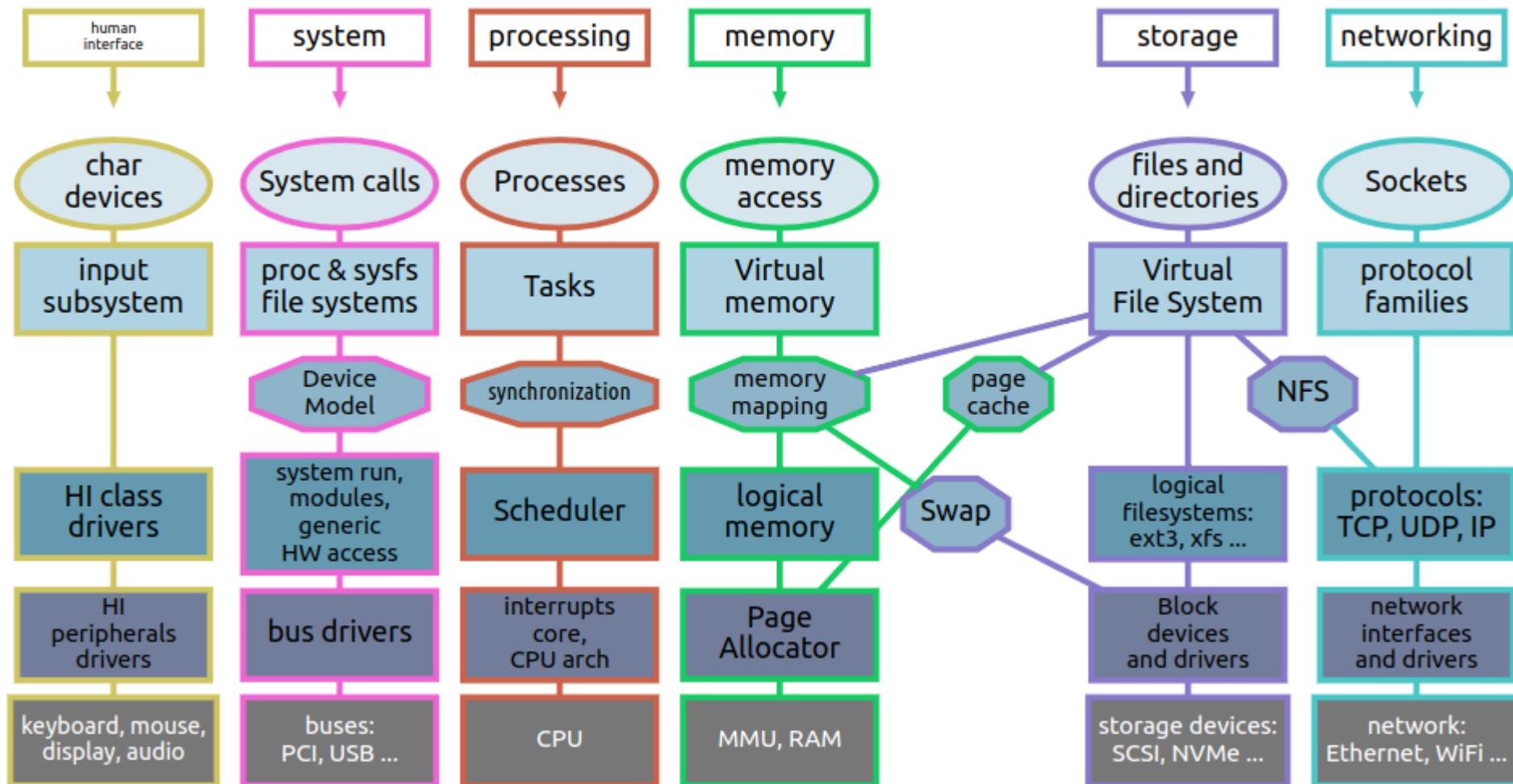
mainline:	6.6-rc4	2023-10-01	[tarball]	[patch]	[inc. patch]	[view diff]	[browse]
stable:	6.5.6	2023-10-06	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]
stable:	6.4.16 [EOL]	2023-09-13	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	6.1.56	2023-10-06	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	5.15.134	2023-10-06	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	5.10.197	2023-09-23	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	5.4.257	2023-09-23	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	4.19.295	2023-09-23	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]
longterm:	4.14.326	2023-09-23	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]
linux-next:	next-20231006	2023-10-06					[browse]

임베디드 리눅스 커널 이해

리눅스 커널 소스 다이어그램

<https://makelinux.github.io/kernel/diagram/>

Linux kernel diagram



임베디드 리눅스 커널 디바이스 드라이버

커널 스케줄러, 프로세스, 쓰레드 이해

임베디드 리눅스 커널 디바이스 드라이버
커널 스케줄러, 프로세스, 쓰레드 이해

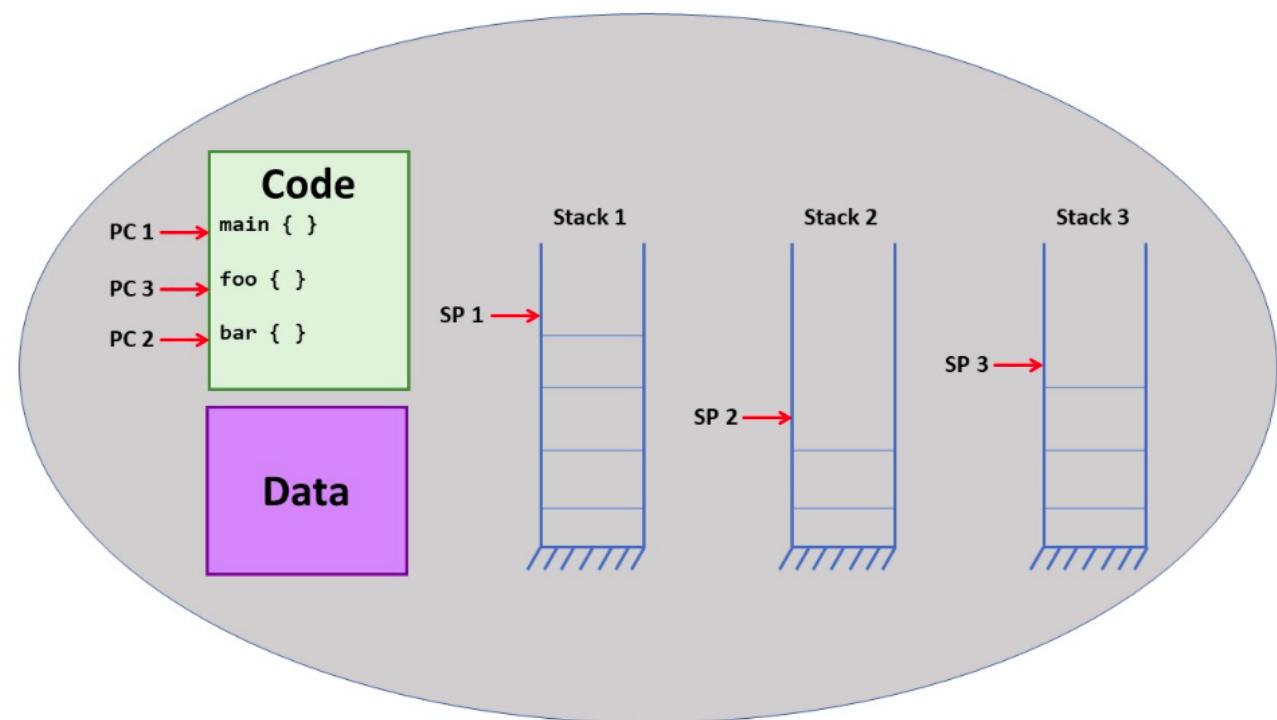
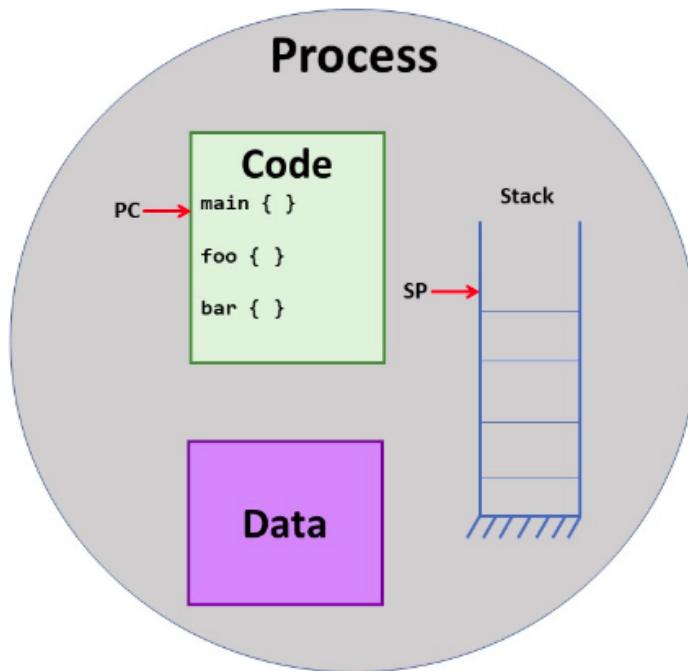
정재준 <rgb13307@naver.com>

커널연구회 <www.kernel.bz>

커널 스케줄러, 프로세스, 쓰레드 이해

프로세스, 쓰레드 개념 이해

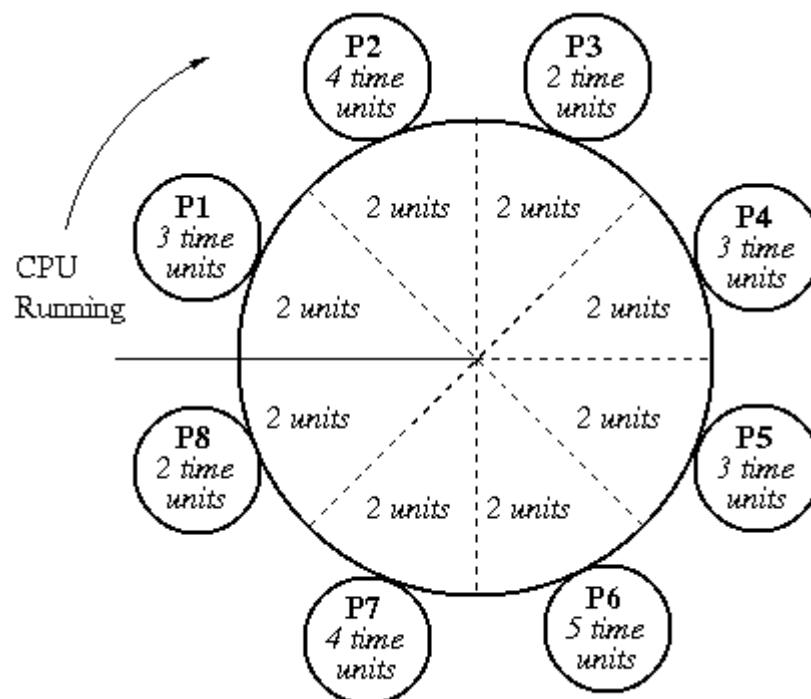
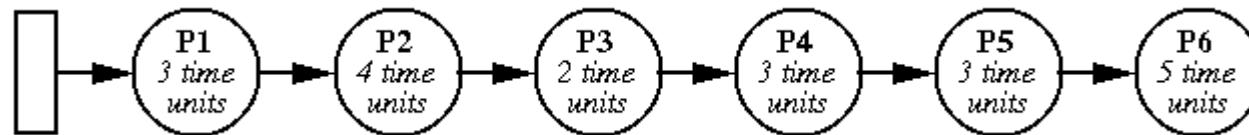
<https://pages.cs.wisc.edu/~bart/537/lecturenotes/processes-threads.html>



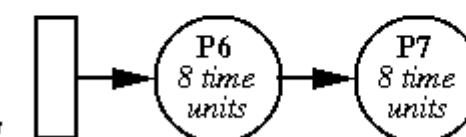
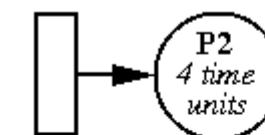
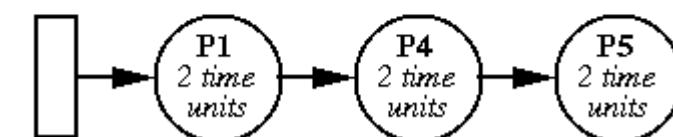
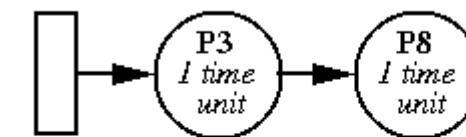
커널 스케줄러, 프로세스, 쓰레드 이해

프로세스 (태스크) 스케줄링 개념 이해

<https://pages.cs.wisc.edu/~bart/537/lecturenotes/s11.html>



High priority



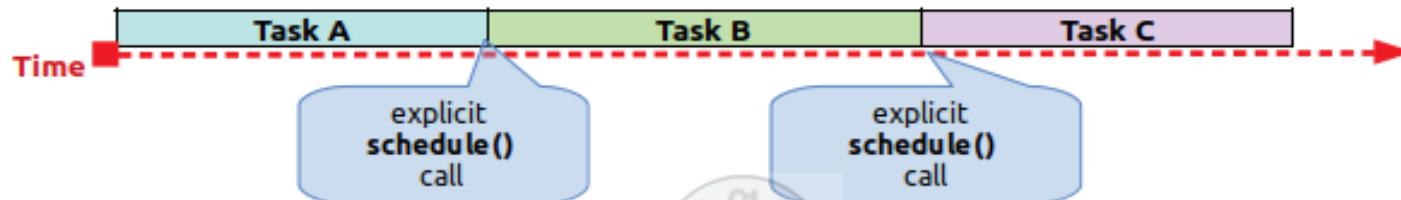
Low priority

커널 스케줄러, 프로세스, 쓰레드 이해

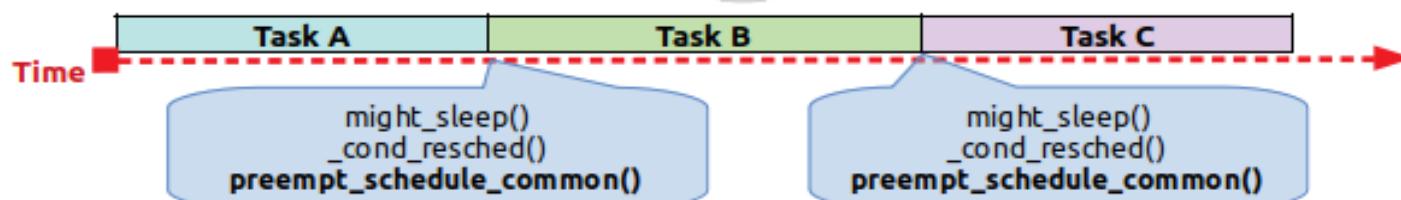
리눅스 커널 스케줄러 문맥교환 개념 이해

<https://www.kernel.bz/>

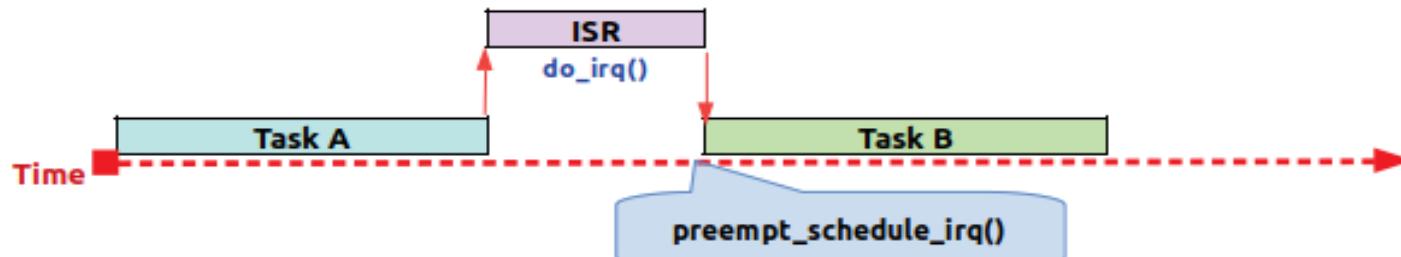
CONFIG_PREEMPT_NONE (Server)



CONFIG_PREEMPT_VOLUNTARY (Desktop)



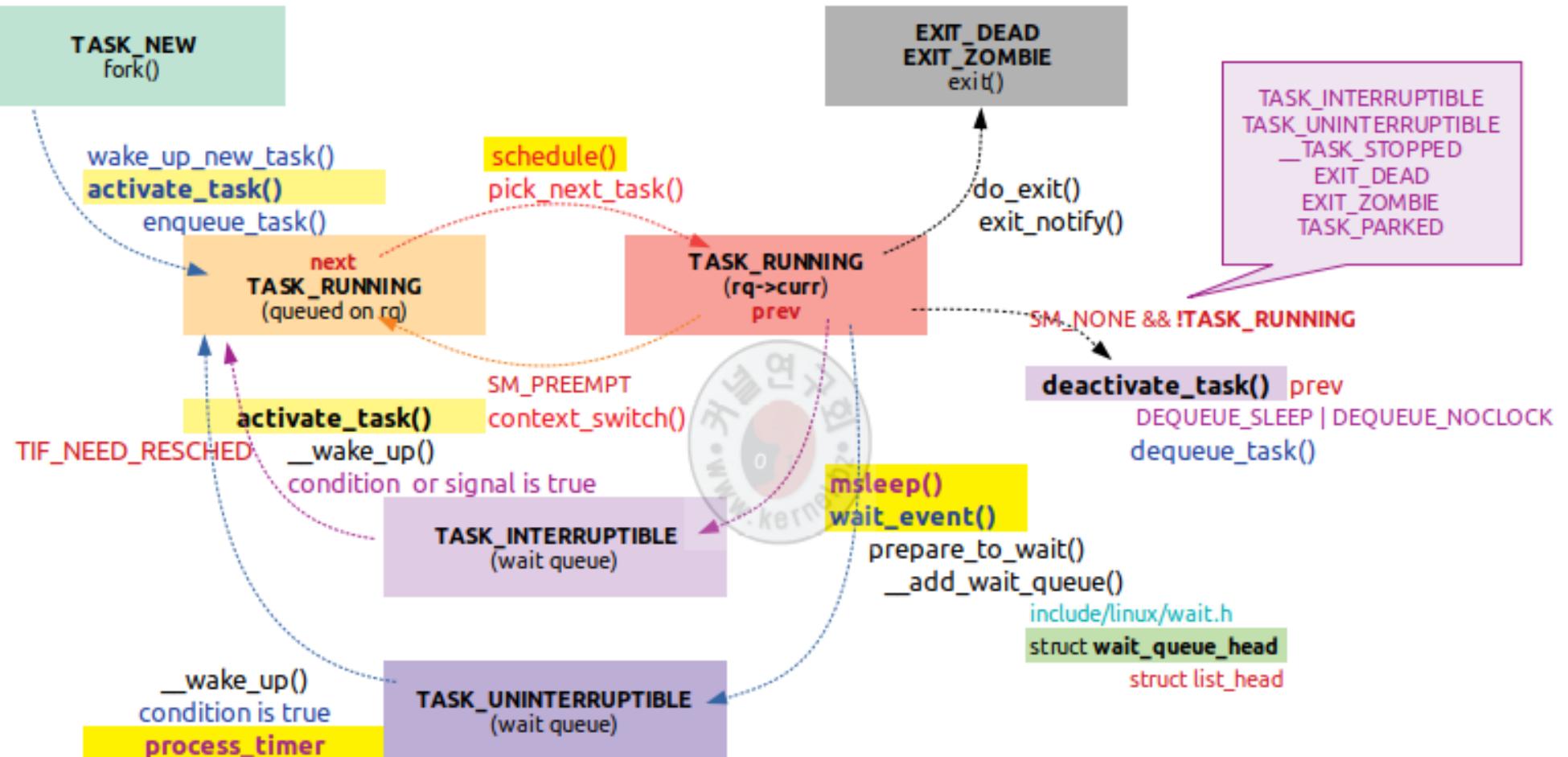
CONFIG_PREEMPT (Low-Latency Desktop)



커널 스케줄러, 프로세스, 쓰레드 이해

리눅스 커널 스케줄러 상태변화 다이어그램

<https://www.kernel.bz/>



커널 스케줄러, 프로세스, 쓰레드 이해

리눅스 커널 스케줄러 종류

<https://www.kernel.bz/>

1. Stop Scheduler (STOP: 구조체명 : stop_sched_class)

다른 일들은 정지하고 하나의 태스크만 실행할 경우

2. DeadLine Scheduler (DL: 구조체명 : dl_sched_class)

마감시간이 정해진 태스크는 DL 스케줄러가 관리

3. RealTime Scheduler (RT: 구조체명 : rt_sched_class)

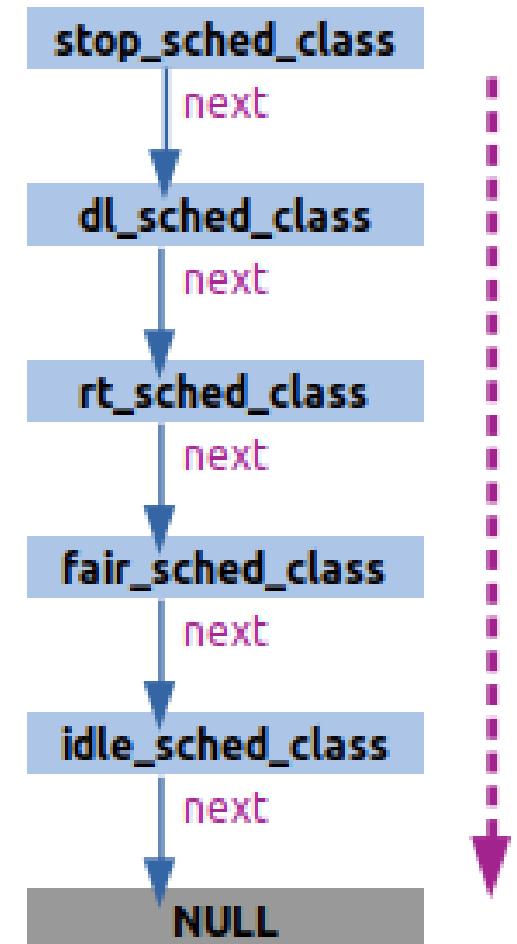
약속시간마다 처리해야 하는 태스크는 RT 스케줄러가 관리

4. Fair Scheduler (CFS: 구조체명 : fair_sched_class)

우선순위별로 공평하게 처리해야 하는 태스크는 CFS 스케줄러가 관리

5. Idle Scheduler (IDLE: 구조체명 : idle_sched_class)

휴식을 취하는 태스크는 IDLE 스케줄러가 관리



커널 스케줄러, 프로세스, 쓰레드 이해

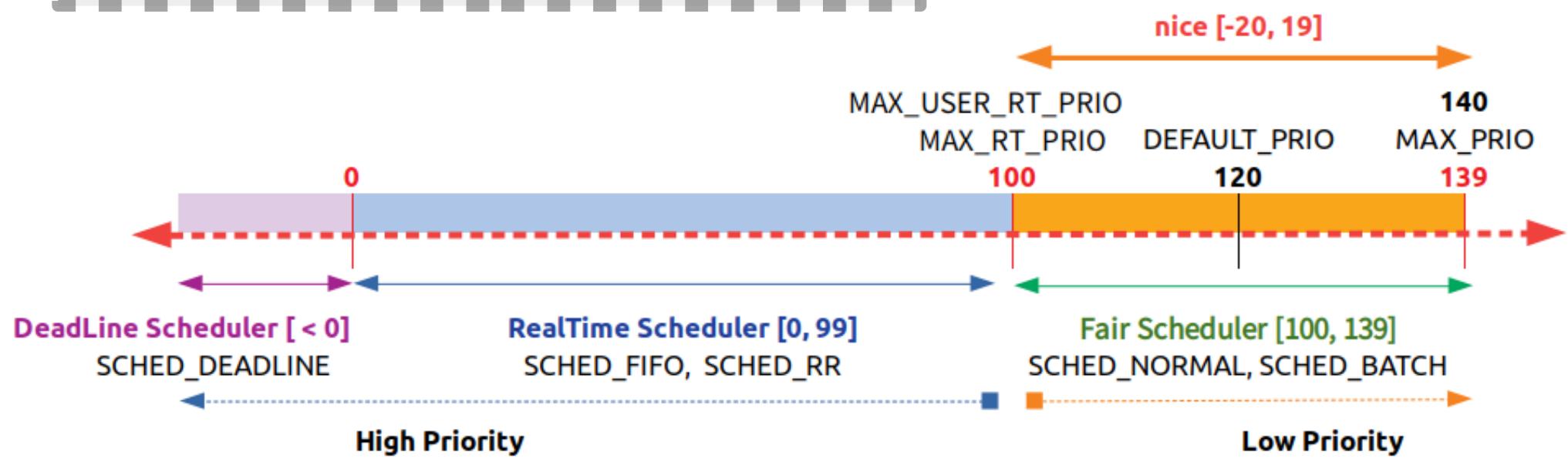
커널 스케줄러 정책 (policy) 과 우선순위 (priority)

<https://www.kernel.bz/>

```
#include <sched.h>

int sched_setscheduler(pid_t pid, int policy,
                      const struct sched_param *param);

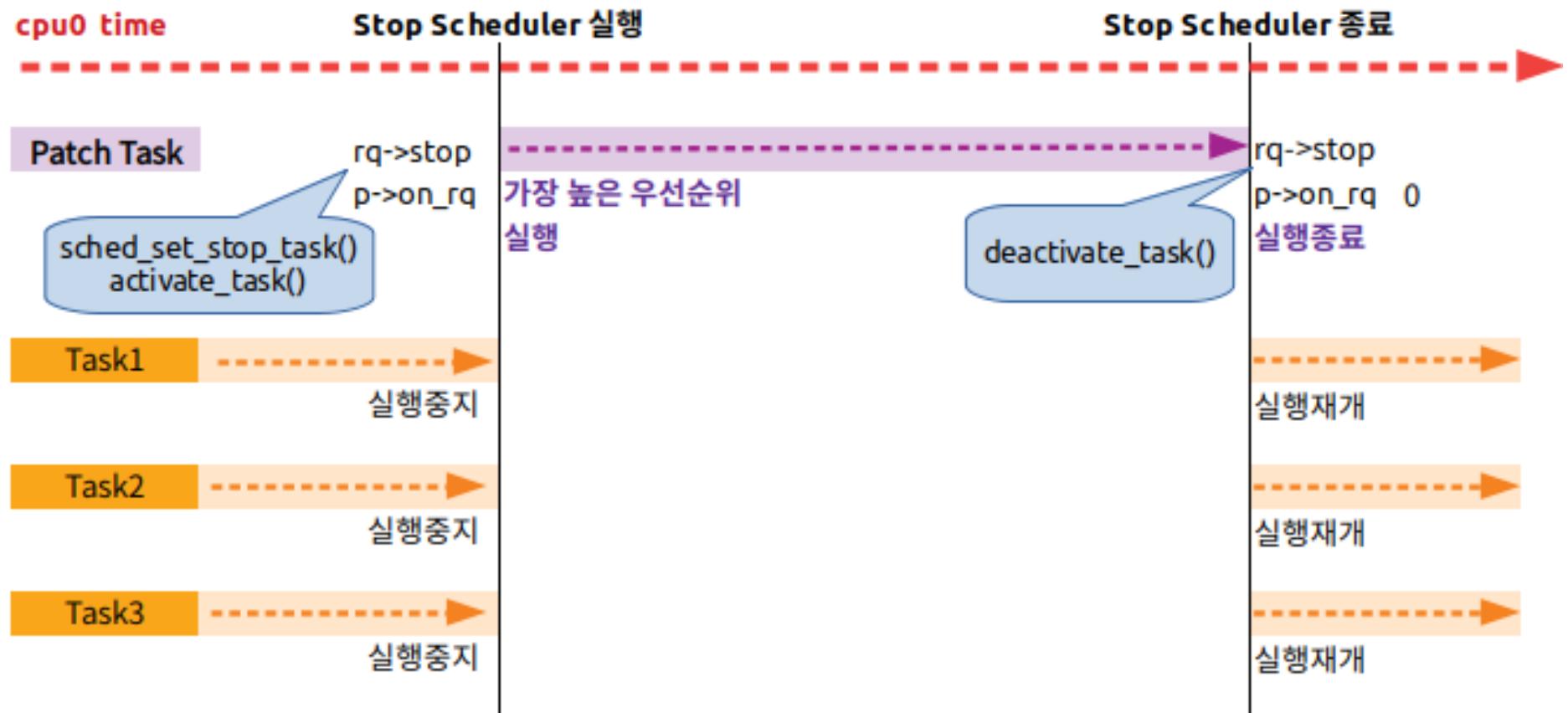
int sched_getscheduler(pid_t pid);
```



커널 스케줄러, 프로세스, 쓰레드 이해

커널 커널 스케줄러 이해 (Stop Scheduler)

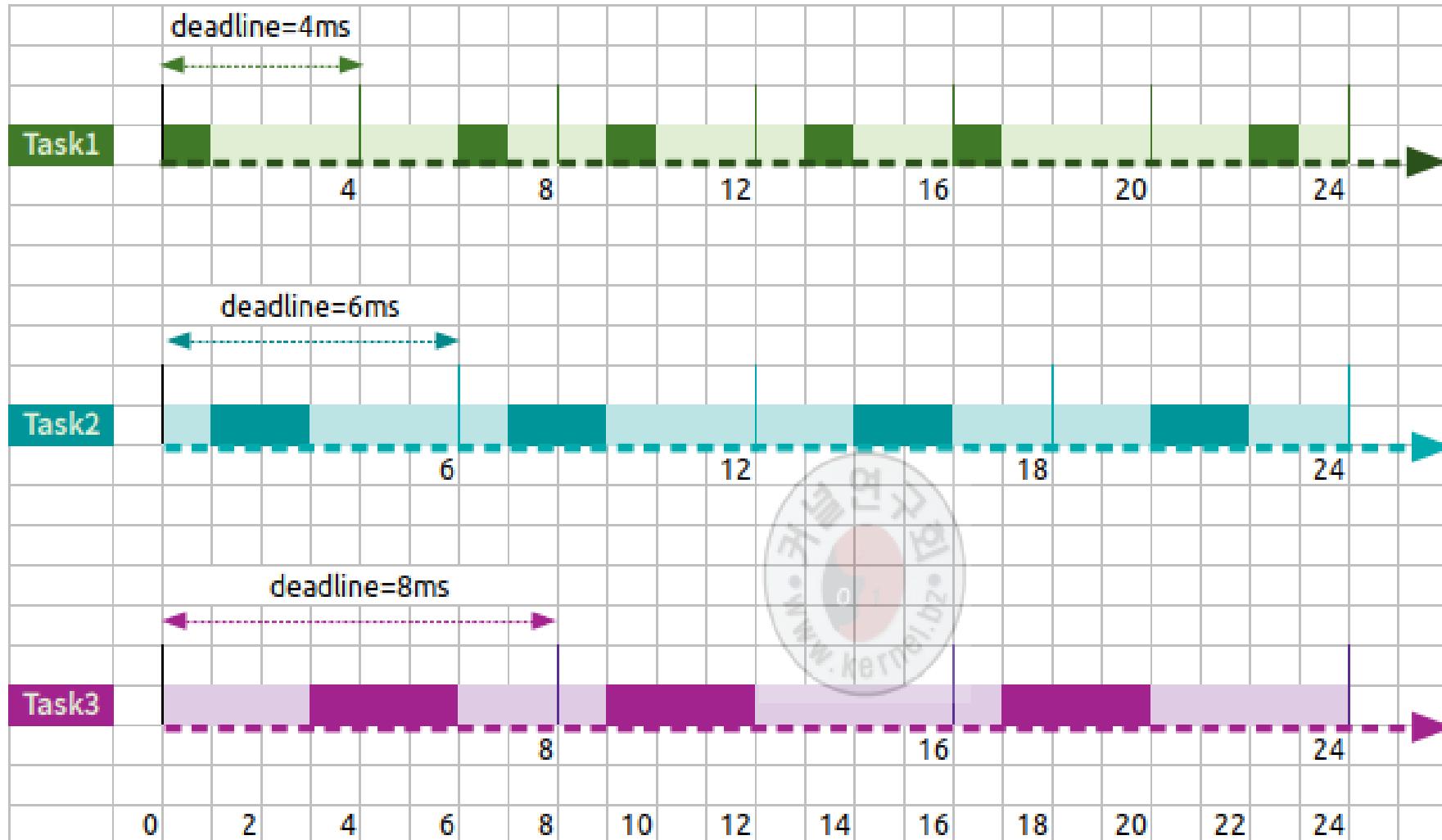
<https://www.kernel.bz/>



커널 스케줄러, 프로세스, 쓰레드 이해

커널 스케줄러 이해 (DL: DeadLine Scheduler)

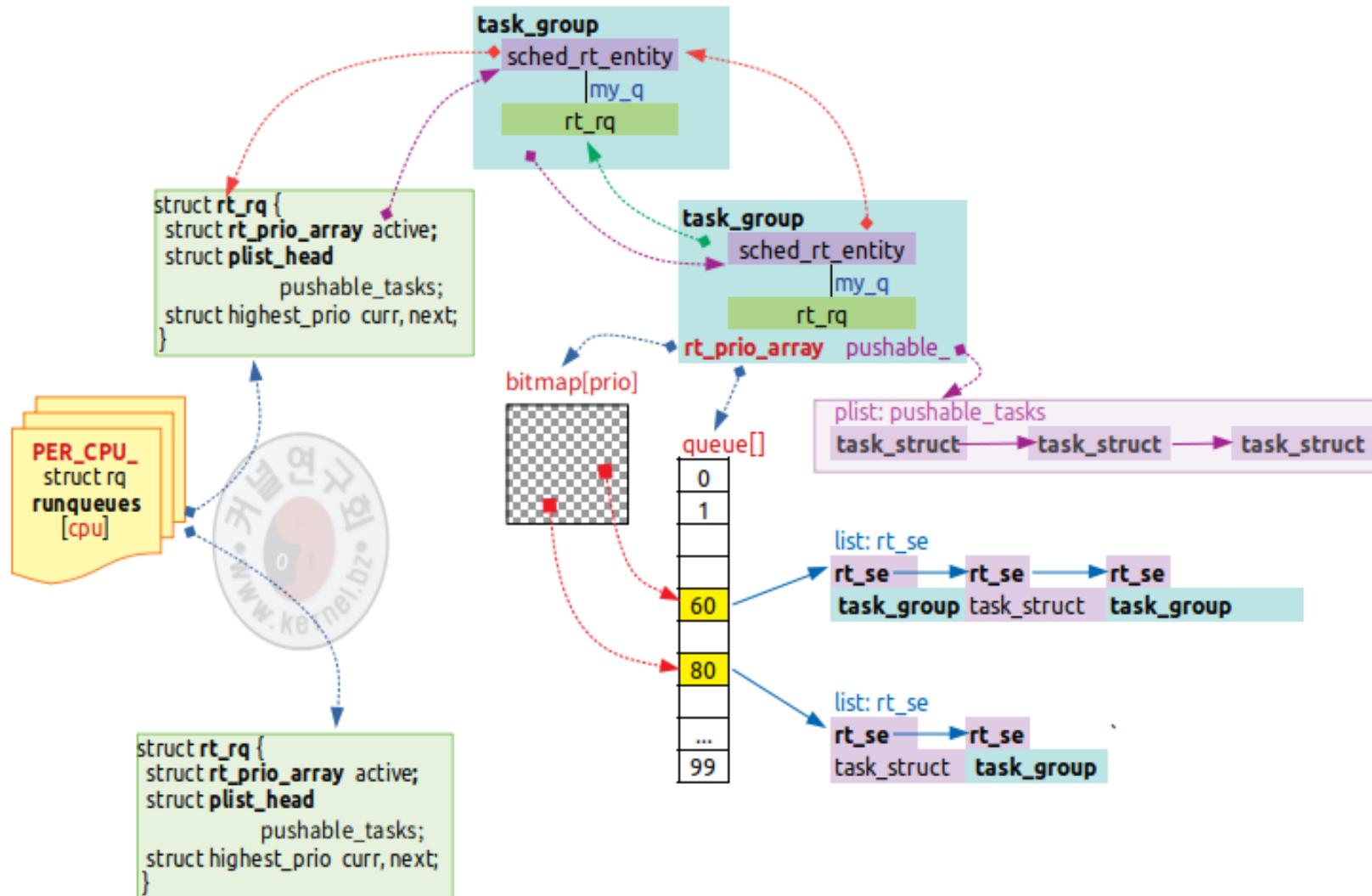
<https://www.kernel.bz/>



커널 스케줄러, 프로세스, 쓰레드 이해

커널 스케줄러 이해 (RT: RealTime Scheduler)

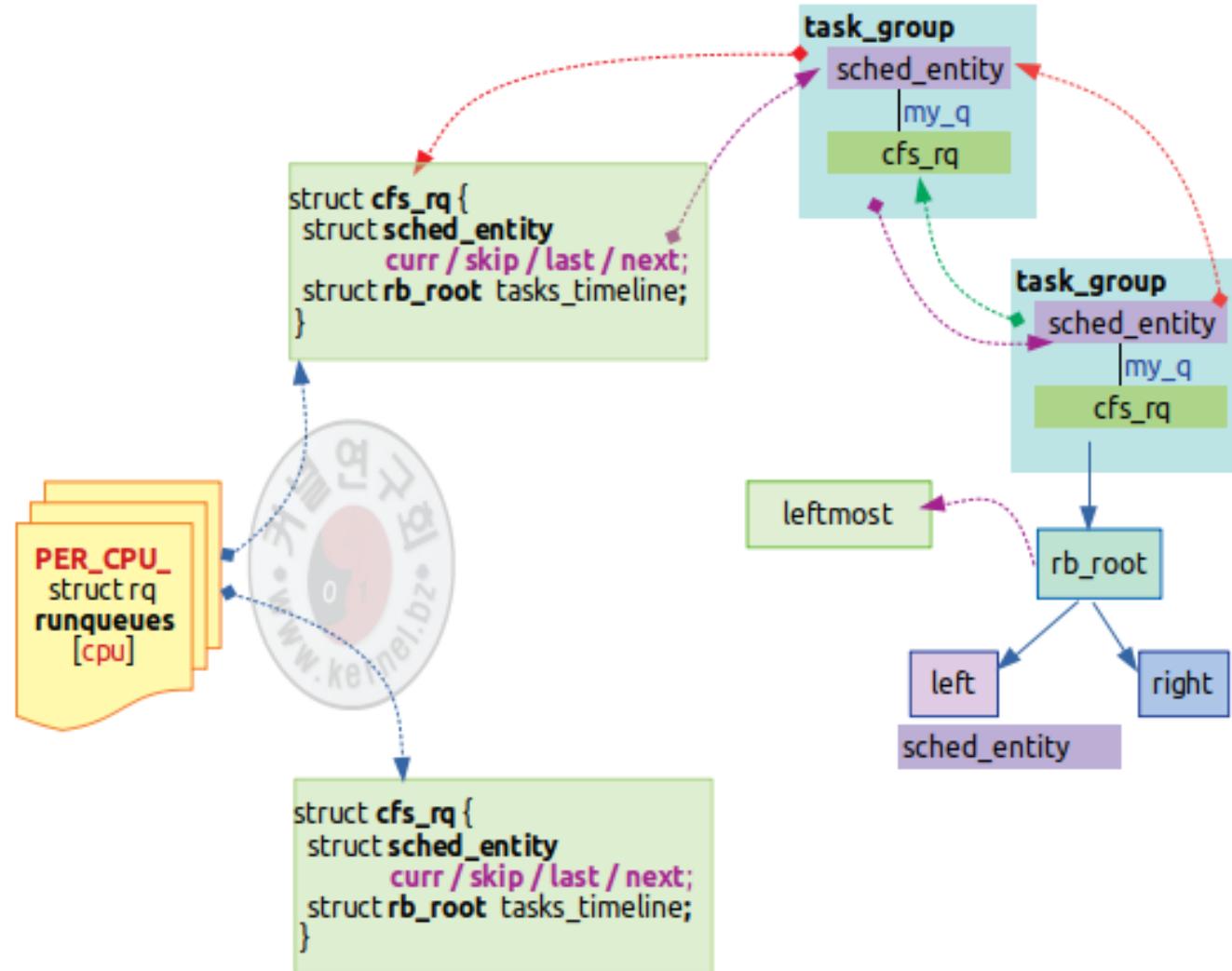
<https://www.kernel.bz/>



커널 스케줄러, 프로세스, 쓰레드 이해

커널 스케줄러 이해 (CFS: Complete Fair Scheduler)

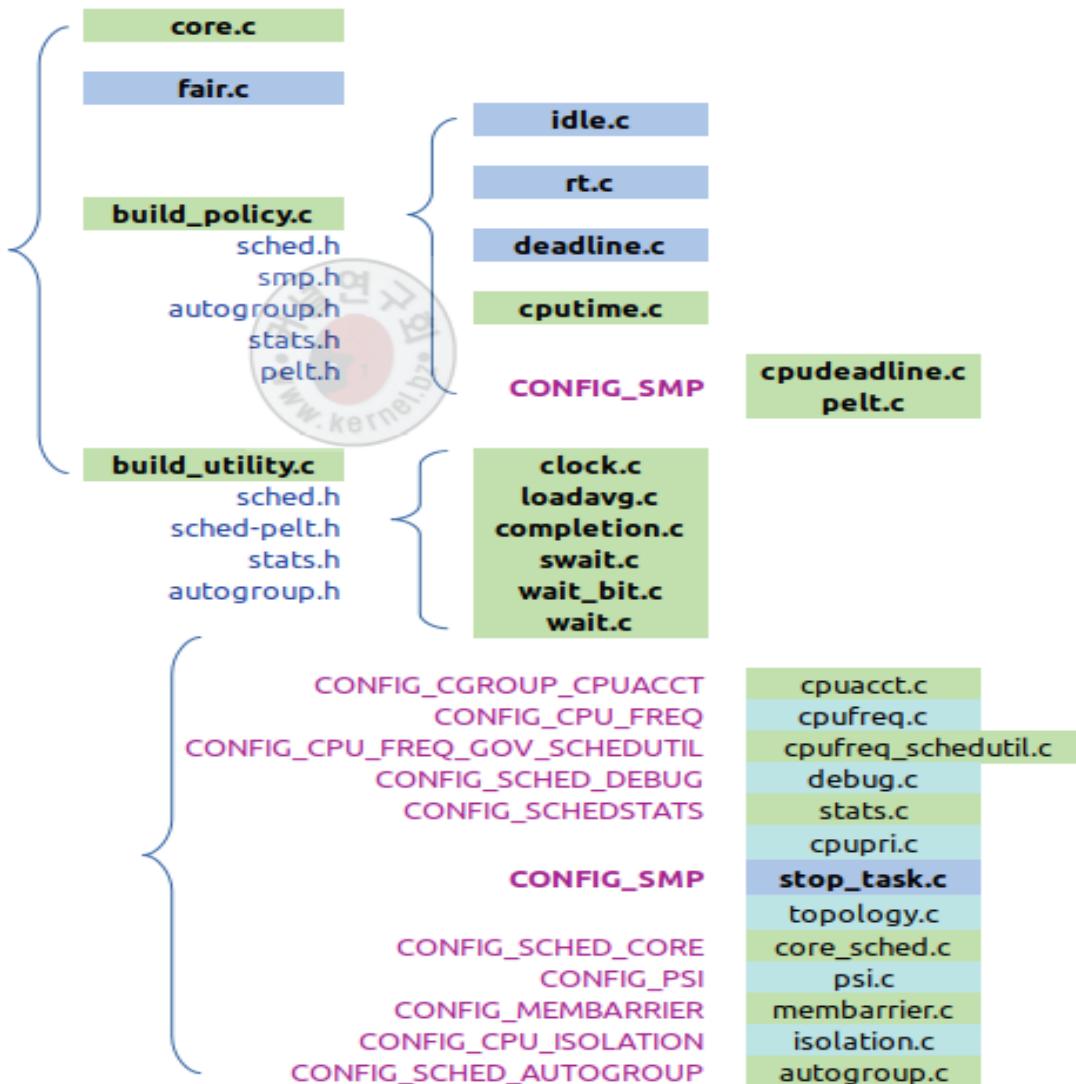
<https://www.kernel.bz/>



커널 스케줄러, 프로세스, 쓰레드 이해

리눅스 커널 스케줄러 소스 빌드 구조

<https://www.kernel.bz/>



임베디드 리눅스 커널 디바이스 드라이버

커널 메모리 주소 관리 이해

임베디드 리눅스 커널 디바이스 드라이버

커널 메모리 주소 관리 이해

정재준 <rgb13307@naver.com>

커널연구회 <www.kernel.bz>

커널 메모리 주소 관리 및 메모리맵 이해

메모리 맵 (메모리 주소 공간)

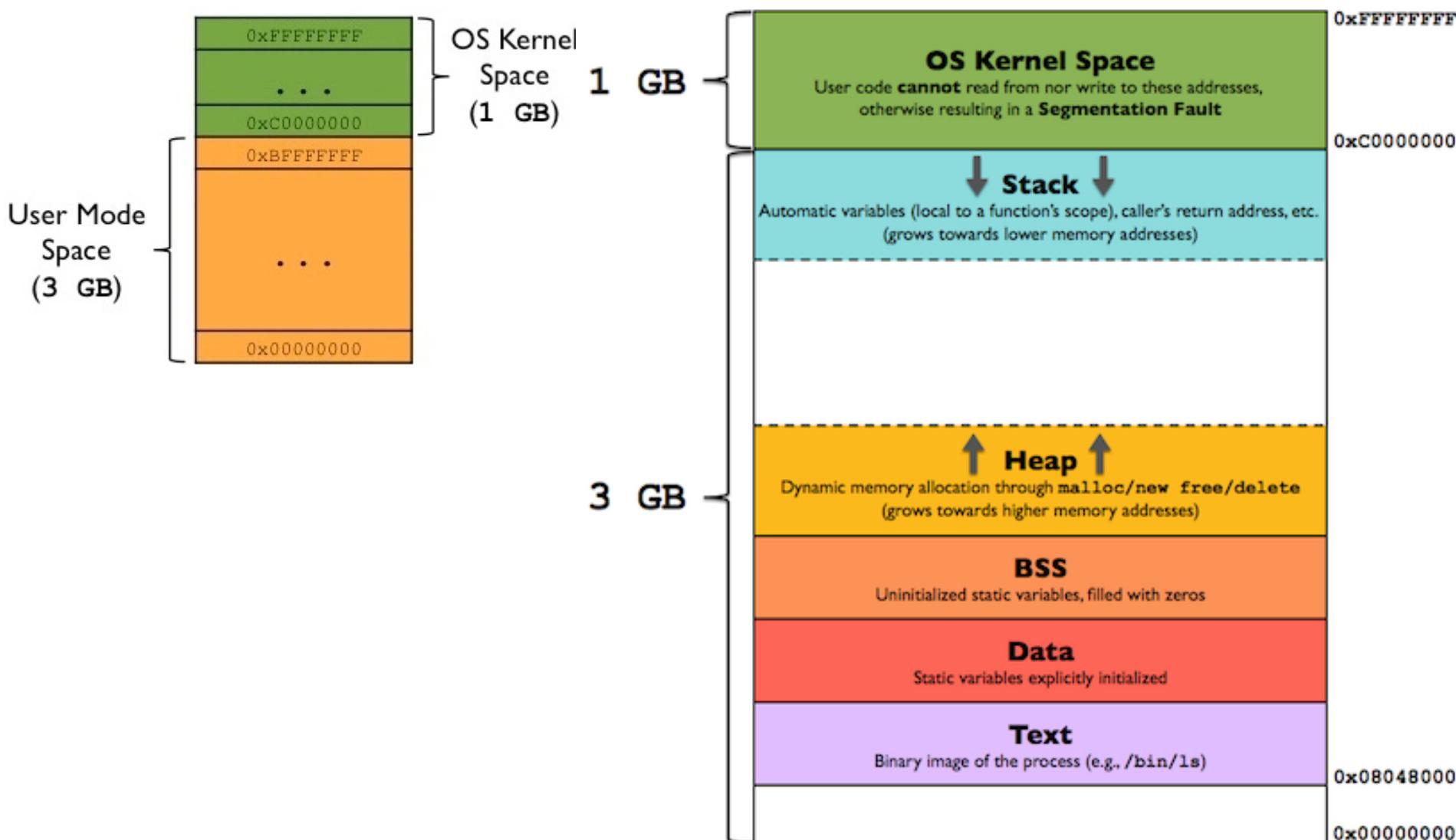
<https://www.kernel.bz/>

bits	address-0	address+1	size	digits		
64 2^{64}	0xFFFF_FFFF_FFFF_FFFF		16E	1.8x10 ¹⁹	1800경	Zeta Yotta
2 ⁶³	0x7FFF_FFFF_FFFF_FFFF	0x8000_0000_0000_0000	8E			
2 ⁶⁰	0x0FFF_FFFF_FFFF_FFFF	0x1000_0000_0000_0000	1E	10 ¹⁸	백경	Exa
2 ⁵⁰	0x0003_FFFF_FFFF_FFFF	0x0004_0000_0000_0000	1P	10 ¹⁵	천조	Peta
48 2^{48}	0x0000_FFFF_FFFF_FFFF	0x0001_0000_0000_0000	256T			
2 ⁴⁷	0x0000_7FFF_FFFF_FFFF	0x0000_8000_0000_0000	128T	x128K		
2 ⁴⁶	0x0000_3FFF_FFFF_FFFF	0x0000_4000_0000_0000	64T			
2 ⁴⁵	0x0000_1FFF_FFFF_FFFF	0x0000_2000_0000_0000	32T			
2 ⁴⁴	0x0000_0FFF_FFFF_FFFF	0x0000_1000_0000_0000	16T	x1M		
2 ⁴⁰	0x0000_00FF_FFFF_FFFF	0x0000_0100_0000_0000	1T	10 ¹²	일조	Tera
32 2^{32}	0x0000_0000_FFFF_FFFF	0x0000_0001_0000_0000	4G			
2 ³¹	0x0000_0000_7FFF_FFFF	0x0000_0000_8000_0000	2G			
2 ³⁰	0x0000_0000_3FFF_FFFF	0x0000_0000_4000_0000	1G	10 ⁹	십억	Giga
2 ²⁹	0x0000_0000_1FFF_FFFF	0x0000_0000_2000_0000	512M			
2 ²⁸	0x0000_0000_0FFF_FFFF	0x0000_0000_1000_0000	256M			
2 ²⁰	0x0000_0000_000F_FFFF	0x0000_0000_0010_0000	1M	10 ⁶	백만	Mega
16 2^{16}	0x0000_0000_0000_FFFF	0x0000_0000_0001_0000	64K			
2 ¹²	0x0000_0000_0000_0FFF	0x0000_0000_0000_1000	4K	page		
2 ¹⁰	0x0000_0000_0000_03FF	0x0000_0000_0000_0400	1K	10 ³	일천	
0 2^0	0x0000_0000_0000_0000	0x0000_0000_0000_0001	1			

커널 메모리 주소 관리 및 메모리맵 이해

메모리 맵 (32 비트 주소공간)

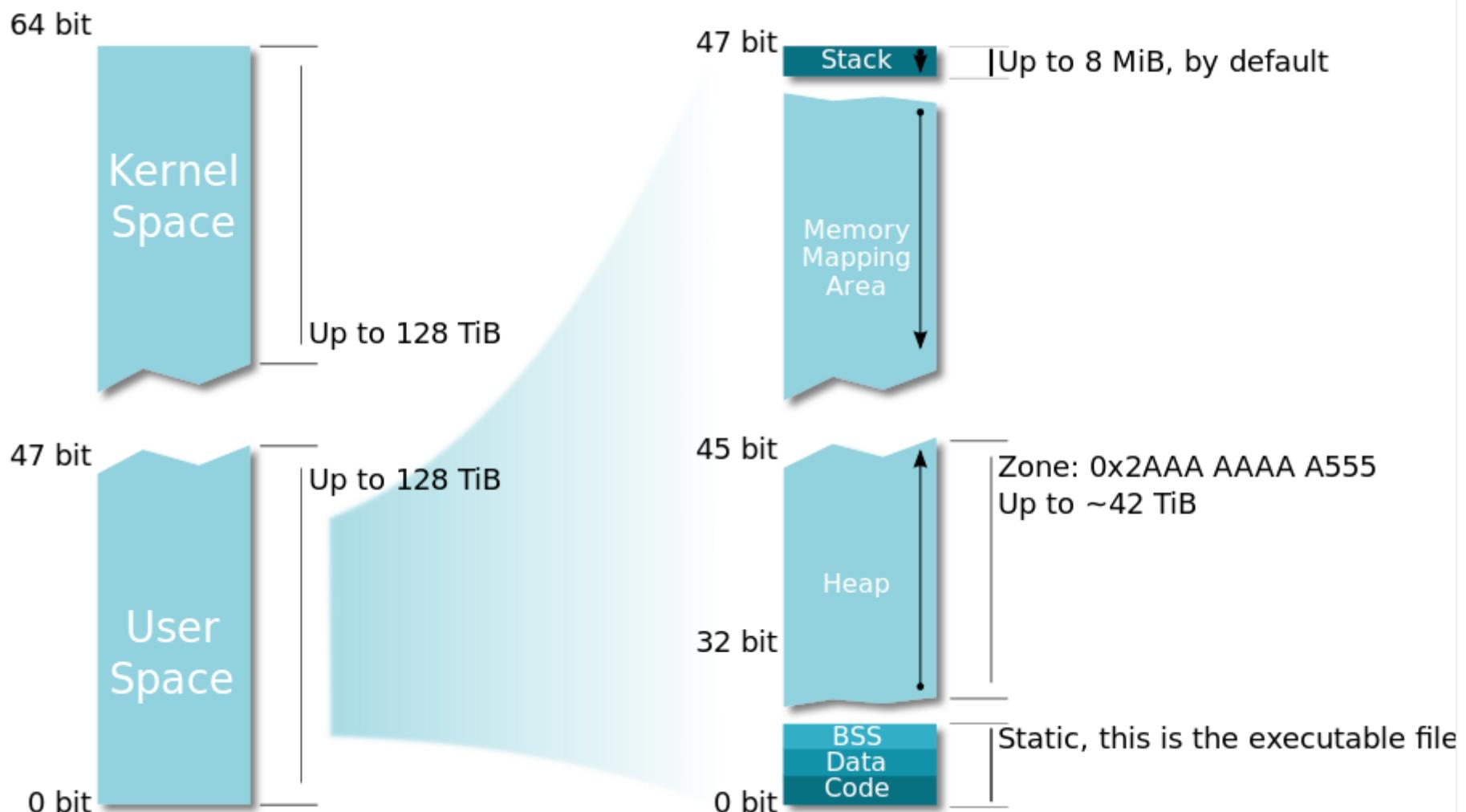
<https://gabrieletolomei.wordpress.com/miscellanea/operating-systems/in-memory-layout/>



커널 메모리 주소 관리 및 메모리맵 이해

메모리 맵 (64 비트 주소공간)

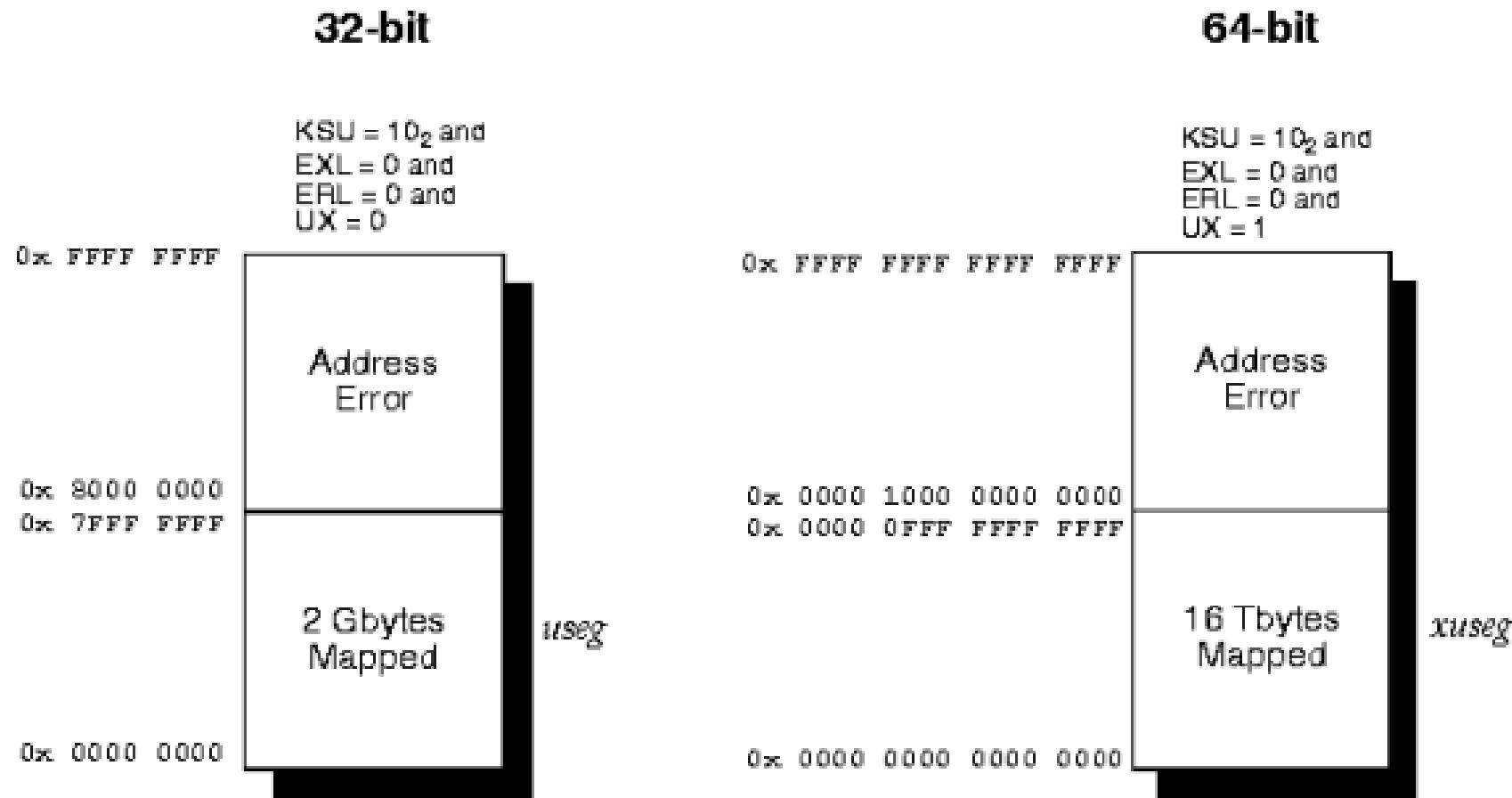
https://en.wikibooks.org/wiki/The_Linux_Kernel/Memory



커널 메모리 주소 관리 및 메모리맵 이해

메모리 맵 (32 비트 vs 64 비트)

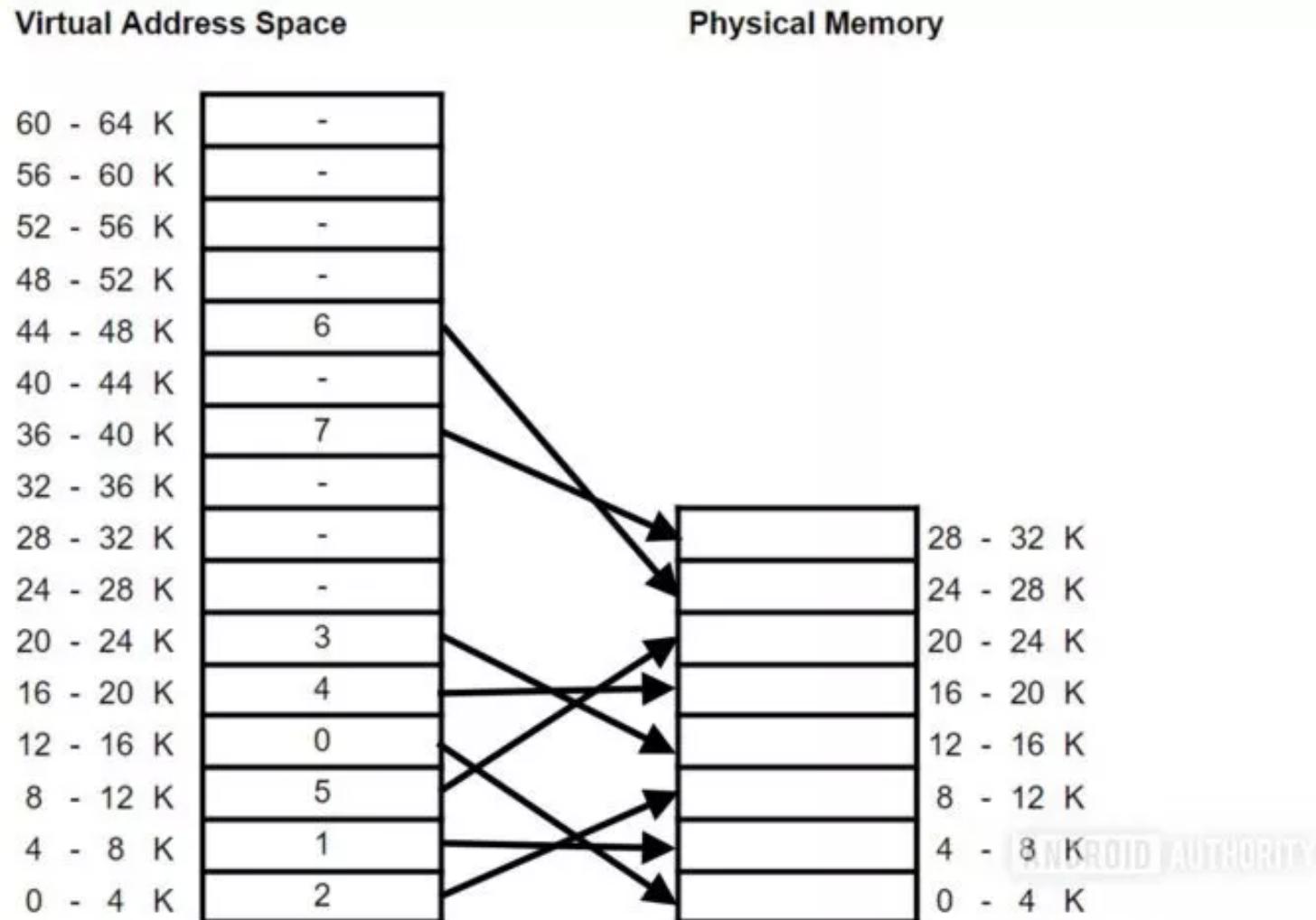
https://techpubs.jurassic.nl/manuals/hdwr/developer/R10K_UM/sgi_html/t5.Ver.2.0.book_330.html



커널 메모리 관리 및 메모리맵 이해

커널 메모리 주소 관리 (가상 논리 주소, 물리주소)

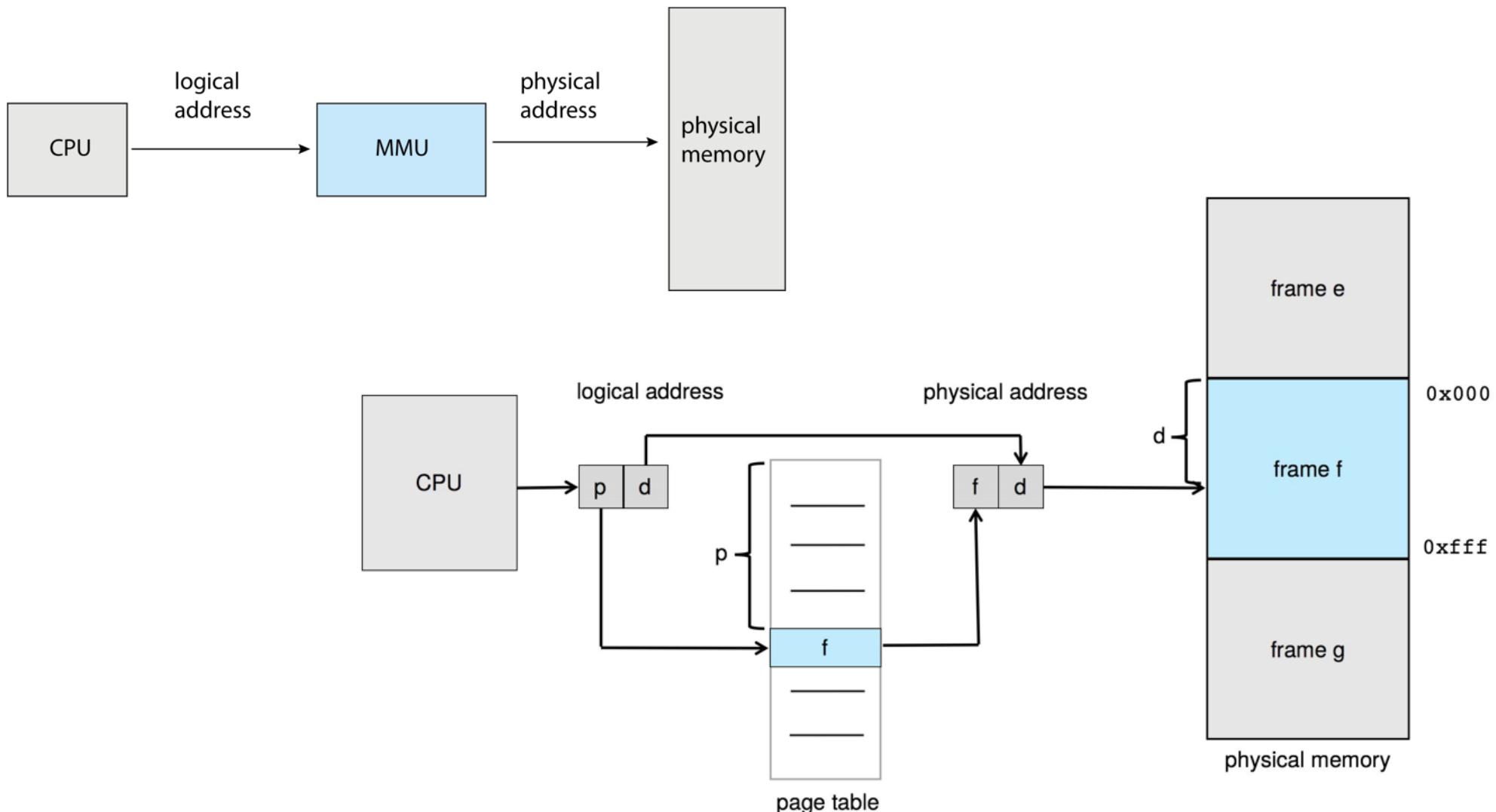
<https://www.androidauthority.com/virtual-memory-explained-3143201/>



커널 메모리 관리 및 메모리맵 이해

커널 메모리 주소 관리 (가상 논리 주소, 물리주소)

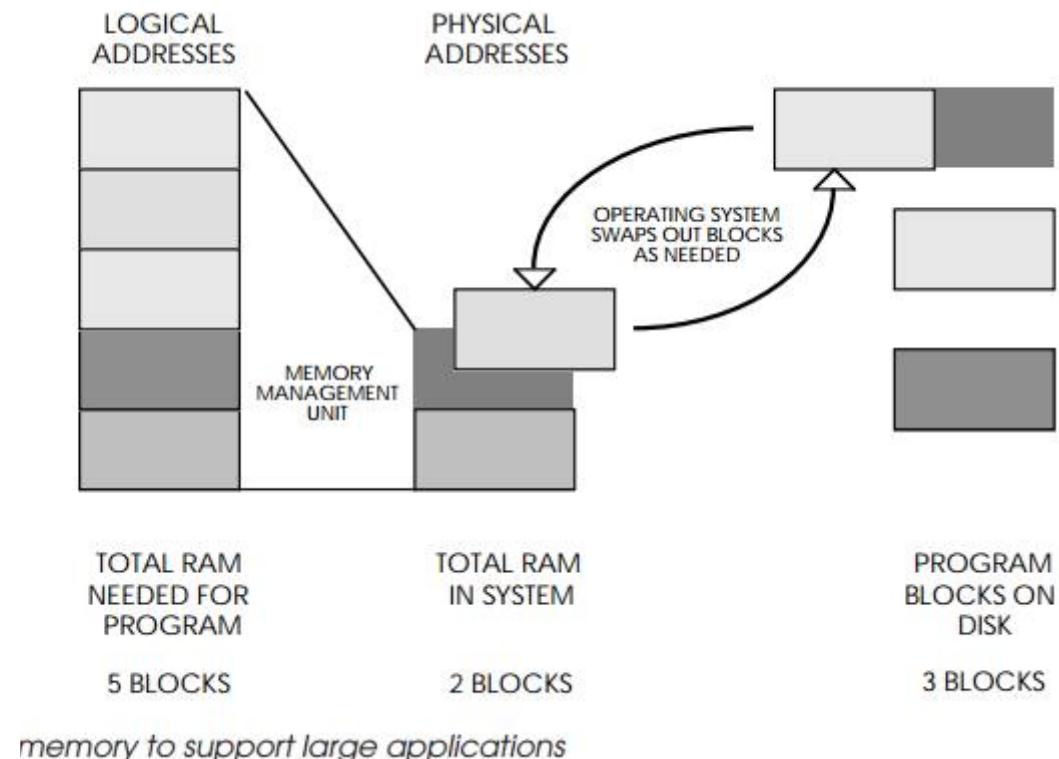
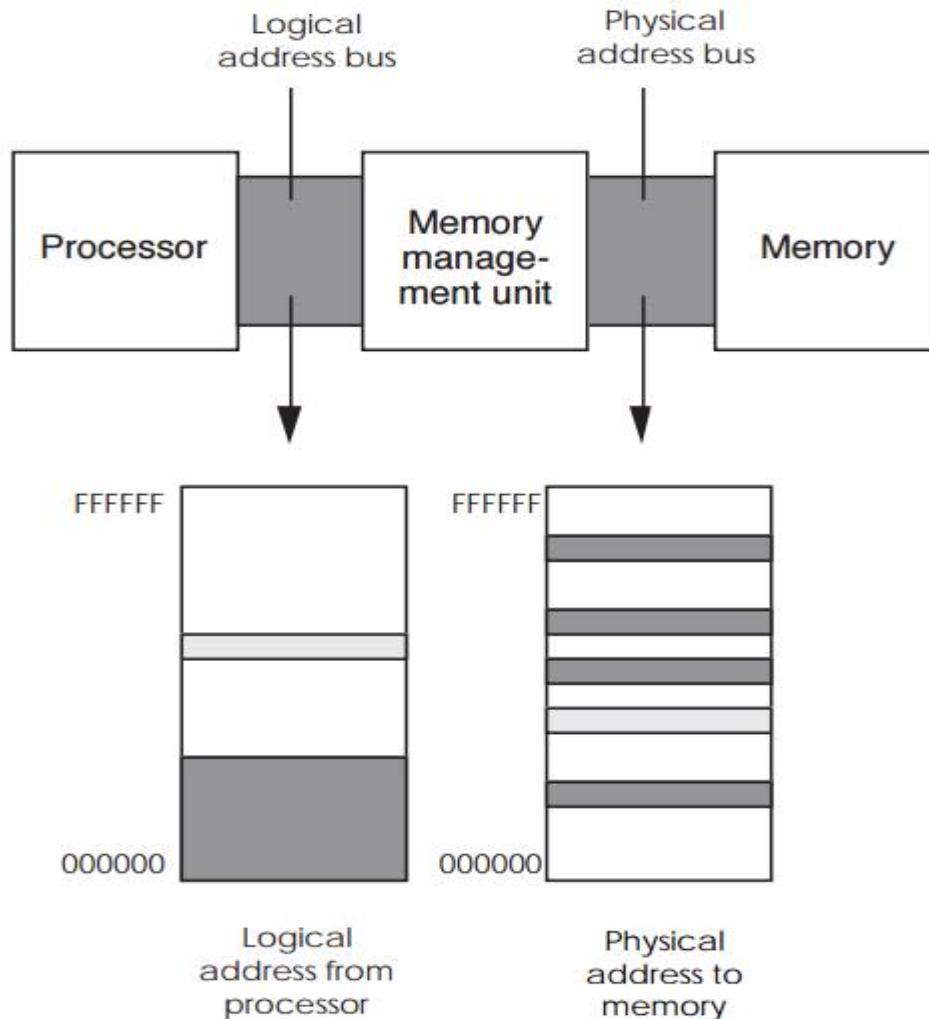
<https://velog.io/@dahyeon/Chapter-9-Memory-Management>



커널 메모리 관리 및 메모리맵 이해

커널 메모리 주소 관리 (가상 논리주소, 물리주소)

https://www.brainkart.com/article/Memory-management-address-translation_7703/



Memory management principles

커널 메모리 관리 및 메모리맵 이해

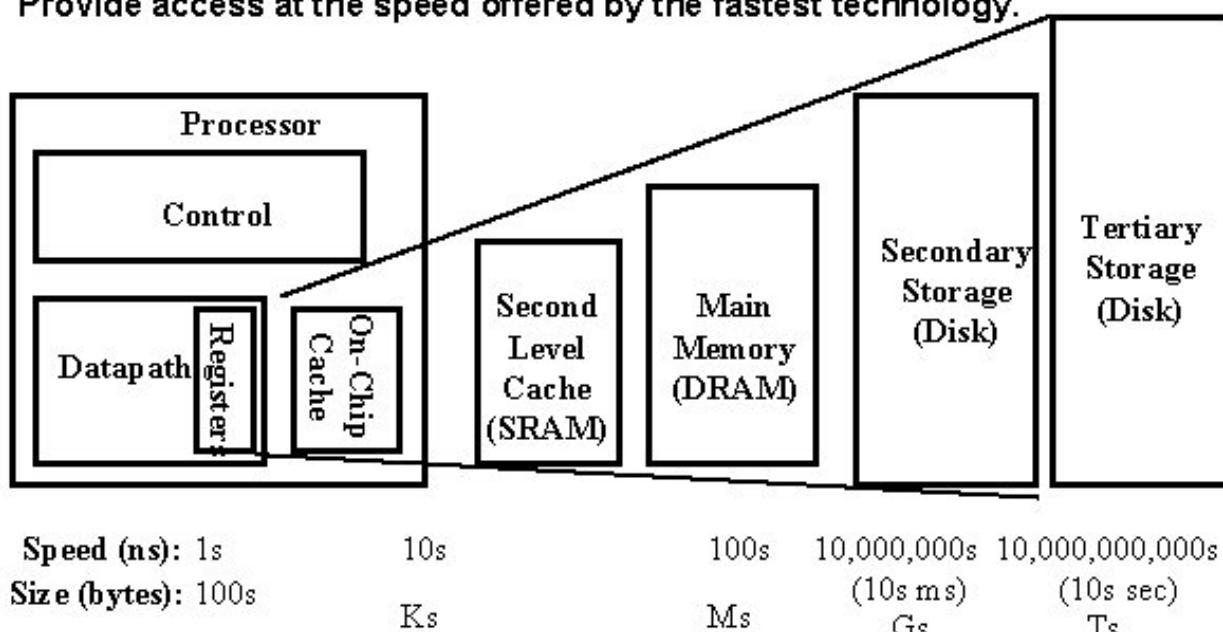
메모리 주소 관리 (메모리 접근 속도 , 캐시메모리)

<https://slidetodoc.com/memory-hierarchy-reasons-virtual-memory-cache-memory-translation/>

Memory Hierarchy of a Modern Computer

- By taking advantage of the principle of locality:

- Present the user with as much memory as is available in the cheapest technology.
- Provide access at the speed offered by the fastest technology.

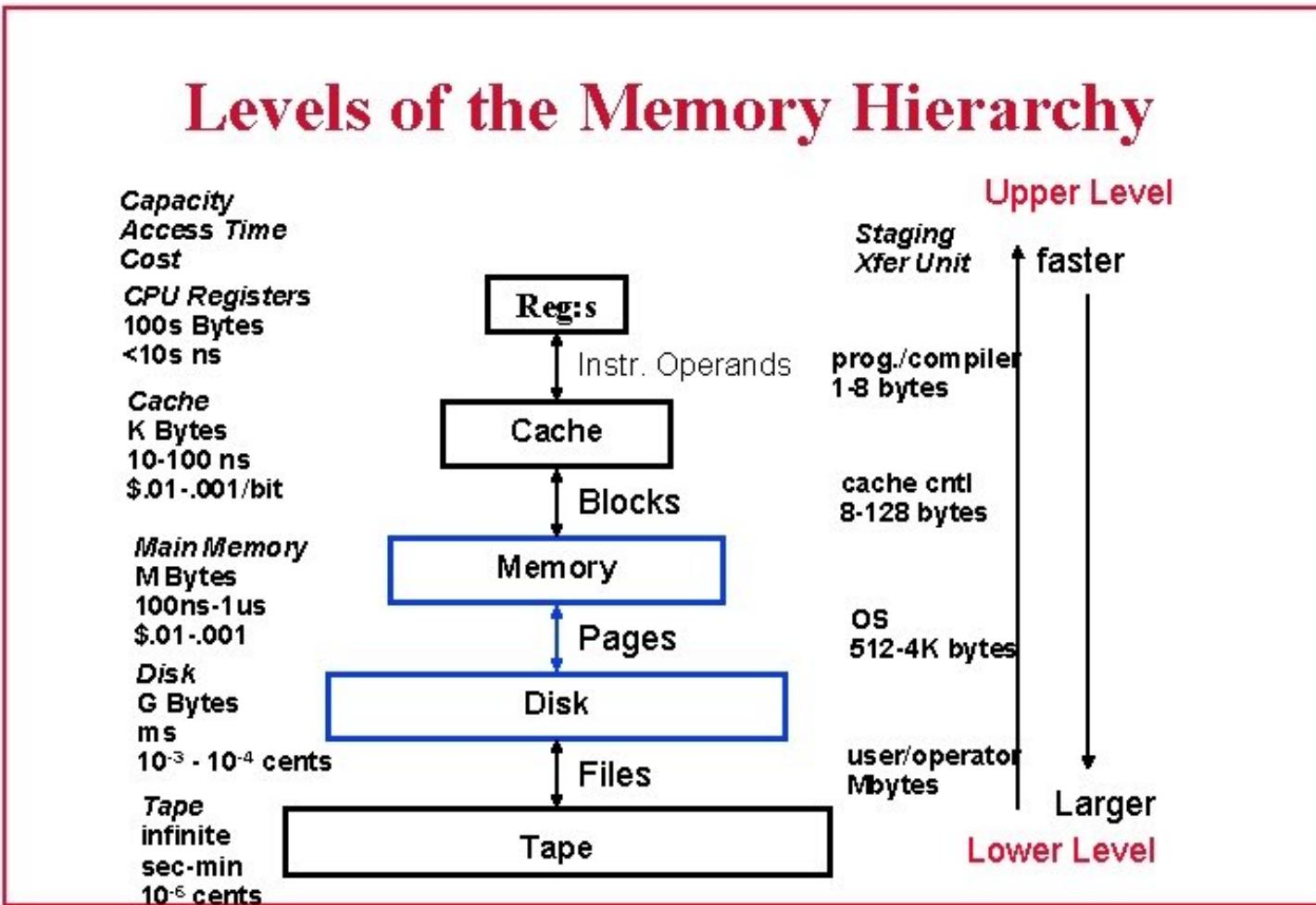


커널 메모리 관리 및 메모리맵 이해

메모리 주소 관리 (메모리 접근 속도 , 캐시메모리)

<https://slidetodoc.com/memory-hierarchy-reasons-virtual-memory-cache-memory-translation/>

Levels of the Memory Hierarchy



DataTeknik Memory Acceleration bild 6

커널 메모리 관리 및 메모리맵 이해

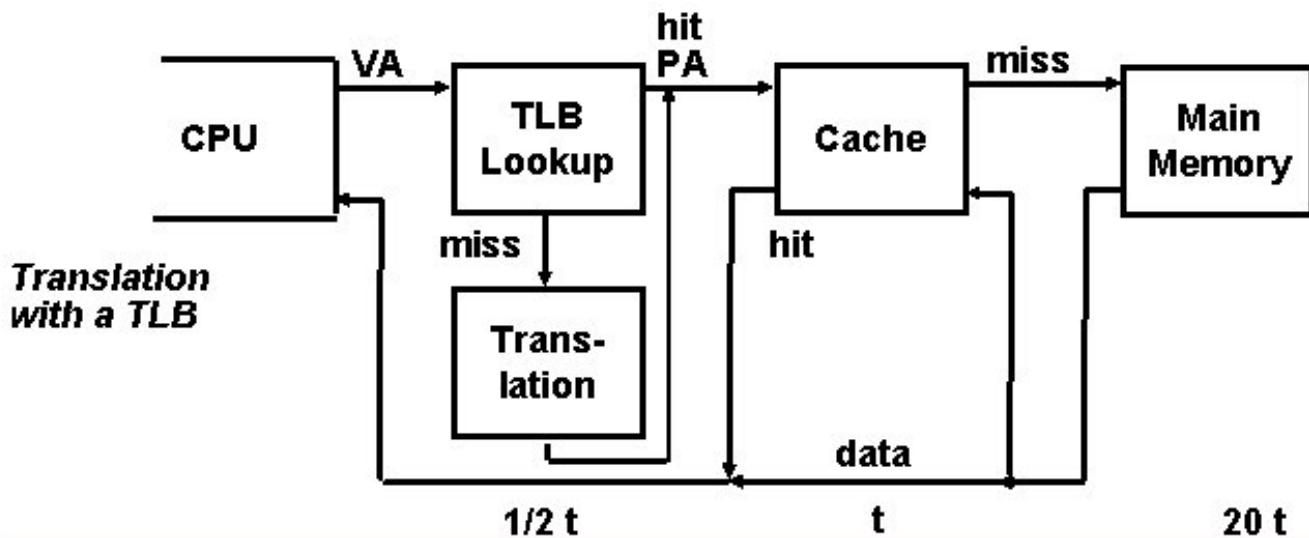
메모리 주소 관리 (메모리 접근 속도 , 캐시메모리)

<https://slidetodoc.com/memory-hierarchy-reasons-virtual-memory-cache-memory-translation/>

Translation Look-Aside Buffers

Just like any other cache, the TLB can be organized as **fully associative, set associative, or direct mapped**

TLBs are **usually small, typically not more than 128 - 256 entries** even on high end machines. This permits fully associative lookup on these machines. Most mid-range machines use small n-way set associative organizations.

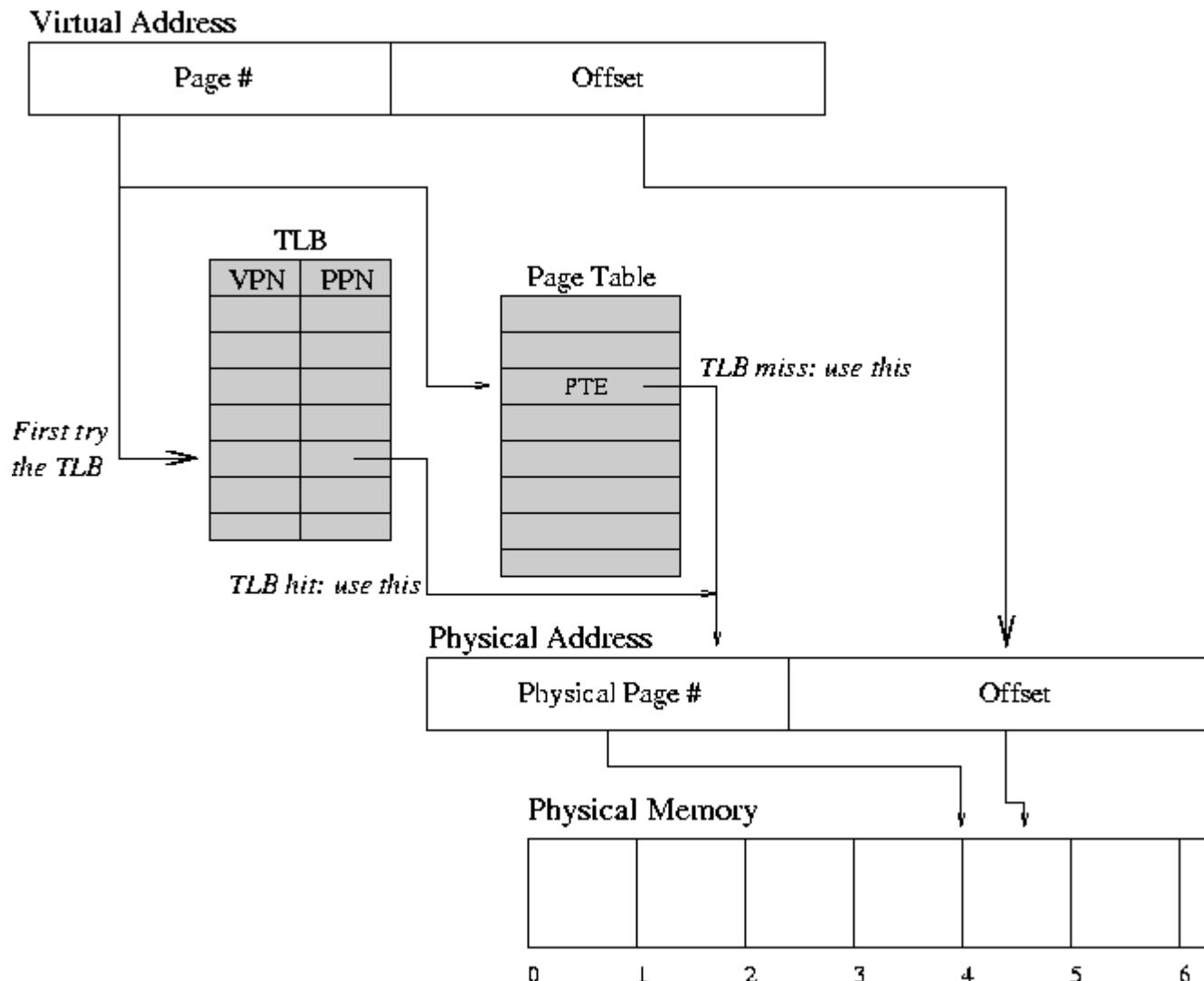


Datoteknik Memory Acceleration bild 37

커널 메모리 관리 및 메모리맵 이해

메모리 주소 관리 (요약 정리)

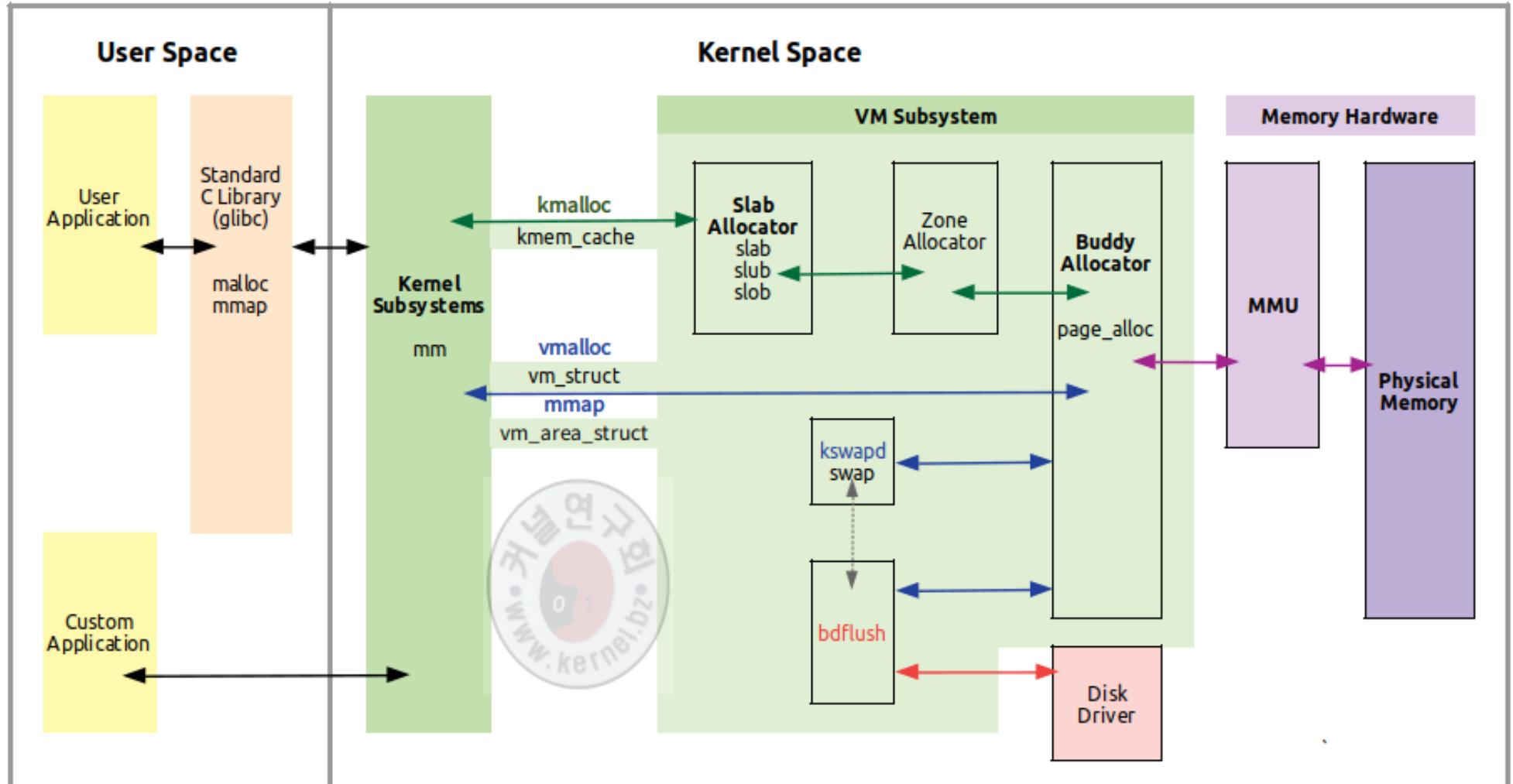
<https://pages.cs.wisc.edu/~bart/537/lecturenotes/s17.html>



커널 메모리 관리 및 메모리맵 이해

리눅스 커널 메모리 주소 관리 (요약 정리)

<https://www.kernel.bz/>



임베디드 리눅스 커널 디바이스 드라이버 공유자원 동기화 및 Locking 이해

임베디드 리눅스 커널 디바이스 드라이버 공유자원 동기화 및 Locking 이해

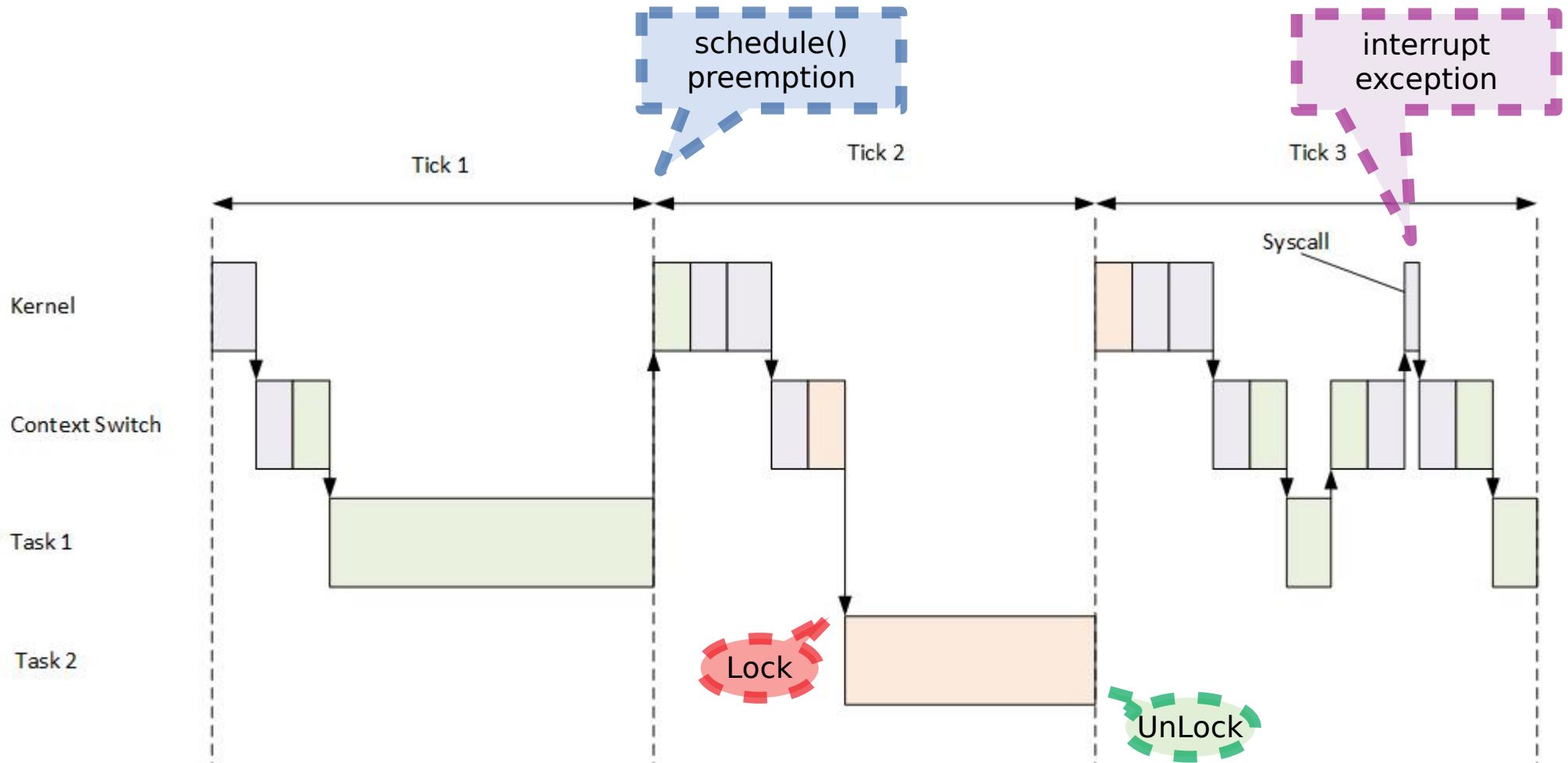
정재준 <rgbi3307@naver.com>

커널연구회 <www.kernel.bz>

공유자원 동기화 및 Locking 이해

공유자원 동기화 및 Locking 개념 이해 1

<https://pages.cs.wisc.edu/~bart/537/lecturenotes/s7.html>



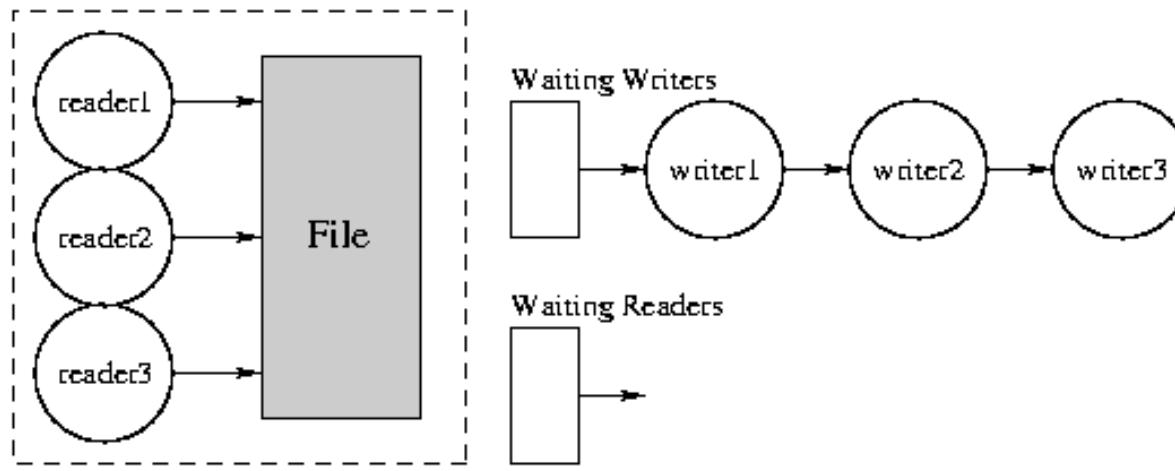
그림출처 : <https://stackoverflow.com/questions/42393253/micro-scheduler-for-real-time-kernel-in-embedded-c-applications>

공유자원 동기화 및 Locking 이해

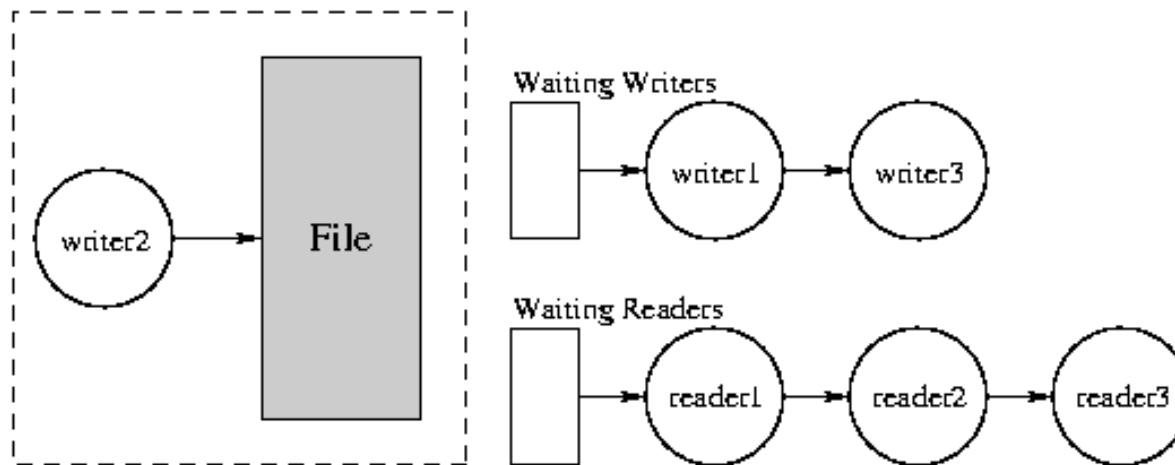
공유자원 동기화 및 Locking 개념 이해 2

<https://pages.cs.wisc.edu/~bart/537/lecturenotes/s7.html>

Multiple Readers



One Writer



공유자원 동기화 및 Locking 이해

리눅스 커널 Locking 종류

<https://www.kernel.bz>

CPU local locks

```
local_lock_t lock;  
  
local_lock(&lock);  
  
/* ... */  
  
local_unlock(&lock);
```

Spinning locks

```
typedef struct spinlock {  
    union {  
        struct raw_spinlock rlock;  
    };  
} spinlock_t;  
  
DEFINE_SPINLOCK(my_lock);  
unsigned long flags;  
  
spin_lock_irqsave(&my_lock, flags);  
  
/* ... */  
  
spin_unlock_irqrestore(&my_lock, flags);
```

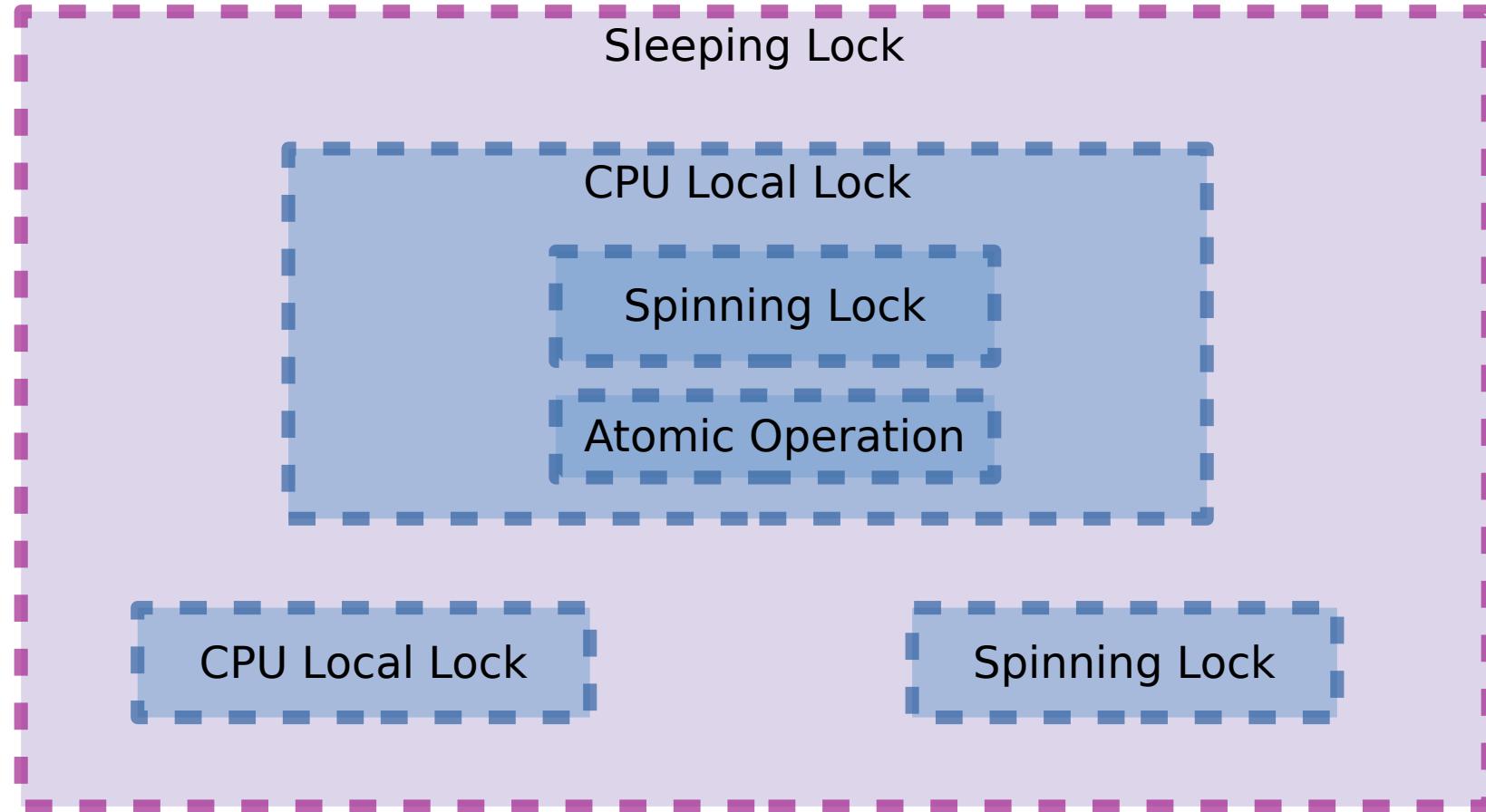
Sleeping locks

```
struct mutex {  
    atomic_long_t owner;  
    raw_spinlock_t wait_lock;  
    struct list_head wait_list;  
};  
DEFINE_MUTEX(my_lock);  
mutex_lock(&my_lock);  
/* ... */  
mutex_unlock(&my_lock);  
  
struct semaphore {  
    raw_spinlock_t lock;  
    unsigned int count;  
    struct list_head wait_list;  
};  
DEFINE_SEMAPHORE(name);  
down(struct semaphore *sem);  
/* ... */  
up(struct semaphore *sem);
```

공유자원 동기화 및 Locking 이해

리눅스 커널 Locking 포함 (nesting) 규칙

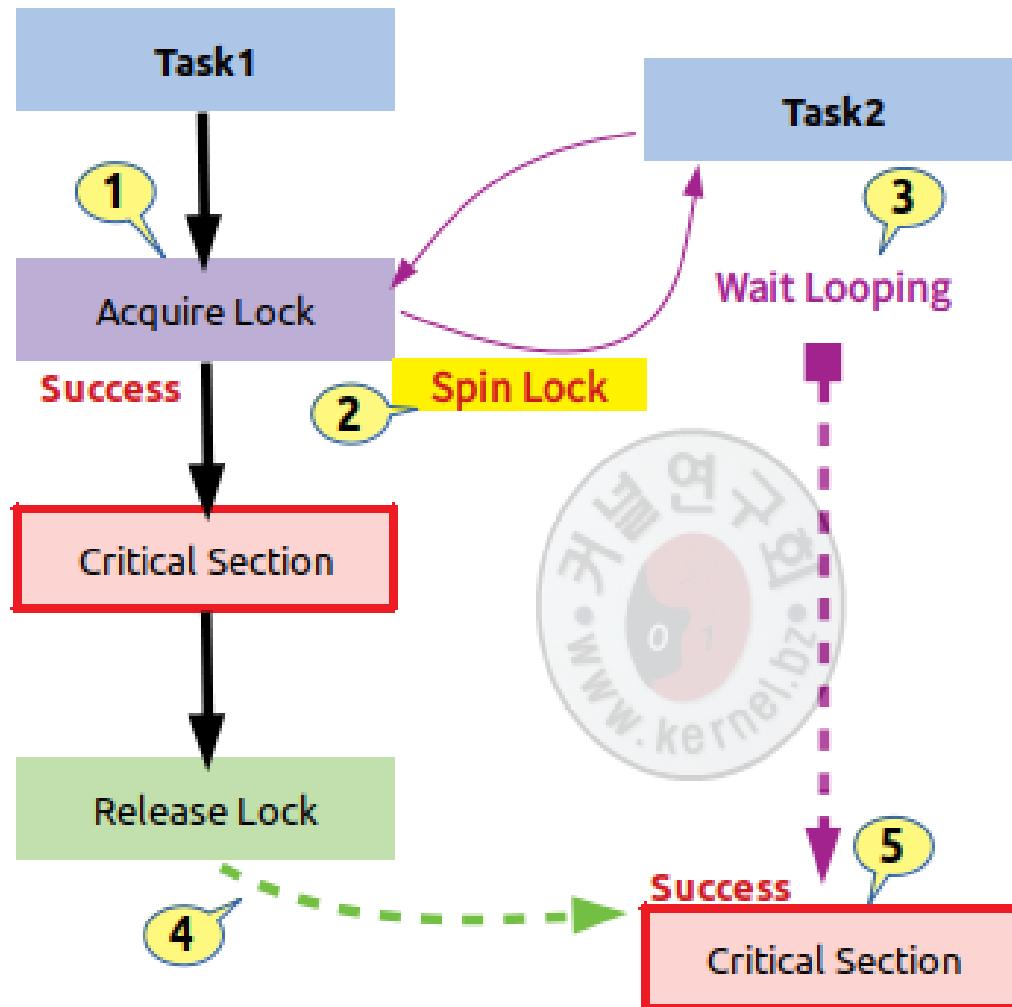
<https://www.kernel.bz>



공유자원 동기화 및 Locking 이해

리눅스 커널 Locking 설명 (SpinLock 개념 이해)

<https://www.kernel.bz/>

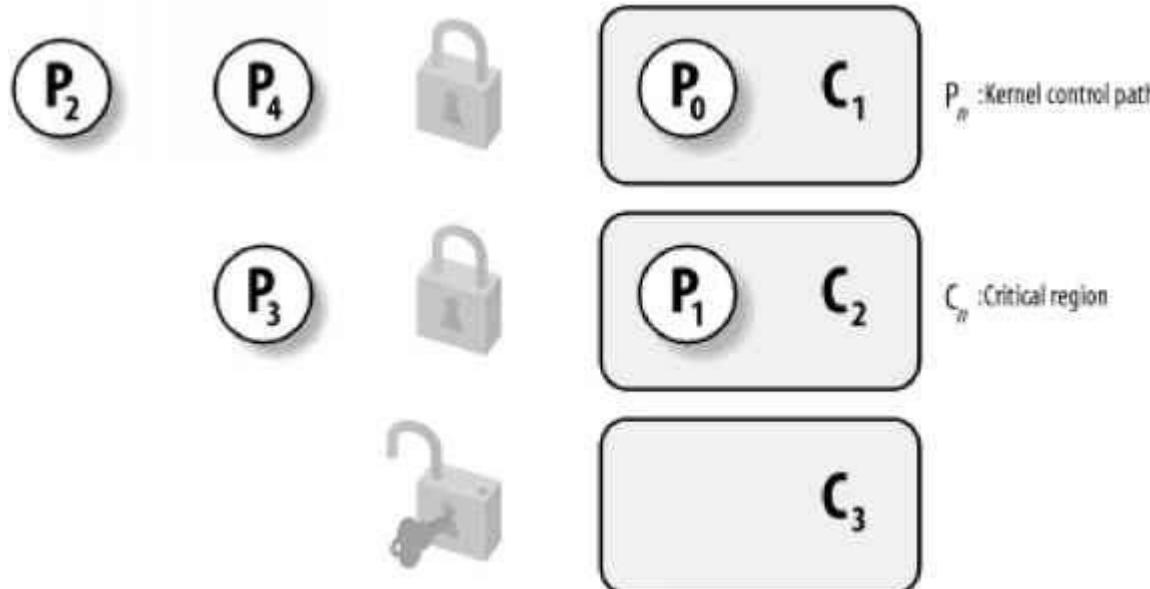


공유자원 동기화 및 Locking 이해

리눅스 커널 Locking 설명 (SpinLock API 함수)

<https://www.kernel.org/doc/html/latest/locking/spinlocks.html>

```
static DEFINE_SPINLOCK(xxx_lock);  
unsigned long flags;  
  
spin_lock_irqsave(&xxx_lock, flags);  
/* ... critical section here ... */  
spin_unlock_irqrestore(&xxx_lock, flags);
```



Local 인터럽트를 불가능
(disable) 으로 설정 .

UP 나 SMP 시스템 모두에서
안전하게 동작 .

공유자원 동기화 및 Locking 이해

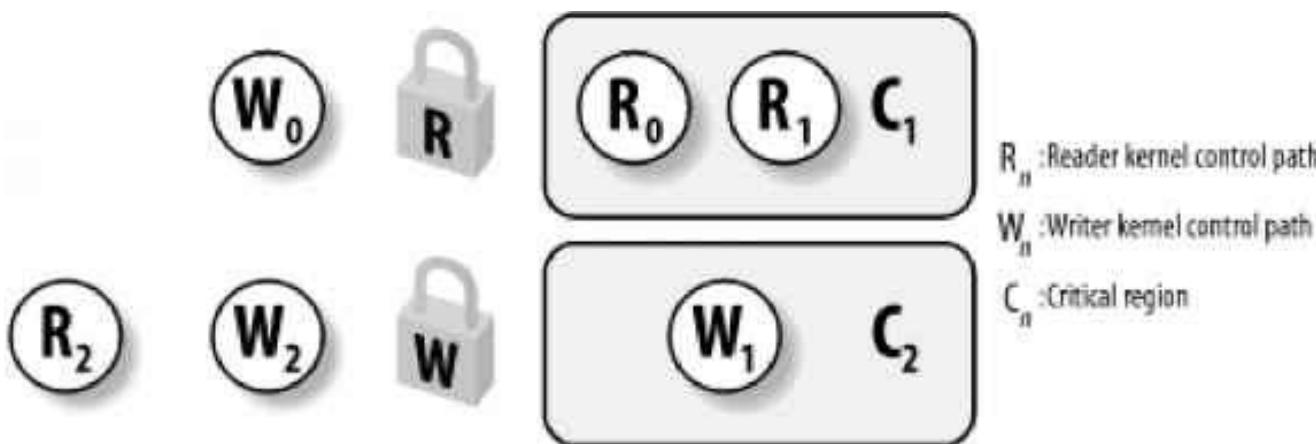
리눅스 커널 Locking 설명 (R/W SpinLock API 함수)

<https://www.kernel.org/doc/html/latest/locking/spinlocks.html>

```
rwlock_t xxx_lock = __RW_LOCK_UNLOCKED(xxx_lock);  
unsigned long flags;
```

```
read_lock_irqsave(&xxx_lock, flags);  
/* .. critical section that only reads the info ... */  
read_unlock_irqrestore(&xxx_lock, flags);
```

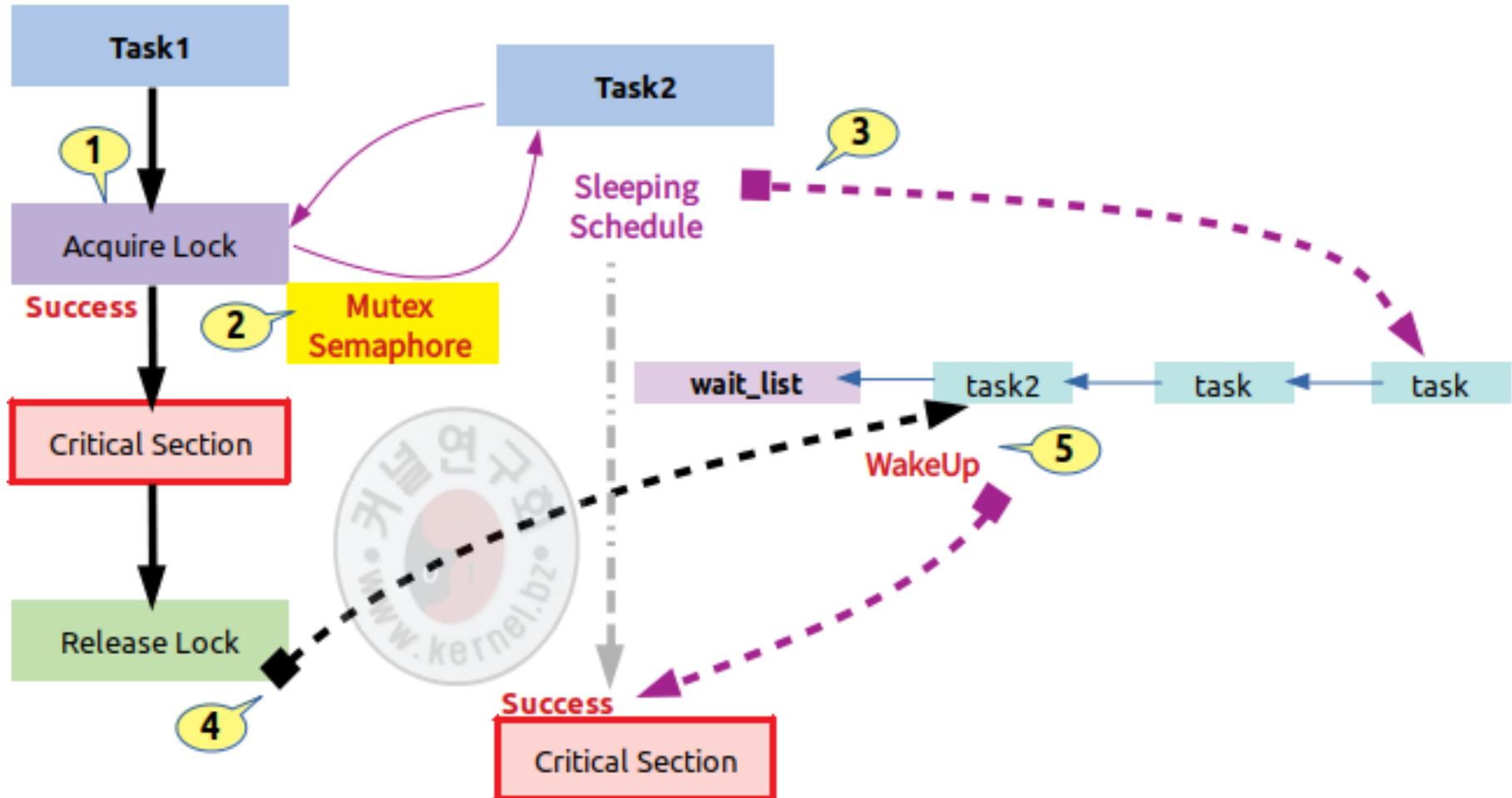
```
write_lock_irqsave(&xxx_lock, flags);  
/* .. read and write exclusive access to the info ... */  
write_unlock_irqrestore(&xxx_lock, flags);
```



공유자원 동기화 및 Locking 이해

리눅스 커널 Locking 설명 (Mutex Lock 개념 이해)

<https://www.kernel.bz>



공유자원 동기화 및 Locking 이해

리눅스 커널 Locking 설명 (Mutex Lock API 함수)

include/linux/mutex.h, kernel/locking/mutex.c

```
struct mutex {  
    atomic_long_t owner;  
    raw_spinlock_t wait_lock;  
    struct list_head wait_list;  
};  
  
#define DEFINE_MUTEX(mutexname) \  
    struct mutex mutexname = __MUTEX_INITIALIZER(mutexname)
```

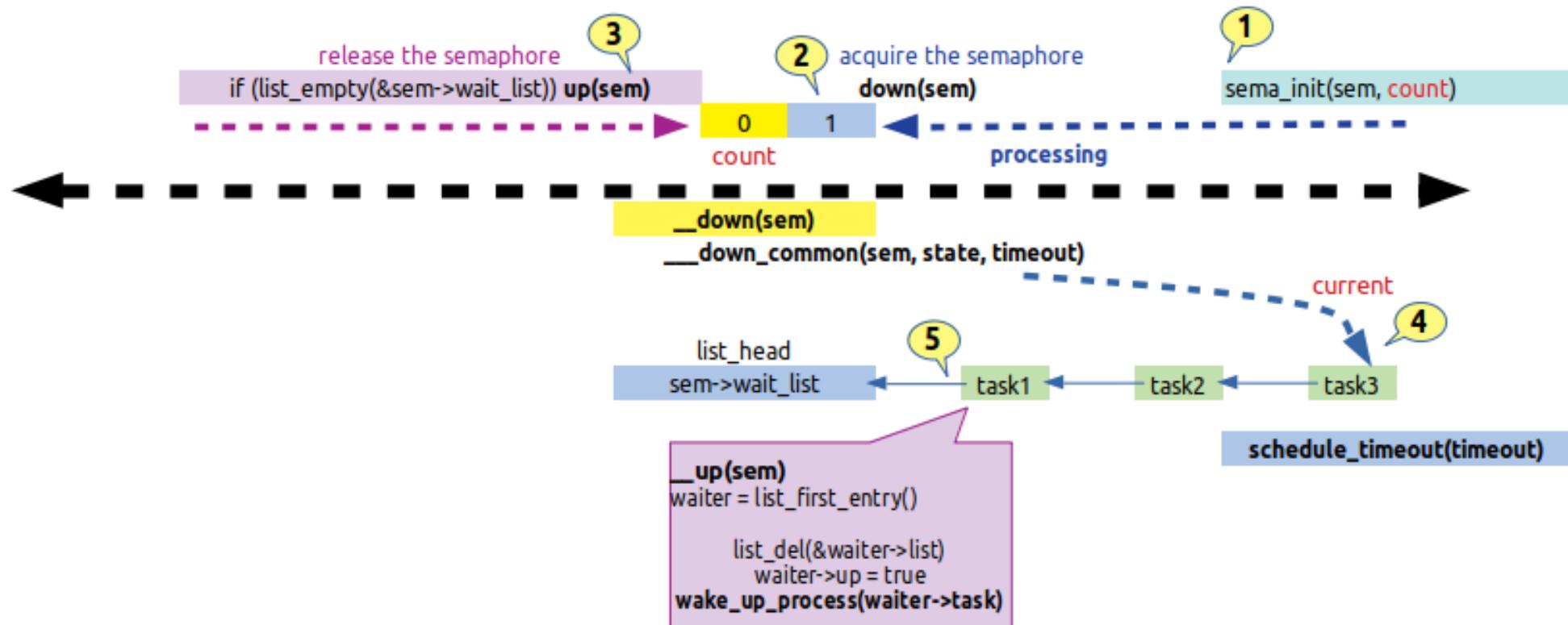
```
void __sched mutex_lock(struct mutex *lock)  
{  
    might_sleep();  
  
    if (!__mutex_trylock_fast(lock))  
        __mutex_lock_slowpath(lock);  
}
```

```
void __sched mutex_unlock(struct mutex *lock)  
{  
    #ifndef CONFIG_DEBUG_LOCK_ALLOC  
        if (__mutex_unlock_fast(lock))  
            return;  
    #endif  
    __mutex_unlock_slowpath(lock, _RET_IP_);  
}
```

공유자원 동기화 및 Locking 이해

리눅스 커널 Locking 설명 (SemaPhore 개념 이해)

include/linux/semaphore.h, kernel/locking/semaphore.c



공유자원 동기화 및 Locking 이해

리눅스 커널 Locking 설명 (Semaphore API 함수)

include/linux/semaphore.h, kernel/locking/semaphore.c

```
struct semaphore {  
    raw_spinlock_t      lock;  
    unsigned int        count;  
    struct list_head   wait_list;  
};  
#define DEFINE_SEMAPHORE(name) \  
    struct semaphore name = _SEMAPHORE_INITIALIZER(name, 1)
```

```
sema_init(struct semaphore *sem, int val);
```

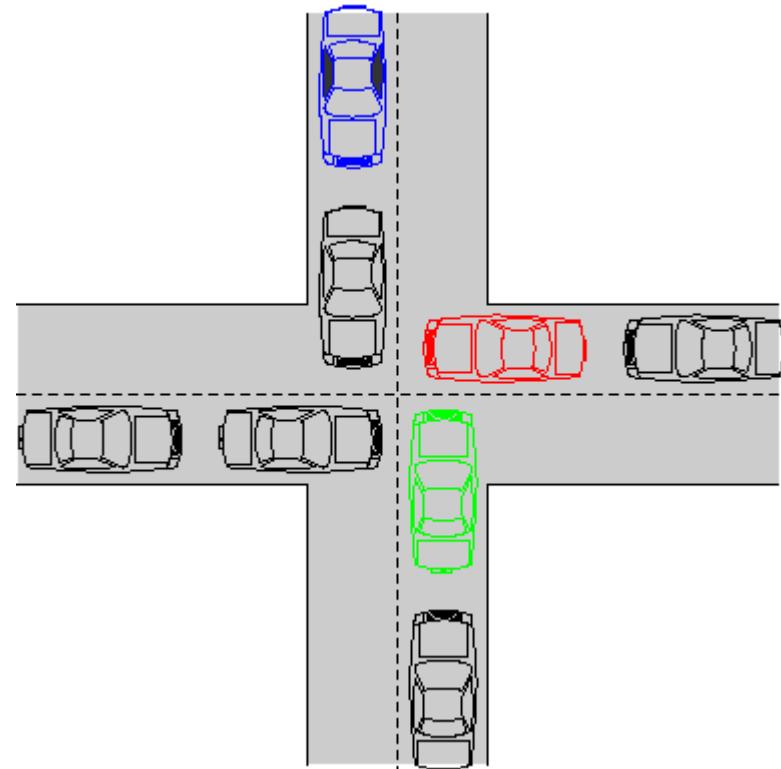
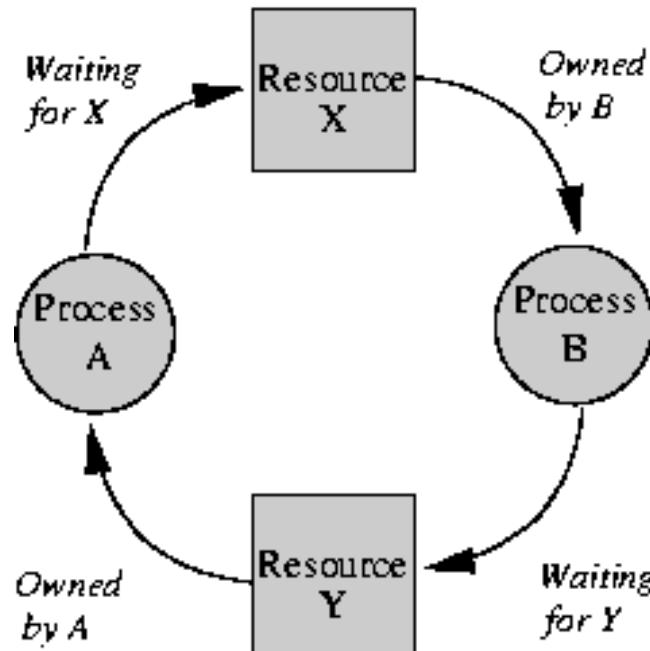
```
void __sched down(struct semaphore *sem)  
{  
    unsigned long flags;  
  
    might_sleep();  
    raw_spin_lock_irqsave(&sem->lock, flags);  
    if (likely(sem->count > 0))  
        sem->count--;  
    else  
        __down(sem);  
    raw_spin_unlock_irqrestore(&sem->lock, flags);  
}
```

```
void __sched up(struct semaphore *sem)  
{  
    unsigned long flags;  
  
    raw_spin_lock_irqsave(&sem->lock, flags);  
    if (likely(list_empty(&sem->wait_list)))  
        sem->count++;  
    else  
        __up(sem);  
    raw_spin_unlock_irqrestore(&sem->lock, flags);  
}
```

공유자원 동기화 및 Locking 이해

공유자원 동기화 및 DeadLock 개념 이해

<https://pages.cs.wisc.edu/~bart/537/lecturenotes/s12.html>

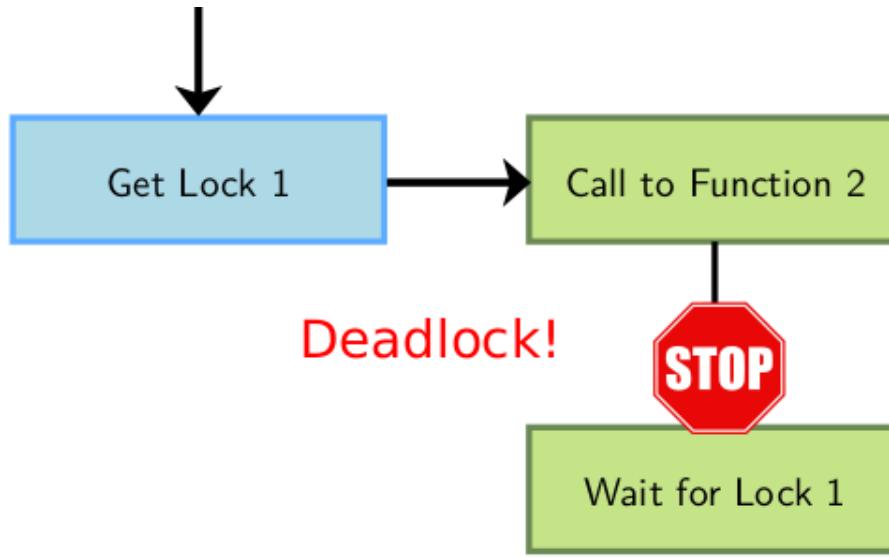


공유자원 동기화 및 Locking 이해

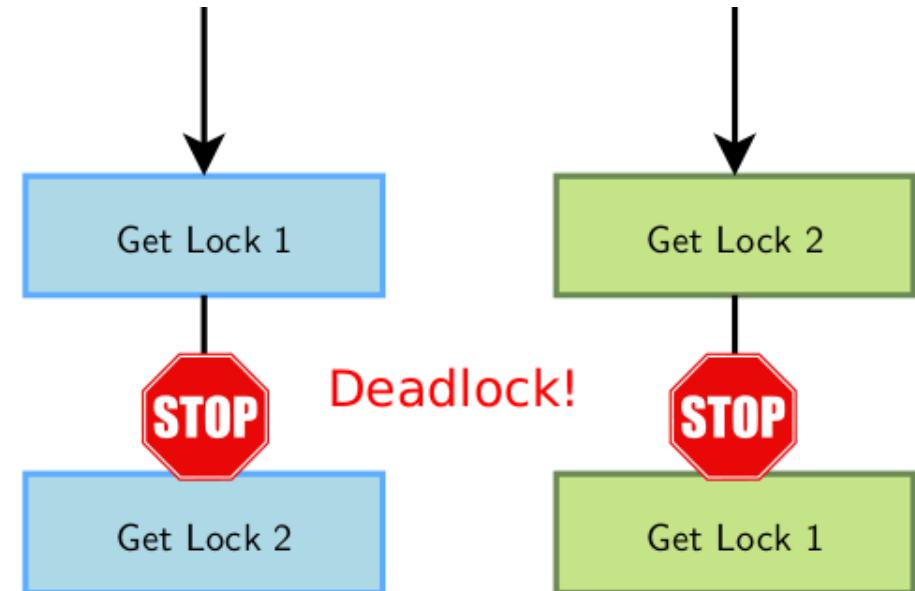
공유자원 동기화 및 DeadLock 발생 예시

<https://www.kernel.bz/>

Locking 중복



Locking 순서 위반



임베디드 리눅스 커널 디바이스 드라이버

커널 모듈과 디바이스 드라이버 이해

임베디드 리눅스 커널 디바이스 드라이버

커널 모듈과 디바이스 드라이버 이해

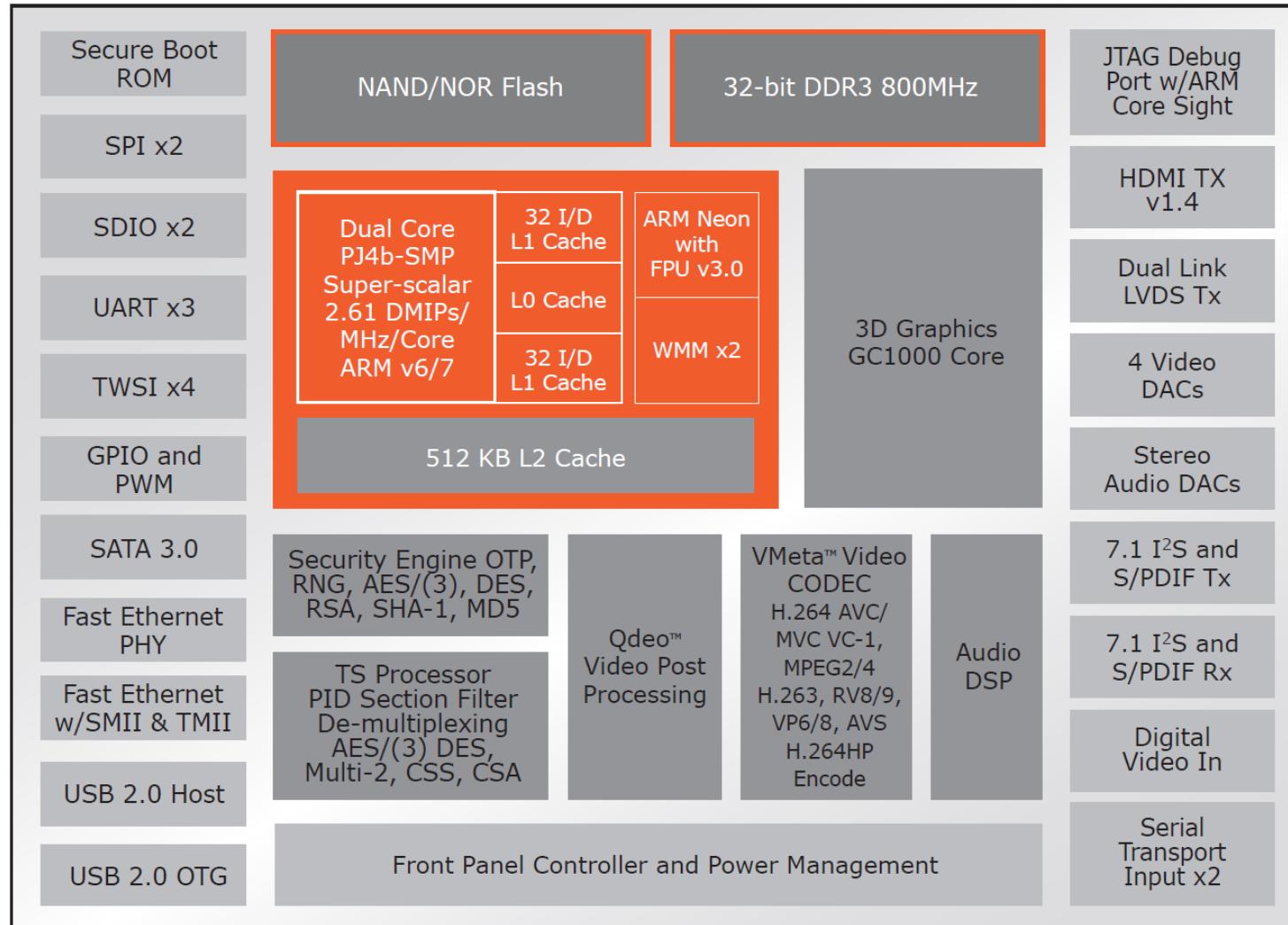
정재준 <rgb13307@naver.com>

커널연구회 <www.kernel.bz>

커널 모듈과 디바이스 드라이버 이해

Device Tree 이해 (라즈베리파이 하드웨어 블럭도)

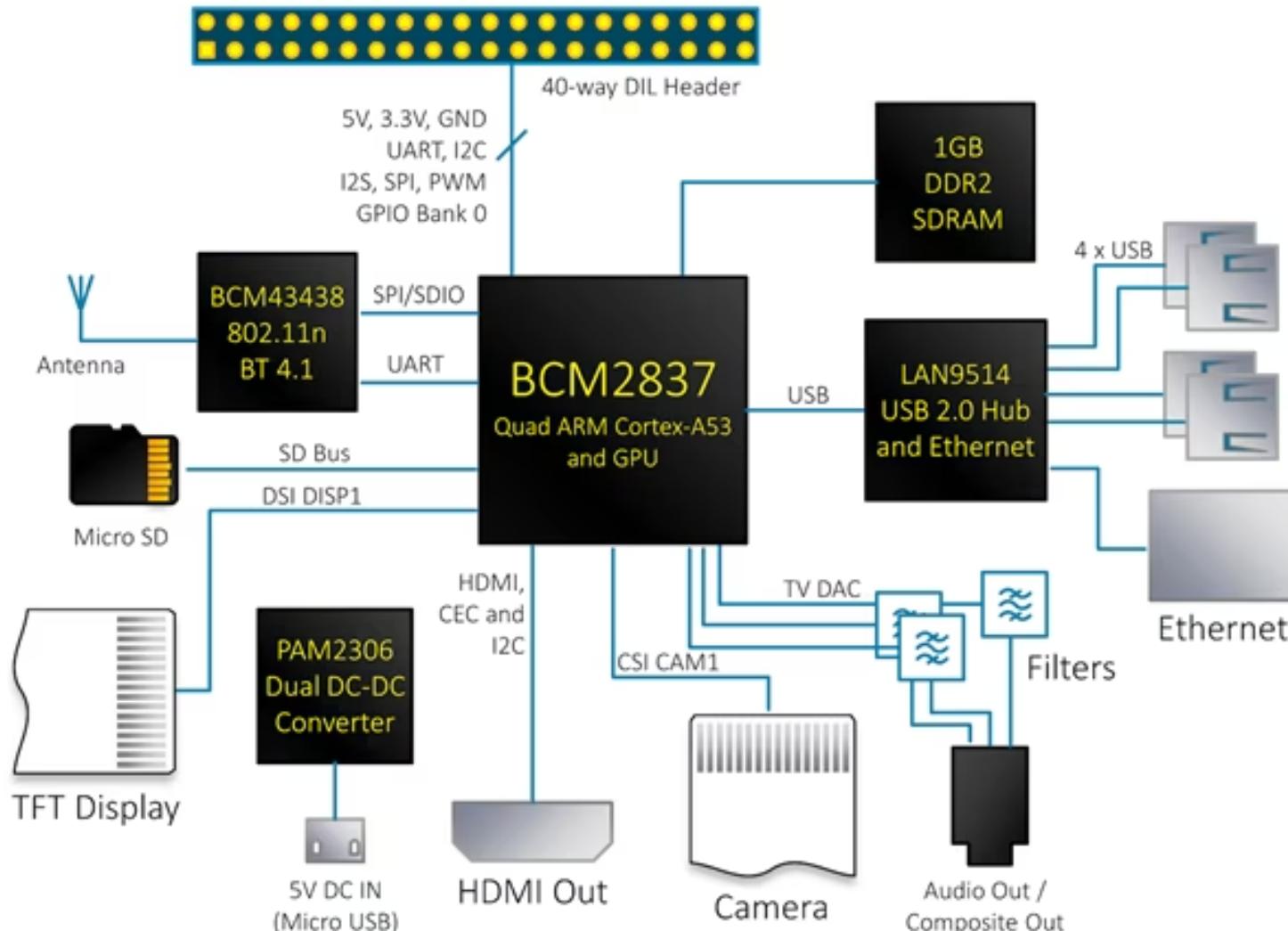
<https://forums.raspberrypi.com/viewtopic.php?t=168713>



커널 모듈과 디바이스 드라이버 이해

Device Tree 이해 (라즈베리파이 하드웨어 블럭도)

<https://community.element14.com/products/raspberry-pi/b/blog/posts/raspberry-pi-3-block-diagram>

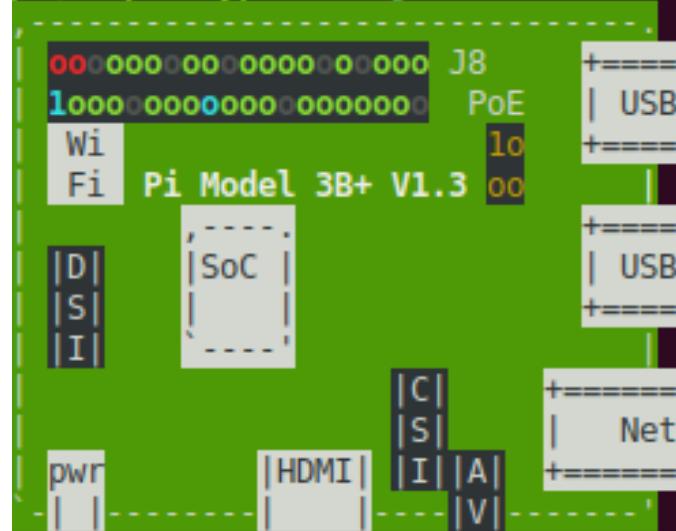


커널 모듈과 디바이스 드라이버 이해

라즈베리파이 pinout

<https://pinout.xyz/>

```
pi@raspberrypi:~/Projects/kernel/linux-v6.1 $ pinout
```



```
Revision          : a020d3
SoC              : BCM2837
RAM              : 1GB
Storage           : MicroSD
USB ports        : 4 (of which 0 USB3)
Ethernet ports   : 1 (300Mbps max. speed)
Wi-fi            : True
Bluetooth        : True
Camera ports (CSI) : 1
Display ports (DSI): 1
```

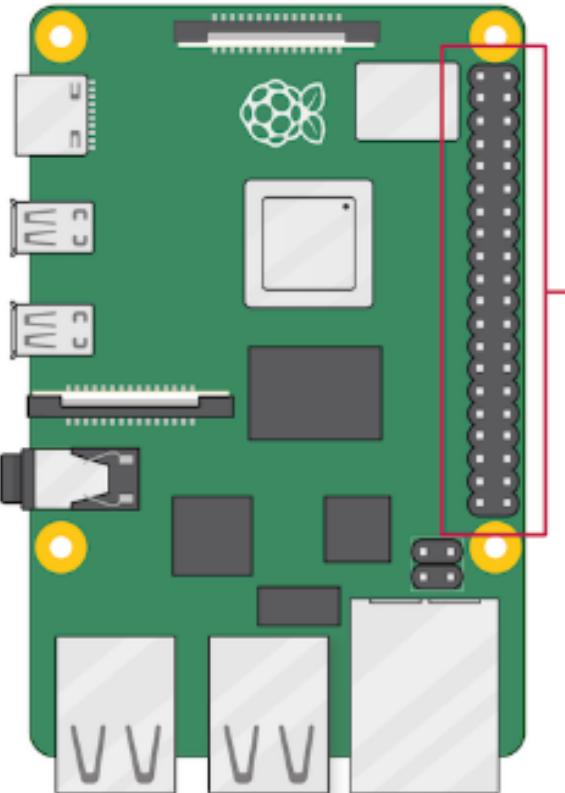
J8:

3V3	(1)	(2)	5V
GPI02	(3)	(4)	5V
GPI03	(5)	(6)	GND
GPI04	(7)	(8)	GPI014
GND	(9)	(10)	GPI015
GPI017	(11)	(12)	GPI018
GPI027	(13)	(14)	GND
GPI022	(15)	(16)	GPI023
3V3	(17)	(18)	GPI024
GPI010	(19)	(20)	GND
GPI09	(21)	(22)	GPI025
GPI011	(23)	(24)	GPI08
GND	(25)	(26)	GPI07
GPI00	(27)	(28)	GPI01
GPI05	(29)	(30)	GND
GPI06	(31)	(32)	GPI012
GPI013	(33)	(34)	GND
GPI019	(35)	(36)	GPI016
GPI026	(37)	(38)	GPI020
GND	(39)	(40)	GPI021

커널 모듈과 디바이스 드라이버 이해

라즈베리파이 pinout 상세정보

<https://pinout.xyz/>



3V3 power	1	2	5V power
GPIO 2 (SDA)	3	4	5V power
GPIO 3 (SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD)
Ground	9	10	GPIO 15 (RXD)
GPIO 17	11	12	GPIO 18 (PCM_CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3V3 power	17	18	GPIO 24
GPIO 10 (MOSI)	19	20	Ground
GPIO 9 (MISO)	21	22	GPIO 25
GPIO 11 (SCLK)	23	24	GPIO 8 (CE0)
Ground	25	26	GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM_FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM_DIN)
Ground	39	40	GPIO 21 (PCM_DOUT)

커널 모듈과 디바이스 드라이버 이해

Device Tree 이해

<https://www.devicetree.org/specifications/>

```
/ {
    node@0 {
        #include <reg>
        #include <string.h>
        #include <list.h>
        #include <byte-data.h>

        a-string-property = "A string";
        a-string-list-property = "first string", "second string";
        a-byte-data-property = [0x01 0x23 0x34 0x56];

        child-node@0 {
            first-child-property;
            second-child-property = <1>;
            a-reference-to-something = <&node1>;
        };
        child-node@1 {};
    };

    node1: node@1 {
        an-empty-property;
        a-cell-property = <1 2 3 4>;
        child-node@0 {};
    };
};
```

Properties of node@0

- Node name: /
- Unit address: node@0
- Property name: a-string-property, a-string-list-property, a-byte-data-property
- Property value: "A string", "first string", "second string", [0x01 0x23 0x34 0x56]
- Label: node1: node@1
- Bytestring: first-child-property, second-child-property
- A phandle (reference to another node): a-reference-to-something
- Four cells (32 bits values): a-cell-property

Device Tree (include 예제)

https://wiki.dreamrunner.org/public_html/Embedded-System/Linux-Device-Tree.html
include/dt-bindings/*.h

```
/ {  
    compatible = "ti,am33xx";  
    [...]  
    ocp {  
        uart0: serial@44e09000 {  
            compatible = "ti,omap3-uart";  
            reg = <0x44e09000 0x2000>;  
            interrupts = <72>;  
            status = "disabled";  
        };  
    };  
};  
am33xx.dtsci
```



```
#include "am33xx.dtsci"  
  
/ {  
    compatible = "ti,am335x-bone", "ti,am33xx";  
    [...]  
    ocp {  
        uart0: serial@44e09000 {  
            pinctrl-names = "default";  
            pinctrl-0 = <&uart0_pins>;  
            status = "okay";  
        };  
    };  
};  
am335x-bone.dts
```



Compiled DTB

```
/ {  
    compatible = "ti,am335x-bone", "ti,am33xx";  
    [...]  
    ocp {  
        uart0: serial@44e09000 {  
            compatible = "ti,omap3-uart";  
            reg = <0x44e09000 0x2000>;  
            interrupts = <72>;  
            pinctrl-names = "default";  
            pinctrl-0 = <&uart0_pins>;  
            status = "okay";  
        };  
    };  
};  
am335x-bone.dtb
```

Note: the real DTB is in binary format.
Here we show the text equivalent of the
DTB contents;

그림참조 : https://wiki.dreamrunner.org/public_html/Embedded-System/Linux-Device-Tree.html

커널 모듈과 디바이스 드라이버 이해

Device Tree 이해 (sysfs에서 정보 확인)

<https://www.kernel.bz/boardPost/118684/3>

```
$ cd /sys/firmware/devicetree/base
```

Device Tree 노드 정보

```
$ tree
```

```
$ cd /sys/bus/soc
```

device 장치 주소 정보

```
$ cd /sys/devices/platform/soc
```

```
$ tree
```



커널 모듈과 디바이스 드라이버 이해

Device Tree 이해 (라즈베리파이 DTB 32 비트)

arch/arm/boot/dts/Makefile

```
dtb-$(CONFIG_ARCH_BCM2835) += \  
    bcm2708-rpi-b.dtb \  
    bcm2708-rpi-b-rev1.dtb \  
    bcm2708-rpi-b-plus.dtb \  
    bcm2708-rpi-cm.dtb \  
    bcm2708-rpi-zero.dtb \  
    bcm2708-rpi-zero-w.dtb \  
    bcm2710-rpi-zero-2.dtb \  
    bcm2710-rpi-zero-2-w.dtb \  
    bcm2709-rpi-2-b.dtb \  
    bcm2710-rpi-2-b.dtb \  
    bcm2710-rpi-3-b.dtb \  
    bcm2710-rpi-3-b-plus.dtb \  
    bcm2711-rpi-4-b.dtb \  
    bcm2711-rpi-400.dtb \  
    bcm2710-rpi-cm3.dtb \  
    bcm2711-rpi-cm4.dtb \  
    bcm2711-rpi-cm4s.dtb
```

```
dtb-$(CONFIG_ARCH_BCM2835) += \  
    bcm2835-rpi-b.dtb \  
    bcm2835-rpi-a.dtb \  
    bcm2835-rpi-b-rev2.dtb \  
    bcm2835-rpi-b-plus.dtb \  
    bcm2835-rpi-a-plus.dtb \  
    bcm2835-rpi-cm1-io1.dtb \  
    bcm2836-rpi-2-b.dtb \  
    bcm2837-rpi-3-a-plus.dtb \  
    bcm2837-rpi-3-b.dtb \  
    bcm2837-rpi-3-b-plus.dtb \  
    bcm2837-rpi-cm3-io3.dtb \  
    bcm2835-rpi-zero.dtb \  
    bcm2835-rpi-zero-w.dtb
```

커널 모듈과 디바이스 드라이버 이해

Device Tree 이해 (라즈베리파이 DTB 64 비트)

arch/arm64/boot/dts/broadcom/Makefile

```
dtb-$(CONFIG_ARCH_BCM2835) += bcm2837-rpi-3-a-plus.dtb \
                           bcm2837-rpi-3-b.dtb \
                           bcm2837-rpi-3-b-plus.dtb \
                           bcm2837-rpi-cm3-io3.dtb
dtb-$(CONFIG_ARCH_BCM2835) += bcm2710-rpi-zero-2.dtb
dtb-$(CONFIG_ARCH_BCM2835) += bcm2710-rpi-zero-2-w.dtb
dtb-$(CONFIG_ARCH_BCM2835) += bcm2710-rpi-2-b.dtb
dtb-$(CONFIG_ARCH_BCM2835) += bcm2710-rpi-3-b.dtb
dtb-$(CONFIG_ARCH_BCM2835) += bcm2710-rpi-3-b-plus.dtb
dtb-$(CONFIG_ARCH_BCM2835) += bcm2711-rpi-4-b.dtb
dtb-$(CONFIG_ARCH_BCM2835) += bcm2711-rpi-400.dtb
dtb-$(CONFIG_ARCH_BCM2835) += bcm2710-rpi-cm3.dtb
dtb-$(CONFIG_ARCH_BCM2835) += bcm2711-rpi-cm4.dtb
dtb-$(CONFIG_ARCH_BCM2835) += bcm2711-rpi-cm4s.dtb
```

커널 모듈과 디바이스 드라이버 이해

Device Tree (라즈베리파이 broadcom CPU Model)

<https://wikimovel.com/index.php/Broadcom>

bcm283x

--> bcm2835(rpi-1) --> bcm2836(rpi-2) --> **bcm2837**(rpi-3)

--> bcm2708(rpi-1) --> **bcm2709**(rpi-2) --> bcm2710(rpi-3) --> **bcm2711**(rpi-4)

```
compatible = "raspberrypi,3-model-b", "brcm,bcm2837";  
  
model = "Raspberry Pi 3 Model B";  
  
arch/arm/boot/dts/bcm2837-rpi-3-b.dts  
#include "bcm2837.dtsi"  
    #include "bcm283x.dtsi"  
    #include "bcm2835-common.dtsi"  
    #include "bcm2835-rpi-common.dtsi"  
#include "bcm2836-rpi.dtsi"  
#include "bcm283x-rpi-smsc9514.dtsi"  
#include "bcm283x-rpi-usb-host.dtsi"
```

```
compatible = "raspberrypi,3-model-b", "brcm,bcm2837";  
model = "Raspberry Pi 3 Model B";
```

```
arch/arm/boot/dts/bcm2710-rpi-3-b.dts  
#include "bcm2710.dtsi"  
#include "bcm2709-rpi.dtsi"  
    #include "bcm2708-rpi.dtsi"  
    #include "bcm2835-rpi.dtsi"  
    #include "bcm270x-rpi.dtsi"  
#include "bcm283x-rpi-smsc9514.dtsi"  
#include "bcm283x-rpi-csi1-2lane.dtsi"  
#include "bcm283x-rpi-i2c0mux_0_44.dtsi"  
#include "bcm271x-rpi-bt.dtsi"
```

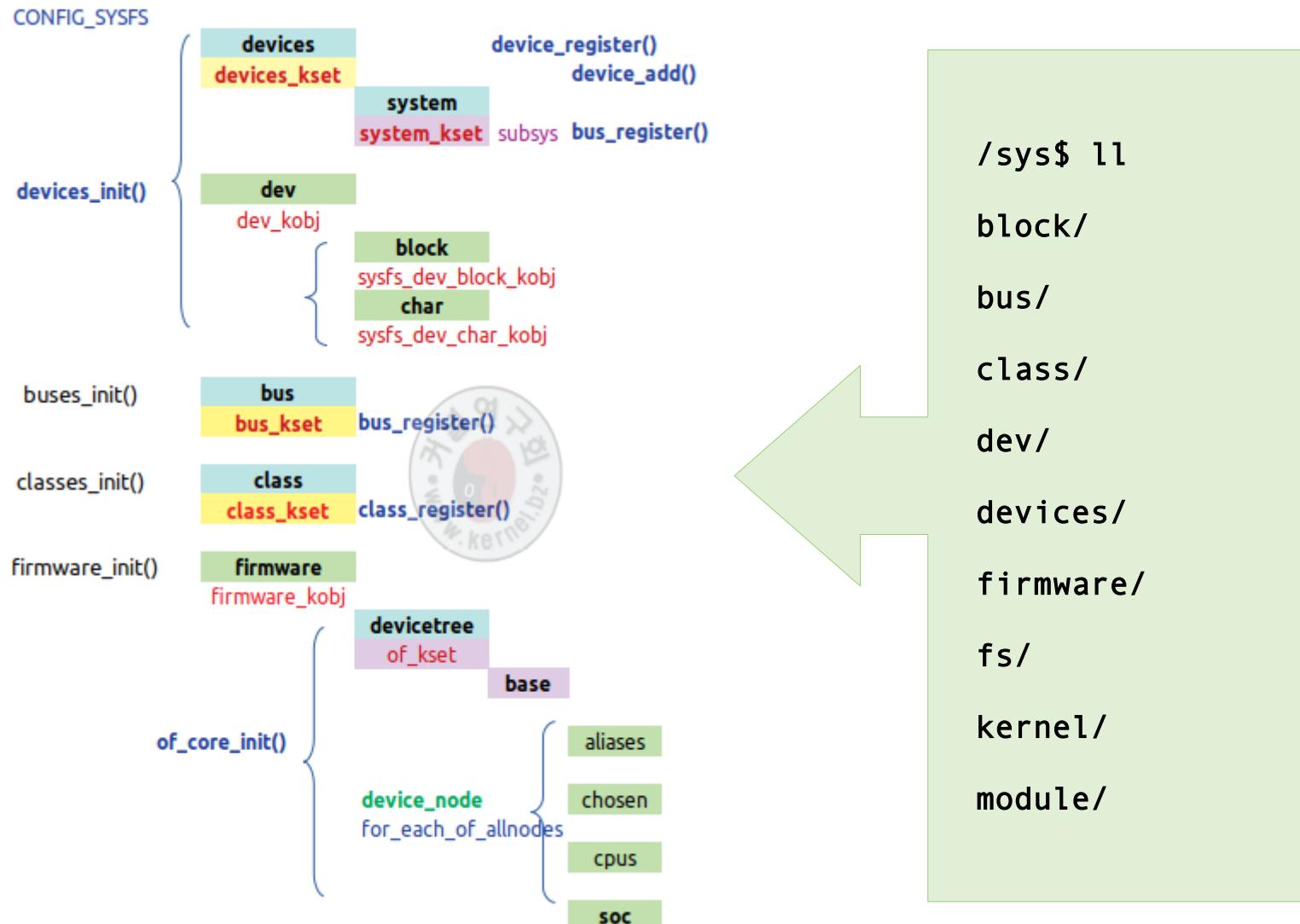
```
compatible = "raspberrypi,4-model-b", "brcm,bcm2711";  
model = "Raspberry Pi 4 Model B";
```

```
arch/arm/boot/dts/bcm2711-rpi-4-b.dts  
#include "bcm2711.dtsi"  
    #include "bcm283x.dtsi"  
    #include "bcm2835-rpi.dtsi"
```

커널 모듈과 디바이스 드라이버 이해

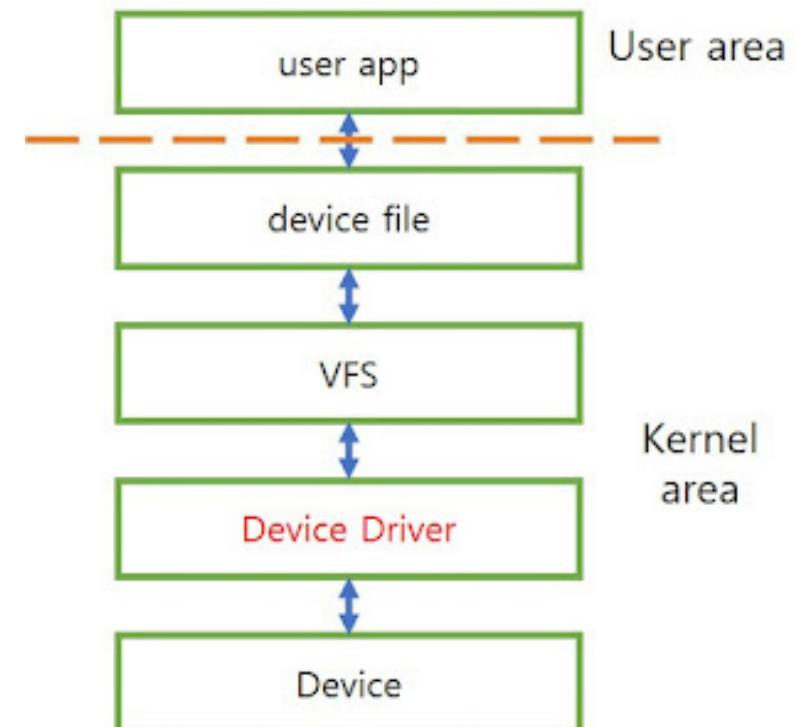
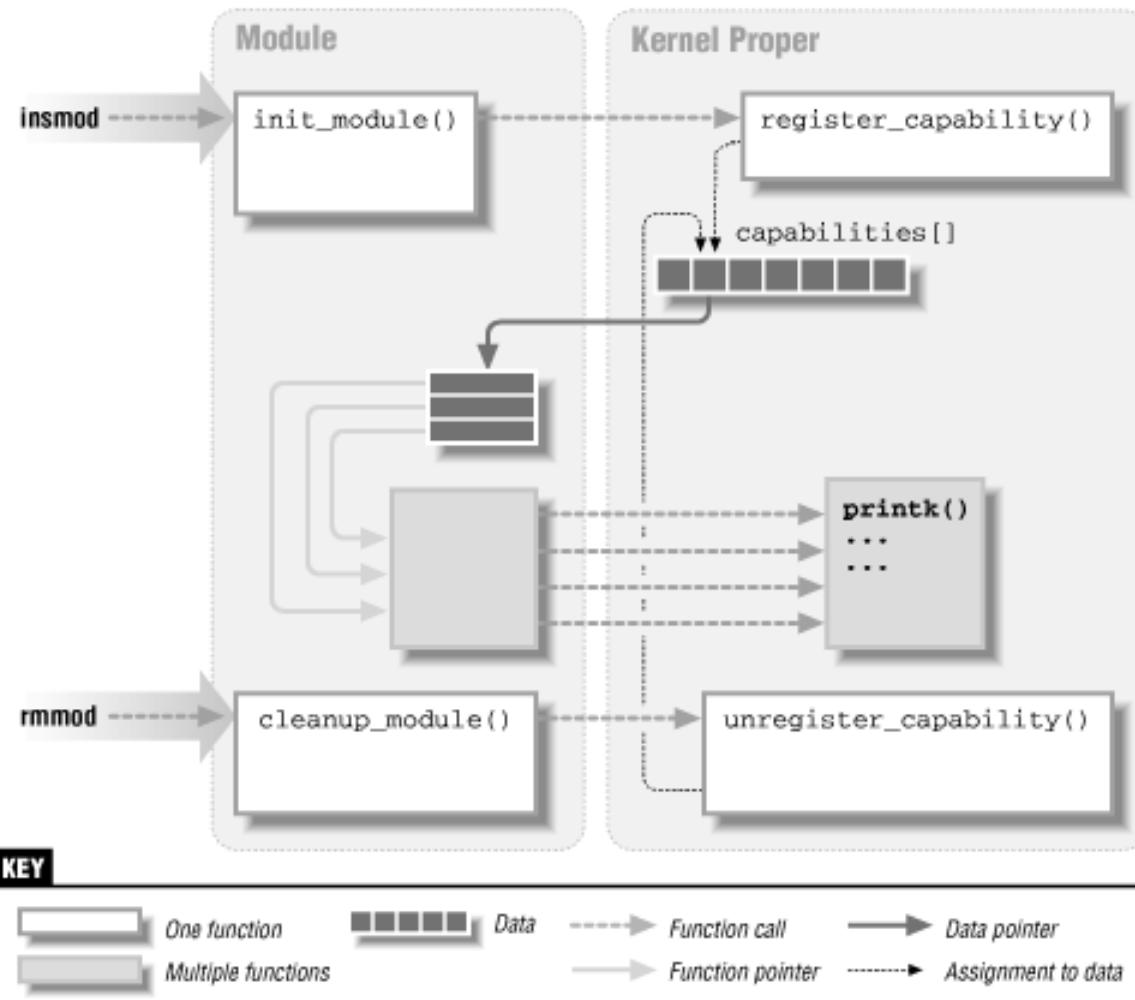
리눅스 커널 디바이스 계층구조 블럭도

<https://www.kernel.bz>



커널 모듈과 디바이스 드라이버 이해

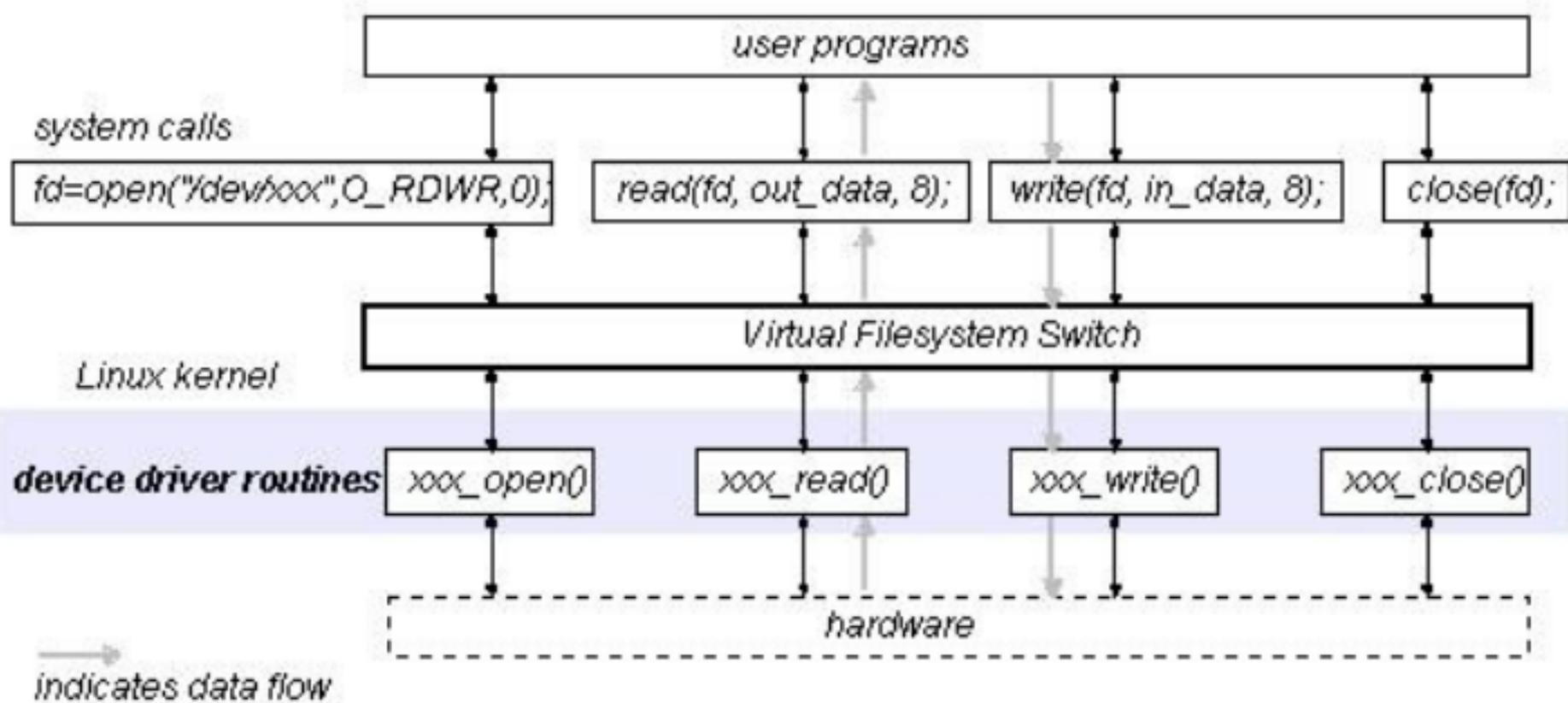
디바이스 드라이버 모듈 insmod / rmmod / lsmod /proc/modules/



커널 모듈과 디바이스 드라이버 이해

디바이스 드라이버 입출력 (read/write) 흐름도

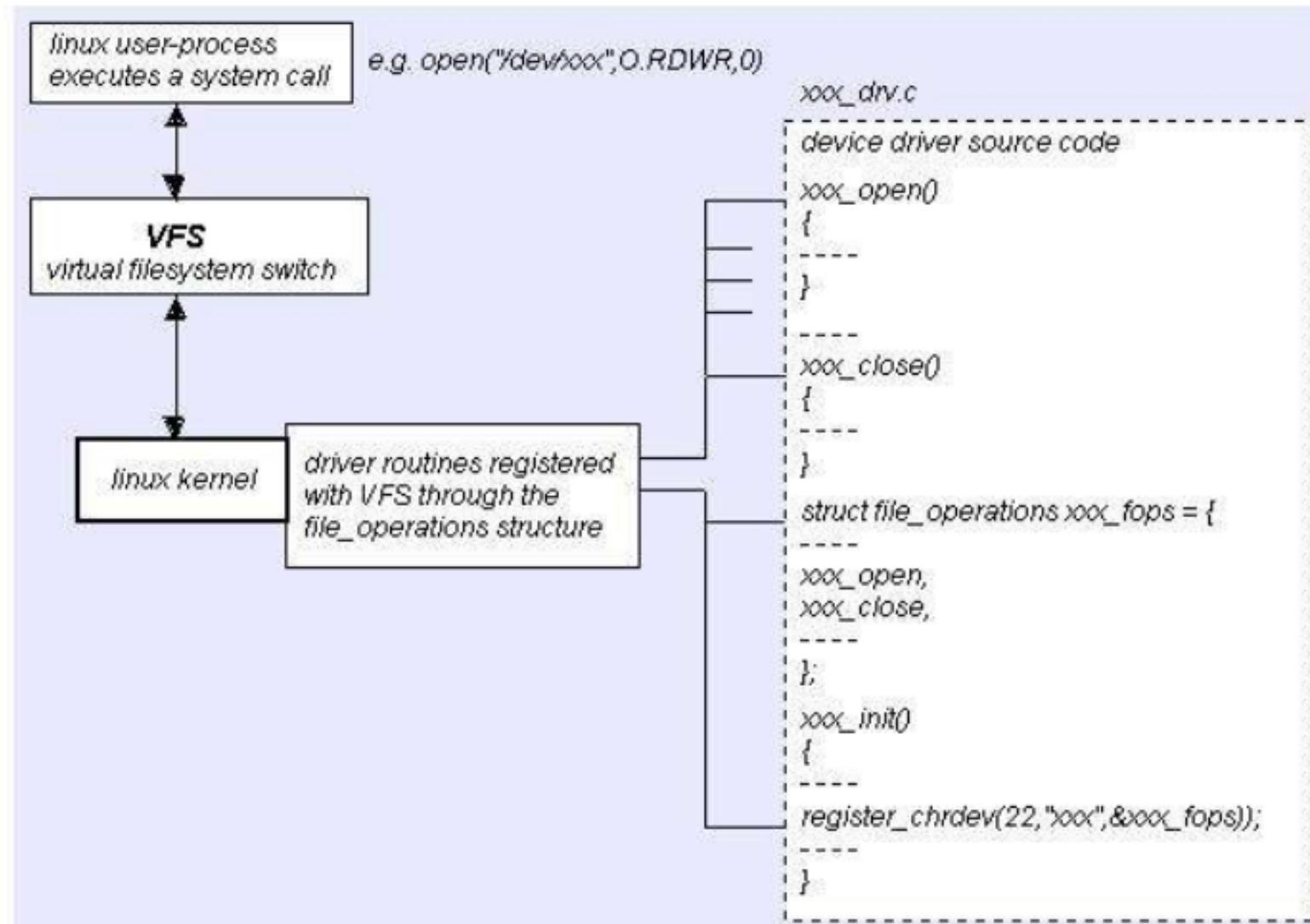
<https://www.makelinux.net/ldd3/chp-3-sect-7.shtml>



커널 모듈과 디바이스 드라이버 이해

디바이스 드라이버 입출력 (read/write) API 함수

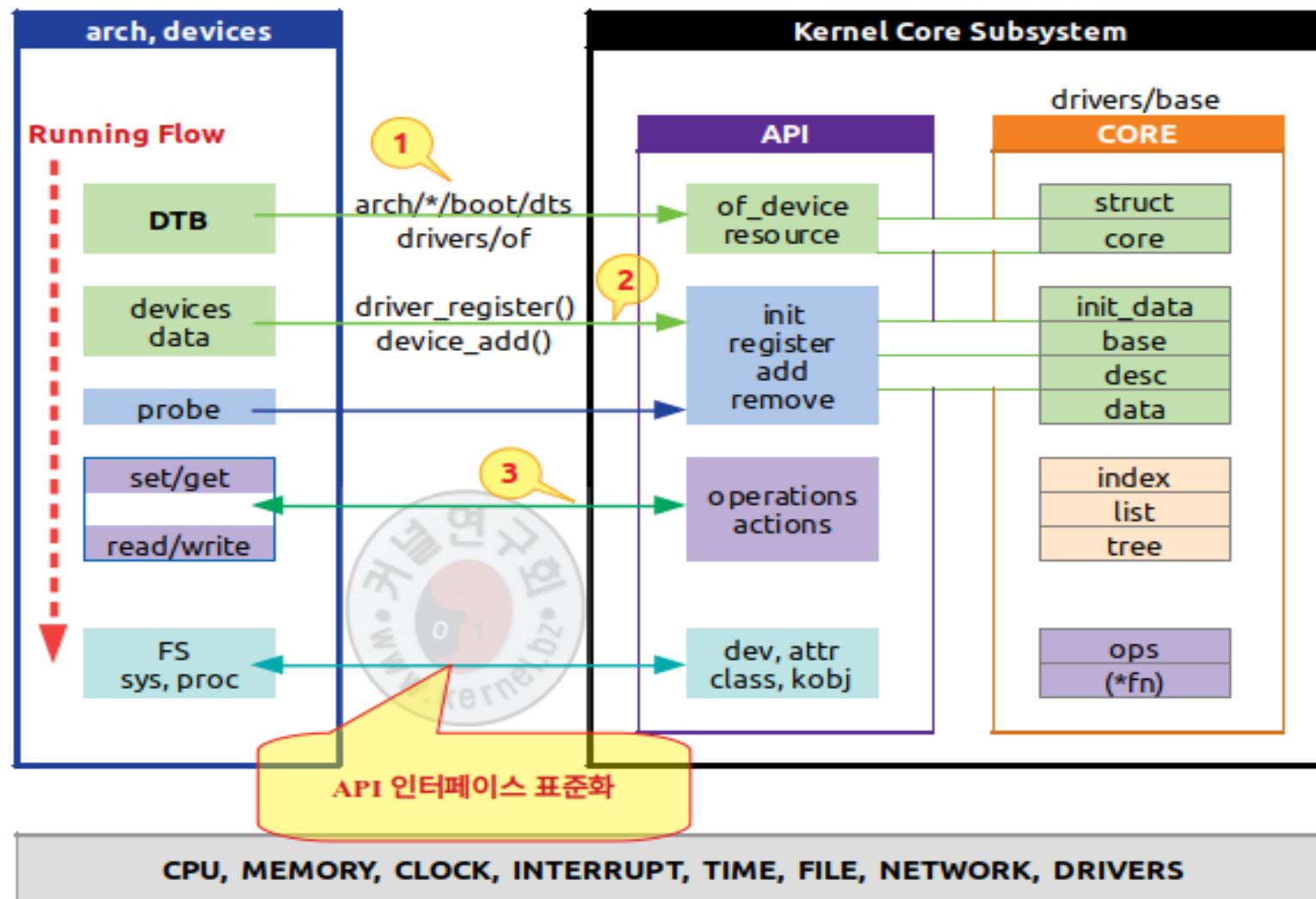
<https://www.makelinux.net/ldd3/chp-3-sect-7.shtml>



커널 모듈과 디바이스 드라이버 이해

리눅스 커널 디바이스 드라이버 전체 흐름 블럭도

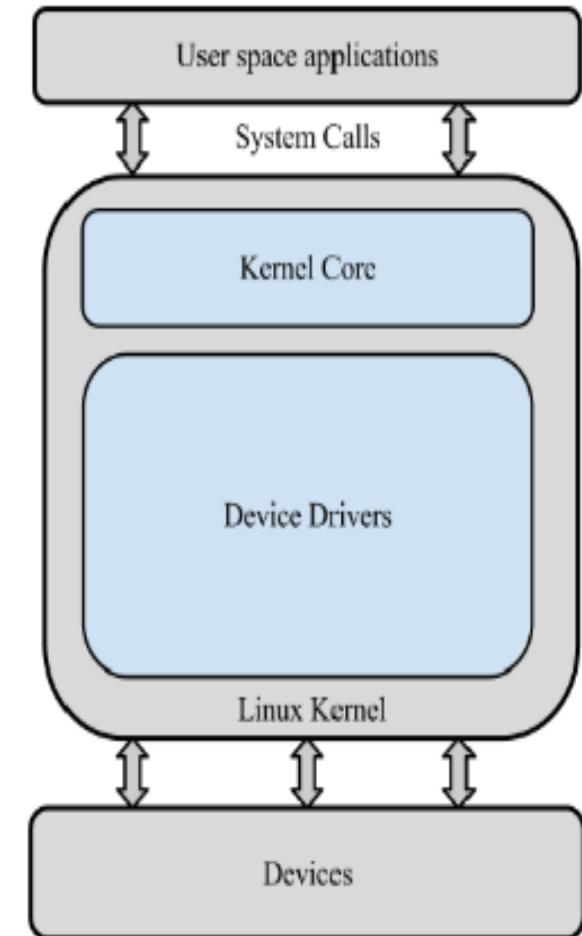
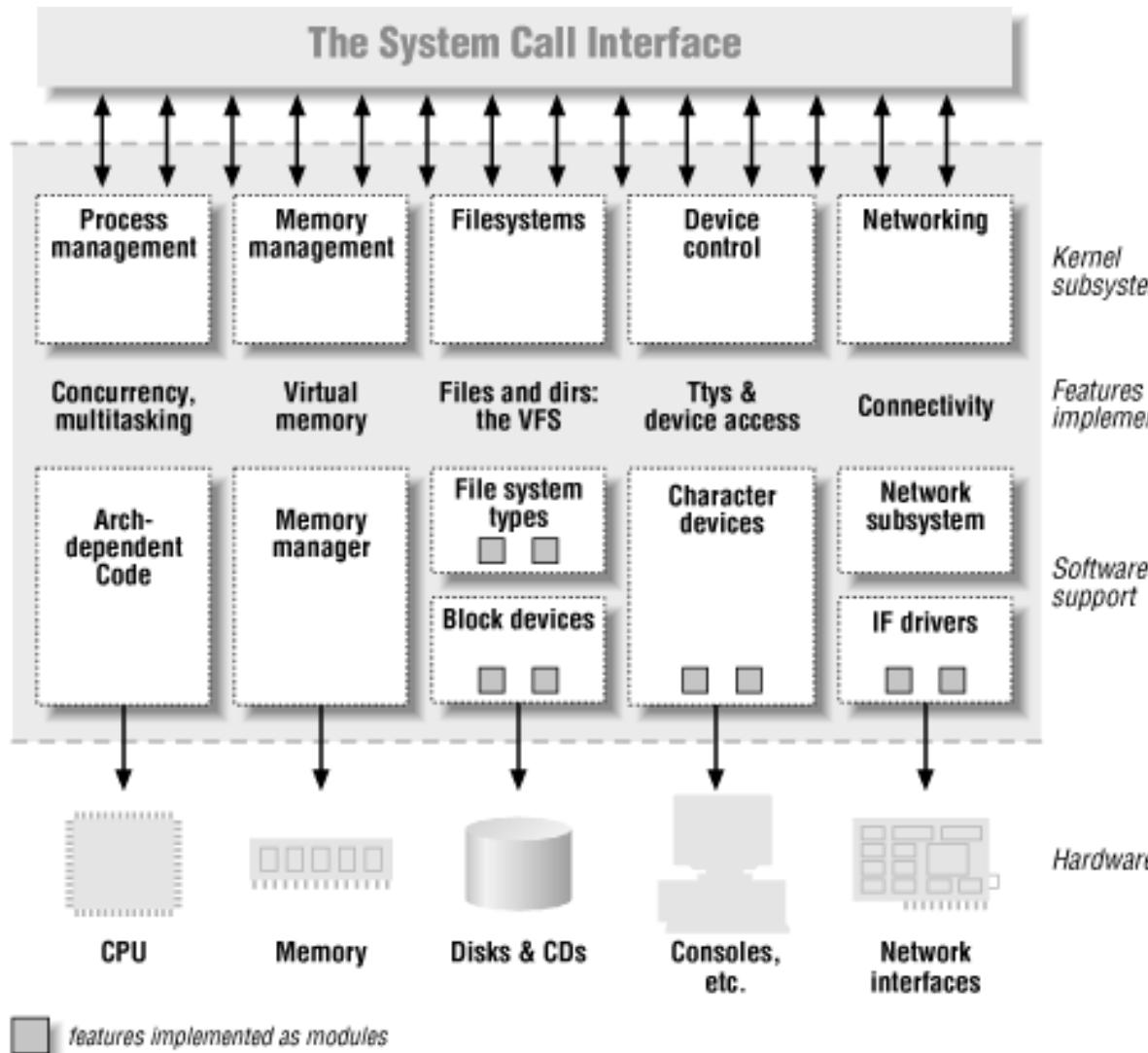
<https://www.kernel.bz/boardPost/118684/5>



시스템콜 이해

시스템콜 인터페이스 이해

<https://www.oreilly.com/library/view/linux-device-drivers/0596000081/ch01s02.html>



리눅스 커널 모듈 프로그래밍

목차



- 모듈 프로그래밍 준비사항
- 모듈 빌드용 **Makefile** 작성
- 모듈 빌드하기
- 모듈 삽입, 삭제, 확인
- 모듈 프로그래밍 실습 예제
 - ✓ **hello.ko**
 - ✓ **my-timer.ko**
 - ✓ **kobject-example.ko**

- 선행학습 (참조) 링크
- [리눅스 커널 모듈 프로그래밍](#)
- <https://www.kernel.bz/boardPost/118679/6?boardPage=2>

모듈 프로그래밍 준비 사항

커널 소스 빌드 및 설치 위치 (경로) 확인



- 리눅스 커널 소스가 빌드 (컴파일) 및 설치 되어 있어야 함 .
- 빌드 및 설치한 커널 버전으로 부팅되어 있어야 함 .
- 커널 소스 및 빌드한 경로 확인 .

로컬 머신에서 빌드한 경우는 **/lib/modules/ 커널버전 /**에서 확인 .

크로스 컴파일한 경우는 빌드 경로를 직접 확인 .

- 커널 모듈 **Makefile** 에 빌드 경로 설정 .
- 실습 예제 소스 :

git clone https://github.com/kernel-bz/kernel-study



모듈 프로그래밍 Makefile 작성 (로컬)

로컬 머신에서 빌드할 경우

make 실행

Makefile --> make hello

```
obj-m += hello.o  
hello:  
    make -C /lib/modules/${shell uname -r}/build M=$(PWD) modules
```

clean:

```
    make -C /lib/modules/${shell uname -r}/build M=$(PWD) clean
```

앞에 반드시 탭 문자 있어야 함

커널 버전

모듈 소스가 있는 경로

커널이 빌드되어 있는 경로 (링크파일로 연결되어 있음)
절대경로로 알려 주어도 됨



모듈 프로그래밍 Makefile(크로스 컴파일)

크로스 컴파일로 빌드할 경우

Makefile --> make hello make 실행

```
obj-m := hello.o
ARCH := arm
ARCC := arm-linux-gnueabihf-
MRUN := $(MAKE) ARCH=$(ARCH) CROSS_COMPILE=$(ARCC)

KDIR := /home/raspi/Projects/kernel/build/build-v5.10-arm-rpi
```

해당 아키텍쳐 (arm) 크로스 컴파일러

```
all: modules
modules:
    $(MRUN) -C $(KDIR) M=$(shell pwd) modules
```

커널이 빌드되어 있는 경로 (절대경로)

```
clean:
    $(MRUN) -C $(KDIR) M=$(shell pwd) clean
```

앞에 반드시 탭 문자 있어야 함

모듈 소스가 있는 경로



모듈 실행

modinfo, insmod, lsmod, rmmod

```
$ modinfo hello.ko
```

모듈 정보 조회

```
$ sudo insmod hello.ko
```

모듈 삽입 (실행)

```
$ lsmod
```

모듈 리스트 출력

```
$ sudo rmmod hello
```

모듈 제거 (삭제)



모듈 프로그래밍 실습 예제 (hello.ko)

module_init(), module_exit()

```
#define pr_fmt(fmt)      "MY-MODULE: " fmt

#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

static int __init example_init(void)
{
    pr_info("Hello world.\n");
    return 0;
}

static void __exit example_exit(void)
{
    pr_info("Goodbye world.\n");
}

module_init(example_init);
module_exit(example_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Linux");
```

모듈이 삽입 (insmod) 될때 실행됨

모듈이 삭제 (rmmod) 될때 실행됨



모듈 프로그래밍 실습 예제 (my-timer.ko)

timer_setup(), mod_timer()

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/timer.h>
#include <linux/jiffies.h>
```

```
my_timer1_callback()
```

```
timer_setup()
```

```
mod_timer()
```

```
module_init(my_timer_init)
module_exit(my_timer_exit)
```

커널 타이머 헤더파일

타이머 인터럽트 함수 (ISR)

타이머 등록 함수

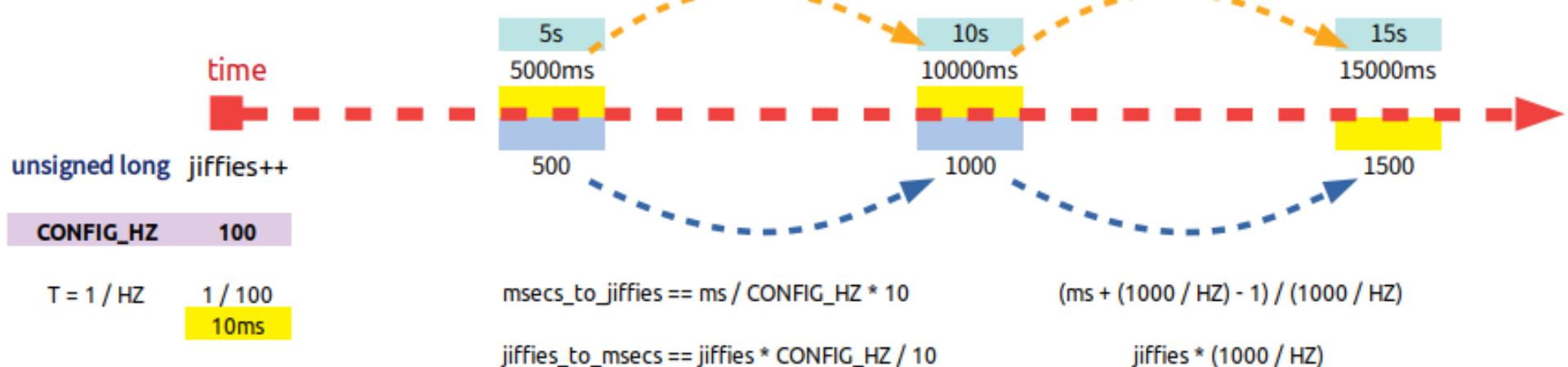
타이머 수정 (설정) 함수

타이머 모듈 삽입 및 제거



모듈 프로그래밍 실습 예제 (my-timer.ko)

jiffies 개념





모듈 프로그래밍 실습 예제 (kobject.ko)

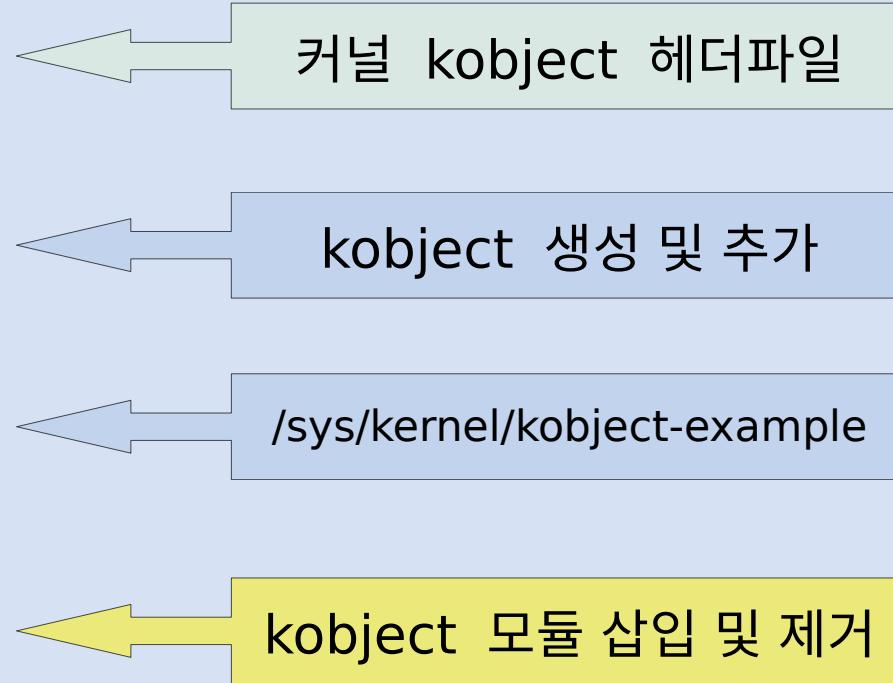
kobject_create_and_add(), sysfs_create_group()

```
#include <linux/kobject.h>
#include <linux/string.h>
#include <linux/sysfs.h>
#include <linux/module.h>
#include <linux/init.h>

kobject_create_and_add()

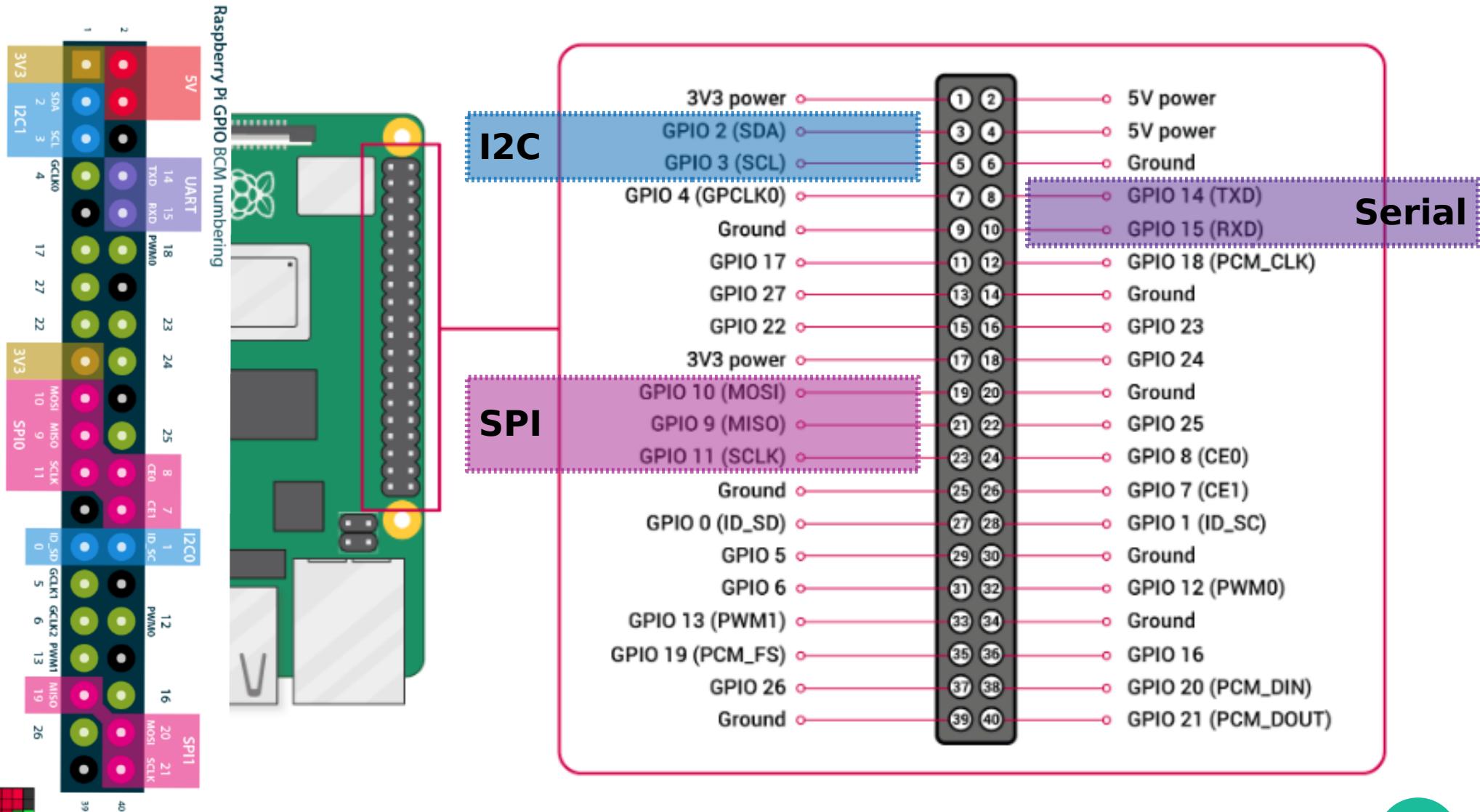
sysfs_create_group()

module_init(example_init)
module_exit(example_exit)
```



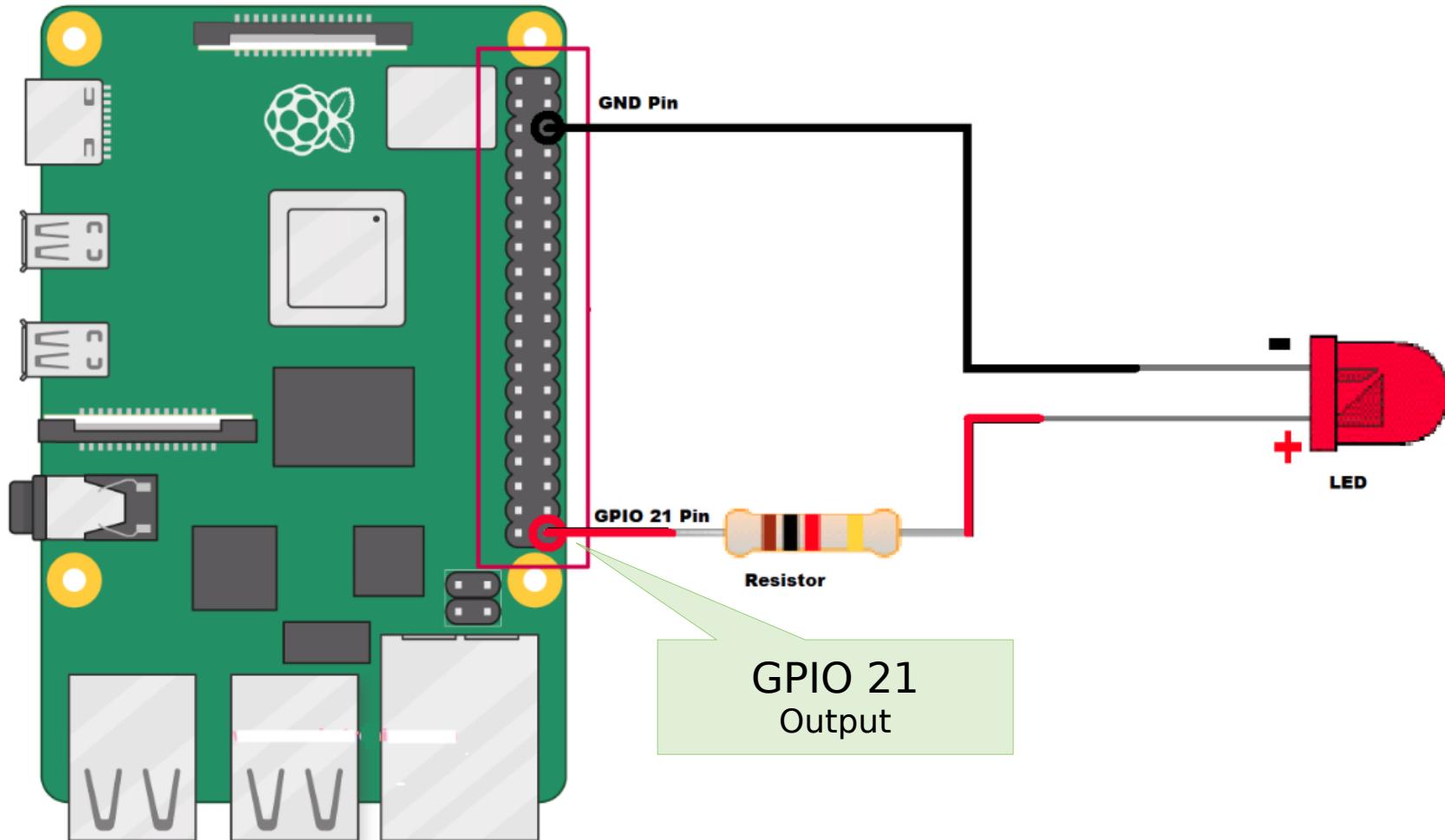
라즈베리파이 40pin Header (pinout)

<https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#gpio-and-the-40-pin-header>
<https://pinout.xyz/>



디바이스 드라이버 실습 예제 (GPIO in/out)

git clone <https://github.com/kernel-bz/kernel-study>



디바이스 드라이버 실습 예제 (GPIO in/out)

gpio_set_value(), gpio_get_value()

gpio_driver_init()

```
cdev_init(&gpio_cdev, &fops);
cdev_add(&gpio_cdev, dev, 1);
class_create();
device_create();

gpio_request(GPIO_21);
gpio_direction_output(GPIO_21, 0);
gpio_export(GPIO_21, false);
```

gpio_write()
gpio_set_value()

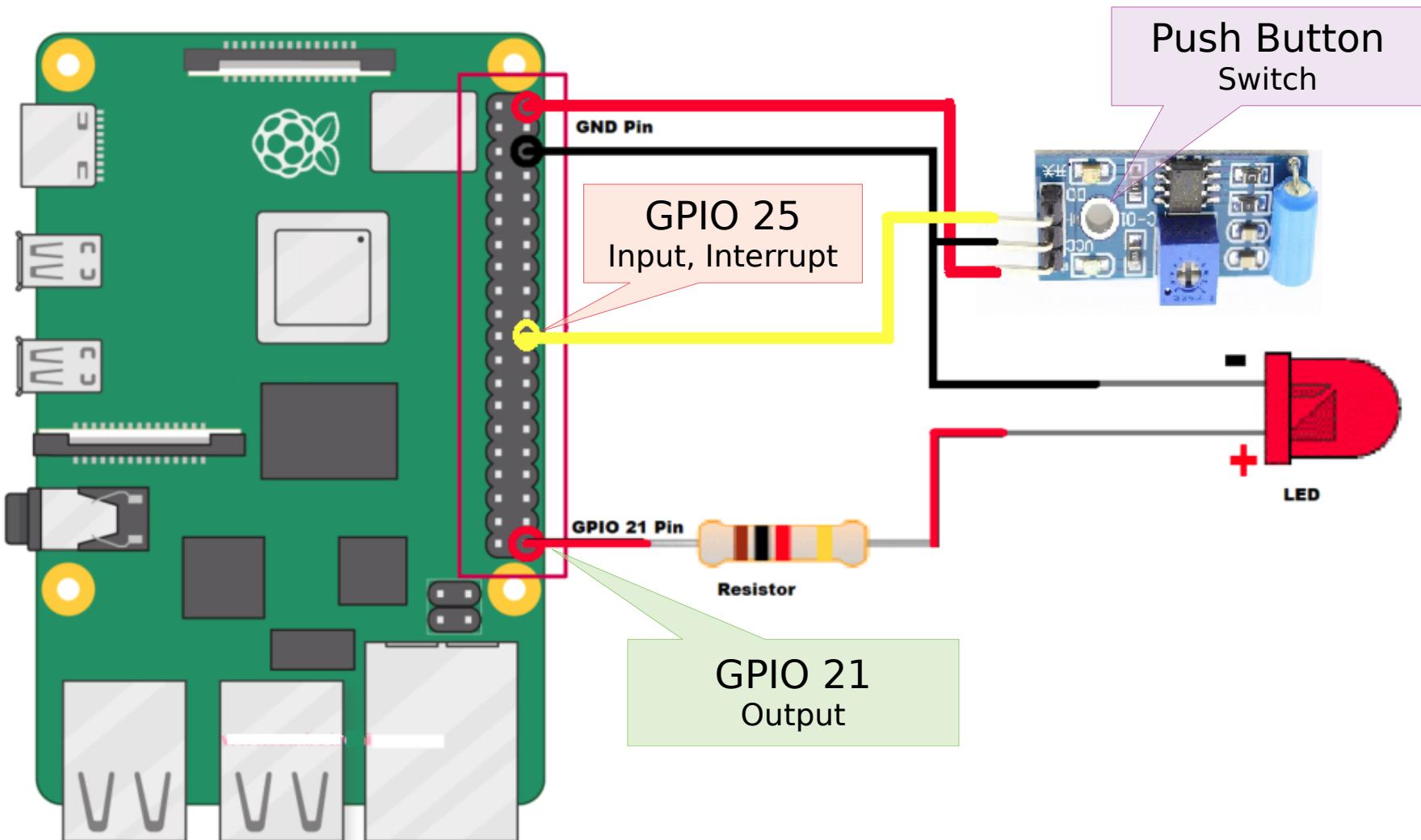
gpio_read()
gpio_get_value()

gpio_driver_exit()

```
/sys/class/gpio/gpio21
echo 1 > /sys/class/gpio/gpio21/value
echo 1 > /dev/gpio_io
/sys/devices/platform/soc/
```

디바이스 드라이버 실습 예제 (GPIO interrupt)

git clone <https://github.com/kernel-bz/kernel-study>



디바이스 드라이버 실습 예제 (GPIO interrupt)

request_irq(gpio_irq, gpio_irq_handler)

gpio_driver_init()

```
cdev_init(&gpio_cdev, &fops);
cdev_add(&gpio_cdev, dev, 1);
class_create();
device_create();
```

```
gpio_direction_output(GPIO_21_OUT, 0);
gpio_direction_input(GPIO_25_IN);
```

```
gpio_irq = gpio_to_irq(GPIO_25_IN);
request_irq(gpio_irq, gpio_irq_handler);
```

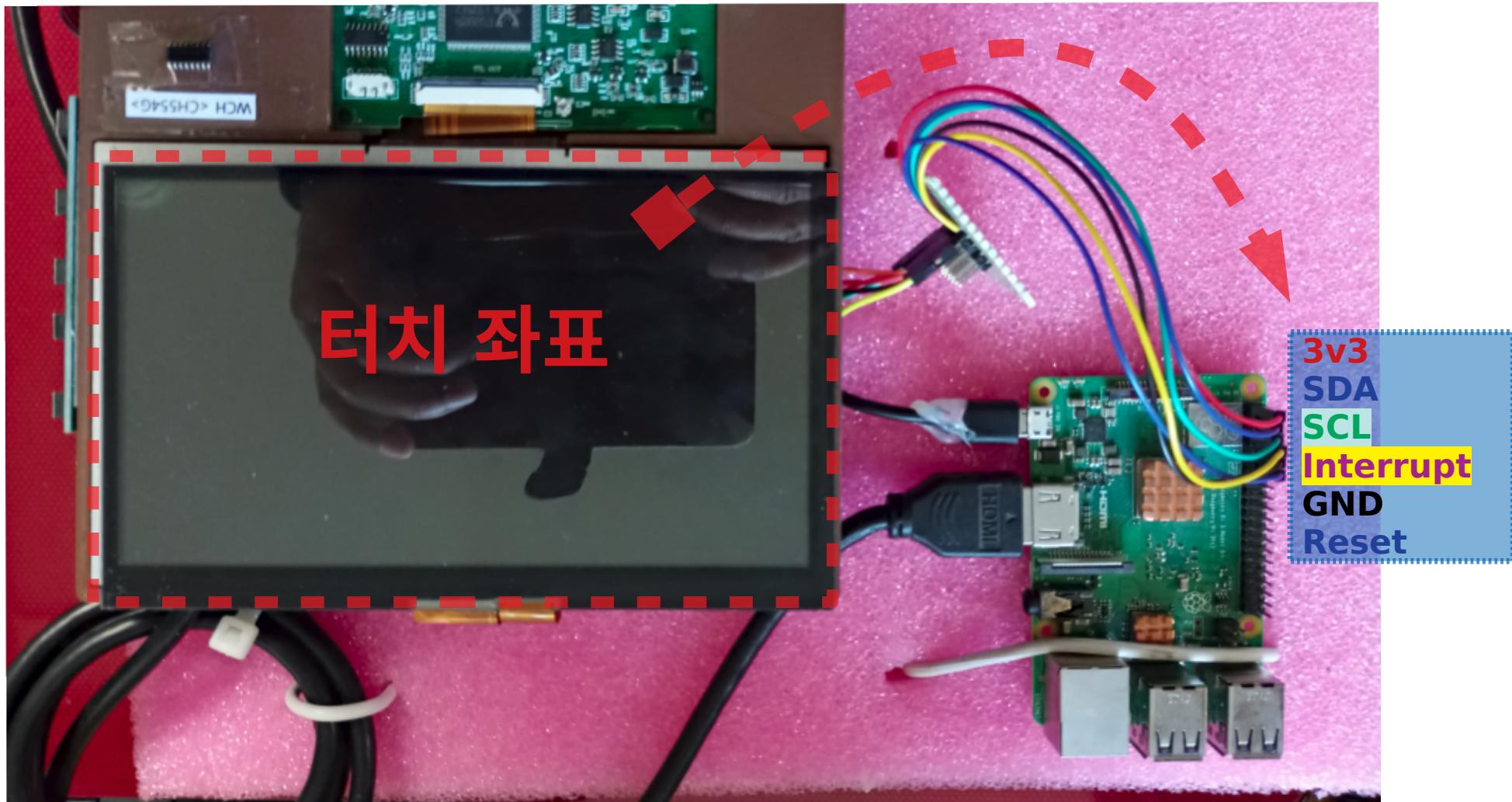
gpio_irq_handler()
gpio_set_value()

```
echo 0 > /dev/gpio_int
echo 1 > /dev/gpio_int
```

gpio_driver_exit()

I2C Device Driver (Device Tree)

input touchscreen, compatible = "goodix,gt9271"



I2C Device Driver (Device Tree)

```
input touchscreen, compatible = "goodix,gt9271"
```

//arch/arm/boot/dts/overlays/README

Name: goodix
Info: Enables I2C connected Goodix gt9271 multiple touch controller using GPIOs 4 and 17 (pins 7 and 11 on GPIO header) for interrupt and reset.
Load: dtoverlay=goodix,<param>=<val>
Params: interrupt GPIO used for interrupt (default 4)
reset GPIO used for reset (default 17)

//boot/config.txt

[all]

dtparam=i2c_arm=on

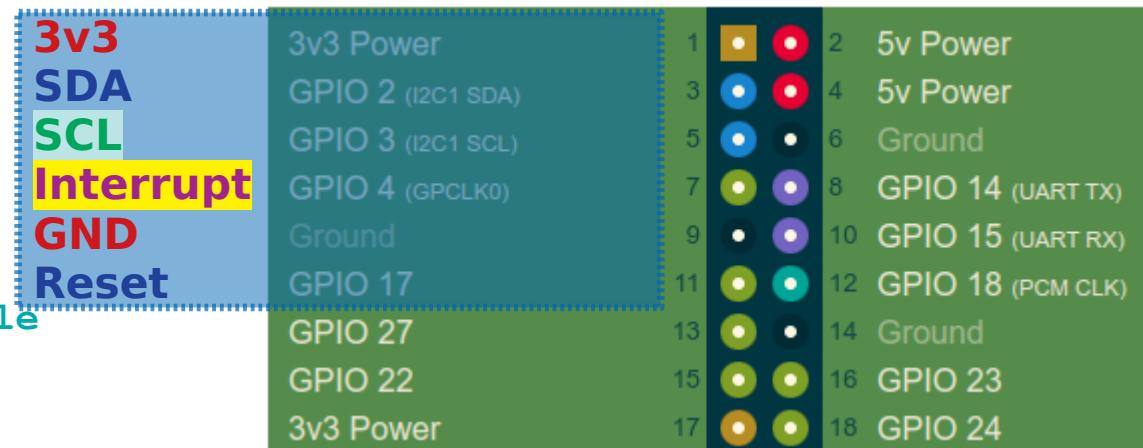
dtparam=i2c_arm baudrate=400000

dtoverlay=goodix

//arch/arm/boot/dts/overlays/Makefile

dtbo-\$ (CONFIG_ARCH_BCM2835) += \

goodix.dtbo \



//Device Tree 속성

[arch/arm/boot/dts/overlays/goodix-overlay.dts](#)

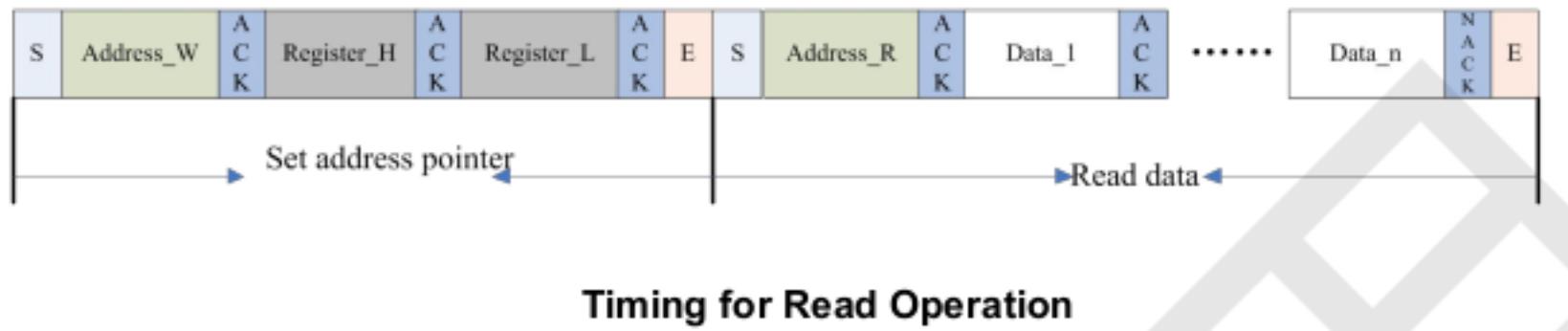
I2C Device Driver (Device Tree)

input touchscreen, compatible = "goodix,gt9271"

https://github.com/goodix/gt9xx_driver_android/blob/master/documents/Potring_Guide.md

c) Reading Data from GT9271

(For example: slave address is 0xBA/0xBB)



b) Writing Data to GT9271

(For example: slave address is 0xBA/0xBB)



I2C Device Driver (Device Tree Source)

input touchscreen, compatible = "goodix,gt9271"

arch/arm/boot/dts/overlays/goodix-overlay.dts

```
//arch/arm/boot/dts/overlays/goodix-overlay.dts
```

```
compatible = "brcm,bcm2835";
```

```
fragment@0 {
    target = <&gpio>;
    __overlay__ {
        goodix_pins: goodix_pins {
            brcm,pins = <4 17>; // interrupt and reset
            brcm,function = <0 0>; // in
            brcm,pull = <2 2>; // pull-up
        };
    };
};
```

```
/sys/firmware/devicetree/base/soc/gpio@7e200000
```

```
$ tree goodix_pins/
```

```
goodix_pins/
└── brcm,function
└── brcm,pins
└── brcm,pull
└── name
└── phandle
```

I2C Device Driver (Device Tree Source)

input touchscreen, compatible = "goodix,gt9271"

arch/arm/boot/dts/overlays/goodix-overlay.dts

```
//arch/arm/boot/dts/overlays/goodix-overlay.dts

fragment@1 {
    target = <&i2c1>;
    __overlay__ {
        #address-cells = <1>;
        #size-cells = <0>;
        status = "okay";

        gt9271: gt9271@14 {
            compatible = "goodix,gt9271";
            reg = <0x14>; // I2C address
            pinctrl-names = "default";
            pinctrl-0 = <&goodix_pins>;
            interrupt-parent = <&gpio>;
            interrupts = <4 2>; // high-to-low edge triggered
            irq-gpios = <&gpio 4 0>; // Pin7 on GPIO header
            reset-gpios = <&gpio 17 0>; // Pin11 on GPIO header
        };
    };
}
```

```
/sys/firmware/devicetree/base/soc
$ tree i2c@7e804000
```

```
i2c@7e804000
└── #address-cells
└── clock-frequency
└── clocks
└── compatible
    └── gt9271@14
        ├── compatible
        ├── interrupt-parent
        ├── interrupts
        ├── irq-gpios
        ├── name
        ├── phandle
        ├── pinctrl-0
        ├── pinctrl-names
        ├── reg
        └── reset-gpios
```

I2C Device Driver (Kconfig)

input touchscreen, compatible = "goodix,gt9271"

drivers/input/touchscreen/Kconfig, Makefile

//menuconfig

Symbol: TOUCHSCREEN_GOODIX [=m]

Type : tristate
Prompt: Goodix I2C touchscreen
 -> Device Drivers
 -> Input device support
 -> Generic input layer (needed for keyboard, mouse, ...)
(1) -> Touchscreens (INPUT_TOUCHSCREEN [=y])

//drivers/input/touchscreen/Kconfig:378

```
config TOUCHSCREEN_GOODIX
    tristate "Goodix I2C touchscreen"
    depends on I2C
    depends on GPIOLIB || COMPILE_TEST
```

//drivers/input/touchscreen/Makefile:47

```
obj-$(CONFIG_TOUCHSCREEN_GOODIX) += goodix.o
```

// 커널 메인라인 소스에 포팅되어 있음

```
drivers/input/touchscreen/goodix.c
```

I2C Device Driver (i2c 장치 등록)

drivers/input/touchscreen/goodix.c

input module_i2c_driver(goodix_ts_driver)

```
//drivers/input/touchscreen/goodix.c
```

```
static SIMPLE_DEV_PM_OPS(goodix_pm_ops, goodix_suspend, goodix_resume);
```

```
static struct i2c_driver goodix_ts_driver = {  
    .probe = goodix_ts_probe,  
    .remove = goodix_ts_remove,  
    .id_table = goodix_ts_id,  
    .driver = {  
        .name = "Goodix-TS",  
        .acpi_match_table = ACPI_PTR(goodix_acpi_match),  
        .of_match_table = of_match_ptr(goodix_of_match),  
        .pm = &goodix_pm_ops,  
    },  
};  
module_i2c_driver(goodix_ts_driver);
```

CONFIG_X86 && CONFIG_ACPI
ACPI_GPIO_SUPPORT(firmware)

Device Tree Matching Table
compatible = "goodix,gt9271"

I2C Device Driver (probe)

drivers/input/touchscreen/goodix.c

goodix_ts_probe(i2c_client, i2c_device_id)

```
goodix_get_gpio_config(ts)
    ts->gpiod_int = devm_gpiod_get_optional(GOODIX_GPIO_INT_NAME)
    ts->gpiod_rst = devm_gpiod_get_optional(GOODIX_GPIO_RST_NAME)
    ts->reset_controller_at_probe = true
    ts->load_cfg_from_disk = true
    ts->irq_pin_access_method = IRQ_PIN_ACCESS_GPIO
```

Device Tree 정보 가져오기

```
regulator_enable(ts->avdd28, ts->vddio)
```

```
devm_add_action_or_reset(goodix_disable_regulators)
```

```
goodix_reset(ts)
goodix_i2c_test()
goodix_read_version()
```

```
//&gt9x_chip_data
ts->chip = goodix_get_chip_data(ts->id)
```

```
if (ts->load_cfg_from_disk)
    request_firmware_nowait()
else
```

장치 firmware 요청 및 읽기

```
    goodix_configure_dev(ts)
```

터치스크린 장치 설정

I2C Device Driver (interrupt)

input touchscreen, goodix_request_irq()

drivers/input/touchscreen/goodix.c

```
//drivers/input/touchscreen/goodix.c
```

인터럽트 처리 함수 (ISR)

```
static irqreturn_t goodix_ts_irq_handler(int irq, void *dev_id)
{
    struct goodix_ts_data *ts = dev_id;

    goodix_process_events(ts);

    if (goodix_i2c_write_u8(ts->client, GOODIX_READ_COOR_ADDR, 0) < 0)
        dev_err(&ts->client->dev, "I2C write end cmd error\n");

    return IRQ_HANDLED;
}
```

터치 좌표 읽기 및 처리

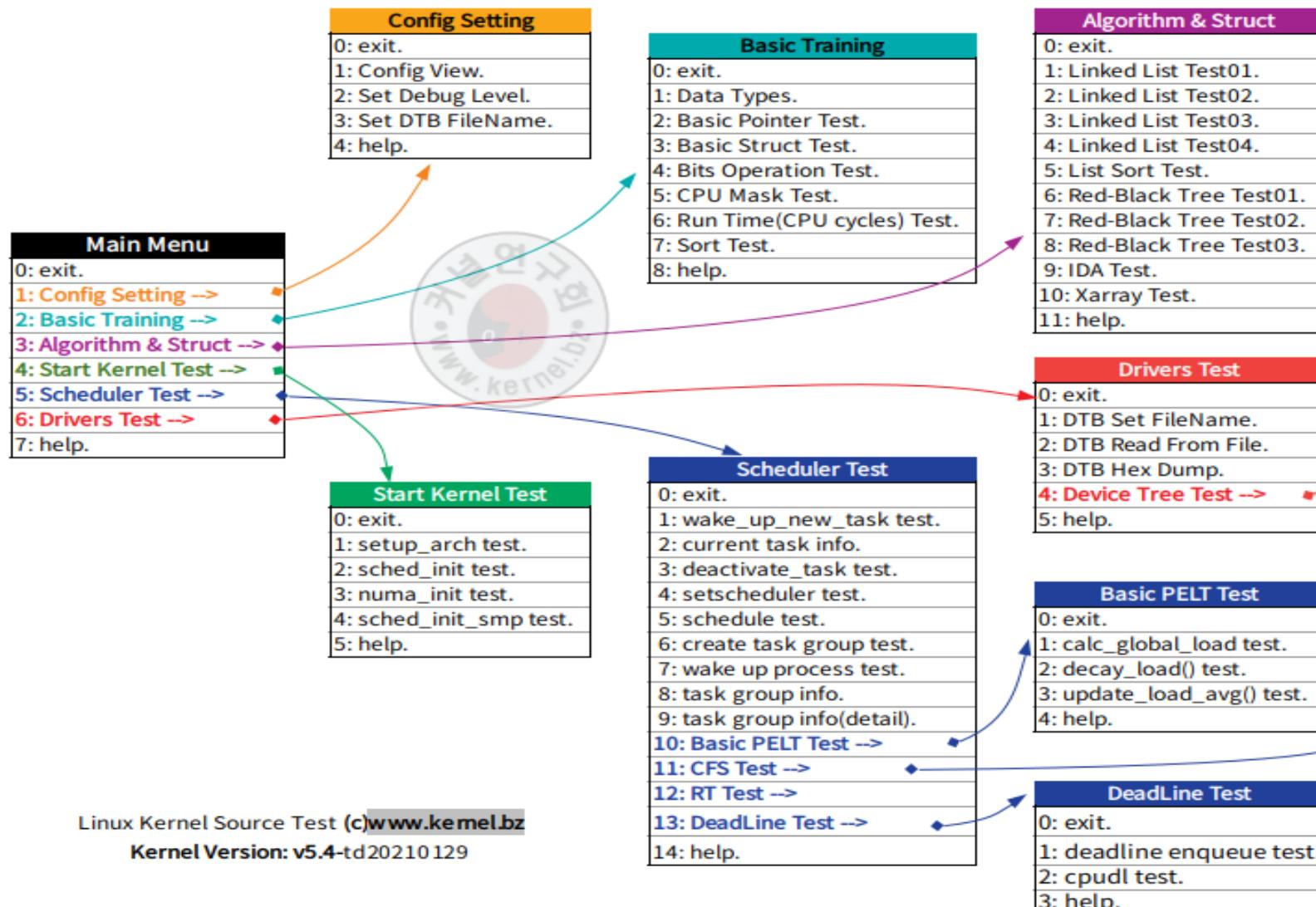
장치 인터럽트 등록

```
static int goodix_request_irq(struct goodix_ts_data *ts)
{
    return devm_request_threaded_irq(&ts->client->dev, ts->client->irq,
                                    NULL, goodix_ts_irq_handler,
                                    ts->irq_flags, ts->client->name, ts);
}
```

커널 소스 실행 분석 프로그램 소개

<https://www.kernel.bz/kanalyzer>

git clone <https://github.com/kernel-bz/linux-kernel-test.git>



임베디드 리눅스 커널 디바이스 드라이버

강의내용 전체 요약 및 질문 응답 (Q&A)

궁금한 내용은 질문 주세요 ~



정재준 <rabi3307@naver.com>
커널연구회 <www.kernel.bz>