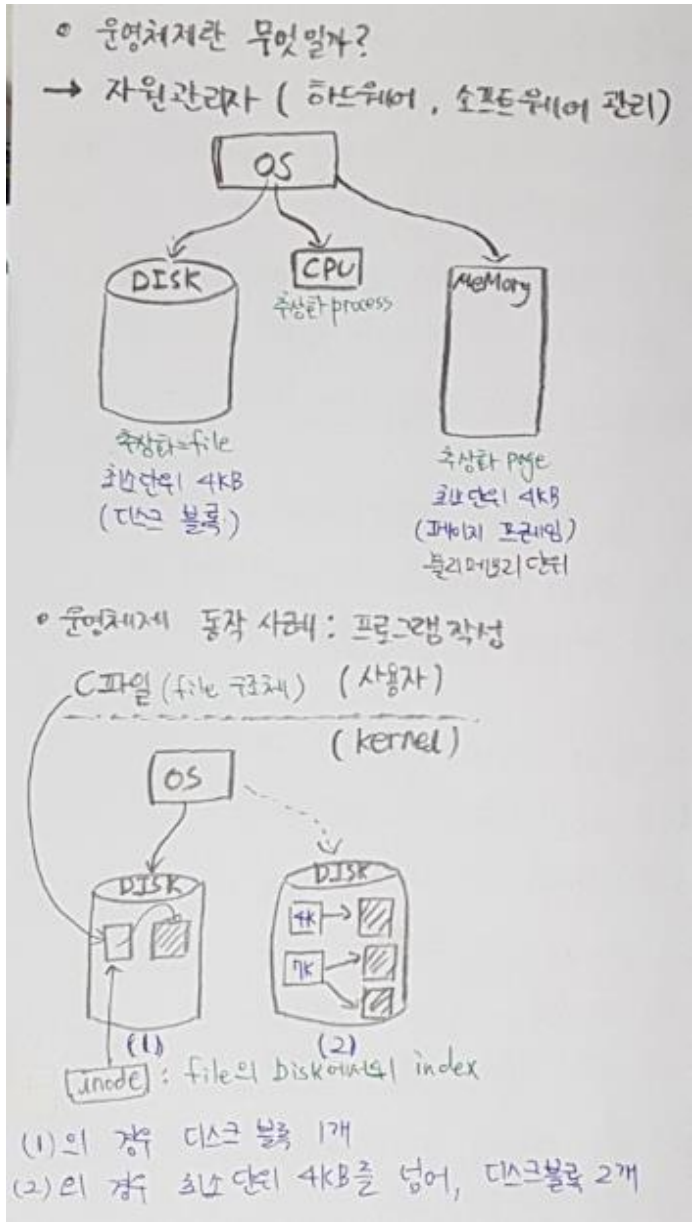
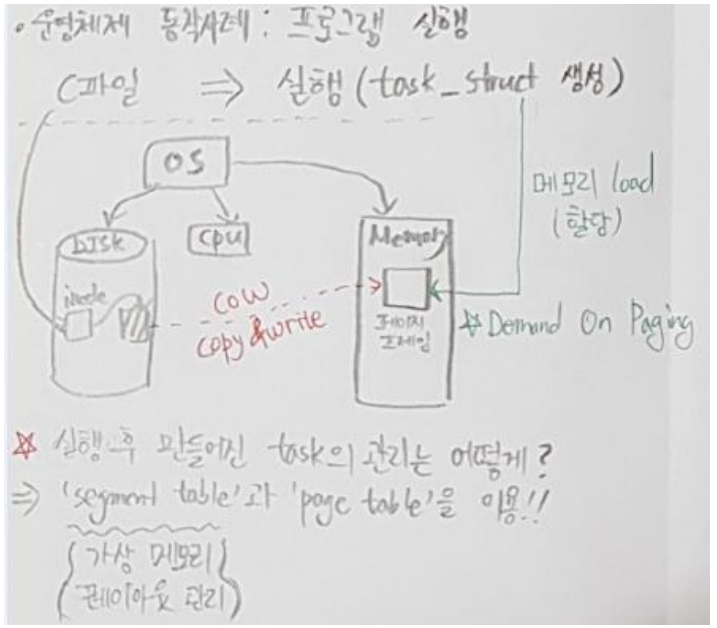
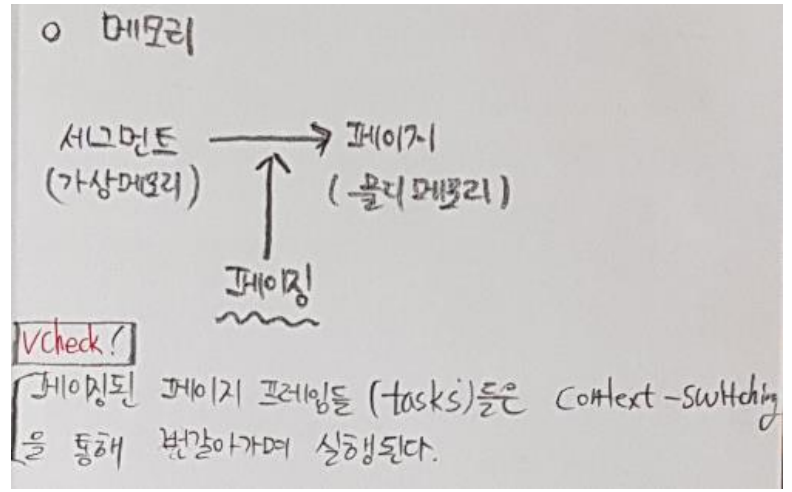
TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – GJ (박현우)
uc820@naver.com

1. 리눅스 커널 내부 구조 1 (운영체제)







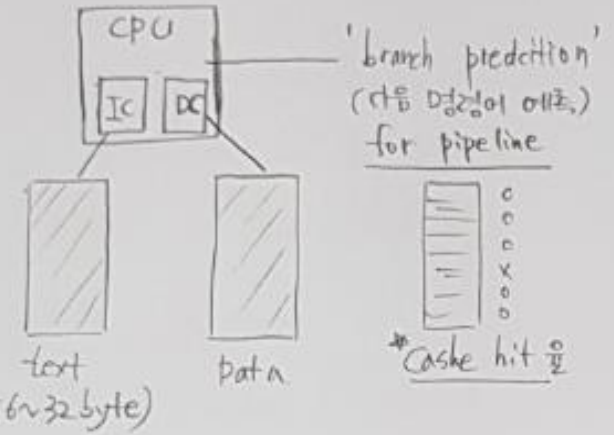
1. 리눅스 커널 내부 구조 2 (동작 비유 1 ~ 2)

○ 운영체제 동작비유 <1>

1) Firmware Level
⇒ OS가 없는 상태

2) 다 직접 코딩하기에 역부족
⇒ 'scheduling'도 안되고 'cache hit'를 알수 없어서 같이 고안됨

* scheduling (context switching) - timing 값
↳ CPU에 할당



3) zImage (리눅스 커널 압축 이미지)
Root Filesystem (Linux만 있음)
⇒ (pwd, (/)
bin usr ...
Bootloader (하드웨어 초기화)
⇒ 하드웨어 초기화 이전은 'floating 상태'
◦ 가상메모리 공간 확보
◦ linux 커널 올림

4) 시스템디자인에만 중점
⇒ 하드웨어 디자인

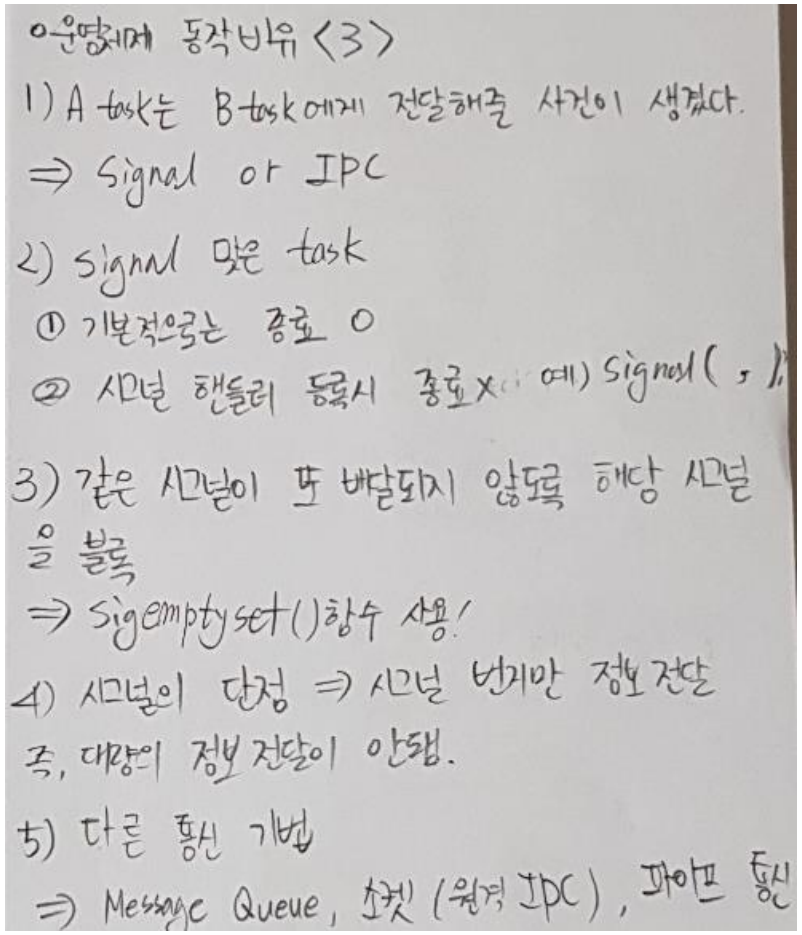
② 운영체제 관리 <5대요소>

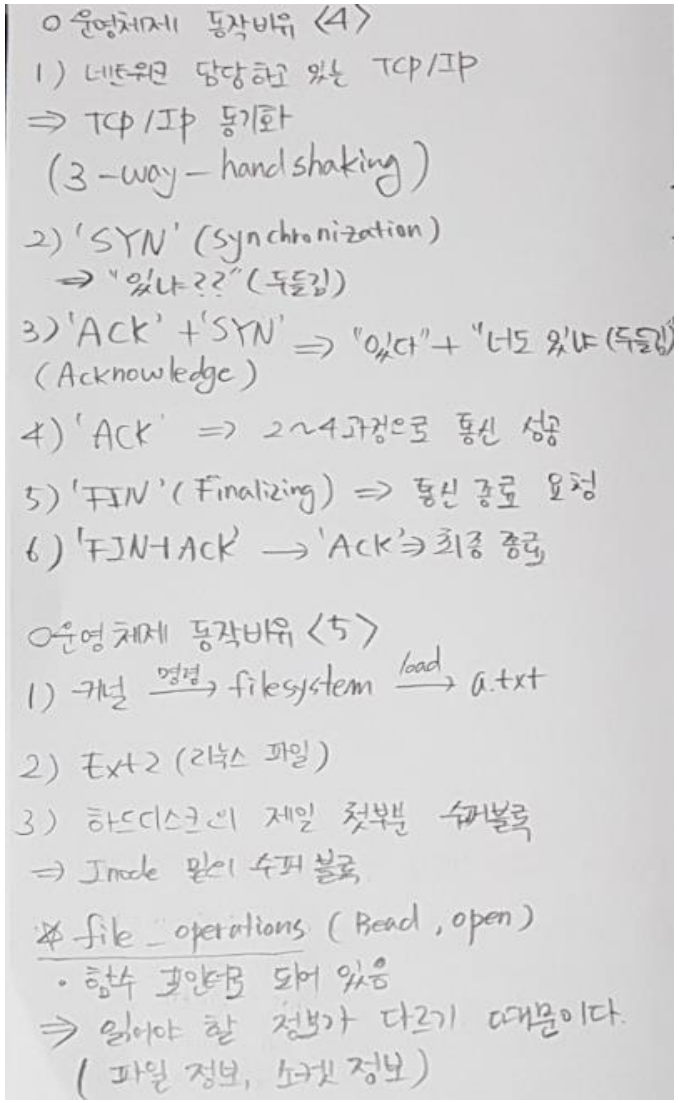
1. filesystem
2. Memory
3. Network
4. Device driver (장치동 software)
5. Process (task)

○ 운영체제 동작비유 <2>

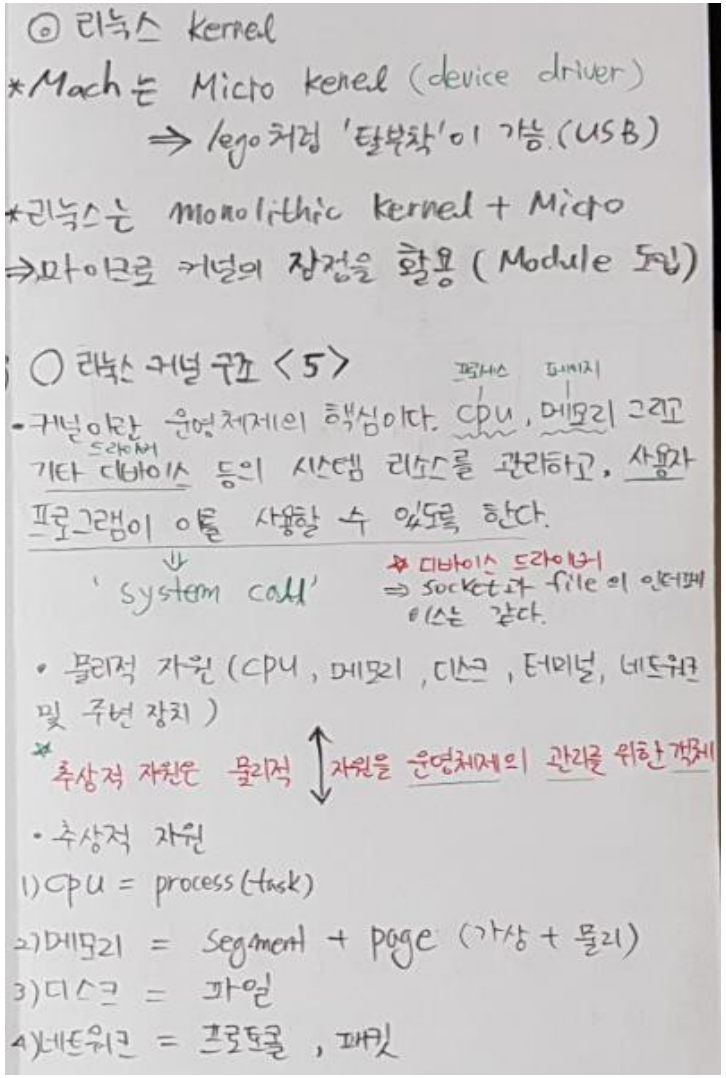
1) FIFO OS ⇒ RTOS는 아님
2) 정해진 time slice 만큼씩만 CPU 자원을 할당
⇒ 범용우선순위 (hybrid 방식) ↔ 정적우선순위
3) 단 하나의 태스크만 실제 running, 나머지는 대기 상태 ⇒ 'single core'
4) 태스크의 속성을 고려한 time slice
우선순위
5) 높은 태스크가 선점(preemtion)을 요청
⇒ 완전히 갈아치우지 다시 scheduling 한다
↔ interrupt는 동작후 복귀 지점으로 회귀

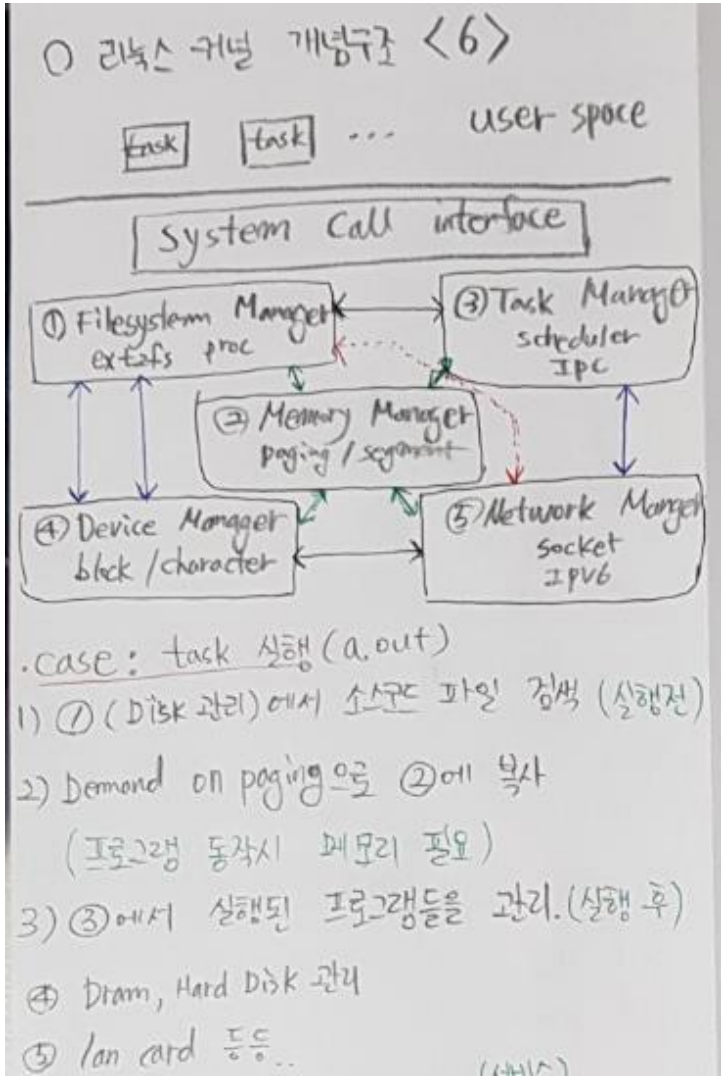
1. 리눅스 커널 내부 구조 2 (동작 비유 3 ~ 5)



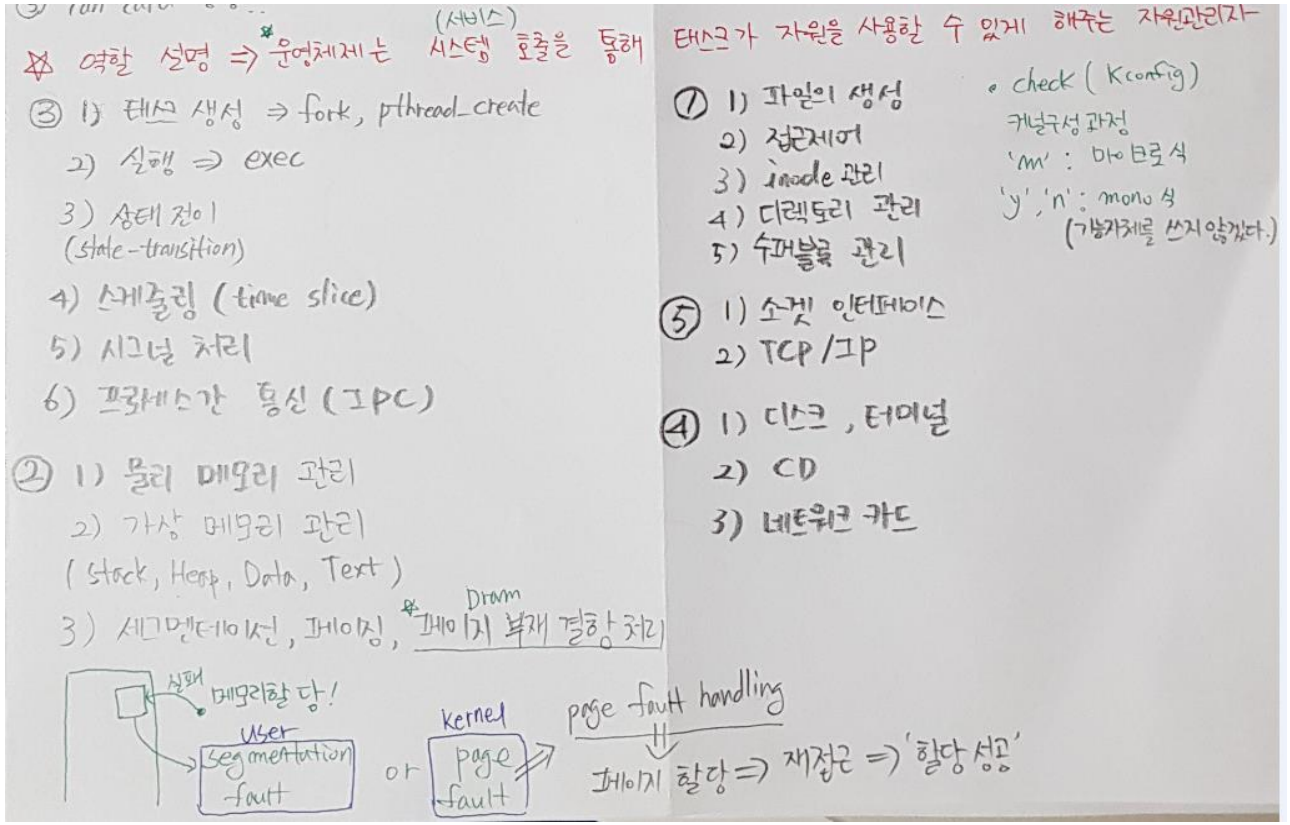


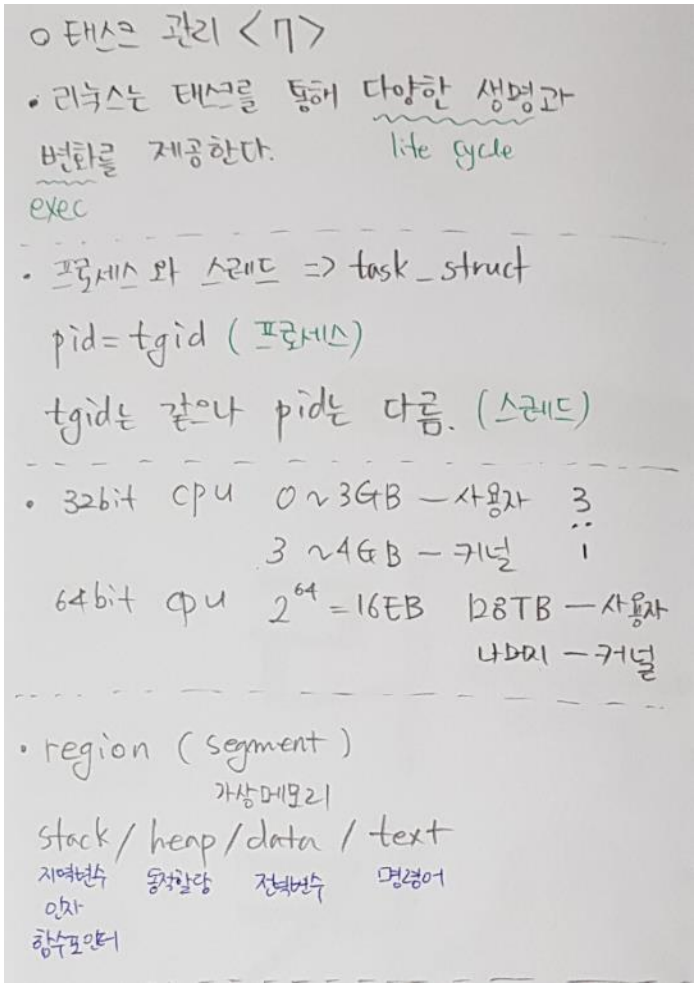
1. 리눅스 커널 내부 구조 3 (Linux kernel & 개념 구조)





1. 리눅스 커널 내부 구조 4 (Linux kernel 역할 & 태스크 관리)





1. 리눅스 커널 내부 구조 5

Kernel/ linux

- arch
-> cpu 모델에 따라서 바뀌어야 할 코드 (context switching code)

- bootloader
-> 하드웨어적 제어

- mm
-> 페이징 수행하는 연산 레지스터

- blackfin
-> 오디오 관련 제어

- openrisc
-> 나만의 cpu설계가 가능해진다.

- x86
->intel cpu

- c6x
-> TI, DSP

- ia64
-> graphic card 설계를 위한 공부

- kernel
-> 커널과 관련된 software

- drivers
-> device driver와 관련된 정보
(pci , armba, fmc(초고속 무선 통신 interface), gpio, gpu, power block, bluetooth, tty(터미널), mtd (nand flash), regulator (정전압) , iio(오실로스코프), dma(메모리에 직접 접근), w1(무선), phy(유선) , net)

- net
-> hamradio (무전 통신), can (통신), irda (적외선)

- wireless
-> ti
- Avr32
-> cortex A

- Nios2
-> 알테라 fpga(인텔 인수), x86에서 밀고 있는 architecture

- arm (하위 호환을 하지 않음 – 종류가 많은 이유)
- Exynos (삼성), omap (TI), zynq (자이링스)
- s3c24xx (삼성)
- keystone (TI DSP), 2 (레이더용)
- ipc18xx, ipc 32xx (NXP) (차량용)
- bcm (라즈베리파이)
- davinci (TI) (블랙박스, 스마트 tv, CCTV)
- stm32 (차량용)
- tegra (NVIDIA)

1. 리눅스 커널 내부 구조 5

Kernel/ linux

- include, lib
-> library

- block
-> block driver device에 관한 것만 있음 (block은 규모가 크다)
-> ssd같은 것을 제어하는 데에 필요한 알고리즘이 있다.

- init
-> 커널의 시작 코드 (vi –t start_kernel -> 9 -> kernel 초기화)

- usr
-> user와 관계된 정보

- Documentation
-> kerne과 관계된 문서들

- ipc
-> 세마포어, 스핀락, mutex

- scripts
-> 커널 구동용 스크립트

- virt
-> cloud service를 위한 기술

- microblaze
-> fpga에서 구동되는 가상의 cpu

- Sparc
-> 슈퍼 컴퓨터용 cpu

- Powerpc
-> 자동차

- mm (메모리 매니지먼트)
-> 가상 메모리, 물리 메모리 관련된 것들이 있다.

- firmware
-> 통신을 하면서 주고 받기 위함.

- net
-> network

- sound
-> VOLT, ALSA driver

- video
-> 영상 스트리밍 (V4L2)

- crypto(암호화, 복호화 알고리즘) , security, certs

- Fs (파일 시스템)
-> open, read, write 와 관련된 것들

1. 리눅스 커널 내부 구조 5

Kernel/ linux

- 실행파일 정보

-> readelf –h a.out
-> elf = 리눅스 실행 파일 포맷
-> dwarf = 디버깅 파일 포맷
-> pe = 윈도우 실행 파일 포맷