

공개SW 역량프라자

www.oss.kr



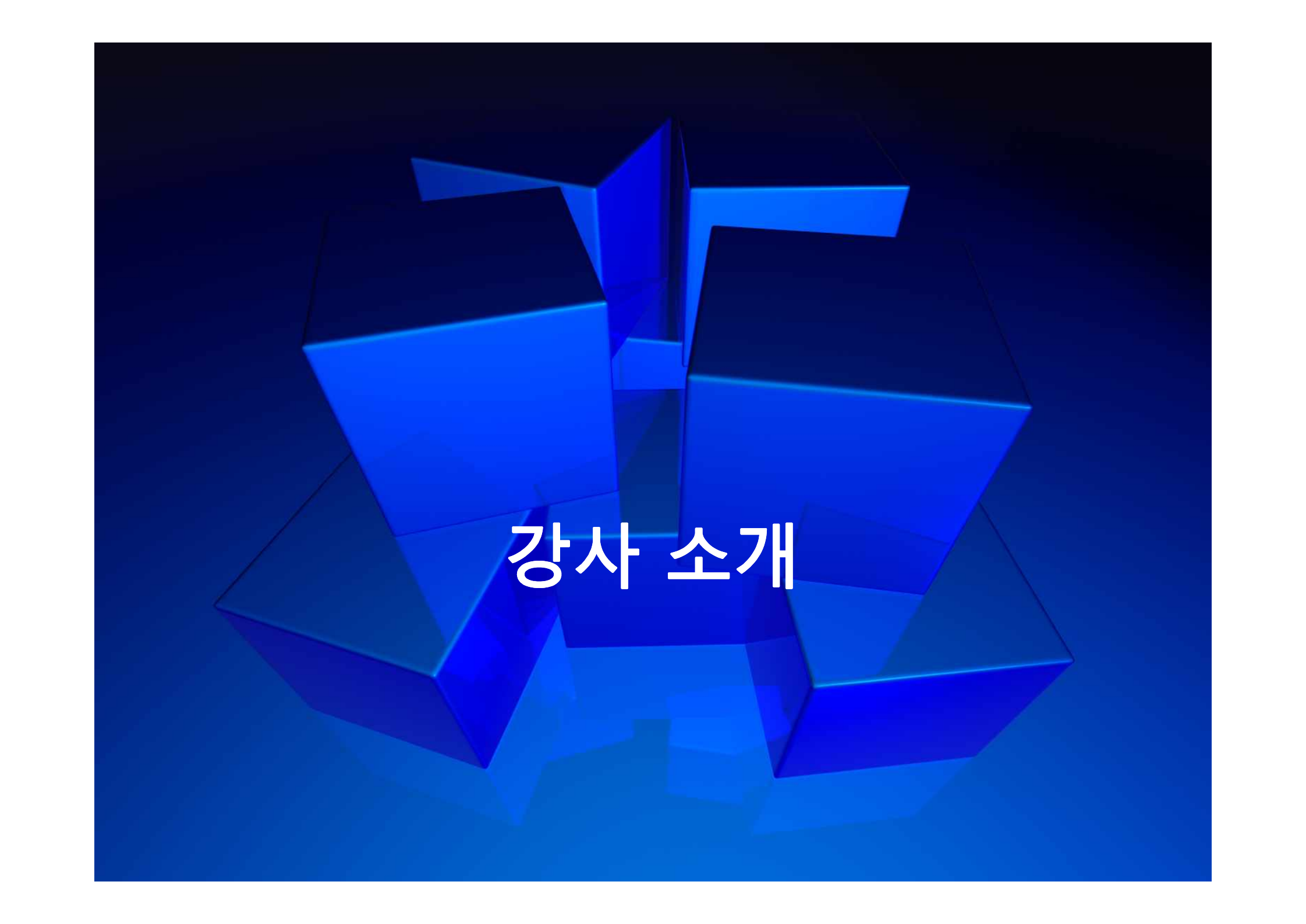
Android is not OS. It's Platform!

2011-05-19



유명환 (뽀뽀강사)

✉ funfun.yoo@gmail.com

The background of the slide features an abstract composition of several 3D rectangular blocks in various shades of blue. These blocks are arranged in a complex, overlapping manner, creating a sense of depth and geometric complexity. The lighting is soft, highlighting the edges and faces of the blocks against a dark, gradient blue background.

강사 소개

강사 소개



(뽀뽀강사) 유 명 환

✉ funfun.yoo@gmail.com

🐦 @funfunyoo

- 충남대, 정보통신공학석사
- 한국전자통신연구원(ETRI) 및 해동정보통신 등 중소기업 근무
- 2004 : 대한민국 창업대전, 중소기업청장상 수상
- 2005 ~ 2008 : (주)넷플러그, 대표이사 역임
- 현재, FUN-MARKET (www.fun-market.co.kr) 대표
- 정보통신연구진흥원, 주간기술동향집필위원 선정 (2005)
- 정보통신연구진흥원, 과제평가위원 선정 (2005)
- 舊 정보통신부, IT 멘토 임명 (2007)
- 식품의약품안전청, 홈헬스케어 의료기기 전문위원 선정 (2009)
- 한국과학기술정보연구원, 과학기술정보협의회 위원 선정 (2009)
- 삼성전자, 삼성SDS, 삼성SDS멀티캠퍼스, LG전자, LG CNS,
한국HP, 비트컴퓨터 등 다수 기관 교육 진행
- 한국 최초로 터널 내 무선 랜 기반 음성 방송 시스템 개발, 수락산 터널 외 전국 설치 (2006)
- 한국 최초로 무선 USB 기반 플랫폼 개발 (2006), 2007 美 CES 전시회에서 공개됨
- 실시간 소변 성분 분석기를 이용한 u-Health 솔루션 및 서비스 개발 (2008 ~ 2009)
- 현재, Wi-Fi/Bluetooth 기반 솔루션 및 안드로이드 TV 리모컨 개발 컨설팅 중

강사 소개



Java Power Tool
저자 John Ferguson Smart
역자 김경영, 고영래, 최성민, 조창식
O'Reilly Media / 988 페이지
2009년 12월 출간 예정



JavaScript: the Missing Manual
저자 David Sawyer McFarland
역자 김태준
O'Reilly Media / 560 페이지
2009년 12월 출간 예정



Maven: the Definitive Guide
저자 Sonatype Company
역자 소프트웨어 인 라이프
O'Reilly Media / 480 페이지
2010년 1월 출간 예정



SCJP Sun Certified Programmer for Java 6 Exam 310-065
저자 David McFarland
역자 김태준
McGraw-Hill / 870 페이지
2010년 1월 출간 예정



JBoss in Action
저자 Javid Jamae, Peter Johnson
역자 김태준
Manning Publication / 500 페이지
2010년 출간 예정



Wicket in Action
저자 Marjyn Dassen, Eelco Hillemius
역자 황재선
Manning Publication / 450 페이지
2010년 출간 예정

세상에서 가장 이해하기 쉬운 임베디드 리눅스 서적!



이 책은 임베디드 리눅스를 처음 접하는 개발자들이 가장 빠른 시간 내에 임베디드 리눅스 개발에 필요한 지식들을 습득하는데 목적을 두고 있다. 임베디드 리눅스 개발을 위해 필요한 마이크로프로세서(CPU)와 운영체제(OS)에 대한 분석부터 임베디드 리눅스 기반의 디바이스 드라이버 프로그래밍까지, 임베디드 리눅스 개발에 필요한 전반적인 내용들을 저자의 풍부한 개발 및 강의 경험을 토대로 최대한 쉬운 표현으로 작성되어 있어 누구나 쉽고 빠르게 관련 지식을 습득할 수 있다.

다들 아시다시피 임베디드 리눅스 시작과는 달리 Windows에서도 임베디드 리눅스 개발이 가능하도록 Cygwin 기반의 개발 환경을 구축하는 방법부터, GUI 기반의 IDE 툴로 현재 각광을 받고 있는 이클립스(Eclipse) 툴을 기반으로 임베디드 리눅스 어플리케이션을 개발하고 이를 실제 타겟 보드에 연동하여 개발하는 내용까지 다루고 있어, 복잡한 명령어와 불편한 사용자 환경 때문에 임베디드 리눅스 개발이 어려웠던 분들께 획기적인 도움이 될 것이다.

이 책의 주요 내용

임베디드 시스템 개발 방법론
마이크로프로세서(CPU), 운영체제(OS), 개발 툴(Tool) 관점에서의 임베디드 시스템 개발 방법론 제시

국내 임베디드 마이크로프로세서(CPU) 및 운영체제(OS) 분석
마이크로프로세서(CPU) 내부 구조 및 동작 원리 분석
RTOS (Real-Time OS)와 Non-RTOS의 비교 분석을 통한 운영체제(OS) 원리 분석

임베디드 리눅스 개발 환경 구축: Windows 및 Linux 기반
Cygwin 기반의 Windows에서의 임베디드 리눅스 개발 환경 구축 방법 / Linux 기반의 임베디드 리눅스 개발 환경 구축 방법

Makefile 문법 및 개발자 전용 라이브러리 작성법
Makefile 구성 요소 및 문법 / 개발자 전용 Static Library, Shared Library 작성 방법

리눅스 커널 모듈 프로그래밍
Firmware, Kernel Module, Device Driver 간 차이점 분석 / Kernel Module 프로그래밍 문법
커널 심볼(Symbol) 공유 및 Startup Parameter 제어 수록

리눅스 디바이스 드라이버 프로그래밍
디바이스 드라이버 동작 원리 분석 / 리눅스 커널 2.4 버전과 2.6 버전 간 디바이스 드라이버 차이점 분석
디바이스 드라이버 문법 / 디바이스 드라이버와 사용자 어플리케이션 간 연동 방법 분석

이클립스(Eclipse) 기반의 임베디드 리눅스 프로그래밍
오픈 소스 기반의 이클립스(Eclipse)를 통한 임베디드 리눅스 프로그래밍
이클립스와 타겟 보드 간 연동 방법 분석

₩ 22,000원



지앤지
지앤지

지앤지
지앤지

뻔뻔하게 배우는 임베디드 리눅스

유명한 저자

Fun+Fun

Embedded

Linux



세상에서 가장 이해하기 쉬운 임베디드 리눅스



저자 소개

유명한 funfun.yoo@gmail.com

재이(FUN)었다는 이유 하나만으로 어플리케이션 개발자에서 임베디드 개발자로 전향하여, 중소기업 프로젝트부터 국가연구소 프로젝트까지 다양한 임베디드 프로젝트를 진행한 베테랑이다.

아무 것도 모르는 상태에서 임베디드 개발자로 거듭나기까지 겪었던 수많은 경험을 때문에 그 누구보다 처음 임베디드 기술을 접하는 개발자들의 마음을 헤아릴 줄 알기에 그들에게 도움이 되고자 2005년부터 현재까지 삼성전자, 삼성 SDS, LG전자, LG CNS, 한국반도체, 현대중공업, 한국전자통신연구원(ETRI), 국방과학연구소(ADD), 한국소프트웨어진흥원(KIPA), 한국정보산업연합회(FKII), KAST EMEDEC, 삼성 SDS 멀티캠퍼스, 비드캠퍼스 등 다양한 기업 및 기관들에서 임베디드 관련 강의를 진행해오고 있으며, 2009년부터 "뻔뻔강사"라는 별명으로 인터넷 카페 등에 임베디드 관련 강좌를 올리면서 수많은 개발자들의 호응을 받고 있다.

"FUN = Funny(재이) + Useful(유익하다) + New(새로운)"
저서 전과의 달인으로 거듭나기 위해 저자는 뻔뻔강사라는 별명달게, 현재 인터넷 사이트 FUN-MARKET(www.fun-market.co.kr)을 통해 기존에 나왔던 교재들과는 사뭇 다른 교재와 강좌, 교육용 키트 등을 보급하고 있으며, 최근에는 로보카 구글 안드로이드(Android)를 주제로 저술하고 있다.

jmonstorty.com

최종: 제2판 2010.12.15

www.funfunstudy.co.kr

뻔뻔강사 교육 포털 사이트

Index



1

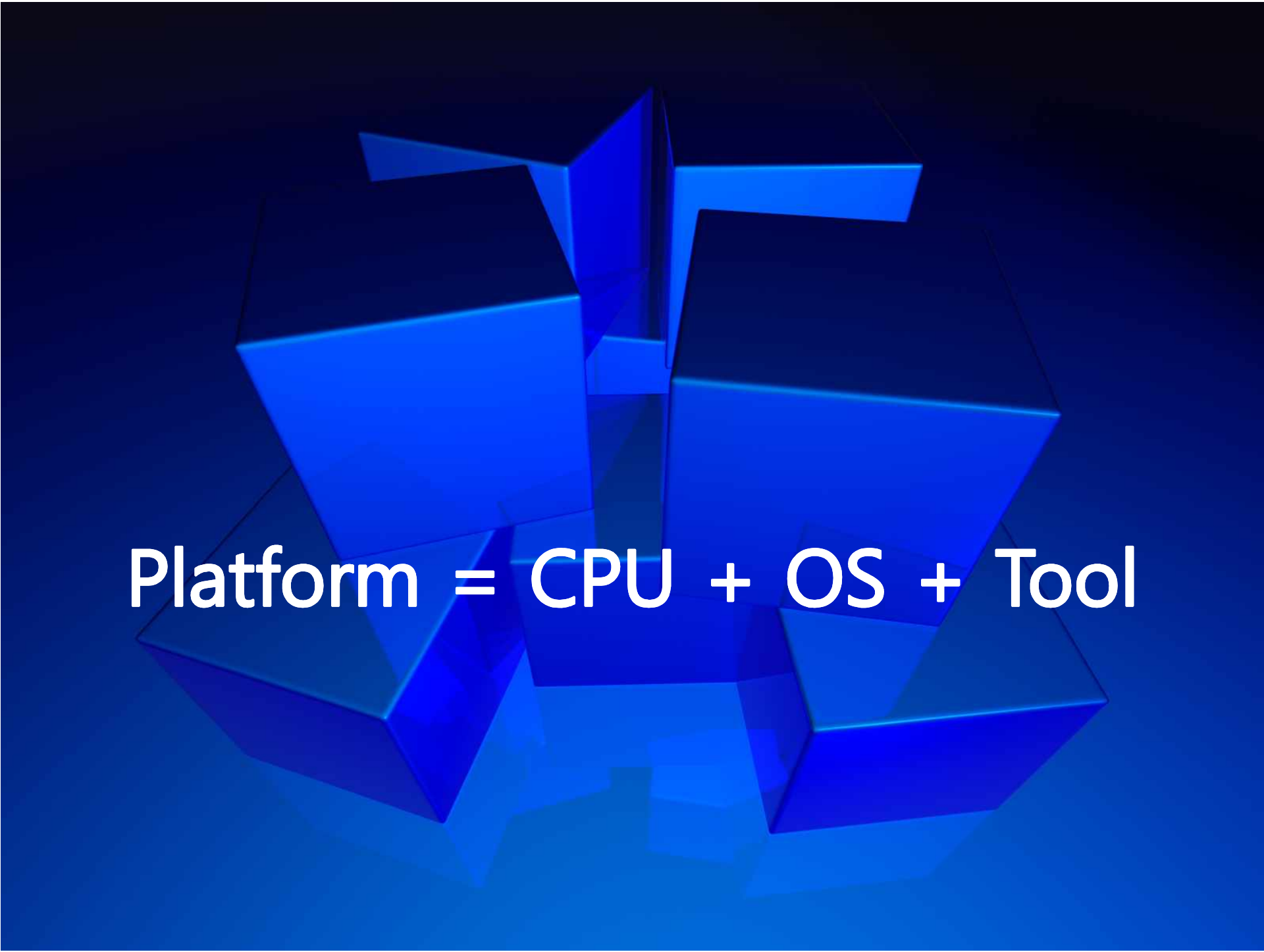
Platform = CPU + OS + Tool

2

리눅스 계층별 내부 구조 분석

3

안드로이드 vs 리눅스

The background of the slide features a dark blue gradient with several translucent, three-dimensional blue geometric shapes. These shapes, which include cubes and rectangular prisms, are arranged in a complex, overlapping pattern. Some shapes are positioned in the foreground, while others are behind them, creating a sense of depth. The lighting appears to come from the upper left, casting soft shadows and highlighting the edges of the shapes.

Platform = CPU + OS + Tool

임베디드 플랫폼

Embedded Platform

CPU

OS

Tool

안드로이드 플랫폼

Android Platform

ARM

Linux

Ubuntu,
Eclipse

왜 플랫폼이 중요한가?

Embedded = Cost Saving

비용 = (눈에 보이는 비용) + (눈에 보이지 않는 비용)

눈에 보이는 비용 = 부품 구매비 + 생산비 + ...

눈에 보이지 않는 비용 = 인건비 + ...

안드로이드 플랫폼



스타워즈 : 모바일 플랫폼 전쟁

Android

Key Applications

API Library Set

Middleware

OS (Kernel) : Linux 2.6

Hardware Reference Design

안드로이드 = 모바일 기기 개발을 위한 오픈 소스 플랫폼

안드로이드 플랫폼

Android = XML + Java + Linux

XML, Java, Linux = 이미 잘 알려져 있고 많이 사용되고 있는 기술

잠재된 개발자가 많다 -> 어플리케이션이 풍부해질 것이다 -> 플랫폼을 장악할 것이다

The background of the slide features a dark blue gradient. Overlaid on this are several translucent, three-dimensional blue geometric shapes, including cubes and rectangular prisms, which are arranged in a complex, overlapping pattern. The lighting creates highlights and shadows on the faces of these shapes, giving them a sense of depth and volume.

리눅스 계층별 내부 구조 분석

리눅스 계층별 내부 구조

User Level

Application

(System) Library

System Call Hacking

Software Interrupt

Kernel Level

System Call

Virtual File System

General File System

Device File System

Network File System

Buffer Cache

TCP/IP Protocol Stack

Block Device Driver

Character Device Driver

Network Device Driver

Device Interface

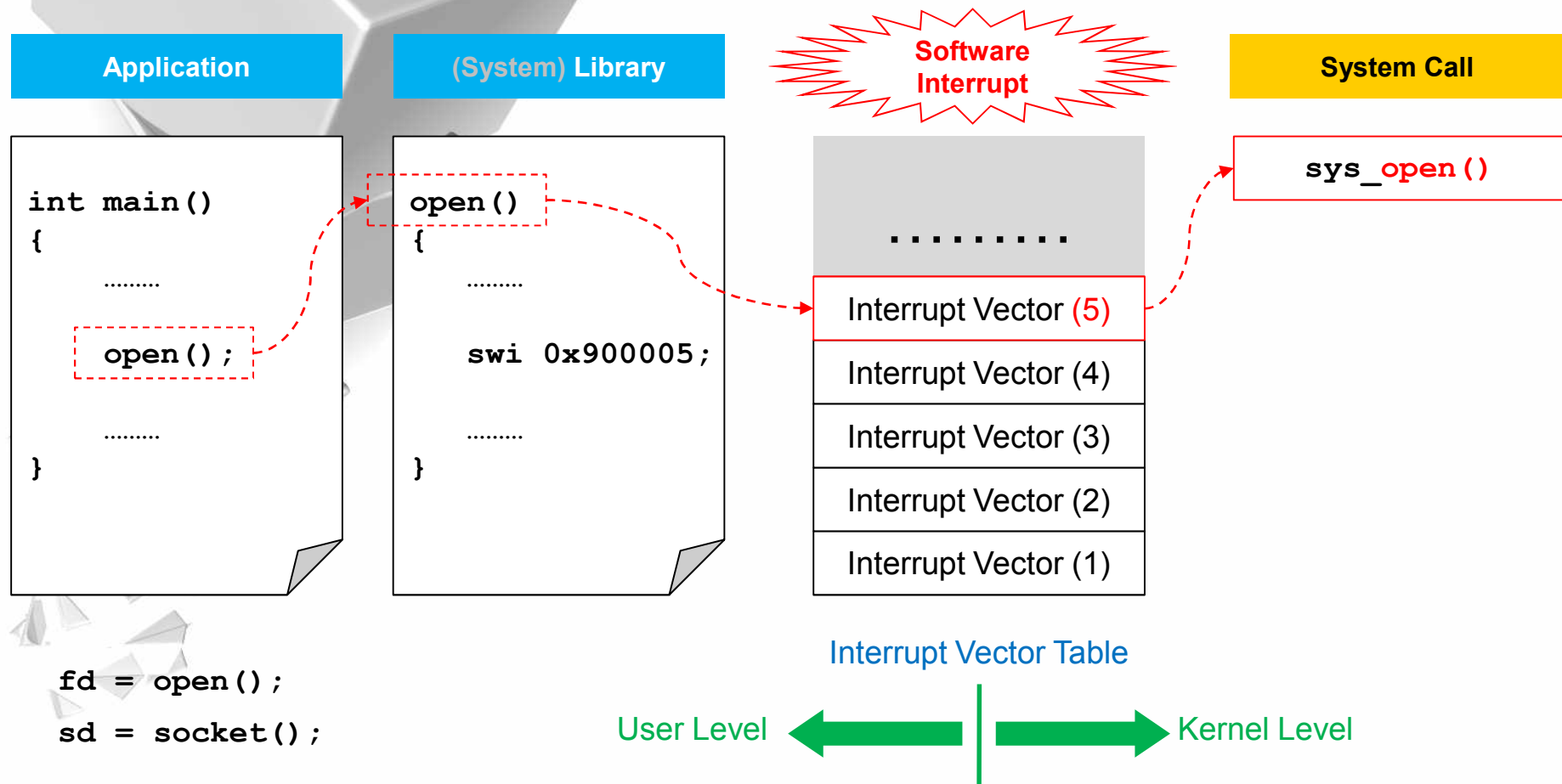
Block Device

Character Device

Network Device



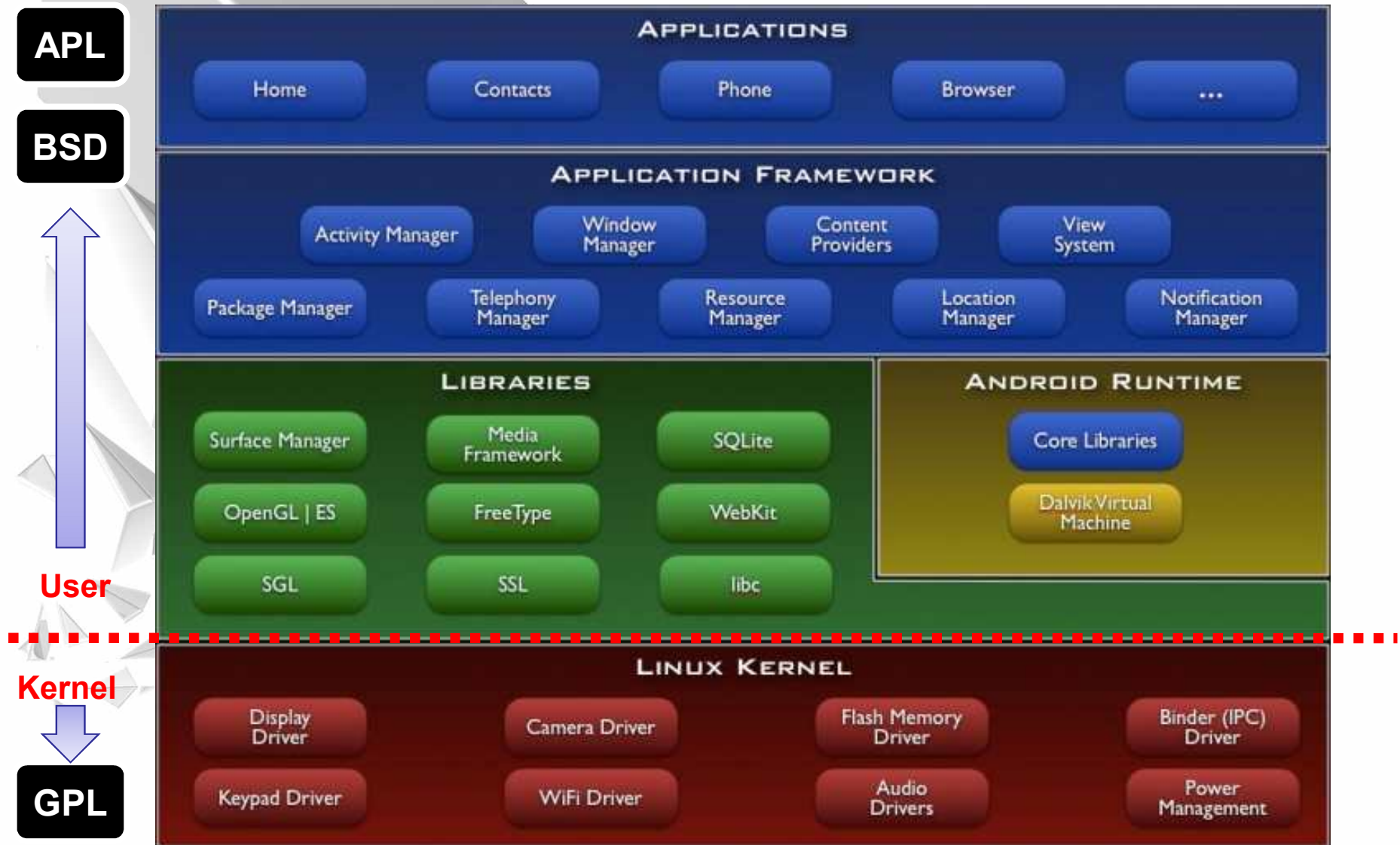
리눅스 시스템 콜 함수 호출 구조



The background of the slide features several overlapping, three-dimensional blue geometric shapes, primarily cubes and rectangular prisms, arranged in a complex, abstract composition. The shapes are rendered with a gradient of blue, from a deep navy blue to a lighter, vibrant blue, creating a sense of depth and volume. The lighting appears to come from the upper left, casting soft shadows and highlighting the edges of the shapes. The overall effect is a modern, tech-oriented aesthetic.

안드로이드 vs 리눅스

안드로이드 계층별 내부 구조

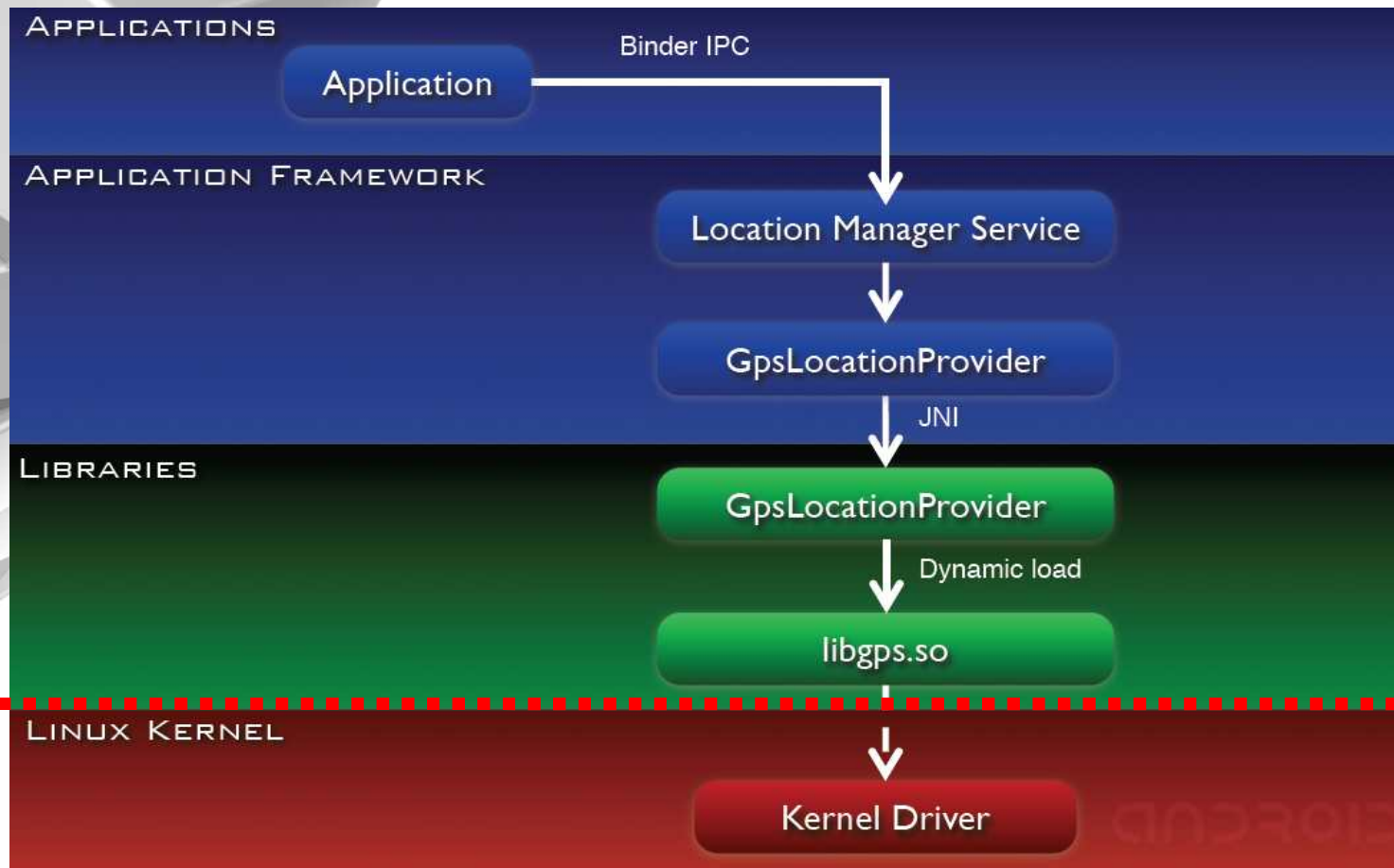


안드로이드 계층별 내부 구조 : 사례

APL

BSD

Example : Location Manager (GPS)



GPL

HAL : Hardware Abstraction Layer

JNI 를 사용하는 이유

- 속도 문제
- Java 에서 하드웨어 제어
- Java 에서 지원되지 않는 특정 운영체제(OS) 서비스와 기능 연동

기존 리눅스와 안드로이드의 드라이버 차이점

- 기존 임베디드 리눅스는 장치 드라이버들이 커널 영역(Kernel Level)에서 동작
- 반면 안드로이드는 장치 드라이버들이 유저 영역(User Level)에서 동작

-> 커널 영역을 제외한 HAL 을 의미!!!

*-> MMU 등을 참조하는 등의 추가적인 작업이 필요하여 조금 불편할 수 있지만,
자신에게 할당되어 있는 메모리 번지만 접근할 수 있기 때문에 보다 더 안전!!!*

안드로이드에서 드라이버들을 유저 영역에 넣은 이유?

- 라이선스 문제
- 안정성
- 플러그인 방식으로 드라이버 배포 가능



HAL : Hardware Abstraction Layer

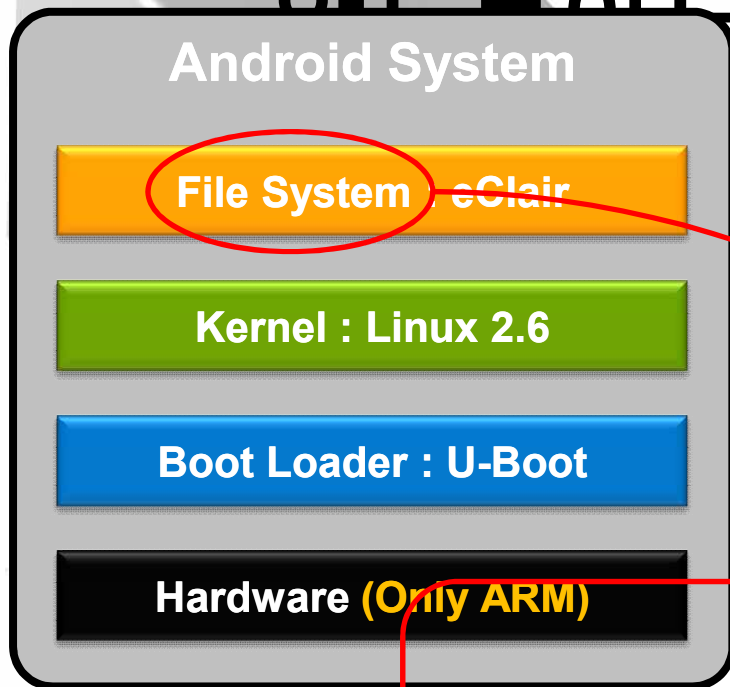
안드로이드에서 HAL 과 같은 미들웨어를 만든 이유?

- 커널 드라이버는 GPL의 라이선스에 적용을 받는다는 문제
- 안드로이드의 모든 라이브러리들이 표준화되어 있지 않다는 문제
- 안드로이드만의 독특한 기능(Binder(IPC), Power Management 등)을 부여하기 위해

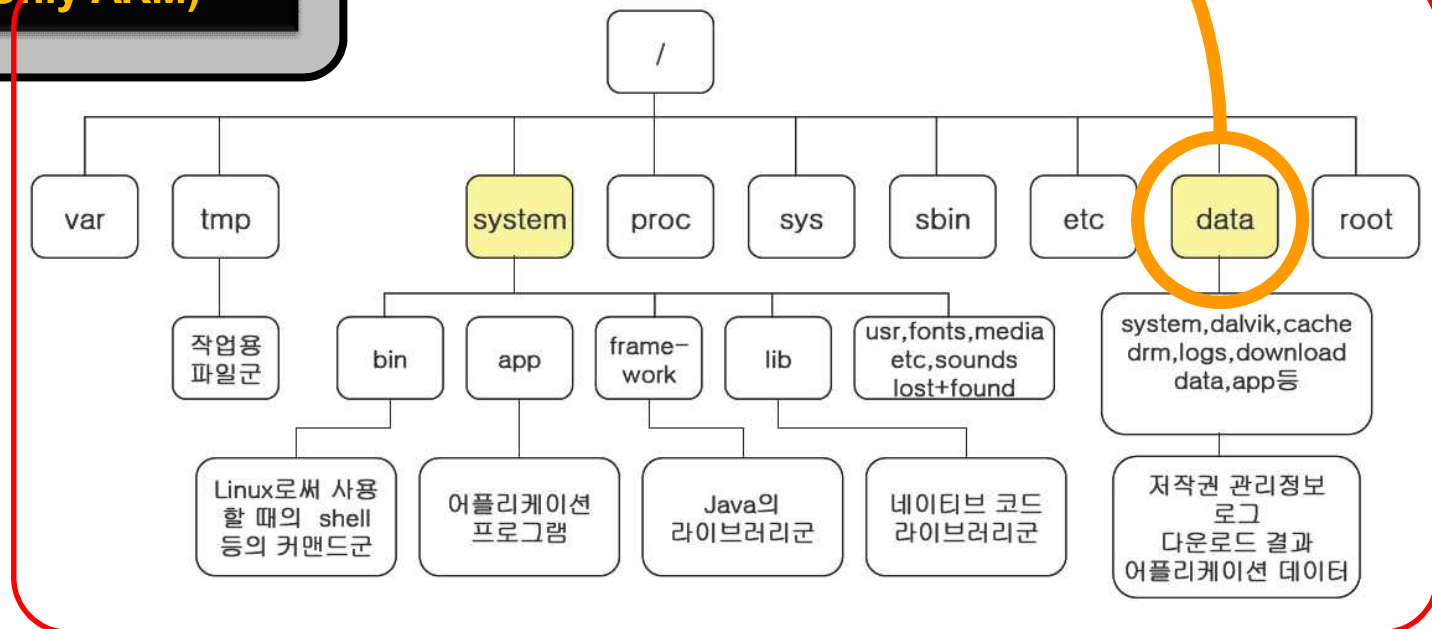
-> 기존 임베디드 리눅스와 달리 안드로이드 용 리눅스 커널에서 추가된 커널 구성 요소들

: *Alarm, Low Memory Killer, Shared Memory Driver(Ashmem), Kernel Debugger,
Binder, Power Management, Logger*

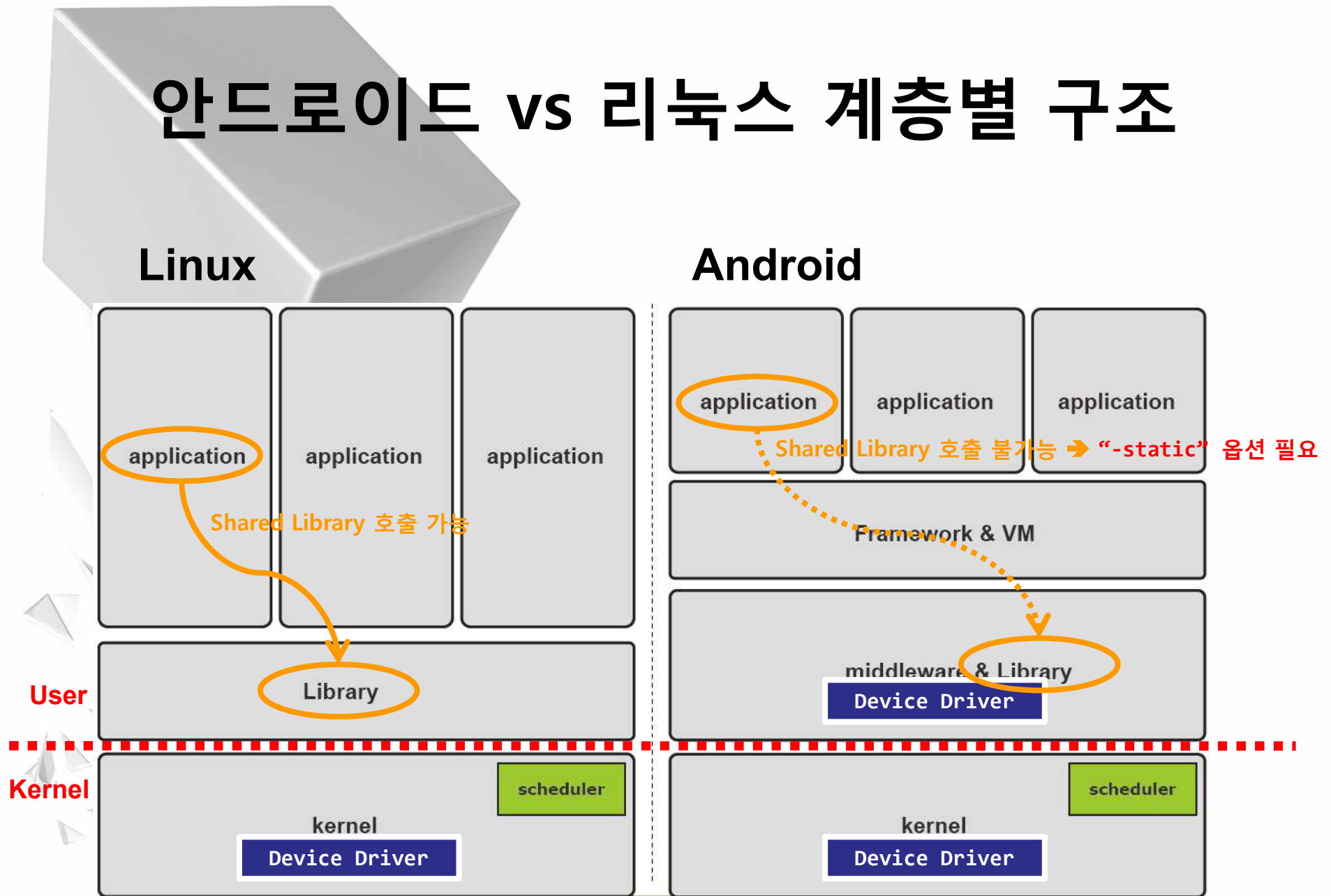
안드로이드 파일 시스템 구조



C/C++ 안드로이드 APP 은
단말기(혹은 Eclipse 에뮬레이터)의
/data 디렉토리에 복사하여 실행!!!



안드로이드 vs 리눅스 계층별 구조



SUN's JVM vs Google's Dalvik

Java Virtual Machine

Sun JVM

Google Dalvik

Java APP

Java APP

Java APP

Java APP

Java APP

Java APP

Java APP 실행 시 호출

Dalvik

Dalvik

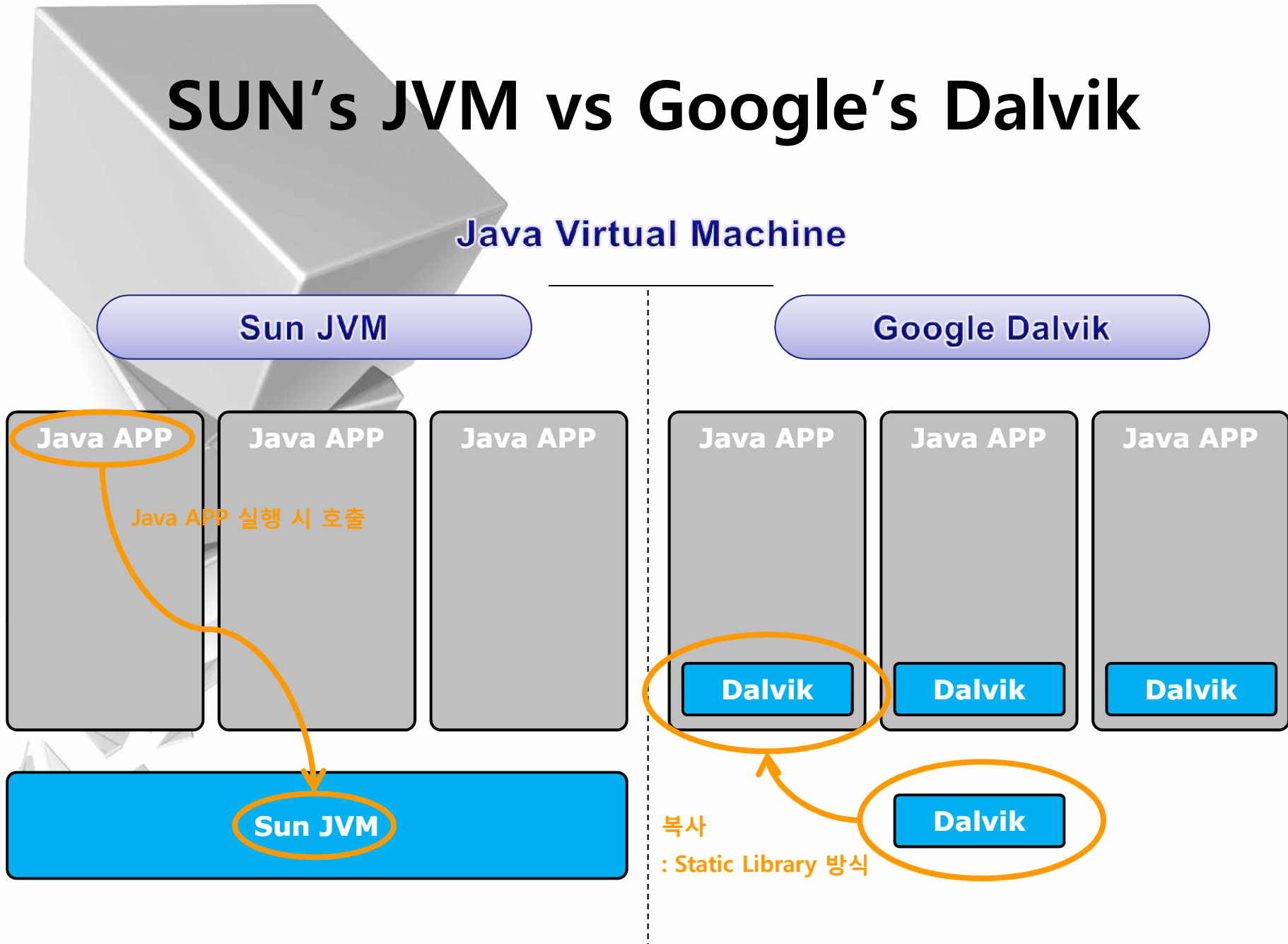
Dalvik

Sun JVM

Dalvik

복사

: Static Library 방식



안드로이드 앱에 **main** 이 없는 이유?

main 함수(메소드)가 없는 소스 코드는 없다!!!

Android Application Source

```
package com.funfun.Hello23v;

import android.app.Activity;
import android.os.Bundle;

public class Hello23v extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

RTOS Application Source

```
void task_start(void *data)
{
    .....
}

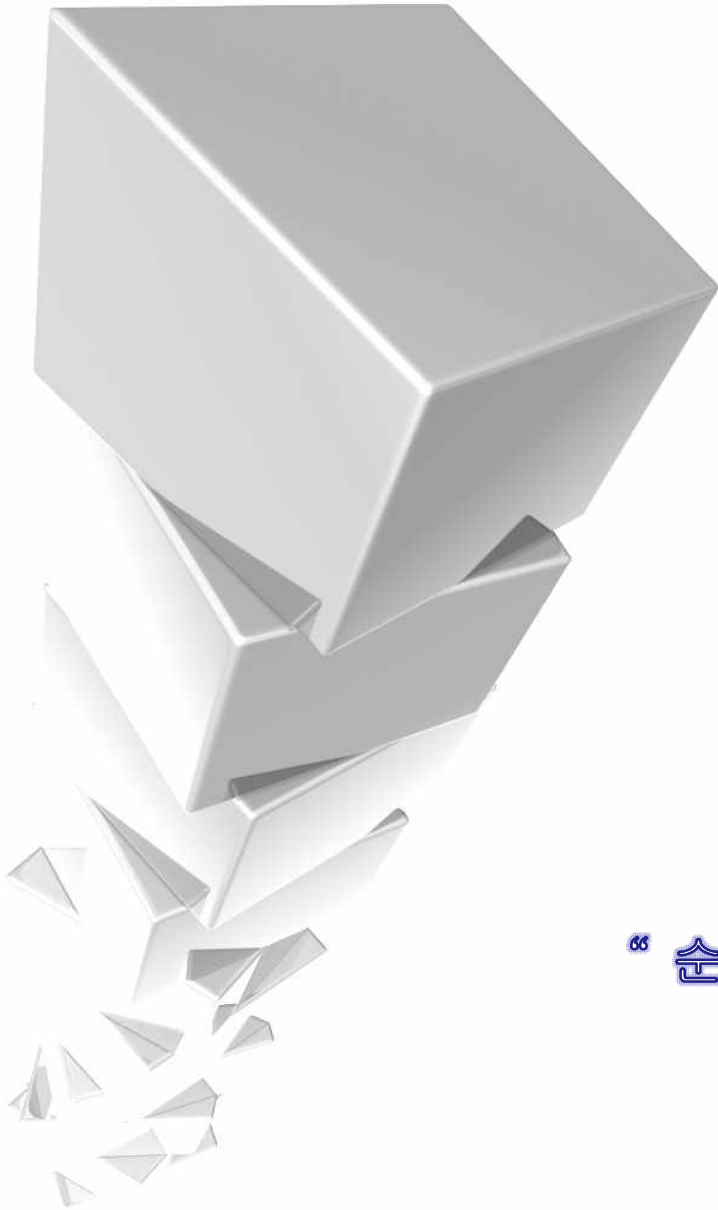
int main(void)
{
    OSInit();

    OSTaskCreate(task_start, (void *)0,
        (void *)&task_start_stk[OS_TASK_DEF_STK_SIZE - 1],
        0);

    OSStart();

    return 0;
}
```


Q & A



“ 순간의 □ □ □ 이 평생을 좌우한다! ”