**20210106 리서치 과제**

( iphone cpu architectures, bitcode, dSYM )

- 요약 -

**1. iphone cpu architectures**

운영체제 : iOS ~~버젼

CPU아키텍쳐 : arm64(최신), armv7s, armv7

코코아 플랫폼 용으로 빌드할 때, "Elements"를 통해 여러 CPU아키텍쳐(64비트, 32비트 등)에 맞는 모든 바이너리를 포함한 Fat Binary를 만들수 있음.

**2. bitcode**

앱 thinning 기술 3가지 중 1가지이다.

- App Slicing : 앱을 앱스토어에서 다운 받을 때, 디바이스에 맞는 실행코드와 리소스만 잘라서 빌드해서 주는 기술

- On Demand Resource(ODR) : 아직 필요하지 않은 리소스들을 앱에 미리 넣지말고 앱스토어에서 가지고있다가 필요할때 앱에 넣어주는 기술

- BitCode = IR : 기계어도, 사람이 이해하는 코드도 아닌 중간코드. App Slicing을 할 때에, 여러 플랫폼을 지원할 수 있는 상태의 코드가 BitCode.

LLVM이라는 녀석이 중간언어를 바이너리 코드로 바꿔준다.

Swift코드 -> [SwiftClang] -> 중간언어! IR! Bitcode! -> [LLVM] -> 바이너리 코드!

**3. dSYM**

컴파일시 결과소스와 원본소스의 매핑정보를 가진 파일.

다른 클라이언트의 크래쉬 리포트를 해석하기 위해서는 이 dSYM파일이 필요한데,

Archive시 생성되며 Archive 창 우측 하단에서 다운로드 가능.

단 예외적으로,

bitcode를 사용할 경우, 앱스토어에서 컴파일을 한번 거쳐서 배포하기 떄문에,

별도의 컴파일을 거친 코드가 보낸 크래쉬 리포트를 해석하기 위해서는

내가 Archive할때 생성한 dSYM이 아닌, 앱스토어가 컴파일했을 때 만들어진 dSYM이 필요하고,

그건 ITunes Connect 에서 다운가능.

* 스크랩 -

1. **iphone cpu architectures**

Cocoa 플랫폼 용으로 빌드할 때, Elements라는 친구를 사용하여 장치, 운영체제버전에 따른 빌드 분기가 가능.

즉, 둘 이상의 플랫폼에 대한 각각의 실행 코드를 포함하는 “유니버셜 바이너리” 또는 “Fat 바이너리”를 만들 수 있음.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

iOS 아키텍쳐는 3개가 있음.

Arm64 (64bit) (최신)

아이폰 : 5S, 6, 6S, SE, 7

아이패드 : Air, Air2, Pro

칩 : A7~

Armv7s (Swift라고도 함. 근데 언어 Swift랑 다른거임 오해 ㄴㄴ)

아이폰 : 5, 5C,

아이패드 4

칩 : A6, A6X

Armv7 (32bit)

위보다 일찍 나온 모든것들

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

Armv6 CPU를 사용하며, 현재의 iOS SDK들과 Elements가 지원하지 않는 아주오래된 (2009년 출시) 모델이 있을 수 있음….

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

프로젝트를 구축할 때 여러 아키텍처를 한번에 선택할 수 있으며 Default는 가장 최신 아키텍처 집합(arm64를 사용하는 것들)을 포함하므로 가장 진보적인 설정임.

ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

현재 애플 매킨토시 제품에는 인텔 CPU가 탑재돼 있다. 그러나 아이폰/아이패드 등 모바일 제품에는 ARM 기반의 애플 자체 설계 CPU가 쓰인다. 이는 완전히 서로 다른 2개의 제품군이 각자 2개의 운영체제와 2종류의 앱을 사용해야 한다는 것을 의미한다.

ㅡㅡㅡㅡㅡ

iOS = internetwork Operation System OS, 즉 운영체제임. 윈도우 느낌

ARM = CPU아키텍쳐임! 인텔 느낌

1. **bitcode**

앱 thinning의 3가지 기술 중 1가지

1. App Slicing : 앱에는 여러 아키텍쳐, 그래픽, 오디오, 디바이스에 따른 실행코드와 리소스가 있고, 앱스토어에서 다운 받을 때 앱스토어가 알아서 필요한 것만 잘라서 주는 것. (armv7, 192kb Audio, Metal High Quality, 1x iPad 처럼)
2. On Demand Resource (ODR)

아직 필요하지 않은 리소스들을 앱에 미리 넣지말고 앱스토어가 가지고 있다가 필요하면 앱에 넣는 기술..

스토어에 저장하는 리소스들을 ODR이라고 부름.

스위프트 빌드 세팅에서 끌 수도 있음.

1. Bitcode = IR(Intermediate Reprosentation) = 중간언어

기계어도, 사람이 이해하는 코드도 아닌 중간단계의 코드.

애플은 다운로드 직전에 비트코드 단계에서 디바이스의 플랫폼과 운영체제에 맞는 바이너리 코드로 컴파일하여 제공함.

LLVM(Low Level Virtual Machine)라는 녀석이 고급언어를 파싱하여 비트코드를 만들고, 백엔드의 빌드 정보를 기반으로 비트코드를 기계어로 컴파일하기도함.

이 중간단계 덕분에 애플이 최신 컴파일러에 맞게 컴파일 옵션을 추가하면, 프로그래머가 소스를 손보지않아도, 최신 컴파일러용으로 자동으로 앱을 컴파일하게됨.

App Slicing 과 함께 사용되는 중.

1. **dSYM**

이상한 숫자들을 해석할 수 있게 해주는 파일.

컴파일러가 컴파일할 때, 기계어를 다시 소스코드 라인으로 매핑하는 정보를 가진 파일.

앱을 Archive 할때 생성.

앱스토어에서 다운 전에 중간코드를 컴파일 할 때도 생성됨.

* 내가 올린 소스와 앱스토어가 최적화한 소스는 다르기 때문에 둘의 dSYM도 다름. 심지어 dSYM는 UUID를 기반으로 묶여있기 때문에 폰마다 다 다름
* 때문에 앱스토어에서 크래쉬 리젝을 당할 경우, 받게되는 크래쉬 리포트는 이 크래쉬가 발생한 앱만의 dSYM이 필요