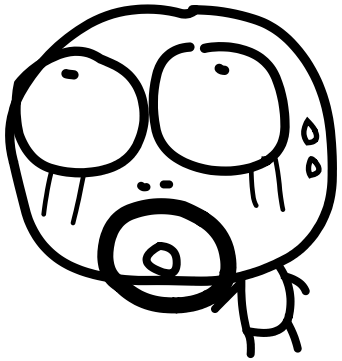
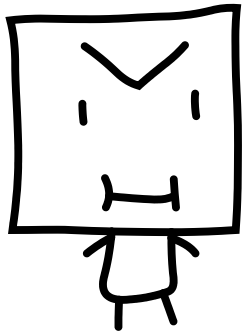


시스템 프로그래밍
오리엔테이션



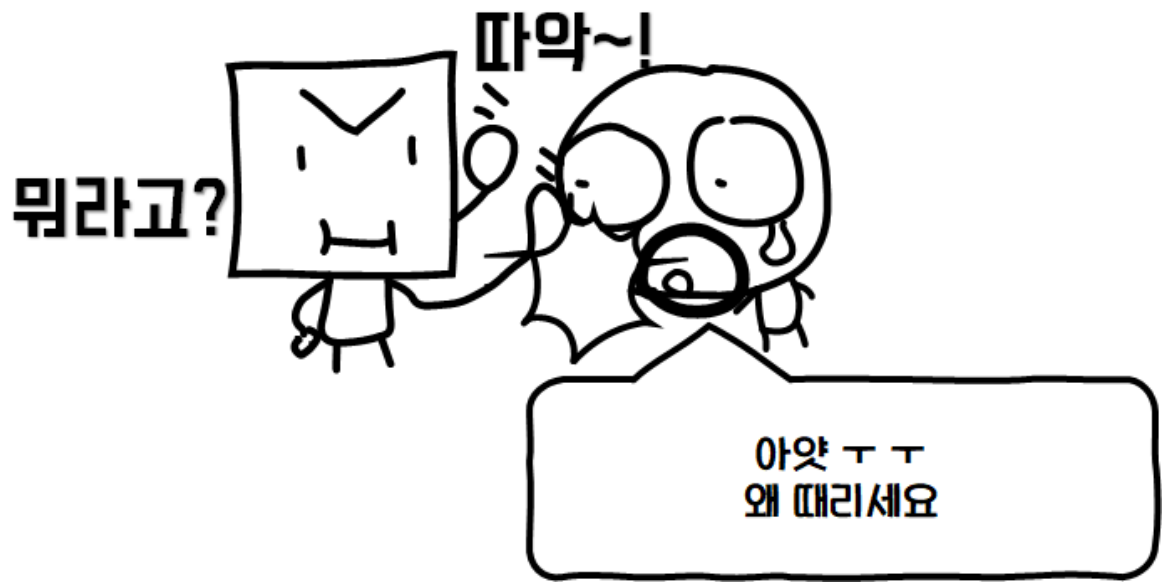
S-개발자 4기 2026-03-10(화)

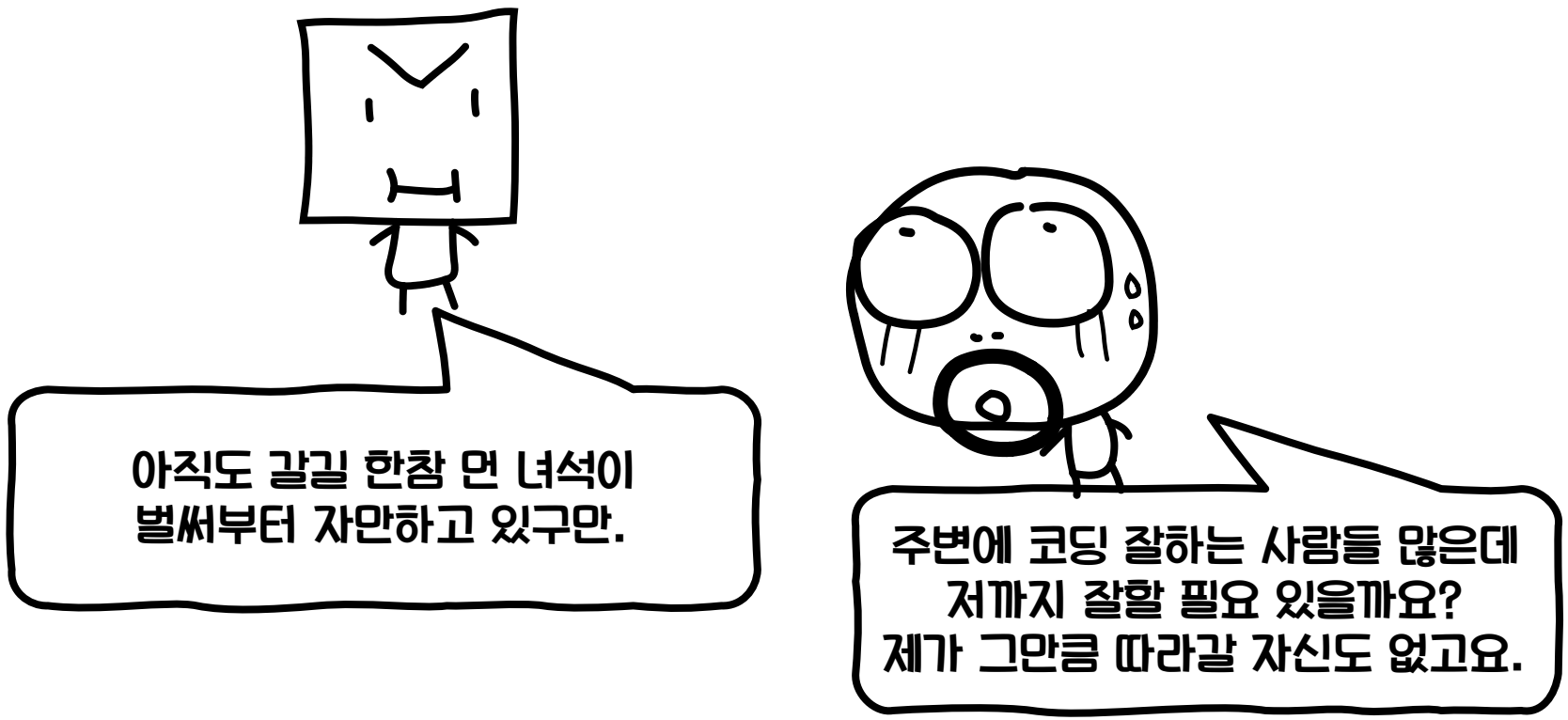
서울 송파구 동남로 130, 2층 제 4강의실

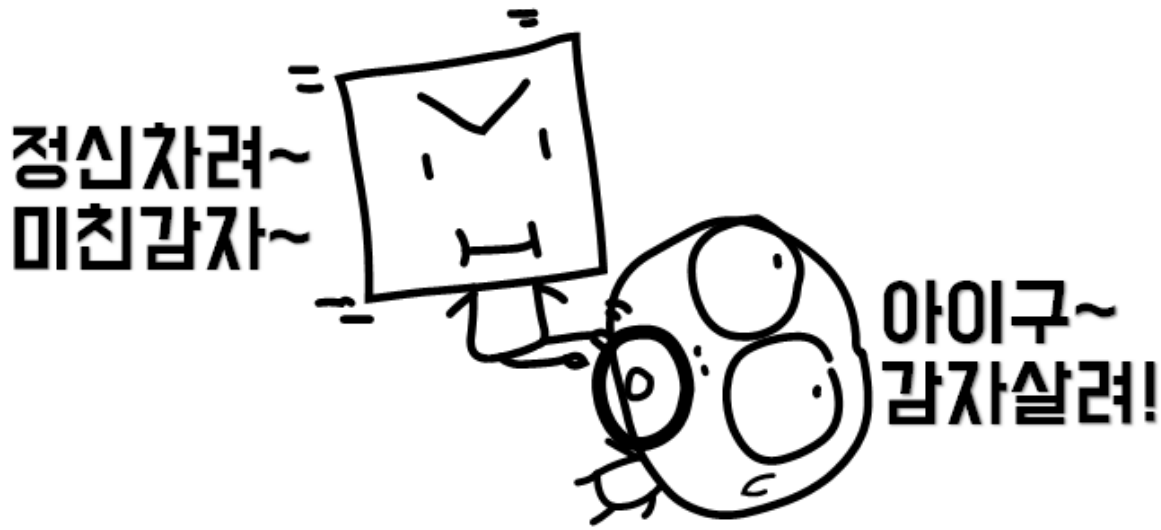
악성코드검거단 MALWARE ARREST TEAM 대표이사 전상현


미친강자!!!
코딩 잘하고 싶나?


아뇨 딱히...
저는 지금도 만족하고 있는데요!!!


따악~!
뭐라고?
아얏 ㅜㅜ
왜 때리세요


아직도 갈길 한참 먼 녀석이
벌써부터 자만하고 있구만.
주변에 코딩 잘하는 사람들 많은데
저까지 잘할 필요 있을까요?
제가 그만큼 따라갈 자신도 없고요.


정신차려~
미친감자~
아이구~
감자살려!


이건 먼~ 공부해야 하는 이유부터 찾아야 겠어.
미친감자 네 녀석이 코딩을 잘해야 하는 이유는 말이야...

AI 시대에 우리는 어떤 개발자가 되어야 할까?

1. AI가 사람이 하는 일의 대부분을 대체하는 것에 대해 여러분 자신도 도태될까봐 두려운가요?
2. 앞으로 어떤 기술을 익혀야 살아남을 수 있을지 막막한가요?
3. 그러다 보니 당장 오늘 뭘 해야 할지 막막해서 정작 아무것도 하지 못한 채 시간만 버리는 중인가요?

만약 그렇다면 저는 이렇게 말하고 싶습니다.

Just Do It.

무엇을?

여러분이 하고 싶은 것을.

그 이유와 목적은?

여러분도 미친감자처럼 이유와 목적을 아직 모르신다면 지금 바로 찾아봅시다.

그리고 그것에 미친 사람처럼 살아봅시다.

얼마나 미쳐야 하냐구요?

미친감자만큼요.

※ 여기서의 “미친”은 불광불급에서 따온 말입니다.

전상현
악성코드검거단㈜ 대표이사

전문 분야

- 3개 분야(보안/멀티미디어/게임) 풀스택 개발자
- 시스템 프로그래머
- 악성코드 분석엔진
- C/C++ Secure 코딩
- SW 아키텍처 설계

주요 경력

- 2022.01~현재 - 악성코드검거단㈜ 대표이사
- 2018.06~현재 - BoB 보안제품개발트랙 책임멘토
- 2016.08~2020.09 - ㈜시큐레터 1대 CTO 역임
- 2011.11~2016.02 - ㈜안랩 선임연구원
- 보안관련 특허 14건 출원(10건 등록)
- 보안 및 개발서적 3편 저술 및 출간

2023

2018

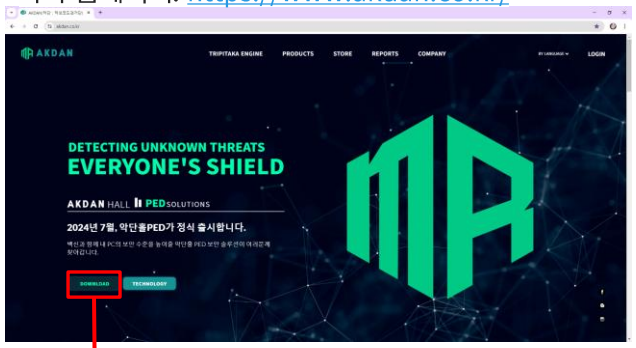
2016

성공한 사이버 공격의 91%는 이메일로부터 시작됩니다.

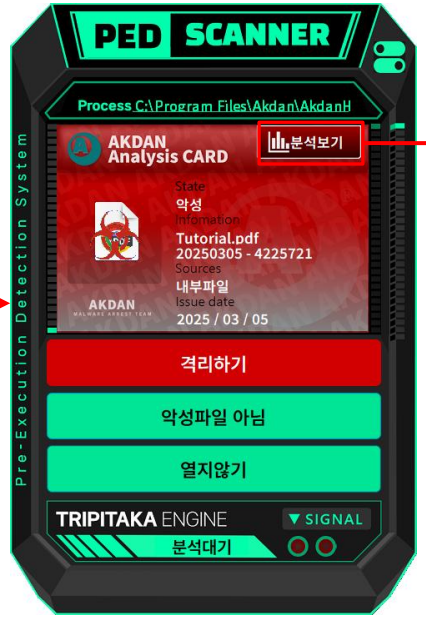
최근 3년간 발견된 악성파일 중 실행파일은 단 3%에 불과, 나머지는 문서파일(HTML, JS, PDF)이 차지합니다. (출처: Virustotal emerging 2024)

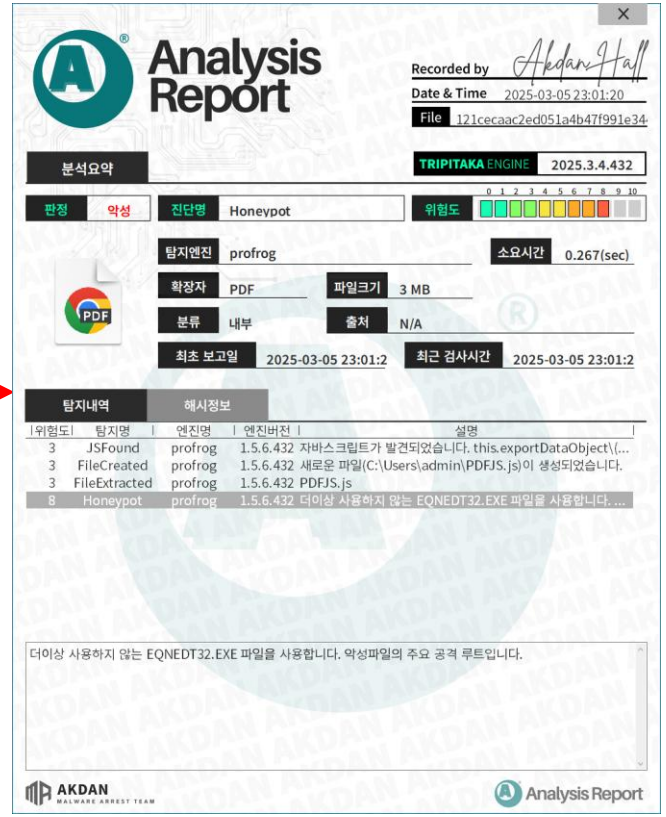
악단홀 PED는 이메일/메신저/웹을 통한 콘텐츠 매개형 악성코드 탐지하는 솔루션입니다.

자사 홈페이지: https://www.akdan.co.kr/



문서파일
실행



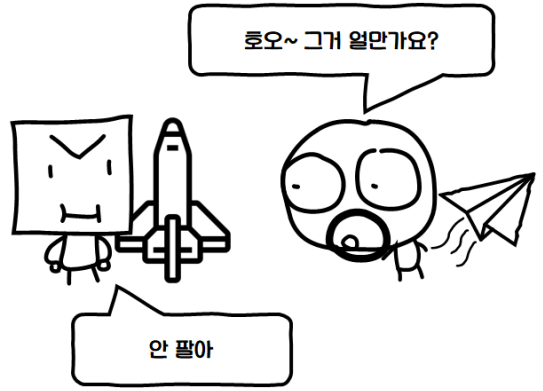


들어가기 전에 수업 개요

과정명	시스템 프로그래밍	강사명	전상현
강의시간	24시간 (4일 x 6시간)		
강의목표	컴퓨터 시스템을 이해한다. 나아가 다양한 시스템을 제어하는 C++ 프로그래밍 기술을 익힌다.		
평가방식	시험 50%, 실습 50%		

강의내용

시간(H)	제목	내용
1H	오리엔테이션	강사 소개 후 간단한 퀴즈와 함께 실행파일의 구조를 이해한다.
2H	변수형과 유니코드	시스템에 존재하는 모든 종류의 변수형을 알아본다. 다국어를 표현하는 문자체계인 유니코드를 이해한다.
3H	개발환경 구축하기	윈도우/리눅스(wsl)에 빌드할 수 있는 환경을 구성한다. 기본적인 개발툴 사용법을 익힌다.
2H	문자열 정복하기	C++과 친해지기 위한 코드를 작성해본다. VisualStudio 디버깅 기술을 배운다.
2H	C++ 컴파일러/링커/디버거 정복하기	유니코드 상호 변환하는 함수를 이해한다. 문자열을 자유자재로 다루는 방법을 배운다.
2H	파일시스템 정복하기	플랫폼별 파일/디렉토리 접근 함수를 이해한다. 파일 시스템을 자유자재로 다루는 방법을 배운다.
2H	데이터 정복하기	STL 자료구조를 이해하고, 실무 응용 기술을 배운다. XML/JSON/INI 등을 자유자재로 다루는 방법을 배운다.
2H	메모리 정복하기	C++의 꽃, 스택 메모리의 장단점을 알아본다. 4가지 영역의 메모리를 자유자재로 활용한다.
2H	소켓 정복하기	UDP/TCP 통신, DATAGRAM/STREAM의 차이를 이해한다. 소켓 통신 시스템을 자유자재로 다루는 방법을 배운다.
6H	**최종실습**	**앞에서 배운 지식과 기술을 활용하여 미니 프로젝트를 진행한다.**



코딩 실력이 곧 우주선이다.

우주같이 광활한 컴퓨터 세계로 여행을 떠나자.

<참고서적>

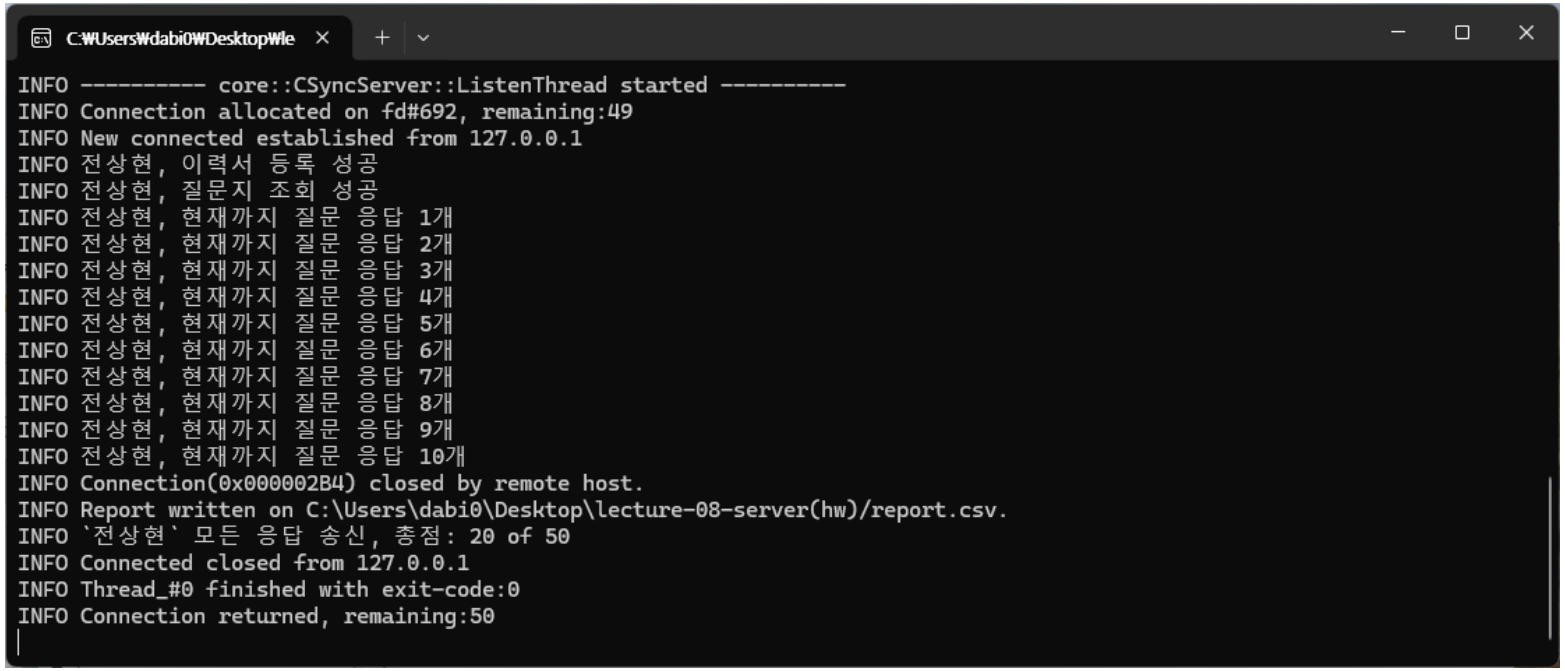
크로스플랫폼 핵심모듈 설계의 기술(2018) – 전상현, 로드북

아무도 알려주지 않은 C++ 코딩의 기술 (2023) – 전상현, 로드북

들어가기 전에 최종 실습에 대해서

1. 강의실에 띄워 둔 서버에 본인이 C++로 개발한 코드로 접근해서
2. 문제를 다운로드 받아 정답을 작성해 제출하여
3. 서버에서 채점한 점수로 최종 실습을 마무리 합니다.
4. 구체적인 프로토콜과 IP 및 PORT 정보는 당일날 아침에 알려줄 것이고,
5. 그 방법은 그 전날까지 배운 모든 코딩과 이론 지식을 총 동원해야 할 것입니다.

우측의 캡쳐 화면은 서버에 성공적으로 접속하여 문제에 대한 답을 제출할 때 서버측에서 실시간으로 보여주고 있는 화면입니다.



공통: git

공통 : https://github.com/profrog-jeon/cppcore.git clone 해두기

Windows: VisualStudio 2026 Community

Windows: PEView, HxD, NotePad++, void사의 everything

Windows: WSL 2.0 Ubuntu

Linux or MAC: VisualStudio Code

Linux or MAC: gcc, g++, cmake

소스코드 관리 디렉토리 구성

1. 윈도우 개발자들은 드라이브 파티션을 둘 이상(C:, D:)으로 나눕니다

2. C 드라이브는 OS에게 양보합니다.

3. D 드라이브에는 포맷해도 남겨둘 자산을 남겨둘 예정입니다.
   - 아직 D 드라이브가 없다면 남는 공간을 파티션 분리해주세요.
   - D:\GIT\ 이하에 소스코드를 솔루션 단위로 분리하여 관리합니다.
   - 리눅스는 /home/{아이디}/GIT/ 이하에 관리합니다.

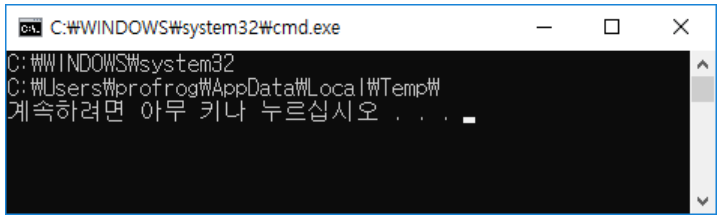
시스템 프로그래밍
맛보기 퀴즈

다짜고짜 퀴즈, 다음 코드 실행 결과는?

```
#include "stdafx.h"
#include "../../Inc/cppcore.h"

int main()
{
	std::tstring strDir1 = GetSystemDirectory();
	std::tstring strDir2 = GetTempPath();

	tprintf(TEXT("%s₩n"), strDir1.c_str()));
	tprintf(TEXT("%s₩n"), strDir2.c_str()));
	return 0;
}
```



위에서 이상한 점은 무엇일까?

다짜고짜 퀴즈2, 다음 코드 실행 결과는?

```
    std::tstring strSysDir = GetSystemDirectory();
    size_t tSysDirLen = strSysDir.length();
#ifdef __linux__
    if( tSysDirLen > 0 && (strSysDir.at(tSysDirLen-1) != TEXT('/') )
       strSysDir += TEXT("/");
#endif
#ifdef _MSC_VER
    if( tSysDirLen > 0 && (strSysDir.at(tSysDirLen-1) != TEXT('\\') )
       strSysDir += TEXT("\\");
#endif
#ifdef __linux__
    std::tstring strSomePath = strSysDir + TEXT("/some/path");
#endif
#ifdef _MSC_VER
    std::tstring strSomePath = strSysDir + TEXT("\\some\\path");
#endif
```

이거 뭐하는 코드일까요?

다짜고짜 퀴즈2, 정답

쉽게 예를 들면 다음과 같습니다. 윈도우는 아래의 모든 경로를 동일하게 인식합니다.

C:\some\path C:/some/path C:\some/path C:/some\\\\\\\\path	C:\some\\path C:/some//path C:\some/\path C:/some\/////////path

리눅스도 비슷합니다. 다음 경로는 모두 같죠.

/some/path
/some\/path
\/some\/\path
	//some/path
	//some/\path
	///some////////path

그런데 리눅스는 특이한 점이 있습니다. 다음 경로는 다르게 인식합니다.

/some\path	/some\\path

각각은 /somepath와 /some₩path라는 단일 디렉토리라고 생각합니다.

중요한 이야기이다.

실행파일 이해하기

프로그래밍 언어란 무엇인가?

한 마디로 정의하면, **"실행가능한 숫자들을 만드는 도구"**이다.

0과 1만 기록할 수 있는 도화지(메모리)에 내용을 채우는 것이 목표다.

컴퓨터 과학이란?

"한정된 메모리 공간에 효율적이고 필요한 정보를 담는 규칙을 배우는 것"이다.

프로그램 개발이란?

"필요한 정보를 담는 새로운 규칙을 만드는 과정"이다.

우리가 알고 있는 모든 언어들 즉,

어셈블러, C언어, C++, C#, JAVA, Python, JavaScript, Ruby, Haskel, Kotlin 등은

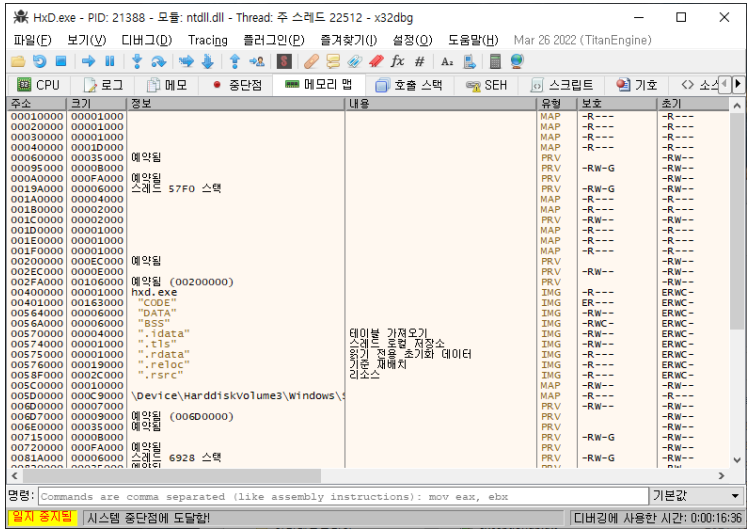
결국 **"기계어를 만들기 위한 도구"**일 뿐 최종 목적지는 같다.

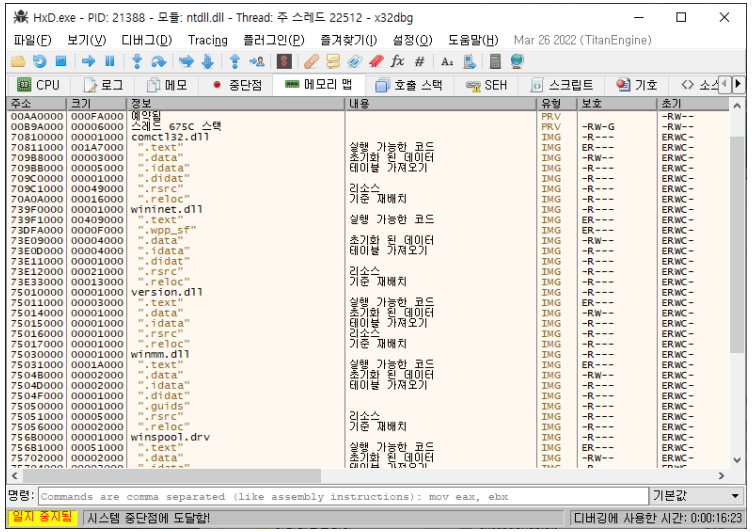
이것은 윈도우, 리눅스, iOS, 구름, 아이폰, 삼성폰, 임베디드 모두 동일하다.

PE 파일이 갖고 있는 데이터와 메모리 공간에 대해서

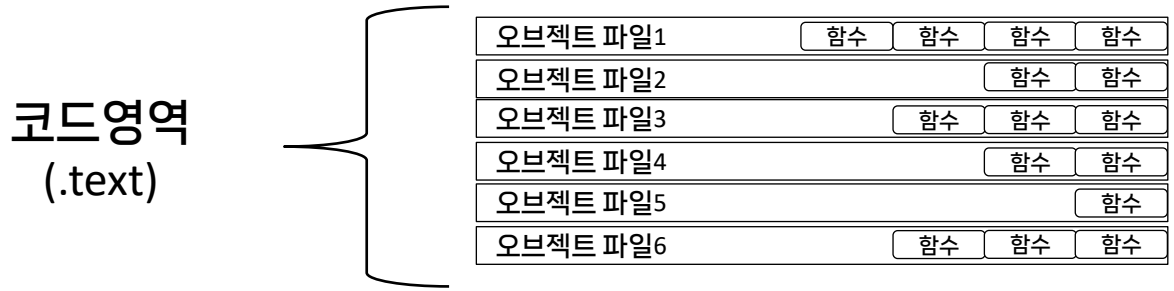
EXE 파일이 실행되면 펼쳐지는 4GB 메모리 공간에는 다음과 같은 정보가 있다.

구분	영역 이름	내용
CODE(.text)	코드 섹션	OPCODE 들이 나열된 기계어 코드가 쓰여진 공간
DATA(.data)	데이터 섹션	전역 변수가 정적 변수가 선언되는 공간
BSS	데이터 섹션 일부	정적 변수 중 값이 0으로 할당되는 변수들
.idata	임포트 테이블 섹션	EXE가 사용하는 암시적 링크의 DLL 정보들
.tls	스레드 지역변수 섹션	스레드 별로 고유하게 사용하는 변수 저장공간
.rdata	읽기전용 데이터 섹션	전역에서 읽기 가능한 읽기 전용 변수 const char* psz = “hello world?”
.reloc	PE 주소 재배치 섹션	ASLR에 의해 기준주소를 임의로 변경한 값을 보관하는 저장공간
.rsrc	리소스 섹션	버전,아이콘, 커서, 비트맵 등 리소스 관련 데이터 저장공간



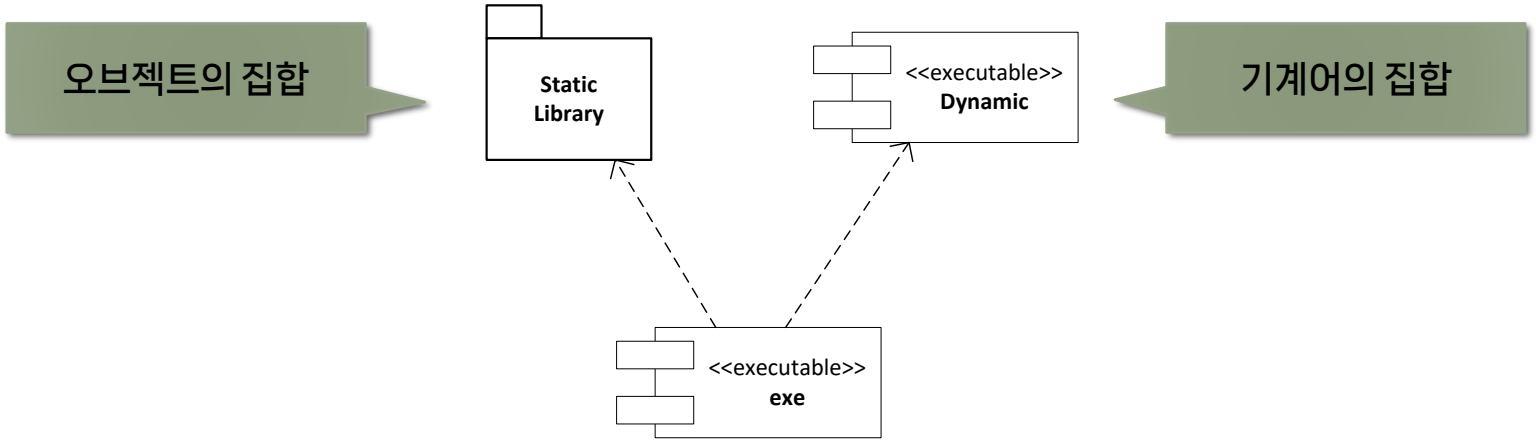


소스 코드와 라이브러리 관계



우리가 짜는 소스코드는 최종적으로 .text 영역에 들어간다.

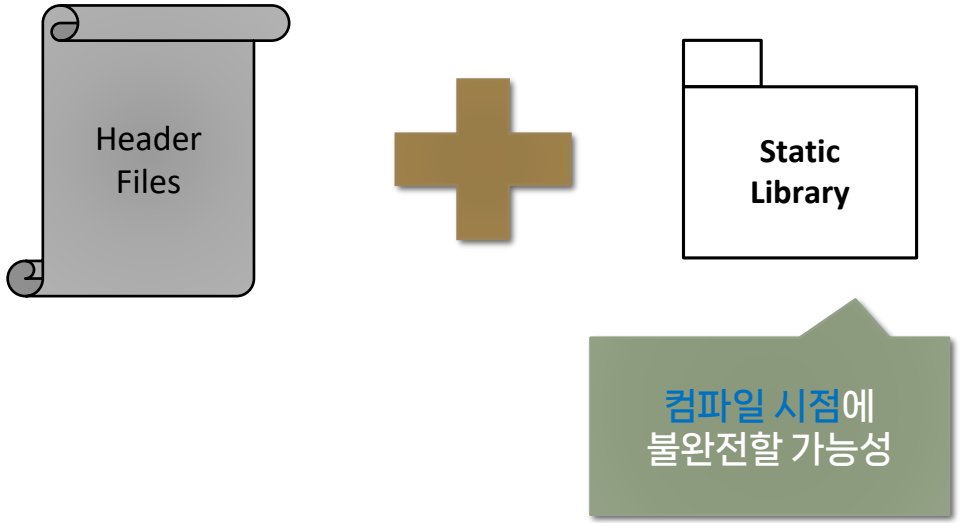
- 최소 단위는 개념적으로는 함수지만, 물리적으로 파일이다.
- 즉, 코드 영역은 파일 단위(오브젝트) 코드의 집합이다.
- 정적 라이브러리도 결국 오브젝트 파일의 묶음일 뿐이다.
- 동적 라이브러리는 기계어의 집합, 즉 코드 그 자체의 독립적인 개체이다.



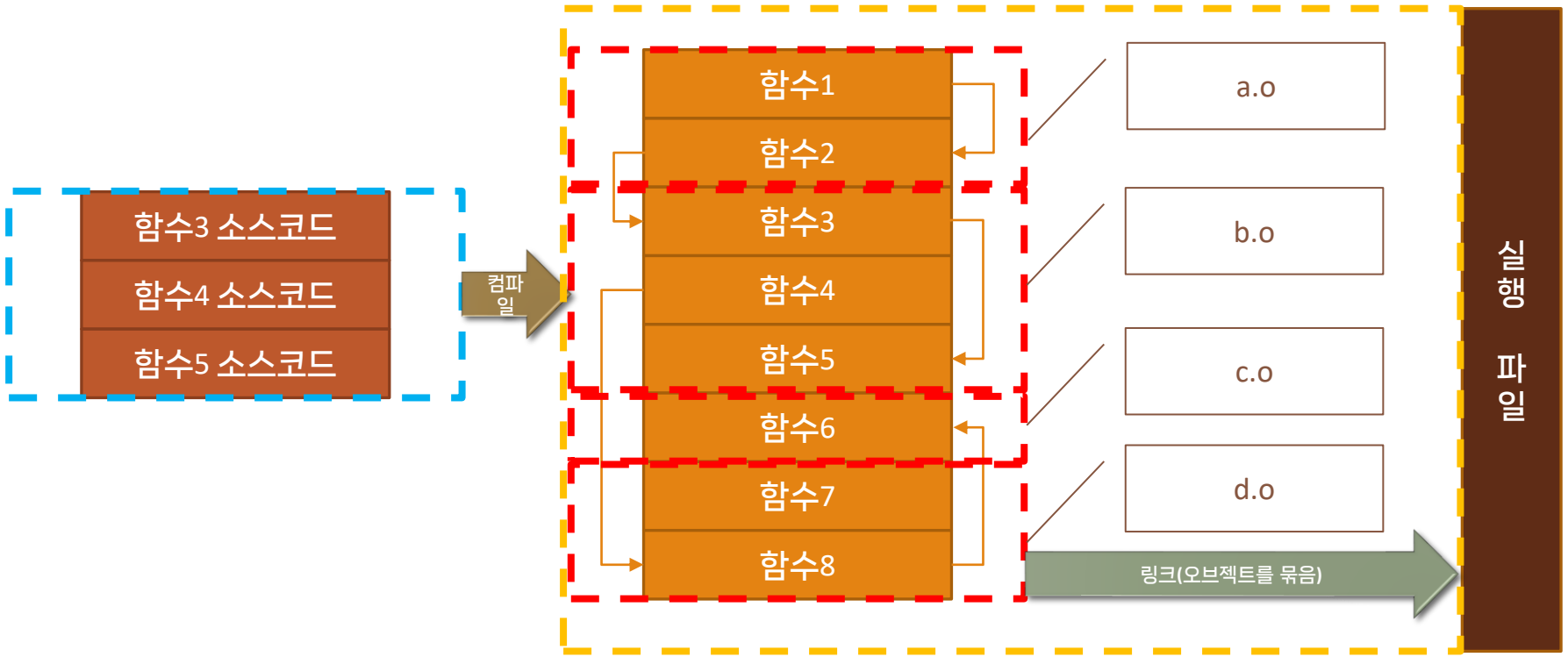
정적 라이브러리란?

정적 라이브러리 패키지

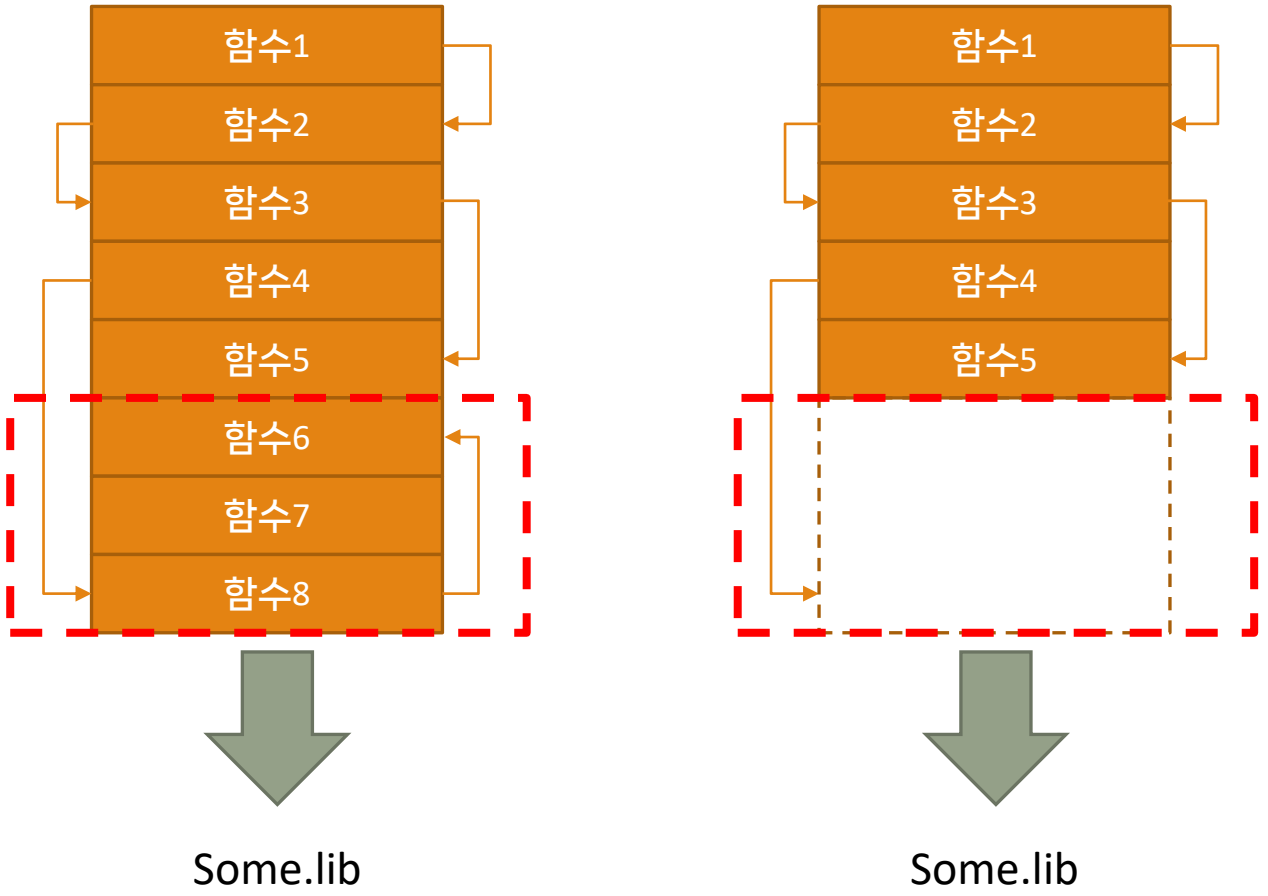
정적 라이브러리는 헤더파일(.h)들과 라이브러리(.lib/.a)파일로 구성됩니다.



소스코드 파일 -> 오브젝트 파일 -> 기계어



불완전한 상태도 허용하는 정적 라이브러리

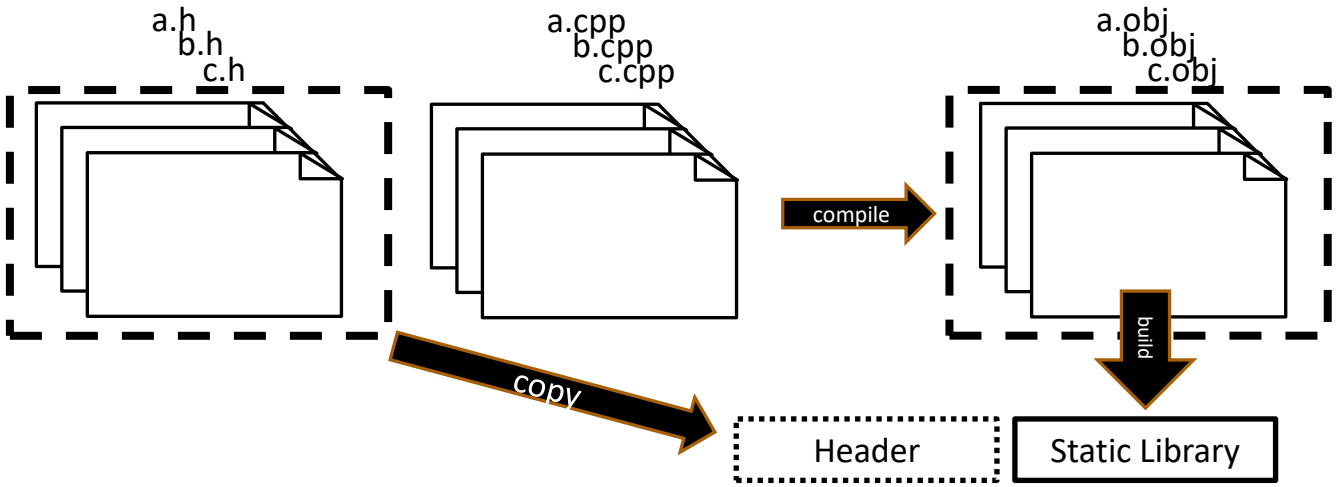


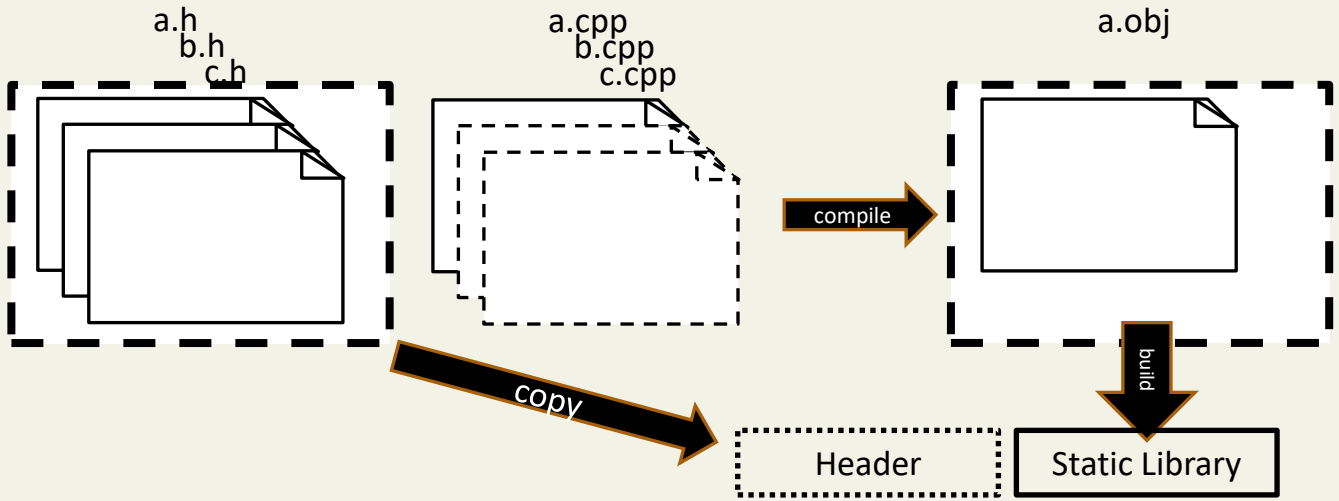
빌드된 lib 파일만으로는 모든 함수가 완전한지 여부는 알 수 없다.

최종 EXE 파일이나 DLL 파일이 빌드되는 과정에서 검증하기 때문이다.

정적 라이브러리인 Lib은 중간 단계 산출물이다.

불완전한 상태도 허용하는 정적 라이브러리



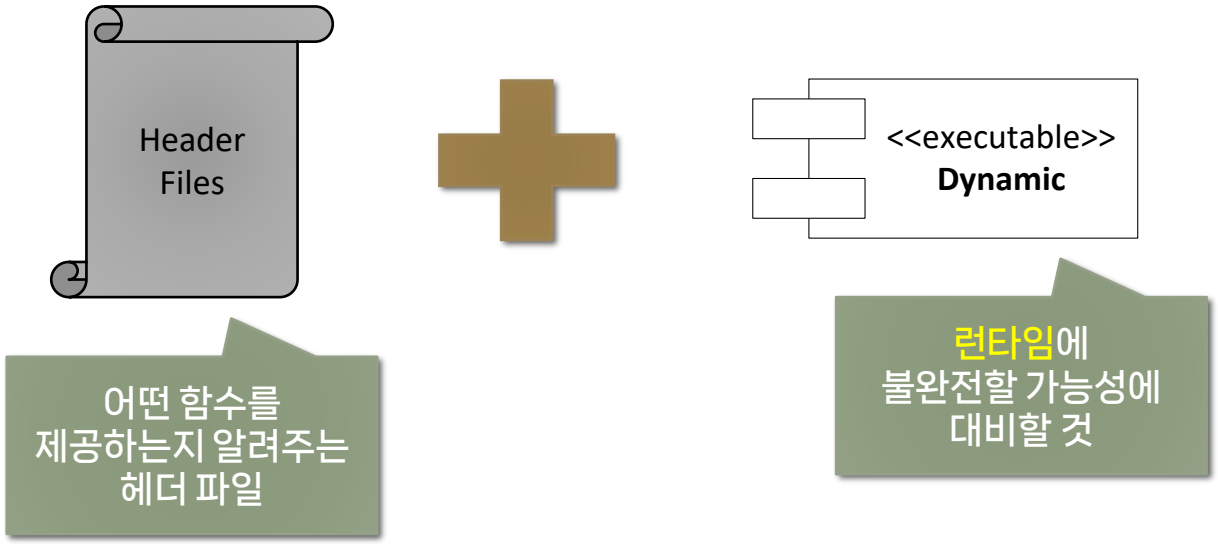


왜?

동적 라이브러리란?

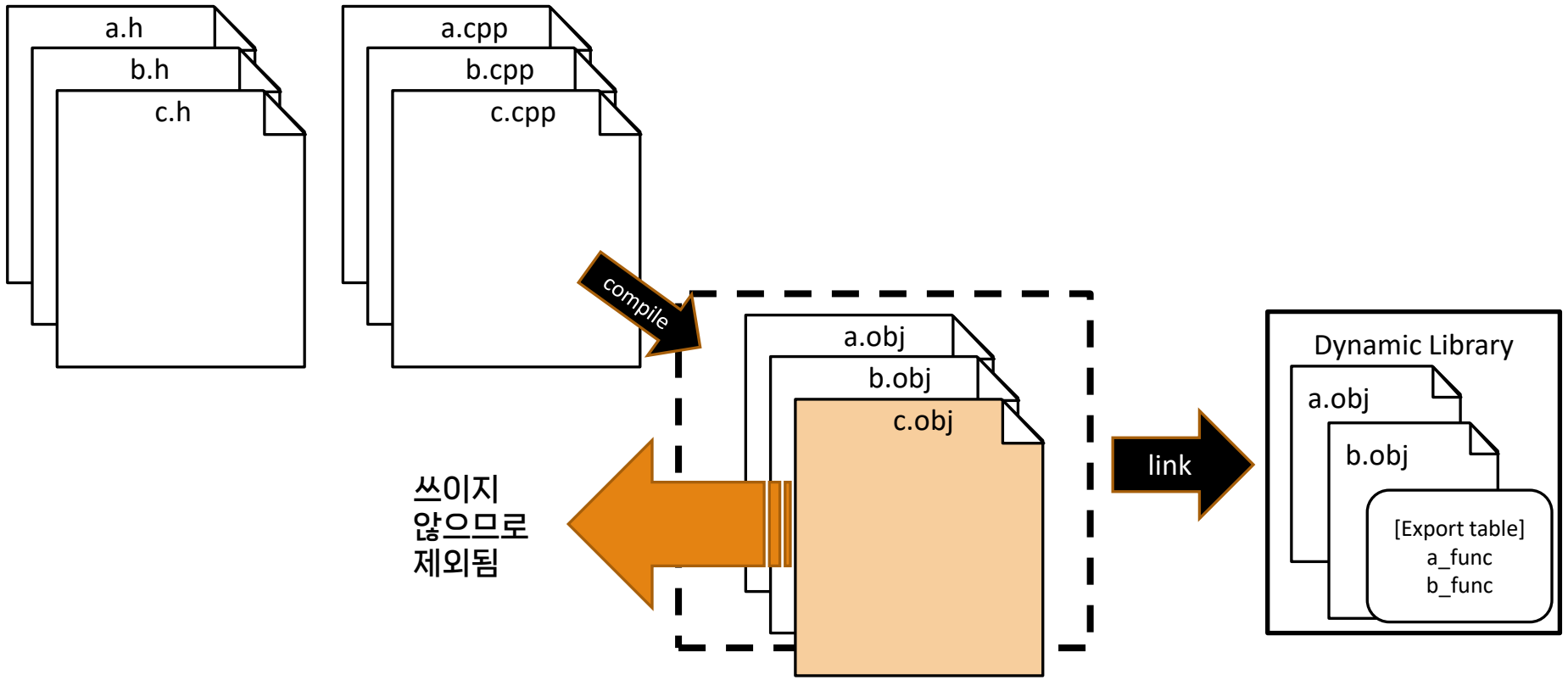
동적 라이브러리의 패키지 형태

동적 라이브러리는 헤더 파일(.h)들과 실행가능한 라이브러리 파일(.dll/.so/.dylib)로 구성됩니다.



동적 라이브러리 특성

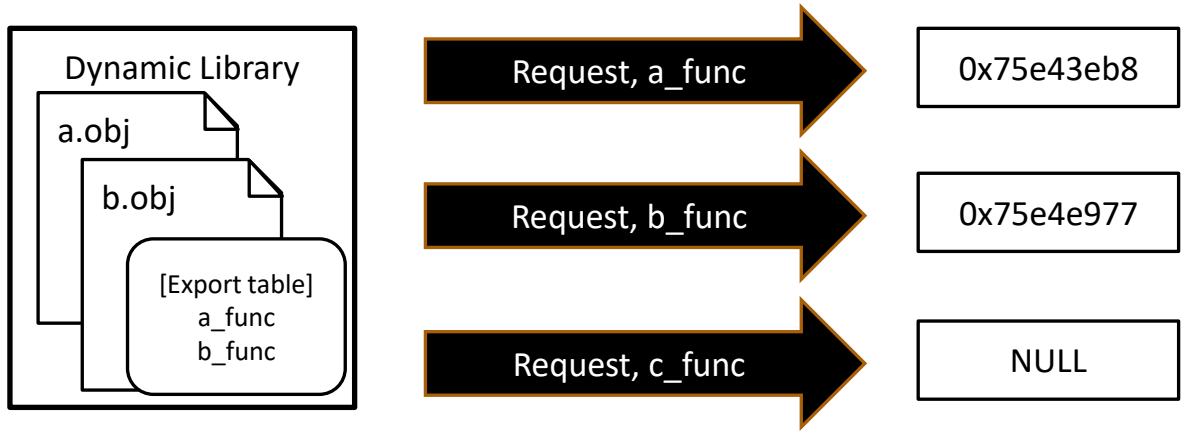
특징	Export 함수들은 반드시 동작할 수 있게 코드나 변수들이 검증되어 빌드된다. 따라서 Export 함수의 규격에 맞게 부르기만 하면 된다.



동적 라이브러리 특성

동적 라이브러리를 EXE가 가져다 쓰는 방법은 암시적 링크와 명시적 링크 두 가지입니다.

	암시적 링크	명시적 링크
사용방법	EXE 파일의 IMPORT 테이블에 DLL 파일을 정의 함 1. DLL을 빌드할 때 생성된 lib 파일을 EXE가 정적 라이브러리처럼 링크	EXE 파일이 실행 중에 직접 DLL을 읽음 1. DLL을 LoadLibrary로 메모리에 로드 2. EXPORT 함수 주소를 GetProcAddress로 얻어냄 3. 함수의 원형으로 강제 캐스팅하여 사용
주의사항	EXE 파일과 동일한 위치에 DLL이 존재해야함 존재하지 않으면 EXE 자체가 실행되지 않음 "... DLL 파일이 없습니다."라는 메시지 박스	DLL이 없으면 LoadLibrary가 NULL 반환할 수 있음 얻어오려는 함수이름이 없으면 NULL 반환할 수 있음
비고		



동적 라이브러리 장점 및 단점

장점	EXE 파일은 그대로 두고 DLL만 업데이트하여 사용할 수 있다. (코드 및 바이너리 수정 범위의 지역화 -> 설계관점에서 중요)
단점	DLL의 변경된 함수를 모른채로 기존 EXE가 동작하면 문제가 발생한다. 정확히는 메모리 접근 오류이다.

고생 많으셨습니다!