라이브러리란, 소프트웨어 개발에서 자주 쓰고 기초적인 함수들을 중복 개발하는 것을 피하기

위해 표준화된 함수 및 데이터 타입을 만들어서 모아 놓은 것을 말한다.

즉, 자주 사용되는 표준적인 함수를 매번 직접 작성해사 사용하는 것은 시간적으로 손해이기 때문에 이를 줄이기 위해 표준화 할 수 있는 함수를 미리 만들어서 모아 놓은 것이다.

한 번 구축해놓은 라이브러리는 다시 만들 필요 없이 불러서 사용할 수 있으므로 개발 속도가

빨라지고 신뢰성도 확보할 수 있다.

이 라이브러리는 언제 메인 프로그램에 연결하는가 에 따라서 Static(정적) 과 Dynamic(동적) 으로 나뉘며, 흔히 볼 수 있는 DLL 파일은 Dynamic Linked Library 의 약자로 후자에 속한다.

Static Linked Library(정적 링크 라이브러리) 의 확장명은 .lib 이고, 컴파일 시점에 라이브러리가 link 되어 실행된다.

특정 기능의 라이브러리를 static 하게 제작한다는 것은 link 단계에서 라이브러리를 실행 바이너리에 포함시킨다는 얘기이다.

즉, 라이브러리의 동작 코드가 이를 사용하는 실행 바이너리 속에 포함되기 때문에 별도의 추가 작업 없이 독립적으로 라이브러리의 함수들을 사용할 수 있다.

하지만 정적 라이브러리를 사용하는 프로그램이 늘어날수록 불필요하게 실행 파일들의 크기가 커지며, 라이브러리가 동시에 여러 실행 바이너리에 포함되어 실행되는 경우 메인 메모리 공간 활용 효율이 떨어지는 등 프로그램을 동시에 여러 실행하는 경우 그다지 바람직하진 않다.

Dynamic Linked Library(동적 링크 라이브러리) 의 확장명은 .dll 이고, 프로그램 실행 시 필요할 때 외부 DLL 파일에서 함수를 실행한다.

즉, 동적 링크 라이브러리는 이를 사용하고자 하는 실행 바이너리에서 필요시 사용할 수 있도록

최소한의 정보만 포함하여 링크하거나, 아예 독립적으로 DLL 을 로드/사용/해제할 수 있다.

이 점은 메모리 절약에 효율적이라는 장점을 가지고 있다.

DLL 을 구현하고 컴파일하고 나면 static library 와는 다르게 Output file 이 2 개 생성된다.

하나는 .lib 파일이고 하나는 .dll 파일이다.

여기서 .lib 파일은 static library 의 .lib 파일과는 전혀 다르다.

Static Library의 .lib 파일은 라이브러리 전체 코드를 포함한 바이너리이며, DLL 의 .lib 파일은 DLL 파일이 제공하고자 하는 함수 정보를 가지는 정보 파일이다.

DLL 의 .lib 파일을 이용하여 링킹하는 것을 암시적 링킹(implicit linking) 이라고 한다.

실행 바이너리를 링크 단계에서 실행 바이너리의 .obj 파일들과 DLL의 .lib 파일을 함께 링크하여 이 정보를 토대로 runtime에 DLL의 함수 코드를 참조하게 되는 것이다.

즉, .lib 파일은 링크 시에 필요하고 .dll 파일은 실행시에 필요하다.

명시적 링킹(Explicit linking)에서는 .lib 파일이 필요하지 않다.

실행 바이너리 링크 단계에서 DLL 함수 정보가 필요하지 않기 때문이다.

다음은 명시적 링킹에 사용하는 세 가지의 함수와 그 역할이다.

LoadLibrary : 필요한 DLL 을 프로세스 가상 메모리에 맵핑

GetProcAddress : DLL 함수의 포인터를 흭득한다.

FreeLibrary : 프로세스 가상 메모리에서 DLL 을 반환

프로세스는 내부적으로 DLL 래퍼런스 카운트를 계산하고, LoadLibrary 호출 시 DLL 의 래퍼런스 카운트는 1 증가, FreeLibrary 가 호출되면 1 감소한다. 그리고 래퍼런스 카운트가 0 이 될 때마다 DLL 은 프로세스 가상 메모리에서 해제된다.

이러한 래퍼런스 카운트를 두는 이유는 프로그램 실행 중에 DLL 을 가상 메모리에 할당, 해제할 수 있도록 하기 위함이다. 암시적 링킹에선 이러한 장점을 얻을 수 없다.

끝으로 두 라이브러리의 차이점을 정리하자면 아래와 같다.

Dynamic Linked Library

1 프로그램 실행 시 필요시만 외부 DLL 파일에서 함수를 참조

2 프로그램 실행 시 프로그램 로딩 시간 단축

3 함수 업그레이드 시 해당 DLL 만 수정 배포할 수 있다.

Static Linked Library

1 필요한 함수를 프로그램 코드에 붙여 프로그램 자체에서 참조

2. 프로그램 실행 후 빠른 처리 시간이 빠르다.

3. 프로그램 실행 파일만 있어도 실행 가능하다.

실행 파일 자체의 크기를 비교하면 당연히 정적 라이브러리를 포함한 프로그램의 크기가 더 크게 된다.