template / linker

EUnS

January 5, 2020

1. linker
   - Object File
   - symbol
   - linker
   - Executable Object Files
2. #include이해
3. template
4. TMP
5. constexpr(c++11)

- 서적 : CSAPP (7장 linker)
- 아마 시프시간에 배우는 것.

Complier

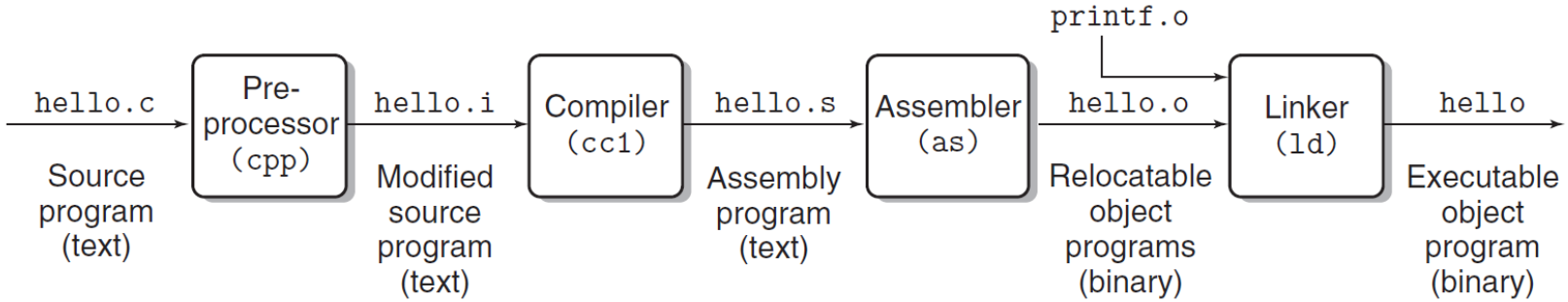

Figure: 컴파일 과정

Object File

- Relocatable ob : compiler,assembler output
- Executable ob : linker output
- shared ob : DLL을 위한 파일 생략

x86-linux : object구조로 ELF(Executable and Linkable Format)사용

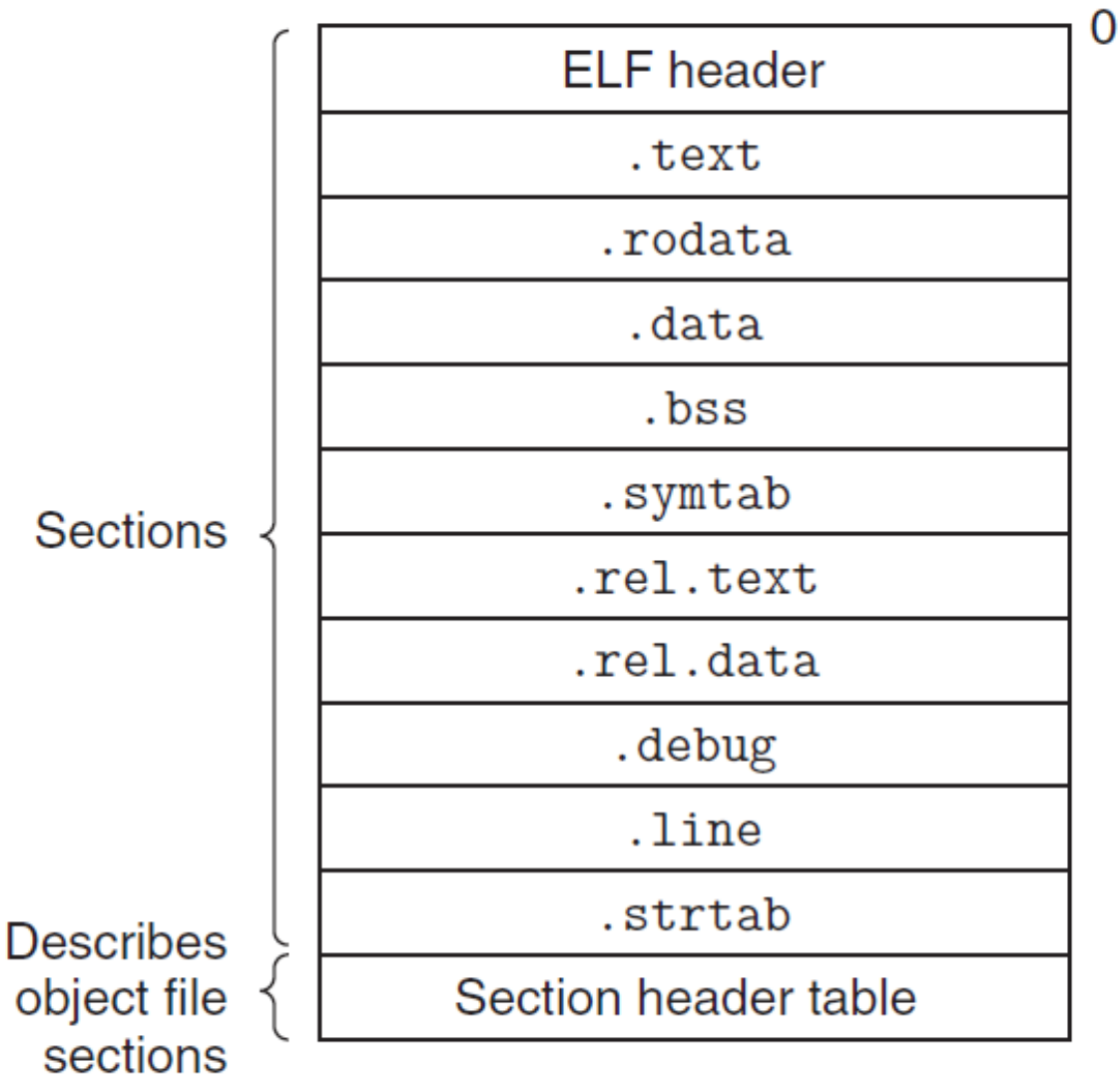

Figure: Typical ELF relocatable object file.

- **.text** The machine code of the compiled program.
- **.rodata** Read-only data such as the format strings in printf statements, and jump tables for switch statements.
- **.data** Initialized global and static C variables.
- **.bss** Uninitialized global and static C variables, along with any global or static variables that are initialized to zero. block satrted by symbol 의 약어
- **.symtab** A symbol table with information about functions and global variables that are defined and referenced in the program.
- **.rel.text** A list of locations in the .text section that will need to be modified when the linker combines this object file with others.
- **.rel.data** Relocation information for any global variables that are referenced or defined by the module.
- **.debug** A debugging symbol table with entries for local variables and typedefs defined in the program, global variables defined and referenced in the program, and the original C source file.
- **.line** A mapping between line numbers in the original C source program and machine code instructions in the .text section.
- **.strtab** A string table for the symbol tables in the .symtab and .debug sections and for the section names in the section headers.

symbol

- Global symbols : 모듈 m에 정의되거나 다른 모듈에 참조될 수 있는 심볼 / 전역변수와 static이아닌 함수
- Global symbols : 다른 모듈에 정의된 전역변수 /다른곳에 정의된 (extern)전역변수와 static이아닌 함수
- Local symbols : static으로 선언된 함수, static 전역변수

Q : 지역 변수는요?
A : 런타임에 스택에 쌓습니다.

linker 단계

1. symbol resolution
2. Relocation

Symbol Resolution

- strong symbol : 초기화된 전역변수, 함수
- weak symbol : 초기화x 전역변수

Rule

1. Multiple strong symbols with the same name are not allowed.
2. Given a strong symbol and multiple weak symbols with the same name, choose the strong symbol.
3. Given multiple weak symbols with the same name, choose any of the weak symbols.

Relocation

1. Relocating sections and symbol definitionss : 여러개의 .data section을 합친다.
2. Relocating symbol references within sections. :모든 심볼 참조를 수정한다. 이후 모든 심볼들이 런타임 메모리 주소를 가진다. 어셈블러가 만든 .rel.data section에 있는 Relocation Entries 자료구조를 가지고 수행한다
   - R_X86_64_PC32 : 상대주소
   - R_X86_64_32. : 절대주소

   이후 인스트럭션과 전역변수들이 런타임 메모리 주소를 가진다.

Executable Object Files

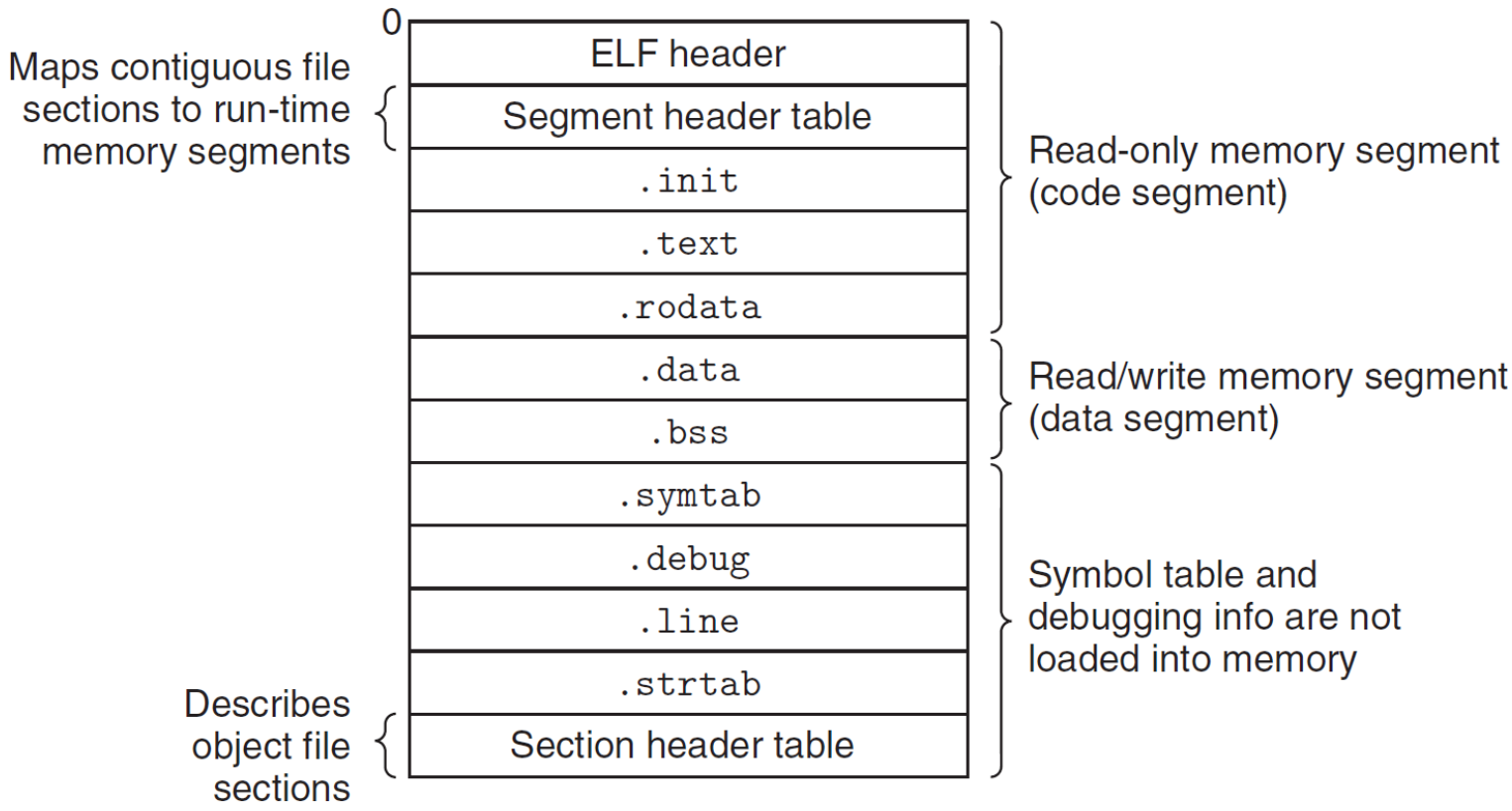

Figure: Typical ELF executable object file.

Loading

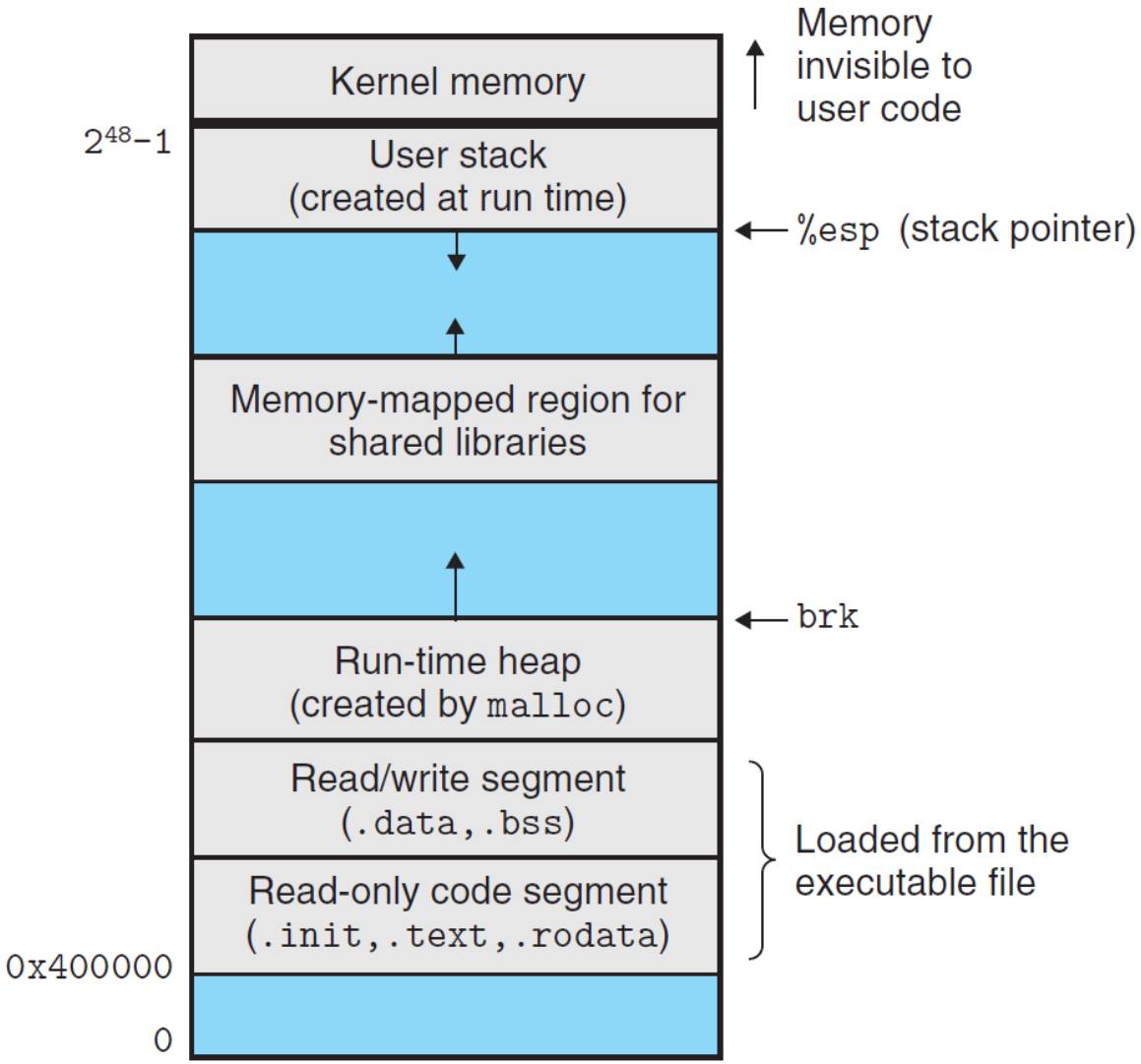

Figure: **Linux x86-64 run-time memory image.**

header file

- 그냥 텍스트파일과 같다.
- 선언을 위한것...
- 실제 구현은 어느 cpp파일에서...
- #include는 전처리 지시자.
- #include로 cpp파일에 그대로 붙여넣는다.

template

- template을 실제로 인자를 넣어서 사용할때 전처리에서 인자를 넣은 코드를 생성함
- 타입 아무거나 넣어도 가능
- 실행 파일크기가 예상과 다르게 커질수있음

```
1   template<typename T>
2   T function(T a , T* b, T& c)
3   {;}
4   template<>
5   char function<char>(char a , char* b, char &c)
6   {;}
7
8   int main()
9   {
10      function(1,0x0000,3);
11      function("s", 0x0000,"asdf");
12  }
13
```

TMP

```
1   template <int N>
2   struct Factorial {
3       static const int result = N * Factorial<N - 1>::
    result;
4   };
5
6   template <>
7   struct Factorial<1> {
8       static const int result = 1;
9   };
10  Factorial<4>::result
11
```

- template의 특성을 이용해서 반복되는 계산을 컴파일타임에 계산을 해놓은다음 그 값을 $O(1)$에 부르는 흑마법
- 이런짓도 가능

TMP는 나쁘다

참고

- 극암의 코드 가독성
- 헬 디버깅
- 어려움
- 써보고 직접 판단 내려보자.

constexpr

- 변수에 사용할때
  - #define 대체가능
  - const 상수 대체가능
  - 컴파일타임에 상수로 대체
- 함수에 사용할때
  - 어느부분 TMP 대체가능
  - 컴파일타임에 계산할수도있고 안할수도있음

```
1
2
```

과제

과제 3-2 : 링크드리스트