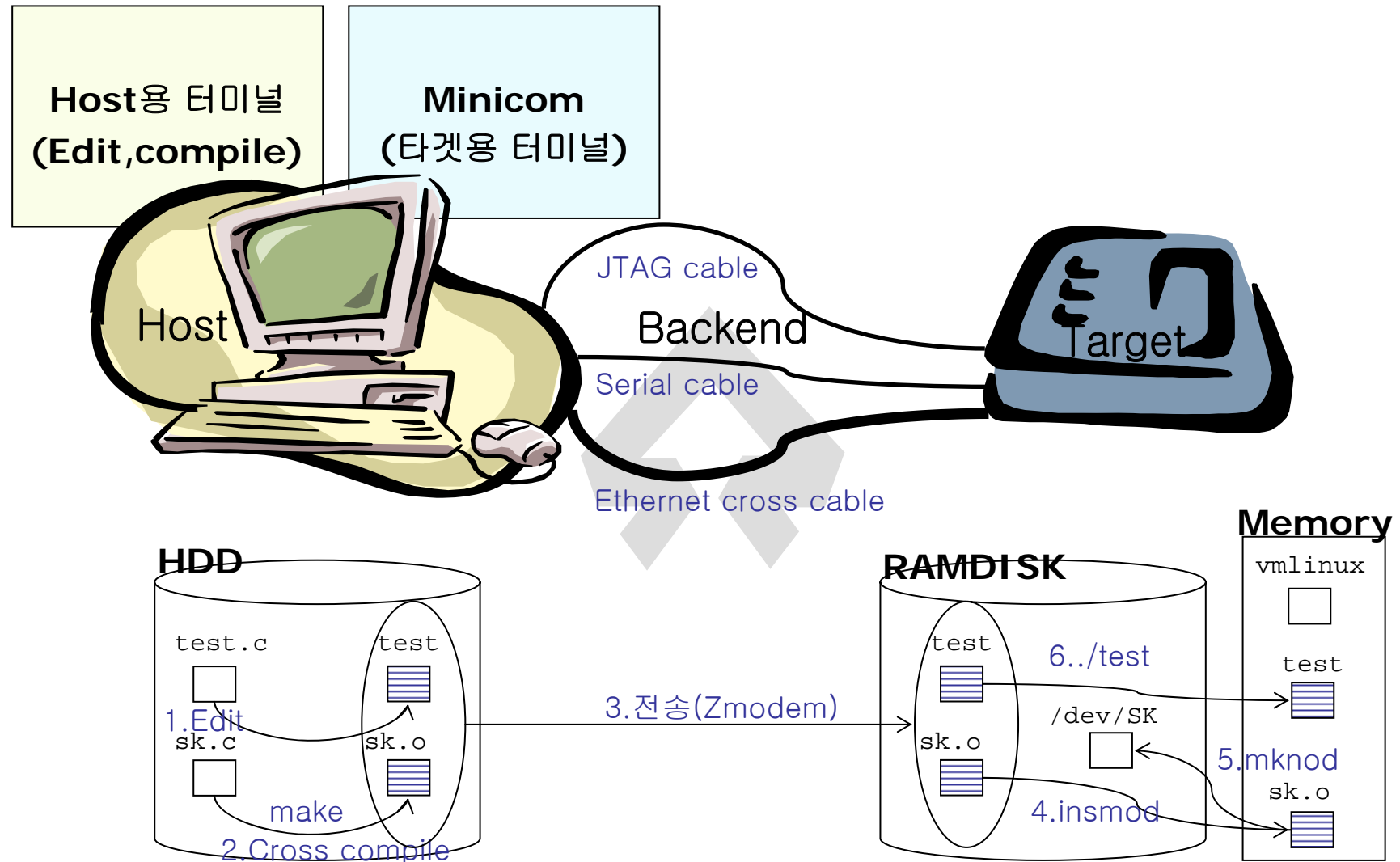


Tool-chain

Written by 조진제
2006. 07. 28



개발환경과 DD 및 App 실행 과정





GNU Tool-Chain 이란?

- GNU

- : GNU's Not Unix!

- 완벽한 unix호환 운영체제를 free software만들기 위해서 1984년 FSF(Free Software Foundation)의 지원으로 GNU project 시작.

- 이후 GPL(GNU General Public License)를 따르는 여러가지 free software를 공개

- Tool Chain의 정의

- : 각종 Source file들을 컴파일하고 build하여 실행 파일을 생성하는데 필요한 여러가지 유틸리티 및 라이브러리의 모임

- GNU Tool Chain의 구성

- : GNU에서 제공하는 tool chain은 세가지로 구성

- 1) GCC(GNU Compiler Collection)

- 2) GNU Binary Utilies

- 3) Libraries(glibc, newlib)

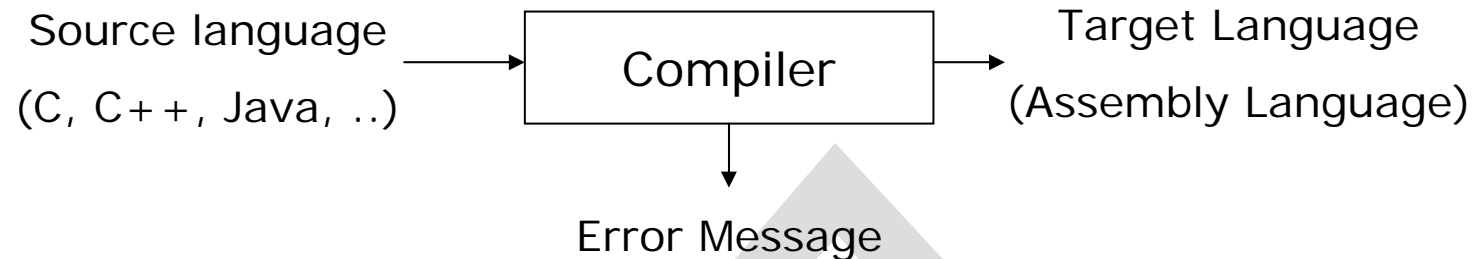
- * glibc : GNU C Library

- * Newlib : C library intended for use on embedded systems



What is compiler?

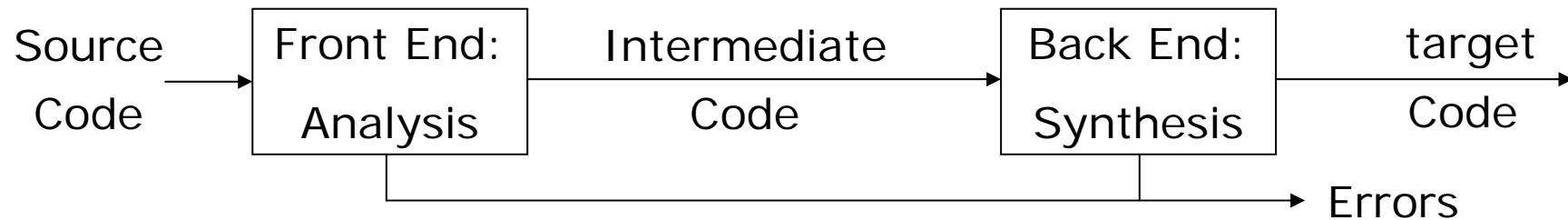
- Compiler
: Translate computer program from one language to another



- Compiler는 왜 필요한가?
: Too difficult to write, debug, maintain programs written in assembly language
- Source Code
: optimized for human readability
- Target Machine Code
: optimized for target hardware
- Goal of compiler
: translate a source code program into an **equivalent** machine code program efficiently



Compiler 구조



- Front End
 - : deal with input language
 - : 어휘(lexical) 분석, 구문(syntax) 분석, 의미(semantic) 분석
 - : build an Intermediate code for the back end
 - Back End
 - : deal with intermediate code
 - : Optimizer, Code Generator
 - * **Optimizer** : transform the intermediate code so to run faster faster and/or to use less space
 - * **Code Generator** : generator target program(usually assembly code)
- from optimized intermediate code



GNU Compiler Collection

- Retargettable
- Available from Free Software Foundation
 - GPL Licensed
 - Free
- Written by Richard Stallman originally(FSF)
- Front-end, back-end system with support for many of each.
- 현재 가장 최신 버전은 GCC 3.3.3 (2004-02-14)
- 그러나 최신버전이 최선은 아니다.
 - Compiler needs **correct**, **stable** and fast code
- More stable and popular version :
 - **gcc 2.95.3**
 - : 상용 RTOS인 vxworks 5.5의 개발환경(Tornado2.2)에도 채용됨.
- Size of GCC
 - : GCC 3.3.2 154MBytes



GNU Compiler Collection

- Front Ends
 - C, C++, Objective C
 - Ada95(GNAT)
 - Fortran77, Fortran95
 - Pascal
 - Modula-2, Modula-3
 - Java(Supported from gcc 3.0)
 - : Java->Bytecode, Bytecode->native code
 - Java->Native code
 - Cobol
 - Chill



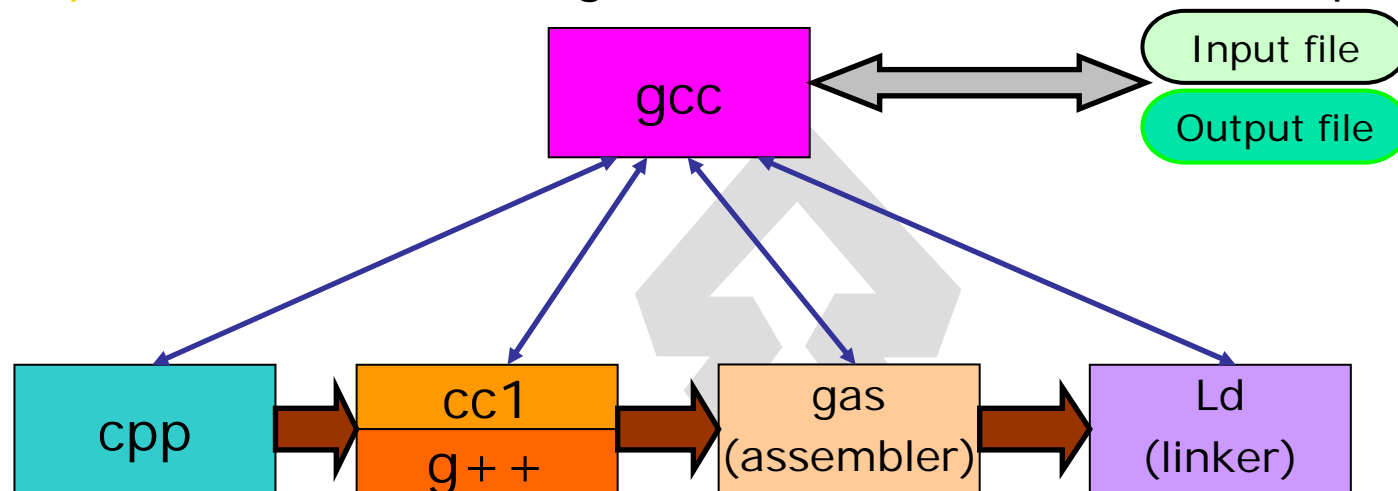
GNU Compiler Collection

- Machine Supported by GCC(back ends)
 - ARM : ARM7, ARM9, ARM9E, ARM10E, StrongARM, Xscale
 - Alpha DEC
 - Intel x86 Families, i860, i960
 - Motorola 680x0, PowerPC
 - Hitachi SH(Sh-1, Sh-2, Sh-3, Sh-3e, Sh-4), H8300
 - MIPS, HP PA-RISC, SUN SPARC
 -
 - Intel IA64, AMD x86-64



GNU Compiler Collection

- Compilation System
 - Preprocessing
 - Compiling
 - Assembling
 - Linking
- **Compiler Driver** (/usr/bin/gcc in linux) coordinates these phases.



- Input files
 - file.c : Preprocessor 과정을 거쳐야 하는 c source file
 - file.i : Preprocessor 과정을 거치지 않아야 하는 c source file
 - file.h : c header file(compile, link 되지 않음)
 - file.s : Assembler code
 - file.S : Preprocessor 과정을 거쳐야 하는 assembler code



Libraries

- C library
 - Any Unix-like operating system needs a C library
 - : the library which defines the "system calls" and other basic facilities
 - such as open, malloc, printf, exit...
- Intent
- - Glibc(GNU Libc)
 - : FSF에서 관리, native UNIX system에서 사용하던 라이브러리를 완벽하게 대체.
 - : 최신버전은 2.3.2
- - Newlib
 - : 많은 프로그래머들에 의해서 생성된 코드들의 모음
 - : Cygnus(1999년에 Redhat에 합병됨)에서 embedded system에서 사용할 수 있도록 패키지화
 - : Newlib은 glibc와는 달리 UNIX system에서 사용되는 라이브러리를 완벽하게 대체하지 못한다.
 - : 최신버전은 1.12.0
- Resource utilization
 - UNIX 환경을 고려하고 작성된 Glibc는 system의 메모리 용량을 고려하지 않고 설계
 - 대신 demanded-page-loaded shred library의 효율성을 중요시함
 - Newlib은 embedded system용으로 설계하기 위하여 메모리 용량에 대한 고려가 많았기 때문에 glibc보다 메모리 사용면에서 효율적이다.



Binary utilities

- binutils
 - : binutils에는 아래와 같은 두개의 주요한 유틸리티와 부가적인 유틸리티들이 포함되어 있다.
 - ld : GNU Linker
 - as : GNU assembler
 - nm : List symbols from object files(subset of objdump)
 - objcopy : Copy and translate object file from one to another
 - objdump : Display information from object files
 - ar : Create, modify, and extract from archive files
 - ranlib : Generate index to archive contents
 - size : List files section size and total size
 - strings : List printable strings from files
 - strip : Discard Symbols



Cross Tool-Chain

- library 이용 방법에 따라 3가지의 방법으로 만들 수 있음.
 - 동일한 gcc, biutils source 사용.
- 1) GNU Libc를 이용하는 GNU tool chain
 - 예) embedded linux 용 tool chain
- 2) Newlib을 이용하는 GNU tool chain
 - 예) RTEMS 용 compiler
 - OS를 이용하지 않는 firmware level
- 3) ulibc를 이용하는 GNU tool chain
 - 예) uclinux용 GNU tool chain
- 4) 자체적인 library를 이용하는 GNU tool chain
 - 예) vxWorks Tornado 개발환경
 - eCos GNU tool chain(수정된 newlib을 kernel/network stack/device driver와 함께 library화
 - GreenHill의 Multi 개발환경



GCC options

- GCC options
 - C 언어의 방언을 처리하는 옵션
 - C++ 언어의 방언을 취급하는 옵션
 - 경고를 요구/제어하는 옵션
 - 디버그를 위한 옵션
 - 최적화 옵션
 - Preprocessor 옵션
 - Assembly 옵션
 - linker 옵션
 - directory 검색용 옵션
 - Machine dependent 옵션
 - : ARM, i386, PowerPC, MIPS, SuperH, Sparc, ...





GCC options

- Options to request or suppress Warnings
 - w : 모든 경고 메시지를 무시한다.(표시하지 않는다).
 - Wall : 경고에 관련된 모든 옵션들을 활성화한다.(일반적인 사용)
- Options for Debugging Your Program or GCC
 - g, -ggdb, -gstabs, -gcoff, -gdwarf ...
- Options Controlling the Preprocessor
 - include *file*
 - : 다른 파일을 처리하기 전에 *file*을 먼저 처리한다.
 - nostdinc
 - : header file을 찾기 위해서 표준 시스템 디렉토리들을 검색하지 않는다.
 - I 옵션으로 주어진 디렉토리와 현재 디렉토리만 검색한다.
 - D*macro* : macro를 정의, -U*macro* : *undefine macro*



GCC options

- Options for Linking

-llibrary

: linking시 *library*로 지정된 library(lib*library*.a)를 찾는다.

-nostartfiles

: linking시 standard system startup file을 이용하지 않는다.

* *-nostdlib* 나 *-nodefaultlibs*가 이용되지 않으면 표준 시스템 라이브러리들은 이용된다.

-nostdlib

: linking시 standard system startup file이나 libraries를 이용하지 않는다. 사용자가 지정한 라이브러리들만 이용된다.

-static

: dynamic linking을 지원하는 시스템에서 shared libraries와의 linking을 막는다.



GCC options

- Options for Linking(2)

- shared

- : shared object file을 생성한다.(-fPIC option도 필요)

- wl, *option*

- : option을 linker의 option으로 전달한다.

- Options for Directory Search

- I*dir*

- : *dir*로 지정된 디렉토리를 header file을 검색하는데 이용되는 디렉토리 리스트에 추가한다.

- L*dir*

- : *dir*로 지정된 디렉토리를 -I로 지정된 라이브러리를 검색하는데 이용되는 디렉토리 리스트에 추가한다.



Binary Utilities

- ar

: 여러 파일들을 하나의 파일(archive)로 묶는다.

static library를 만드는데 이용된다.

ar은 옵션에 따라서 다음과 같은 일을 수행한다.

- archive로부터 file을 지운다.(-d)
- archive내의 file들의 리스트를 출력한다.(-p)
- archive에 새로운 파일을 추가한다.(-r) 동일한 파일이 있으면 교체한다.
- archive의 내용을 표시한다.(-t)
- archive로부터 file을 추출한다.(-x)

- ranlib

: archive의 색인을 만들어서 archive에 추가한다.

색인보려면 'nm -s'나 'nm --print-armap'을 사용한다.



Binary Utilities

- nm

: object file내의 symbol을 출력한다.

각각의 symbol에 대해서 value, type, name의 세가지를 출력한다.

-type

A : 심볼이 절대값이라서 더 링크해도 변하지 않는다.

B : 심볼이 bss 영역에 저장된다.

D : 심볼이 data 영역에 위치한다.

N : 심볼에 디버그 심볼이다.

T : 심볼이 text 영역에 있다.

U : 심볼이 정의되지 않았다.

R : 심볼이 읽기전용 섹션에 있다.

? : 알 수 없거나 object file 형식에 의존하는 심볼이다.



Binary Utilities

- strip

: object file에서 symbol을 제거한다.

최종 실행 코드에서 심볼들이 필요하지 않다면 파일의 사이즈를 줄이는데 유용하다. 여러 옵션들을 이용하여 심볼들을 부분적으로 제거할 수 있다.

- Strings

: object file내의 printable string들을 출력한다.

- Size

: object file의 각 section의 size를 출력한다.



Binary Utilities

- Objdump
 - : object file에 대한 각종 정보들을 출력하거나 역어셈블한다.
 - display file header (-f)
 - display section headers (-h)
 - disassemble a file (-D , -d)
 - disassemble with source code (-S)
 - display information about library files (-a)
 - display all header information (-x = -a -f -h -r -t)



Binary Utilities

- Objcopy

: 하나의 object file 내용을 다른 object file로 복사한다.

object file의 format을 변경하는데 이용된다.

-O bfdname

: bfdname 형식으로 출력파일을 작성한다.

-R sectionname

: 출력파일에서 sectionname 섹션을 제거한다. 이 옵션을 잘못 사용하면 출력파일을 사용하지 못하게 만들 수 있다.

-g

: 입력파일에서 디버깅 심볼을 복사하지 않는다.

- addr2line

: object file의 주소를 source code상의 줄번호로 변환한다.

A thick blue vertical bar with a fine diagonal texture is positioned on the left side of the slide.

Executable and Linkable Format

Written by 조진제
2006. 07. 28



Object files

- Object files
 - Contain the generated binary code and data
 - Linkable : used as input by link editor.
 - Executable : loaded into memory and run as a program.
 - Loadable : loaded into memory as a library along with a program.
- Contents of an object file
 - Header information: size of code, name of source file, creation date
 - Object code
 - Relocation information
 - Symbols
 - Debugging information
- Various object formats
 - COM, EXE, PE(Portable Executable)
 - a.out, COFF, ELF

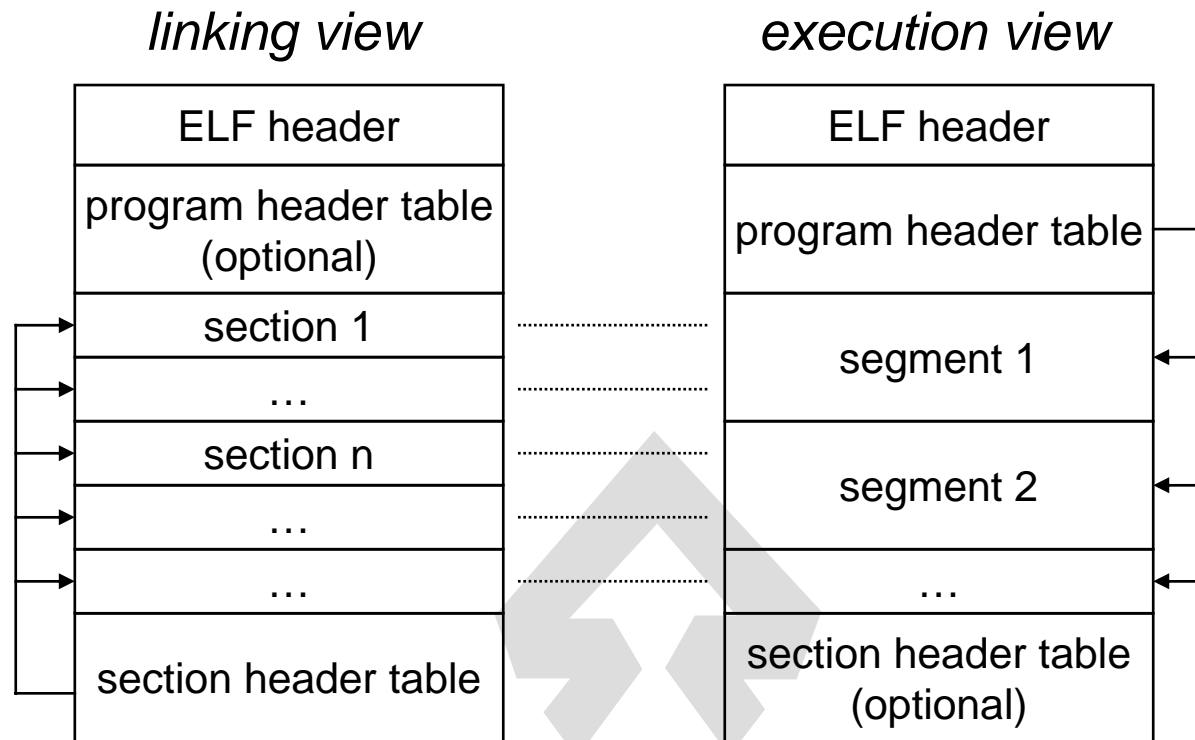


ELF File Format

- ELF (Executable and Linking Format)
 - Improved format of COFF which was originally intended for cross-compiled embedded systems
 - * COFF : didn't work well for a timesharing system.
 - not support C++, problem in dynamic linking.
 - Designed to support cross-compilation, dynamic linking
 - Adopted by the Unix System V(AT&T), Linux, and BSD Unix
 - Better support shared library than old a.out format
 - Better, more complete information for debuggers
- Three different types of an object in a common format
 - Relocatable object: have section tables
 - Executable object: have program header tables
 - Shared object: have both of them



Two views of ELF File

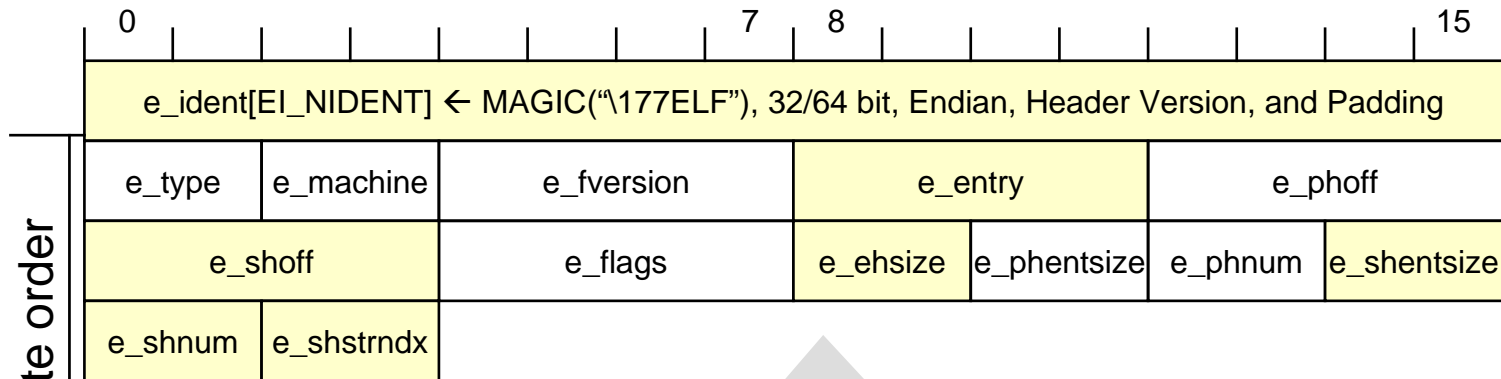


- 하나의 segment는 여러 개의 section들로 구성되어 있다.
ex. 하나의 loadable read-only segment는 code section, read-only data section, dynamic linker를 위한 symbol들로 구성되어 있다.
- section들은 linker에 의해 처리되어 지는 부분이고 segment는 loader에 의해 memory로 mapping되는 부분이다.



ELF Header

- ELF header format



e_ident	ELF identification (mach indep)	e_flags	arch specific flags
e_type	file type (relocatable, executable, ..)	e_ehsize	ELF header size
e_machine	2 = SPARC, 3 = x86, 28 = arm	e_phentsize	size of one entry in the program hdr
e_fversion	object file version (always 1)	e_phnum	number of entries in the program hdr
e_entry	virtual addr of entry point	e_shentsize	size of one entry in the section hdr
e_phoff	file offset of the program header	e_shnum	number of entries in the section hdr
e_shoff	file offset of the section header	e_shstrndx	section hdr table idx of the section name string table



ELF Section Header(1/4)

- Section header format (only for relocatable and shared object)

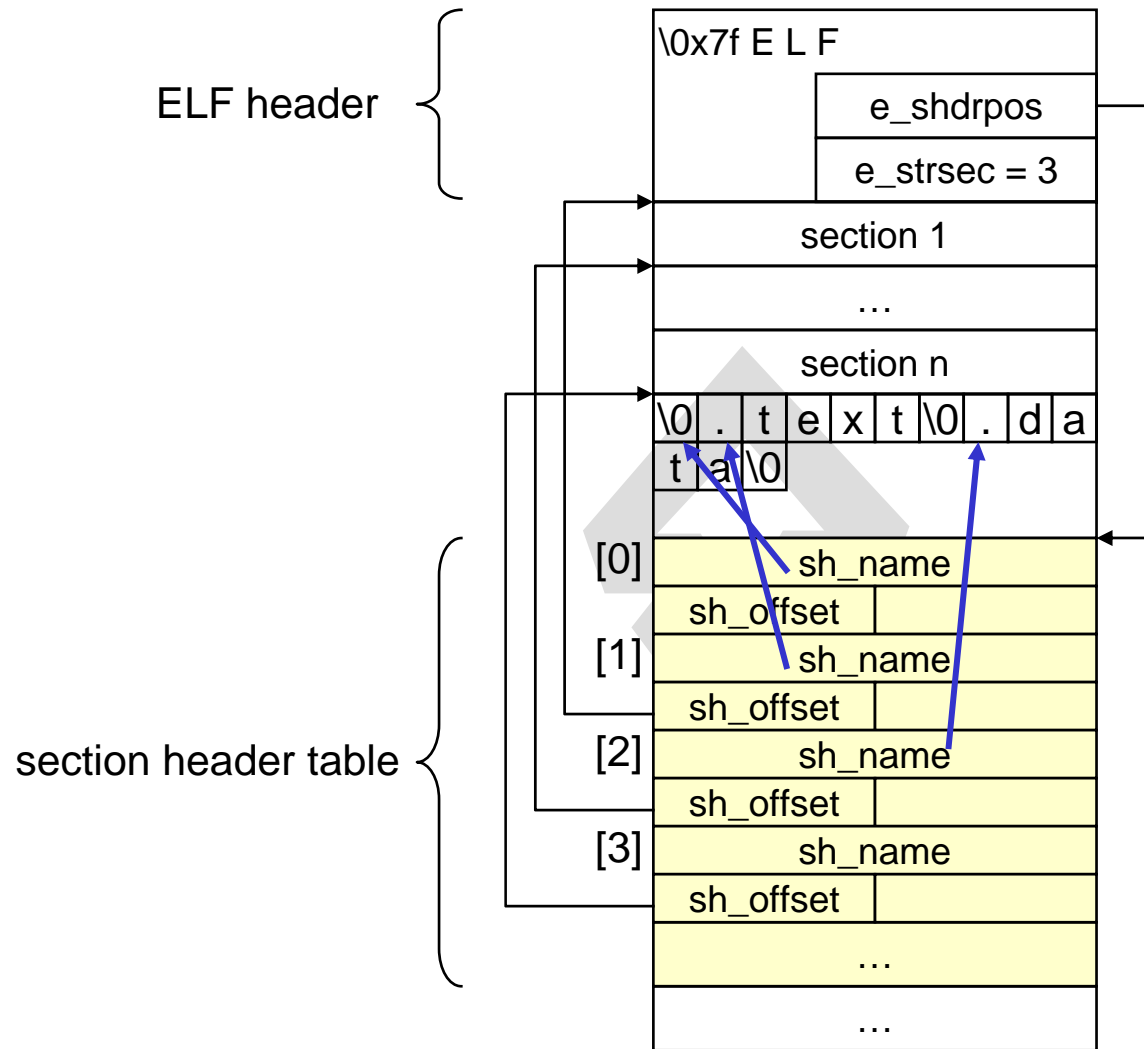
0						7	8							15
sh_name				sh_type				sh_flags				sh_addr		
sh_offset				sh_size				sh_link				sh_info		
sh_align				sh_entsize										

sh_name	name of the section	sh_size	section size
sh_type	categorizes the section's contents and semantics	sh_link	section header table index link
sh_flags	section attribute (bit flag)	sh_info	extra information
sh_addr	address at which the section's first byte should reside	sh_align	address alignment
sh_offset	file offset of the section in the file	sh_entsize	size of entries in the section if it holds a table of fixed-size entries



ELF Section Header(2/2)

- Example





ELF Sections(1/4)

- Section types
 - PROGBITS: program contents such as .text, .data, and debugger info
 - NOBITS: no space is allocated in the file itself. for .bss
 - SYMTAB and DYNsym: symbol tables such as .symtab, and .dynsym
 - STRTAB: string table such as .strtab, and .dynstr
 - REL and RELA: relocation information. .rel.text, .rel.data, and .rel.rodata
 - DYNAMIC and HASH: dynamic linking information and the runtime symbol hash table
- Flag bits
 - ALLOC (memory allocation), WRITE (write perm.), and EXECINSTR (executable)



ELF Sections(2/4)

- .text
 - PROGBITS type, ALLOC+EXECINSTR attribute.
 - Contains 'text'(executable instructions).
- .data
 - PROGBITS type, ALLOC+WRITE attribute
 - contains the initialized data
- .rodata
 - PROGBITS type, ALLOC attribute
 - contains read-only data
- .bss(Block Storage Start from IBM704 asm)
 - NOBITS type, ALLOC+WRITE attribute.
 - contains uninitialized data.
- .rel.text
 - REL or RELA type.
 - Relocation info for .text section
 - Address of instruction that will needed to be modified in the executable



ELF Sections(3/4)

- .rel.data
 - REL or RELA
 - Relocation info for .data section
 - Address of pointer data that will need to be modified in the merged executable
- .init and .fini
 - PROGBITS type, ALLOC+EXECINSTR attribute.
 - Similar to .text
 - Code to be executed when the program starts up or terminates.
 - C and fortran don't need, but essential for C++.
- .symtab and .dynsym
 - SYMTAB and DYNsym types respectively.
 - contains symbol tables
 - the dynamic linker symbol table is ALLOC type.
- .strtab and .dynstr
 - STRTAB type.
 - tables of name strings for a symbol table or the section names for section table.
- .dynstr section is ALLOC type.



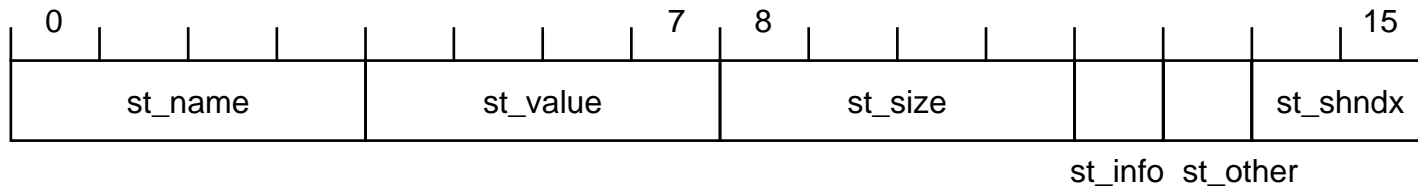
ELF Sections(4/4)

- `.interp` section
 - Contains the name of the program to use as an interpreter
 - Ex. `"/lib/ld-linux.so"` for dynamically linked executables
 - Same concept with the self-running interpreted text files (shell scripts, and etc.)
- `.got`(global offset table), `.plt`(procedure linkage table)
 - used for dynamic linking
- `.debug`
 - contains symbols for the debugger
- `.line`, `.comment`
- `.hash`
 - symbol hash table



ELF Symbol Table

- SYMTAB or DYNsym section
- Symbol table entry format



st_name	index into the object file's symbol string table (.strtab)	st_info	symbol binding(local, global, or weak) and type (data, functions ...)
st_value	value of the associated symbol	st_other	no defined meaning currently
st_size	size associated with the symbol	st_shndx	relevant section header table index



ELF Program Header

- Program header format (only for executable object)

0		7	8		15
p_type		p_offset		p_vaddr	p_paddr
p_filesz		p_memsz		p_flags	p_align

p_type	Type of the segment	p_filesz	The number of bytes in the file image of the segment
p_offset	The offset from the beginning of the file at which the first byte of the segment reside.	p_memsz	The number of bytes in the memory image of the segment
p_vaddr	The virtual address at which the first byte of the segment resides in memory	p_flags	Flags relevant to the segment
p_paddr	Physical address of the segment	p_align	address alignment

```

[root@localhost skl# gcc -DMODULE -o sk.o sk.c
/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/../../../../crt1.o(.text+0x18): In function `_start':
./sysdeps/i386/elf/start.S:77: undefined reference to `main'
/tmp/ccQ8u36v.o(.text+0xf): In function `init_module':
: undefined reference to `printk'
/tmp/ccQ8u36v.o(.text+0x2c): In function `cleanup_module':
: undefined reference to `printk'
collect2: ld returned 1 exit status
[root@localhost skl# gcc -DMODULE -v -c sk.c
Reading specs from /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/specs
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --enable-shared --enable-threads=posix --disable-checking --with-system-zlib --enable-__cxa_atexit --host=i386-redhat-linux
Thread model: posix
gcc version 3.2.2 20030222 (Red Hat Linux 3.2.2-5)
 /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/cc1 -lang-c -v -D__GNUC__=3 -D__GNUC_MINOR__=2 -D__GNUC_PATCHLEVEL__=2 -D__GXX_ABI_VERSION=102 -D__ELF__ -Dunix -Dgnu_linux__ -Dlinux -D__ELF__ -D__unix__ -Dgnu_linux__ -Dlinux__ -Dunix -Dlinux -Asystem=posix -D__NO_INLINE__ -D__STDC_HOSTED__=1 -Acpu=i386 -Amachine=i386 -Di386 -D__i386__ -D__tune_i386__ -DMODULE sk.c -quiet -dumpbase sk.c -version -o /tmp/ccgUDQMd.s
GNU CPP version 3.2.2 20030222 (Red Hat Linux 3.2.2-5) (cpplib) (i386 Linux/ELF)
GNU C version 3.2.2 20030222 (Red Hat Linux 3.2.2-5) (i386-redhat-linux)
    compiled by GNU C version 3.2.2 20030222 (Red Hat Linux 3.2.2-5).
ignoring nonexistent directory "/usr/local/include"
ignoring nonexistent directory "/usr/i386-redhat-linux/include"
#include "...": search starts here:
#include <...> search starts here:
 /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/include
 /usr/include
End of search list.
as -V -Qy -o sk.o /tmp/ccgUDQMd.s
GNU assembler version 2.13.90.0.18 (i386-redhat-linux) using BFD version 2.13.90.0.18 20030206
[root@localhost skl# ls
app app.c app_header.txt gcc_app.txt sk.c sk.o sk_header.txt
[root@localhost skl#

```



```
[root@localhost sk]# gcc -v -o app app.c
Reading specs from /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/specs
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --enable-shared --enable-threads=posix --disable-checking --with-system-zlib --enable-__cxa_atexit --host=i386-redhat-linux
Thread model: posix
gcc version 3.2.2 20030222 (Red Hat Linux 3.2.2-5)
/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/cc1 -lang-c -v -D__GNUC__=3 -D__GNUC_MINOR__=2 -D__GNUC_PATCHLEVEL__=2 -D__GXX_ABI_VERSION=102 -D__ELF__ -Dunix -D__gnu_linux__ -Dlinux -D__ELF__ -D__unix__ -D__gnu_linux__ -D__linux__ -D__unix__ -D__linux__ -Dsystem=posix -D__NO_INLINE__ -D__STDC_HOSTED__=1 -Acpu=i386 -Amachine=i386 -Di386 -D__i386__ -D__i386__ -D__tune_i386__ app.c -quiet -dumpbase app.c -version -o /tmp/ccCBCVMe.s
GNU CPP version 3.2.2 20030222 (Red Hat Linux 3.2.2-5) (cppplib) (i386 Linux/ELF)
GNU C version 3.2.2 20030222 (Red Hat Linux 3.2.2-5) (i386-redhat-linux)
        compiled by GNU C version 3.2.2 20030222 (Red Hat Linux 3.2.2-5).
ignoring nonexistent directory "/usr/local/include"
ignoring nonexistent directory "/usr/i386-redhat-linux/include"
#include "...": search starts here:
#include <...> search starts here:
  /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/include
  /usr/include
End of search list.
as -V -Qy -o /tmp/ccdDfKwj.o /tmp/ccCBCVMe.s
GNU assembler version 2.13.90.0.18 (i386-redhat-linux) using BFD version 2.13.90.0.18 20030206
/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/collect2 --eh-frame-hdr -m elf_i386 -dynamic-linker /lib/ld-linux.so.2 -o app /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/../../../../crt1.o /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/../../../../crti.o /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/crtbegin.o -L/usr/lib/gcc-lib/i386-redhat-linux/3.2.2 -L/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/../../../../tmp/ccdDfKwj.o -lgcc -lgcc_eh -lc -lgcc -lgcc_eh /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/crtend.o /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/../../../../crtfn.o
[root@localhost sk]# ls
app app.c app_header.txt gcc_app.txt sk.c sk.o sk_header.txt
[root@localhost sk]#
```