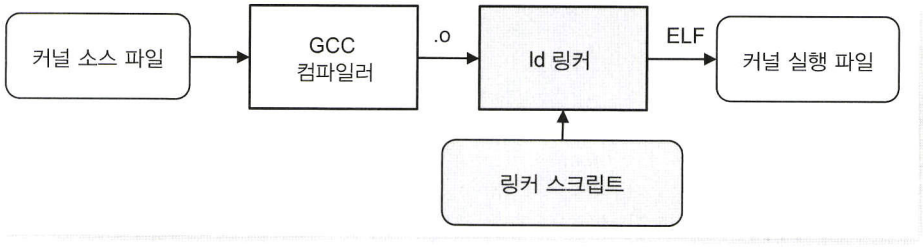
APPENDIX E

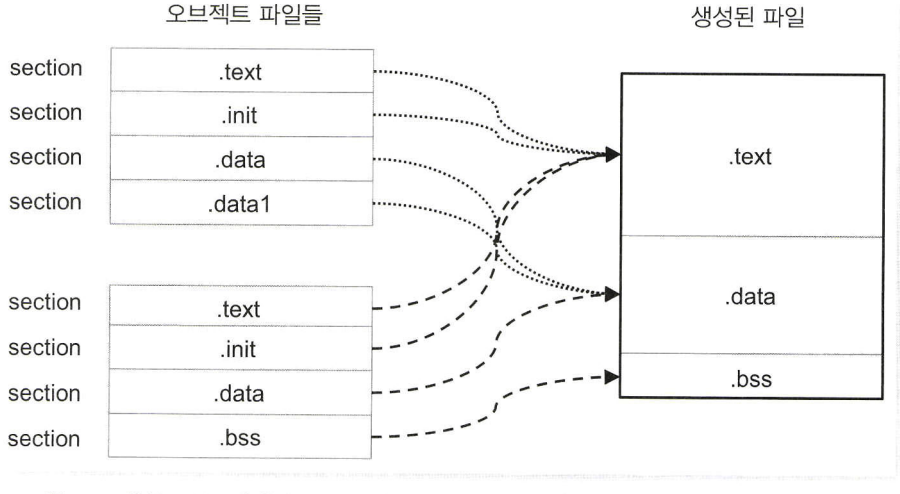
링커 스크립트 파일의 구조

이번 장에서는 링커 스크립트 파일의 구조에 대해서 알아볼 것이다. 링커 스크립트 파일은 리눅스 소스를 컴파일하여 커널 이미지를 생성할 때 링커가 참조하는 파일로, 섹션들을 어떻게 배치할 것인지에 대한 정보를 가지고 있다. 그림 E-1과 같이 소스 파일을 컴파일하여 오브젝트 파일이 생성되면 링커에 의해서 오브젝트 파일들이 하나로 통합되어 실행 파일이 생성된다. 바로 링커가 오브젝트 파일을 실행 파일로 통합할 때 어떻게 합칠 것인지에 대한 내용을 담고 있다.



:: 그림 E-1 **컴파일 과정에서 링커와 링커 스크립트 파일**

커널 소스 파일들은 많은 파일로 구성되어 있어 파일들이 컴파일되면 파일당 오브젝트 파일이 생성된다. 오브젝트 파일들은 특정한 속성들을 가지는 섹션(section)들로 구성되어 있고, 링커가 동일한 특성을 가지는 섹션을 링커 디스크립트 파일에 기술된 내용을 기반으로 구성하게 된다. 그림 E-2는 오브젝트 파일의 섹션과 링커에 의해 생성되는 최종 산출물 간 관계를 보여주고 있다.



:: 그림 E-2 **섹션으로 구성된 오브젝트 파일들과 링커에 의해 생성된 파일**

E.1 ARM 링커 스크립트 파일 구성

리눅스의 링커 스크립트 파일은 ARM의 경우 vmlinux.lds.S 파일과 vmlinux.lds.h 파일을 이용해서 구성하는데, 리눅스의 링커 스크립트 파일인 vmlinux.lds.S는 다음과 같이 정의되어 있다.

:: arch/arm/kernel/vmlinux.lds.S

```
#include <asm-generic/vmlinux.lds.h>

...

OUTPUT_ARCH(arm)
ENTRY(stext)

#ifndef __ARMEB__
jiffies = jiffies_64;
#else
jiffies = jiffies_64 + 4;
#endif

SECTIONS
{
#ifdef CONFIG_XIP_KERNEL
    . = XIP_VIRT_ADDR(CONFIG_XIP_PHYS_ADDR);
#else
    . = PAGE_OFFSET + TEXT_OFFSET;
#endif
    .text.head : {
        _stext = .;
        _sinittext = .;
        *(.text.head)
    }

    .init : {                       /* Init code and data */                ❶
        INIT_TEXT
        _einittext = .;
        __proc_info_begin = .;
            *(.proc.info.init)
    ...

    }

    ...

    .text : {                   /* Real text segment */                     ❷
        _text = .;              /* Text and read-only data */
            __exception_text_start = .;
            *(.exception.text)
            __exception_text_end = .;
```

```
    ...

    RODATA                                                          ❸

    _etext = .;                     /* End of text and rodata section */

    ...

    .data : AT(__data_loc) {                                        ❹
        _data = .;                  /* address in memory */

    ...

        _edata = .;
    }
    _edata_loc = __data_loc + SIZEOF(.data);

    .bss : {                                                        ❺
        __bss_start = .; /* BSS                                */
        *(.bss)
        *(COMMON)
        _end = .;
    }

    ...

}

    ...
```

코드 섹션 ❶은 init 섹션이 커널 이미지 내에 위치할 곳을 지정하고, 코드 섹션 ❷는 data 섹션이 위치할 곳을 지정한다. 코드 섹션 ❸은 읽기 전용 영역인 rodata 섹션이 위치할 곳을 지정한다. 코드 섹션 ❹와 코드 섹션 ❺는 data 섹션과 bss 섹션이 위치할 곳을 지정한다. rodata 섹션의 위치를 지정하는 RODATA는 vmlinux.lds.h 파일에 다음과 같이 정의되어 있다.

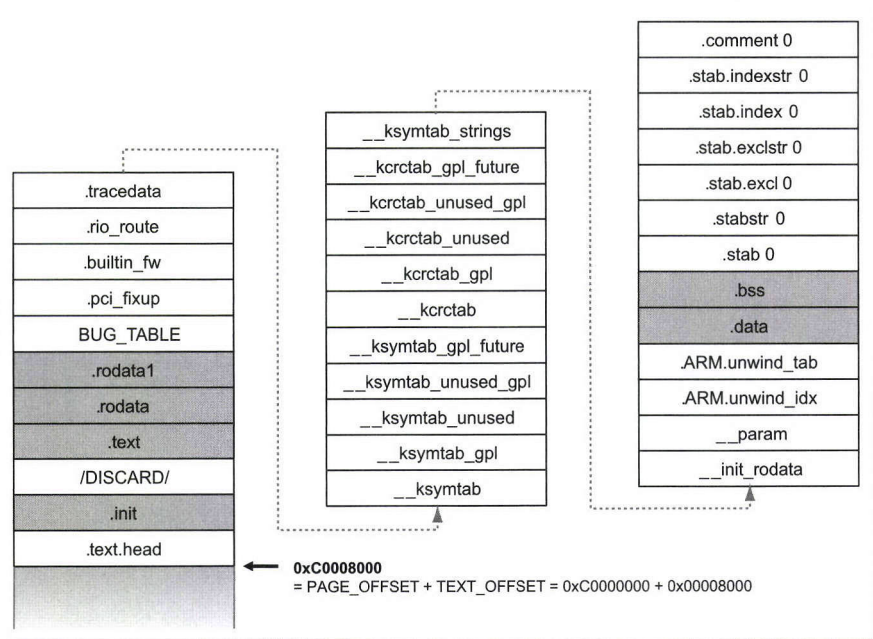
:: asm-generic/vmlinux.lds.h

```
...

#define RO_DATA(align)                                                  \
    . = ALIGN((align));                                                 \
    .rodata         : AT(ADDR(.rodata) - LOAD_OFFSET) {                 \
        VMLINUX_SYMBOL(__start_rodata) = .;                             \
        *(.rodata) *(.rodata.*)                                         \
        *(__vermagic)           /* Kernel version magic */              \
        *(__markers_strings)    /* Markers: strings */                  \
        *(__tracepoints_strings)/* Tracepoints: strings */              \
    }                                                                   \
                                                                        \
```

```
    .rodata1        : AT(ADDR(.rodata1) - LOAD_OFFSET) {                    \
        *(.rodata1)                                                         \
    }                                                                       \
                                                                            \
    BUG_TABLE                                                               \
                                                                            \
    /* PCI quirks */                                                        \
    .pci_fixup        : AT(ADDR(.pci_fixup) - LOAD_OFFSET) {                \
        VMLINUX_SYMBOL(__start_pci_fixups_early) = .;                       \

    ...

    }                                                                       \
    . = ALIGN((align));

#define RODATA RO_DATA(4096)

    ...
```

E.2 링커 스크립트 구성도

vmlinux.lds.S 파일과 vmlinux.lds.h의 간략한 구조를 이전 절에서 알아보았다. 이번 절에서는 이 두 파일을 분석하여 실제 리눅스의 링커 스크립트가 어떻게 구성되어 있는지 구조도를 구성해볼 것이다. 구조도에서 보여주는 섹션들은 모두 vmlinux.lds.S 파일과 vmlinux.lds.h 파일을 분석하여 얻은 결과로 실제 커널 이미지가 어떤 섹션들로 구성되는지를 알 수 있다.



:: 그림 E-3 링커 스크립트 파일 내 섹션 구조