2009년 06월 22일

Map file 분석

음음 . 우리 Linker Output으로 Map file이라는 걸 보았었죠. Map file이라는 건 어떤 걸 담고 있는 걸까요. 뭘 망설이세요. Map file을 열어보면 되지요. 으흐흐 Map file을 열어보면 이거 보통 내기가 아닌 걸 알 수 있을 거에요. - 어떻게 여냐고요. 그냥 notepad로 여세요. 그런데, 엄청 한참 걸릴 거에요. 그 대신 ~ 추천하는 바는 혹시 Ultra Edit라는 Editor를 사용하신다면 이걸 이용하세요. 뭔 원리가 있는지 모르겠지만 금방 열어 제껴 준답니다 -
끝없는 뭔가의 나열 스크롤의 압박. 도대체 내가 필요한 정보를 찾을 수 있기나 한 건지 잘 모르겠다는 비명소리. 이해 합니다. 이해해. 퓹.
실은 엄청난 정보가 들어 있다고 해도 4가지 Format으로 그 정보들을 나열하고 있는 것에 불과한 거에요.

전체 그림을 보면
1. Image Symbol Table
2. Memory Map of the image
3. Image component sizes
4. 전체 Layout

으로 이루어져 있고요. 하나 하나 살펴 보면
Image Symbol Table
ⓐ Linker가 만들어낸 Symbol과 주소 그리고 Region
ⓑ User가 만들어낸 Symbol과 주소 Size 그리고 속해 있는 object Memory Map of the image
ⓒ Scatter Loading에 맞춘 Region에 따라 구획을 나누어서
ⓓ 주소와 Size, type, 그리고 section과 Object. 요놈이 Linker Output Section이 될 거에요.
Image component sizes
ⓔ 각 object 또는 library에 대한 RO, RW, ZI가 차지하는 Size. 요놈이 Linker Input Section이 되겠죠. 전체 Layout
ⓕ 전체적인 Memory에 RO, RW, ZI가 얼마나 차지 하는지에 대한 정보인 거죠. 하나 하나 matching해 볼까요?

```
==============================================================================

Image Symbol Table

ⓐ  Mapping Symbols

    Sym     Value         Execution Region

    $a      0x00000000    BOOT
    $f      0x000117f4    BOOT
    $t      0x000117f8    MAIN_APP_IMAGE
    $d      0x003e3868    MAIN_APP_IMAGE

    Global Symbols

    Symbol Name           Value     Ov Type        Size  Object(Section)

ⓑ   __main                0x00000000   ARM Code       0  bootvector.o

(Int_Vect)
    _main                 0x00000000   ARM Code       0  bootvector.o
```

```
(Int_Vect)
    Image$$BB_RAM$$Length 0x0000000c   Number         0  anon$$obj.o ABSOLUTE
    boot_info_symbol      0x00000020   Data           0  bootvectors.o

(Int_Vect)
    boot_number           0x00000024   Data           0  bootvectors.o

(Int_Vect)

================================================================================

Memory Map of the image

  Image Entry point : 0x00000000

ⓒ Load Region BB_ROM
   (Base: 0x00000000, Size: 0x010c3ab8, Max: 0xffffffff, ABSOLUTE)

ⓒ   Execution Region BOOT
     (Base: 0x00000000, Size: 0x000117f8, Max: 0x0003c000, ABSOLUTE)

    Base Addr    Size         Type   Attr  Idx  E Section Name  Object

ⓓ  0x00000000   0x00000060   Code   RO  7040   * IntVect       bootvectors.o
    0x00001320   0x00000054   Code   RO  6646     .text         io.o
    0x0000c760   0x00000060   Data   RO  6648     .constdata    io.o
    0x00011650   0x00000040   Data   RW  6647     .data         io.o

    Execution Region MAIN_APP
    (Base: 0x000117f8, Size: 0x003d5f8c, Max: 0x00400000, ABSOLUTE)

    Base Addr    Size         Type   Attr  Idx  E Section Name  Object

    0x000117f8   0x000003fc   Code   RO   960     .text         adc.o


================================================================================

Image component sizes


ⓔ   Code    RO Data    RW Data    ZI Data      Debug   Library/Object Name

     14760        564          0          0     144252   controls.lib
     24672      22208       3096          4       2364   ec.lib
      3376         88          0          0      38420   bci.lib
     28140    4033708      45508     230264     805024   graphic_lib.a
     30196       8152       1312        268     278720   security.lib
      1932         92          0          0      83256   db.lib
      4020          9         84     185680   ezTips.a
       364        180          0         16          0   CMC_MN.o
       236          0          0          0          0   KernelDispatcher.o
       758          0          0       2072          0   HALinterrupt.o


    --------------------------------------------------------------------------


================================================================================
ⓕ    Code    RO Data    RW Data    ZI Data      Debug
```

```
   6373664   12805833     569538   15028594  171448836   Grand Totals

==============================================================================
    Total RO  Size(Code + RO Data)             19179497 (18729.98kB)
    Total RW  Size(RW Data + ZI Data)          15598132 (15232.55kB)
    Total ROM Size(Code + RO Data + RW Data)   19749035 (19286.17kB)

==============================================================================
```

오호, 간단하지요? 이걸 보면 Memory에 우리가 만든 것들이 어떻게 자리 잡는지 훤히 다 알 수 있겠어요. 물론 ctrl-F로 찾아보지 않으면 너무 힘들겠지만요. 여하튼 이 구조를 잘 모르고서는 너무나 긴 이 file에는 무슨 내용이 담겨 있는지 알기 힘드니까, 한번 짚고 넘어가는데요. 보통 저 같은 경우에는 ctrl-F로 Image Symbol Table을 찾아서 거기에서 Symbol을 찾아서 어디에 위치하는지 찾을 때가 많고요, 마지막의 전체 Layout에서 Memory양이 맞게 되어 있는지를 찾아볼 때가 많죠.

symbol, linker, file, table, map, format, memory, 원리, output, image, arm, elf