benny5609的专栏

人生,到世上走一遭,只不过是单纯的体验与学习认识，当我们在临死的时候，可以光荣地对自己说:"我已领略过"便不枉此生。

个人资料

benny5609

访问：1141669次
积分：19114
等级：BLOG 7
排名：第314名

原创：810篇 转载：12篇

COFF格式

标签：header compiler numbers struct 平台 table

2007-09-17 12:00 592人阅读 评论(0) 收藏 举报

分类： Windows（170）

COFF – 通用对象文件格式(Common Object File Format)，是一种很流行的对象文件格式(注意:这里不说它是“目标”文件，是为了和编译器产生的目标文件(*.o/*.obj)相区别，因为这种格式不只用于目标文件，库文件、可执行文件也经常是这种格式)。大家可能会经常使用VC吧？它所产生的目标文件(*.obj)就是这种格式。其它的编译器，如

GCC（GNU Compiler Collection）、ICL（Intel C/C++ Compiler）、VectorC，也使用这种格式的目标文件。不仅仅是C/C++，很多其它语言也使用这种格式的对象文件。统一格式的目标文件为混合语言编程带来了极大的方便。

当然，并不是只有这一种对象文件格式。常用格式的还有OMF-对象模型文件（Object Module File）以及ELF-可执行及连接文件格式（Executable and Linking Format）。OMF是一大群IT巨头在n年制定的一种格式，在Windows平台上很常见。大家喜欢的Borland公司现在使用的目标文件就是这种格式。MS和Intel在n年前用的也是这种格式，现在都改投异侧，用COFF格式了。ELF格式在非Windows平台上使用得比较多，在Windows平台基本上没见过。做为程序员，很有必要认识一下这些你经常打交道的家伙！不过这次让我介绍COFF先！

COFF的文件结构

让我们先来看一下COFF文件的整体结构，看看它到底长得什么样！

File Header
Optional Header
Section Header 1
.....
Section Header n
Section Data
Relocation Directives

如左图：

COFF文件一共有8种数据，自上而下分别为：

1. 文件头（File Header）
2. 可选头（Optional Header）
3. 段落头（Section Header）
4. 段落数据（Section Data）
5. 重定位表（Relocation Directives）
6. 行号表（Line Numbers）
7. 符号表（Symbol Table）
8. 字符串表（String Table）

Linux / Unix / Ubuntu (24)
Multithread (31)
Net/ Socket (11)
Ogre (3)
Python/ Perl /Lua (4)
SQLite3 (12)
Steps (0)
SVG / Flash /XML (1)
Technology Translate (4)
Thread (16)
Tips (1)
TTS (17)
UI / Control / File (29)
WinCE (31)
Windows (171)
人物传奇 (5)
server (1)

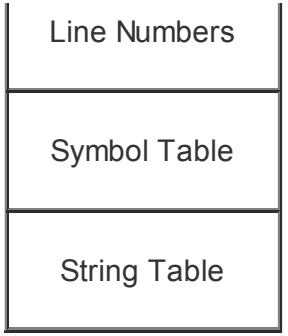
文章存档

2012年05月 (2)
2012年02月 (1)
2012年01月 (1)
2011年12月 (2)
2011年11月 (1)

展开

阅读排行

C语言中#define的用法(转



其中，除了段落头可以有多个节（因为可以有多个段落）以外，其它的所有类型的节最多只能有一个。

文件头：顾名思义，它就是COFF文件的头，它用来保存COFF文件的基本信息，如文件标识，各个表的位置等等。

可选头：再顾名思义，它也是一个头，还是可选的，而且可有可无。在目标文件中，基本上都没有这个头；但在其它的文件中（如：可执行文件）这个段用来保存在文件头中没有描述到的信息。

段落头：又顾……（不顾了，再顾有人要打我了J），这个头（怎么这么多的头啊？！）是用来描述段落信息的，每个段落都有一个段落头来描述。段落的数目在文件头中会指出。

段落数据：这通常是COFF文件中最大的数据段，每个段落真正的数据就保存在这个位置。至于怎么区分这些数据是哪个段落的，不要问我，去问段落头。

重定位表：这个表通常只存在于目标文件中，它用来描述COFF文件中符号的重定位信息。至于为什么要重定位，请回家看看你的操作系统的书籍。

符号表：这个表用来保存COFF文件中所用到的所有符号的信息，连接多个COFF文件时，这个表帮助我们重定位符号。调试程序时也要用到它。

字符串表：不用我说，大家也知道它用来保存字符串的。可是字符串保存给谁看呢？不知道了吧！？问我啊！J符号表是以记录的形式来描述符号信息的，但它只为符号名称留置了8个字符的空间，早期的小程序还将就能行，可在现在的程序中，一个符号名动不动就数十个字符，8个字符怎么能够？没办法，只好把这些名称存在字符串表中。而符号表中只记录这些字符串的位置。

文件的结构大体上就是这样了。长得是丑了点，不过还算它的设计者有点远见。可扩充性设计得不错，以致于沿用至今。了解了文件的整体结构，现在让我们来逐个段落分析它。

(73606)
boost::function用法详解
(28279)
我的ubuntu8.04安装经验 (8081)
MFC中CList类使用注意 (7832)
SQLite 数据库加密的一种 (7295)
VC_CEDIT_SetSel() (7287)
BCGControlBar的使用 (7269)
将CString转换为double((6642)
更多的VC经验 (6478)
为英雄无敌3写个游戏修改 (6395)

评论排行

VC6如何使用VS2005中的 (15)
C语言中#define的用法(转 (12)
boost::function用法详解 (9)
关于auto_ptr_ref的一点问 (9)
SQLite 数据库加密的一种 (5)
BCGControlBar的使用 (5)
动态链接库大总结 (5)
opentelnet.exe 源代码 (5)
SQLite3使用总结 (5)
VC修改应用程序图标(MF (5)

推荐文章

* 程序员10月书讯，评论得书

文件头

文件头，自然是从文件的0偏移处开始，它的结构很简单。用C的结构描述如下：

```
typedef struct {
unsigned short usMagic;  // 魔法数字
unsigned short usNumSec;  // 段落（Section）数
unsigned long ulTime;  // 时间戳
unsigned long ulSymbolOffset;  // 符号表偏移
unsigned long ulNumSymbol;  // 符号数
unsigned short usOptHdrSZ;  // 可选头长度
unsigned short usFlags;  // 文件标记
} FILEHDR;
```

结构中usMagic成员是一个魔法数字（Magic Number），在I386平台上的COFF文件中它的值为0x014c。如果COFF文件头中魔法数字不为0x014c，那就不用看了，这不是一个I386平台的COFF文件。其实这就是一个平台标识。

第二个成员usNumSec是一个无符号短整型，它用来描述段落的数量。段落头（Section Header）的数目就是它。

ulTime成员是一个时间戳，它用来描述COFF文件的建立时间。当COFF文件为一个可执行文件时，这个时间戳经常用来当做一个加密用的比对标识。

ulSymbolOffset是符号表在文件中的偏移量，这是一个绝对偏移量，要从文件头开始计数。在COFF文件的其它节中，也存在这种偏移量，它们都是绝对偏移量。

ulNumSymbol成员给出了符号表中符号记录的数量。

usOptHdrSZ是可选头的长度，通常它为0。而可选头的类型也是从这个长度得知的，针对不同的长度，我们就要选择不同的处理方式。

usFlag是COFF文件的属性标记，它标识了COFF文件的类型，COFF文件中所保存的数据等等信息。

其值如下：

值	名称	说明
0x0001	F_RELFLG	无重定位信息标记。这个标记指出COFF文件中没有重定位信息。通常

* Android中Xposed框架篇---修改系统位置信息实现自身隐藏功能

* Chromium插件(Plugin)模块(Module)加载过程分析

* Android TV开发总结--构建一个TV app的直播节目实例

* 架构设计:系统存储--MySQL简单主从方案及暴露的问题

		在目标文件中这个标记们为0,在可执行文件中为1。
0x0002	F_EXEC	可执行标记。这个标记指出 COFF 文件中所有符号已经解析,COFF 文件应该被认为是可执行文件。
0x0004	F_LNNO	< FONT>文件中所有行号已经被去掉。
0x0008	F_LSYMS	< FONT 无符号标记。此标记说明>文件中的符号信息已经被去掉。
0x0100	F_AR32WR	些标记指出文件是 32 位的 Little-Endian COFF 文件。

注:Little-Endian,记不得它的中文名称了。它是指数据的排列方式。比如:十六进制的0x1234以Little- Endian方式在内存中的顺序为0x34 0x12。与之相反的是Big-Endian,这种方式下,在内存中的顺序是0x12 0x34。

这个表的内容并不全面,但在目标文件中,常用的也就只有这些。其它的标记我将在以后介绍PE格式时给出。

可选头

可选头接在文件头的后面,也就是从COFF文件的0x0014 偏移处开始。长度可以为0。不同长度的可选头,其结构也不同。标准的可选头长度为24或28字节,通常是28啦。这里我就只介绍长度为28的可选头。(因 为这种头的长度是自定义的,不同的人定义的结果就不一样,我只能选一种最常用的头来介绍,别的我也不知道)

这种头的结构如下:

```
typedef struct {
unsigned short usMagic;   // 魔法数字
unsigned short usVersion;   // 版本标识
unsigned long ulTextSize;   // 正文(text)段大小
unsigned long ulInitDataSZ;   // 已初始化数据段大小
unsigned long ulUninitDataSZ;   // 未初始化数据段大小
unsigned long ulEntry;   // 入口点
unsigned long ulTextBase;   // 正文段基址
unsigned long ulDataBase;   // 数据段基址(在PE32中才有)
} OPTHDR;
```

第一个成员usMagic还是魔法数字,不过这回它的值应该为0x010b或0x0107。当值为0x010b时,说明COFF文件

是一个一般的可执行文件；当值为，0x0107时，COFF则为一个ROM镜像文件。

usVersion是COFF文件的版本，ulTextSize是这个可执行COFF的正文段长度，ulInitDataSZ和ulUninitDataSZ分别为已初始化数据段和未初始化数据段的长度。

ulEntry是程序的入口点，也就是COFF载入内存时正文段的位置（EIP寄存器的值），当COFF文件是一个动态库时，入口点也就是动态库的入口函数。

ulTextBase是正文段的基址。

ulDataBase是数据段基址。

其实在这些成员中，只要注意usMagic和ulEntry就可以了。

段落头

段落头紧跟在可选头的后面（如果可选头的长度为0，那么它就是紧跟在文件头后）。它的长度为36个字节，如下

```
typedef struct {
char          cName[8];   // 段名
unsigned long ulVSize;   // 虚拟大小
unsigned long ulVAddr;   // 虚拟地址
unsigned long ulSize;   // 段长度
unsigned long ulSecOffset;   // 段数据偏移
unsigned long ulRelOffset;   // 段重定位表偏移
unsigned long ulLNOffset;   // 行号表偏移
unsigned short ulNumRel;   // 重定位表长度
unsigned short ulNumLN;   // 行号表长度
unsigned long ulFlags;   // 段标识
} SECHDR;
```

这个头可是个重要的头头，我们要用到的最终信息就由它来描述。一个COFF文件可以不要其它的节，但文件头和段落头这两节是必不可少的。

cName 用来保存段名，常用的段名有.text，.data，.comment，.bss等。.text段是正文段，通常也就是代码段；.data是数据段，在这 个数据段中所保存的数据是初始化过的数据；.bss段也可以用来保存数据，不过这里的数据是未初始化的，这个段也是一个空段；.comment段，看名字 也知道，它是注释段，用来保存一些编译信息，算是对COFF文件的注释。

ulVSize是段数据载入内存时的大小。只在可执行文件中有效，在目标文件中总为0。如果它的长度大于段的实际长度，则多的部分将用0来填充。

ulVAddr是段数据载入或连接时的虚拟地址。对于可执行文件来说，这个地址是相对于它的地址空间而言。当可执行文件被载入内存时，这个地址就是段中数据的第一个字节的位置。而对于目标文件而言，这只是重定位时，段数据当前位置的一个偏移量。为了计算方便，便定位的计算简化，它通常设为0。

ulSize这才是段中数据的实际长度，也就是段数据的长度，在读取段数据时就由它来确定要读多少字节。

ulSecOffset是段数据在COFF文件中的偏移量。

ulRelOffset是该段的重定位信息的偏移量。它指向了重定位表的一个记录。

ulLNOffset是该段的行号表的偏移量。它指向的是行号表中的一个记录。

ulNumRel是重定位信息的记录数。从ulRelOffset指向的记录开始，到第ulNumRel个记录为止，都是该段的重定位信息。

ulNumLN和ulNumRel相似。不过它是行号信息的记录数。

ulFlags是该段的属性标识。其值如下表：

值	名称	说明
0x0020	STYP_TEXT	正文段标识，说明该段是代码。
0x0040	STYP_DATA	数据段标识，有些标识的段将用来保存已初始化数据。
0x0080	STYP_BSS	< FONT>有这个标识段也是用来保存数据，不过这里的数据是未初始化数据。

注意，在BSS段中，ulVSize、ulVAddr、ulSize、ulSecOffset、ulRelOffset、ulLNOffset、ulNumRel、ulNumLN的值都为0。（上表只是部分值，其它值在PE格式中介绍，后同）

上一篇 再读PE结构（一）
下一篇 PE学习笔记（一）

我的同类文章

Windows（170）

• ModifyStyle, ModifyStyleEx	2010-03-18	阅读 1082	• 解Issue	2008-09-18	阅读 523
• vc查看宏展开之后的结果的...	2008-09-17	阅读 1220	• Using Windows XP Visual ...	2008-07-06	阅读 985
• 借助VMware实现单机使用...	2008-07-05	阅读 789	• .wmf metafile format	2008-07-02	阅读 755
• VC编译选项	2008-06-23	阅读 514	• 根据文件句柄，获取文件名	2008-06-18	阅读 852
• 半透明渐变的窗口效果	2008-06-18	阅读 917	• 我自己的PE文件RVA-VA-Of...	2008-06-18	阅读 1718
• 困扰80后御宅族的十大烦恼	2008-06-18	阅读 542			

更多文章

猜你在找

《C语言/C++学习指南》数据库篇(MySQL& sqlite)
C++语言基础
虚拟设备驱动程序结构Vxd教程3
编程高手箴言

gameres_中文
ogdev_中文
才女CJ
english comics

social

木沐慕的博客
天涯
案例分析

wince

Windows Mobile 中文社区
norains的专栏Blog
goooglеman的工作日志Blog
51WinCE-CSDN分站Blog
专注 ARM 技术Blog
chen's Blog
Jake Lin's Blog

Visual Studio 2015开发C++程序的基本使用
《C语言/C++学习指南》语法篇（从入门到精通）
汇编C语言C++基础原理系列经典教程
转贴的一篇小技巧汇总
LD说明文档--2LD命令行命令翻译
Iczelion 的 Win32Asm VxD 汇编教程 三

查看评论

暂无评论

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap