Inf 文件简介

（参考和翻译文件来源如下

我看了 MSDN 和 OSR 的解释，加了自己的理解，也翻译了一些内容，也去掉了一些内容，英文好的自己看好了。相信该作者 **Brian Catlin** 是个厉害的角色。

http://www.wd-3.com/archive/InfFiles.htm

MSDN

OSR

翻译 zhoujiamurong

虽然不是俺写的，翻译了好些天，谁叫我初中英语差呢！

zhoujiamurong@163.com 欢迎美女来信，哈哈。

）

Inf 是一个头疼的东西，主要它是一个不用语言而是用配置文件来处理的文件，安装的时候无法调试，(Windbg 可以在驱动中设断点，但不能在.inf 文件中停住)，只有一个日志文件可以参考，而且一个 INF 文件用到了不同的语法，完全看懂 Inf 文件曾经是我的一个梦想，原来我都是 Copy 别人的文件再改改，其实我没有完全弄明白，那好，请我们先沉着迎战，这篇文章是我看过的写的最好的一个讲 Inf 文件的文章，其实我看的好像就只有这一篇，哈哈（不算 MSDN 和 OSR）。

给大脑不大的熊的 Inf 文件

(原文标题很奇怪，看了就知道了，这个可是潜心找到的，不容易找的，真的是可遇不可求啊)

从前的一天，我在百英亩的森林里散步，一个动物过来搭话。

"不好意思"一个小粉猪问道. "但但但是你知道Windows设备驱动，因为，如果你知道，我们的朋友需要一些帮助"，他说，指向一个小熊。(小熊前面有好多修饰词，俺不懂没有翻译，好像是穿着吊带裤，哈哈)

"你需要什么帮助?"，我问这个小熊.

"嗯"，小熊回答道, "我是一个大脑不大的小熊，我写了一个Windows设备驱动，但我不知道如何安装它".

"你所要做的是，写一个.INF 文件,它描述你所控制的驱动和设备，告诉Windows操作系统", 我说. "嗯，顺便问一下,你的驱动控制的是什么设备呢?".

"它是一个蜂蜜(hunny)过滤驱动. 它去掉泥土和蜜蜂和其他的杂物。这可是我的发明。"，它相当骄傲的说（hunny应该是蜂蜜，不过我没有查到这个单词）

"嗯"，我自己想, "这个大脑不大的小熊还不错嘛".

"你想，也就是说，你是否知道如何去写一个这种"电鹰覆文件""，小熊问道.

"当然, 为什么我们不坐下来，我来告诉你关于.INF 文件的所有信息，嗨，都到哪里去了?回来！它们不是那么难！"

一小群动物出现了, 一个接一个, 带着恐惧的眼神。

"别担心, .INF 文件不是你们想象的那么高深", 我说道,试图平复它们的担心.

"W-W唔, 不不不是大大大多数人人说的那样", 小粉猪说道, 明显带着恐惧.

"你先冷静一下我的朋友, 常常不安不能让你成长为一个健康大方的大猪的。", 我说道.

小猪显然冷静下来, 但仍然带着对我的疑虑.

Inf 文件的任务

问一下驱动程序的编写者"什么是写一个 Windows 设备驱动最糟糕的部分？"大多数的回答是，"写一个.Inf 文件"这个原因是 DDK 关于.INF 文件的文档没有提供一个程序性的方法去描述一个.INF 文件，如，"如果你想要实现 X,那么你做 Y 紧接着 Z"。
这篇文章是描述.INF 文件的基本知识和如何在安装一个设备驱动过程中使用它。

Inf 文件可以做很多事情，但 97%的 inf 文件是三个任务
一、**识别特定的设备**。主要是通过硬件 ID 和兼容 ID 号，系统从总线驱动读到的硬件 ID 号（这个是要看 USB Spec）和 Inf 文件中的硬件 ID 号进行比对，当匹配上后，系统知道这个 Inf 文件就是为了这个设备来用的。(匹配不上了，当然不要进行后面的工作)
二、**将文件从介质(如安装光盘)拷贝到系统**。主要是 Sys 文件，当然还有 DLL 和 co-installers 文件、应用程序以及其他文件。
三、**在注册表中添加一个入口**。这是描述设备和它相关的设备，提供设备或驱动指定配置信息，描述给服务驱动管理器的"服务".（这个服务，并不同于一个开机运行的一个应用程序）

就三个任务，简单吧！
请注意两个容易混淆的问题！
一、Inf 文件并不是同一个语言文件一样，从文件开头运行到文件结束，（这个问题就迷惑了我好久）Inf 文件的运行顺序是基于安装阶段的不同。
二、绝大多数的节(Section)是被层次定义好的。（也就是说，这个节的名字是以前某个节定义的，或者系统定义的）

INF 文件的结构

Inf 文件类似 ini 配置文件的做法，使用了一个叫做节的 Section 东西，如同下面
[SectionName]
Inf 文件是不管大小写的，名字中不要有空格, tabs, 其他控制符号, 以及[%];"\ ，有的符号其实在某些情况是可以用的，不过不要没事找事。更不要使用句点符号，（因为它有别的用处，被微软霸占了，微软强抢民女啊），请使用"_"或者"-".
每个节 Section 中有 0 到多个属性（directive，也可以翻译成键，我觉得翻译成属性，好理解点），每个属性可以赋值，使用等号=。使用分号";"作为注释。

而且 Inf 这个文件没有语言中的 If-Elseif-Else 这样的分支语句。那么不同的 Windows 如何解决呢？

```
[SectionName.NT]
```

它使用.NT 表示这个节只能在 Windows2000 或以上才运行。

先前我们曾经说过，Inf 文件并不是按照文件中的顺序来运行的，所以节可以放在任何地方(其实我都按照顺序放的，本来调试都不方便，不想给自己找事，理论上是可以乱序的)，但是有个节是必须在最前面的，它是[Version],有个节一直在最后，它是[Strings]

因为在设备管理器和驱动中要使用一些字符串，所以在 Inf 文件中定义了 [Strings]这个节，这个是最简单了，你只当是 C 语言的宏好了。下面是个例子

```
[Strings]
my_company = "California Death Beams and Stuff"
my_device = "Basselope 2003 Stealth Weapons System"
```

当我们要使用这个字符串的时候，我们就用%%将字符串包起来，如

```
[Manufacturer]
%my_company% = install_my_stuff
```

世界上很多电脑都是用的英语作为它的语言显示，但坦白的说，使用非英语的人们都是喜欢无论什么都用自己的语言来显示所有的东西。你可以在你的 Inf 文件中使用这样的宏定义所有你想要显示的字符串，系统设备安装器会根据你安装系统的语言来选择要显示的字符串。实现这个要使用多[Strings]节,每个 Strings 节需要带一个语言 ID 的后缀.这个语言 ID 是在 SDK 中一样的，是由四个 16 进制数来组成的，其中低 10 位是标识主语言，高 6 位标识子语言。(主语言如中文，子语言如简体中文或繁体中文)，例如，英语的 ID 是 0409，这个语言 ID 的定义在 SDK 中，看一下例子：

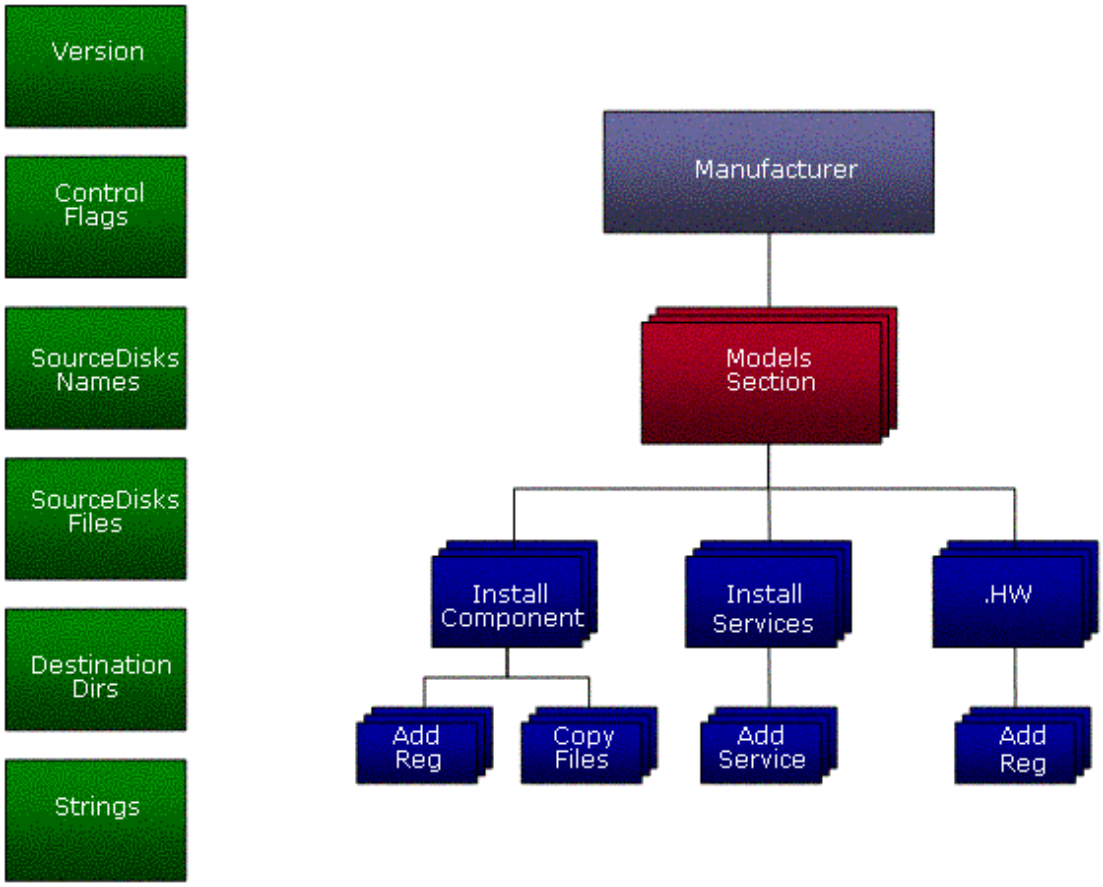
```
[Strings]             ; Default, US English
one = "One"
two = "Two"

[Strings.040c] French (Standard)
one = "Un"
two = "Deux"

[Strings.080a]        ; Spanish (Mexico)
one = "Uno"
two = "Dos"
```

使用上面的[Strings]节，无论你的系统是法语还是墨西哥语等，都会使用一个节来定义两个字符串，如果是没有的语言，他还是会使用默认的语言定义，在实现本地化的过程中， Inf 文件支持多语言经常被提到。(不过比起应用程序本地化来说，简单多了)

通用节

如果你看 DDK 文档的 Inf 文件部分，就会发现系统设备安装器可以理解 39 个节(这个节还在不断的加，不保证正确)。每种类安装器，如 Network, Video，可以加自己的节，或一个存在的节的指令。(这句话我理解不太好，英语好的帮忙翻译下)

39 个节好像有点让人望而却步， 但是，97%的 Inf 文件只是使用大约 12 个节，当然 DDK 里面是 Inf 文件权威的信息。
正如我先前所说的，Inf 文件是层次序列的，如图



（ 图 1 ）
左边一例的绿色的节不是层次的一部分，而是贯穿整个 Inf 文件的全局节。那么层次的根是[Manufacturer]节。(记住了，这个才是命根子啊！)
一些节的名字是需要保留的(因为是系统定义的)，因为这些节是被系统设备安装器调用的(如： Manufacturer, Strings, Version, .Service, .HW, 等.) 那么其他的节，你自己起名字，因为它们只是在你的 Inf 文件中被引用。

“等一下”，小熊说：“说掉了点东西。”
"什么？我们来看 Manufacturer, Models, Install Component, AddReg, CopyFiles, .Services, .HW, Version, Control Flags, SourceDisksNames, SourceDisksFiles, DestinationDirs, 以及 Strings. 嘢, 都在这了.什么掉了?", 我说.

"蜂蜜(Hunny)".

"蜂蜜(Hunny)?",我问道。

"是的",小熊重复道, "你弄掉了蜂蜜(Hunny)".

"嗯… Microsoft在.INF文件中没有一个叫蜂蜜(Hunny)的节",我说道.

"为什么不定义?", 迷惑的小熊问道.

"哦, .INF文件是用于安装设备驱动的，你不能安装一个蜂蜜(Hunny)到你的电脑系统中啊。"

"什么都比不上蜂蜜(Hunny)", 它自负的说道.

"没错, 我....相信是那样, 但是蜂蜜是相当的粘的,而且-"

"粘性是最好的部分", 它打断道.

"好,但我不认为它做电路非常好，而且根据记录，我更愿意你不要安装蜂蜜到我的系统中去！", 我相当坚定的说道.

"OK", 它勉强答应。.

"好了,我可以继续这个课程么?", 我问道.

"好, 请继续", 它回答.

Version

这是.INF 文件标准的头,而且它指定什么样的操作系统和什么样的类安装器可以匹配这个.INF文件.

```
;+
; Identify this .INF file
;-

[Version]
Signature = "$Chicago$"        ; or $Windows NT$ for 2000/XP
Class = Fishbowl ;My class name
ClassGUID = {30320101-C613-11d2-9647-0020AFEB03E0}; My GUID
Provider = %my_company%              ; My company
CatalogFile = Fishbowl.cat           ; WHQL checksum file
DriverVer = 04/01/2003,1.0.121.1     ; Driver date & version
```

Signature 指定什么样的操作系统可以匹配这个.INF文件。$Chicago$指示所有的版本可以匹配 (为什么是芝加哥Chicago?谁知道?我知道芝加哥是Windows 95的代码名称,但是为什么不得不用这个，我说不出来) , 而$Windows NT$指示Windows 2000 和更高的版本。

;只有这$Windows NT$, $Windows 95$, $Chicago$三种值

;Signature Value	Meaning
;$Windows NT$	NT-based operating systems
;$Windows 95$	Windows 9x/Me
;$Chicago$	**All Windows Operation Systems**

Class 指定设备是哪个类的类名. 看DevGUID.H中的一列预定义的类和它们的GUID. 在这个例子中，我创建了自己的类,叫做玻璃鱼缸类. 如果你创建了自己的类名，你就必须也有一个ClassInstall32节(section).

ClassGUID 指定设备类的GUID. 类和类GUID最好一致，否则安装会失败. 如果你创建你自己的类，使用DDK或SDK中GUIDGEN工具创建一个新的GUID. 如果你使用一个已知的类，如：Mouse,那么你到DevGUID.H中去找GUID.

Provider 指定谁写了这个.INF 文件. 一般的,这个是生产这个设备的公司名，但它经常是"Microsoft", 因为他们为厂商写了很多.INF 文件.

CatalogFile 指定驱动包含的数字签名文件和.INF 文件. 这个文件是在设备或驱动公司在通过了WHQL的硬件兼容性测试之后所收到文件和Logo. 开发期间, 你没有一个数字文件，所以你可以省略这个属性(directive)或指向一个空文件.

DriverVer 指定这个驱动是什么时候建的和它的版本号. 当它发现有多个文件时，系统设备安装器使用这个信息去选择使用那个.INF 文件.(请按照它的格式来，否则会识别不出来)

ControlFlags

这个节指定哪个设备在安装的过程中需要特殊的处理.

```
;+
; This section prevents the user from being able to load
; the driver unless the hardware is present
;-

[ControlFlags]
ExcludeFromSelect = *
```

ExcludeFromSelect 是一个最普通的属性在ControlFlags节中. 它包含了一个列表，列表中包括哪些设备被这个.INF 文件安装，存在于可枚举的总线 (这种总线可以标识在它的总线上的每个设备，如PCI 总线，但不是 ISA 总线). 这个属性通知添加硬件向导 (Add Device Wizard) 不允许用户去选择任何指定的设备来手工安装. 这个驱动当设备被 PnP 管理器检测到的时候就加载了. 如果所有的用.INF 文件安装的设备都是在可枚举的总线，那么你可以写一个星号(*),否则，指定每个设备的硬件ID或兼容ID，用逗号分开。 (个人理解：如果是支持PNP的驱动，直接写个*就应该可以了，如果是老式的驱动，就要写好硬件ID和兼容ID).

SourceDisksNames

这个节指定安装磁盘，在安装过程中要将这个磁盘中的文件拷贝到系统中去。(个人理解，一般只是一个虚拟的磁盘)

```
;+
; Identify the installation media
;-

[SourceDisksNames]
1 = "Fishbowl Installation Disk #1"
2 = "Fishbowl Installation Disk #2"
```

这个为SourceDisksNames节定制的属性不同于先前我们讨论的属性. 它的格式是:

disk-id = disk-description[,[tag-file],[unused,path][,flags]]

disk-id 是要给简单的磁盘的序列号. 一般的, 磁盘是从1号开始, 而后每次加一.

disk-description 是一个字符串表示磁盘. 这个字符串将显示给用户，什么时候系统设备安装器需要插入磁盘到系统. 如下图：

tag-file 是一个在那磁盘上已知的唯一文件名。（附上原文，我总觉得这句翻译的不对劲 is the name of a file known to be only on that disk. ）我通常使用记事本在每个磁盘上创建一个长度为零的文件, 如 Disk1, Disk2, 等.

unused 在Windows 2000和以后版本不使用.

path 允许一个在磁盘上指定一个子目录.

flags 允许标志文件被指定一个 .cab文件中的文件名. 在WinXP和后来版本，如果柜(cabinet)文件被使用, 那么标识文件名允许跟标志域.（这个参数你还是看DDK吧，我自己都翻译的晕晕的）

SourceDisksFiles

这个节标识在安装过程中所有的文件都被拷贝到系统中.

```
;+
; Identify the files on the installation media
;-

[SourceDisksFiles]
Fish.hex = 1              ; on disk #1
Fishbowl.sys = 2  ; on disk #2
```

同SourceDisksNames节一样，这个节也是与众不同的. 在这个例子中，每个文件都要拷贝到系统中去，磁盘上有什么文件要标识出来. 这个节联合SourceDisksFiles节一起来工作. 这个属性的格式是:

filename = disk-id[,[sub-directory][,size]]

filename 是一个在一个安装磁盘中的文件名.

disk-id 是一个包含该文件的磁盘的ID号，而且在SourceDisksNames节中列有对应的磁盘.

sub-directory 标识一个文件在磁盘上的子目录.

size 是文件未压缩的字节大小. 一般没有用.

DestinationDirs

这个节指定文件将被拷贝到系统的那个目录，通过提供一个节名的列表和目录数字，节名又包括了文件名列表.

```
;+
; Identify where the files will be copied to
;-

[DestinationDirs]
Fb-W98-Driver-Files = 10,System32\Drivers
Fb-W2K-Driver-Files = 12

[Fb-W98-Driver-Files]
driver.sys
another_file.dat

[Fb-W2K-Driver-Files]
driver.sys
different_file.dat
```

这个节的属性格式是这样的:

section-name = directory-id[,sub-directory]

section-name 是一个.INF文件的编写者创建的一个节的名字. 这个节包含了一个文件名的列表 (它也被列在SourceDisksFiles节中)，文件会被拷贝到指定的目录. 作为选择，保留的关键字"DefaultDestDir"要被使用，它为所有的没有明确指定目的目录的文件指定目的目录.

directory-id 是一个为系统目录预定义的标识. 为驱动写的最常用的目录ID是: 11 = %WinDir%\System32，和 12 = %WinDir%\System32\Drivers， %WinDir%是预定义环境变量，它指向你的系统目录，而不是指向你的strings节中的宏.

(这个里面的目录数字原来把我难住了，我就是不知道12和一些数字是什么目录？气死我了，到处找不到，总不能在百度谷歌中搜索12吧，后来DDK和OSR中才找到，要找的人点下面)

Using Dirids 相关资料

sub-directory 标识一个目录ID对应目录的子目录.

Strings

这个节提供.INF文件要用的字符串的宏定义.

```
;+
; String substitution definitions
;-

[Strings]
ClassName = "Fishbowl"
MfgName = "Sannas Consulting, LLC."
Fishbowl = "Fishbowl USB Demonstration Device"
ServiceDesc = "Fishbowl driver"
```

格式是这样的:

macro-name = "some string"

macro-name 为字符串而定义的宏的名称, 而且它会通过使用百分号包起来在别处被引用,, 如 %macro-name%.

"some string" 是宏的定义, 而且他会替代所有出现在.INF文件中调用的宏.

Manfacturer

这个节非常像一个目录; 它列出了所有的这个.INF文件支持的设备的制造厂商(manufacturer). 对于第三方, 它一般很少列出多个厂商; 然而,微软写的 .INF文件可能在一个文件中支持一打或者更多的厂商. Manufacturer节是剩下的节所组成的系统的根 (看前面的图1).

```
;+
; The root of the hierarchy for this file
;-

[Manufacturer]
%MfgName% = Sannas
```

这个节的语法也不同于在.INF文件中其他的节. 如果你看这个指令, 它显示一个引用的字符串(通过宏替换)给了一个标识符作为它的值. 好, 不是那么回事. 等号标识("=") 不是一个作为赋值操作符使用, 而是作为一个分隔符. 这个节是在安装过程的早期使用的，是当系统设备安装器搜索.INF文件去匹配硬件ID或兼容ID的时候. 这个节的格式是:

manufacturer-name = models-section-name

manufacturer-name 是一个标识厂商的字符串.

models-section-name 是一个节的名字(是由.INF文件的创建者起的)，它包含了一个设备模型的列表，这个.INF文件中支持的厂商的模型.

Models

这个节为.INF支持的特定的厂商标识各种各样的设备模型. 这个名字是由.INF的编写者起的, 而且引用在Manufacturer节.

```
;+
; Supported devices
;-

[our_devices]
%device1-name% = product-install, hardware_ID, hardware_ID
%device2-name% = product-install, compatible_ID
```

在这个节中属性是和我们在Manufacturer节中看到的类似. 这个节的格式是:

device-description = install-section-name, PNP-id[,PNP-id]…

device-description 是一个描述设备的字符串(一般通过宏).

install-section-name 是个.INF文件创建者起的节的名称, 它包含了安装设备驱动的属性.

PNP-id 是一个硬件ID或兼容ID.

在下面的例子里,当为一个特定的设备搜索一个.INF文件时, 系统设备安装器将会在Manufacturer节开始. 它会搜索.INF文件的models节[California_Death_Beams], 这个节包含了.INF文件支持的那个厂商的设备列表. 如果在[California_Death_Beams] models节中没有要找的硬件ID和兼容ID, 它会去到下一个Manufacturer 节的入口.它会搜索.INF文件的[Texas_Techo_Gizmo] models节, 去检测硬件ID 和兼容ID列表. 如果它发现了一个匹配的, 那么它认定这个.INF文件支持这个设备, 然后指定install-section-name节将会为这个设备安装驱动.

```
[Manufacturer]
%cdbs% = California_Death_Beams
%ttg% = Texas_Techno_Gizmo

[California_Death_Beams]
%phaser% = install-phaser, pci\ven_9876&dev_1234
%blaster% = install-blaster, pci\ven_9876%dev_5678

[Texas_Techno_Gizmo]
%bass% = install-bass, dog\type_basselope&model_2003

[install-phaser]

[install-blaster]

[install-bass]
```

```
[install-bass.NT]

[strings]
cdbs = "California Death Beams and Stuff, Inc."
ttg = "Texas Techo Gizmo, LLC."
phaser = "Hand-held phaser, model 1234"
blaster = "Hand-held blaster, model 5678"
bass = "Basselope 2003 Stealth Weapons Platform"
```

InstallComponent

(读到这里请再回头看看图1)

这个节(在DDK文档中也叫做DDInstallSection), 履行请求去安装设备驱动的行为. 这个节的名字是.INF文件编写者起的, 而且在Models中引用.

```
;+
; Install the component
;-

[Fishbowl]
CopyFiles = Fb-W98-Driver-Files
AddReg = AddReg-Win98, AddReg-All

;+
; Identify the driver
;-

[Fb-W98-Driver-Files]
Fishbowl.sys
Fish.hex

;+
; Add entries to the registry - Win98
;-

[AddReg-Win98]
HKR,,DevLoader,,*ntkern
HKR,,NTMPDriver,,Fishbowl.sys

;+
; Add entries to the registry - Win98 and WinNT
;-
```

```
[AddReg-All]
HKR,,DebugLevel,0x00010001,2
HKR,,FirmwareFile,0x00000000,\SystemRoot\System32\Drivers\Fish.he
x
HKR,,AutoloadFirmware,0x00010001,1
```

这个节支持许多不同的属性；最常用的是CopyFiles和AddReg.

*CopyFiles*属性指定一个或者多个被.INF 编写者创建的节的名字，它们包含了一个要拷贝的文件列表 (先前指定在SourceDisksFiles节中).

AddReg属性指定一个或多个被.INF 编写者创建的节的名字，它们包含了一个添加注册表入口的列表.

.Services

在InstallComponent节已经运行,系统设备安装器会搜索.INF 文件中的一个节，这个节的名字和InstallComponent节一样，但带着后缀".Services". 这个.Services节是用于在注册表中的"services"键添加入口，为服务控制管理器描述驱动.

```
;+
; Add our service - WinNT
;-

[Fishbowl.NT.Services]
AddService = Fishbowl, 0x00000002, Add-Service

;+
; Describe the service
;-

[Add-Service]
DisplayName    = %ServiceDesc%
ServiceType    = 1                  ; SERVICE_KERNEL_DRIVER
StartType      = 3                  ; SERVICE_DEMAND
ErrorControl   = 1              ; SERVICE_ERROR_NORMAL
ServiceBinary  = %11%\Drivers\Fishbowl.sys
```

这个节支持许多不同的属性，但最普通的是AddService.

*AddService*属性指定一个由.INF 文件编写者创建的节名，它包含了一串描述驱动的属性.

（添加注册表的位置是HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\%驱动名%）

.HW

这个节主要用于安装PnP过滤驱动. 在.Services节运行后，系统设备安装器会搜索.INF 文件的一个节名，同InstallComponent节名同样, 带一个后缀".HW". 这个.HW节用于给硬件键添加入口(该键位于HKLM\System\CCS\Enum\<enumerator>\<device-id>\<instance-id>).

```
;+
; Add the filter information to the registry
;-

[Snoopy.NT.HW]
AddReg  = AddReg-HW


;+
; Putting this in the registry causes the filter to be
; loaded
;-

[AddReg-HW]
HKR,,"UpperFilters",0x00010000,"Snoopy"
```

这个节有许多不同的属性, 但大部分普通的是AddReg, 它指定一个节(是被.INF 文件的创建者起的)包含一个添加到注册表的入口. 指定键"UpperFilters"去加载你的驱动作为一个上层过滤驱动(在FDO的顶部), 或键"LowerFilters" 去加载你的驱动作为一个下层过滤驱动(FDO的下面).

"OK", 我问成长中的.INF 文件编写者, "有何问题?"

它们说它们都理解了.

"OK, 现在你们该应用你们刚获得的一点知识了. 下面有一些问题，你们应该能够回答",我说道.

提问

读完了这篇文章, 你应该可以回答下面的问题:

在一个.INF文件中什么节是用于安装硬件指定功能过滤驱动?

Manufacturer节用于做什么?
什么类型的设备必须在[ControlFlags]ExcludeFromSelect属性中有它的ID列表? 为什么?
Strings节用于做什么?

什么是你知道的! 这个毛绒绒的小家伙答对了所有的问题, 而且它们溜进了密林深处去进行它们的各种工程. 当小猪走开时, 他还对我保持怀疑, 当它看着他的背影.

"帮我给Oscar Mayer问好!", 我给它说.

关于作者:

Brian Catlin is a Windows driver consultant based on the West Coast. His company, Azius Developer Training, offers training seminars in a variety of subject areas, including driver programming.