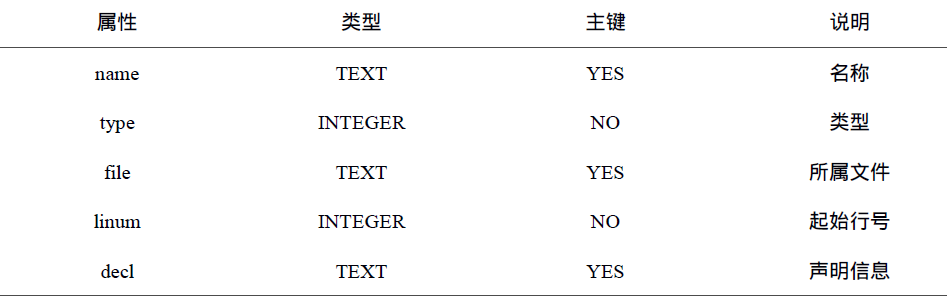
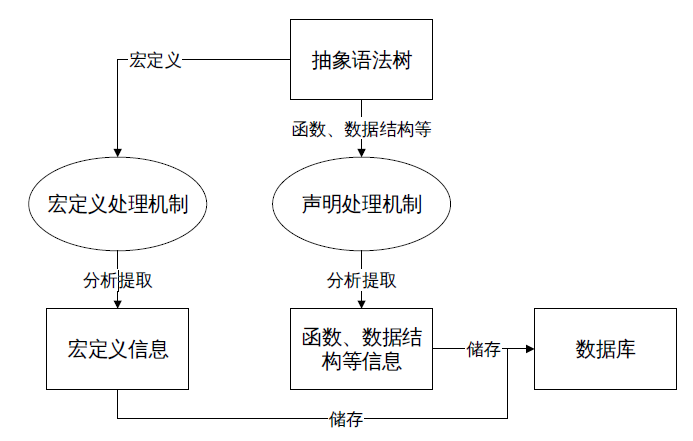
**设备驱动依赖关系及差异分析（王欢）**

## 设备驱动依赖接口分析

根据C 语言的语法规则，接口应该包括函数、宏定义、结构体、联合体、枚举结构、外部引用和类型别名。针对内核接口存储的数据，有五个属性，包括名称、类型、所属文件、起始行号和具体声明信息。如表格所示。



这些信息属于语法分析层次，因此从抽象语法树获取这个信息。



设备驱动接口依赖分析需要区分出哪些接口是自身定义的，哪些是来自内核的。例如最鲜明的函数接口，解决此问题是根据内核接口存储的数据中的属性--所属文件来区分。

## 设备驱动依赖接口差异分析

**五种差异类型：**

ADD 接口只存在于较新版本的内核中

DELETE 接口只存在于较旧版本的内核中

FILE CHANGED 接口所属于的文件发生了变化，但声明没有变化

DECL CHANGED 接口声明发生了变化，但所属于的文件没有变化

ALL CHANGED 接口所属的文件和声明都发生了变化

**差异分析算法：**

算法3-1：接口差异分析算法。

输入：设备驱动依赖的内核接口api，两个版本的内核接口信息API\_OLD 和API\_NEW

输出：ADD，DELETE，DECL\_CHANGED，FILE\_CHANGED 和ALL\_CHANGED

DifferenceAnalysis(api):

1. api\_old = API\_OLD.find(api)

2. api\_new = API\_NEW.find(api)

3. IF(api\_old == ∅ AND api\_new≠∅)

4. ADD = ADD ⋃ api;

5. ELIF(api\_old≠∅ AND api\_new ==∅)

6. DELETE = DELETE ⋃ api;

7. ELIF(api\_old≠∅ AND api\_new≠∅)

8. IF(api\_old.declaration≠api\_new.declaration AND api\_old.file≠api\_new.file)

9. ALL\_CHANGED = ALL\_CHANGED ⋃ api;

10. ELIF(api\_old.declaration≠api\_new.declaration)

11. DECL\_CHANGED = DECL\_CHANGED ⋃ api;

12. ELIF(api\_old.file≠api\_new.file)

13. FILE\_CHANGED = FILE\_CHANGED ⋃ api;

算法中的api\_old 表示从旧版本Linux 内核接口信息中寻找到的目标接口信息，api\_new 表示从新版本Linux 内核接口信息中寻找的目标接口信息。。当api\_old为空，api\_new 不为空时，表示目标接口仅存在于新版本Linux 内核中，所以应该属于ADD 类型的差异，反之则表示仅存在于旧版本Linux 内核中，应该属于DELETE 类型的差异。当新旧两个版本Linux 内核中均存在目标接口时，对比各自的声明信息和所属文件。声明信息和所属文件全部不相同表示目标接口存在ALL\_CHANGED 类型的差异， 声明信息不同而所属文件相同表示存在DECL\_CHANGED 类型的差异，反之表示存在FILE\_CHANGED 类型的差异。

**差异分析的结果存储：**

