Android HAL 設計之迷思

Google在android平台上去調用Linux driver有個HAL的設計,目的就是想要Android平台跟Linux kernel layer切開.基本精神就是利用一個id去獲取某個hardware device driver的控制權.到時候若linux 進版, 只要換掉HAL就行了,而無需去更動Android平台架構.在設計HAL層有兩個很重要的struct一定要知道,分別 hw\_module\_t 和 hw\_device\_t. hw\_module\_t結構裡面記錄著module id和open device的方法. 而hw\_device\_t結構則記錄著hw\_module\_t的資料和版本.所以要開發HAL會需要這兩個資料結構. 在早期的Android 版本就有寫了一個HAL的樣板. 後來要拿來在新版的android上作開發,發現每次要去get hal module時, 系統就會掛掉.從hw\_get\_module搭配Module id的使用一直到註冊hw\_device\_t和hw\_module\_t這兩個資料結構都檢查過了,照理說按規則施工,一定成功阿.後來就去看了一下hw\_get\_module函數的實作,發現更動還蠻大的,多了一個新的hw\_get\_module\_by\_class函數,新的函數多了一個參數,中間多的一個參數算是用來分便同一個driver不同用途.例如audio.**primary**.goldfish.so或是audio.**a2dp**.default.so. 除此之外, hw\_module\_t註冊方式也不同了, 例如

舊版

|  |
| --- |
| const struct cust\_module\_t HAL\_MODULE\_INFO\_SYM = {  common: {  tag: HARDWARE\_MODULE\_TAG,  version\_major: 1,  version\_minor: 0,  id: EFFECTS\_HARDWARE\_MODULE\_ID,  name: "Customize name",  author: "Some body",  methods: &s\_module\_methods,  reserved: {}  },  }; |

新版

|  |
| --- |
| struct cust\_module\_t HAL\_MODULE\_INFO\_SYM = {    .common = {  .tag = HARDWARE\_MODULE\_TAG,  .module\_api\_version = 1,  .hal\_api\_version = 0,  .id = CUST\_HARDWARE\_MODULE\_ID,  .name = "cust module name",  .author = "BoWeiChen",  .methods = &s\_module\_methods,  },  }; |

新版的跟舊版的差別在舊版在宣告HAL\_MODULE\_INFO\_SYM時要用const去修飾,而新版的絕不能用const去修飾, 原因在於新版的android在load函數裡的這一行 hmi = (struct hw\_module\_t \*)dlsym(handle, sym); 這個hmi所指得資料形態是struct hw\_module\_t而非const struct hw\_module\_t. 也就因為如此,拿舊版的HAL樣板套在新版的Android上跑會掛掉.