## [Linux动态频率调节系统CPUFreq之一：概述](http://blog.csdn.net/droidphone/article/details/9346981)

<http://blog.csdn.net/droidphone/article/details/9346981>

### sysfs接口

我们先从CPUFreq提供的sysfs接口入手，直观地看看它提供了那些功能。以下是我的电脑输出的结果：

droidphone@990:~$ cd /sys/devices/system/cpu

droidphone@990:/sys/devices/system/cpu$ ls

cpu0 cpu3 cpu6 cpuidle offline power release

cpu1 cpu4 cpu7 kernel\_max online present uevent

cpu2 cpu5 cpufreq modalias possible probe

所有与CPUFreq相关的sysfs接口都位于：/sys/devices/system/cpu下面，我们可以看到，8个cpu分别建立了一个自己的目录，从cpu0到cpu7，我们再看看offline和online以及present的内容：

droidphone@990:/sys/devices/system/cpu$ cat online

0-7

droidphone@990:/sys/devices/system/cpu$ cat offline

8-15

droidphone@990:/sys/devices/system/cpu$ cat present

0-7

droidphone@990:/sys/devices/system/cpu$

online代表目前正在工作的cpu，输出显示编号为0-7这8个cpu在工作，offline代表目前被关掉的cpu，present则表示主板上已经安装的cpu，由输出可以看到，我的主板可以安装16个cpu（因为intel的超线程技术，其实物理上只是8个），第8-15号cpu处于关闭状态（实际上不存在，因为present只有0-7）。

接着往下看：

droidphone@990:/sys/devices/system/cpu/cpu0$ ls

cache cpuidle microcode power thermal\_throttle uevent

cpufreq crash\_notes node0 subsystem topology

droidphone@990:/sys/devices/system/cpu/cpu0$ cd cpufreq/

droidphone@990:/sys/devices/system/cpu/cpu0/cpufreq$ ls

affected\_cpus related\_cpus scaling\_max\_freq

bios\_limit scaling\_available\_frequencies scaling\_min\_freq

cpuinfo\_cur\_freq scaling\_available\_governors scaling\_setspeed

cpuinfo\_max\_freq scaling\_cur\_freq stats

cpuinfo\_min\_freq scaling\_driver

cpuinfo\_transition\_latency scaling\_governor

droidphone@990:/sys/devices/system/cpu/cpu0/cpufreq$

在我的电脑上，部分的值如下：

cpuinfo\_cur\_freq:   1600000

cpuinfo\_max\_freq:  3401000

cpuinfo\_min\_freq:   1600000

scaling\_cur\_freq:    1600000

scaling\_max\_freq:  3401000

scaling\_min\_freq:   1600000

所以，我的cpu0的最低运行频率是1.6GHz，最高是3.4GHz，目前正在运行的频率是1.6GHz，前缀cpuinfo代表的是cpu硬件上支持的频率，而scaling前缀代表的是可以通过CPUFreq系统用软件进行调节时所支持的频率。cpuinfo\_cur\_freq代表通过硬件实际上读到的频率值，而scaling\_cur\_freq则是软件当前的设置值，多数情况下这两个值是一致的，但是也有可能因为硬件的原因，有微小的差异。scaling\_available\_frequencies会输出当前软件支持的频率值，看看我的cpu支持那些频率：

droidphone@990:/sys/devices/system/cpu/cpu0/cpufreq$ cat scaling\_available\_frequencies

3401000 3400000 3000000 2800000 2600000 2400000 2200000 2000000 1800000 1600000

Oh，从1.6GHz到3.4GHz，一共支持10挡的频率可供选择。scaling\_available\_governors则会输出当前可供选择的频率调节策略：

conservative ondemand userspace powersave performance

一共有5中策略供我们选择，那么当前系统选用那种策略？让我们看看：

dong@dong-990:/sys/devices/system/cpu/cpu0/cpufreq$ cat scaling\_governor

ondemand

OK，我的系统当前选择ondemand这种策略，这种策略的主要思想是：只要cpu的负载超过某一个阀值，cpu的频率会立刻提升至最高，然后再根据实际情况降到合适的水平。详细的情况我们留在后面的章节中讨论。scaling\_driver则会输出当前使用哪一个驱动来设置cpu的工作频率。

当我们选择userspace作为我们的调频governor时，我们可以通过scaling\_setspeed手工设置需要的频率。powersave则简单地使用最低的工作频率进行运行，而performance则一直选择最高的频率进行运行。

### 软件架构

首先，CPU的硬件特性决定了这个CPU的最高和最低工作频率，所有的频率调整数值都必须在这个范围内，它们用cpuinfo\_xxx\_freq来表示。然后，我们可以在这个范围内再次定义出一个软件的调节范围，它们用scaling\_xxx\_freq来表示，同时，根据具体的硬件平台的不同，我们还需要提供一个频率表，这个频率表规定了cpu可以工作的频率值，当然这些频率值必须要在cpuinfo\_xxx\_freq的范围内。有了这些频率信息，CPUFreq系统就可以根据当前cpu的负载轻重状况，合理地从频率表中选择一个合适的频率供cpu使用，已达到节能的目的。至于如何选择频率表中的频率，这个要由不同的governor来实现，目前的内核版本提供了5种governor供我们选择。选择好适当的频率以后，具体的频率调节工作就交由scaling\_driver来完成。CPUFreq系统把一些公共的逻辑和接口代码抽象出来，这些代码与平台无关，也与具体的调频策略无关，内核的文档把它称为CPUFreq Core（/Documents/cpufreq/core.txt）。另外一部分，与实际的调频策略相关的部分被称作cpufreq\_policy，cpufreq\_policy又是由频率信息和具体的governor组成，governor才是具体策略的实现者，当然governor需要我们提供必要的频率信息，governor的实现最好能做到平台无关，与平台相关的代码用cpufreq\_driver表述，它完成实际的频率调节工作。最后，如果其他内核模块需要在频率调节的过程中得到通知消息，则可以通过cpufreq notifiers来完成。由此，我们可以总结出CPUFreq系统的软件结构如下：

### cpufreq\_policy

一种调频策略的各种限制条件的组合称之为policy，代码中用cpufreq\_policy这一数据结构来表示：