# 电源

## Linux电源类（Linux power supply class）

参考

power\_supply.h (include\linux) 、

power\_supply\_core.c (drivers\power)、

power\_supply\_sysfs.c (drivers\power) 、

power\_supply\_class.txt (Documentation\power)

### 概述

电源类提供了电池、UPS、AD、DC等电源在用户空间的接口。

该类定义了一些核心属性，这些属性可以通过sysfs或者uenvt接口访问。

每种属性都有特殊的含义和单位，由于这些属性普遍适用于各种电源，但是实际应用中有些电源可能无法提供某些属性，所以这些属性驱动可以不提供。

电源类是可扩展的，允许驱动程序定义其自身需要的属性，抛弃不需要的属性。

它还集成了LED框架，用来表示电池充电/完全充电状态和AC / USB电源线上状态。

### 属性

定义于内核文件：power\_supply\_sysfs.c (drivers\power)

## linux驱动

### 重要数据结构和函数

#### struct power\_supply

定义于power\_supply.h (include\linux)

对应着电源实例。

struct power\_supply {

 const char \*name;//名称，对应于/sys/class/power\_supply/xxx文件夹

 enum power\_supply\_type type;//电源类型，标示该电源为电池、主电源、USB电源等

 enum power\_supply\_property \*properties;//提供的属性

 size\_t num\_properties;// properties数组的大小

 /\*对应着当此电源变化时需要通知的电源模块的name\*/

 char \*\*supplied\_to;

 size\_t num\_supplicants;// supplied\_to数组的大小

/\*读取属性值\*/

 int (\*get\_property)(struct power\_supply \*psy,

 enum power\_supply\_property psp,

 union power\_supply\_propval \*val);

/\*写属性值\*/

 int (\*set\_property)(struct power\_supply \*psy,

 enum power\_supply\_property psp,

 const union power\_supply\_propval \*val);

/\*设置属性为可写的属性\*/

 int (\*property\_is\_writeable)(struct power\_supply \*psy,

 enum power\_supply\_property psp);

/\*外部电源变化时所作的工作\*/

 void (\*external\_power\_changed)(struct power\_supply \*psy);

 void (\*set\_charged)(struct power\_supply \*psy);

 /\* For APM emulation, think legacy userspace. \*/

 int use\_for\_apm;

 /\* private \*/

 struct device \*dev;

 struct work\_struct changed\_work;

 spinlock\_t changed\_lock;

 bool changed;

 struct wake\_lock work\_wake\_lock;

#ifdef CONFIG\_LEDS\_TRIGGERS

 struct led\_trigger \*charging\_full\_trig;

 char \*charging\_full\_trig\_name;

 struct led\_trigger \*charging\_trig;

 char \*charging\_trig\_name;

 struct led\_trigger \*full\_trig;

 char \*full\_trig\_name;

 struct led\_trigger \*online\_trig;

 char \*online\_trig\_name;

 struct led\_trigger \*charging\_blink\_full\_solid\_trig;

 char \*charging\_blink\_full\_solid\_trig\_name;

#endif

};

#### 函数power\_supply\_register

power\_supply\_core.c (drivers\power)

电源驱动注册函数。

int power\_supply\_register(struct device \*parent, struct power\_supply \*psy)

{

 struct device \*dev;

 int rc;

 dev = kzalloc(sizeof(\*dev), GFP\_KERNEL);

 if (!dev)

 return -ENOMEM;

 device\_initialize(dev);

 dev->class = power\_supply\_class;

 dev->type = &power\_supply\_dev\_type;

 dev->parent = parent;

 dev->release = power\_supply\_dev\_release;

 dev\_set\_drvdata(dev, psy);

 psy->dev = dev;

 INIT\_WORK(&psy->changed\_work, power\_supply\_changed\_work);

 rc = kobject\_set\_name(&dev->kobj, "%s", psy->name);

 if (rc)

 goto kobject\_set\_name\_failed;

 rc = device\_add(dev);

 if (rc)

 goto device\_add\_failed;

 spin\_lock\_init(&psy->changed\_lock);

 wake\_lock\_init(&psy->work\_wake\_lock, WAKE\_LOCK\_SUSPEND, "power-supply");

 rc = power\_supply\_create\_triggers(psy);

 if (rc)

 goto create\_triggers\_failed;

 /\*告诉系统电源有变化，并向用户空间发送uenvt \*/

 power\_supply\_changed(psy);

 goto success;

create\_triggers\_failed:

 wake\_lock\_destroy(&psy->work\_wake\_lock);

 device\_del(dev);

kobject\_set\_name\_failed:

device\_add\_failed:

 put\_device(dev);

success:

 return rc;

}

该函数的主要工作是向系统添加电源设备、创建电源属性文件，最后通知系统和用户电源有变化。

#### 函数power\_supply\_changed

power\_supply\_core.c (drivers\power)

向内核通知电源变化，并向用户空间发送uevent，通知用户电源发生变化。

void power\_supply\_changed(struct power\_supply \*psy)

{

 unsigned long flags;

 dev\_dbg(psy->dev, "%s\n", \_\_func\_\_);

 spin\_lock\_irqsave(&psy->changed\_lock, flags);

 psy->changed = true;

 wake\_lock(&psy->work\_wake\_lock);

 spin\_unlock\_irqrestore(&psy->changed\_lock, flags);

 schedule\_work(&psy->changed\_work);

}

### 总结

电源驱动一般需要做以下几个部分的工作：

1. 定义struct power\_supply，该定义可以是全局的或者嵌入到驱动专有数据结构中，实现其结构字段
2. const char \*name;字段

名称

1. enum power\_supply\_type type;电源类型字段
2. enum power\_supply\_property \*properties;属性字段

定义一个全局enum power\_supply\_property类型的数组，数组中保存该电源需要上报的属性。将该数组的首地址赋给properties字段。

1. 实现get\_property函数，用来读取属性值
2. 如果需要写属性值，则实现set\_property函数和property\_is\_writeable函数， property\_is\_writeable函数判断是否需要写权限，是则返回1，否则返回0；
3. 调用函数power\_supply\_register将定义的power\_supply注册到内核中
4. 当电源的参数有变化时，调用power\_supply\_changed函数通知系统和用户。