【代码解析】W3自定义WPS按钮事件

2017年6月27日
21:55

一、如何自定义按钮事件
问题：
如题

解答：
1、通过设备BSP相关的资源文件定义Button，通过可能是一个GPIO口，具体参考W2C的GPIO定义与W3的GPIO定义。

2、在./gpio-button-hotplug.c文件中找到button_map[]，为button取一个名字。如下代码定义了wps_smartcfg。

```
static struct bh_map button_map[] = {
    BH_MAP(BTN_0,          "BTN_0"),
    BH_MAP(BTN_1,          "BTN_1"),
    BH_MAP(BTN_2,          "BTN_2"),
    BH_MAP(BTN_3,          "BTN_3"),
    BH_MAP(BTN_4,          "BTN_4"),
    BH_MAP(BTN_5,          "BTN_5"),
    BH_MAP(BTN_6,          "BTN_6"),
    BH_MAP(BTN_7,          "BTN_7"),
     BH_MAP(BTN_8,          "BTN_8"),
     BH_MAP(BTN_9,          "BTN_9"),
     BH_MAP(KEY_POWER,      "power"),
     BH_MAP(KEY_RESTART,      "reset"),
     BH_MAP(KEY_RFKILL,      "rfkill"),
     //BH_MAP(KEY_WPS_BUTTON,      "wps"),
     BH_MAP(KEY_WPS_BUTTON,      "wps_smartcfg"),
     BH_MAP(KEY_WIMAX,      "wwan"),
};
```

3、点击按钮，触发按钮发送uevent事件，触发过程可参考参考W2C的GPIO定义与W3的GPIO定义，事件内容如下：

```
{{"HOME":"\/","PATH":"\/sbin:\/bin:\/usr\/sbin:\/usr
\/bin","SUBSYSTEM":"button","ACTION":"pressed","BUTTON":"wps_smartcfg","SEEN":"2","SEQNUM"
:"1065"}}
{{"HOME":"\/","PATH":"\/sbin:\/bin:\/usr\/sbin:\/usr
\/bin","SUBSYSTEM":"button","ACTION":"released","BUTTON":"wps_smartcfg","SEEN":"0","SEQNUM
":"1066"}}
```

其中BUTTON的值会写上wps_smartcfg。

4、procd会通过hotplug函数，打开netlink接口，监听uevent事件，代码如下：

```
void hotplug(char *rules)
{
    struct sockaddr_nl nls;
    int nlbufsize = 512 * 1024;

```

```
6     rule_file = strdup(rules);
7     memset(&nls,0,sizeof(struct sockaddr_nl));
8     nls.nl_family = AF_NETLINK;
9     nls.nl_pid = getpid();
10     nls.nl_groups = -1;
11
12     if ((hotplug_fd.fd = socket(PF_NETLINK, SOCK_DGRAM | SOCK_CLOEXEC,
NETLINK_KOBJECT_UEVENT)) == -1) {
13         ERROR("Failed to open hotplug socket: %s\n", strerror(errno));
14         exit(1);
15     }
16     if (bind(hotplug_fd.fd, (void *)&nls, sizeof(struct sockaddr_nl))) {
17         ERROR("Failed to bind hotplug socket: %s\n", strerror(errno));
18         exit(1);
19     }
20
21     if (setsockopt(hotplug_fd.fd, SOL_SOCKET, SO_RCVBUFFORCE,
&nlbufsize, sizeof(nlbufsize)))
22         ERROR("Failed to resize receive buffer: %s\n", strerror(errno));
23
24     json_script_init(&jctx);
25     queue_proc.cb = queue_proc_cb;
26     uloop_fd_add(&hotplug_fd, ULOOP_READ);
27 }
```

第3~22行，创建netlink接口。

第24~26行，将neklink接口加入uloop框架进行监听，

5、收到uevent消息以后，通过hotplug_handle函数进行处理。

```
1 static void hotplug_handler(struct uloop_fd *u, unsigned int ev)
2 {
3     int i = 0;
4     static char buf[4096];
5     int len = recv(u->fd, buf, sizeof(buf), MSG_DONTWAIT);
6     void *index;
7     if (len < 1)
8         return;
9
10     blob_buf_init(&b, 0);
11     index = blobmsg_open_table(&b, NULL);
12     while (i < len) {
13         int l = strlen(buf + i) + 1;
14         char *e = strstr(&buf[i], "=");
15
16         if (e) {
17             *e = '\0';
18             blobmsg_add_string(&b, &buf[i], &e[1]);
19         }
20         i += l;
21     }
```

```
22      blobmsg_close_table(&b, index);
23      hotplug_handler_debug(b.head);
24      json_script_run(&jctx, rule_file, blob_data(b.head));
25  }
```

6、hotplug_handle函数通过配置脚本hotplug.json进行处理，脚本关键点如下：

```
1 □  [ "if",
2         [ "and",
3             [ "has", "BUTTON" ],
4             [ "eq", "SUBSYSTEM", "button" ],
5         ],
6         [ "exec", "/etc/rc.button/%BUTTON%" ]
7     ],
8     [ "if",
9         [ "eq", "SUBSYSTEM",
10            [ "net", "input", "usb", "usbmisc", "ieee1394", "block",
"atm", "zaptel", "tty", "button" ]
11         ],
12         [ "exec", "/sbin/hotplug-call", "%SUBSYSTEM%" ]
13    ],
```

第1~7行，如果有BUTTON项并且SUBSYSTEM等于button，执行脚本/etc/rc.button/*，这里会执行/etc/rc.button/wps_smartcfg应用。

第8~13行，如果SUBSYSTEM有button，则通过/sbin/hotplug-call botton，去执行/etc/hotplug.d/button目录下wps_smartcfg的脚本

Tips：

不论是/etc/rc.button/或/etc/hotplug.d/下脚本，脚本中都必须带#!/bin/sh，否则无法指定脚本内容。