一.注册监控

```
gboolean register_watch()
{
   int fd = inotify_init1(IN_NONBLOCK);  //初始化inotify

   if (fd == -1)
   {
      return FALSE;
   }

   gchar *watch_path = "/home/liuhw";       //指定监控路径

   unsigned int watch_flag = IN_CREATE | IN_DELETE_SELF | IN_MOVED_TO;    //指定监控信号

   int wd = inotify_add_watch(fd, watch_xdg_path, watch_flag);          //根据监控路径与信号,新建一个inotify
   if (wd == -1)
   {
      close(fd);
      return FALSE;
   }

   GIOChannel *ch = g_io_channel_unix_new(fd);        //根据文件描述符fd,创建通道.
   g_io_add_watch(ch, G_IO_IN, watch_callback, wd);       //watch_callback 为监控到信号时的回调函数.
   return TRUE;
}
```

二.回调函数

```
static gboolean watch_callback(GIOChannel *source, GIOCondition condition, gpointer u)
{
   char buffer[1024];

   int n = read(wd, buffer, sizeof(buffer));
   while (n > 0)
   {
      int off = 0;
      while (off < n)
      {
         struct inotify_event *event = (struct inotify_event *)&buffer[0];  //强制转换为inotify_event 结构体

         if (event->mask & IN_CREATE)
         {
            if (strcmp("itep-power-manager", event->name) == 0)
            {
               g_printf("creat %s", event-name);
            }

         }

         else if (event->mask & IN_MOVED_TO)
         {
            if (strcmp("itep-power-manager", event->name) == 0)
            {
              g_printf("creat %s", event-name);
            }

         }
         off = off + sizeof(*event) + event->len;
      }
      n = read(wd, buffer, sizeof(buffer));
```

```
    }
}
```

三、删除监控

```
inotify_rm_watch(fd, wd);       //注册时的fd 与 wd
g_io_channel_shutdown(source, TRUE, NULL);   //关闭通道，source 为注册时的GIOChannel *ch;
g_io_channel_unref(source);       //释放通道空间；
close(fd);       //关闭文件描述符；
```