4.4 触控屏应用接口

4.4.1 输入子系统简介

连接操作系统的输入设备，可不止一种，也许是一个标准 PS/2 键盘，也许是一个 USB 鼠标，或者是一块触摸屏，甚至是一个游戏机摇杆，Linux 在处理这些纷繁各异的输入设备的时候，采用的办法还是找中间层来屏蔽各种细节，请看下图：

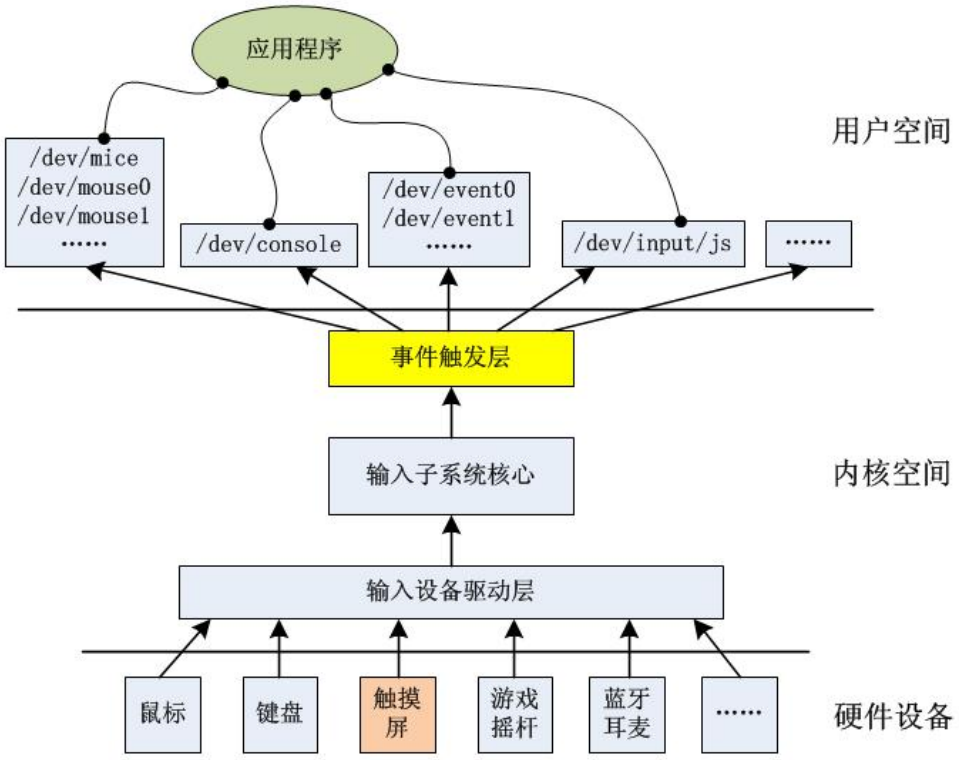

图 4-12 输入子系统

在 Linux 的内核中，对输入设备的使用，实际上运用了 3 大块来管理，他们分别是所谓的输入设备驱动层、输入子系统核心层，以及事件触发层。他们各自的工作分别是：

1，输入设备驱动层：

每一种设备都有其特定的驱动程序，他们被妥当地装载到操作系统的设备模型框架内，封装硬件所提供的功能，向上提供规定的接口。

2，核心层：

此处将收集由设备驱动层发来的数据，整合之后触发某一事件。

3，事件触发层：

这一层是我们需要关注的，我们可以通过在用户空间读取相应设备的节点文件来获知某设备的某一个动作。在最靠近应用程序的事件触发层上，内核所获知的各类输入事件，比如键盘被按了一下，触摸屏被滑了一下等，都将被统一封装在一个叫做 input_even 的结构体当中，这个结构体定义如下：(/usr/include/Linux/input.h)

```
vincent@ubuntu:/usr/include/Linux/$ cat input.h -n
     1  #ifndef _INPUT_H
     2  #define _INPUT_H
```

```
 3
.....
20
21 struct input_event {
22      struct timeval time;
23      __u16 type;
24      __u16 code;
25      __s32 value;
26 };
27
.....
```

该结构体有 4 个成员，其含义分别如下：

一、time：输入事件发生的时间戳，精确到微秒。时间结构体定义如下：

```
struct timeval
{
    __time_t tv_sec; // 秒
    long int tv_usec; // 微秒（1 微秒 = 10-3 毫秒 = 10-6 秒）
};
```

二、type：输入事件的类型。比如：

EV_SYN：事件间的分割标志，有些事件可能会在时间和空间上产生延续，比如持续按住一个按键。为了更好地管理这些持续的事件，EV_SYN 用以将他们分割成一个个的小的数据包。

EV_KEY：用以描述键盘，按键或者类似键盘的设备的状态变化。

EV_REL：相对位移，比如鼠标的移动，滚轮的转动等。

EV_ABS：绝对位移，比如触摸屏上的坐标值。

EV_MSC：不能匹配现有的类型，这相当于当前暂不识别的事件。比如在 Linux 系统中按下键盘中针对 Windows 系统的“一键杀毒”按键，将会产生该事件。

EV_LED：用于控制设备上的 LED 灯的开关，比如按下键盘的大写锁定键，会同时产生 ”EV_KEY” 和 ”EV_LED” 两个事件。

三、code：这个“事件的代码”用于对事件的类型作进一步的描述。比如：当发生 EV_KEY 事件时，则可能是键盘被按下了，那么究竟是哪个按键被按下了呢？此时查看 code 就知道了。当发生 EV_REL 事件时，也许是鼠标动了，也许是滚轮动了。这时可以用 code 的值来加以区分。

四、value：当 code 都不足以区分事件的性质的时候，可以用 value 来确认。比如由 EV_REL 和 REL_WHEEL 确认发生了鼠标滚轮的动作，但是究竟是向上滚还是向下滚呢？再比如由由 EV_KEY 和 KEY_F 确认了发生键盘上 F 键的动作，但究竟是按下呢还是弹起呢？这时都可以用 value 值来进一步判断。

4.4.2 TSLIB 库详解

如果没有 TSLIB 库的支持，虽然我们的确可以直接从/etc/event0 读取触摸屏数据，

但这些没有经过任何处理的粗糙的原始数据离实用性还相距甚远，是无法使用的。

要使用 TSLIB 库，先下载它，网址是：
https://github.com/kergoth/tslib

打开这个网页之后，点击右边的“Download ZIP”按钮，选择保存路径即可。然后解压。你会看到这样的目录结构：

```
vincent@ubuntu:~/tslib-1.4$ tree -L 2 -d
.
├── autom4te.cache
├── CVS
├── etc
│   └── CVS
├── m4
│   ├── CVS
│   ├── external
│   └── internal
├── plugins
│   └── CVS
├── src
│   └── CVS
└── tests
    └── CVS
```

其中，目录 CVS/是版本管理相关的文件，不用管它。从上到下看到，先是有个叫 etc 的目录，里面有个文件是 etc/ts.conf，缺省内容如下：

```
vincent@ubuntu:~/tslib-1.4/etc$ cat ts.conf -n
     1  # Uncomment if you wish to use the Linux
     2  # input layer event interface
     3  # module_raw input
     4
     5  # Uncomment if you're using a Sharp
     6  # Zaurus SL-5500/SL-5000d
     7  # module_raw collie
     8
     9  # Uncomment if you're using a Sharp
    10  # Zaurus SL-C700/C750/C760/C860
    11  # module_raw corgi
    12
    13  # Uncomment if you're using a device with
    14  # a UCB1200/1300/1400 TS interface
    15  # module_raw ucb1x00
    16
```

```
17  # Uncomment if you're using an HP iPaq h3600 or similar
18  # module_raw h3600
19
20  # Uncomment if you're using a Hitachi Webpad
21  # module_raw mk712
22
23  # Uncomment if you're using an IBM Arctic II
24  # module_raw arctic2
25
26  module pthres pmin=1
27  module variance delta=30
28  module dejitter delta=100
29  module linear
```

TSLIB 所支持的各个功能模块是可以以插件的方式被独立加载的，比如去抖、消噪等，这些功能模块被编译成库，并使用 etc/ts.conf 来配置。注意到，凡是以 module 开头的，就是可选的加载模块，且他们的加载顺序按照在 etc/ts.conf 中排列的次序。当你需要某个模块时，只需要将 module 前面的注释符#去掉即可（注意空格也要去掉），比如：

```
module_raw input
module variance delta=30
module dejitter delta=100
module linear
```

模块 module_raw input 是必须要加载的，其他的模块所使用的数据都由他提供。variance 模块用以消除电磁噪音，dejitter 模块用以去抖，linear 模块用于校正，这几个模块按次序加载，最终给到用户的就是可以直接使用的数据了。

plugins 目录就是 TSLIB 支持的插件源码所在地，编译之后，这些插件库会以隐藏文件的方式统一存放在 plugins/.libs 里面。src 无疑是最重要的目录了，里面存放的就是 TSLIB 的核心源代码，包括获取触摸屏文件描述符，加载插件模块，参数解析，读取配置文件信息等等。最后还有一个 tests 目录，里面存放了一些简单的演示代码。

对 TSLIB 有个粗浅的了解之后，接下来就可以编译它了，为了编译成功，请确保你已经在你的系统中安装了以下两个工具：

```
sudo apt-get install automake
sudo apt-get install libtool
```

如果工具都已经安装妥当，下一步就是在 TSLIB 的主目录中直接执行如下命令：

```
vincent@ubuntu:~/tslib-1.4$ ./autogen.sh
```

该脚本将会自动检查当前系统的编译环境信息，最终生成一个 configure 文件，接下来执行这个文件，并给他必要的参数：

```
vincent@ubuntu:~/tslib-1.4$ echo   \
“ac_cv_func_malloc_0_nonnull=yes” > \
arm-linux.cache

vincent@ubuntu:~/tslib-1.4$ ./configure \
--host=arm-none-linux-gnueabi \
--cache-file=arm-linux.cache \
--prefix=/usr/local/tslib
```

请注意：
1，echo 语句是为了避免编译时找不到 ac_cv_func_malloc_0_nonnull 而报错。
2，执行./configure 命令时，各参数含义如下：
--host 指明当前系统所使用的交叉编译工具的前缀，根据具体情况而变。
--cache-file 指明运行 configure 脚本时的缓存文件。
--prefix 指明 TSLIB 的安装路径，根据具体情况而变。

接下来执行 make 和 make install：

```
vincent@ubuntu:~/tslib-1.4$ make
vincent@ubuntu:~/tslib-1.4$ sudo make install
```

这样，在刚才--prefix 所指定的目录下就会出现编译好了的 TSLIB 的文件了：

```
vincent@ubuntu:/usr/local/tslib$ tree -L  2
.
├── bin/
│   ├── ts_calibrate
│   ├── ts_harvest
│   ├── ts_print
│   ├── ts_print_raw
│   └── ts_test
├── etc/
│   └── ts.conf
├── include/
│   └── tslib.h
└── lib/
    ├── libts-0.0.so.0 -> libts-0.0.so.0.1.1
    ├── libts-0.0.so.0.1.1
    ├── libts.la
    ├── libts.so -> libts-0.0.so.0.1.1
    ├── pkgconfig/
    └── ts/
```

这个安装目录下的所有文件，全部拷贝到开发板当中，其中 lib 目录下存放了 TSLIB 库文件，lib/ts 中存放的是各个模块的库。要想正确使用 TSLIB，在开发板中还需要做以下工作：

1，修改 tslib/etc/ts.conf：
找到 ”# module_raw input” 这一行，并将前面的井号和空格去掉。

2，修改/etc/profile：
在该文件的末尾添加如下命令：

```
export TSLIB_ROOT=/tslib/lib
export TSLIB_TSDEVICE=/dev/event0
export TSLIB_FBDEVICE=/dev/fb0
export TSLIB_CONFFILE=/tslib/etc/ts.conf
export TSLIB_PLUGINDIR=/tslib/lib/ts
export TSLIB_CONSOLEDEVICE=none
export TSLIB_CALIBFILE=/tslib/calibration
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/tslib/lib
```

解释一下上面的环境变量：
TSLIB_ROOT 指明 TSLIB 库在开发板中的具体位置，要以实际情况为准。
TSLIB_TSDEVICE 指明开发板触摸屏的设备节点文件名称。
TSLIB_FBDEVICE 指明开发板 LCD 的设备节点文件名称。
TSLIB_CONFFILE 指明 TSLIB 库的配置文件的具体位置，要以实际情况为准。
TSLIB_PLUGINDIR 指明 TSLIB 库的插件模块的具体位置，要以实际情况为准。
TSLIB_CONSOLEDEVICE 指明终端名称，none 意为让系统自动匹配。
TSLIB_CALIBFILE 指明校正文件的位置，该文件在执行 ts_calibrate 之后自动生成。
LD_LIBRARY_PATH 是开发板系统的动态库链接路径

重启开发板，让系统重新读取/etc/profile 文件内容即可。下面是执行 tests/ts_print 并点击触摸屏左上角的效果：

```
[root@Linux /tslib/bin]# ./ts_print
1836.173284:    7    0    132
1836.173284:    7    0    132
1836.204533:    4    2    133
1836.235774:    6    2    133
1836.267034:    4    2    132
1836.298282:    4    2    132
1836.329532:    1    2    131
1836.360771:    4    2    130
1836.392033:    4    2    130
1836.423282:    6    2    130
1836.445390:    5    1      0
```

可以看到，输出的信息是已经经过校正的触摸屏的数据，每一列的信息分别是时间戳、X 轴坐标、Y 轴坐标和压力值。