1、消息机制与触摸处理的机制是类似的最关键的时异步线程的唤醒。当然如果是自己就没必要了

2、`void Looper::wake() {  nWrite = write(mWakeWritePipeFd, "W", 1);}`

3、记得是唤醒。不可能无缘无故就有办法访问另一个线程，只是由另一个线程向当前线程写东西，然后唤醒

4、Handler发送消息的实质是同过Looper来进行的，Looper与相应线程绑定，绑定哪个线程，唤醒哪个。

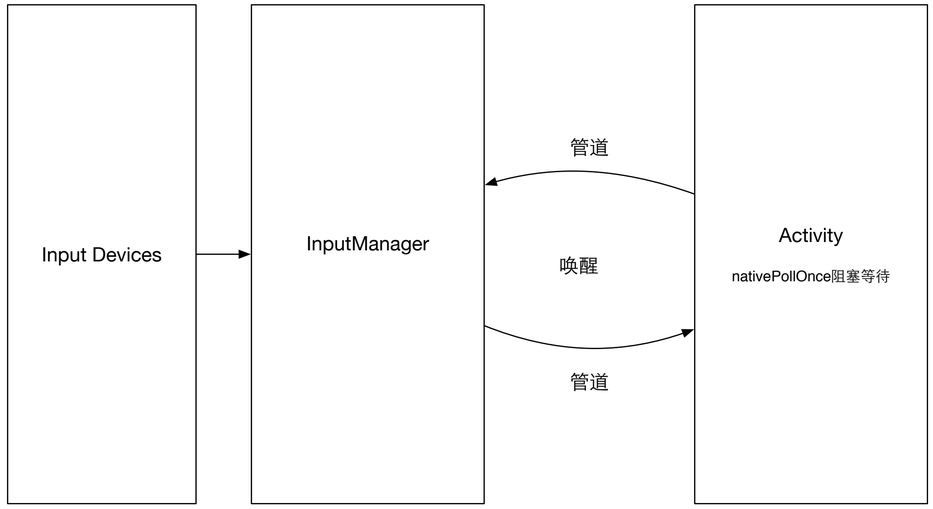
sMainThreadHandler与mH

Looper

MessageQueue

ThreadLocal

如果想要在一个线程中出创建Handler，就必须开启Looper循环



Looper
MessageQueue NativePollOne

ActivityThread
mLooper

epoll_wait
等待其他线程传入
也可以说是等待设备事件

```
1.  int pid = Process.start("android.app.ActivityThread",
2.                  mSimpleProcessManagement ? app.processName : null, uid, uid,
3.                  gids, debugFlags, null);
```

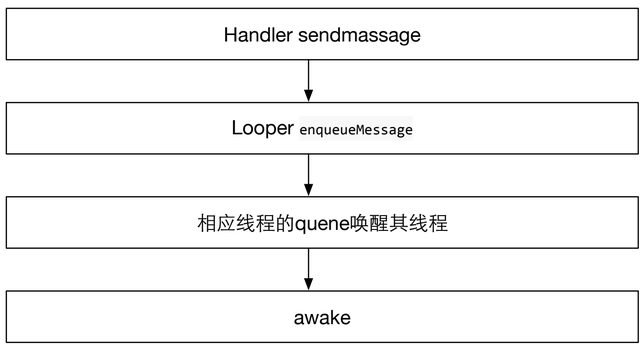
Handler是不是与触摸事件不太一致

Handler发送消息（子线程向主线程发送）
先发送，然后唤醒，不是直接执行，窗口事件的监听是其他线程唤醒主线程

```
final Message next() {
    int pendingIdleHandlerCount = -1; // -1 only during first iteration
    int nextPollTimeoutMillis = 0;

    for (;;) {
        if (nextPollTimeoutMillis != 0) {
            Binder.flushPendingCommands();
        }
        nativePollOnce(mPtr, nextPollTimeoutMillis);
```

Handler发送消息流程



Handler对应Thread于Looper如果一个线程中没有Looper，就不必有Handler

```
1.
2.  enqueueMessage(Message msg, long when) {
3.          ......
4.
5.          final boolean needWake;
6.          synchronized (this) {
7.              ......
8.
9.              msg.when = when;
10.             //Log.d("MessageQueue", "Enqueing: " + msg);
11.             Message p = mMessages;
12.             if (p == null || when == 0 || when < p.when) {
13.                 msg.next = p;
14.                 mMessages = msg;
15.                 needWake = mBlocked; // new head, might need to wake up
16.             } else {
17.                 Message prev = null;
18.                 while (p != null && p.when <= when) {
19.                     prev = p;
20.                     p = p.next;
21.                 }
22.                 msg.next = prev.next;
23.                 prev.next = msg;
24.                 needWake = false; // still waiting on head, no need to wake up
25.             }
26.
27.         }
28.         if (needWake) {
29.             nativeWake(mPtr);
30.
```

可以看出其实和其他InputManager通知的原理是一样的

```
1.  void Looper::wake() {
2.      ......
3.
4.      ssize_t nWrite;
5.      do {
6.          nWrite = write(mWakeWritePipeFd, "W", 1);
7.      } while (nWrite == -1 && errno == EINTR);
8.
9.      .......
10. }
```