新的套接字想要加入到
poll函数中去监听都需要
调用这个函数

virEventPollAddHandle

eventLoop.handles[eventLoop.handlesCount].watch = watch;
eventLoop.handles[eventLoop.handlesCount].fd = fd;
eventLoop.handles[eventLoop.handlesCount].events =
virEventPollToNativeEvents(events);
eventLoop.handles[eventLoop.handlesCount].cb = cb;
eventLoop.handles[eventLoop.handlesCount].ff = ff;
eventLoop.handles[eventLoop.handlesCount].opaque =
opaque;
eventLoop.handles[eventLoop.handlesCount].deleted = 0;

virEventPollInterruptLocked

safewrite(eventLoop.wakeupfd[1], &c, sizeof(c)) != sizeof(c)

每次调用virEventPollAddHandle函数将
新的套接字、回掉函数等信息加入
event_loop中，都会掉一下：
virEventPollInterruptLocked这个函数，这
个伟大的函数回去往poll函数所监听的一个
套接字上发一个字符的信息，这个诡异的
套接字就是在初始化event_loop时加入到
poll所监听的套接字集合里的，当这个诡异
套接字接收到信息以后，触发poll函数返回
继而通过virEventPollDispatchHandles函数
去执行virEventPollHandleWakeup函数，
virEventPollHandleWakeup函数读出刚才
那个伟大的函数写入的那个无聊的一个字节
后就无聊的退出了留下了一片寂寞，but整
个event_loop是个while循环，随着刚才
wakeup函数无聊的结束和留下的那片寂寞
程序进入了下一个轮回，当进入下个轮回后
fds = virEventPollMakePollFDs(&nfds)函数
就会发挥它的神功，将event_loop中的套接字
跟新到poll监听的套接字集合中。
以上解释就是virEventPollInterruptLocked函数
写一个字节到管道，然后virEventPollHandleWakeup
函数将这一个字符读出，这两个表面看似无聊，实际
确意义非凡的操作的原因；

vshEventLoop

while(1)
quit

no

virEventRunDefaultImpl

virEventPollRunOnce

fds = virEventPollMakePollFDs(&nfds)

将加入到event_loop中的
套接字导出到返回值fds中

ret = poll(fds, nfds, timeout)

block在这

poll监听上面这个函数新
导出的套接字们，poll函数
会阻塞在这里，等待所监听
的套接字中至少有一个可读
然后返回，线程继续往下
执行

return

if (ret > 0 &&
virEventPollDispatchHandles(nfds, fds) < 0)
goto error;

执行分发函数，将poll
中监听到有可读信息的
套接字跟evnet_loop.handles
中套接字进行匹配，从匹
配到的event_loop.handle
中取出callback函数去执行

break