在Libev中，使用poll作为backend时，涉及到下面几种数据结构：

int **\***pollidxs**;**

int pollidxmax**;**

struct pollfd **\***polls**;**

int pollmax**;**

int pollcnt**;**

polls就是struct pollfd结构的数组，pollmax是该数组的实际大小，pollcnt表示该数组中，有效结构的个数。也就是监控的描述符个数。

pollidxs是整型数组，以描述符fd为下标，pollidxs[fd]表示该fd对应的struct pollfd结构在polls中的下标。pollidxs[fd] == -1表示该描述符无效。pollidxmax表示pollidxs数组的大小。

因此，上面的结构满足等式：

idx **=** pollidxs **[**fd**];**

polls **[**idx**].**fd **==** fd**;**

1：poll\_init

以poll为backend时，相关的函数和数据结构初始化。代码如下：

backend\_mintime **=** 1e-3**;**

backend\_modify **=** poll\_modify**;**

backend\_poll **=** poll\_poll**;**

pollidxs **=** 0**;** pollidxmax **=** 0**;**

polls **=** 0**;** pollmax **=** 0**;** pollcnt **=** 0**;**

2：poll\_modify

主要设置相关fd的struct pollfd结构，完整代码如下：

static void poll\_modify **(**struct ev\_loop **\***loop**,** int fd**,** int oev**,** int nev**)**

**{**

int idx**;**

**if** **(**oev **==** nev**)**

**return;**

array\_needsize **(**int**,** pollidxs**,** pollidxmax**,** fd **+** 1**,** pollidx\_init**);**

idx **=** pollidxs **[**fd**];**

**if** **(**idx **<** 0**)** /\* need to allocate a new pollfd \*/

**{**

pollidxs **[**fd**]** **=** idx **=** pollcnt**++;**

array\_needsize **(**struct pollfd**,** polls**,** pollmax**,** pollcnt**,** EMPTY2**);**

polls **[**idx**].**fd **=** fd**;**

**}**

assert **(**polls **[**idx**].**fd **==** fd**);**

**if** **(**nev**)**

polls **[**idx**].**events **=**

**(**nev **&** EV\_READ **?** POLLIN **:** 0**)**

**|** **(**nev **&** EV\_WRITE **?** POLLOUT **:** 0**);**

**else** /\* remove pollfd \*/

**{**

pollidxs **[**fd**]** **=** **-**1**;**

**if** **(**expect\_true **(**idx **<** **--**pollcnt**))**

**{**

polls **[**idx**]** **=** polls **[**pollcnt**];**

pollidxs **[**polls **[**idx**].**fd**]** **=** idx**;**

**}**

**}**

**}**

代码还是比较好理解的，首先取得pollidxs [fd]的值，也就是该fd对应的struct pollfd结构在polls数组中的下标。如果该值为-1，说明该fd是第一次加入监控，相应的设置pollidxs[fd]和polls [idx]。

如果env不为0，则设置struct pollfd结构中的events，如果它为0，表示要去除该fd的监控，首先置pollidxs [fd] = -1表示该fd无效，然后去除相应的struct pollfd结构，首先--pollcnt，如果idx == pollcnt，则直接退出即可，否则，将polls数组中的最后一个元素置换到polls[idx]中，并设置相应的pollidxs元素。

3：poll\_poll

主要调用poll函数，等待描述符事件的触发。主要是对触发的描述符和事件调用fd\_event。主要代码如下：

res **=** poll **(**polls**,** pollcnt**,** timeout **\*** 1e3**);**

**if** **(**expect\_false **(**res **<** 0**))**

**{**

**...**

**}**

**else**

**for** **(**p **=** polls**;** res**;** **++**p**)**

**{**

assert **((**"libev: poll() returned illegal result, broken BSD kernel?"**,** p **<** polls **+** pollcnt**));**

**if** **(**expect\_false **(**p**->**revents**))** /\* this expect is debatable \*/

**{**

**--**res**;**

**if** **(**expect\_false **(**p**->**revents **&** POLLNVAL**))**

fd\_kill **(**EV\_A\_ p**->**fd**);**

**else**

fd\_event **(**

EV\_A\_

p**->**fd**,**

**(**p**->**revents **&** **(**POLLOUT **|** POLLERR **|** POLLHUP**)** **?** EV\_WRITE **:** 0**)**

**|** **(**p**->**revents **&** **(**POLLIN **|** POLLERR **|** POLLHUP**)** **?** EV\_READ **:** 0**));**

**}**

**}**