<https://www.cnblogs.com/sduzh/p/6793879.html>

在通过epoll\_ctl(2)向epoll中添加被监视文件描述符时，会将ep\_poll\_callback()作为回调函数添加被监视文件的等待队列中。下面分析ep\_poll\_callback()函数

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | 1004 static int ep\_poll\_callback(wait\_queue\_t \*wait, unsigned mode, int sync, void \*key)  1005 {  1006         int pwake = 0;  1007         unsigned long flags;  1008         struct epitem \*epi = ep\_item\_from\_wait(wait);  1009         struct eventpoll \*ep = epi->ep;  1010         int ewake = 0; |

1008行首先调用ep\_item\_from\_wait()来获取到与被监视文件描述符相关联的结构体struct epitem，获取方法就是利用container\_of宏。

1009行再根据struct epitem的ep字段获取到代表epoll对象实例的结构体struct eventpoll。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | 1012         if ((unsigned long)key & POLLFREE) {  1013                 ep\_pwq\_from\_wait(wait)->whead = NULL;  1014                 /\*  1015                  \* whead = NULL above can race with ep\_remove\_wait\_queue()  1016                  \* which can do another remove\_wait\_queue() after us, so we  1017                  \* can't use \_\_remove\_wait\_queue(). whead->lock is held by  1018                  \* the caller.  1019                  \*/  1020                 list\_del\_init(&wait->task\_list);  1021         } |

判断返回的事件掩码里是否设置了标志位POLLFREE（什么时候会设置该标志？），如果是则将当前等待对象从文件描述符的等待队列中删除(疑问：注释是什么意思？为什么不需要加锁？）。

接下来对epoll的实例加锁：

|  |  |
| --- | --- |
| 1 | 1023         spin\_lock\_irqsave(&ep->lock, flags); |

接下来判断epitem中的事件掩码是不是并没有包括任何poll(2)事件，如果是的话，则解锁后直接返回：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | 1025         /\*  1026          \* If the event mask does not contain any poll(2) event, we consider the  1027          \* descriptor to be disabled. This condition is likely the effect of the  1028          \* EPOLLONESHOT bit that disables the descriptor when an event is received,  1029          \* until the next EPOLL\_CTL\_MOD will be issued.  1030          \*/  1031         if (!(epi->event.events & ~EP\_PRIVATE\_BITS))  1032                 goto out\_unlock; |

什么时候会出现上述情况呢？注释里也说了，就是在设置了EPOLLONESHOT标志的时候。对EPOLLONESHOT标志的处理是在epoll\_wait()的返回过程，调用ep\_send\_events\_proc()的时候，如果设置了EPOLLONESHOT标志则将EP\_PRIVATE\_BITS以外的标志位全部清0：

|  |  |
| --- | --- |
| 1  2 | 1552                         if (epi->event.events & EPOLLONESHOT)  1553                                 epi->event.events &= EP\_PRIVATE\_BITS; |

接下来判断返回的事件里是否有用户真正感兴趣的事件，没有则解锁后返回，否则继续。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | 1034         /\*  1035          \* Check the events coming with the callback. At this stage, not  1036          \* every device reports the events in the "key" parameter of the  1037          \* callback. We need to be able to handle both cases here, hence the  1038          \* test for "key" != NULL before the event match test.  1039          \*/  1040         if (key && !((unsigned long) key & epi->event.events))  1041                 goto out\_unlock; |

如果此时就绪链表rdllist没有被其他进程访问，则直接将当前文件描述符添加到rdllist链表中，否则的话添加到ovflist链表中。ovflist默认值是EP\_UNACTIVE\_PTR，epoll\_wait()遍历rdllist之前会把ovflist设置为NULL，遍历完再恢复为EP\_UNACTIVE\_PTR，因此通过判断ovflist的值是不是EP\_UNACTIVE\_PTR可知此时rdllist是不是正在被访问。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | 1049         if (unlikely(ep->ovflist != EP\_UNACTIVE\_PTR)) {  1050                 if (epi->next == EP\_UNACTIVE\_PTR) {  1051                         epi->next = ep->ovflist;  1052                         ep->ovflist = epi;  1053                         if (epi->ws) {  1054                                 /\*  1055                                  \* Activate ep->ws since epi->ws may get  1056                                  \* deactivated at any time.  1057                                  \*/  1058                                 \_\_pm\_stay\_awake(ep->ws);  1059                         }  1060  1061                 }  1062                 goto out\_unlock;  1063         }  1064  1065         /\* If this file is already in the ready list we exit soon \*/  1066         if (!ep\_is\_linked(&epi->rdllink)) {  1067                 list\_add\_tail(&epi->rdllink, &ep->rdllist);  1068                 ep\_pm\_stay\_awake\_rcu(epi);  1069         } |

如果是描述符是添加到ovflist链表中，说明此时已经有ep\_wait()准备返回了，因此不用再唤醒epoll实例的等待队列，因此1062行直接跳到解锁处；否则的话，则唤醒因为调用epoll\_wait()而等待在epoll实例等待队列上的进程（这里最多只会唤醒一个进程）：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | 1075         if (waitqueue\_active(&ep->wq)) {  1076                 if ((epi->event.events & EPOLLEXCLUSIVE) &&  1077                                         !((unsigned long)key & POLLFREE)) {  1078                         switch ((unsigned long)key & EPOLLINOUT\_BITS) {  1079                         case POLLIN:  1080                                 if (epi->event.events & POLLIN)  1081                                         ewake = 1;  1082                                 break;  1083                         case POLLOUT:  1084                                 if (epi->event.events & POLLOUT)  1085                                         ewake = 1;  1086                                 break;  1087                         case 0:  1088                                 ewake = 1;  1089                                 break;  1090                         }  1091                 }  1092                 wake\_up\_locked(&ep->wq);  1093         } |

如果epoll实例的poll队列非空，也会唤醒等待在poll队列上的进程，不过是在解锁后才会进行唤醒操作。

|  |  |
| --- | --- |
| 1  2 | 1094         if (waitqueue\_active(&ep->poll\_wait))  1095                 pwake++; |

最后解锁并返回：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | 1097 out\_unlock:  1098         spin\_unlock\_irqrestore(&ep->lock, flags);  1099  1100         /\* We have to call this outside the lock \*/  1101         if (pwake)  1102                 ep\_poll\_safewake(&ep->poll\_wait);  1103  1104         if (epi->event.events & EPOLLEXCLUSIVE)  1105                 return ewake;  1106  1107         return 1; |

注意到ep\_poll\_callback()的返回值和EPOLLEXCLUSIVE标志有关，该标志是用来处理这种情况：当多个进程中的不同epoll实例在监视同一个文件描述符时，如果该文件描述符上有事件发生，则所有的epoll实例所在进程都将被唤醒，这样有可能造成“惊群”(thundering herd)。关于EPOLLEXCLUSIVE可以看[这里](https://lwn.net/Articles/667087/)。