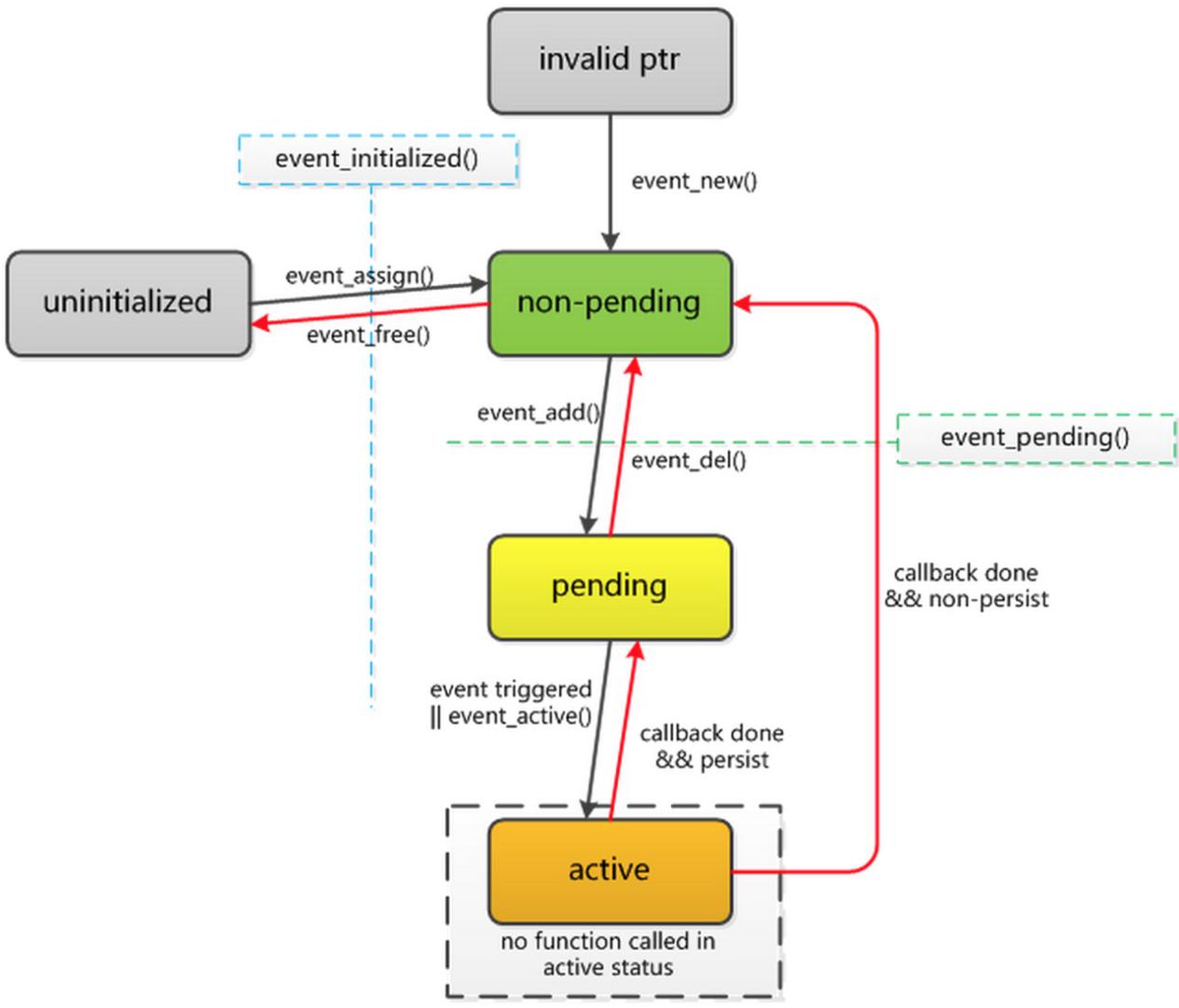
**事件状态**

已初始化状态(initialized)：调用了event\_new后

等待状态(pending)：调用了event\_add之后

激活状态(active)：

待触发事件的触发条件满足之后（如可读可写），在该状态的事件回调函数会被调用，在调用回调函数之前，event会被设为已初始化状态或等待状态（设置了EV\_PERSIST标志时）



**event\_pending**

判断事件是否处于待处理状态

int event\_pending(const struct event \*ev, short what, struct timeval \*tv\_out);

返回值：是>0，否等于0

示例：

ev = event\_new(base, 0, EV\_READ|EV\_PERSIST, cb\_func, (char\*)"Reading");

pending = event\_pending(ev, EV\_READ|EV\_WRITE|EV\_SIGNAL, NULL);

如果事件处于待处理状态，则pending被设置为EV\_READ，应用event\_new设置了EV\_READ，而event\_pending也指定的EV\_READ

**获取event的信息**

如下获取event对应的信息

evutil\_socket\_t event\_get\_fd(const struct event \*ev);

struct event\_base \*event\_get\_base(const struct event \*ev);

short event\_get\_events(const struct event \*ev);

event\_callback\_fn event\_get\_callback(const struct event \*ev);

void \*event\_get\_callback\_arg(const struct event \*ev);

int event\_get\_priority(const struct event \*ev);