第11章 定时器

socket选项SO_RCVTIMEO,SO_SNDTIMEO

表 11-1　SO_RCVTIMEO 和 SO_SNDTIMEO 选项的作用

系统调用	有效选项	系统调用超时后的行为
send	SO_SNDTIMEO	返回 -1，设置 errno 为 EAGAIN 或 EWOULDBLOCK
sendmsg	SO_SNDTIMEO	返回 -1，设置 errno 为 EAGAIN 或 EWOULDBLOCK
recv	SO_RCVTIMEO	返回 -1，设置 errno 为 EAGAIN 或 EWOULDBLOCK
recvmsg	SO_RCVTIMEO	返回 -1，设置 errno 为 EAGAIN 或 EWOULDBLOCK
accept	SO_RCVTIMEO	返回 -1，设置 errno 为 EAGAIN 或 EWOULDBLOCK
connect	SO_SNDTIMEO	返回 -1，设置 errno 为 EINPROGRESS

SIGALRM 信号函数

- 利用信号处理函数处理任务。

IO复用系统调用的超时参数

- 不断更新定时参数反映剩余时间

高性能定时器

- 时间轮
- slot 槽，每次转动称为一个滴答tick，一个滴答的时间称为槽间隔si

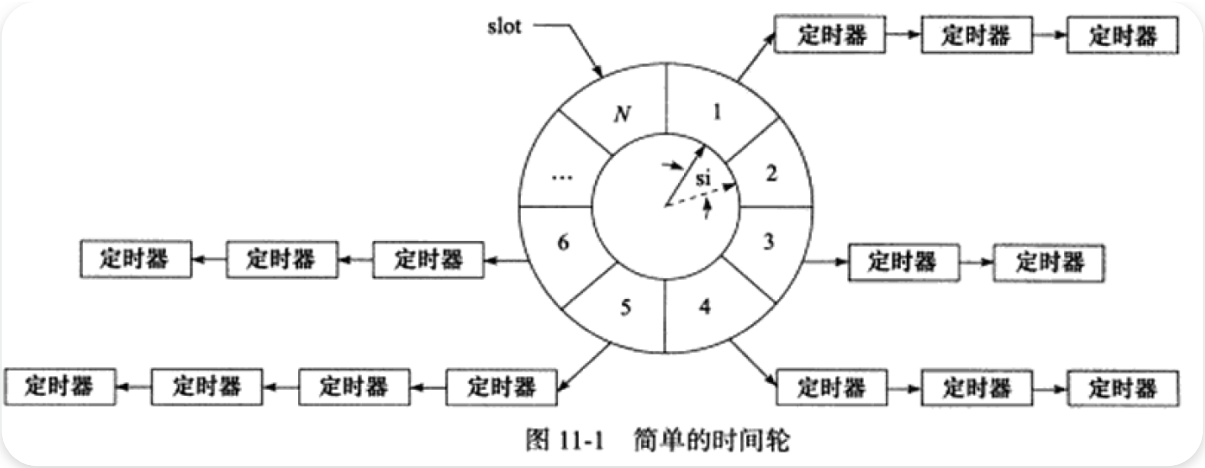

图 11-1　简单的时间轮

- 时间堆：将所有定时器中超时时间最小的定时器作为心搏间隔，一旦心搏函数被调用，时间最小的定时器必然到期，就可以处理该定时器。然后再从剩余的定时器中找到最小的超时时间
  - 堆排序
  - 堆的调整
    - 插入：先在最后一个结点插入一个空结点 如果x放到最后不破坏堆结构，则插入完成。否则进行**上虑操作（即交换空穴与它的父结点，不断执行；**
    - 删除：首先在根结点创建一个空穴 由于堆现在少了一个元素，我们可以将堆的最后一个值x移动到某个地方，若x可以被放入不破坏结构，则删除完成。否则执行下虑操作：**即交换空穴和它的两个子结点的较小者 不断重复**。直到x可以被放入空穴。
  - 数组存储：**数组中任意一个位置i上的元素，左孩子2i+1,右孩子2i+2**，父亲结点则在[(i-1)/2]
  - 初始化已包含N个元素的数组，初始化为一个最小堆
    - 每个插入，效率低
    - 对[(N-1)/2]～0个元素（非叶结点）执行**下虑操作：即交换当前结点和它的两个子结点的较小者 不断重复**
  - 上虑操作：

```
  //从从空穴到根结点进行上虑操作  即若父亲结点大于最后一个结点则交换
while(hole>0){
    parent = (hole-1)/2;
    if(array[parent]->expire<=array[hole]->expire){
        break;
    }
    array[hole] = array[parent];
    hole=parent;
}
array[hole] = timer;

}
```

  - 下虑操作

```
void percolate_down(int hole){
heap_timer* temp = array[hole];
int child = 0;
//hole*2+1 即有左孩子 即有孩子 如果2i+1都大于总数了说明没有孩子了
while((hole*2+1)<=(cur_size-1)){ //进行下虑操作 一直与空洞的值对比 交换空洞（下标）
    child = hole*2+1;//左孩子
    if(child<cur_size-1&&(array[child+1]->expire<array[child]->expire)){ //左右孩子
        child++;
    }
    if(array[child]->expire<temp->expire){ //每次都和temp比 交换空洞 实际只有赋值 下标操
                array[hole] = array[child];
        array[hole] = array[child];
    }
    else{
        break;
    }
    hole = child;
}
array[hole] = temp; //最后把值写入空穴里面
```

- throw std::exception()与 throw(std::exception)