epoll是Linux内核为处理大批量句柄而作了改进的poll，是Linux下多路复用IO接口select/poll的增强版本，它能显著减少程序在大量并发连接中只有少量活跃的情况下的系统CPU利用率。

一、epoll的优点

支持一个进程打开大数目的socket描述符。

IO效率不随FD数目增加而线性下降。

内核微调。

二、epoll的使用

epoll有2种工作方式：LT和ET。

LT（level triggered，水平触发）是缺省的工作方式，并且同时支持block和no-block socket.在这种做法中，内核告诉你一个文件描述符是否就绪了，然后你可以对这个就绪的fd进行IO操作。如果你不作任何操作，内核还是会继续通知你的，所以，这种模式编程出错误可能性要小一点。传统的select/poll都是这种模型的代表。

ET （edge-triggered，边缘触发）是高速工作方式，只支持no-block socket。在这种模式下，当描述符从未就绪变为就绪时，内核通过epoll告诉你。然后它会假设你知道文件描述符已经就绪，并且不会再为那个文件描述符发送更多的就绪通知，直到你做了某些操作导致那个文件描述符不再为就绪状态了（比如，你在发送，接收或者接收请求，或者发送接收的数据少于一定量时导致了一个EWOULDBLOCK 错误）。但是请注意，如果一直不对这个fd作IO操作（从而导致它再次变成未就绪），内核不会发送更多的通知（only once）。

epoll相关的系统调用有3个：epoll_create, epoll_ctl和epoll_wait。在头文件<sys/epoll.h>

1. int epoll_create(int size);

参数size：用来告诉内核要监听的数目一共有多少个。

返回值：成功时，返回一个非负整数的文件描述符，作为创建好的epoll句柄。调用失败时，返回-1，错误信息可以通过errno获得。

说明：创建一个epoll句柄，size用来告诉内核这个监听的数目一共有多大。这个参数不同于select()中的第一个参数，给出最大监听的fd+1的值。需要注意的是，当创建好epoll句柄后，它就是会占用一个fd值，所以在使用完epoll后，必须调用close()关闭，否则可能导致fd被耗尽。

2. int epoll_ctl(int epfd, int op, int fd, struct epoll_event *event);

参数epfd：epoll_create()函数返回的epoll句柄。

参数op：操作选项。

参数fd：要进行操作的目标文件描述符。

参数event：struct epoll_event结构指针，将fd和要进行的操作关联起来。

返回值：成功时，返回0，作为创建好的epoll句柄。调用失败时，返回-1，错误信息可以通过errno获得。

说明：epoll的事件注册函数，它不同与select()是在监听事件时告诉内核要监听什么类型的事件，而是在这里先注册要监听的事件类型。

参数op的可选值有以下3个：

EPOLL_CTL_ADD：注册新的fd到epfd中；

EPOLL_CTL_MOD：修改已经注册的fd的监听事件；

EPOLL_CTL_DEL：从epfd中删除一个fd；

struct epoll_event结构如下：

```
1. typedef union epoll_data {
2.         void *ptr;
3.         int fd;
4.         __uint32_t u32;
5.         __uint64_t u64;
6. } epoll_data_t;
7.
8. struct epoll_event {
9.         __uint32_t events; /* Epoll events */
10.         epoll_data_t data; /* User data variable */
11. };
```

events可以是以下几个宏的集合：

EPOLLIN ：表示对应的文件描述符可以读（包括对端SOCKET正常关闭）；

EPOLLOUT：表示对应的文件描述符可以写；

EPOLLPRI：表示对应的文件描述符有紧急的数据可读（这里应该表示有带外数据到来）；

EPOLLERR：表示对应的文件描述符发生错误；

EPOLLHUP：表示对应的文件描述符被挂断；

EPOLLET： 将EPOLL设为边缘触发(Edge Triggered)模式，这是相对于水平触发(Level Triggered)来说的。

EPOLLONESHOT：只监听一次事件，当监听完这次事件之后，如果还需要继续监听这个socket的话，需要再次把这个socket加入到EPOLL队列里

3. int epoll_wait(int epfd, struct epoll_event * events, int maxevents, int timeout);

参数epfd：epoll_create()函数返回的epoll句柄。

参数events：struct epoll_event结构指针，用来从内核得到事件的集合。

参数　maxevents：告诉内核这个events有多大

参数　timeout: 等待时的超时时间，以毫秒为单位。

返回值：成功时，返回需要处理的事件数目。调用失败时，返回0，表示等待超时。

说明：等待事件的产生。