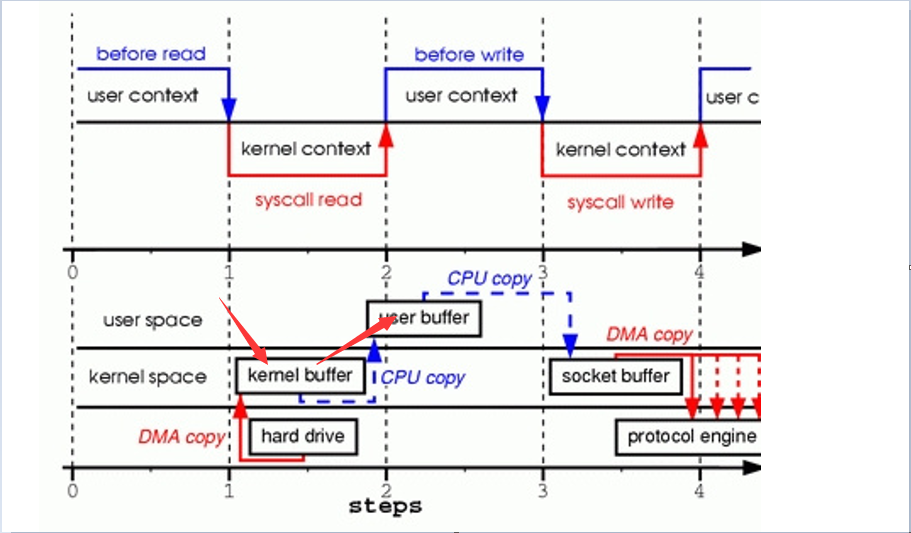
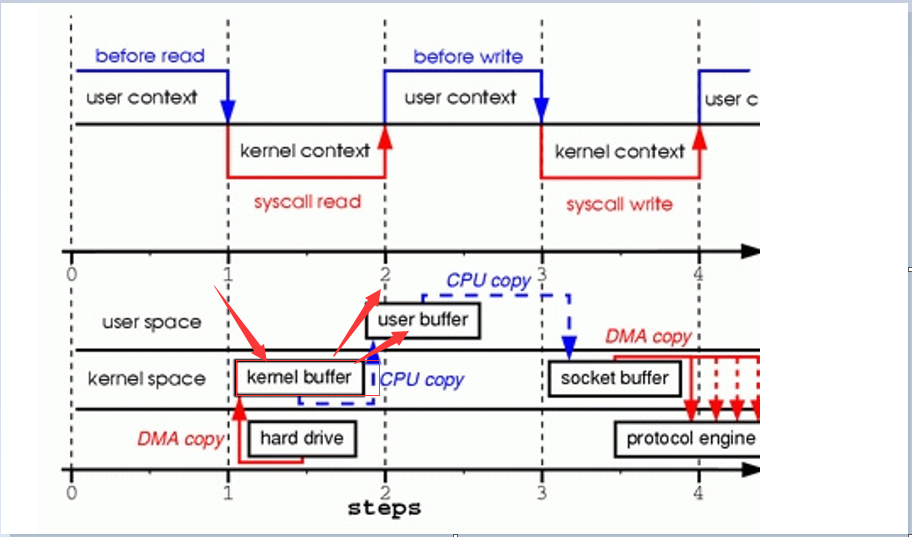
阻塞：当我们调用read、send、write等等系统调用API时，会把内核的缓存区里的数据拷贝到用户态的缓存区里，如果这个时候内核缓存区里没有数据，则会等待内核把数据准备好，此时，应用进程处于一种挂起状态。



非阻塞：当我们的应用程序调用write、send、read（所有的IO接口都是的），如果仅仅是从内核缓存区里把数据copy走，或者没有数据copy，此时也会直接返回，明显我们的用户进程并没有任何的等待，或者说处于挂起的状态，这种就是非阻塞。



默认创建的socket是阻塞的，但是可以调用下面这个函数设置为非阻塞：



非阻塞意味着在调用IO接口时，都不会发送等待，会立即返回。

同步

int function(xxxxx)

{

char buf[1024] = {0};

int size = read(fd, buf, bsize);

}

int read(xxxxx)

{

xxxxxjj

kk

。。。

memcpy(buf, kener\_buffer, size);

}

内核把数据准备好了之后，它也是通过read返回的，read一返回，其实也就意味着数据是否准备好。

int callback(char\* buffer, size)

{

printf(buffer);

}

int gun(oooo)

{

char buffer[1024] = {0};

s\_read(fd, callback, buffer, size);

}

内核：

int s\_read(fd, callback, buffer, size)

{

update\_fd\_set(fd, callback) 🡺 更新了fd的结构，也就是设置了一个回调函数

send\_signal(sig);

}

int process\_signal(sig)

{

copy\_data\_from\_disk\_2\_kener\_buffer()

copy\_data\_from\_kener\_buffer\_2\_user\_buffer();

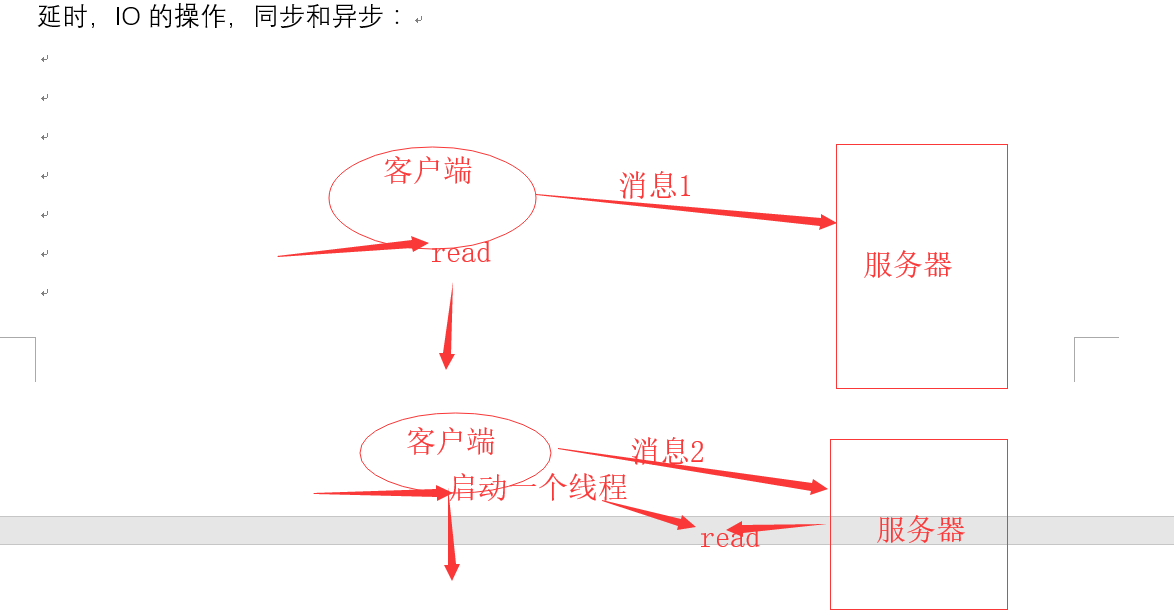
callback(user\_buffer, size);

}

异步，我调用read时，只是告诉你一个回调函数，并且read已经返回，但是此时并不代表数据准备好了，只有当内核调用了我们的callback才意味着数据已经准备好了。

信号，通知：

延时，IO的操作，同步和异步：



ssize\_t read(int fd, void \*buf, size\_t count);

int aio\_read(struct aiocb \*aiocbp);

struct aiocb {

/\* The order of these fields is implementation-dependent \*/

int aio\_fildes; /\* File descriptor \*/

off\_t aio\_offset; /\* File offset \*/

volatile void \*aio\_buf; /\* Location of buffer \*/

size\_t aio\_nbytes; /\* Length of transfer \*/

int aio\_reqprio; /\* Request priority \*/

struct sigevent aio\_sigevent; /\* Notification method \*/

int aio\_lio\_opcode; /\* Operation to be performed;

lio\_listio() only \*/

/\* Various implementation-internal fields not shown \*/

};

struct sigevent {

int sigev\_notify; /\* Notification method \*/

int sigev\_signo; /\* Notification signal \*/

union sigval sigev\_value; /\* Data passed with notification \*/

void (\*sigev\_notify\_function) (union sigval); /\* Function used for thread

notification (SIGEV\_THREAD) \*/

void \*sigev\_notify\_attributes; /\* Attributes for notification thread

(SIGEV\_THREAD) \*/

pid\_t sigev\_notify\_thread\_id; /\* ID of thread to signal (SIGEV\_THREAD\_ID) \*/

};