**blocking and non-blocking IO**

In simple terms, you can do blocking and non-blocking I/O with your phone.

When you call someone and talk to them, you must first wait until they pick up, then exchange information, and finally, hang up when finished. This is *blocking*I/O*.*

When you send text messages, you initiate the exchange by sending something you wrote, go off and do something else instead of waiting, and react at your own discretion when a signal alerts you that you have a response. This is *non-blocking* I/O.

Since we're talking about programming, it's obviously your program initiating a data exchange with some I/O device at the other end, but the principle is precisely the same.

Blocking and Non-Blocking IO are two different possible ways of servicing any IO operation. It is easy to understand the concept if you try to understand it from literal meaning, let me break it down :

* Blocking IO : From the name, we know that something is being blocked for IO operation. So, next we shall know what is being blocked, why is it getting blocked ?   
          IO operation are completed at hardware at base level, blocking here means the normal operation of your system is set to wait state until your IO operation is completed. In other words you cannot do anything unless whatever operations requested are being serviced.  
          This is the technique used in all synchronous devices or real-time systems, in cases where you expect your IO to complete for further processing. This comes with disadvantage of over waiting for operations to complete and system getting stalled, but IO operations are guaranteed to be safe and data is not obsolete or data is consistent.
* Non-blocking IO : Here the IO operations are performed independent of other system operations. Your system wont let you to go into wait state,

rather it can perform other operations which are independent of required IO.  
        Response from IO devices are furnished to the system once the operation completes, however system wouldn't stall in that time. Once you get the response, it can trigger some function to be completed. Any new requests sent to the same IO device will be queued and responses are also returned one after other. This is mostly achieved through parallel programming / creating multiple threads for each IO.