Out-of-band data is an optional feature supported by some communication protocols,  
allowing higher-priority delivery of data than normal. Out-of-band data is sent ahead  
of any data that is already queued for transmission. TCP supports out-of-band data , but UDP doesn’t. The socket interface to out-of-band data is heavily influenced by TCP’s implementation of out-of-band data. Out-of-band data is also referred to as “urgent” data in the context of TCP. TCP only supports a single byte of urgent data but allows urgent data to be delivered out of band from normal data delivery mechanisms. Generating urgent data is done with the MSG\_OOB flag to one of the send functions, the last byte of what is sent being treated as urgent data. Once urgent data has been received, the SIGURG signal is sent to denote that the urgent data has been sent successfully. Another thing to note about urgent data is that TCP queues only one byte of urgent data. In the case of another byte arriving, the previous and currently existing urgent data byte is discarded and replaced by the new urgent data. Normally, the recv functions will block when no data is immediately available and the send functions will block when there is not enough room in the socket’s output queue to send the message. When the socket is in nonblocking mode, these functions will fail instead of blocking, and they set errno to either EWOULDBLOCK or EAGAIN. When this happens, we can use either poll or select to determine when we can receive or transmit data. The Single UNIX Specification includes support for a general asynchronous I/O mechanism, but the socket mechanism has its own way of handling it. It uses a “signal-based I/O” based around the SIGNO signal. This method makes sure to establish socket ownership so signals can be delivered to the proper processes, and inform the socket that we want it to signal us when I/O operations won’t block.

\* With socket-based asynchronous I/O, we can arrange to be sent the SIGIO signal  
when we can read data from a socket or when space becomes available in a socket’s  
write queue. Enabling asynchronous I/O is a two-step process.  
1. Establish socket ownership so signals can be delivered to the proper processes.  
2. Inform the socket that we want it to signal us when I/O operations won’t block.  
We can accomplish the first step in three ways.  
1. Use the F\_SETOWN command with fcntl.  
2. Use the FIOSETOWN command with ioctl.  
3. Use the SIOCSPGRP command with ioctl.  
To accomplish the second step, we have two choices.  
1. Use the F\_SETFL command with fcntl and enable the O\_ASYNC file flag.  
2. Use the FIOASYNC command with ioctl.