18.7 Socket Implementation of UICI

If an attempt is made to write to a pipe or socket that no process has open for reading, `write` generates a SIGPIPE signal in addition to returning an error and setting `errno` to EPIPE. As with most signals, the default action of SIGPIPE terminates the process. Under no circumstances should the action of a client cause a server to terminate. Even if the server creates a child to handle the communication, the signal can prevent a graceful termination of the child when the remote host closes the connection. The socket implementation of UICI handles this problem by calling `u_ignore_sigpipe` to ignore the SIGPIPE signal if the default action of this signal is in effect.

The `hton1` and `htons` functions convert the address and port number fields to network byte order. The `setsockopt` call with SO_REUSEADDR permits the server to be restarted immediately, using the same port. This call should be made before `bind`.

If `setsockopt`, `bind` or `listen` produces an error, `u_open` saves the value of `errno`, closes the socket file descriptor, and restores the value of `errno`. Even if close changes `errno`, we still want to return with `errno` reporting the error that originally caused the return.

The `accept` function

After setting up a passive listening socket (socket, `bind` and `listen`), the server handles incoming client connections by calling `accept`. The parameters of `accept` are similar to those of `bind`. However, `bind` expects `*address` to be filled in before the call, so that it knows the port and interface on which the server will accept connection requests. In contrast, `accept` uses `*address` to return information about the client making the connection. In particular, the `sin_addr` member of the `struct sockaddr_in` structure contains a member, `s_addr`, that holds the Internet address of the client. The value of the `*address_len` parameter of `accept` specifies the size of the buffer pointed to by `address`. Before the call, fill this with the size of the `*address` structure. After the call, `*address_len` contains the number of bytes of the buffer actually filled in by the `accept` call.

```
SYNOPSIS

#include <sys/socket.h>

int accept(int socket, struct sockaddr *restrict address,
            socklen_t *restrict address_len);
```

If successful, `accept` returns the nonnegative file descriptor corresponding to the accepted socket. If unsuccessful, `accept` returns -1 and sets `errno`. The following table lists the mandatory errors for `accept`. 