

- [Previous: Chapter 4 Transport Selection and Name-to-Address Mapping](#)
- [Next: Appendix B Live Code Example](#)

Appendix A UNIX Domain Sockets

Introduction

UNIX domain sockets are named with UNIX paths. For example, a socket might be named `/tmp/foo`. UNIX domain sockets communicate only between processes on a single host. Sockets in the UNIX domain are not considered part of the network protocols because they can only be used to communicate between processes on a single host.

Socket types define the communication properties visible to a user. The Internet domain sockets provide access to the TCP/IP transport protocols. The Internet domain is identified by the value `AF_INET`. Sockets exchange data only with sockets in the same domain.

Socket Creation

The `socket(3SOCKET)` call creates a socket in the specified family and of the specified type.

```
s = socket(family, type, protocol);
```

If the protocol is unspecified (a value of `0`), the system selects a protocol that supports the requested socket type. The socket handle (a file descriptor) is returned.

The family is specified by one of the constants defined in `sys/socket.h`. Constants named *AF_suite* specify the address format to use in interpreting names as shown in [Table 2-1](#).

The following creates a datagram socket for intramachine use:

```
s = socket(AF_UNIX, SOCK_DGRAM, 0);
```

Use the default protocol (the *protocol* argument is `0`) in most situations.

Binding Local Names

A socket is created with no name. A remote process has no way to refer to a socket until an address is bound to it. Communicating processes are connected through addresses. In the UNIX family, a connection is composed of (usually) one or two path names. UNIX family sockets need not always be bound to a name, but, when bound, there can never be duplicate ordered sets such as: `local pathname` or `foreign pathname`. The path names cannot refer to existing files.

The `bind(3SOCKET)` call allows a process to specify the local address of the socket. This provides `local pathname`, while `connect(3SOCKET)` and `accept(3SOCKET)` complete a socket's association by fixing the remote half of the address. `bind(3SOCKET)` is used as follows:

```
bind (s, name, namelen);
```

The socket handle is *s*. The bound name is a byte string that is interpreted by the supporting protocol(s). UNIX family names contain a path name and a family. The example shows binding the name `/tmp/foo` to a UNIX family socket.

```
#include <sys/un.h>
...
struct sockaddr_un addr;
...
strcpy(addr.sun_path, "/tmp/foo");
addr.sun_family = AF_UNIX;
bind (s, (struct sockaddr *)&addr,
      strlen(addr.sun_path) + sizeof (addr.sun_family));
```

When determining the size of an `AF_UNIX` socket address, null bytes are not counted, which is why `strlen(3C)` use is fine.

The file name referred to in `addr.sun_path` is created as a socket in the system file name space. The caller must have write permission in the directory where `addr.sun_path` is created. The file should be deleted by the caller when it is no longer needed. `AF_UNIX` sockets can be deleted with [unlink\(1M\)](#).

Connection Establishment

Connection establishment is usually asymmetric. One process acts as the client and the other as the server. The server binds a socket to a well-known address associated with the service and blocks on its socket for a connect request. An unrelated process can then connect to the server. The client requests services from the server by initiating a connection to the server's socket. On the client side, the `connect(3SOCKET)` call initiates a connection. In the UNIX family, this might appear as:

```
struct sockaddr_un server;
server.sun_family = AF_UNIX;
...
connect(s, (struct sockaddr *)&server, strlen(server.sun_path)
+ sizeof (server.sun_family));
```

See "[Connection Errors](#)" for information on connection errors. "[Data Transfer](#)" tells you how to transfer data. "[Closing Sockets](#)" tells you how to close a socket.

- [Previous: Chapter 4 Transport Selection and Name-to-Address Mapping](#)
- [Next: Appendix B Live Code Example](#)