[Linux named sockets howto](https://superuser.com/questions/502875/linux-named-sockets-howto)

A Unix/Linux socket file is basically a two-way FIFO.  Since sockets were originally created as a way to manage network communications, it is possible to manipulate them using the send() and recv() system calls.  However, in the Unix spirit of “everything is a file”, you can also use write() and read().  You need to use socketpair() or socket() to create named sockets.  A tutorial for using sockets in C can be found here: [Beej's Guide to Unix IPC: Unix Sockets](https://beej.us/guide/bgipc/html/multi/unixsock.html).

The socat command line utility is useful when you want to play around with sockets without writing a "real" program.  It is similar to netcat and acts as an adapter between different networking and file interfaces

#include <fcntl.h>

#include <sys/un.h>

#include <sys/socket.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

int main(int argc, char \*\*argv)

{

// The following line expects the socket path to be first argument

char \* mysocketpath = argv[1];

// Alternatively, you could comment that and set it statically:

//char \* mysocketpath = "/tmp/mysock";

struct sockaddr\_un namesock;

int fd;

namesock.sun\_family = AF\_UNIX;

strncpy(namesock.sun\_path, (char \*)mysocketpath, sizeof(namesock.sun\_path));

fd = socket(AF\_UNIX, SOCK\_DGRAM, 0);

bind(fd, (struct sockaddr \*) &namesock, sizeof(struct sockaddr\_un));

close(fd);

return 0;

}

[What's the difference between Unix socket and TCP/IP socket?](https://serverfault.com/questions/124517/whats-the-difference-between-unix-socket-and-tcp-ip-socket)

A [**UNIX socket**](http://en.wikipedia.org/wiki/Unix_domain_socket) is an inter-process communication mechanism that allows bidirectional data exchange between processes running on the same machine.

[**IP sockets**](http://en.wikipedia.org/wiki/Internet_socket) (especially TCP/IP sockets) are a mechanism allowing communication between processes over the network. In some cases, you can use TCP/IP sockets to talk with processes running on the same computer (by using the loopback interface).

UNIX domain sockets know that they’re executing on the same system, so they can avoid some checks and operations (like routing); which makes them faster and lighter than IP sockets. So if you plan to communicate with processes on the same host, this is a better option than IP sockets.

Edit: As per [Nils Toedtmann's comment](https://serverfault.com/questions/124517/whats-the-difference-between-unix-socket-and-tcp-ip-socket#comment814984_124518): UNIX domain sockets are subject to file system permissions, while TCP sockets can be controlled only on the packet filter level.

You can list your own machine local unix sockets with the following command:

netstat -a -p --unix