

[\[Team LiB. \]](#)[◀ PREVIOUS](#)[NEXT ▶](#)

3.1 Introduction

This chapter begins the description of the sockets API. We begin with socket address structures, which will be found in almost every example in the text. These structures can be passed in two directions: from the process to the kernel, and from the kernel to the process. The latter case is an example of a value-result argument, and we will encounter other examples of these arguments throughout the text.

The address conversion functions convert between a text representation of an address and the binary value that goes into a socket address structure. Most existing IPv4 code uses `inet_addr` and `inet_ntoa`, but two new functions, `inet_pton` and `inet_ntop`, handle both IPv4 and IPv6.

One problem with these address conversion functions is that they are dependent on the type of address being converted: IPv4 or IPv6. We will develop a set of functions whose names begin with `sock_` that work with socket address structures in a protocol-independent fashion. We will use these throughout the text to make our code protocol-independent.

[\[Team LiB. \]](#)[◀ PREVIOUS](#)[NEXT ▶](#)