Socket Quick Guide

Previous Page | Next Page

Here is the list all the functions related to socket programming. You can have detail of these functions in respective pages.

Port and Service Functions:

Unix provides following functions to fetch service name from the /etc/services file.

- **struct servent *getservbyname(char *name, char *proto):** This call takes service name and protocol name and returns corresponding port number for that service.
- **struct servent *getservbyport(int port, char *proto):** This call takes port number and protocol name and returns corresponding service name.

Byte Ordering Functions:

- **unsigned short htons(unsigned short hostshort):** This function converts 16-bit (2-byte) quantities from host byte order to network byte order.
- **unsigned long htonl(unsigned long hostlong):** This function converts 32-bit (4-byte) quantities from host byte order to network byte order.
- **unsigned short ntohs(unsigned short netshort):** This function converts 16-bit (2-byte) quantities from network byte order to host byte order.
- **unsigned long ntohl(unsigned long netlong):** This function converts 32-bit quantities from network byte order to host byte order.

IP Address Functions:

- **int inet_aton(const char *strptr, struct in_addr *addrptr):** This function call converts the specified string, in the Internet standard dot notation, to a network address, and stores the address in the structure provided. The converted address will be in Network Byte Order (bytes ordered from left to right). This returns 1 if string was valid and 0 on error.
- **in_addr_t inet_addr(const char *strptr):** This function call converts the specified string, in the Internet standard dot notation, to an integer value suitable for use as an Internet address. The converted address will be in Network Byte Order (bytes ordered from left to right). This returns a 32-bit binary network byte ordered IPv4 address and INADDR_NONE on error.
- **char *inet_ntoa(struct in_addr inaddr):** This function call converts the specified Internet host address to a string in the Internet standard dot notation.

Socket Core Functions:

- **int socket (int family, int type, int protocol):** This call gives you a socket descriptor that you can use in later system calls or it gives you -1 on error.
- **int connect(int sockfd, struct sockaddr *serv_addr, int addrlen):** The connect

function is used by a TCP client to establish a connection with a TCP server. This call returns 0 if it successfully connects to the server otherwise it gives you -1 on error.

- **int bind(int sockfd, struct sockaddr *my_addr,int addrlen):** The bind function assigns a local protocol address to a socket. This call returns 0 if it successfully binds to the address otherwise it gives you -1 on error.
- **int listen(int sockfd,int backlog):** The listen function is called only by a TCP server to listen for the client request. This call returns 0 on success otherwise it gives you -1 on error.
- **int accept (int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen):** The accept function is called by a TCP server to accept client request and to establish actual connection. This call returns non negative descriptor on success otherwise it gives you -1 on error.
- **int send(int sockfd, const void *msg, int len, int flags):** The send function is used to send data over stream sockets or CONNECTED datagram sockets. This call returns the number of bytes sent out otherwise it will return -1 on error.
- **int recv(int sockfd, void *buf, int len, unsigned int flags):** The recv function is used to receive data over stream sockets or CONNECTED datagram sockets. This call returns the number of bytes read into the buffer otherwise it will return -1 on error.
- **int sendto(int sockfd, const void *msg, int len, unsigned int flags, const struct sockaddr *to, int tolen):** The sendto function is used to send data over UNCONNECTED datagram sockets. Put simply, when you use scoket type as SOCK_DGRAM. This call returns the number of bytes sent otherwise it will return -1 on error.
- **int recvfrom(int sockfd, void *buf, int len, unsigned int flags struct sockaddr *from, int *fromlen):** The recvfrom function is used to receive data from UNCONNECTED datagram sockets. Put simply, when you use scoket type as SOCK_DGRAM. This call returns the number of bytes read into the buffer otherwise it will return -1 on error.
- **int close(int sockfd):** The close function is used to close the communication between client and server. This call returns 0 on success otherwise it will return -1 on error.
- **int shutdown(int sockfd, int how):** The shutdown function is used to gracefully close the communication between client and server. This function gives more control in caomparision of close function. This call returns 0 on success otherwise it will return -1 on error.
- **int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *errorfds, struct timeval *timeout):** This function is used to read or write to multiple sockets.

Socket Helper Functions:

- **int write(int fildes, const void *buf, int nbyte):** The write function attempts to write nbyte bytes from the buffer pointed to by buf to the file associated with the open file descriptor, fildes. Upon successful completion, write() returns the number of bytes actually written to the file associated with fildes. This number is never greater than nbyte. Otherwise, -1 is returned.
- **int read(int fildes, const void *buf, int nbyte):** The read function attempts to read nbyte bytes from the file associated with the open file descriptor, fildes, into the buffer pointed to by buf. Upon successful completion, write() returns the number of bytes actually written to the file associated with fildes. This number is never greater than nbyte. Otherwise, -1 is returned.
- **int fork(void):** The fork function create a new process. The new process is called child process will be an exact copy of the calling process (parent process).
- **void bzero(void *s, int nbyte):** The bzero function places nbyte null bytes in the string

s. This function will be used to set all the socket structures with null values.

- **int bcmp(const void *s1, const void *s2, int nbyte):** The bcmp function compares byte string s1 against byte string s2. Both strings are assumed to be nbyte bytes long.
- **void bcopy(const void *s1, void *s2, int nbyte):** The bcopy function copies nbyte bytes from string s1 to the string s2. Overlapping strings are handled correctly.
- **void *memset(void *s, int c, int nbyte):** The memset function is also used to set structure variables in the same way as bzero.

Previous Page | Next Page