Sockets

Basics

Program - A program is an executable file residing on a disk in a directory. A program is read into memory and is executed by the kernel as a result of an exec() function. The exec() has six variants, but we only consider the simplest one (exec()) in this course.

Process - An executing instance of a program is called a process. Sometimes, task is used instead of process with the same meaning. UNIX guarantees that every process has a unique identifier called the process ID. The process ID is always a non-negative integer.

File descriptors - File descriptors are normally small non-negative integers that the kernel uses to identify the files being accessed by a particular process. Whenever it opens an existing file or creates a new file, the kernel returns a file descriptor that is used to read or write the file. As we will see in this course, sockets are based on a very similar mechanism (socket descriptors).

Functions

socket()

```
#include <sys/types.h>
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

- Creates an endpoint for communicaition and returns a file descriptor that refers to that endpoint.
- Domain specifies a communication domain or a protocol family eg., IPv4 or IPv6.
- Type specifies communication semantics - eg., TCP or UDP.
- The protocol specifies a particular protocol to be used with the socket. Normally only a single protocol exists to support a particular socket type within a given protocol family, in which case protocol can be specified as 0.
- Returns file descriptor on success, -1 on error.

setsockopt()

```c
#include <sys/socket.h>

int setsockopt(int sockfd, int level, int optname, const void *optval,
socklen_t optlen);
```

- getsockopt() and setsockopt() manipulate options for the socket referred to by the file descriptor sockfd.
- Using this function fixes the error of address already in use.

bind()

```c
#include <sys/types.h>
#include <sys/socket.h>

int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

- When a socket is created with socket(2), it exists in a name space (address family) but has no address assigned to it.
- bind() assigns the address specified by addr to the socket referred to by the file descriptor sockfd,
- addrlen specifies the size, in bytes, of the address structure pointed to by addr.

listen()

```c
#include <sys/types.h>
#include <sys/socket.h>

int listen(int sockfd, int backlog);
```

- listen() marks the socket referred to by sockfd as a passive socket, that is, as a socket that will be used to accept incoming connection requests using accept(2).
- The sockfd argument is a file descriptor that refers to a socket of type SOCK_STREAM or SOCK_SEQPACKET.
- The backlog argument defines the maximum length to which the queue of pending connections for sockfd may grow. If a connection request arrives when the queue is full, the client may receive an error with an indication of ECONNREFUSED or, if the underlying protocol supports retransmission, the request may be ignored so that a later reattempt at connection succeeds.

accept ()

```
#include <sys/types.h>
#include <sys/socket.h>

int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```

- The accept() system call is used with connection-based socket types (SOCK_STREAM, SOCK_SEQPACKET). - It extracts the first connection request on the queue of pending connections for the listening socket, sockfd, creates a new connected socket, and returns a new file descriptor refer- ring to that socket.
- The newly created socket is not in the listening state. The original socket sockfd is unaffected by this call.

connect()

```
#include <sys/types.h>
#include <sys/socket.h>

int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

- The connect() system call connects the socket referred to by the file descriptor sockfd to the address specified by addr.
- The addrlen argument specifies the size of addr. The format of the address in addr is determined by the address space of the socket sockfd

send()

```
#include <sys/types.h>
#include <sys/socket.h>

ssize_t send(int sockfd, const void *buf, size_t len, int flags);
ssize_t sendto(int sockfd, const void *buf, size_t len, int flags const
struct sockaddr *dest_addr socklen_t addrlen);
ssize_t sendmsg(int sockfd, const struct msghdr *msg, int flags);
```

- The system calls send(), sendto(), and sendmsg() are used to transmit a message to another socket.
- The send() call may be used only when the socket is in a connected state (so that the intended recipient is known).
- The only difference between
- send() and write() is the presence of flags. With a zero flags argument, send() is equivalent to write(2).

close()

```
#include <unistd.h>

int close(int fd);
```

- close() closes a file descriptor, so that it no longer refers to any file and may be reused.
- close() returns zero on success. On error, -1 is returned, and errno is set appropriately.

shutdown()

```
#include <sys/socket.h>

int shutdown(int sockfd, int how);
```

- The shutdown() call causes all or part of a full-duplex connection on the socket associated with sockfd to be shut down.
- If how is SHUT_RD, further receptions will be disallowed. If how is SHUT_WR, further transmissions will be disallowed. If how is SHUT_RDWR, further receptions and transmissions will be disallowed.
- On success, zero is returned. On error, -1 is returned, and errno is set appropriately.

recvfrom()

```
#include <sys/types.h>
#include <sys/socket.h>

ssize_t recv(int sockfd, void *buf, size_t len, int flags);
ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags, struct
sockaddr *src_addr, socklen_t *addrlen);
ssize_t recvmsg(int sockfd, struct msghdr *msg, int flags);
```

- The recv(), recvfrom(), and recvmsg() calls are used to receive messages from a socket. They may be used to receive data on both connectionless and connection-oriented sockets.
- These calls return the number of bytes received, or -1 if an error occurred. In the event of an error, errno is set to indicate the error.

References

- Socket Programming Basics
- Socket Programming - GFG

- SO_REUSEPORT
- Beej Guide
- Oracle Docs