```
CS3650: socket programming

// 1. This is a simple example of a client sending "hello world!"
//    to a server.

     [server]                   [client]
        |                          |
     fd = socket(...)           fd = socket(...)
     bind(fd,...)                  |
     listen(fd,...)                |
        |                          |
 new_fd = accept(fd,...)      +--connect(fd,...)
        |                    /     |
        |<------------------+      |
        |------------------------->|
        |                          |
     new_fd <====================> fd


// 2. Server code

  // assuming the following helper function will fill in the "struct sockaddr"
  void init_sockaddr(struct sockaddr *in_addr, const char *ip, int port);

  // return a file descriptor
  int listen_socket() {
      int fd = socket(AF_INET, SOCK_STREAM, 0);

      struct sockaddr addr;
      init_sockaddr(&addr, NULL, 3650 /*port number*/);

      bind(fd, &addr, sizeof(addr));

      listen(fd, 128);

      struct sockaddr tmp;
      socklen_t addr_size = sizeof(tmp);
      int new_fd = accept(fd, &tmp, &addr_size);

      close(fd); // stop accepting more connections

      return new_fd;
  }

  int main() {
      int new_fd = listen_socket();

      char buf[1024] = {0};
      recv(new_fd, &buf, 1024, 0); // receiving data
      printf("%s\n", buf);

      close(new_fd);
  }

```

```

// 3. Client code

int connect_socket() {
    int fd = socket(AF_INET, SOCK_STREAM, 0);

    struct sockaddr serv_addr;
    init_sockaddr(&serv_addr, "127.0.0.1" /* ip */, 3650 /* port */);

    connect(fd, &serv_addr, sizeof(serv_addr));

    return fd;
}

int main() {
    int fd = connect_socket();

    char *hello = “hello world!";
    send(fd, hello, strlen(hello), 0); // sending data

    close(fd);
}


```

4. Socket programming interfaces:

a) socket, send, and recv

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

  socket() creates an endpoint for communication and returns a descriptor.

* ssize_t **send**(int socket, const void *buffer, size_t length, int flags);

  send a message from a socket

* ssize_t **recv**(int socket, void *buffer, size_t length, int flags);

  receive a message from a socket

b) bind, listen, and accept (server side)

* int **bind**(int socket, const struct sockaddr *address, socklen_t address_len);

  bind() assigns a name to an unnamed socket.

* int **listen**(int socket, int backlog);

  listen for connections on a socket

* int **accept**(int socket, struct sockaddr *restrict address,
             socklen_t *restrict address_len);

  accept a connection on a socket

c) connect (client side)

* int **connect**(int socket, const struct sockaddr *address, socklen_t address_len);

  initiate a connection on a socket