

listen() to the socket() with datagrams



The Rolling Stones' secret computer was hidden in this very guitar, supplied to them by Evil Robots.

A Fabulous Look at the Hidden World of Socket Programming Among Superannuated Rock Stars by Asad Sayeed

socket

- Here is the call to socket:

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int domain, int type, int protocol);
```

- It returns a file descriptor.
- domain is usually `AF_INET`
- Type is either `SOCK_STREAM` or `SOCK_DGRAM`
- Usually set protocol to 0.

bind

- Unfortunately, a socket is useless by itself—you need a port!

- `bind()` gets you a port for listening:

```
#include <sys/types.h>
#include <sys/socket.h>
int bind(int sockfd, struct sockaddr *my_addr, int addrlen);
```

- `sockfd`—the socket file descriptor
- `my_addr`—a `sockaddr`
- `addrlen` is just `sizeof(struct sockaddr)`

bind example

- One way of using bind:

```
int sockfd;  
struct sockaddr_in my_addr;
```

```
sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

```
my_addr.sin_family = AF_INET;  
my_addr.sin_port = htons(MYPORT);  
my_addr.sin_addr.s_addr = inet_addr("132.241.5.10");  
bzero(&(my_addr.sin_zero), 8);
```

```
bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr));
```



Evil robots have trapped the Stones
behind the screen!

- Note: you should also check return values for errors in real application.

connect

- connect doesn't need bind—you aren't waiting on a port.

- very similar call though:

```
#include <sys/types.h>
#include <sys/socket.h>
int connect(int sockfd, struct sockaddr *serv_addr, int addrlen);
```

- Lets you send to other systems (ie, opens a foreign port).



Unbeknownst to rock-kind, Yoko is an Evil Robot. Observe as she establishes a connection to the robot armies!

listen

- You often need to wait for a port:

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

- `sockfd`—as usual, the socket file descriptor.
- `backlog`—the number of incoming connections allowed on the queue (you can only deal with one at a time, see).
- So to listen, first you would `socket()`, then you would `bind()`, then you would `listen()`.
- But what happens when someone connects?

Paul McCartney
just accepts
their dominion.



accept

- Once you're listening, you need to accept a connection when it comes.

```
#include <sys/socket.h>
int accept(int sockfd, void *addr, int *addrlen);
```

- `addr` is just the same `sockaddr`, except it's filled in by `accept`—represents other end.

Example:

```
bind(sockfd, (struct sockaddr *)&my_addr,
      sizeof(struct sockaddr));
listen(sockfd, BACKLOG);
sin_size = sizeof(struct sockaddr_in);
new_fd = accept(sockfd, &their_addr, &sin_size);
```

- `new_fd`—lets you read/write connection

send and recv and close

- Really easy to communicate now:

```
int send(int sockfd, const void *msg, int len, int flags);  
int recv(int sockfd, void *buf, int len, unsigned int flags)
```

```
#include <unistd.h>  
int close(int fd);
```

- `msg/buf, len`—the message to send or receive. `flags` is usually 0.
- `close` is just the regular file close.

..got to be a joker he just do what he please...

- Some credit goes to: <http://www.coding-zone.co.uk/cpp/articles/140101networkprogramming.shtml> – a complete tutorial.
- The man pages!



Yoda to the rescue is.
Yoda to rock and roll
ready is.
Yoda Evil Robots fight
will.
And then, Yoda Beatles clone
army fight will!