Sockets

Apr. 10rd

Server

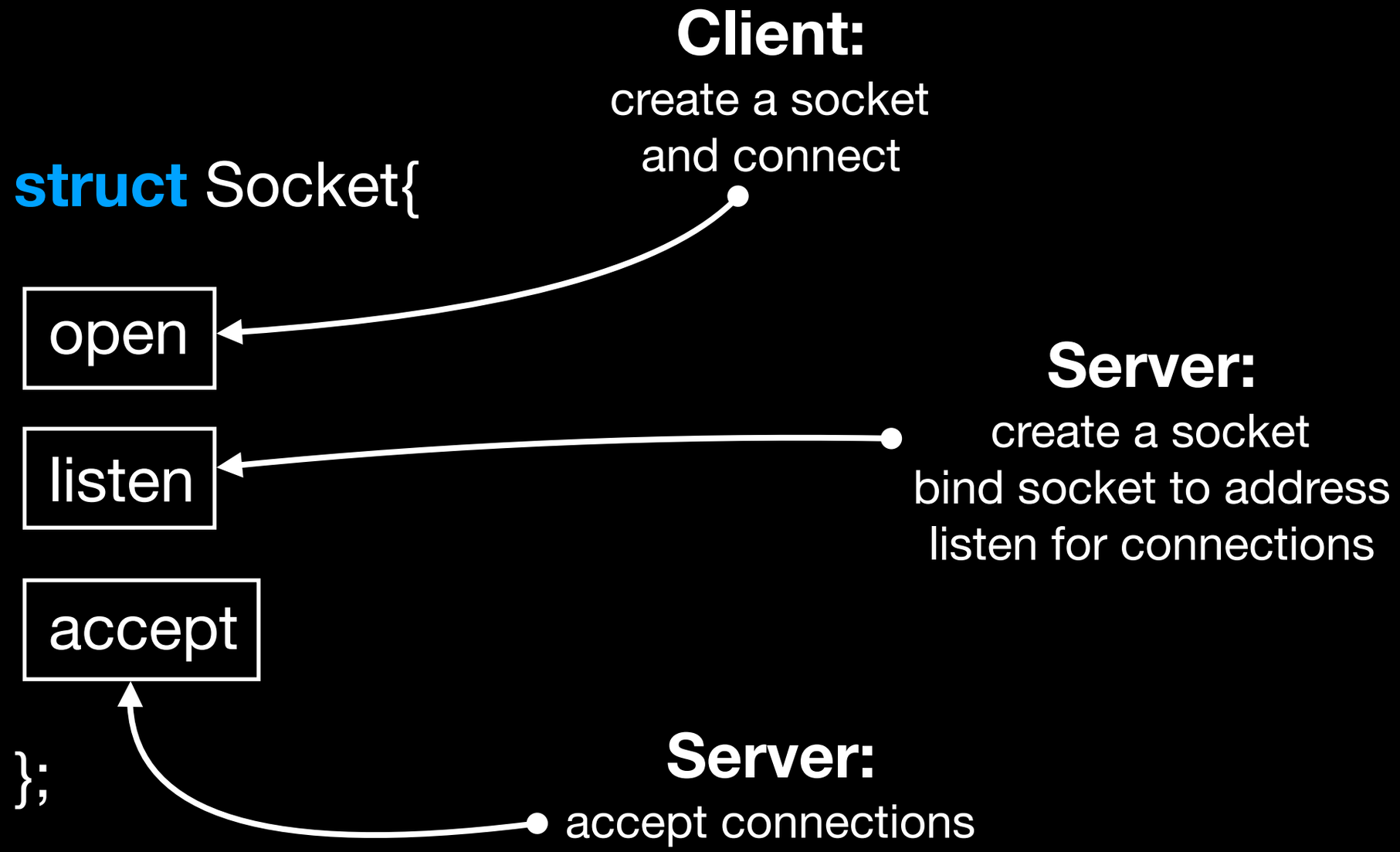
```
int socket(int domain, int type, int protocol)
```
Create a new socket

```
int bind(int socket,
  const struct sockaddr *address,
  socklen_t address_len)
```
Associate socket with an address

```
int listen(int socket, int backlog)
```
Listen for connections

```
int accept(int socket,
  struct sockaddr *address,
  socklen_t address_len)
```
Accept the connection, creating a new socket for the connection

Abstraction



listen

Output:
nothing (or maybe an error)

Input:
a port

```
void Socket::listen(string port){
    fd=socket( /*stuff goes in here*/);
}
```

returns a file descriptor

stored in the struct

listen

```
void Socket::listen(string port){

  fd = socket( /*stuff*/ );

  bind( fd, /*stuff*/ );

  listen(fd,/*stuff*/ );

}
```

Stuff

```
void Socket::listen(string port){

  getaddrinfo(NULL, port.c_str(), &hints,  &info);
  fd = socket(info->ai_family, info->ai_socktype, 0);
  bind( fd, info->ai_addr, info->ai_addrlen);
  listen(fd, /*stuff*/);
  freeaddrinfo(info);
}
```

Stuff

Use our hostname

```
void Socket::listen(string port){
  struct addrinfo hints;
  struct addrinfo *info;
  getaddrinfo(NULL, port.c_str(), &hints,  &info);

  fd = socket(info->ai_family, info->ai_socktype, 0);
  bind( fd, info->ai_addr, info->ai_addrlen);
  listen(fd, /*stuff*/);
  freeaddrinfo(info);
}
```

Stuff

```
void Socket::listen(string port){
  struct addrinfo hints;
  struct addrinfo *info;
  memset(&hints,0, sizeof(hints));
  hints.ai_flags = AI_PASSIVE; //will have bind called
  hints.ai_family = AF_INET6; //or AF_INET for IPv4
  hints.ai_socktype = SOCK_STREAM;
  getaddrinfo(NULL, port.c_str(), &hints,  &info);
  fd = socket(info->ai_family, info->ai_socktype, 0);
  bind( fd, info->ai_addr, info->ai_addrlen);
  listen(fd, /*stuff*/);
  freeaddrinfo(info);
}
```

Stuff

```
void Socket::listen(string port){
  struct addrinfo hints;
  struct addrinfo *info;
  memset(&hints,0, sizeof(hints));
  hints.ai_flags = AI_PASSIVE; //will have bind called
  hints.ai_family = AF_INET6; //or AF_INET for IPv4
  hints.ai_socktype = SOCK_STREAM;
  getaddrinfo(NULL, port.c_str(), &hints,  &info);
  fd = socket(info->ai_family, info->ai_socktype, 0);
  bind( fd, info->ai_addr, info->ai_addrlen);
  listen(fd, 4); //number of connections to queue
  freeaddrinfo(info);
}
```

accept

Output:
a socket for communication

Input:
nothing

```
Socket Socket::accept(){
  Socket sock;
  sock.fd = accept ( /*stuff goes in here*/);
  return sock;
}
```

accept

```
Socket Socket::accept(){
  struct addrinfo address;
  socklen_t size = sizeof(address);
  memset(&address, 0, sizeof(address));
  Socket sock;
  sock.fd = accept (&address,&size);
  return sock;
}
```