Socket Programming in C and Java

Karl Fürlinger
fuerling@in.tum.de

- Sockets are *one* way of doing inter-process communication (IPC) between processes on the *same* or on *different* machines in Unix
- First introduced with BSD 4.1c in 1983
  - Now available in all flavors of Unix and also on Windows („winsock“)
  - Especially well suited for and often used in client-server applications
  - http, ftp, ssh, … are sockets-based

Sockets in General

Karl Fürlinger
fuerling@in.tum.de

- Socket: *one* endpoint of a bi-directional communication link
  - A communication link always consists of two sockets
  - Setting up a communication link: create two sockets and connect them
- Client-server applications
  - Non-symmetric roles
  - Server waits for a connection from a client
  - One server can usually handle many clients: fork()
  - Loose coupling: client and server only need to speak the same protocol, no need to use the same programming language or the same architecture on both parts

Socket Programming in C

Karl Fürlinger
fuerling@in.tum.de

- A socket is accessed through a descriptor (system-provided integer)
  - Descriptor returned form or passed to functions
  - Many similarities with files (file descriptors); example: read and write functions
  - Difference to files in creation and control of options (more complex)

- Socket creation and usage in C: (all in <sys/sockets.h>
  - socket()
  - socketpair()
  - bind()
  - listen()
  - accept()
  - connect()
  - read()
  - write()
  - close()

Create a Socket

Karl Fürlinger
fuerling@in.tum.de

- int socket (int domain, int type, int protocol)
  - Domain is AF_UNIX or AF_INET
    - AF_INET for processes on different or on the same host
    - AF_UNIX for processes on the same machine only
  - Type is the style of communication:
    - SOCK_STREAM: connection-oriented „stream“: no record boundaries, guaranteed delivery
    - SOCK_DGRAM: connection-less „datagram“: sending individual records, delivery is not guaranteed
  - Protocol allows the specification of the underlying protocol to be used;
    - AF_INET and stream goes with TCP
    - AF_INET and datagram goes with UDP
    - The sysem will chose the most appropriate when specifying 0
  - Our application uses AF_INET and SOCK_STREAM

Assing a name to a socket: bind

Karl Fürlinger
fuerling@in.tum.de

- int bind(int sockfd, struct sockaddr *my_addr, socklen_t, addrlen)
  - Assigns a name to a socket, so that it can be referenced by another process
  - In the AF_UNIX domain, a name is established using the file system
  - In the AF_INET domain a name consists of an internet (IP) address and a port number
  - Example: 132.159.32.1:80
  - Ports below 1024 are reserved

Connect to another socket: connect

Karl Fürlinger
fuerling@in.tum.de

- int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);
  - Specify the local socket (sockfd) and the name of the remote socket
  - Remote socket must have been bound to that name
  - When connect succeeds, the connection is set up and you can read/write to the socked specified by sockfd

Setting up a Server to accept connections

Karl Fürlinger
fuerling@in.tum.de

- Create (socket()) and assign name (bind())
- Change to listening mode
  - int listen(int s, int backlog)
  - backlog is the length of the queue where unhandled connection requests are placed
  - non-blocking
- Wait for connections
  - int accept (int s, struct sockaddr *addr, socklen_t *addrlen);
  - blocking
  - returns a *new* socket used to talk to client, client's address information is placed in name
  - The old socket is used to handle new connection requests

Read/Write and Send/Recv

Karl Fürlinger
fuerling@in.tum.de

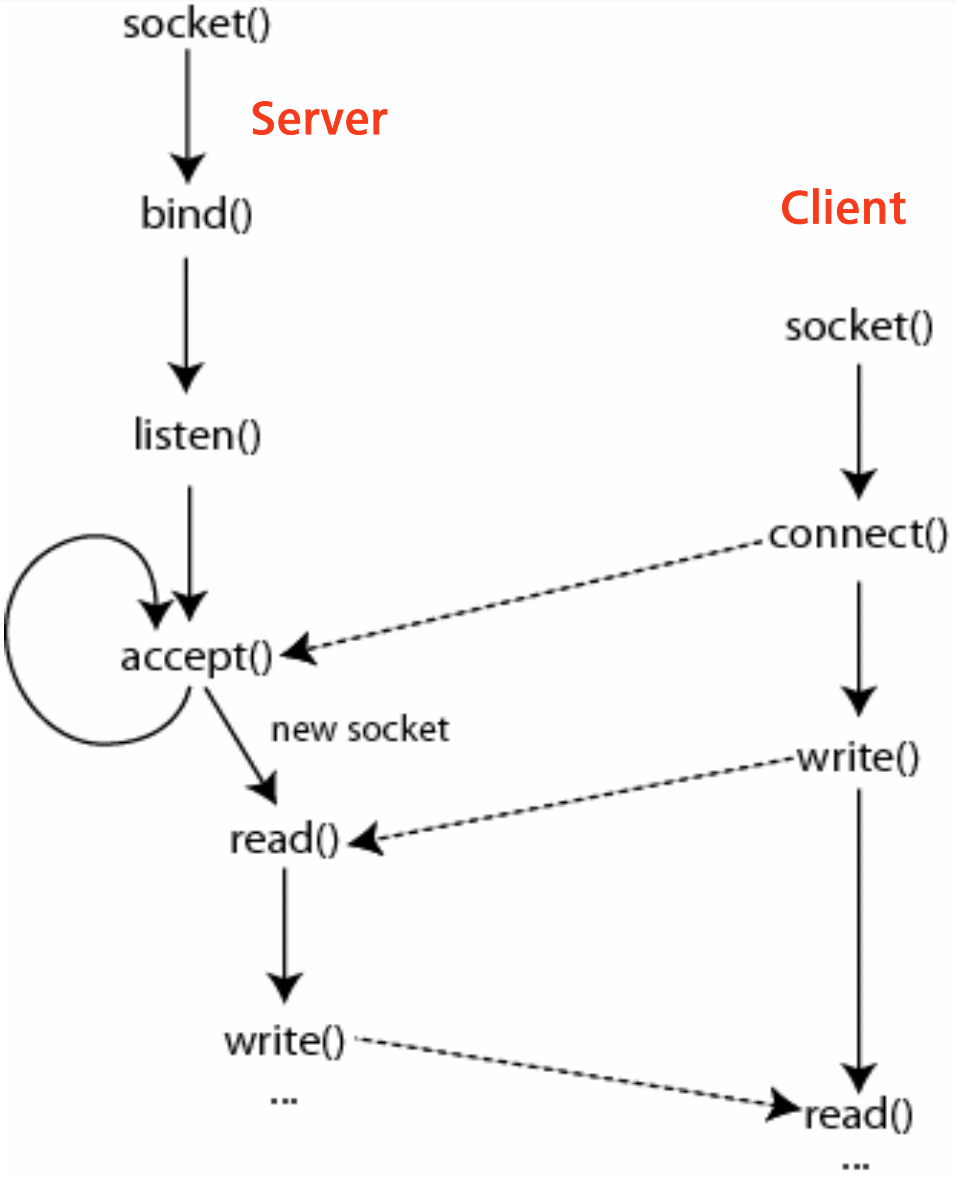
- ssize_t read (int fd, void *buf, size_t count);
- ssize_t write (int fd, const void *buf, size_t count);

- int send (int s, const void *msg, size_t len, int flags);
- int recv (int s, void *buf, size_t len, int flags);

- Also: int recv(int s, void *buf, size_t len, int flags);
  - Recvfrom, recvmsg, sendto sendmsg…

Client and Server

Karl Fürlinger
fuerling@in.tum.de



Sockets in Java

Karl Fürlinger
fuerling@in.tum.de

- Easier, Java hides a lot of details
- Two classes in java.net.*
  - Socket
  - ServerSocket
  - Example:
    mysock = new Socket(<hostname>, <port>);
    servsock = new ServerSocket(<port>);
  - Read from and write to socket via input and output streams
    mysock.getOutputStream()
    mysock.getInputStream()

- Server implementation in Java
  - Blocking accept();

```
Socket clientSocket = null;
try {
    clientSocket = serverSocket.accept();
}
catch (IOException e) {}
```