Berkley sockets in examples

Bartosz Chodorowski

<chomzee@ethernet.pl>

Selected choice from computer science
December 7, 2009

General information

- **Berkley sockets**, also known as **BSD sockets**

General information

- **Berkley sockets**, also known as **BSD sockets**
- A library for developing applications in the C programming language

General information

- **Berkley sockets**, also known as **BSD sockets**
- A library for developing applications in the C programming language
- Inter-process communication, most commonly for communications across a computer network

General information

- **Berkley sockets**, also known as **BSD sockets**
- A library for developing applications in the C programming language
- Inter-process communication, most commonly for communications across a computer network
- Originated with the 4.2BSD Unix operating system

General information

- **Berkley sockets**, also known as **BSD sockets**
- A library for developing applications in the C programming language
- Inter-process communication, most commonly for communications across a computer network
- Originated with the 4.2BSD Unix operating system
- Most popular API for network communication

General information

- **Berkley sockets**, also known as **BSD sockets**
- A library for developing applications in the C programming language
- Inter-process communication, most commonly for communications across a computer network
- Originated with the 4.2BSD Unix operating system
- Most popular API for network communication
- Most other programming languages use similar interface: Java, Perl, Python, Ruby, SWI-Prolog, ...

socket()

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

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

socket()

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

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

- Creates socket — an endpoint for communication and returns a file descriptor

socket()

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

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

- Creates socket — an endpoint for communication and returns a file descriptor
- domain can be one of:
  - PF_INET for IPv4
  - PF_INET6 for IPv6
  - PF_UNIX for local communication

socket()

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

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

- Creates socket — an endpoint for communication and returns a file descriptor
- domain can be one of:
  - PF_INET for IPv4
  - PF_INET6 for IPv6
  - PF_UNIX for local communication
- type is:
  - SOCK_STREAM for TCP
  - SOCK_DGRAM for UDP
  - SOCK_RAW for "raw sockets"

socket()

- protocol is usually one of:
  - IPPROTO_TCP
  - IPPROTO_UDP

bind()

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

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

bind()

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

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

- Assigns a socket to an address

bind()

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

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

- Assigns a socket to an address
- Must be called before accepting connections (server side)

listen()

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

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

listen()

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

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

- Server side, prepares socket for incoming connections

listen()

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

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

- Server side, prepares socket for incoming connections
- Not necessary for datagram sockets

listen()

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

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

- Server side, prepares socket for incoming connections
- Not necessary for datagram sockets
- backlog specifies number of pending connections that can be queued

accept()

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

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

accept()

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

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

- Accepts pending connection, deletes it from the queue, creates a new socket

accept()

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

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

- Accepts pending connection, deletes it from the queue, creates a new socket
- Server side, stream sockets only

connect()

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

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

connect()

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

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

- Connects given socket to remote host

connect()

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

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

- Connects given socket to remote host
- Not essential for datagram sockets, but may also be used

getaddrinfo() and getnameinfo()

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

int getaddrinfo(const char *node, const char *service,
        const struct addrinfo *hints,
        struct addrinfo **res);

void freeaddrinfo(struct addrinfo *res);

int getnameinfo(const struct sockaddr *sa, socklen_t salen,
        char *host, size_t hostlen,
        char *serv, size_t servlen,
        int flags);
```

getaddrinfo() and getnameinfo()

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

int getaddrinfo(const char *node, const char *service,
        const struct addrinfo *hints,
        struct addrinfo **res);

void freeaddrinfo(struct addrinfo *res);

int getnameinfo(const struct sockaddr *sa, socklen_t salen,
        char *host, size_t hostlen,
        char *serv, size_t servlen,
        int flags);
```

- Will be explained in an example

gethostbyname() and gethostbyaddr()

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

struct hostent *gethostbyname(const char *name);
struct hostent *gethostbyaddr(const char *addr, int len,
        int type);
```

gethostbyname() and gethostbyaddr()

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

struct hostent *gethostbyname(const char *name);
struct hostent *gethostbyaddr(const char *addr, int len,
        int type);
```

- Deprecated way of translating addresses

Examples

Questions?

Bibliography

- **“UNIX Network Programming, Volume 1, Second Edition: Networking APIs: Sockets and XTI”**, W. Richard Stevens
- **“Beej’s Guide to Network Programming”**, Brian "Beej Jorgensen" Hall, `http://beej.us/guide/bgnet/`
- **Wikipedia** (always handy), `http://en.wikipedia.org/`

Thank you for your attention.