Programación con sockets en C bajo UNIX

- Comunicacion entre Procesos
- Dominios y tipos.
- Superservidores
- Daemons
- Referencias

Comunicación entre procesos

- Dentro de la compu
  - En el shell (ejem. $ who | grep root)
  - UNIX Pipes
  - Socket pairs
- Fuera de la compu.
  - BSD Sockets

Dominios y tipos

- La llamada socket()

  Sirve para crear un apuntador a un socket.

  Prototipo:

  cc [flag ...] file ... -lsocket -lnsl [library ...]

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

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

Dominios

- PF_UNIX
- PF_INET
- PF_INET6
- PF_X25, PF_IPX, PF_APPLETALK, etc...

Tipos

- SOCK_STREAM
- SOCK_DGRAM
- SOCK_RAW
- SOCK_SEQPACKET
- SOCK_RDM

SOCK_RAW

- Programacion de protocolos de bajo nivel, encapsulamiento capa 2.
  - Nuevos protocolos capa 3 en adelante...
  - Sniffers
  - Monitoring

SOCK_DGRAM

- User Datagram Protocol (UDP)
- No es confiable para informacion secuencial.
- Implementa protocolos simples
  - DNS, TFTP, DHCP

SOCK_STREAM

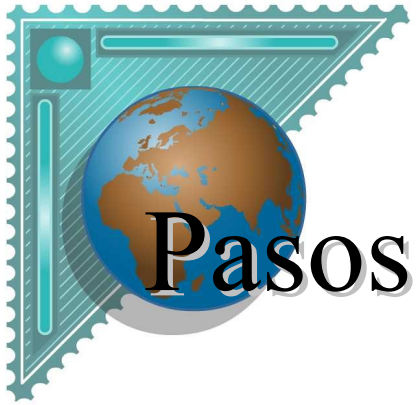
- Conectado
- Three Way Handshake
- Secuencial
- Full Duplex

Cliente/Servidor

- Secuencia de llamadas para crear un cliente:
  - Crear un socket usando socket()
  - Instanciar la estructura sockaddr_in
  - Conectarlo usando la llamada connect()
  - Leer/Escribir en el socket

Estructura de sockaddr_in

```
struct sockaddr_in{
    short           sin_family;
    u_short         sin_port;
    struct in_addr  sin_addr;
    char            sin_zero[8];
}
struct in_addr{
    // etc...
    u_long      s_addr;
    // etc...
}
```

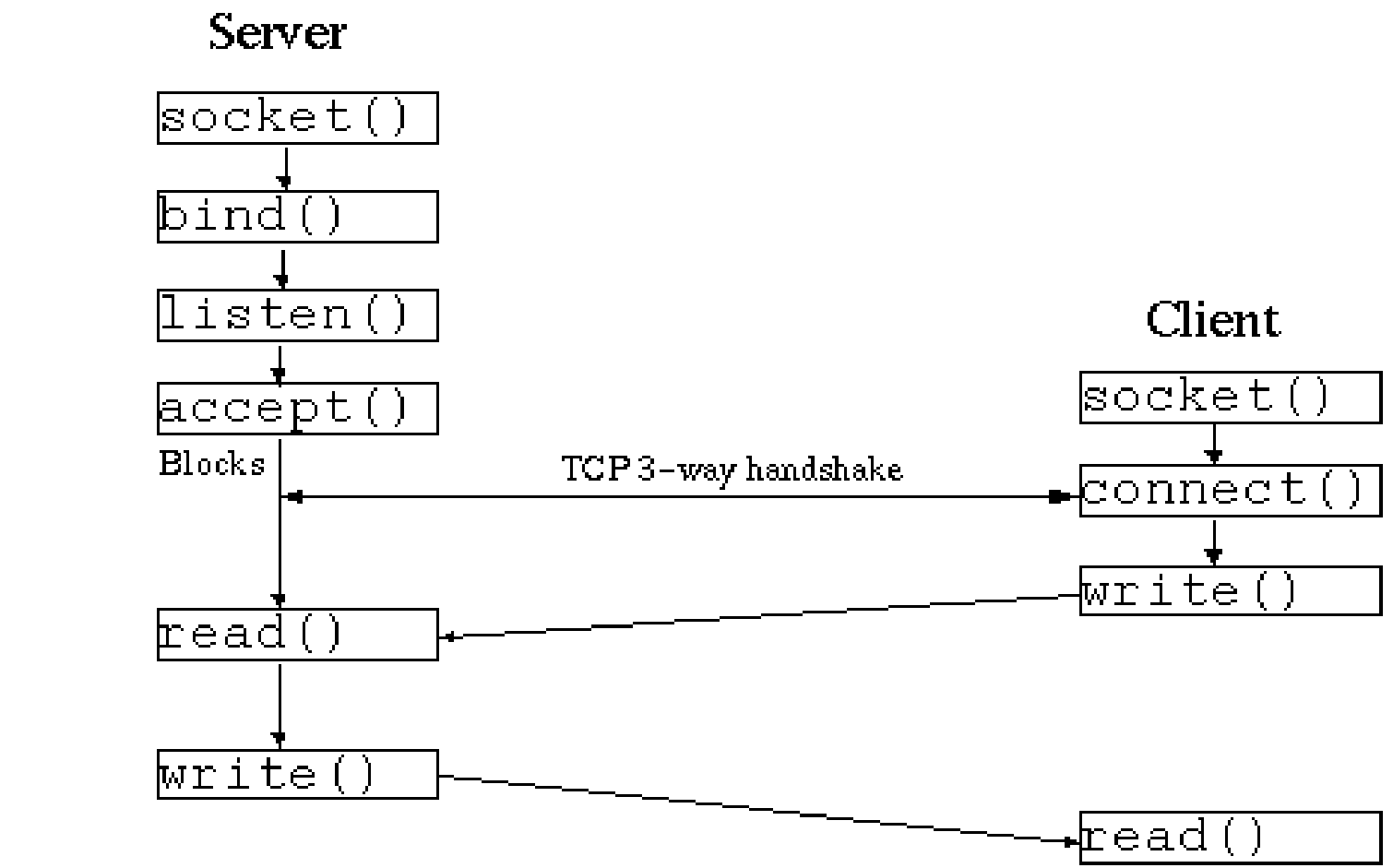


Pasos para crear un servidor

- socket()
- Asignar una dirección al socket usando bind()
- Poner el socket en modo de escuchar con listen()
- Bloquear en espera de una conexion con accept()
- Leer/Escribir en el socket conectado

Diagrama de establecimiento de CX

Fuente: http://www.cdt.luth.se/net/courses/smd068/labslides/img13.htm



Prototipo de bind()

cc [flag ...] file ... -lsocket -lnsl [library ...]

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

int bind(int s, const struct sockaddr *name, int namelen);
```

Prototipo de listen()

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

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

Prototipo de accept()

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

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

Servidores concurrentes

- Mono proceso
- Multiproceso

Super servidores

- Administran varios servidores tcp/udp
- Inetd
- xinetd

Demonios

- Session leader setsid()
- umask (0)
- chdir(“/”)
- Proceso aislado
- Manejo de senales

Referencias

- Advanced programming in the UNIX environment
- TCP/IP en UNIX, Jose Miguel Alonso, rama
- Etc...