Programación en C moderno

Álvaro Neira Ayuso <alvaro@soleta.eu>

c) Libevent: Biblioteca para manejar eventos

- Introducción
- Manejo de eventos y sockets
- Ejemplo: Servidor que acepte y reciba conexiones a partir de eventos

Introducción

- Es una biblioteca de software que proporciona un soporte de notificaciones de eventos asíncronos.
- Proporciona un mecanismo para ejecutar funciones cuando ocurre un evento específico en un descriptor de archivo.
- También puede ejecutarse después de que se haya alcanzado un tiempo de espera

Introducción

- Biblioteca programada en C
- Diseñada y escrita por Marc Lehmann and Emanuele Giaquinta.
- Iniciada en 2007.

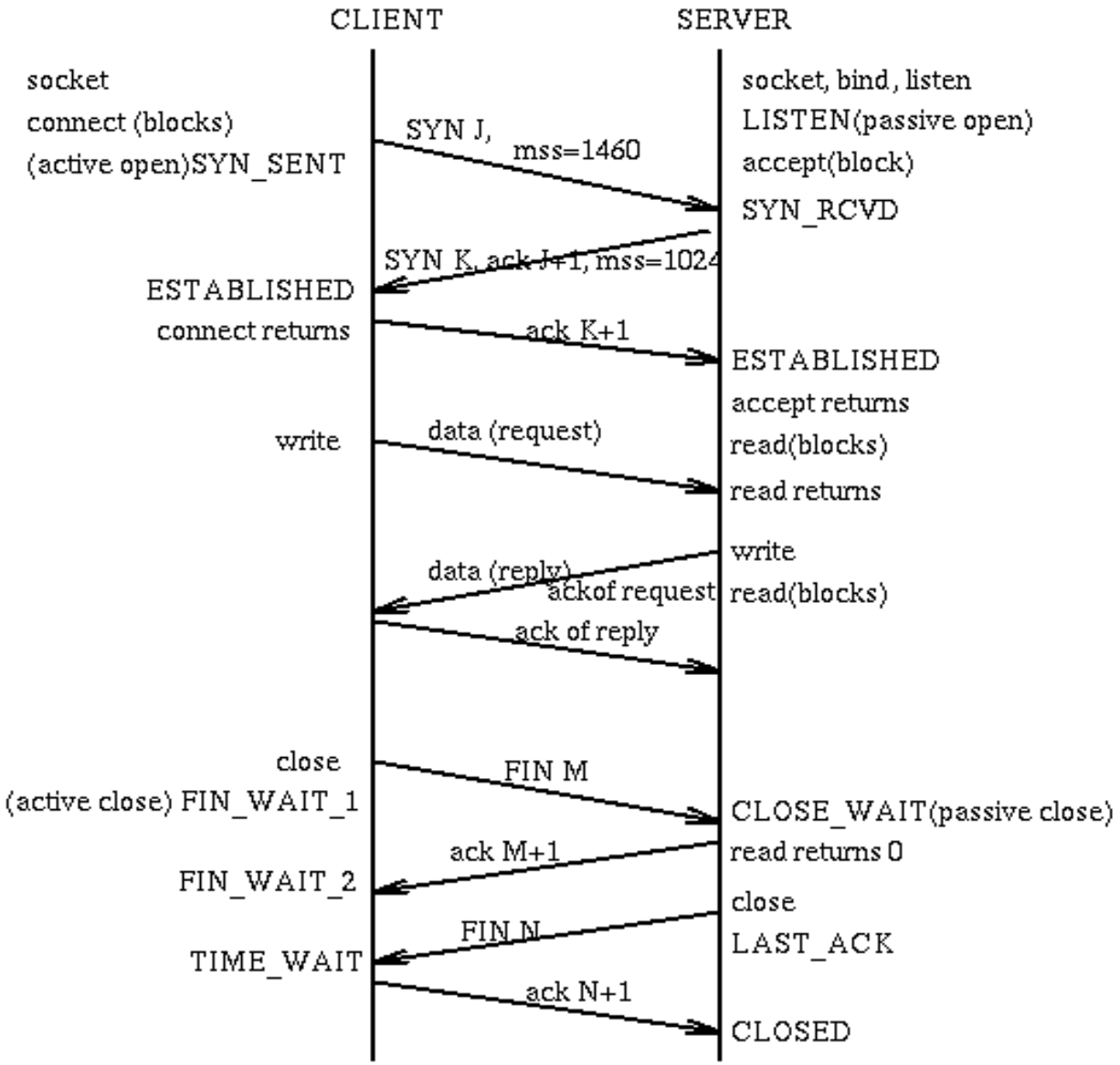
Manejo de sockets

- Concepto abstracto por el cuál dos programas pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.
- Los sockets de Internet constituyen el mecanismo para la entrega de paquetes de datos.
- Permite conexiones de varios tipos UDP, TCP, Socket unix...

Tipos de sockets: Sockets de flujo

- Están libres de errores.
- Si por ejemplo, enviáramos por el socket de flujo tres objetos "A, B, C", llegarán al destino en el mismo orden "A, B, C".
- Estos sockets usan TCP ("Transmission Control Protocol").

Tipos de sockets: Sockets TCP



Tipos de sockets: Sockets de Datagramas

- Éstos usan UDP (“User Datagram Protocol”), y no necesitan de una conexión accesible como los Sockets de Flujo.
- Se construye un paquete de datos con información sobre su destino y se enviará sin necesidad de una conexión.

Tipos de sockets Sockets UDP

Tipos de sockets: Socket Unix

- IPC (socket de comunicación interprocesos) es un socket virtual.
- Similar a un socket de Internet que se utiliza en los sistemas operativos POSIX para comunicación entre procesos
- Los sockets de dominio UNIX utilizan el sistema de archivos como su dirección de espacio de nombres
- Son vistos por los procesos como archivos de un sistema de archivos

Tipos de sockets: Sockets Unix

Estructuras

- Para utilizar sockets debemos conocer una serie de estructuras que albergan información importante sobre nuestras conexiones
- En este apartado vamos a dar las dos mas importantes

Estructuras

- La primera de ellas es struct sockaddr, la cual contiene información del socket.

```
struct sockaddr
{
    unsigned short sa_family;  /* familia de la dirección */
    char sa_data[14];      /* 14 bytes de la dirección del
    protocolo */
};
```

Estructuras

- Struct sockaddr_in, la cual nos ayuda a hacer referencia a los elementos del socket.

```
struct sockaddr_in
{
    short int sin_family;        /* Familia de la Dirección */
    unsigned short int sin_port; /* Puerto */
    struct in_addr sin_addr;     /* Dirección de Internet */
    unsigned char sin_zero[8];   /* Del mismo tamaño que struct sockaddr */
};
```

Funciones importantes para Sockets: Socket()

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

- Domain: Se podrá establecer como AF_INET (usar los protocolos de Internet), o como AF_UNIX (sockets para la comunicación interna del sistema). Éstas son las más usadas.
- Type: Aquí se debe especificar la clase de socket que queremos usar. Las variables que deben aparecer son SOCK_STREAM o SOCK_DGRAM según queramos usar sockets de Flujo o de Datagramas, respectivamente.
- Protocol: Aquí, simplemente se puede establecer el protocolo a 0.

La función socket() nos devuelve un descriptor de socket,

Funciones importantes para Sockets: Bind()

int bind(int fd, struct sockaddr *my_addr, int addrlen);

- Fd: Es el descriptor de fichero socket devuelto por la llamada a socket().
- My_addr: es un puntero a una estructura sockaddr
- Addrlen: contiene la longitud de la estructura sockaddr a la cuál apunta el puntero my_addr. Se debería establecer como sizeof(struct sockaddr).

Funciones importantes para Sockets: Connect()

int connect(int fd, struct sockaddr *serv_addr, int addrlen);

- Fd: Debería configurarse como el fichero descriptor del socket, el cuál fue devuelto por la llamada a socket().
- serv_addr: Es un puntero a la estructura sockaddr la cuál contiene la dirección IP destino y el puerto.
- Addrlen: Análogamente de lo que pasaba con bind(), este argumento debería establecerse como sizeof(struct sockaddr).

Devolverá -1 si ocurre algún error.

Funciones importantes para Sockets: Listen()

int listen(int fd, int backlog);

- Fd: Es el fichero descriptor del socket, el cual fue devuelto por la llamada a socket()
- Backlog: Es el número de conexiones permitidas.

  Devolverá -1 en caso de error

Funciones importantes para Sockets: Accept()

int accept(int fd, void *addr, int *addrlen);

- Fd: Es el fichero descriptor del socket, que fue devuelto por la llamada a listen().
- Addr: Es un puntero a una estructura sockaddr_in en la cuál se pueda determinar qué nodo nos está contactando y desde qué puerto.
- Addrlen: Es la longitud de la estructura a la que apunta el argumento addr, por lo que conviene establecerlo como sizeof(struct sockaddr_in).

Devuelve -1 en caso de error

Funciones importantes para Sockets: Send()

int send(int fd,const void *msg,int len,int flags);

- Fd: Es el fichero descriptor del socket, con el cual se desea enviar datos.
- Msg: Es un puntero apuntando al dato que se quiere enviar.
- Len: es la longitud del dato que se quiere enviar (en bytes).
- Flags: deberá ser establecido a 0.

Devolverá -1 en caso de error. En caso contrario devolverá el número de bytes escritos. Para conexiones DGRAM, debemos usar sendto.

Funciones importantes para Sockets: Recv()

int recv(int fd, void *buf, int len, unsigned int flags);

- Fd: Es el descriptor del socket por el cual se leerán datos.
- Buf: Es el búfer en el cual se guardará la información a recibir.
- Len: Es la longitud máxima que podrá tener el búffer.
- Flags: Por ahora, se deberá establecer como 0.

Devolverá -1 en caso de error. En caso contrario, devuelve número de bytes leídos en el búfer.

Funciones importantes para Sockets: Recvfrom()

int recvfrom(int fd,void *buf, int len, unsigned int flags struct sockaddr *from, int *fromlen);

- Fd: Lo mismo que para recv()
- Buf: Lo mismo que para recv()
- Len: Lo mismo que para recv()
- Flags: Lo mismo que para recv()
- From: Es un puntero a la estructura sockaddr.
- Fromlen: Es un puntero a un entero local que debería ser inicializado a sizeof(struct sockaddr).

Devolverá -1 en caso de error, en caso contrario devuelve el número de bytes leídos.

Funciones importantes para Sockets: Close()

close(fd);

La función close() es usada para cerrar la conexión de nuestro descriptor de socket.

Funciones importantes para Sockets: Gethostname()

int gethostname(char *hostname, size_t size);

- hostname. Es un puntero a un array que contiene el nombre de la máquina.
- size. La longitud del array que contiene al nombre de la máquina (en bytes).

Funciones importantes para Sockets: Gethostbyname()

struct hostent *gethostbyname(const char *name)

- Name: Puntero a un array que contiene el nombre de la máquina

Ejemplo

Uso de libev y como compilar

- Para utilizar las funciones que nos proporciona libev, debemos incluir la biblioteca ev.h

```
#include <ev.h>
```

- Si nuestro programa utiliza funciones proporcionadas por libev, debe ser compilado de la siguiente manera:

```
gcc -o main main.c -lev
```

Funciones importantes libev: ev_io_init

ev_io_init (ev_io *, callback, int fd, int events)

- ev_io: Estructura que contendrá información útil para usarla en el callback.
- fd: Descriptor el cuál producirá el evento
- callback: Función la cuál se ejecutará al producirse el evento designado en events
- events: Parámetro el cuál se designa el evento por el cuál se va a llamar al callback

Funciones importantes libev: ev_default_loop

struct ev_loop *ev_default_loop(unsigned int flags)

Devuelve la estructura loop la cuál es necesaria para iniciar el bucle.

Funciones importantes libev: ev_io_start

ev_io_start(loop, ev_TYPE *watcher)

- Inicia (activa) el observador dado. Solamente los observadores activos recibirán eventos. Si el observador ya no es más activa va a suceder.

Funciones importantes libev: ev_loop

ev_loop(loop, 0)

- Inicia un bucle, el cuál espera cualquier evento para ejecutar las funciones.

Funciones importantes libev: Declarar funciones tipo callback

void nombrefuncion_cb(struct ev_loop *loop, struct ev_io *watcher, int revents)

- Loop: la estructura creada a partir de ev_default_loop
- Watcher: contendrá el descriptor añadido en el ev_io_init
- Revents: El flag que contiene el evento activado

Herramientas Importantes: Tcpdump

- tcpdump -n udp -i wlan0 'port 80'

  Escucha el tráfico UDP en la interfaz wlan0 y cuyo puerto sea el 80

- tcpdump -i wlan0 -v icmp

  Escucha el tráfico ICMP en la interfaz wlan0

Herramientas Importantes: Netcat

- nc -l -u -p 9999

  Crea un socket de tipo escucha, cuya familia es IPv4 y de tipo UDP, el cuál escuchará en el puerto 9999

- nc -u localhost 9999

  Crea un socket cliente de tipo UDP, el cuál se escribirá a un socket de nuestra máquina que escucha en el puerto 999

Herramientas Importantes: Wireshark

Ejemplo

Bibliografía

- http://beej.us/guide/bgnet/
- http://beej.us/guide/bgipc/output/html/multipage/unixsock.html