13/12/22

OE A2

client main

```
#include "unp.h"

int
main (int argc, char * argv)
{
    int sock_fd;
    struct sockaddr_in servaddr;
    struct sctp_event_subscribe events;
    int echo_to_all = 0;
    if (argc < 2)
        err_quit ("Missing host argu");

    if (argc > 2)
    {   printf ("echoing msg to all streams");
        echo_to_all = 1;
    }
    sockfd = socket (AF_INET, SOCK_SEQPACKET, IPPROTO_SCTP);
    bzero (&servaddr, sizeof (servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl (INADDR_ANY);
    servaddr.sin_port = htons (SERV_PORT);
    inet_pton (AF_INET, argv[1], &servaddr.sin_addr);

    bzero (&events, sizeof (events));
    events.sctp_data_io_event = 1;
    setsockopt (sock_fd, IPPROTO_SCTP, SCTP_EVENTS, &events, sizeof (events));
    if (echo_to_all == 0)
        sctpstr_cli (stdin, sock_fd, (SA*) &servaddr, sizeof (servaddr));
    else
        sctpstr_cli_echoall (stdin, sock_fd, (SA*) &servaddr, sizeof (servaddr));
    close (sock_fd);
    return 0;
}
```

Server main

```
#include "unp.h"

int
main (int argc, char * argv)
{
    int sockfd, msg_flags;
    char readbuf [BUFFSIZE];
    struct sockaddr_in servaddr, cliaddr;
    struct sctp_sndrcvinfo sri;
    struct sctp_event_subscribe events;
    int stream_increment = 1;
    socklen_t len;
    size_t rd_sz;

    if (argc == 2)
        stream_increment = atoi (argv[1]);
    sockfd = socket (AF_INET, SOCK_SEQPACKET, IPPROTO_SCTP);
    bzero (&servaddr, sizeof (servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl (INADDR_ANY);
    servaddr.sin_port = htons (SERV_PORT);
    Bind (sockfd, (SA *) &servaddr, sizeof (servaddr));
    bzero (&events, sizeof (events));
    events.sctp_data_io_event = 1;
    Setsockopt (sock_fd, IPPROTO_SCTP, SCTP_EVENTS, &events,
        sizeof (events))
```

```
Listen ( sock_fd, LISTENQ);

for (;;){

    len = sizeof (struct sockaddr_in);

rd_sz = Sctp_recvmsg (sock_fd, readbuf, sizeof (readbuf),
        (SA*) &cliaddr, &len, &sri, &msg_flags);

if ( stream_increment) {
            sri.sinfo_stream++;
            if ( sri.sinfo_stream >=
                sctp_get_no_strms(sock_fd, (SA*)&cliaddr, len) }
            sri.sinfo_stream = 0;
        }
        sctp_sendmsg (sock_fd, readbuf, rd_sz, (SA*) &cli addr, len,
                sri.sinfo_ppid, sri.sinfo_flags, sri.sinfo_stream, 0, 0);
    }
}.
```

(2) Significance of

* daemon_init function.

→ This func we can call (normally from a server) to daemonize the process.

→ This func should be suitable for use on all various of UNIX, but some offer a library function called daemon that provides similar features

→ BSD offers the daemon func, as does Linux.

* Daemon_init func used in Daytime Server as a Daemon.

There are only two changes:

We call our daemon_init func as soon as the program starts, and we call our err_msg func, instead of printf, to print clients IP addr and port.

Indeed, if we want our programs to be able to run as daemon, we must avoid calling printf and fprintf func and use our err_msg function instead.

how we check argc and issue the appropriate usage msg before calling daemon_init.

This allows the user starting the daemon to get immediate feedback if comm'd has incorrect no of arguments.

After calling daemon_init, all subsequent error msg go to syslog.

If we run this program on our Linux host linux and then check the /var/log/messages files after connecting from the same machine, we have.

```
Tue 10 09:54:37 linux daytime tcpsrv2[24288]:
connection from 127.0.0.1.55862
```

the date, time, and hostname are prefixed automatically by the syslogd daemon.

inetd Daemon function.

* The 4.3 BSD release simplified ~~this~~ by providing an Internet super server: the inetd daemon

* This daemon can be used by services that use either TCP or UDP.
* It does not handle other protocols such as Unix domain sockets.
* This daemon fix two problems

1. It simplifies writing daemon processes since most of the startup details are handled by inetd. This obviates the need for each server to call our daemon_init function.

2. It allows a single process (inetd) to be waiting for incoming client requests for multiple services, instead of one process for each service. This reduce the total no of processes in system.

(*) daemon_inetd function.

daemon_inetd function daemonizes process run by inetd.

This function is trivial, compared to daemon_init, because all of the daemonization steps are performed by inetd when it starts. All that we do is set the daemon_proc flag for our error function and call openlog with same arguments.

→ (3) multicast socket options.

The API support for traditional multicasting requires only 5 new socket options.

Command	Datatype	Discription
IP_MULTICAST_IF	struct in_addr	Specify default interface for outgoing multicasts
IP_MULTICAST_TTL	u_char	Specify TTL for outgoing multicast
IP_MULTICAST_LOOP	u_char	Enable or disable loopback of outgoing multicasts.
IPV6_MULTICAST_IF	u_int	Specify default interface for outgoing multicasts
IPV6_MULTICAST_HOPS	int	Specify hop limit for outgoing multicast
IPV6_MULTICAST_LOOP	u_int	Enable or disable loopback of outgoing multicasts.

- IP_MULTICAST_IF, IPV6_MULTICAST_IF

- Specify the interface for outgoing multicast datagrams sent on this socket.
- This interface is specified as either an in_addr structure for IPV4 or an interface index for IPV6.

- IP_MULTICAST_TTL, IPV6_MULTICAST_HOPS.

- Set the IPV4 TTL or the IPV6 hop limit for outgoing multicast datagrams.
- If this is not specified, both will default to 1, which restricts the datagram to the local subnet.

- IP_MULTICAST_LOOP, IPV6_MULTICAST_LOOP.

- Enable or disable local loopback of multicast datagrams.
- By default, loopback is enabled

(*) Simple Network time Protocol (SNTP)

- NTP is a sophistical protocol for synchronizing clocks across a WAN or a LAN, and can often achieve millisecond accuracy.
- RFC 1305 describes the protocol in detail and RFC 2030 describes SNTP as simplified but protocol compatible version intended for hosts that do not need the complexity of a complete NTP implementation.
- It is common for a few hosts on a LAN to synchronize their clocks across the Internet to other NTP hosts & then distribute this on the LAN using either broadcasting or multicasting.