Today

- test review
- examples (udp, ntp)

assignment 7

library makefile

```
SIMSRC = mpsim.c drive.c userin.c
SIMOBJ = mpsim.o drive.o userin.o
LIBSRC = simlib.c ftnsubs.c
LIBOBJ = simlib.o ftnsubs.o

CFLAGS = -O -DBCOPY

all: mpsim simlib client

mpsim: $(SIMOBJ)
        cc  -o mpsim $(SIMOBJ)

$(SIMOBJ): sim.h simlib.h

simlib: $(LIBOBJ)
        ar cr simlib.a $(LIBOBJ)
        ranlib simlib.a

$(LIBOBJ): simlib.h

client: client.o simlib.a
cc -o client client.o simlib.a
```


TCP/UDP echo server

text: 8.15

- concurrent TCP server
- plus iterative UDP server
- uses select()

```
/* create listening TCP socket */
 listenfd = Socket(AF_INET, SOCK_STREAM, 0);

 bzero(&servaddr, sizeof(servaddr));
 servaddr.sin_family      = AF_INET;
 servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
 servaddr.sin_port        = htons(SERV_PORT);

 Setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(on));
 Bind(listenfd, (SA *) &servaddr, sizeof(servaddr));

 Listen(listenfd, LISTENQ);
```



```
 /* create UDP socket */
 udpfd = Socket(AF_INET, SOCK_DGRAM, 0);

 bzero(&servaddr, sizeof(servaddr));
 servaddr.sin_family      = AF_INET;
 servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
 servaddr.sin_port        = htons(SERV_PORT);

 Bind(udpfd, (SA *) &servaddr, sizeof(servaddr));

 Signal(SIGCHLD, sig_chld);      /* must call waitpid() */
```

```
FD_ZERO(&rset);
maxfdp1 = max(listenfd, udpfd) + 1;
for ( ; ; ) {
        FD_SET(listenfd, &rset);
        FD_SET(udpfd, &rset);
        if ( (nready = select(maxfdp1, &rset, NULL, NULL, NULL)) < 0) {
                if (errno == EINTR)
                        continue;               /* back to for() */
                else
                        err_sys("select error");
        }

        if (FD_ISSET(listenfd, &rset)) {
                len = sizeof(cliaddr);
                connfd = Accept(listenfd, (SA *) &cliaddr, &len);     }

                if ( (childpid = Fork()) == 0) {   /* child process */
                        Close(listenfd);        /* close listening socket */
                        str_echo(connfd);       /* process the request */
                        exit(0);
                }
                Close(connfd);     /* parent closes connected socket */
            }
```



```
if (FD_ISSET(udpfd, &rset)) {
        len = sizeof(cliaddr);
        n = Recvfrom(udpfd, mesg, MAXLINE,
          0, (SA *) &cliaddr, &len);

        Sendto(udpfd, mesg, n, 0, (SA *) &cliaddr, len);
 }
```

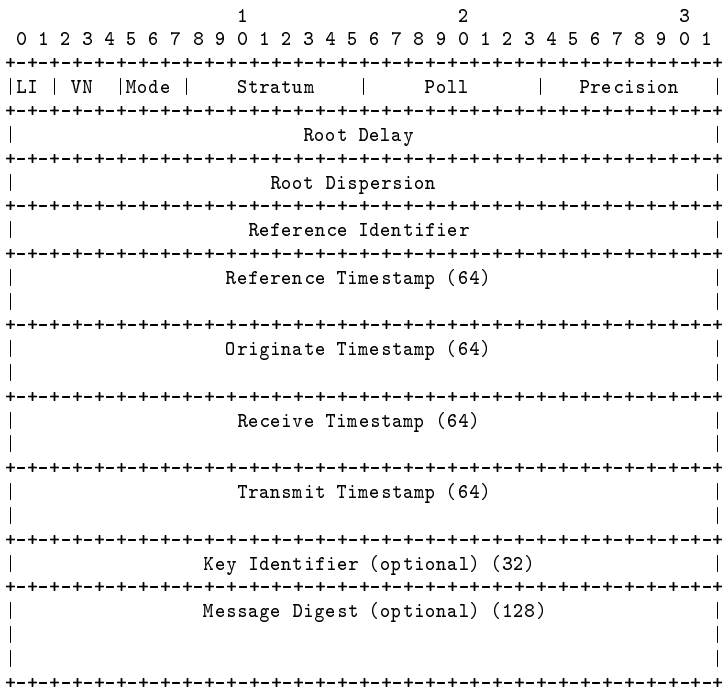

NTP

Network Time Protocol

- having a good time
- handy for logs, NFS, Kerberos and setting your watch
- RFC 1119 and later
- hierarchical time service
- GPS, WWV, atomic clocks at level 1
- synchronizes time and frequency
- time within 10ms
- frequency within 1ms/day
- rollover 2036

implementation

- xntpd daemon
- ntp.conf lists peers
- UDP port 123 request/reply
- asks time from peers and measures round-trip delay
- selects best peer
- adjust local clock and frequency (**adjtime**)
- answers time requests
- uses SIGIO and select()
- error proportional to variance in round-trip delay
- broadcast/multicast options




```
ACHERNAR.ntp > TYCO.ntp: v3 client strat 4 poll 6 prec -15
         4500 004c cb4f 0000 3c11 f74c 80a9 5e3f
         80a9 5c73 007b 007b 0038 0000 1b04 06f1
         0000 1348 0000 0dd6 80a9 5c73 bbbe ee38
         1b6f e000 bbbe ee38 1aed 1000 bbbe ee38
         1b6f e000 bbbe ee57 fa18 0000
TYCO.ntp > ACHERNAR.ntp: v3 server strat 3 poll 6 prec -17 (DF)
         4500 004c 452f 4000 ff11 7a6c 80a9 5c73
         80a9 5e3f 007b 007b 0038 5019 1c03 06ef
         0000 12a1 0000 0865 c0f9 0119 bbbe ee32
         1b77 0000 bbbe ee57 fa18 0000 bbbe ee57
         fa39 d000 bbbe ee57 fa5d 7000


DUNCAN.ntp > 128.169.95.255.ntp: v3 bcast strat 3 poll 6 prec -16 (D
         4500 004c 7d50 4000 0111 3cac 80a9 5e53
         80a9 5fff 007b 007b 0038 3f7a 1d03 06f0
         0000 1cd5 0000 2da9 80a7 01de bbbe ec90
         5a44 3000 0000 0000 0000 0000 0000 0000
         0000 0000 bbbe eec9 4154 f000
```


xntpd.c

```
main()...

      if (fork())
         exit(0);

      {
         int s;
         for (s = getdtablesize(); s >= 0; s--)
            (void) close(s);
         (void) open("/", 0);
         (void) dup2(0, 1);
         (void) dup2(0, 2);
         (void) setpgrp(0, getpid());
         s = open("/dev/tty", 2);
         if (s >= 0) {
            (void) ioctl(s, (u_long) TIOCNOTTY, (char *) 0);
            (void) close(s);
         }
   openlog(cp, LOG_PID | LOG_NDELAY, LOG_NTP);
   syslog(LOG_INFO, Version);
signal's and init stuff ...
```


main loop

```
   for (;;) {
      omask = sigblock(BLOCKSIGMASK);
      if (alarm_flag) {    /* alarmed? */
         was_alarmed = 1;
         alarm_flag = 0;
      }
      rbuflist = getrecvbufs();  /* get received buffers */

      if (!was_alarmed && rbuflist == (struct recvbuf *)0) {
         /*
          * Nothing to do.  Wait for something.
          */
         sigpause(omask);
         if (alarm_flag) {    /* alarmed? */
            was_alarmed = 1;
            alarm_flag = 0;
         }
         rbuflist = getrecvbufs();  /* get received buffers */
      }
      (void)sigsetmask(omask);
```

```
   /*
    * Out here, signals are unblocked.  Call timer routine
    * to process expiry.
    */
   if (was_alarmed) {
      timer();
      was_alarmed = 0;
   }

   /*
    * Call the data procedure to handle each received
    * packet.
    */
   while (rbuflist != (struct recvbuf *)0) {
      rbuf = rbuflist;
      rbuflist = rbuf->next;
      (rbuf->receiver)(rbuf);
      freerecvbuf(rbuf);
   }
   /*
    * Go around again
    */
}
```


ntp_io.c

```
   (void) signal(SIGIO, input_handler);
   BLOCKIO(i);
   inter_list[0].sin.sin_family = AF_INET;
   inter_list[0].sin.sin_port = port;
   inter_list[0].sin.sin_addr.s_addr = INADDR_ANY;
    if ((vs = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
           syslog(LOG_ERR, "vs=socket(AF_INET, SOCK_DGRAM) %m");
           exit(1);
    }
... creates socket for each network interface ...
   if ((fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
      syslog(LOG_ERR, "socket() failed: %m");
      exit(1);
 }
   if (fd > maxactivefd)
      maxactivefd = fd;
   FD_SET(fd, &activefds);
   if (setsockopt(fd, SOL_SOCKET, SO_REUSEADDR,
           (char *)&on, sizeof(on))) {
      syslog(LOG_ERR, "setsockopt SO_REUSEADDR on fails: %m");
   }
```



```
if (bind(fd, (struct sockaddr *)addr, sizeof(*addr)) < 0) {
   syslog(LOG_ERR, "bind() fails: %m");
   exit(1);
}
if (fcntl(fd, F_SETOWN, getpid()) == -1)
{
   syslog(LOG_ERR, "fcntl(F_SETOWN) fails: %m");
   exit(1);
}
if (fcntl(fd, F_SETFL, FNDELAY|FASYNC) < 0) {
   syslog(LOG_ERR, "fcntl(FNDELAY|FASYNC) fails: %m");
   exit(1);
   /*NOTREACHED*/
}
     UNBLOCKIO(i);
```


SIGIO handler

first thing: get timeofday

```
again:
   fds = activefds;
   tvzero.tv_sec = tvzero.tv_usec = 0;
   n = select(maxactivefd+1, &fds, (fd_set *)0,
     (fd_set *)0, &tvzero);
   if (n == 0) return;
   else if (n == -1) {
      syslog(LOG_ERR, "select() error: %m");
      return;
   }
    /* Check out the reference clocks first, if any */
   if (refio != 0) {
      for (rp = refio; rp != 0 && n > 0; rp = rp->next) {
         fd = rp->fd;
         if (FD_ISSET(fd, &fds)) {
 ...
   for (i = ninterfaces-1; i >= 0 && n > 0; i--) {
            fd = inter_list[i].bfd;
         if (FD_ISSET(fd, &fds)) {
            n--;
            fromlen = sizeof(struct sockaddr_in);
            rb->recv_length = recvfrom(fd,
                (char *)&rb->recv_space,
                sizeof(rb->recv_space), 0,
                (struct sockaddr *)&rb->recv_srcadr,
                &fromlen);
... goto again;
```

next time

sample client servers

PVM

XDR