```
:::::::::::::: who0.c ::::::::::::::
#include       <stdio.h>
#include       <fcntl.h>
#include       <utmp.h>

/*
 *      who version 0
 *
 *              main outline but no substance
 */

main()
{
        int     fd;                     /* for file des of utmp */
        struct utmp current_record;     /* hold info from file */
        int     reclen = sizeof(struct utmp);

        fd = open( UTMP_FILE, O_RDONLY );
        if ( fd == -1 )
        {
                perror( "who0" );
                exit(1);
        }

        while ( read ( fd , &current_record , reclen ) == reclen )
                show_info( &current_record );

        close ( fd );
}
:::::::::::::: who1.c ::::::::::::::
#include       <stdio.h>
#include       <sys/types.h>
#include       <utmp.h>
#include       <fcntl.h>
/*
 *      who version 1          - read /etc/utmp and list info therein
 */

/* #define     SHOWHOST */

main(int ac, char **av)
{
        struct utmp     utbuf;          /* read info into here */
        int             utmpfd;         /* read from this descriptor */

        if ( (utmpfd = open( UTMP_FILE, O_RDONLY )) == -1 ){
                fprintf(stderr,"%s: cannot open %s\n", *av, UTMP_FILE);
                exit(1);
        }

        while ( read( utmpfd, &utbuf, sizeof(utbuf) ) == sizeof(utbuf) )
                show_info( &utbuf );
        close( utmpfd );
        return 0;                       /* went ok                      */
}
/*
 *      show info()
 *                      displays the contents of the utmp struct
 *                      in human readable form
 *                      *note* these sizes should not be hardwired
 */
show_info( struct utmp *utbufp )
{
        printf("%-8.8s", utbufp->ut_name);            /* the logname */
        printf(" ");                                  /* a space     */
        printf("%-8.8s", utbufp->ut_line);            /* the tty     */
        printf(" ");                                  /* a space     */
        printf("%10ld", utbufp->ut_time);             /* login time  */
        printf(" ");                                  /* a space     */
#ifdef  SHOWHOST
        printf("(%s)", utbufp->ut_host);              /* the host    */
#endif
        printf("\n");                                 /* newline     */
}
```

```
:::::::::::::: who2.c ::::::::::::::
#include        <stdio.h>
#include        <sys/types.h>
#include        <utmp.h>
#include        <fcntl.h>
#include        <time.h>
/*
 *      who version 2          - read /etc/utmp and list info therein
 *                             - surpresses empty records
 *                             - formats time nicely
 */

/* UTMP_FILE is a symbol defined in utmp.h */
/* #define     SHOWHOST */

main(int ac, char **av)
{
        struct utmp     utbuf;          /* read info into here */
        int             utmpfd;         /* read from this descriptor */

        if ( (utmpfd = open( UTMP_FILE, O_RDONLY )) == -1 ){
                fprintf(stderr,"%s: cannot open %s\n", *av, UTMP_FILE);
                exit(1);
        }

        while ( read( utmpfd, &utbuf, sizeof(utbuf) ) == sizeof(utbuf) )
                show_info( &utbuf );
        close( utmpfd );
}
/*
 *      show info()     displays the contents of the utmp struct
 *                      in human readable form
 *                      * displays nothing if record has no user name
 */
show_info( struct utmp *utbufp )
{
        if ( utbufp->ut_type != USER_PROCESS )
                return;

        printf("%-8.8s", utbufp->ut_name);           /* the logname */
        printf(" ");                                 /* a space     */
        printf("%-8.8s", utbufp->ut_line);           /* the tty     */
        printf(" ");                                 /* a space     */
        showtime( utbufp->ut_time );                 /* display time       */
#ifdef SHOWHOST
        if ( utbufp->ut_host[0] != '\0' )
                printf(" (%s)", utbufp->ut_host);    /* the host    */
#endif
        printf("\n");                                /* newline     */
}

showtime( time_t timeval )
/*
 *      displays time in a format fit for human consumption
 *      uses ctime to build a string then picks parts out of it
 *       Note: %12.12s prints a string 12 chars wide and LIMITS
 *       it to 12chars.
 */
{
        char    *ctime();               /* convert long to ascii      */
        char    *cp;                    /* to hold address of time    */

        cp = ctime( &timeval );                 /* convert time to string     */
                                        /* string looks like          */
                                        /* Mon Feb  4 00:46:40 EST 1991 */
                                        /* 0123456789012345.          */
        printf("%12.12s", cp+4 );       /* pick 12 chars from pos 4   */
}
```

```
:::::::::::::: Makefile ::::::::::::::
#
# makefile for lecture 2
#

who1: who1.c
	cc -o who1 who1.c

who2: who2.c
	cc -o who2 who2.c

who3: who3.o utmplib.o
	cc -o who3 who3.o utmplib.o

llcopy: llcopy.c
	cc llcopy.c -o llcopy

#
# tests if llcopy works like cp
#
copytest: llcopy
	last -20 > data
	llcopy data data.copy
	if diff data data.copy ; then echo sucess ; else echo error ; fi


:::::::::::::: llcopy.c ::::::::::::::
#include	<stdio.h>
#include	<fcntl.h>
/**
 **	low level copy - uses read and write with tunable buffer size
 **	usage: llcopy src dest
 **/

#define BUFFERSIZE	4096
#define COPYMODE	0644

main( int ac, char *av[] )
{
	int	in_fd, out_fd, n_chars;		/* file descriptors and i/o count */
	char	buf[BUFFERSIZE];		/* a buffer			*/
						/*
						 *	check args
						 */
	if ( ac != 3 ){
		fprintf( stderr, "usage: %s source destination\n", *av);
		exit(1);
	}
						/*
						 *	open files
						 */
	if ( ( in_fd=open(av[1], O_RDONLY) ) == -1 )
		oops("Cannot open ", av[1]);

	if ( ( out_fd=creat( av[2], COPYMODE ) ) == -1 )
		oops( "Cannot creat", av[2]);
						/*
						 *	copy files
						 */
	while ( (n_chars = read( in_fd , buf, BUFFERSIZE )) > 0 )
		if ( write( out_fd, buf, n_chars ) != n_chars )
			oops("Write error to ", av[2]);
						/*
						 *	close them up
						 */
	if ( close(in_fd) == -1 || close(out_fd) == -1 )
		oops("Error closing files","");
}

oops(s1, s2)
char *s1, *s2;
{
	fprintf(stderr,"Error: %s%s\n", s1, s2);
	exit(1);
}
```

```
:::::::::::::: who3.c ::::::::::::::
#include       <stdio.h>
#include       <sys/types.h>
#include       <utmp.h>
#include       <fcntl.h>
#include       <time.h>

/*
 *      who version 3.0               - read /etc/utmp and list info therein
 *                             - surpresses empty records
 *                             - formats time nicely
 *                             - buffers input (using utmplib)
 */

#define SHOWHOST

int main(int ac, char **av)
{
        struct utmp    *utbufp,       /* holds pointer to next rec  */
                       *utmp_next();  /* returns pointer to next    */

        if ( utmp_open( UTMP_FILE ) == -1 ){
                fprintf(stderr,"%s: cannot open %s\n", *av, UTMP_FILE);
                exit(1);
        }
        while ( ( utbufp = utmp_next() ) != ((struct utmp *) NULL) )
                show_info( utbufp );
        utmp_close( );
        return 0;
}
/*
 *      show info()
 *                     displays the contents of the utmp struct
 *                     in human readable form
 *                     * displays nothing if record has no user name
 */
show_info( struct utmp *utbufp )
{
        if ( utbufp->ut_type != USER_PROCESS )
                return;

        printf("%-8.8s", utbufp->ut_name);           /* the logname */
        printf(" ");                                 /* a space     */
        printf("%-8.8s", utbufp->ut_line);           /* the tty     */
        printf(" ");                                 /* a space     */
        showtime( utbufp->ut_time );                 /* display time        */
#ifdef SHOWHOST
        if ( utbufp->ut_host[0] != '\0' )
                printf(" (%s)", utbufp->ut_host);    /* the host    */
#endif
        printf("\n");                                /* newline     */
}

showtime( time_t timeval )
/*
 *      displays time in a format fit for human consumption
 *      uses ctime to build a string then picks parts out of it
 *      Note: %12.12s prints a string 12 chars wide and LIMITS
 *      it to 12chars.
 */
{
        char    *ctime();              /* convert long to ascii      */
        char    *cp;                   /* to hold address of time    */

        cp = ctime( &timeval );               /* convert time to string      */
                                       /* string looks like          */
                                       /* Mon Feb  4 00:46:40 EST 1991 */
                                       /* 0123456789012345.          */
        printf("%12.12s", cp+4 );      /* pick 12 chars from pos 4   */
}
```

```
:::::::::::::: utmplib.c ::::::::::::::
/* utmplib.c  - functions to buffer reads from utmp file
 *
 *      functions are
 *              utmp_open( filename )   - open file
 *                      returns -1 on error
 *              utmp_next( )            - return pointer to next struct
 *                      returns NULL on eof
 *              utmp_close()            - close file
 *
 *      reads NRECS per read and then doles them out from the buffer
 */
#include        <stdio.h>
#include        <fcntl.h>
#include        <sys/types.h>
#include        <utmp.h>
#include       <unistd.h>

#define NRECS   16
#define NULLUT  ((struct utmp *)NULL)
#define UTSIZE  (sizeof(struct utmp))

static  char    utmpbuf[NRECS * UTSIZE];                /* storage      */
static  int     num_recs;                               /* num stored   */
static  int     cur_rec;                                /* next to go   */
static  int     fd_utmp = -1;                           /* read from    */

static  int  utmp_reload();

int utmp_open( char *filename )
{
        fd_utmp = open( filename, O_RDONLY );           /* open it      */
        cur_rec = num_recs = 0;                         /* no recs yet  */
        return fd_utmp;                                 /* report       */
}

struct utmp *utmp_next()
{
        struct utmp *recp;

        if ( fd_utmp == -1 )                            /* error ?      */
                return NULLUT;
        if ( cur_rec==num_recs && utmp_reload()==0 )    /* any more ?   */
                return NULLUT;
                                        /* get address of next record    */
        recp = (struct utmp *) &utmpbuf[cur_rec * UTSIZE];
        cur_rec++;
        return recp;
}

static int utmp_reload()
/*
 *      read next bunch of records into buffer
 */
{
        int     amt_read;

        amt_read = read(fd_utmp, utmpbuf, NRECS*UTSIZE);   /* read data    */
       if ( amt_read < 0 )                   /* mark errors as EOF   */
               amt_read = -1;

        num_recs = amt_read/UTSIZE;           /* how many did we get?       */
        cur_rec  = 0;                         /* reset pointer      */
        return num_recs;                      /* report results     */
}

int utmp_close()
{
       int rv = 0;
        if ( fd_utmp != -1 )                       /* don't close if not   */
                rv = close( fd_utmp );             /* open                 */
       return rv;
}
```