1

```
/*****************************************************************************
*****************************************************************************

        W E G A - Quelle          (C) FSU Jena RZ Bereich Systemprogrammierun

        KERN 3.2          Modul:   datem.c

        Bearbeiter:       J. Richter
        Datum:            $D$
        Version:          $R$

*****************************************************************************
*****************************************************************************/

char *xxxvers = "@(#)datem.c     1.4";

#include <stdio.h>
#include <signal.h>
#include <utmp.h>
#include <time.h>
#include <fcntl.h>
#include <ctype.h>

#define YES       1
#define NO        0
#define WTMP      "/usr/adm/wtmp"
#define UTMP      "/etc/utmp"
#define LSTR      20
#define max(a,b)  (a>b?a:b)
#define T88       (long)567990000 /* time on 1.1.88 */
#define MAXVOR    (long)24

static long int ldate, newdate, setdate;
static int year, month, day, hour, minute;
struct tm *localtime();
char *ctime(), *cmdname();
extern int daypermonth[];
static int check = YES;
static int checkchange = NO;

main(argc, argv)
char ** argv;
{
        char bu[512];
        int rb, fdc, aus();
        int cvdate(), cvtime();
        long int ls, lseek();
        struct utmp utmp;
        struct tm *tm;
        int mknocheck();
#ifndef D
        if(getuid()){
                pstr(stdout, cmdname(argv[0]));
                pstr(stdout, ": Permission denied\n");
                exit(1);
        }
        signal(SIGINT, SIG_IGN);
        signal(SIGTERM, SIG_IGN);
#endif
        signal(SIGQUIT, mknocheck);
        if(argc > 0 && strcmp(argv[1], "-s") == 0) check = NO;
        ldate = 0l;
        if((fdc = open(UTMP, O_RDONLY)) < 0){
                pstr(stdout, "\nLast date not available\n");
                check = NO;
        } else {
```

2

```
                for(;;){
                        if((rb = read(fdc, &utmp, sizeof(struct utmp))) < 1) g
                        if(rb < sizeof(struct utmp)){
                                pstr(stdout, "\nread error ");
                                pstr(stdout, UTMP);
                                check = NO;
                        } else
                                ldate = max(utmp.ut_time, ldate);
                }
e0:
                close(fdc);
        }
        setdate = ldate + 60l;
        if(setdate < T88){
                pstr(stdout, "can't accept time from ");
                pstr(stdout, UTMP);
                pstr(stdout, "\n");
                check = NO;
        }
        beep();
d0:
        if(! check) goto d3;
        pstr(stdout, "\nLast known date and time: ");
        pstr(stdout, ctime(&ldate));
d3:
        pstr(stdout, "\n");
d1:
        tm = localtime(&setdate);
        month = tm->tm_mon + 1;
        year = tm->tm_year;
        day = tm->tm_mday;
        hour = tm->tm_hour;
        minute = tm->tm_min;
#ifndef D
        alarm(300);
#else
        alarm(15);
#endif
        if(! check) pstr(stdout, "! ");
        pstr(stdout, "Enter date (MM/DD/YY or <cr>)      : ");
        signal(SIGALRM, aus);
        fgets(bu, sizeof(bu), stdin);
        alarm(0);
1       if(checkchange){
                pstr(stdout, "\n");
                checkchange = NO;
                goto d1;
        }
        if(*bu == '\n') goto d2;
        if(cvdate(bu)) goto d1;
        if(checkkal()) goto d1;
d2:
        if(! check) pstr(stdout, "! ");
        pstr(stdout, "Enter time (HH:MM)                 : ");
        fgets(bu, sizeof(bu), stdin);
        if(checkchange){
                pstr(stdout, "\n");
                checkchange = NO;
                goto d2;
        }
        if(*bu == '\n') {
                pstr(stdout, "A time must be specified\n");
                goto d2;
        }
        if(cvtime(bu)) goto d2;
        if(checktim()) goto d2;
```

3

```
        mknewtim();
        if(check){
                if(ldate > newdate){
                        pstr(stdout, "Do not set back clock\n");
                        goto d0;
                }
                else
                        if(newdate > setdate + MAXVOR*241*601*601){
                                pstr(stdout, "can't accept (too far away)\n");
                                goto d0;
                        }
        }
        aus(0);
}

cvdate(b)
char *b;
{
        register char *t = b;
        char *fields();
        register int r = 3;
        for(; *t ; t++)
                if(*t == '\n') *t = '\0';
        strcat(b, "///");
        r -= stoi  (fields(b,1,'/'), "%d", &month);
        r -= stoi  (fields(b,2,'/'), "%d", &day  );
        r -= stoi  (fields(b,3,'/'), "%d", &year );
        if(r) pstr(stdout, "Bad date format. Try again\n");
        return(r);
}

cvtime(b)
char *b;
{
        register char *t = b;
        char *fields();
        register int r = 2;
        for(; *t ; t++)
                if(*t == '\n') *t = '\0';
        strcat(b, ":::");
        r -= stoi  (fields(b,1,':'), "%d", &hour );
        r -= stoi  (fields(b,2,':'), "%d", &minute);
        if(r) pstr(stdout, "Bad time format. Try again\n");
        return(r);
}

char *fields(buff, no, ch)
char ch, *buff;
{
        static char fILSTR1;
        register int i;
        register char *s, *ptrf = f;

        s = buff;
        for(i=1; i<no; i++){
                while(*s != ch) s++;
                s++;
        }
        while(*s!=ch) *ptrf++ = *s++;
        *ptrf = '\0';
        return(f);
}

aus(sig)
{
        char s[16];
```

4

```
        spstr(s,"%02d%02d%02d%02d%02d",
                month, day, hour, minute, year);
#ifdef D
        pstr(stdout, "Date:");
        pstr(stdout, s);
        pstr(stdout, "\n");
#else
        execl("/bin/date", "date", s, 0);
        pstr(stdout, "/bin/date command missing in system!\n");
#endif
        exit(0);
}

checktim()
{
        if(hour < 0 || hour > 23
                || minute < 0 || minute > 59){
                pstr(stdout, "Bad time. Try again\n");
                return(1);
        }
        return(0);
}

checkkal()
{
        daypermonth[1] = (schaltjahr((long)year)? 29:28);
        if( month < 1 || month > 12
                        || day < 1 ||<day > daypermonth[month-1]
                        || year < 0 || year > 99){
                pstr(stdout, "Bad date. Try again\n");
                return(1);
        }
        return(0);
}

mknewtim()
{
        extern long int ktime();
        long int lday = (long)day;
        long int lmon = (long)month;
        long int lyear = (long)year;
        long int lhour = (long)hour;
        long int lmin = (long)minute;
        newdate = ktime(lday, lmon, lyear, lhour, lmin);
        return(0);
}

stoi(s, dumm, i)
char *dumm, *s;
int *i;
{
        register char *t;
        if(*s == '\0') return(0);
        *i = 0;
        for(t=s; *t; t++){
                if(!isdigit(*t)) return(0);
                *i *= 10;
                *i += (*t - '0');
        }
        return(1);
}

char *ito2s(i)
{
        static char f[] = "00";
        register di = i / 10;
```

5

```
        f[0] = di + '0';
        f[1] = (i - di * 10) + '0';
        return(f);
}

spstr(s, dumm, i1, i2, i3, i4, i5)
char *s, *dumm;
{
        strcpy(s, itoa2s(i1));
        strcat(s, itoa2s(i2));
        strcat(s, itoa2s(i3));
        strcat(s, itoa2s(i4));
        strcat(s, itoa2s(i5));
}

mknocheck()
{
        check = NO;
        checkchange = YES;
}
```

```
/*  K T I M E  -- wandelt Datum in long-Wert um (sek)                */

#include <stdio.h>

#define T88      (long)567990000 /* time on 1.1.88 */
#define TYEAR    (long)31536000  /* seconds in a common year */
#define TDAY     (long)86400     /* seconds in a day */
#define THOUR    (long)3600      /* seconds in an hour */
#define TMIN     (long)60        /* seconds in a minute */
extern long timezone;
int daypermonth [] =
{
        31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

long ktime (day, month, year, hour, min)
long int day, month, hour, year, min;
{
        long int t;
        register int i;

        if(year < 88) year += (long)100;
        t = T88;
        for (i=88; i<year; i++)          /* zyklus ueber jahre ab 1988 */
        {
                t += TYEAR;
                if(schaltjahr((long)i)) /* korrektur fuer schaltjahre */
                        t += TDAY;
        }

        if(schaltjahr(year))             /* korrektur wenn schaltjahr */
                daypermonth[1] = 29;

        month -= 2;                     /* zyklus ueber abgeschlossenen monate */
        for(i=0; i<=month; i++)
                t += ( (long) daypermonth[i] * TDAY);

        for(i=1; i<day; i++)     /* zyklus ueber tage des laufenden monats */
                t += TDAY;       /* (abgeschlossene tage!) */

        t += hour * THOUR;                /* die laufende stunde */
        t += min * TMIN;                  /* und minute */
        /* tzset(); */

        return(t  /* + timezone */ );    /*eine stunde korrektur wegen gmt-me>*/
}

schaltjahr(i)
long int i;
{
        int ii = i;
        /*
         * ein jahr ist schaltjahr, wenn die jahreszahl durch 4,
         *      aber nicht durch 100 teilbar ist.
         */
        if(((ii%4) == 0)  && ((ii%100) != 0)) return(1);
        return(0);
}
```