

```

; *****
;
; CTIME.INC (Retro Unix 8086 v1 - /bin/ls - list file or directory)
; -----
;
; RETRO UNIX 8086 (Retro Unix == Turkish Rational Unix)
; Operating System Project (v0.1) by ERDOGAN TAN (Beginning: 11/07/2012)
; Retro UNIX 8086 v1 - /bin/ls file
;
; [ Last Modification: 28/11/2013 ]
;
; Derivation from UNIX Operating System (v1.0 for PDP-11)
; (Original) Source Code by Ken Thompson (Bell Laboratories, 1971-1972)
;
; *****
;
; Derived from 'ctime.c' file of original UNIX v5 (usr/source/s3/ctime.c)
;
; LS0.ASM, 19/11/2013 - 24/11/2013
; include ctime.inc
;
; *****

; .8086

;timezone equ 5*60*60

ctime:  ; ctime(at)
        ; int *at;
        ; {
        ;     return(asctime(localtime(at)));
        ; }

        ; DX:AX = unix epoch time (in seconds)

        ;call localtime
        ;call asctime

        ;retn

localtime:
        ; localtime(tim)
        ; int tim[];
        ; {
        ;     register int *t, *ct, dayno;
        ;     int daylbegin, daylend;
        ;     int copyt[2];
        ;     t = copyt;
        ;     t[0] = tim[0];
        ;     t[1] = tim[1];
        ;     dpadd(t, -timezone);
        ;     ct = gmtime(t);
        ;     dayno = ct[YDAY];
        ;     if (nixonflg && (ct[YEAR]>74 || ct[YEAR]==74 && (dayno > 5 ||
        ;         dayno==5 && ct[HOURL]>=2))) {
        ;         daylight = 1;
        ;         daylbegin = -1;
        ;         daylend = 367;
        ;     } else {
        ;         daylbegin = sunday(ct, 119); /* last Sun in Apr */
        ;         daylend = sunday(ct, 303); /* last Sun in Oct */
        ;     }
        ;     if (daylight &&
        ;         (dayno>daylbegin || (dayno==daylbegin && ct[HOURL]>=2)) &&
        ;         (dayno<daylend || (dayno==daylend && ct[HOURL]<1))) {
        ;         dpadd(t, 1*60*60);
        ;         ct = gmtime(t);
        ;         ct[ISDAY]++;
        ;     }
        ;     return(ct);
        ; }

        ;sub    ax, timezone
        ;sbb    dx, 0

        ;push   dx
        ;push   ax

```

```

        call    gmtime
; if (nixonflg && (ct[YEAR]>74 || ct[YEAR]==74 && (dayno > 5 ||
;   dayno==5 && ct[HOURL]>=2))) {
        ;cmp    byte ptr [nixonflg], 0
        ;jna    short lt1
        ;cmp    word ptr [year], 1974
        ;jb     short lt1
        ;ja     short lt0
        ;cmp    word ptr [yday], 5
        ;jb     short lt1
        ;ja     short lt0
        ;cmp    word ptr [hour], 2
        ;jb     short lt1
; nixonflag > 0
;lt0:
        ;mov     word ptr [daylight], 1
        ;mov     word ptr [daylbegin], -1
        ;mov     word ptr [daylend], 367
;       ;jmp     short lt2

; } else {
;lt1:
;       mov     cx, 119
;       call    sunday ; sunday(ct, 119); /* last Sun in Apr */
;       mov     word ptr [daylbegin], cx
;       mov     cx, 303
;       call    sunday ; sunday(ct, 303); /* last Sun in Oct */
;       mov     word ptr [daylend], cx
;lt2:
; if (daylight &&
;   (dayno>daylbegin || (dayno==daylbegin && ct[HOURL]>=2)) &&
;   (dayno<daylend || (dayno==daylend && ct[HOURL]<1))) {

;       pop     ax
;       pop     dx

        ;cmp     byte ptr [daylight], 0
        ;jna     short lt5

        ;mov     cx, word ptr [yday]

        ;cmp     cx, word ptr [daylbegin]
        ;jb     short lt5
        ;ja     short lt3
        ;cmp     word ptr [hour], 2
        ;jb     short lt5
        ;jmp     short lt4
;lt3:
        ;cmp     cx, word ptr [daylend]
        ;jb     short lt4
        ;ja     short lt5
        ;cmp     word ptr [hour], 1
        ;jnb     short lt5
;lt4:
        ;add     ax, 1*60*60
        ;adc     dx, 0
        ;call    gmtime
        ;inc     word ptr [isday]
;lt5:
;       }
;       return(ct);
;       }

        ;retn

asctime:
; asctime(t)
;int *t;
;{
;       register char *cp, *ncp;
;       register int *tp;
;
;       cp = cbuf;
;       for (ncp = "Day Mon 00 00:00:00 1900\n"; *cp++ = *ncp++);
;       ncp = &"SunMonTueWedThuFriSat"[3*t[6]];
;       cp = cbuf;

```

```

;      *cp++ = *ncp++;
;      *cp++ = *ncp++;
;      *cp++ = *ncp++;
;      cp++;
;      tp = &t[4];
;      ncp = &"JanFebMarAprMayJunJulAugSepOctNovDec"[( *tp)*3];
;      *cp++ = *ncp++;
;      *cp++ = *ncp++;
;      *cp++ = *ncp++;
;      cp = numb(cp, *--tp);
;      cp = numb(cp, *--tp+100);
;      cp = numb(cp, *--tp+100);
;      cp = numb(cp, *--tp+100);
;      cp =+ 2;
;      cp = numb(cp, t[YEAR]);
;      return(cbuf);
;}

;;mov  di, offset Cbuf
;;mov  si, offset ncp0
;;mov  cx, 13
;;movsw
;
mov     di, offset Cbuf
;mov    si, word ptr [wday]
;shl    si, 1
;shl    si, 1
;add    si, offset ncpl
;movsw
;movsw
mov     si, word ptr [month]
shl     si, 1
shl     si, 1
add     si, offset ncp2 - 4
movsw
movsw
mov     ax, word ptr [day]
;mov    cx, 10
mov     cl, 10
call    numb
mov     al, 20h
stosb
;
mov     ax, word ptr [year]
mov     ch, 100
div     ch
push    ax ;
cbw ; century (19, 20)
call    numb
pop     ax
mov     al, ah
cbw ; year (0 to 99)
call    numb
mov     al, 20h
stosb
;
mov     si, word ptr [wday]
shl     si, 1
shl     si, 1
add     si, offset ncpl
movsw
movsw
;
mov     ax, word ptr [hour]
call    numb
mov     al, ':'
stosb
mov     ax, word ptr [minute]
call    numb
mov     al, ':'
stosb
mov     ax, word ptr [second]
call    numb
mov     al, 20h
stosb
;mov    ax, word ptr [year]
;mov    ch, 100
;div    ch

```

```

;push  ax ;
;cbw ; century (19, 20)
;call  numb
;pop   ax
;mov   al, ah
;cbw ; year (0 to 99)
;call  numb
;mov   al, 20h
;stosb
;xor   al, al
;stosb

retn

gmtime:
; 24/11/2013 (yday, wday)
; Retro UNIX 8086 v1 - UNIX.ASM (20/06/2013)
; Retro UNIX 8086 v1 feature/procedure only!
; 'convert_from_epoch' procedure prototype:
;         UNIXCOPY.ASM, 10/03/2013
; 30/11/2012
; Derived from DALLAS Semiconductor
; Application Note 31 (DS1602/DS1603)
; 6 May 1998
;
; INPUT:
; DX:AX = Unix (Epoch) Time
;
; ((Modified registers: AX, DX, CX, BX))
;
mov cx, 60
call div32
;mov word ptr [imin], ax ; whole minutes
;mov word ptr [imin]+2, dx ; since 1/1/1970
mov word ptr [second], bx ; leftover seconds
; mov cx, 60
call div32
;mov word ptr [ihrs], ax ; whole hours
;mov word ptr [ihrs]+2, dx ; since 1/1/1970
mov word ptr [minute], bx ; leftover minutes
; mov cx, 24
mov cl, 24
call div32
;mov word ptr [iday], ax ; whole days
; ; since 1/1/1970
mov word ptr [wday], ax ; 24/11/2013
; mov word ptr [iday]+2, dx ; DX = 0
mov word ptr [hour], bx ; leftover hours
add ax, 365+366 ; whole day since
; ; 1/1/1968
; adc dx, 0 ; DX = 0
; mov word ptr [iday], ax
push ax
mov cx, (4*365)+1 ; 4 years = 1461 days
call div32
pop cx
;mov word ptr [lday], ax ; count of quadyrs (4 years)
push bx
;mov word ptr [qday], bx ; days since quadyr began
cmp bx, 31 + 29 ; if past feb 29 then
cmc ; add this quadyr's leap day
adc ax, 0 ; to # of qadyrs (leap days)
;mov word ptr [lday], ax ; since 1968
;mov cx, word ptr [iday]
xchg cx, ax ; CX = lday, AX = iday
sub ax, cx ; iday - lday
mov cx, 365
;xor dx, dx ; DX = 0
; AX = iday-lday, DX = 0
call div32
;mov word ptr [iyrs], ax ; whole years since 1968
; jday = iday - (iyrs*365) - lday
;mov word ptr [jday], bx ; days since 1/1 of current year
add ax, 1968 ; compute year
mov word ptr [year], ax
mov dx, ax
;mov ax, word ptr [qday]
pop ax

```

```

    cmp ax, 365                ; if qday <= 365 and qday >= 60
    ja short @f                ; jday = jday + 1
    cmp ax, 60                 ; if past 2/29 and leap year then
    cmc                        ; add a leap day to the # of whole
    adc bx, 0                   ; days since 1/1 of current year
@@:
    ; mov word ptr [jday], bx
    ; mov word ptr [yday], bx ; 24/11/2013
    mov cx, 12                 ; estimate month
    xchg cx, bx                ; CX = jday, BX = month
    mov ax, 366                 ; mday, max. days since 1/1 is 365
    and dx, 11b                ; year mod 4 (and dx, 3)
@@:
    ; Month calculation        ; 0 to 11 (11 to 0)
    cmp cx, ax                  ; mday = # of days passed from 1/1
    jnb short @f
    dec bx                      ; month = month - 1
    shl bx, 1
    mov ax, word ptr DMonth[BX] ; # elapsed days at 1st of month
    shr bx, 1                   ; bx = month - 1 (0 to 11)
    cmp bx, 1                   ; if month > 2 and year mod 4 = 0
    jna short @b                ; then mday = mday + 1
    or dl, dl                   ; if past 2/29 and leap year then
    jnz short @b                ; add leap day (to mday)
    inc ax                      ; mday = mday + 1
    jmp short @b
@@:
    inc bx                      ; -> bx = month, 1 to 12
    mov word ptr [month], bx
    sub cx, ax                  ; day = jday - mday + 1
    inc cx
    mov word ptr [day], cx

    ; ax, bx, cx, dx is changed at return
    ; output ->
    ; [year], [month], [day], [hour], [minute], [second]
    ; [yday] -> 24/11/2013
    ; [wday] -> 24/11/2013
    ;
    ; 24/11/2013
    mov ax, word ptr [wday] ; [iday]
    xor dl, dl ; xor dx, dx
    ; dx = 0
    add ax, 4
    ; NOTE: January 1, 1970 was THURSDAY
    ; ch = 0
    mov cl, 7
    div cx
    mov word ptr [wday], dx ; week of the day, 0 to 6
    ; 0 = sunday ... 6 = saturday
    ; mov word ptr [isday], 0

    retn

div32:
    ; Input -> DX:AX = 32 bit dividend
    ; CX = 16 bit divisor
    ; output -> DX:AX = 32 bit quotient
    ; BX = 16 bit remainder
    mov bx, dx
    xchg ax, bx
    xor dx, dx
    div cx                      ; at first, divide DX
    xchg ax, bx                 ; remainder is in DX
                                ; now, BX has quotient
                                ; save remainder
    div cx                      ; so, DX:AX divided and
                                ; AX has quotient
                                ; DX has remainder
    xchg dx, bx                 ; finally, BX has remainder

    retn

;sunday:
    ; sunday(at, ad)
    ; int *at;
    ; {
    ; register int *t, d;
    ; t = at;

```

```

;      d = ad;
;      d = ad + dysize(t[YEAR]) - 365;
;      return(d - (d - t[YDAY] + t[WDAY] + 700) % 7);
;      }

;mov    dx, word ptr [year]
;call   dysize
;sub     ax, 365
; add    cx, ax
;      test word ptr [year], 11b
;      jnz short @f
;      ; CX = 119 (77h) or CX = 303 (12Fh)
;      inc cx
;      inc cl
;@@:
;      mov ax, cx
;      add ax, word ptr [wday]
;      adc ax, 700
;      add ax, 700
;      sub ax, word ptr [yday]
;      xor dx, dx
;      mov bx, 7
;      div bx
;      div bl
;      sub cx, bx
;      retn

;dysize:
; dysize(y)
; {
;     if((y%4) == 0)
;         return(366);
;     return(365);
; }

;      mov ax, 365
;      test dx, 11b
;      jnz short @f
;      inc ax
;      inc al
;@@:
;      retn

numb:    ; AX = 0 to 99
;
;      numb(acp, n)
;      {
;          register char *cp;
;
;          cp = acp;
;          cp++;
;          if (n>=10)
;              *cp++ = (n/10)%10 + '0';
;          else
;              *cp++ = ' ';
;          *cp++ = n%10 + '0';
;          return(cp);
;      }
;
;mov     cl, 10
;cmp     ax, 10
;jnb     short nb1
;mov     ah, al
;xor     al, al ; 0
;jmp     short nb2

nb1:
;div     cl
;mov     dl, ah

nb2:
;add     al, '0'
;stosb   ; digit 1
;mov     al, ah
;add     al, '0'
;stosb   ; digit 2
;retn

```

```

;;; DATA

;daylight: db 1 ; int daylight 1; /* Allow daylight conversion */
;nixonflg: db 0 ; int nixonflg 0; /* Daylight time all year around */
;daylbegin: dw 0
;daylend: dw 0

ct:
; 24/11/2013 (re-order)
;
; Retro UNIX 8086 v1 - UNIX.ASM
; 09/04/2013 epoch variables
; Retro UNIX 8086 v1 Prototype: UNIXCOPY.ASM, 10/03/2013
;

second: dw 0
minute: dw 0
hour: dw 0
day: dw 1
month: dw 1
year: dw 1970
wday: dw 0 ; 24/11/2013
yday: dw 0 ; 24/11/2013
yday: dw 0 ; 24/11/2013
yday: dw 0 ; 24/11/2013

DMonth:
dw 0
dw 31
dw 59
dw 90
dw 120
dw 151
dw 181
dw 212
dw 243
dw 273
dw 304
dw 334

;ncp0: db "Day Mon 00 00:00:00 1970", 0, 0
;ncp1: db "Sun Mon Tue Wed Thu Fri Sat "
;ncp2: db "Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec "

cbuf: ; char cbuf[26]
      db 26 dup(0)

COMMENT @
;; ctime.c (Unix v5)

#
/*
* This routine converts time as follows.
* The epoch is 0000 Jan 1 1970 GMT.
* The argument time is in seconds since then.
* The localtime(t) entry returns a pointer to an array
* containing
* seconds (0-59)
* minutes (0-59)
* hours (0-23)
* day of month (1-31)
* month (0-11)
* year-1970
* weekday (0-6, Sun is 0)
* day of the year
* daylight savings flag
*
* The routine corrects for daylight saving
* time and will work in any time zone provided
* "timezone" is adjusted to the difference between
* Greenwich and local standard time (measured in seconds).
* In places like Michigan "daylight" must
* be initialized to 0 to prevent the conversion
* to daylight time.
*
* "nixonflg", if set to 1, will
* cause daylight savings time all year around

```

```

* independently of "daylight".
*
* The routine does not work
* in Saudi Arabia which runs on Solar time.
*
* asctime(tvec))
* where tvec is produced by localtime
* returns a ptr to a character string
* that has the ascii time in the form
*   Thu Jan 01 00:00:00 1970n0000\
*   01234567890123456789012345
*   0         1         2
*
* ctime(t) just calls localtime, then asctime.
*/
char    cbuf[26];
int     dmsize[12]
{
    31,
    28,
    31,
    30,
    31,
    30,
    31,
    31,
    30,
    31,
    30,
    31
};

int timezone    5*60*60;
int tzname[]
{
    "EST",
    "EDT",
};
int    daylight 1;    /* Allow daylight conversion */
int    nixonflg 0;    /* Daylight time all year around */

#define SEC      0
#define MIN      1
#define HOUR     2
#define MDAY     3
#define MON      4
#define YEAR     5
#define WDAY     6
#define YDAY     7
#define ISDAY    8

ctime(at)
int *at;
{
    return(asctime(localtime(at)));
}

localtime(tim)
int tim[];
{
    register int *t, *ct, dayno;
    int daylbegin, daylend;
    int copyt[2];

    t = copyt;
    t[0] = tim[0];
    t[1] = tim[1];
    dpadd(t, -timezone);
    ct = gmtime(t);
    dayno = ct[YDAY];
    if (nixonflg && (ct[YEAR]>74 || ct[YEAR]==74 && (dayno > 5 ||
        dayno==5 && ct[ HOUR]>=2))) {
        daylight = 1;
        daylbegin = -1;
        daylend = 367;
    } else {
        daylbegin = sunday(ct, 119); /* last Sun in Apr */
        daylend = sunday(ct, 303); /* last Sun in Oct */
    }
}

```



```

    }
    if (daylight &&
        (dayno>daylbegin || (dayno==daylbegin && ct[HOURL]>=2)) &&
        (dayno<daylend || (dayno==daylend && ct[HOURL]<1))) {
        dpadd(t, 1*60*60);
        ct = gmtime(t);
        ct[ISDAY]++;
    }
    return(ct);
}

sunday(at, ad)
int *at;
{
    register int *t, d;

    t = at;
    d = ad;
    d = ad + dysize(t[YEAR]) - 365;
    return(d - (d - t[YDAY] + t[WDAY] + 700) % 7);
}

gmtime(tim)
int tim[];
{
    register int d0, d1;
    register *tp;
    static xtime[9];
    extern int ldivr;

    /*
     * break initial number into
     * multiples of 8 hours.
     * (28800 = 60*60*8)
     */

    d0 = ldiv(tim[0], tim[1], 28800);
    d1 = ldivr;
    tp = &xtime[0];

    /*
     * generate hours:minutes:seconds
     */

    *tp++ = d1%60;
    d1 /= 60;
    *tp++ = d1%60;
    d1 /= 60;
    d1 += (d0%3)*8;
    d0 /= 3;
    *tp++ = d1;

    /*
     * d0 is the day number.
     * generate day of the week.
     */

    xtime[WDAY] = (d0+4)%7;

    /*
     * year number
     */
    for(d1=70; d0 >= dysize(d1); d1++)
        d0 -= dysize(d1);
    xtime[YEAR] = d1;
    xtime[YDAY] = d0;

    /*
     * generate month
     */

    if (dysize(d1)==366)
        dmsize[1] = 29;
    for(d1=0; d0 >= dmsize[d1]; d1++)
        d0 -= dmsize[d1];
    dmsize[1] = 28;
    *tp++ = d0+1;
    *tp++ = d1;

```

```

        xtime[ISDAY] = 0;
        return(xtime);
    }

asctime(t)
int *t;
{
    register char *cp, *ncp;
    register int *tp;

    cp = cbuf;
    for (ncp = "Day Mon 00 00:00:00 1900\n"; *cp++ = *ncp++;)
    ncp = &"SunMonTueWedThuFriSat"[3*t[6]];
    cp = cbuf;
    *cp++ = *ncp++;
    *cp++ = *ncp++;
    *cp++ = *ncp++;
    cp++;
    tp = &t[4];
    ncp = &"JanFebMarAprMayJunJulAugSepOctNovDec"[( *tp)*3];
    *cp++ = *ncp++;
    *cp++ = *ncp++;
    *cp++ = *ncp++;
    cp = numb(cp, *--tp);
    cp = numb(cp, *--tp+100);
    cp = numb(cp, *--tp+100);
    cp = numb(cp, *--tp+100);
    cp += 2;
    cp = numb(cp, t[YEAR]);
    return(cbuf);
}

dysize(y)
{
    if((y%4) == 0)
        return(366);
    return(365);
}

numb:

numb(acp, n)
{
    register char *cp;

    cp = acp;
    cp++;
    if (n>=10)
        *cp++ = (n/10)%10 + '0';
    else
        *cp++ = ' ';
    *cp++ = n%10 + '0';
    return(cp);
}

@

```