

```

; *****
;
; UNIXFDFS.ASM
; -----
;
; RETRO UNIX 8086 (Retro Unix == Turkish Rational Unix)
; Operating System Project (v0.1) by ERDOGAN TAN (Beginning: 11/07/2012)
; 1.44 MB Floppy Disk
; Bootable Unix (RUFFS) File System Installation/Formatting Code
;
; UNIXFDFS.ASM -> Last Modification: 21/04/2014
;
; Derivation from UNIX Operating System (v1.0 for PDP-11)
; (Original) Source Code by Ken Thompson (1971-1972)
; <Bell Laboratories (17/3/1972)>
; <Preliminary Release of UNIX Implementation Document>
;
; *****

; 21/04/2014 (tty8=COM1, tty9=COM2)
; 22/12/2013
; 09/07/2013

RUFFS_INSTL      SEGMENT PUBLIC 'CODE'
                  assume cs:RUFFS_INSTL,ds:RUFFS_INSTL,es:RUFFS_INSTL,ss:RUFFS_INSTL

ruffs_fd_format  proc near
                  ; 28/10/2012
                  ; 19/9/2012
                  ; 14/8/2012
                  ; 13/8/2012
                  ; 12/8/2012

                  org 100h

INSTALL:

; - - - - -
; see if drive specified
; - - - - -

                mov si, offset 80h                ; PSP command tail
                mov cl, byte ptr [SI]
                or cl, cl
                jz short ruffs_fd_format_7         ; jump if zero

ruffs_fd_format_1:
                inc si
                mov al, byte ptr [SI]
                cmp al, ' '                        ; is it SPACE ?
                jne short ruffs_fd_format_2

                dec cl
                jne short ruffs_fd_format_1
                jmp short ruffs_fd_format_7

ruffs_fd_format_2:
                cmp al, "f"
                jne short ruffs_fd_format_3
                inc si
                mov al, byte ptr [SI]
                cmp al, "d"
                jne short ruffs_fd_format_7
                inc si
                mov ax, word ptr [SI]
                cmp al, '0'
                jb short ruffs_fd_format_7
                cmp al, '1'
                ja short ruffs_fd_format_7
                cmp ah, 20h
                ja short ruffs_fd_format_7
                mov byte ptr [RUFFS_DRIVE], al
                sub al, '0'
                jmp short ruffs_fd_format_5

```

```

rufs_fd_format_3:
    cmp al, 'A'
    jb short rufs_fd_format_7
    cmp al, 'B'                                ; A - Z
    jna short rufs_fd_format_4
    cmp al, 'a'                                ; a - z
    jb short rufs_fd_format_7
    cmp al, 'b'
    ja short rufs_fd_format_7

    sub al, 'a'-'A'                            ; to upper case

; - - - - -
; Write message
; - - - - -

rufs_fd_format_4:
    mov byte ptr [RUFSDRIVE], al
    sub al, 'A'                                ; make it zero based

rufs_fd_format_5:
    mov dl, al
    mov byte ptr [bsDriveNumber], dl
    mov ah, 08h
    int 13h                                    ; return disk parameters
    push cs
    pop es                                    ; restore es
    jc rufs_fd_format_17

    cmp bl, 04h                                ; Drive Type
    jb rufs_fd_format_17

    mov si, offset Msg_DoYouWantToFormat
    call PRINT_MSG

rufs_fd_format_6:
    xor ax, ax
    int 16h                                    ; wait for keyboard command
    cmp al, 'C'-40h
    je short rufs_fd_format_8
    cmp al, 27
    je short rufs_fd_format_8
    and al, 0DFh
    cmp al, 'Y'                                ; Yes?
    je short rufs_fd_format_10                 ; write
    cmp al, 'N'                                ; No?
    je short rufs_fd_format_9                 ; no write (exit)

rufs_fd_format_7:
    mov si, offset UNIX_Welcome
    call PRINT_MSG

rufs_fd_format_8:
    mov si, offset UNIX_CRLF
    call PRINT_MSG

    int 20h

infinite_loop: jmp short infinite_loop

rufs_fd_format_9:
    mov si, offset msg_NO
    call PRINT_MSG

    jmp short rufs_fd_format_8

```

```

;- - - - -
; get drive parameters
;- - - - -

rufs_fd_format_10:
    mov si, offset msg_YES
    call PRINT_MSG

rufs_fd_format_11:
    xor ax, ax
    int 1Ah                                ; get time of day
    mov word ptr [bsVolumeSerial], dx
    mov word ptr [bsVolumeSerial]+2, cx    ; set unique volume ID

rufs_fd_format_12:
    mov si, offset Msg_installing_file_system
    call PRINT_MSG

    mov dl, byte ptr [bsDriveNumber] ; 14/8/2012

    call unix_fs_install
    jnc short rufs_fd_format_14

    mov ah, byte ptr [Error]

rufs_fd_format_13: ; loc_rw_error
    mov al, ah
    push ax
    mov si, offset Msg_Disk_RW_Error
    call PRINT_MSG
    pop ax
    call proc_hex
    mov word ptr [Str_Err], ax
    mov si, Offset Msg_Error_Number
    call PRINT_MSG

    int 20h

rufs_fd_format_14:
    mov si, offset Msg_OK
    call PRINT_MSG

rufs_fd_format_15:
    mov si, offset Msg_writing_boot_sector
    call PRINT_MSG

    mov byte ptr [RetryCount], 4

rufs_fd_format_16:
    mov ax, 0301h                                ; write to disk

    mov bx, offset Start                        ; location of boot code

    mov cx, 1                                ; cylinder = 0
                                           ; sector = 1
    mov dh, 0                                ; head = 0
    mov dl, byte ptr [bsDriveNumber]

    int 13h
    jnc short rufs_fd_format_17
    dec byte ptr [RetryCount]
    jnz short rufs_fd_format_16

    jmp short rufs_fd_format_13

rufs_fd_format_17:
    mov si, offset Msg_OK
    call PRINT_MSG

    ;int 20h
    jmp rufs_fd_format_8

rufs_fd_format endp

```

```

PRINT_MSG      proc near
                mov     BX,07h
                mov     AH,0Eh

PRINT_MSG_LOOP:
                lodsb                    ; Load byte at DS:SI to AL
                and     AL,AL
                jz      short PRINT_MSG_OK

                int     10h                ; BIOS Service func ( ah ) = 0Eh
                                           ; Write char as TTY
                                           ; ↑AL-char BH-page BL-color
                jmp     short PRINT_MSG_LOOP

PRINT_MSG_OK:
                retn

PRINT_MSG      endp

proc_hex        proc      near

                db 0D4h,10h                ; Undocumented inst. AAM
                                           ; AH = AL / 10h
                                           ; AL = AL MOD 10h
                or  AX,'00'                ; Make it ZERO (ASCII) based

                xchg  AH,AL

; 1999
                cmp  AL,'9'
                jna  pass_cc_al
                add  AL,7

pass_cc_al:
                cmp  AH,'9'
                jna  pass_cc_ah
                add  AH,7

pass_cc_ah:

; 1998
                retn

proc_hex        endp

;;;;
include          uninstall.asm
include          unixproc.asm
;;;;

; - - - - -
;  messages
; - - - - -

UNIX_Welcome:
                db 0Dh, 0Ah
                db 'RETRO UNIX 1.44 MB Floppy Disk (RUFFS) Format Utility'
                db 0Dh, 0Ah
                db 'by Erdogan TAN [21/04/2014]'
                db 0Dh,0Ah
                db 0Dh,0Ah
                db 'Usage: unixfdfs [Drive] '
                db 0Dh,0Ah
                db 0Dh,0Ah
                db "Drive names:"
                db 0Dh,0Ah
                db 0Dh,0Ah
                db "fd0      (Floppy Disk 1)", 0Dh, 0Ah
                db "fd1      (Floppy Disk 2)", 0Dh, 0Ah
                db "...", 0Dh, 0Ah
                db "A:      (Floppy Disk 1)", 0Dh, 0Ah
                db "B:      (Floppy Disk 2)", 0Dh, 0Ah
                db 0Dh, 0Ah
                db 0

```

```

Msg_DoYouWantToFormat:
    db 07h
    db 0Dh, 0Ah
    db 'WARNING!'
    db 0Dh, 0Ah
    db 'All data on the drive will be erased.'
    db 0Dh, 0Ah
    db 0Dh, 0Ah
    db 'Do you want to format drive '

RUFSD_DRIVE:
    db 'A: (Yes/No)? ', 0

Msg_Installing_File_System:
    db 0Dh, 0Ah
    db "Installing UNIX v1 File Sytem...", 0

Msg_Writing_Boot_Sector:
    db 0Dh, 0Ah
    db "Writing UNIX boot sector...", 0

Cursor_Pos:    dw 0

Msg_Volume_Name:
    db 0Dh, 0Ah
    db "Volume Name: ", 0

Msg_OK:
    db ' OK.', 0

msg_YES:
    db ' YES'
    db 0
msg_NO:
    db ' NO'
    db 0

; 12/8/2012
msg_disk_rw_error:
    db 0Dh, 0Ah
    db 'Disk r/w error!'
    db 0

msg_error_Number:
    db 0Dh, 0Ah
    db 'Error No:'
str_err:
    dw 3030h
    db 'h'
UNIX_CRLF:
    db 0Dh, 0Ah, 0

Error_Code:    db 0

RetryCount:    dw 0

str_volume_name: db 15 dup (0)

    db 'Turkish Rational UNIX', 0
    db 'RETRO UNIX 8086 by Erdogan TAN', 0
    db '11/07/2012', 0, '21/04/2014', 0

    db 1 dup (?)
                                ; trick for assembler
                                ; to keep 'start'
                                ; at 7C00h

```

```

BF_BUFFER equ 700h
BF_INODE equ 600h
inode_flg equ 600h
inode_nlks equ 602h
inode_uid equ 603h
inode_size equ 604h
inode_dskp equ 606h
inode_ctim equ 616h
inode_mtim equ 61Ah
inode_reserved equ 61Eh

boot_file_load_address equ 7E00h
boot_file_segment equ 7E0h

        org 7C00h

;+++++
;±
;±          PROCEDURE unixbootsector
;±
;+++++

unixbootsector proc    near

Start:
        jmp     short @f

; RETRO UNIX 8086 FS v0.1 BootSector Identification (Data) Block
; 29-10-2012 RUFS 1.44MB FD Boot Sector

bsFSystemID:    db 'RUFS'
bsVolumeSerial: dd 0
                db 'fd'
bsDriveNumber:  db 0
bsReserved:     db 0 ; 512 bytes per sector
bsSecPerTrack:  db 18
bsHeads:        db 2
bsTracks:       dw 80
bs_BF_I_number: dw 0
                db '@'

@@:
        mov ax, cs
        mov ds, ax
        mov es, ax

        cli
        mov ss, ax
        mov sp, 0FFFEh
        sti

        mov ax, word ptr [bs_BF_I_number]

        or ax, ax
        jz loc_no_bootable_disk

        mov byte ptr [bsDriveNumber], DL ; from INT 19h

        ;;call load_boot_file
        ;;jc short loc_unix_bl_error
load_boot_file:
        ;; 22/12/2013
        ;; 28/10/2012
        ;; 20/10/2012
        ;;
        ;; RETRO UNIX v1 FS
        ;; Boot sector version
        ;;
        ;; loads boot file
        ;;
        ;; ax = i-number

```

```

load_bf_1:
i_get:
    ;; 22/12/2013
    ; 20/10/2010 (i_i)
    ; 14/10/2012
    ; boot sector version of "iget" procedure
    ; Derived from (original) UNIX v1 source code
    ; PRELIMINARY release of Unix Implementation Document,
    ; 20/6/1972
    ; input -> AX = inode number
    ; RETRO UNIX v1 FS
    ; boot sector version
    ;; return => if cf=1 error number in [Error]

    ;;cmp ax, word ptr [i_i] ; AX (R1) = i-number of current file
    ;;je short i_get_3

    ;; mov di, ax ; i-number
    add ax, 47 ; add 47 to inode number
    push ax ;
    shr ax, 1 ; divide by 16
    shr ax, 1
    shr ax, 1
    shr ax, 1
        ; ax contains block number of block in which
        ; inode exists
    call dsk_rd
    pop dx ;
    ;;jc short i_get_3 ; Error code in AH
    jc loc_unix_bl_error

    ;;mov word ptr [i_i], di
i_get_1:
    and dx, 0Fh ; (i+47) mod 16
    shl dx, 1
    shl dx, 1
    shl dx, 1
    shl dx, 1
    shl dx, 1
        ; DX = 32 * ((i+47) mod 16)
        ; DX points to first word in i-node i.
    mov di, BF_INODE
        ; inode is address of first word of current inode
    mov cx, 16 ;

    mov si, bx ; offset Buffer

    add si, dx
i_get_2:
    ; copy new i-node into inode area of (core) memory
    rep movsw
;;i_get_3:
    ;;retn

lbf_2:
    ;; 22/12/2013

    mov bx, inode_flg

    test word ptr [bx], 10h ; executable file attribute bit
    ;;jz short load_bf_stc
    jz loc_unix_bl_error

    mov bx, inode_size ; offset

    ;; 22/12/2013
    ;;cmp word ptr [bx], 0
    ;;jna short load_bf_stc
    ;;jna short loc_unix_bl_error
    mov ax, word ptr [bx]
    and ax, ax
    jz loc_unix_bl_error

    mov word ptr [b_base], boot_file_load_address

    ;xor ax, ax
    ;mov word ptr [b_off], ax ; u_off is file offset

```

```

;; 22/12/2013
xor dx, dx
mov word ptr [b_off], dx ; u_off is file offset

;mov bx, inode_size
;;mov ax, word ptr [bx]
mov word ptr [b_count], ax

;;mov ax, word ptr [i_i]
;;call read_i
;;jc short load_bf_retn

read_i:
;; 22/12/2013
; 28/10/2012
; 14/10/2012
; Boot sector version of "readi" procedure
; Derived from (original) UNIX v1 source code
; PRELIMINARY release of Unix Implementation Document,
; 20/6/1972
;;AX (R1) = i-number
; RETRO UNIX v1 FS
; Boot sector version
;
; read from an i-node
;
; xor dx, dx ; 0
mov word ptr [b_nread], dx ; accumulated number of bytes transmitted
;;cmp word ptr [b_count], dx ; is number of byte to read greater than 0
;;jna short read_i_retn

read_i_1:
; AX = I-Number
;;push ax
;;call i_get ; get i-node into i-node section of core
mov bx, inode_size
mov dx, word ptr [bx] ; file size in bytes in r2 (DX)
sub dx, word ptr [b_off] ; subtract file offset
;;jna short read_i_3
; jna read_i_retn ; 22/12/2013
cmp dx, word ptr [b_count]
; are enough bytes left in file to carry out read
jnb short read_i_2
mov word ptr [b_count], dx

read_i_2:
;;call m_get ; returns physical block number of block in file
;; ; where offset points

m_get:
;; 22/12/2013
; 05/03/2013
; 03/03/2013
; 28/10/2012
; 20/10/2012
; Boot sector version of "mget" procedure
; Derived from (original) UNIX v1 source code
; PRELIMINARY release of Unix Implementation Document,
; 20/6/1972
;

m_get_0:
mov bl, byte ptr [b_off]+1
xor bh, bh
mov si, inode_flg
test word ptr [si], 4096 ; 1000h
; is this a large or small file
jnz short m_get_1 ; large file

test bl, 0F0h ; !0Fh ; error if BX (R2) >= 16
jnz short m_get_5

and bl, 0Eh ; clear all bits but bits 1,2,3
mov ax, word ptr inode_dskp[bx] ; AX = R1, physical block number

jmp short m_get_3

```



```

m_get_1:      ; large file
               ; 05/03/2013
               ; 03/03/2013
               ;mov ax, bx
               ;mov cx, 256
               ;xor dx, dx
               ;div cx
               ;and bx, 1FEh ; zero all bit but 1,2,3,4,5,6,7,8
               ; gives offset in indirect block
               ;push bx
               ;mov bx, ax ; calculate offset in i-node for pointer
               ; to proper indirect block
               ;and bx, 0Eh
               ;mov ax, word ptr inode_dskp[bx]
               and bl, 0FEh
               ;push bx
               ; mov di, bx
               mov si, bx ; 22/12/2013
               mov bx, inode_dskp
               mov ax, word ptr [BX]
               or ax, ax
               ;jz short m_get_4
               jz short loc_unix_bl_error ; 22/12/2013

m_get_2:
               call dsk_rd ; read indirect block
               ;jc short m_get_5
               jc short loc_unix_bl_error ; 22/12/2013
               ;pop ax
               ;add bx, ax ; R5, first word of indirect block
               ;add bx, di
               add bx, si ; 22/12/2013
               mov ax, word ptr [BX] ; put physical block no of block
               ; in file sought in R1 (AX)

m_get_3: ; 2
               ; ax = R1, block number of new block
               ;cmp ax, 1
               ;retn
               or ax, ax
               jz short loc_unix_bl_error ; 22/12/2013

m_get_4:
               ;stc

m_get_5:
               ;pop bx
               ;retn
               ;jc short loc_unix_bl_error ; 22/12/2013

               ; AX = Physical block number
               call dsk_rd ; read in block, BX points to 1st word of data in
               ; buffer
               ;jc short read_i_3
               ;jc short_read_i_retn
               jc short loc_unix_bl_error ; 22/12/2013

readi_sioreg:
               mov si, word ptr [b_off] ; R2
               mov cx, si ; cx = R3, si = R2
               or cx, 0FE00h ; set bits 9...15 of file offset in R3
               and si, 1FFh ; calculate file offset mod 512
               add si, bx ; offset Buffer ; si now points to 1st byte in buffer
               ; where data is to be placed
               mov di, word ptr [b_base] ; R1
               neg cx ; 512 - file offset(mod512) in R3 (cx)
               cmp cx, word ptr [b_count]
               jna short @f ; 2f

               mov cx, word ptr [b_count]

@@:
               add word ptr [b_nread], cx ; r3 + number of bytes
               ; xmitted during write is put into
               ; u_nread
               sub word ptr [b_count], cx
               add word ptr [b_base], cx ; points to 1st of remaining
               ; data bytes
               add word ptr [b_off], cx ; new file offset = number
               ; of bytes done + old file offset

```

```

; end of readi_sioreg
; DI = file (user data) offset
; SI = sector (I/O) buffer offset
; CX = byte count

rep movsb
;pop ax

cmp word ptr [b_count], 0
ja read_i_1

read_i_retn: ;; 22/12/2013
;retn

;;read_i_3:
;; pop ax ; i-number

;;read_i_retn:
;; retn

;;; jc short load_bf_retn

mov cx, word ptr [b_nread]
mov bx, inode_size

;; cmp cx, word ptr [bx]
;; retn

;;load_bf_stc:
;; stc

;;load_bf_retn:
;; retn

;;; jc short loc_unix_bl_error

loc_launch_bootfile:
mov si, offset msg_CRLF
call print_string

mov ax, boot_file_segment ; 7E0h
mov ds, ax
mov es, ax
cli
mov ss, ax
;mov sp, 0FFFEh
sti

mov dl, byte ptr [bsDriveNumber]

; MASM.EXE don't accept
; jmp 07E0h:0000h
; for OP Code: EA0000E007
db 0EAh
dw 0
dw 07E0h

NeverComeHere: jmp short NeverComeHere

loc_no_bootable_disk:
mov si, offset msg_press_any_key
call print_string
xor ax, ax
int 16h
int 19h

loc_unix_bl_error:
mov si, offset unix_bfl_error_msg
call print_string
jmp short NeverComeHere

unixbootsector endp

```

```

dsk_rd  proc near
    ; 22/12/2013
    ; 28/10/2012 (bf_buff_s)
    ; 20/10/2012
    ; 14/10/2012
    ; fd boot sector version of "dskrd" procedure
    ; Derived from (original) UNIX v1 source code
    ; PRELIMINARY release of Unix Implementation Document,
    ; 20/6/1972
    ; RETRO UNIX v1 FS
    ; floppy disk boot sector version
    ; return => if cf=1 error number in [Error]

    ; ax = sector/block number

    ;cmp ax, word ptr [bf_buff_s] ; buffer sector
    ;je short dsk_rd_3

    ;mov si, ax

    mov bx, BF_BUFFER ; offset Buffer

    xor ch, ch
    mov cl, 4 ; Retry count
dsk_rd_1:
    push cx
    mov dx, 18          ; Sectors per track, 18
    div dl
    mov cl, ah          ; Sector (zero based)
    inc cl              ; To make it 1 based
    shr al, 1           ; Convert Track to Cylinder
    adc dh, 0           ; Heads (0 or 1)

    mov dl, byte ptr [bsDriveNumber] ; Physical drive number
    mov ch, al

    mov ah, 2           ; 2=read
    mov al, 01h
    int 13h             ; BIOS Service func ( ah ) = 2
                        ; Read disk sectors
                        ; BIOS Service func ( ah ) = 3
                        ; Write disk sectors
                        ; ↑AL-sec num CH-cyl CL-sec
                        ; DH-head DL-drive ES:BX-buffer
                        ; ↓CF-flag AH-stat AL-sec read
    pop cx
    jnc short dsk_rd_2
    loop dsk_rd_1
dsk_rd_2:
    ;mov word ptr [bf_buff_s], si
dsk_rd_3:
    retn

dsk_rd  endp

print_string  proc near
    mov     BX, 07
    mov     AH, 0Eh
loc_print:
    lodsb                    ; Load byte at DS:SI to AL
    and     AL,AL
    je      short loc_return ; If AL = 00h then return

    int     10h              ; BIOS Service func ( ah ) = 0Eh
                        ; Write char as TTY
                        ; ↑AL-char BH-page BL-color
    jmp     short loc_print
loc_return:
    retn

print_string  endp

unix_bfl_error_msg:
    db 07h, "UNIX boot error!"

```

```
msg_CRLF:          db 0Dh, 0Ah, 0

msg_press_any_key:
                  db 07h
                  db "Not a bootable floppy disk!"
                  db 0Dh,0Ah

b_base: dw 0
b_off: dw 0
b_count: dw 0
b_nread: dw 0

;bf_buff_s: dw 0

;;i_i:            db 2 dup (0)

                  org 7DFEh

bsBootSign:       dw 0AA55h

RUFS_INSTL        ends

                  end  INSTALL
```