

```

; *****
; INIT.ASM - process control initialization (Retro Unix 8086 v1 - /etc/init)
; -----
;
; 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
;
; [ Last Modification: 17/01/2014 ]
;
; 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> (Section E.12)
;
; *****

; Derived from 'init.s' file of original UNIX v1

; INIT09.ASM, 17/01/2014
; INIT08.ASM, 24/10/2013, 04/11/2013, 7/12/2013, 10/12/2013

.8086

; UNIX v1 system calls
_rele equ 0
_exit equ 1
_fork equ 2
_read equ 3
_write equ 4
_open equ 5
_close equ 6
_wait equ 7
_creat equ 8
_link equ 9
_unlink equ 10
_exec equ 11
_chdir equ 12
_time equ 13
_mkdir equ 14
_chmod equ 15
_chown equ 16
_break equ 17
_stat equ 18
_seek equ 19
_tell equ 20
_mount equ 21
_umount equ 22
_setuidequ 23
_getuidequ 24
_stime equ 25
_quit equ 26
_intr equ 27
_fstat equ 28
_emt equ 29
_mdate equ 30
_stty equ 31
_gtty equ 32
_ilgins equ 33

;;;
ESCKey equ 1Bh
EnterKey equ 0Dh

sys macro syscallnumber, arg1, arg2, arg3
; Retro UNIX 8086 v1 system call.
ifnb <arg1>
    mov bx, arg1
endif
ifnb <arg2>
    mov cx, arg2
endif
ifnb <arg3>
    mov dx, arg3
endif
mov ax, syscallnumber
int 20h
endm

```

```

; Retro UNIX 8086 v1 system call format:
; sys syscall (ax) <arg1 (bx)>, <arg2 (cx)>, <arg3 (dx)>

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

START_CODE:
        sys _intr, 0 ; disable time-out function
        sys _quit, 0 ; disable quit (ctrl+brk) signal
        ;
        sys _open, cttty, 0 ; open tty0
        jc error
        sys _open, cttty, 1 ; for read and write
        jc error
        ;
        sys _write, 1, msg_te, sizeof_mte
        jc error
@@:
        sys _read, 0, tchar, 1
        jc error

        ;sys _close, 0 ; close input file/tty
        jc error
        ;sys _close, 1 ; close output file/tty
        jc error

        mov al, byte ptr [tchar]

        cmp al, ENTERKey
        je short multiuser

        cmp al, ESCKey
        jne short @b

singleuser:
help:
        sys _close, 0 ; close input file/tty
        jc error
        sys _close, 1 ; close output file/tty
        jc error
        ;
        sys _open, cttty, 0 ; open control tty
        jc error
        sys _open, cttty, 1 ; for read and write
        jc error
        ;
        sys _exec, shell, shellp
        ;
        jmp short singleuser

multiuser:
        sys _close, 0 ; close input file/tty
        jc error
        sys _close, 1 ; close output file/tty
        jc error
        ;
        sys _mount, fdl, usr ; root directory on mounted fdl
                                ; disk is /usr
        sys _creat, utmp, 14 ; truncate /tmp/utmp
        jc error
        sys _close, ax ; close it
        mov byte ptr [zero]+8, 0 ; put identifier
                                ; in output buffer
        call wtmprec ; go to write acting info
        jc error

        mov si, offset itab ; address of table to SI

; create shell processes
@@:
        lodsw ; 'x', x=0, 1... to AX
        and ax, ax
        ;jz short pwait ; branch if table end
        jz short @f
        mov byte ptr [ttyx]+8, al ; put symbol in ttyx
        mov di, si
        call dfork ; go to make new init for this ttyx
        stosw ; save child id in word offer

```

```

                                ; '0', '1',...etc.
mov si, di
jmp short @b                    ; set up next child
@@:
;;
;; 10/12/2013
;; 'Enable Multi Tasking' (Time-Out)
;; system call (Retro UNIX 8086 v1 feature only !)
sys _emt, 1
;;

; wait for process to die
pwait:
    ;sys _write, 1, beep, 1 ; 10/12/2013
    ;
    sys _wait                    ; wait for user to terminate process
    mov si, offset itab ; initialize for search
    mov dx, ax

; search for process id
@@:
    lodsw                        ; bump SI to child id location
    or ax, ax
    jz short pwait              ; ? something silly

    lodsw
    cmp dx, ax                  ; which process has terminated
    jne short @b                ; not this one

; take name out of utmp
    sub si, 4                    ; process is found, point x to 'x'
                                ; for it
    ;push si                    ; save address on stack
    mov dx, word ptr [SI] ; move 'x' to DX
    sub dx, '0'                 ; remove zone bits from character
    shl dx, 1                   ; generate proper
    shl dx, 1                   ; offset
    shl dx, 1                   ; for
    shl dx, 1                   ; seek
    mov di, offset zero
    xor ax, ax ; 0 ; clear
    mov cx, 8                   ; output buffer
    rep stosw
    sys _open, utmp, 1 ; open file for writing
    jc short @f                 ; if can't open, create user anyway
    mov di, ax                  ; save file desc
    sys _seek, ax, dx, 0 ; move to proper
                                ; pointer position
    sys _write, di, zero, 16 ; zero this position in
    sys _close, di ; close file

; re-create user process
@@:
    ;pop si                     ; restore 'x' to SI
    lodsw                       ; move it to AX
    mov di, si
    mov byte ptr [ttyx]+8, al ; get correct ttyx
    mov byte ptr [zero]+8, al
                                ; move identifier to output buffer
    call wtmprec                 ; go to write accting into
    call dfork                   ; fork
    stosw                       ; save id of child
    jmp pwait ; go to wait for next process end

dfork:
    mov bx, offset @f ; return address for new process
    sys _fork
    jc short dfork ; try again
    retn

@@: ; to new copy of init
    ;sys _quit, 0 ; disable quit (ctrl+brk) signal
    ;sys _intr, 0 ; disable time-out function
    ;sys _chown, ttyx, 0
    ;sys _chmod, ttyx, 15
    ;
    xor bx, bx
    xor ch, ch
    mov cl, byte ptr [ttyx]+8

```

```

sub cl, '0'
; 17/01/2014
;mov dx, 0FF00h
mov dh, 0FFh ; do not set cursor position
; do not set serial port parameters
;
sys _stty
jc short terminate
;
sys _open, ttyx, 0 ; open this ttyx for reading
; and wait until someone calls
;jc help ; branch if trouble
jc short terminate
sys _open, ttyx, 1 ; open this ttyx for writing
; after user call
;jc help ; branch if trouble
jc short terminate
; 07/12/2013
; set console tty for current process

;
sys _exec, getty, gettyp ; getty types <login> and
; executes login which logs user
; in and executes sh-

terminate:
sys _exit ; HELP!

;help1:
jmp help

wtmprec:
sys _time ; get time
mov word ptr [zero]+10, ax ; more to output
mov word ptr [zero]+12, dx ; buffer

sys _open, wtmp, 1 ; open accounting file
jc short @f
mov si, ax ; save file descriptor

sys _seek, ax, 0, 2 ; move pointer to end of file
;push si ; save file descriptor
;jc short @f

sys _write, si, zero, 16 ; write accting info
;pop bx ; restore file descriptor
;jc short @f

sys _close, si ; close file

@@:
retn

here:
hlt
jmp short here

error:
mov si, offset msg_err
call print_msg
jmp short @b

print_msg:
mov ah, 0Eh
mov bl, 7
mov bh, byte ptr [ttyx]+8
sub bh, '0'

@@:
lodsb ; Load byte at DS:SI to AL
and al, al
jz short @f

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

jmp short @b

@@:
retn

```

```

EVEN
tchar:  db 0

EVEN
ctty:   db "/dev/tty", 0
EVEN
shell:  db "/bin/sh", 0
shellm: db "-", 0

;EVEN
usr:    db "/usr",0
EVEN
fd1:    db "/dev/fd1", 0

EVEN
utmp:   db "/tmp/utmp", 0
wtmp:   db "/tmp/wtmp", 0
ttyx:   db "/dev/ttyx", 0
getty:  db "/etc/getty",0

EVEN
shellp: dw shellm
        dw 0
gettyp: dw getty
        dw 0
itab:
        db '0',0, 0,0
        db '1',0, 0,0
        db '2',0, 0,0
        db '3',0, 0,0
        db '4',0, 0,0
        db '5',0, 0,0
        db '6',0, 0,0
        db '7',0, 0,0
        ; serial ports (COM1, COM2)
        db '8',0, 0,0
        db '9',0, 0,0
        dw 0
zero:
        db 8 dup(0)
        db 6 dup(0)
        db 2 dup(0)

msg_te:
        db 0Dh, 0Ah
        db 'Type ENTER to start in multi user mode', 0Dh, 0Ah
        db 'or type ESC to start in single user mode.'
        db 0Dh, 0Ah
sizeof_mte equ $ - offset msg_te
        db 0

msg_err:
;beep: db 07h ; 10/12/2013
        db 0Dh, 0Ah
        db 'Error ! '
nextline:
        db 0Dh, 0Ah, 0

UNIX    ends

; / init -- process control initialization
;
;      sys      intr; 0
;      sys      quit; 0
;      sys      38. / get console switches
;      cmp      r0,$173030
;      bne      lf
;help:
;      clr      r0
;      sys      close
;      mov      $1,r0
;      sys      close
;      sys      open; ctty; 0
;      sys      open; ctty; 1
;      sys      exec; shell; shellp
;      br       help
;l:
;      sys      mount; rk1; usr

```

```

;      sys      mount; rk2; ssys
;      sys      mount; rk3; crp
;      mov      '$0,r1
;1:
;      movb     r1,tapx+8
;      sys      chmod; tapx; 17
;      inc      r1
;      cmp      r1,$'8
;      blo      1b
;      sys      creat; utmp; 16
;      sys      close
;      sys      unlink; dpdlock
;      sys      fork
;      br       daemon
;      sys      fork
;      br       dirass
;      sys      fork
;      br       dds
;      movb     '$x,zero+8.
;      jsr      pc,wtmprec
;      mov      $itab,r1
;      br       1f
;
;daemon:
;      sys      exec; etcdpd; etcdpdp
;      sys      exit
;
;dirass:
;      sys      chdir; usrmel
;      sys      exec; melda; meldap
;      sys      exit
;
;dds:
;      sys      exec; usrdd; usrddp
;      sys      exit
;
; // create shell processes
;
;1:
;      mov      (r1)+,r0
;      beq      pwait
;      movb     r0,ttys+8
;      jsr      pc,dfork
;      mov      r0,(r1)+
;      br       1b
;
; // wait for process to die
;
;pwait:
;      sys      wait
;      mov      $itab,r1
;
; // search for process id
;
;2:
;      tst      (r1)+
;      beq      pwait
;      cmp      r0,(r1)+
;      bne      2b
;
; // take name out of utmp
;
;      sub      $4,r1
;      mov      r1,-(sp)
;      mov      (r1),r1
;      sub      '$0,r1
;      cmp      r1,$'a-'0
;      blo      2f
;      sub      '$a-'0-10.,r1 / map a-z into 10. on
;2:
;      asl      r1
;      asl      r1
;      asl      r1
;      asl      r1
;      mov      r1,0f
;      mov      $zero,r1
;2:
;      clr      (r1)+
;      cmp      r1,$zero+16.

```

```

;     blo      2b
;     sys      open; utmp; 1
;     bes      2f
;     mov      r0,r1
;     sys      seek; 0:..; 0
;     mov      r1,r0
;     sys      write; zero; 16.
;     mov      r1,r0
;     sys      close
;
; // re-create user process
;
;2:
;     mov      (sp)+,r1
;     mov      (r1)+,r0
;     movb     r0,TTYX+8
;     movb     r0,zero+8.
;     jsr      pc,wtmprec
;     jsr      pc,dfork
;     mov      r0,(r1)+
;     br       pwait
;
;dfork:
;     sys      fork
;     br       lf
;     bes      dfork
;     rts      pc
;1:
;     sys      quit; 0
;     sys      intr; 0
;     sys      chown; TTYX; 0
;     sys      chmod; TTYX; 15
;     sys      open; TTYX; 0
;     bes      help1
;     sys      open; TTYX; 1
;     bes      help1
;     sys      exec; getty; gettyp
;     sys      exit                / HELP!
;
;help1:
;     jmp      help
;
;wtmprec:
;     mov      r1,-(sp)
;     sys      time
;     mov      r0,zero+10.
;     mov      r1,zero+12.
;     sys      open; wtmp; 1
;     bes      2f
;     mov      r0,r2
;     sys      seek; 0; 2
;     mov      r2,r0
;     sys      write; zero; 16.
;     mov      r2,r0
;     sys      close
;2:
;     mov      (sp)+,r1
;     rts      pc
;
;etcdpdp:
;     etcdpd; 0
;meldap:
;     melda; 0
;usrddp:
;     usrdd; 0
;usrdd: </usr/demo/dds\0>
;melda: </usr/mel/da\0>
;usrmel:</usr/mel\0>
;rk1: </dev/rk1\0>
;rk2: </dev/rk2\0>
;rk3: </dev/rk3\0>
;usr: </usr\0>
;ssys: </sys\0>
;crp: </crp\0>
;ctty: </dev/tty\0>
;shell:</bin/sh\0>
;shellm:<-\0>
;dpdlock:
;     </usr/dpd/lock\0>

```

```
;etcdpd:
;    </etc/dpd\0>
;tapx: </dev/tapx\0>
;utmp: </tmp/utmp\0>
;wtmp: </tmp/wtmp\0>
;ttyx: </dev/ttyx\0>
;getty:</etc/getty\0>
;    .even
;
;shellp:shellm
;    0
;gettyp:getty
;    0
;itab:
;    '0; ..
;    '1; ..
;    '2; ..
;    '3; ..
;    '4; ..
;    '5; ..
;    '6; ..
;    '7; ..
;    '8; ..
;    'a; ..
;    'b; ..
;    0
;
;    .bss
;offset: .+.2
;zero:  .+.8.; .+.6; .+.2.

end  START_CODE
```