

```
; *****
; LOGIN.ASM (Retro Unix 8086 v1 - /bin/login)
; -----
;
; 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/login file
;
; [ Last Modification: 27/06/2014 ]
;
; Derivation from UNIX Operating System (v1.0 for PDP-11)
; (Original) Source Code by Ken Thompson (Bell Laboratories, 1971-1972)
;
; *****

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

; LOGIN03.ASM, 27/06/2014
; LOGIN02.ASM, 07/11/2013 .. 06/12/2013

.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
_setuideo equ 23
_getuideo equ 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

;;;

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:
; from 'sysexec' system calls
; cs=ds=es=ss
; ax=bx=cx=dx=si=di=bp=0
; (stack pointer -sp- points to
; to the head of arguments list which is
; on top the stack, backward from 'ecore'.)
; sp = offset argc (argument count)
;

sys      _quit, 0
sys      _intr, 0

call     ttyn
; ah = 0
mov      byte ptr [ttyx]+8, al
cmp      al, 'x'
je       short @f
sub      al, '0'
jz       short @f
shl      ax, 1
shl      ax, 1
shl      ax, 1
shl      ax, 1
mov      word ptr [s_off], ax

@@:
pop      dx ; argument count
pop      ax ; pointer to argument 0
           ; executable file name
cmp      dx, 1
jna      short login
dec      dx ; dec dl
pop      si ; pointer to argument 1
           ; user name
mov      di, offset uname
mov      bx, di
add      bx, 8

@@:
lodsb
stosb
and      al, al
jz       short @f
cmp      di, bx
jb       short @b

@@:
dec      dx
jz       short login
pop      si
mov      di, offset passwd

@@:
lodsb
stosb
or       al, al
jz       short login
cmp      di, offset passwd + 8
jb       short @b

login:
mov      byte ptr [BX], 0 ; uname + 8
mov      ax, offset passwdf
call     fopen
jnc      short lg0
mov      si, offset msgNoPswdf
call     msg
sys      _exit

lg0:
call     guname

lg1:
mov      si, offset uname
call     compar
je       short lg3 ; zf = 1 --> match
```

```

lg2:
    ;mov    bx, offset pbuf
    call   getc
    jc     short sorry
    cmp    al, 0Dh ; \n
    jne    short lg2
    call   getc
    ;jc     short sorry
    ;cmp    al, 0Ah
    ;jne    short sorry
    jmp    short lg1

lg3:
    call   getc
    jc     short sorry
    cmp    al, ':'
    je     short lg4
    push   ax
    call   gpasswd
    ;mov    si, offset _word
    pop    ax
    mov    ah, byte ptr [SI]
    cmp    al, ah
    jne    short sorry
    inc    si
    ; SI = offset _word + 1
    call   compar
    jne    short sorry

lg4:
    ; get UID
    xor    cx, cx ; 0

lg5:
    push   cx
    call   getc
    cmp    al, ':'
    je     short lg6
    mov    cl, al
    sub    cl, '0'
    xor    ch, ch
    pop    dx
    mov    ax, 10
    mul    dx
    add    cx, ax
    jmp    short lg5

lg6:
    pop    cx ; UID
    sys    _chown, ttyx ; cx = arg 2
    mov    word ptr [uid], cx

lg7:
    call   getc
    cmp    al, ':'
    jne    short lg7 ; / skip ident field
    mov    di, offset dirbuf

lg8:
    call   getc
    cmp    al, ':'
    je     short lg9
    stosb
    jmp    short lg8

lg9:
    xor    al, al
    stosb
    sys    _chdir, dirbuf
    jnc    short lg10
    mov    si, offset msgNoDir
    call   msg
    ;jmp    short sorry

sorry:
    mov    si, offset msgIL
    call   msg
    xor    ax, ax
    mov    word ptr [uname], ax
    mov    word ptr [passwd], ax
    jmp    login

lg10:
    mov    bx, offset uname + 7

```

```

lg11:
    cmp     byte ptr [BX], 0
    ja      short lg12
    mov     byte ptr [bx], 20h
    dec     bx
    jmp     short lg11

lg12:
    mov     si, offset ttyx + 8
    cmp     byte ptr [SI], 'x'
    je      short lg14
    sys     _open, utmp, 1
    jc      short lg13
    mov     di, ax
    mov     ax, word ptr [s_off]
    sys     _seek, di, ax, 0
    mov     al, byte ptr [SI]
    mov     byte ptr [uname]+8, al
    sys     _time
    mov     word ptr [uname]+10, ax
    mov     word ptr [uname]+12, dx
    sys     _write, di, uname, 16
    ;mov     bx, di
    ;sys     _close
    sys     _close, di

lg13:
    ;cmp     byte ptr [SI], 'x'
    ;je      short lg14
    sys     _open, wtmp, 1
    jc      short lg14
    mov     di, ax
    sys     _seek, di, 0, 2
    sys     _write, di, uname, 16
    ;mov     bx, di
    ;sys     _close
    sys     _close, di

lg14:
    call    getc
    cmp     al, 0Dh ; \n
    je      short lg16
    mov     di, offset shell

lg15:
    mov     al, ah
    stosb
    call    getc
    cmp     al, 0Dh ; \n
    jne     short lg15
    xor     al, al ; 0
    stosb

lg16:
    mov     bx, word ptr [pbuf]
    sys     _close
    mov     ax, offset motd
    call    fopen
    jc      short lg18

lg17:
    call    getc
    jc      short lg18
    mov     byte ptr [uname], al
    sys     _write, 1, uname, 1
    jmp     short lg17

lg18:
    mov     bx, word ptr [pbuf]
    sys     _close
    sys     _stat, mailf, pbuf
    jc      short lg19
    mov     al, byte ptr [pbuf]+6 ; file size
    and     al, al
    jna     short lg19
    mov     si, offset msgMail
    call    mesg

lg19:
    mov     bx, word ptr [uid]
    sys     _setuid
    sys     _exec, shell, shellp
    mov     si, offset msgNoSh
    call    mesg
    sys     _exit

```

```
gpasswd:
    mov     di, offset passwd
    cmp     byte ptr [DI], 1
    jnb     short gp2
    mov     si, offset msgPswd
    call    mesg

gp1:
    call    tgetc
    cmp     al, 08h
    je      short gp3
    cmp     al, 127
    je      short gp3
    stosb
    and     al, al
    jz      short gp2
    mov     byte ptr [chr], '*'
    call    tputc
    cmp     di, offset passwd + 9
    jb      short gp1
    dec     di
    jmp     short gp1

gp2:
    mov     si, offset passwd
    call    crypt
    ;mov     si, offset _word
    retn

gp3:
    ; Backspace
    ; (Retro UNIX 8086 v1 modification)
    cmp     di, offset passwd
    jna     short gp1
    ;mov     byte ptr [chr], 08h
    call    tputbs
    jmp     short gp1

guname:
    mov     di, offset uname
    cmp     byte ptr [DI], 1
    jnb     short gun2
    xor     ax, ax ; mov ax, 0
    stosw
    stosw
    stosw
    stosw
    mov     si, offset msgName
    call    mesg
    mov     di, offset uname

gun1:
    call    tgetc
    cmp     al, 08h
    je      short gun3
    cmp     al, 127
    je      short gun3
    stosb
    and     al, al
    jz      short gun2
    call    tputc
    cmp     di, offset uname + 9
    jb      short gun1
    dec     di
    jmp     short gun1

gun2:
    retn

gun3:
    ; Backspace
    ; (Retro UNIX 8086 v1 modification)
    cmp     di, offset uname
    jna     short gun1
    ;mov     byte ptr [chr], 08h
    call    tputbs
    jmp     short gun1
```

```

compar:
    ; SI = uname or _word
    ; (encrypted passwd)
;mov    bx, offset pbuf

cmp_0:
    call    getc
    jnc     short cmp_1
    pop     ax
    jmp     sorry

cmp_1:
    mov     ah, al
    ; AH = character
    lodsb
    cmp     al, ah
    je      short cmp_0
    and     al, al
    jnz     short cmp_2
    cmp     ah, ':'

cmp_2:
    ;ZF = 1 --> match
    retn

tgetc:
    sys     _read, 0, chr, 1
    and     ax, ax
    jnz     short @f
    sys     _exit

@@:
    mov     al, byte ptr [chr]
    cmp     al, 0Dh
    jne     short @f
    xor     al, al

@@:
    retn

tputbs:
    mov     byte ptr [chr], 08h
    dec     di

tputc: ; 27/06/2014
    sys     _write, 1, chr, 1
    retn

mesg:
    mov     dx, si

@@:
    lodsb
    and     al, al
    jnz     short @b
    sub     si, dx
    xchg    si, dx
    ; dx = string length
    sys     _write, 1, si
    retn

; // return name of current tty

ttyn:
    push    di
    push    si
    push    dx
    mov     byte ptr [ttynname], 'x'
    sys     _fstat, 1, buf ; get tty file status
                        ; file descriptor = 1
                        ; (standard output)

    jc      short erl
    sys     _open, dev, 0
    jc      short erl
    ;
    mov     si, word ptr [buf]
    mov     di, ax

@@:
    sys     _read, di, buf, 10
    jc      short er
    cmp     ax, 10
    jne     short er
    mov     dx, word ptr [buf]
    cmp     dx, si
    jne     short @b
    mov     dx, word ptr [buf]+2

```

```

        cmp     dx, 'tt'
        jne     short er
        mov     dx, word ptr [buf]+4
        cmp     dl, 'y'
        jne     short er
        ;or     dh, dh
        ;jz     short er
        cmp     dh, '0'
        jb     short er
        cmp     dh, '9'
        ja     short er
        cmp     byte ptr [buf]+6, 0
        jne     short er
        mov     byte ptr [ttyname], dh
er:
        sys     _close, di
er1:
        mov     al, byte ptr [ttyname]
        xor     ah, ah
        pop     dx
        pop     si
        pop     di
        retn

; open a file for use by get(c|w)
;
fopen:
        ; ax = file name offset
        mov     di, offset pbuf
        sys     _open, ax, 0
        jc     short @f
        stosw
        xor     ax, ax ; 0
        stosw
        retn
@@:
        mov     ax, 0FFFFh
        stosw
        retn

; get characters from input file
;
getc:
        push    si
        mov     si, offset pbuf
        mov     ax, word ptr [SI]+2 ; char count
        and     ax, ax
        jnz     short gch1
gch0:
        mov     bx, word ptr [SI]
        mov     cx, offset pbuf + 6 ; read buff. addr.
        mov     word ptr [SI]+4, cx ; char offset
        ;xor     ax, ax
        ;mov     word ptr [SI]+2, ax ; 0
        mov     dx, 512
        sys     _read ; sys _read, bx, cx, dx
        jc     short gch2
        or      ax, ax
        jz     short gch3
gch1:
        dec     ax
        mov     word ptr [SI]+2, ax
        mov     bx, word ptr [SI]+4
        mov     al, byte ptr [BX]
        inc     bx
        mov     word ptr [SI]+4, bx
        xor     ah, ah
        pop     si
        retn
gch2:
        xor     ax, ax
gch3:
        pop     si
        stc
        retn

```

```

; crypt -- password incoding
;
;; Original Unix v5 (PDP-11) 'crypt'
;; code has been converted to
;; Retro UNIX 8086 v1 'crypt'
;; procedure in 'login.asm'
;; (by Erdogan Tan - 12/11/2013).
;
;
;crypt:
;    mov     r1,-(sp)
;    mov     r2,-(sp)
;    mov     r3,-(sp)
;    mov     r4,-(sp)
;    mov     r5,-(sp)
;
;    mov     r0,r1
;    mov     $key,r0
;    movb    $004,(r0)+
;    movb    $034,(r0)+

crypt:
;mov     si, offset passwd
;mov     di, offset key
;mov     al, 4
;stosb
;mov     al, 28
;stosb

;1:
;    cmp     r0,$key+64.
;    bhis    lf
;    movb    (r1)+,(r0)+
;    bne     lb
;1:
;    dec     r0

cryp0:
;    lodsb
;    stosb
;    and     al, al
;    jz      short cryp1
;    cmp     di, offset key + 64
;    jb      short cryp0

cryp1:
;    dec     di

;
;
; fill out key space with clever junk
;
;    mov     $key,r1
;1:
;    movb    -1(r0),r2
;    movb    (r1)+,r3
;    xor     r3,r2
;    movb    r2,(r0)+
;    cmp     r0,$key+128.
;    blo     lb

;
; fill out key space with clever junk

;    mov     si, offset key
cryp2:
;    mov     bl, byte ptr [DI]-1
;    lodsb
;    xor     al, bl
;    stosb
;    cmp     di, offset key + 128
;    jb      short cryp2
;
;
;
; establish wheel codes and cage codes
;
;    mov     $wheelcode,r4
;    mov     $cagecode,r5
;    mov     $256.,-(sp)

```



```

;2:
;   clr    r2
;   clr    (r4)
;   mov    $wheeldiv,r3
;3:
;   clr    r0
;   mov    (sp),r1
;   div    (r3)+,r0
;   add    r1,r2
;   bic    $40,r2
;   bis    shift(r2),(r4)
;   cmp    r3,$wheeldiv+6.
;   bhis   4f
;   bis    shift+4(r2),(r5)
;4:
;   cmp    r3,$wheeldiv+10.
;   blo    3b
;   sub    $2,(sp)
;   tst    (r4)+
;   tst    (r5)+
;   cmp    r4,$wheelcode+256.
;   blo    2b
;   tst    (sp)+
;
;
;//      establish wheel codes and cage codes

      mov    si, offset wheelcode
      mov    di, offset cagecode
      mov    ax, 256
      push   ax ; *
      mov    bp, sp
cryp3:
      sub    dx, dx ; 0
      mov    word ptr [SI], dx ; 0
      mov    bx, offset wheeldiv
cryp4:
      mov    ax, word ptr [BP]
      mov    cl, byte ptr [BX]
      div    cl
      add    dl, ah
      inc    bx
      and    dl, 01Fh
      push   bx
      mov    bx, offset shift
      add    bx, dx
      mov    ax, word ptr [BX]
      or     word ptr [SI], ax
      pop    cx
      cmp    cx, offset wheeldiv + 3
      jnb    short cryp5
      add    bx, 4
      mov    ax, word ptr [BX]
      or     word ptr [DI], ax
cryp5:
      mov    bx, cx
      cmp    bx, offset wheeldiv + 5
      jb     short cryp4
      sub    word ptr [BP], 2
      lodsw
      inc    di
      inc    di
      cmp    si, offset wheelcode + 256
      jb     short cryp3
      pop    ax ; *

;   .data
;shift: 1;2;4;10;20;40;100;200;400;1000;2000;4000;10000;20000;40000;100000
;       1;2
;wheeldiv: 32.; 18.; 10.; 6.; 4.
;       .bss
;cagecode: .+.256.
;wheelcode: .+.256.
;       .text
;
;//
;//      make the internal settings of the machine
;//      both the lugs on the 128 cage bars and the lugs
;//      on the 16 wheels are set from the expanded key

```

```

//
;      mov     $key,r0
;      mov     $cage,r2
;      mov     $wheel,r3
;1:
;      movb    (r0)+,r1
;      bic     $!177,r1
;      asl     r1
;      mov     cagecode(r1),(r2)+
;      mov     wheelcode(r1),(r3)+
;      cmp     r0,$key+128.
;      blo     1b

//      make the internal settings of the machine
//      both the lugs on the 128 cage bars and the lugs
//      on the 16 wheels are set from the expanded key
cryp6:
      mov     bx, offset key
      mov     si, offset cage
      mov     di, offset wheel
cryp7:
      mov     cl, byte ptr [BX]
      inc     bx
      and     cx, 7Fh
      shl     cl, 1
      xchg    cx, bx
      mov     ax, word ptr [BX + cagecode]
      mov     word ptr [SI], ax
      inc     si
      inc     si
      mov     ax, word ptr [BX + wheelcode]
      stosw
      mov     bx, cx
      cmp     bx, offset key + 128
      jb      short cryp7

//
//
//      now spin the cage against the wheel to produce output.
//
;      mov     $word,r4
;      mov     $wheel+128.,r3
;3:
;      mov     -(r3),r2
;      mov     $cage,r0
;      clr     r5
;1:
;      bit     r2,(r0)+
;      beq     2f
;      incb    r5
;2:
;      cmp     r0,$cage+256.
;      blo     1b

//
//      now spin the cage against the wheel to produce output.
//
cryp8:
      mov     di, offset _word
      mov     bx, offset wheel + 128
cryp9:
      dec     bx
      dec     bx
      mov     dx, word ptr [BX]
      mov     si, offset cage
      sub     cx, cx ; 0
cryp10:
      lodsw
      test    ax, dx
      jz      short cryp11
      inc     cl
cryp11:
      cmp     si, offset cage + 256
      jb      short cryp10

//
//      we have a piece of output from current wheel
//      it needs to be folded to remove lingering hopes of
//      inverting the function

```

```

;
;      mov     r4,-(sp)
;      clr     r4
;      div     $26.+26.+10.,r4
;      add     $'0,r5
;      cmp     r5,$'9
;      blos    1f
;      add     $'A-'9-1,r5
;      cmp     r5,$'Z
;      blos    1f
;      add     $'a-'Z-1,r5
;1:
;      mov     (sp)+,r4
;      movb    r5,(r4)+
;      cmp     r4,$word+8.
;      blo     3b
;
;
;      mov     (sp)+,r5
;      mov     (sp)+,r4
;      mov     (sp)+,r3
;      mov     (sp)+,r2
;      mov     (sp)+,r1
;      mov     $word,r0
;      rts     pc
;      .bss
;key:    .+.128.
;word:   .+.32.
;cake:   .+.256.
;wheel:  .+.256.

;
;      we have a piece of output from current wheel
;      it needs to be folded to remove lingering hopes of
;      inverting the function
;
      mov     ax, cx
      mov     dl, 26+26+10
      div     dl
      mov     al, ah
      add     al, '0'
      cmp     al, '9'
      jna     short cryp12
      add     al, 'A'-'9'-1
      cmp     al, 'Z'
      jna     short cryp12
      add     al, 'a'-'Z'-1
cryp12:
      stosb
      cmp     di, offset _word + 8
      jnb     short cryp9
      mov     si, offset _word
      retn

EVEN
shellp:
      dw mshell
      dw 0
utmp:   db '/tmp/utmp'
      db 0
wtmp:   db '/tmp/wtmp'
      db 0
shell:  db '/bin/sh'
      db 0
shp1    equ offset shell + 32 - offset shpad
shpad:  db shp1 dup (0)

mshell: db '-'
      db 0
motd:   db '/etc/motd'
      db 0
mailf:  db 'mailbox'
      db 0
EVEN
passwdf: db '/etc/passwd'
      db 0
ttyx:   db '/dev/tty' ; db '/dev/ttyx'
      db 0

```

```

EVEN
uname: db 16 dup(0) ; db 16 dup (0)
      dw 0 ; db 0
passwd: db 8 dup(0)
      dw 0 ; db 0
dirbuf: db 32 dup(0)
;shbuf: db 32 dup(0)
;ttyb:  db 6 dup(0)
uid:    dw 0
chr:    dw 0

;; ttyn data
;EVEN
dev:    db '/dev', 0
EVEN
buf:    db 34 dup(0)
;EVEN
ttyname:dw 0

s_off:  dw 0
;
msgName: db 0Dh, 0Ah, 'Name: ', 0
EVEN
msgPswd: db 0Dh, 0Ah, 'Password: ', 0
EVEN
msgIL:   db 0Dh, 0Ah, 'Login incorrect !', 0
;EVEN
msgNoSh: db 0Dh, 0Ah, 'No Shell !'
nextline: db 0Dh, 0Ah, 0
EVEN
msgNoPswdf:
      db 0Dh, 0Ah, "Can't open password file !"
      db 0Dh, 0Ah, 0
EVEN
msgNoDir:
      db 0Dh, 0Ah, 'No directory !'
      db 0Dh, 0Ah, 0
EVEN
msgMail:
      db 0Dh, 0Ah, 'You have mail.'
      db 0Dh, 0Ah, 0
EVEN
key:    db 128 dup(0)
_word:  db 10 dup(0) ; db 32 dup(0)
cage:   db 256 dup(0)
wheel:  db 256 dup(0)
shift:  dw 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768
      dw 1, 2
wheeldiv: db 32, 18, 10, 6, 4
EVEN
cagecode: dw 256 dup(0)
wheelcode: dw 256 dup(0)

;EVEN
pbuf:    db 518 dup (0)

      dw 417 ; 01A1h

UNIX     ends

;;;;;;;;;;;;;
;; login.s
;
;/ login -- enter new user
;
;.globl ttyn
;.globl crypt
;.globl fopen
;.globl getc
;.globl msg
;
;      sys      quit; 0
;      sys      intr; 0
;      jsr      pc,ttyn
;      movb     r0,ttyn+8.
;      sub      '$0,r0
;      cmp      r0,$'a-'0
;      blo      1f
;      sub      '$a-'0-10.,r0 / map a-z into 10. on

```

```

;1:
;   asl    r0
;   asl    r0
;   asl    r0
;   asl    r0
;   mov    r0,offset
;   mov    (sp)+,r5
;   tst    (sp)+
;   dec    r5
;   ble    login
;   mov    (sp)+,r4
;   mov    $uname,r1
;2:
;   movb   (r4)+,(r1)+
;   bne    2b
;   dec    r5
;   ble    login
;   mov    (sp)+,r4
;   mov    $passwd,r1
;2:
;   movb   (r4)+,(r1)+
;   bne    2b
;login:
;   clrb   uname+8.
;   mov    $passwdf,r0
;   jsr    r5,fopen; pbuf
;   bec    1f
;   jsr    r5,mesg; <Can't open password file\n\0>; .even
;   sys    exit
;1:
;   jsr    pc,guname
;1:
;   jsr    r5,compar; uname
;   br     .+4
;   br     2f
;3:
;   jsr    r5,getc; pbuf
;   bes    sorry
;   cmp    r0,$'\n
;   bne    3b
;   br     1b
;sorry:
;   jsr    r5,mesg; <Login incorrect\n\0>; .even
;   mov    pbuf,r0
;   sys    close
;   clr    uname
;   clr    passwd
;   br     login
;2:
;   jsr    r5,getc; pbuf
;   cmp    r0,$':
;   beq    2f
;   mov    r0,-(sp)
;   jsr    pc,gpasswd
;   cmpb   (r0)+,(sp)+
;   bne    sorry
;   mov    r0,0f
;   jsr    r5,compar; 0:..
;   br     sorry
;2:
;   clr    r1
;2:
;   jsr    r5,getc; pbuf
;   cmp    r0,$':
;   beq    2f
;   mpy    $10.,r1
;   sub    $'0,r0
;   add    r0,r1
;   br     2b
;2:
;   mov    r1,0f
;   sys    chown; ttyx; 0:..
;   mov    r1,uid
;1:
;   jsr    r5,getc; pbuf
;   cmp    r0,$':
;   bne    1b                / skip ident field
;   mov    $dirbuf,r1

```

```

;1:
;   jsr    r5,getc; pbuf
;   cmpb   r0,$':
;   beq    1f
;   movb   r0,(r1)+
;   br     1b
;1:
;   clrb   (r1)
;   sys    chdir; dirbuf
;   bec    1f
;   jsr    r5,mesg; <No directory\n\0>; .even
;   br     sorry
;1:
;   mov     $uname+8.,r1
;1:
;   tstb   -(r1)
;   bne    1f
;   movb   $' ,(r1)
;   br     1b
;1:
;   cmpb   ttyx+8.,$'x
;   beq    1f
;   sys    open; utmp; 1
;   bes    1f
;   mov     r0,r2
;   sys    seek; offset:..; 0
;   movb   ttyx+8.,uname+8.
;   sys    time
;   mov     r0,uname+10.
;   mov     r1,uname+12.
;   mov     r2,r0
;   sys    write; uname; 16.
;   mov     r2,r0
;   sys    close
;1:
;   cmpb   ttyx+8.,$'x
;   beq    1f
;   sys    open; wtmp; 1
;   bes    1f
;   mov     r0,r1
;   sys    seek; 0; 2
;   sys    write; uname; 16.
;   mov     r1,r0
;   sys    close
;1:
;   jsr    r5,getc; pbuf
;   cmp    r0,$'\n
;   beq    1f
;   mov     $shell,r1
;2:
;   movb   r0,(r1)+
;   jsr    r5,getc; pbuf
;   cmp    r0,$'\n
;   bne    2b
;   clrb   (r1)
;1:
;   mov     pbuf,r0
;   sys    close
;   mov     $motd,r0
;   jsr    r5,fopen; pbuf
;   bes    1f
;2:
;   jsr    r5,getc; pbuf
;   bes    1f
;   mov     r0,uname
;   mov     $1,r0
;   sys    write; uname; 1
;   br     2b
;1:
;   mov     pbuf,r0
;   sys    close
;   sys    stat; mailf; pbuf
;   bes    1f
;   tst     pbuf+6
;   beq    1f
;   jsr    r5,mesg; <You have mail\n\0>; .even
;1:
;   mov     uid,r0
;   sys    setuid

```

```

;      sys      exec; shell; shellp
;      jsr      r5,mesg; <No Shell\n\0>; .even
;      sys      exit
;
;gpasswd:
;      mov      $passwd,r1
;      tstb     (r1)
;      bne      3f
;      clr      r0
;      sys      gtty; ttyb
;      bic      $10,ttyb+4          / turn off echo
;      clr      r0
;      sys      stty; ttyb
;      jsr      r5,mesg; <Password: \0>; .even
;2:
;      jsr      pc,tgetc
;      movb     r0,(r1)+
;      beq      1f
;      cmp      r1,$passwd+9.
;      blo      2b
;      dec      r1
;      br       2b
;1:
;      bis      $10,ttyb+4          / turn on echo
;      clr      r0
;      sys      stty; ttyb
;      jsr      r5,mesg; <\n\0>; .even
;3:
;      mov      $passwd,r0
;      jsr      pc,encrypt
;      clrb     8(r0)
;      rts      pc
;
;uname:
;      mov      $uname,r1
;      tstb     (r1)
;      bne      1f
;      clr      (r1)+
;      clr      (r1)+
;      clr      (r1)+
;      clr      (r1)+
;      mov      $uname,r1
;      jsr      r5,mesg; <Name: \0>; .even
;2:
;      jsr      pc,tgetc
;      movb     r0,(r1)+
;      beq      1f
;      cmp      r1,$uname+9.
;      blo      2b
;      dec      r1
;      br       2b
;1:
;      rts      pc
;
;compar:
;      mov      (r5)+,r4
;1:
;      jsr      r5,getc; pbuf
;      bes      2f
;      cmpb     r0,(r4)+
;      beq      1b
;      cmp      r0,$':
;      bne      1f
;      tstb     -(r4)
;      bne      1f
;      tst      (r5)+
;1:
;      rts      r5
;2:
;      tst      (sp)+
;      jmp      sorry
;
;tgetc:
;      clr      r0
;      sys      read; ch; 1
;      tst      r0
;      bne      1f
;      sys      exit

```

```

;1:
;   mov     ch,r0
;   cmp     r0,$'\n
;   bne     lf
;   clr     r0
;1:
;   rts     pc
;
;shellp:
;   mshell
;   0
;utmp: </tmp/utmp\0>
;wtmp: </tmp/wtmp\0>
;shell: </bin/sh\0>; . =shell+32.
;mshell:<-\0>
;motd: </etc/motd\0>
;mailf: <mailbox\0>
;passwd:</etc/passwd\0>
;ttyx: </dev/ttyx\0>
;.even
;.bss
;uname: . =.+16.
;passwd: . =.+8.
;dirbuf: . =.+32.
;shbuf: . =.+32.
;ttyb: . =.+6
;uid: . =.+2
;ch: . =.+2
;pbuf: . =.+518.

;;;;;;;;;;;;;
;; ttyn.s
;
; // return name of current tty
;
;.globl ttyn, _ttyn
;
;_ttyn:
;   mov     2(sp),r0
;   br      lf
;ttyn:
;   clr     r0
;1:
;   mov     $'x,name
;   tst     -(sp)
;   sys     fstat; buf
;   bes     er1
;   mov     buf+2,(sp)
;   sys     open; dev; 0
;   bes     er1
;   mov     r0,r1
;1:
;   mov     r1,r0
;   sys     read; buf; 16.
;   bes     er
;   cmp     r0,$16.
;   bne     er
;   mov     $buf,r0
;   cmp     (r0)+,(sp)
;   bne     lb
;   cmp     (r0)+,$"tt
;   bne     lb
;   cmpb    (r0)+,$'y
;   bne     lb
;   tstb    (r0)+
;   beq     lb
;   cmpb    (r0),$'\0
;   bne     lb
;   movb    -(r0),name
;
;er:
;   mov     r1,r0
;   sys     close
;
;er1:
;   tst     (sp)+
;   movb    name,r0
;   rts     pc
;

```



```

; .data
; dev:  </dev\0>
; .even
; .bss
; buf:  .+.40.
; name: .+.2

;;;;;;;;;;;;;
;; get.s (unix v5)
;
; fopen -- open a file for use by get(c|w)
;
; fopen:
;     mov     r1, -(sp)
;     mov     (r5)+, r1
;     mov     r0, 0f
;     sys     0; 9f
; .data
; 9:
;     sys     open; 0:..; 0
; .text
;     bes     1f
;     mov     r0, (r1)+
;     clr     (r1)+
;     mov     (sp)+, r1
;     rts     r5
; 1:
;     mov     $-1, (r1)
;     mov     (sp)+, r1
;     sec
;     rts     r5
;

;;;;;;;;;;;;;
;; get.s (unix v5)
;
; getc -- get characters from input file
;
; getc:
;     mov     r1, -(sp)
;     mov     (r5)+, r1
;     dec     2(r1)
;     bge     1f
;     mov     r1, r0
;     add     $6, r0
;     mov     r0, 0f
;     mov     r0, 4(r1)
;     mov     (r1), r0
;     sys     0; 9f
; .data
; 9:
;     sys     read; 0:..; 512.
; .text
;     bes     2f
;     tst     r0
;     bne     3f
; 2:
;     mov     (sp)+, r1
;     sec
;     rts     r5
; 3:
;     dec     r0
;     mov     r0, 2(r1)
; 1:
;     clr     r0
;     bisb    *4(r1), r0
;     inc     4(r1)
;     mov     (sp)+, r1
;     rts     r5

;;;;;;;;;;;;;
;; crypt.s (unix v5)
;
; // crypt -- password incoding
;
; //     mov     $key, r0
; //     jsr     pc, crypt
;

```

```

; .globl crypt, word
;
; crypt:
;     mov     r1, -(sp)
;     mov     r2, -(sp)
;     mov     r3, -(sp)
;     mov     r4, -(sp)
;     mov     r5, -(sp)
;
;     mov     r0, r1
;     mov     $key, r0
;     movb    $004, (r0)+
;     movb    $034, (r0)+
;1:
;     cmp     r0, $key+64.
;     bhis    1f
;     movb    (r1)+, (r0)+
;     bne     1b
;1:
;     dec     r0
;
; //
; //     fill out key space with clever junk
; //
;     mov     $key, r1
;1:
;     movb    -1(r0), r2
;     movb    (r1)+, r3
;     xor     r3, r2
;     movb    r2, (r0)+
;     cmp     r0, $key+128.
;     blo     1b
; //
; //     establish wheel codes and cage codes
; //
;     mov     $wheelcode, r4
;     mov     $cagecode, r5
;     mov     $256., -(sp)
;2:
;     clr     r2
;     clr     (r4)
;     mov     $wheeldiv, r3
;3:
;     clr     r0
;     mov     (sp), r1
;     div     (r3)+, r0
;     add     r1, r2
;     bic     $40, r2
;     bis     shift(r2), (r4)
;     cmp     r3, $wheeldiv+6.
;     bhis    4f
;     bis     shift+4(r2), (r5)
;4:
;     cmp     r3, $wheeldiv+10.
;     blo     3b
;     sub     $2, (sp)
;     tst     (r4)+
;     tst     (r5)+
;     cmp     r4, $wheelcode+256.
;     blo     2b
;     tst     (sp)+
; //
;     .data
; shift: 1;2;4;10;20;40;100;200;400;1000;2000;4000;10000;20000;40000;100000
;     1;2
; wheeldiv: 32.; 18.; 10.; 6.; 4.
;     .bss
; cagecode: .+.256.
; wheelcode: .+.256.
;     .text
; //
; //
; //     make the internal settings of the machine
; //     both the lugs on the 128 cage bars and the lugs
; //     on the 16 wheels are set from the expanded key
; //
;     mov     $key, r0
;     mov     $cage, r2

```

```

;      mov     $wheel,r3
;1:
;      movb    (r0)+,r1
;      bic     $!177,r1
;      asl     r1
;      mov     cagecode(r1),(r2)+
;      mov     wheelcode(r1),(r3)+
;      cmp     r0,$key+128.
;      blo     1b
;
;
;
;      now spin the cage against the wheel to produce output.
;
;      mov     $word,r4
;      mov     $wheel+128.,r3
;3:
;      mov     -(r3),r2
;      mov     $cage,r0
;      clr     r5
;1:
;      bit     r2,(r0)+
;      beq     2f
;      incb    r5
;2:
;      cmp     r0,$cage+256.
;      blo     1b
;
;
;      we have a piece of output from current wheel
;      it needs to be folded to remove lingering hopes of
;      inverting the function
;
;      mov     r4,-(sp)
;      clr     r4
;      div     $26.+26.+10.,r4
;      add     $'0,r5
;      cmp     r5,$'9
;      blos    1f
;      add     $'A-'9-1,r5
;      cmp     r5,$'Z
;      blos    1f
;      add     $'a-'Z-1,r5
;1:
;      mov     (sp)+,r4
;      movb    r5,(r4)+
;      cmp     r4,$word+8.
;      blo     3b
;
;
;      mov     (sp)+,r5
;      mov     (sp)+,r4
;      mov     (sp)+,r3
;      mov     (sp)+,r2
;      mov     (sp)+,r1
;      mov     $word,r0
;      rts     pc
;      .bss
;key:      .+.128.
;word:     .+.32.
;cage:     .+.256.
;wheel:    .+.256.

```

```
end START_CODE
```