```
// roff for multics submission

// 3/8/69

// M. D. McIlroy

manifest {
   Skip = $81000
   Form =$8014
   Maxline = 360
      }


global {
   Close : 7
   InitializeIO : 21
   Open : 6
   Readch : 15
   Writech : 16
      }

global {
   Ad : 101
   Ce : 102
   Conv : 103
   Dev : 104
   Ds : 105
   Ef : 106
   Efc : 107
   Efi : 108
   Efl : 109
   Eh : 110
   Ehc : 111
   Ehi : 112
   Ehl : 113
   Fi : 114
   Fp : 115
   In : 116
   Input : 117
   Ll : 118
   Lp : 119
   Ma : 120
   Ma1 : 121
   Ma2 : 122
   Ma3 : 123
   Ma4 : 124
   NNp : 125
   Nc : 126
   Ncp : 127
   Nl : 128
```

```
    Np : 129
    Nr : 130
    Of : 131
    Ofc : 132
    Ofi : 133
    Ofl : 134
    Oh : 135
    Ohc : 136
    Ohi : 137
    Ohl : 138
    Output : 139
    Pa : 140
    Pl : 141
    Print : 142
    Px : 143
    Roman : 144
    Un : 145

    Char : 200
    Char0 : 201
    Rawchar : 600
    Rawchar0 : 601
    Rawchar1 : 602
    Rawchar2 : 603
    Rawchar3 : 604
             }


let Roff(From, To, Device) be {Roff

    let C1_37 = table 6,'*b','*t','*n','*d','*k','\'
    let C2_37 = table 0,'*b','*t','*n','*d','*k',' '
    let C1_202 = table 4,'*b','*t','*n',Form
    let C2_202 = table 0,'*b','*t','*n',Form
    let C1_1050 = table 12,'*b','*k','*n','*d','*k','[','\',']',
       '''','{','}','~'
    let C2_1050 = table 0,'*b','*t','*n','*d','*k',' ',' ',' ',
       ' ',' ',' ',' ',' '
    let C1,C2 = C1_37,C2_37

    let v = vec 702
    let Vt = vec 8*Maxline+200
    InitializeIO(v,700)
    Input := Open("roff_input", "read")
    Output := Open("roff_output", "write")
    Output[10] := $8777   //For speed only

    Ad := true   // Adjust
    Ce := 0   //Center
    Ds := false   //Double space
    Fi := true   //Fill
    Pa := false   //Page numbering
    In := 0   //Indent to here
    Un := 0   //Undent to here
    Ehi,Ohi,Efi,Ofi := In   //Even,odd,head,foot indents
    Ll := 60   //Line length
```

```
    Ehl,Ohl,Efl,Ofl := L1   //Heading lengths
    Ehc,Ohc,Efc,Ofc := 0   //No of hcars in headings
    Nc := 0   //Number of characters in Char
    Fp := rv From   //First page to print
    Lp := rv To = 0 -* 100000, rv To   //Last Page to print
    NNp := 1   //Next page number
    Np := 0   //Current page number
    Print := Fp<=Np<=Lp   //Whether to issue output
    Nl := 0   //last used line number
    Nr := 0    //Number of raw input characters
    Pl := 66   //Paper length
    Ma := 6   //Margin
    Ncp := 0   //Number of haracters in page number
    Roman := false   //Roman numeral pagination
    Dev := rv Device   //Device code = rv 37,1050,2741,202
    Setmargin(Ma)

    Char := lv Char0  //Text buffer
    Rawchar := lv Rawchar0
    Eh := Vt
    Sethead(Eh, lv Ehi, lv Ehl, lv Ehc)
    Oh := Vt+Maxline
    Sethead(Oh, lv Ohi, lv Ohl, lv Ohc)
    Ef := Vt+2*Maxline
    Sethead(Ef, lv Efi, lv Efl, lv Efc)
    Of := Vt+3*Maxline
    Sethead(Of, lv Ofi, lv Ofl, lv Ofc)
     Px := Vt+4*Maxline   //Page number buffer

    Conv := Vt+4*Maxline+10    //output conversion table
    if Dev=202 then C1,C2 := C1_202,C2_202
    if Dev=1050 | Dev=2741 then C1,C2 := C1_1050,C2_1050
    for i = 0 to $837 do Conv[i] := Skip
    for i = $8040 to $8176 do Conv[i] := i
    for i = 1 to C1[0] do Conv[C1[i]] := C2[i]

//Here is the main program

    if Dev=202 then Write(Form)

    while Readline() do     {
       test Rawchar1='.'
          then Control()
          else Text()     ]
    Break()
    Eject()

    Close(Input)
    Close(Output)

    return ]Roff

and Control() be {

    manifest { L9 = $81000 }
    manifest {
```

```
    AD = 'a'*L9 + 'd'
    AR = 'a'*L9 + 'r'
    BP = 'b'*L9 + 'p'
    BR = 'b'*L9 + 'r'
    CE = 'c'*L9 + 'e'
    DS = 'd'*L9 + 's'
    EF = 'e'*L9 + 'f'
    EH = 'e'*L9 + 'h'
    FI = 'f'*L9 + 'i'
    FO = 'f'*L9 +'o'
    HE = 'h'*L9 + 'e'
    IN = 'i'*L9 + 'n'
    LI = 'l'*L9 + 'i'
    LL = 'l'*L9 + 'l'
    MA = 'm'*L9 + 'a'
    NA = 'n'*L9 +'a'
    NE = 'n'*L9 + 'e'
    NF = 'n'*L9 + 'f'
    NO = 'n'*L9 + 'o'
    OF = 'o'*L9 + 'f'
    OH = 'o'*L9 + 'h'
    OP = 'o'*L9 + 'p'
    PA = 'p'*L9 + 'a'
    PL = 'p'*L9 + 'l'
    RO = 'r'*L9 + 'o'
    SK = 's'*L9 + 'k'
    SP = 's'*L9 + 'p'
    SS = 's'*L9 + 's'
    TR = 't'*L9 + 'r'
    UN = 'u'*L9 + 'n'
       }
switchon (Rawchar2*L9 + Rawchar3) intc {
   default: return

   case AD: Break(); Ad := true; return

   case AR: Roman := false; return

   case BP: Break(); Eject(); return

   case BR: Break(); return

   case CE: Break(); Ce := Number(); Need(Ce); return

   case DS: Break(); Ds := true; return

   case EF: Sethead(Ef, lv Efi, lv Efl, lv Efc); return

   case EH: Sethead(Eh, lv Ehi, lv Ehl,lv Ehc); return

   case FI: Break(); Fi := true; return

   case FO: Sethead(Ef, lv Efi, lv Efl, lv Efc)
      Sethead(Of, lv Ofi, lv Ofl, lv Ofc); return

   case HE: Sethead(Eh, lv Ehi, lv Ehl, lv Ehc)
```

```
      Sethead(Oh, lv Chi, lv Ohl, lv Chc); return

   case IN: In,Un := Number();return

   case LI: for i = 1 to Number() test Readline()
      then Text() else break; return

   case LL: Ll := Number(); return

   case MA: Setmargin(Number()); return

   case NE: Need(Number()); return

   case NF: Break(); Fi := false; return

   case NA:
   case NC: Break(); Ad := false; return

   case OF: Sethead(Of, lv Ofi, lv Ofl, lv Ofc); return

   case OH: Sethead(Oh, lv Ohi, lv Ohl, lv Ohc); return

   case OP: Break(); Eject(); NNp := Np+Np%2+1; return

   case PA: Break(); Eject(); NNp := Number(); return

   case PL: Pl := Number(); return

   case RC: Roman := true; return

   case SK: NNp := NNp+Number(); return

   case SP: Break()
      for i = 1 to Min(Number(),Pl-Ma-Nl) do {
         Spacing(); Newline() }
      Need(2); return

   case SS: Break(); Ds := false; Need(2); return

   case TR: Translate(); return

   case UN: Break(); Un := Max(0,In-Number()); return }}

and Text() be {
   if Rawchar1=' ' | Rawchar1='*t' then Break()
   while Nr>1 & Rawchar[Nr-1]=' ' do
      Nr := Nr - 1
   for i = 1 to Nr do
      Char[Nc+1],Nc := Rawchar[i],Nc+1
   if Ce>0 do {
      Center(); return }
   unless Fi do {
      Break(); return }

   { let Ne,Ns,Nc1,Ne1 = 0,-1,0,0   //Ne,Ns = elements, gaps
   for i = 1 to Nc-1 do {   //Nc1,Ne1 = characters,elements at last gap
```

```
      Ne := Ne + Width(Char[i])
      if Un+Ne > Ll then break
      if Char[i]!=' ' & Char[i+1]=' ' then
         Nc1,Ne1,Ns := i,Ne,Ns+1 }
   if Nc1>=Nc-1 & Un+Ne<Ll then return
   if Nc1 = 0 do {
      Break(); return }
   Spacing()
   Blank(Un)
   test Ad
      then Adjust(Nc1,Ne1,Ns)
      else for i = 1 to Nc1 do Write(Char[i])
   Newline()
   while Char[Nc1+1]=' ' & Nc1<Nc do
      Nc1 := Nc1 + 1
   Nc := Nc  - Nc1
   for i = 1 to Nc do
      Char[i] := Char[Nc1+i]
   Un := In
      } repeat }

and Adjust(Nc1,Ne1,Ns) be {Adjust   //Nc,Ne,Ns=chars,elements,gaps
   let Pad = Ll - Un - Ne1   //Pad=total padding needed
   let Space = (Ns>0 -* (Pad-1)/Ns+1, 0)   //Space=padding per gap
   for i = 1 to Nc1 do {
      Write(Char[i])
      if Char[i]!=' ' & Char[i+1]=' ' then {
         let m = Min(Space,Pad)
         Blank(m)
         Pad := Pad - m }Adjust

and Break() be {
   if Nc=0 then return
   Spacing()
   Blank(Un)
   for i = 1 to Nc do Write(Char[i])
   Newline()
   Un := In
   Nc := 0 }

and Spacing() be {
   if Nl>0 do {
      if Ds & Nl<Pl-Ma do Newline()
      if Nl<Pl-Ma | 2*Ma>=Pl return
      Eject() }
   Print,Np,NNp := Fp<=NNp<=Lp,NNp,NNp+1
   for i = 1 to Ma1 do
      test Dev=202 then Nl := Nl+1
      else Newline()
   test Np%2=0
      then Title(Eh,Ehi,Ehl,Ehc)
      else Title(Oh,Ohi,Ohl,Ohc)
   for i = 1 to Ma2 do Newline() }

and Eject() be {
   unless Nl=0 do {
```

```
      for i = Nl+1 to Pl-Ma+Ma3 do Newline()
      test Np%2=0
         then Title(Ef,Efi,Efl,Efc)
         else Title(Cf,Cfi,Cfl,Cfc)
      test Dev=202 then Write(Form)
      else for i = 1 to Ma4 do Newline() }
   Nl := 0 }

and Need(n) be {
   if n*(Ds-*2,1)+Nl > Pl-Ma then Eject() }

and Number() = valof {
   let n, Anydigit = 0,false
   for i = 1 to Nr do {
      let Digit = Rawchar[i]-48
         if Digit<10 & Digit>=0 do
            n,Anydigit := 10*n+Digit,true }
      resultis (Anydigit -* n,1) }

and Width(Char) =
   $8040<=Char<=$8176 -* 1,
   Char='*b' -* -1,
   0

and Center() be {
   let Ne = 0
   for i = 1 to Nc do Ne := Ne+Width(Char[i])
   Spacing()
   Blank((Ll-In-Ne)/2 + In)
   for i = 1 to Nc do
      Write(Char[i])
   Newline()
   Nc := 0
   Ce := Ce-1 }


and Setmargin(M) be {
   Ma := Max(1,M)
   Ma2,Ma3 := Ma>1 -* 1,0
   Ma1,Ma4 := Ma-1-Ma2 }

and Sethead(Head;Lhi,Lhl,Lhc) be {
   let i,j,k,c = 4,0,0,0
   while i<Nr & Rawchar[i]=' ' do
      i := i + 1
   if i<Nr then c := Rawchar[i]
   while j<4 & i<Nr do {
      if Rawchar[i]=c then j := j+1
      Head[k+1],k,i := Rawchar[i],k+1,i+1 }
   while j<4 do
      Head[k+1],k,j := c,k+1,j+1
   rv Lhi, rv Lhl, rv Lhc := In,Ll,k }

and Title(Head,Hi,Hl,Hc) be {
   let i,j = 2,0
   let c = Head[1]
```

```
   let D = vec 3   //D[i]=gap before ith subfield
   let H = vec 3   //H[i]=width of ith subfield
   H[1],H[2],H[3] := 0
   Pagen()
   for k = 1 to Hc do {
      let h = Head[k]
      test h=c then j :=j+1
      else test h='%' then H[j] := H[j]+Ncp
      else H[j] := H[j]+Width(h) }
   D[1],D[2] := Hi, (Hl-Hi-H[2])/2
   D[3],D[2] := (Hl-Hi-D[2]-H[2]-H[3]), D[2]-H[1]
   for k = 1 to 3 do {
      Blank(D[k])
      while Head[i]!=c & i<Hc do {
         test Head[i]='%'
            then for j = 1 to Ncp do Write(Px[j])
            else Write(Head[i])
         i := i + 1 }
      i := i + 1 }
   Newline() }

and Pagen() be {
   let I = table 0, 'i', 'x', 'c', 'm'
   let V = table 0, 'v', 'l', 'd'
   let D = vec 5
   let i,n = 0,Np
   Ncp := 0
   while n>0 do
      D[i+1],n,i := n%10,n/10,i+1
   while i>0 do {
      let d = D[i]
      test Roman then
         test d%5 = 4 then
            Px[Ncp+1],Px[Ncp+2],Ncp := I[i],(d>=5-*I[i+1],V[i]),Ncp+2
         else   { if d>=5 then Px[Ncp+1],Ncp := V[i],Ncp+1
            for k = 1 to d%5 do
               Px[Ncp+1],Ncp := I[i],Ncp+1 }
      else Px[Ncp+1],Ncp := d+48,Ncp+1
      i := i-1 }}

and Newline() be  {
   Write('*n')
   Nl := Nl+1 }

and Blank(n) be {
   for i = 1 to n do Write(' ') }

and Readline() = valof {
   if Np>Lp then resultis false
   Nr := 0
   { Nr := Nr + 1
   Readch(Input, lv Rawchar[Nr])
   if Rawchar[Nr] = $8377 then resultis false
      } repeatuntil Rawchar[Nr]='*n' | Nr>Maxline
   Rawchar[Nr] := ' '
   resultis true }
```

```
and Write(c) be {
   let Cc = Conv[c]
   if !Print | Cc=Skip | c>$8176 return
   Writech(Output,Cc) }

and Translate() be {
   let i = 4
   while i<=Nr-1 & Rawchar[i]=' ' do
      i := i + 1
   while i<=Nr-1 do
      Conv[Rawchar[i]],i := Rawchar[i+1],i+1 }

and Min(a,b) = a<b -* a, b
and Max(a,b) = a>b -* a, b
```