UTS FOR DESIGN ENGINEERS

PAT GELSINGER

1/23/84

OUTLINE

I. WHY UTS

II. SOME USEFUL FEATURES of UTS

III. SOME EXAMPLES

IV. QUESTIONS

WHY UTS ??

1. UNIX RUNS ON ALL machines we as DE's need to use. HARDWARE INDEPENDENT.
   a. IBM
   b. VAX
   c. WORK STATIONS
   d. 86/3XX
   e. CRAY ??
2. UNIX is a VERY PRODUCTIVE ENVIRONMENT
   a. Rich SET OF TOOLS
   b. MANY USEFUL FEATURES
   c. DESIGNED FOR PROGRAMMERS. WE AS DE's SPEND MUCH TIME AS PROGRAMMERS

WHY UTS cont.

3. WELL KNOWN

   a. MANY NCG's TRAINED ON UNIX

   b. MANY UNIVERSITY'S HAVE DESIGN TOOLS TARGETED FOR UNIX

4. GENERALLY ACCEPTED / PREFERRED

   a. FUTURE IS VERY BRIGHT FOR UNIX.

      - DE FACTO STANDARD FOR MINI/MICRO/WORK STATIONS.
      - GOOD FUTURE SUPPORT
      - MANY FUTURE ENHANCEMENTS

   b. RELIABLE CHOICE FOR O.S. WILL

      - NOT BECOME EXTINCT LIKE TOPS-10, TOPS-20, CMS?, VMS?

SOME USEFUL FEATURES

1. PIPES, I/O REDIRECTION

   a. Allow consistent interface between programs.

   b. ENCOURAGES divide + conquer programming Techniques

   eg. num | pr -h | sed — | \
   umpunch —

   c. Eliminates need for many temporary files.

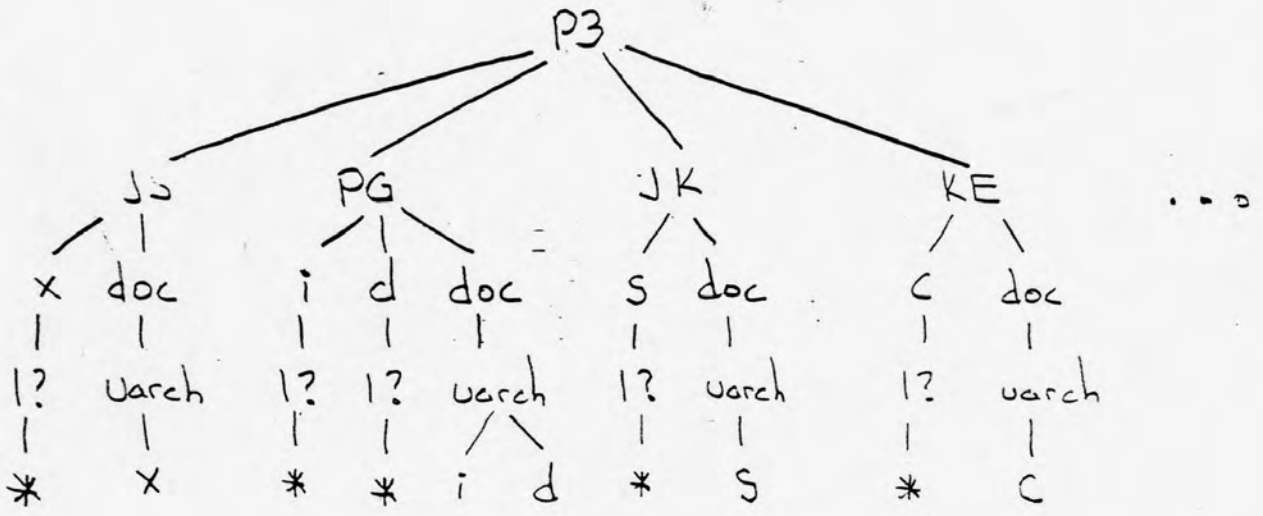
Useful Features cont.

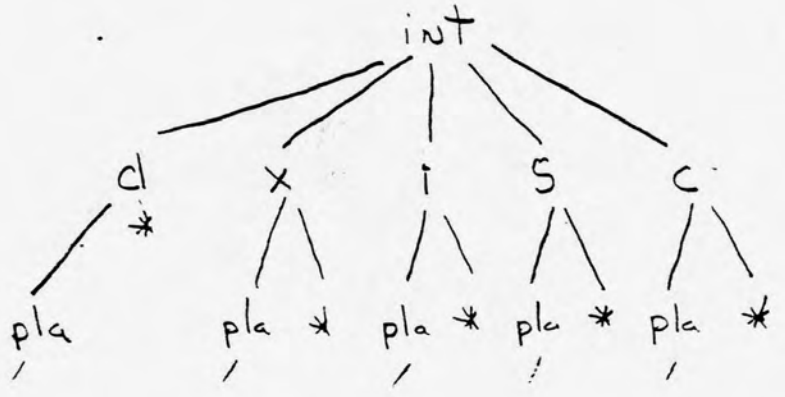
2. Hierarchy

   a. Easy management of large Number of files

   b. Eases Management of multi man projects

1)





Useful Features (cont.)

3. Shell (sh or csh)
   a. Flexible user interface
   b. Powerful Interpretive interface language
   c. Consistent user interface on multiple machines

examples:

1)
```
foreach i ( */*.msl )
    grep -s sd32 $i
    if ( $status == 0) then
        echo sd32 in $i
    endif
end
```

2)
```
history
    cat /local/bin/print
    print !$
!c
    cat /usp/p3/pgelsing/i/14/iunit.msl
    ^usp^usr^
```

3)
```
alias p 'print -n -h -t'
p file
```

Useful features (cont.)

4) shell variables

```
setenv ALL=( pgelsing jslager jkrauskopf .. )
mail $ALL
```

5) meta characters.

```
p */?unit.msl
```

4. grep - Useful for pattern searching in large + multiple files

```
grep sd32 */*.msl
```

5. Sed - stream editing
   - Powerful regular expressions

```
sed -n '/^[A-Z][0-9]*_.*\./p'
```

(see examples)

Useful Features (cont.)

6. awk - pattern scanning/processing/editing

```
awk '{ printf "%20s %10s\n", $1, $2 }'
```

(see examples)

7. save/rest - file/program maintenance

```
save iunit.mst
After update of xyz field encodings
^D
```

8. make - source program maintenance
   - define set of dependencies and update rules
   - make keeps all files current

Examples

1) pc - pla compilation program

2) cv.var - convert "var" in "file" to "new.var"

3) print - print files from UTS

4) makefile - makefile to recreate P3 RTL model

5) insert.ep - insert labels micro assemble into entry point logmin source

6) cmpdir - compare to trees

Example (1)

```
1    : 'pc: pla compile, This program takes pla input notation and produces
2    : 'a pla output file, pla_name.srx is assumed as the input file'
3    : 'Patrick Gelsinger   Intel Corp,   6/27/83'
4
5    if (| "$1" = "-h" |) || (| $# = 0 |)
6    then
7        echo "useage: $0 (|-h|) pla_name"
8        echo " $0 compiles the given pla into a .bin format,"
9        echo "A pla_name.srx is assumed to exist,"
10       exit
11   fi
12   PLA=$1
13   if (| @( $# = 1 @) -a @( -f $(<PLA>).srx @) |)
14   then
15        echo $LGM
16        $LGM <<MEND
17        do $(<PLA>).srx
18        print pla $PLA % $(<PLA>).out
19        write $PLA % $(<PLA>).bin
20        exit
21   MEND
22        mv $(<PLA>).out $(<PLA>).pla
23   elif (| @( $# = 2 @) -a @( ".$1" = ".-r" @) -a @( -f $2.srx @) |)
24   then
25        PLA=$2
26        echo $LGM
27        $LGM <<MEND
28        do $(<PLA>).srx
29        reduce $PLA
30   yes
31   n
32        print pla spm$PLA % $(<PLA>).out
33        write spm$PLA % $(<PLA>).bin
34   exit
35   MEND
36        mv $(<PLA>).out $(<PLA>).pla
37   elif (| -f $(<PLA>).pla |)
38   then
39        $MSL <<END
40        $P3/tool/CNVPLA.RIM
41   $(<PLA>)
42   END
43        rm -f /tmp/pc.$$
44   else
45       echo "$0: $(<PLA>).srx does not exist!!"
46       exit
47   fi
48
```

Example (2)

```
1     : 'A little program to convert and old variable name in some file'
2     : 'into a new variable name in a new file'
3     : 'Useage: conv.var old_name new_name (|ifile|) (|ofile|)'
4     :
5     : 'Patrick Gelsinger   4/1/83'
6
7     OVAR="" ; NVAR="" ; IFILE="" ; OFILE="";
8
9     if (| $# -lt 2 |)
10    then
11        echo "useage: $0 old_name new_name (|in_file (|out_file|) |)"
12        echo "(Note: in_file maybe equal to out_file if you trust it)"
13        exit
14    fi
15    OVAR=$1
16    NVAR=$2
17    if (| $# = 3 |)
18    then
19        IFILE=$3
20    elif (| $# = 4 |)
21    then
22        IFILE=$3
23        OFILE=$4
24    fi
25    if (| "$IFILE" = "" |)
26    then
27        cat > /tmp/I$$
28        IFILE=/tmp/I$$
29    fi
30    if grep -s $OVAR < $IFILE
31    then
32         cat /dev/null
33    else
34         echo $OVAR does not exists in input file: $IFILE
35         exit
36    fi
37    sed "s/#$OVAR@((|#@_a-zA-Z0-9|)@)/$NVAR@1/g
38         s/@((|#@_a-zA-Z0-9|)@)$OVAR$/@1$NVAR/g
39         s/#$OVAR$/$NVAR/g
40         s/@((|#@_a-zA-Z0-9|)@)$OVAR@((|#@_a-zA-Z0-9|)@)/@1$NVAR@2/g" @
41         < $IFILE > /tmp/O$$
42    if (| "$OFILE" = "" |)
43    then
44        cat /tmp/O$$
45    else
46        mv /tmp/O$$ $OFILE
47    fi
48    rm -f /tmp/*$$
49
```

Example (3)

Jan 23 08:11 1984 /local/bin/print Page 1

```
1     : 'print files to the any line printer. This program prints'
2     : 'the file with a banner.'
3     : 'four options exist, -c print the file with carriage control,'
4     : 'and -n number the file before printing it, -h print the file'
5     : 'with page headers, -t translate special characters before send'
6     : 'to the printer'
7     : ' Pat Gelsinger      7/26/83'
8     TRANSLATE=0 ; HEADER=0 ; NUMBER=0 ; CARCON=0
9     FF='ascii ff'
10    LF='ascii lf'
11    COMF=/tmp/prt.$$
12    NCOMF=/tmp/nprt.$$
13    cat <<'END' > $COMF
14       for i
15       do
16       ascii ff
17       banner 'basename $i'
18       ls -l $i
19       ascii ff
20    END
21    echo -n 'cat $i' >> $COMF
22    for i
23    do
24         case $i in
25            -h) echo -n '| pr -h $i ' >> $COMF
26                HEADER=1 ;;
27            -n) echo -n '| num ' >> $COMF
28                NUMBER=1 ;;
29            -c) echo -n "| sed 's/^1/$FF/
30                s/^ /$LF/
31                s/^0/$LF$LF/
32                s/^-/$LF$LF$LF/' " >> $COMF
33                CARCON=1 ;;
34            -t) echo -n "| sed 's/@(/@(@/g
35                s/@)/@)@/g
36                s/@</@(@</g
37                s/@>/@>@/g'" >> $COMF
38                TRANSLATE=1 ;;
39             -*) echo "$0: unknown switch $i"; exit ;;
40             *)  FILE="$FILE $i"
41         esac
42    done
43    echo >> $COMF
44    echo done >> $COMF
45    : check to see if the prtsite is valid
46    if grep -s "^$prtsite$" <<END
47    aloha
48    A13
49    AL03
50    AL3
51    A103
52    sc04
53    sc01
54    sj01
55    a103
56    c2
```

```
57    ttl3
58    ttl5
59    system
60    sz1
61    END
62    then
63       prtsite=$prtsite
64    else
65       echo $0: $prtsite, not a valid printer
66       exit
67    fi
68
69    # handle pipes to print
70    if (! "$FILE" = "" !)
71    then
72        sed '1,4d
73             s/cat $i/cat/
74             $d' < $COMF > $NCOMF
75        ( date ; banner 'logname' ; sh $NCOMF ) | opr -v rscs -t $prtsite
76        rm -f $NCOMF $COMF
77        exit
78    fi
79    for i in $FILE
80    do
81        if (! ! -f $i !)
82        then
83             echo "$0: $i does not exist"
84             rm -f $COMF $NCOMF
85             exit
86        fi
87    done
88    ( echo ; date ; pwd ; banner 'logname' ; sh $COMF $FILE
89    rm -f $COMF $NCOMF ) @
90    | opr -v rscs -t $prtsite
91
92
```

Example (4)


```
1     # makefile for the iunit
2
3     LUSM = /usr/p3/lib/usim
4     STUB = header SIMUL.RIM ISTUB.RIM SYSMOD.RIM
5     HEAD = header
6     UCD = /usr/p3/icdu/u/ucode.lab
7
8     all: $STUB IUNIT.RIM iunit.cal pla
9
10    IUNIT.RIM: $HEAD iunit.msl iunit.sym iunit.dec iunit.var
11        c iunit.msl
12
13    SIMUL.RIM: $LUSM/simul.msl
14        c $LUSM/simul.msl
15
16    SYSMOD.RIM: global.sym iunit.cal istub.cal $LUSM/findname.msl header
17        c sysmod.msl
18    global.sym: global.var
19        entersymbol global
20    global.var: iunit.dec istub.dec
21        mk.global.var >/dev/null
22    iunit.cal: iunit.dec
23        mk.cal iunit >/dev/null
24    istub.cal: istub.dec
25        mk.cal istub >/dev/null
26
27    iunit.sym: iunit.var
28        entersymbol -var unit
29
30    ISTUB.RIM: istub.msl istub.sym istub.dec header
31        c istub.msl
32    istub.sym: istub.msl
33        entersymbol stub
34
35    header: ../header.msl $LUSM/userin.hdr $LUSM/symtbl.hdr @
36                $LUSM/plamod.hdr $LUSM/memod.hdr @
37                istub.dec iunit.dec global.var
38        c ../header.msl; rm HEADER.RIM
39
40    pla::
41         @cd pla ; make UCD=$UCD
42
43    rom::
44         @cd rom ; make
45
```

```
1     # create the spimduck model (sequence is: kunit iunit, cunit,
2     # dunit, sunit, punit, munit ).
3     # Pat Gelsinger 12/16/83
4     LUSM = /usr/p3/lib/usim
5
6     all: header ucode cunit dunit kunit iunit sunit punit munit @
7          SIMUL.RIM SYSMOD.RIM ASTUB.RIM
8
9     ### for JK to make life easier (sorry PG)
10    us: header ucode cunit dunit kunit iunit sunit punit munit @
11         SIMUL.RIM SYSMOD.RIM ASTUB.RIM
12         usim
13
14    ### create all the units, Use the individual units makefile with STUB=
15    ### so that any stub related stuff is not created.
16
17    ASTUB.RIM: a/astub.sym a/astub.msl a/astub.dec header
18         cd a ; c astub.msl ; cp ASTUB.RIM ..
19    a/astub.sym: a/astub.msl
20        cd a ; entersymbol stub
21    a/astub.cal: a/astub.dec
22        cd a ; mk.cal astub >/dev/null
23
24    ucode::
25         cd u ; make
26    cunit::
27         cd c ; make STUB= HEAD=../header ; @
28         cp CUNIT.RIM ..
29    kunit::
30         cd k ; make STUB= HEAD=../header ; @
31         cp KUNIT.RIM ..
32    dunit::
33         cd d ; make STUB= HEAD=../header ; @
34         cp DUNIT.RIM ..
35    iunit::
36         cd i ; make STUB= HEAD=../header UCD=/usr/p3/d2/int2/u/ucode.lab
37         cp IUNIT.RIM ..
38    sunit::
39         cd s ; make STUB= HEAD=../header ; @
40         cp SUNIT.RIM ..
41    punit::
42         cd p ; make STUB= HEAD=../header ; @
43         cp PUNIT.RIM ..
44    munit::
45         cd m ; make STUB= HEAD=../header ; @
46         cp MUNIT.RIM ..
47
48    ### create all the other stuff needed for the model.
49
50    SIMUL.RIM: $LUSM/simul.msl header
51        c $LUSM/simul.msl
52    SYSMOD.RIM: global.sym  sysmod.msl  $LUSM/findname.msl header @
53                i/iunit.cal d/dunit.cal c/cunit.cal k/kunit.cal @
54                s/sunit.cal p/punit.cal m/munit.cal a/astub.cal
55        c sysmod.msl
56    global.sym: global.var
```

```
57        entersymbol global
58    global.var:       i/iunit.dec c/cunit.dec d/dunit.dec k/kunit.dec @
59                      s/sunit.dec p/punit.dec m/munit.dec a/astub.dec
60        mk.all.g.var i/iunit.dec c/cunit.dec d/dunit.dec k/kunit.dec @
61                      s/sunit.dec p/punit.dec m/munit.dec a/astub.dec
62
63    ### NOTICE that new.memod.hdr is used rather than memod.hdr. This new
64    # memod includes the sload event which is needed when doing jumps etc
65
66    header: header.msl $LUSM/userin.hdr $LUSM/symtbl.hdr @
67                $LUSM/plamod.hdr $LUSM/new.memod.hdr @
68                i/iunit.dec c/cunit.dec d/dunit.dec global.var  @
69                k/kunit.dec s/sunit.dec p/punit.dec m/munit.dec a/astub.d
70        c header.msl; rm HEADER.RIM; cp header c ; @
71        cp header d ; cp header i  ; cp header k ; @
72        cp header s ; cp header p  ; cp header m ; cp header a
73
```

Example (5)


```
1     : 'automatically include entry points from the microcode'
2     : ' Input is the entpla.src and output is the entpla.srx'
3     if   (| $# = "0" |) ; then
4         echo "useage: $0 entry_point_file"
5         exit
6     elif (| ! -f $1 |) ; then
7         echo "$0: cannot open $1"
8         exit
9     fi
10    sed -e '
11    s/@./_/g
12    s/@(([A-Z0-9_|)(|A-Z0-9_|)*@)  *@(([0-9A-F|)@) *$/@1 = ent<H00@2>/
13    s/@(([A-Z0-9_|)(|A-Z0-9_|)*@)  *@(([0-9A-F|)(|0-9A-F|)@) *$/@1 = ent<H0@
14    s/@(([A-Z0-9_|)(|A-Z0-9_|)*@)  *@(([0-9A-F|)(|0-9A-F|)(|0-9A-F|)@) *$/@1
15    /#EA/(<
16    s/#EA//
17    s/$/,es<01>/
18    >)
19    /#PUSH/(<
20    s/#PUSH//
21    s/$/,es<10>/
22    >)
23    /#POP/(<
24    s/#POP//
25    s/$/,es<11>/
26    >)
27    /#SSB/(<
28    s/#SSB//
29    s/$/,ssb/
30    >)
31    /#LK/(<
32    s/#LK//
33    s/$/,lcka/
34    >)
35    /#IOPL/(<
36    s/#IOPL//
37    s/$/,iopl/
38    >)
39    s/$/ ,/
40    $s/@,$/;/
41    '< $1 | @
42    awk '(<printf "%20s %1s %30s %1s@n",$1,$2,$3,$4>)' > /tmp/ep.$$
43    sed -n '1,/(< BEGIN ENTRY POINTS >)/p' < entpla.src > /tmp/top.$$
44    sed -n '/(< END ENTRY POINTS >)/,$p' < entpla.src > /tmp/bot.$$
45    cat /tmp/top.$$ /tmp/ep.$$ /tmp/bot.$$ > entpla.srx
46    rm -f /tmp/*.$$
47
```

```
57    ST021 = ent<h021> ,
58    ST022 = ent<h022> ,
59    ST023 = ent<h023> ,
60    ST024 = ent<h024> ,
61    ST025 = ent<h025> ,
62    ST026 = ent<h026> ,
63    ST027 = ent<h027> ,
64    ST028 = ent<h028> ,
65    ST029 = ent<h029> ,
66    ST02A = ent<h02A> ,
67    ST02B = ent<h02B> ,
68    ST02C = ent<h02C> ,
69    ST02D = ent<h02D> ,
70    ST02E = ent<h02E> ,
71    ST02F = ent<h02F> ,
72    ST030 = ent<h030> ,
73    ST031 = ent<h031> ,
74    ST032 = ent<h032> ,
75    ST033 = ent<h033> ,
76    ST034 = ent<h034> ,
77    ST035 = ent<h035> ,
78    ST036 = ent<h036> ,
79    ST037 = ent<h037> ,
80    ST038 = ent<h038> ,
81    ST039 = ent<h039> ,
82    ST03A = ent<h03A> ,
83    ST03B = ent<h03B> ,
84
85    (< *************** ENTRY POINT FIELD ********************************** >
86    (< Automated entry point inclusion, never touch the next 2 lines >)
87    (< BEGIN ENTRY POINTS >)
88                  ALIAXL =                          ent<H0BB> ,
89                  ALMITM =          ent<H0CC>,es<01>,lcks ,
90                  ALMRTM =          ent<H0DA>,es<01>,lcks ,
91                  ALRITR =                          ent<H0B9> ,
92                  ALRMTR =               ent<H0C2>,es<01> ,
93                  ALRRTR =                          ent<H0B3> ,
94                  BSRCIM =               ent<H151>,es<01> ,
95                  BSRCIR =                          ent<H14A> ,
96                  BSRCRM =               ent<H13A>,es<01> ,
97                  BSRCRR =                          ent<H134> ,
98                    BTIM =               ent<H12F>,es<01> ,
99                    BTIR =                          ent<H12C> ,
100                   BTRM =               ent<H122>,es<01> ,
101                   BTRR =                          ent<H11F> ,
102               CALLDLCP =                          ent<H35A> ,
103               CALLILCP =                          ent<H35B> ,
104                CALLSHD =                          ent<H323> ,
105                CALLSIM =               ent<H32B>,es<01> ,
106                CALLSIR =                          ent<H329> ,
107                  CHECK =                          ent<H35C> ,
108                CHKRPLM =                          ent<H35D> ,
109                CHKRPLR =                          ent<H35E> ,
110                  CLRTS =                          ent<H35F> ,
111                CMPSNR =                           ent<H279> ,
112               CMPSREP =                           ent<H263> ,
```

```
1     ALIAXL                          BB
2     ALMITM@EA@LK                    CC
3     ALMRTM@EA@LK                    DA
4     ALRITR                          89
5     ALRMTR@EA                       C2
6     ALRRTR                          83
7     BSRCIM@EA                       151
8     BSRCIR                          14A
9     BSRCRM@EA                       13A
10    BSRCRR                          134
11    BTIM@EA                         12F
12    BTIR                            12C
13    BTRM@EA                         122
14    BTRR                            11F
15    CALLDLCP                        35A
16    CALLILCP                        35B
17    CALLSHD                         323
18    CALLSIM@EA                      328
19    CALLSIR                         329
20    CHECK                           35C
21    CHKRPLM                         35D
22    CHKRPLR                         35E
23    CLRTS                           35F
24    CMPSNR                          279
25    CMPSREP                         263
26    CTAXLI                          8F
27    CTMI@EA                         D5
28    CTMR@EA                         E3
29    CTRI                            8D
30    CTRM@EA                         C7
31    CTRR                            85
32    CWD                             238
33    DIVAXM@EA                       224
34    DIVAXR                          21C
35    ENTER0                          34C
36    ENTER16@PUSH                    32E
37    ENTER32@PUSH                    332
38    ESC1                            360
39    ESC2                            361
40    ESC3                            362
41    ESC4                            363
42    ESC5                            364
43    ESC6                            365
44    ESC7                            366
45    ESC8                            367
46    EXTCIM@EA                       192
47    EXTCIR                          18C
48    EXTCRM@EA                       1A7
49    EXTCRR                          1A1
50    FALC                            AF
51    HALTC                           368
52    HALTP                           369
53    IAAA.S                          1E1
54    IAAD                            1F2
55    IAAM                            1E9
56    ICERET                          36A
```

Example 6


```
1     #
2     # This program is to do an intelligent comparison of two directorys
3     # showing files that are added, and files that have changed.
4     #
5     # Pat Gelsinger  1/6/84
6     #
7     set TDIR=/local/tmp
8     unset HELP SUBDIR DIREX FILEX NOFDIF
9     unset DIR1 DIR2
10    set SWITCH
11    foreach i ( $* )
12       switch ($i)
13       case -*:
14                foreach k ( `echo $i | sed -e 's/*,//' -e 's/,/ &/g'` )
15                   switch ($k)
16                   case d:
17                            set DIREX ; breaksw
18                   case s:
19                            set SUBDIR ; breaksw
20                   case h:
21                            set HELP   ; breaksw
22                   case a:
23                            set DIREX FILEX ; breaksw
24                   case e:
25                            set FILEX  ; breaksw
26                   case f:
27                            set NOFDIF  ; breaksw
28                   default:
29                            echo $0@: Unknown switch $k
30                            set HELP   ; breaksw
31                   endsw
32                end
33                set SWITCH="$SWITCH $i"
34                breaksw
35       default:
36                if ( $?DIR1 == 0 ) then
37                    set DIR1=$i
38                else if ( $?DIR2 == 0 )
39                    set DIR2=$i
40                else
41                    echo $0@: Invalid command line option: $i
42                    set HELP
43                endif
44                breaksw
45       endsw
46    end
47    if ( $?DIR2 == 0 ) then
48        set DIR2=`pwd`
49    endif
50    if ( $?HELP == 1 || $?DIR1 == "0" ) then
51       echo "useage: " $0 @(|-hasdfe@|) first_dir @(|second_dir@|)
52       echo "        " compare first_dir to second_dir, if no
53       echo "        " second_dir compare current dir to second_dir.
54       echo "        " -h  print this message
55       echo "        " -a  turn on all options
56       echo "        " -s  Do NOT check through all subdirectories
```

```
57      echo "          " -d  show subdirs that do not exist in both dirs
58      echo "          " -e  show files that do not exist in both dirs
59      echo "          " -f  Do NOT show files that exist but differ
60      exit (1)
61    endif
62
63    # sanity check on directories
64    if ( ! -d $DIR2 ) then
65       echo $0:q: $DIR2 is not a directory
66    endif
67    if ( ! -d $DIR1 ) then
68       echo $0:q: $DIR1 is not a directory
69    endif
70    if ( ! -d $DIR1 || ! -d $DIR2 ) then
71        exit (1)
72    endif
73
74    @ i=1
75    foreach k ( $DIR1 $DIR2 )
76      ls -1F $k | tee $TDIR/$$.$i.ls | grep
77      sed -n -e 's/[*/=@|]$//p' > $TDIR/$$.$i.dir
78      sed -e '/[*/=@|]$/d' \
79          -e 's/[*/=@|]*$//' < $TDIR/$$.$i.ls > $TDIR/$$.$i.fil
80      @ i++
81    end
82    unset i
83
84    # Begin comparison
85    # Fork a sub csh so that we can cummulate all comparisons and see if the
86    # are any differences before printing any messages
87
88    csh <<END > $TDIR/$$.all
89
90    # Do directory comparison first
91    if ( $?DIREX == 1 ) then
92      comm -23 $TDIR/$$.1.dir $TDIR/$$.2.dir > $TDIR/$$.d
93      if ( -f $TDIR/$$.d && ! -z $TDIR/$$.d ) then
94         echo "#    "The following dirs are in "$DIR1" but not in "$DIR2"
95         sed 's/^/#         /' < $TDIR/$$.d
96      endif
97      comm -13 $TDIR/$$.1.dir $TDIR/$$.2.dir > $TDIR/$$.d
98      if ( -f $TDIR/$$.d && ! -z $TDIR/$$.d ) then
99         echo "#    "The following dirs are in "$DIR2" but not in "$DIR1"
100        sed 's/^/#         /' < $TDIR/$$.d
101     endif
102   endif
103
104   # Do file comparisons now.
105   if ( $?FILEX == 1 ) then
106     comm -23 $TDIR/$$.1.fil $TDIR/$$.2.fil > $TDIR/$$.f
107     if ( -f $TDIR/$$.f && ! -z $TDIR/$$.f ) then
108        echo "#    "The following files are in "$DIR1" but not in "$DIR2"
109        sed 's/^/#         /' < $TDIR/$$.f
110     endif
111     comm -13 $TDIR/$$.1.fil $TDIR/$$.2.fil > $TDIR/$$.f
112     if ( -f $TDIR/$$.f && ! -z $TDIR/$$.f ) then
```

```
113       echo "#    "The following files are in @"$DIR2@" but not in @"$DIR
114       sed 's/#/         /' < $TDIR/$$.f
115     endif
116   endif
117
118   # Now show files that are in both directories but are different
119   if ( $?NOFDIF == 0 ) then
120     foreach i ( @'comm -12 $TDIR/$$.1.fil $TDIR/$$.2.file' )
121         cmp -s $DIR1/@$i $DIR2/@$i
122         if ( @$status != 0 ) then
123             echo @$i >> $TDIR/$$.c
124         endif
125     end
126     if ( -f $TDIR/$$.c && ! -z $TDIR/$$.c ) then
127       echo "#    "The following files are in @"$DIR1@" and @"$DIR2@"
128       echo "#    "BUT are different:
129       sed 's/#/         /' < $TDIR/$$.c
130     endif
131   endif
132
133   END
134   if ( -f $TDIR/$$.all && ! -z $TDIR/$$.all ) then
135       echo '@ '$DIR1 $DIR2 comparison
136       cat $TDIR/$$.all
137   endif
138
139   # search subdirectories last
140   # recursively search the common subdirectories
141   if ( $?SUBDIR == 0 ) then
142     foreach i ( 'comm -12 $TDIR/$$.1.dir $TDIR/$$.2.dir' )
143         $0 $SWITCH $DIR1/$i $DIR2/$i
144     end
145   endif
146   rm $TDIR/$$.*
147
```