
Subject: diagrami: Simple Diagrammer for IDL & PV-WAVE Language Programs

Posted by [grunes](#) on Thu, 29 Aug 1996 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

Archive-name: diagrami

Submitted-by: grunes@ssdd.nrl.navy.mil (Mitchell R Grunes)

****diagrami: Simple Diagrammer for IDL & PV-WAVE Language Programs****

The most recent changes: column alignment fixed, improved Unix and Vax VMS scripts, handled @file lines.

If you like or dislike it, send e-mail. DON'T POST to alt.sources.

Bug reports must include sample code on which it failed.

CONTENTS:

diagrami.for Fortran language source code.

diagrami.bat MS-DOS procedure to run without answering questions.

diagrami.sh Unix csh procedure to run without answering questions.

diagrami.vax VAX VMS DCL procedure to run without answering questions.

---diagrami.for-----CUT HERE-----

c EXAMPLE OF OUTPUT (looks better if you choose IBM PC line graphics):

```
c  +----- pro Sample,a,b,c                | 1
c  |      a=indgen(15)^2                    | 2
c  |+----- if a eq b then begin           | 3
c  ||      print,'A equals B'               | 4
c  ||      c=0                             | 5
c  |+----- else begin                    | 6
c  ||      print,'A does not equal B'       | 7
c  ||      c=1                             | 8
c  |+----- endif                        | 9
c  +----- end                            | 10
```

c Diagrams IDL and PV-Wave begin(or case)-end constructs, functions

c and procedures, places a * next to goto and return statements.

c

c Program by Mitchell R Grunes, ATSC/NRL (grunes@ssdd.nrl.navy.mil).

c Revision date: 8/25/96.

c If you find it useful, or find a problem, please send me e-mail.

c -----

c This program was written in FORTRAN, for historic reasons.

c This was written in Fortran 77 (with common extensions) for

c portability. It should also compile under Fortran 90 and Fortran 95,

c provided you tell the compiler it is in card format.

c-----

c I hope this works for you, but bear in mind that nothing short of
c a full-fledged language parser could really do the job. Perhaps
c worth about what you paid for it. (-:

c Versions: To diagram Fortran: diagramf.for
c IDL/PV-WAVE: diagrami.for
c C: diagramc.for
c MS-DOS procedures to call above programs without asking so many questions,
c append output to file diagram.out:
c Fortran: diagramf.bat (card format)
c diagram9.bat (free format)
c IDL/PV-WAVE: diagrami.bat
c C: diagramc.bat
c Similar Unix csh procedures:
c Fortran: diagramf.sh (card format)
c diagram9.sh (free format)
c IDL/PV-WAVE: diagrami.sh
c C: diagramc.sh
c Similar Vax VMS DCL procedures:
c Fortran: diagramf.vax (card format)
c diagram9.vax (free format)
c IDL/PV-WAVE: diagrami.vax
c C: diagramc.vax

```
program diagrami          ! Diagrammer for IDL and
                          ! PV-WAVE
character*80 filnam,filnam2
```

```
print*, 'IDL source filename?'
read(*, '(a80)') filnam
print*, filnam
```

```
print*, 'Output file (blank=screen)?'
read(*, '(a80)') filnam2
print*, filnam2
```

```
print*, 'Column in which to write line #'s ',
& ' (67 for 80 col screen, 0 for none):'
read*, LCol
print*, LCol
```

```
print*, 'Use IBM PC graphics characters (0=no):'
read*, iGraphics
print*, iGraphics
```

```
call diagram(filnam, filnam2, LCol, iGraphics)
end
```

C-----

```

subroutine diagram(filnam,filnam2,LCol,iGraphics)
c Program by Mitchell R Grunes, ATSC/NRL (grunes@ssdd.nrl.navy.mil).
character*80 filnam,filnam2
character*160 a,b
character*5 form
character*8 fm
character*1 c
logical find
external find
common icol,icol1
logical fout

```

c Symbols which will mark block actions:

```

character*1 BlockBegin (2) /'+','+'/ ! Start of block
character*1 BlockEnd (2) /'+','+'/ ! End of block
character*1 BlockElse (2) /'+','+'/ ! Else construct
character*1 BlockContinue (2) /'|','|'/ ! Block continues w/o change
character*1 BlockHoriz (2) /'-','-'/ ! Horizontal to start of line

```

c Same, but allows horizontal line to continue through:

```

character*1 BlockBeginH (2) /'+','+'/ ! Start of block
character*1 BlockEndH (2) /'+','+'/ ! End of block
character*1 BlockElseH (2) /'+','+'/ ! Else construct

```

```

if(iGraphics.ne.0)then
  iGraphics=1

```

```

BlockBegin (1)=char(218) ! (1)=normal
BlockEnd (1)=char(192)
BlockElse (1)=char(195)
BlockContinue(1)=char(179)
BlockHoriz (1)=char(196)
BlockBeginH (1)=char(194)
BlockEndH (1)=char(193)
BlockElseH (1)=char(197)

```

```

BlockBegin (2)=char(214) ! (2)=DO/FOR loops (doubled)
BlockEnd (2)=char(211) ! (not yet used)
BlockEnd (2)=char(211)
BlockElse (2)=char(199)
BlockContinue(2)=char(186)
BlockHoriz (2)=char(196)
BlockBeginH (2)=char(209)
BlockEndH (2)=char(208)
BlockElseH (2)=char(215)

```

```

endif

```

```

open(1,file=filnam,status='old')
fout=filnam2.gt.' '

```

```

if(fout)open(2,file=filnam2,status='unknown')

                                ! ASCII 12 is a form feed
if(fout)write(2,*)char(12),
& '=====---',filnam(1:LenA(filnam)),'-----'

if(fout) write(2,'(11x,a50,a49,/)' ) ! Write column header
& ' .....1.....2.....3.....4.....5',
& ' .....6.....7.....8.....9.....'
if(.not.fout)write(*,'(11x,a50,a49,/)' ),
& ' .....1.....2.....3.....4.....5',
& ' .....6.....7.....8.....9.....'

i1=0                                ! # nest levels before
                                ! current line
i2=0                                ! # nest levels on
                                ! current line
i3=0                                ! # of nest levels after
                                ! current line
i4=0                                ! not 0 to flag start or end
                                ! of block
InSub=0                            ! Inside a subroutine or
                                ! function?
nMain=0                            ! no mainline program yet
InCase=0                           ! not inside case
iContinue=0                        ! not continued from prior line
nline=0
10  a=' '
    read(1,'(a160)',end=99)a
    nline=nline+1
    fm=' '
    write(fm,'(i5)')nline
    form=fm

if(a(1:1).eq.char(12))then
    if(fout)write(2,'(a1,:)')char(12)
    if(.not.fout)print*,'-----FORM FEED-----'
    b=a(2:160)
    a=b
endif

b=' '                                ! Turn tabs to spaces
j=1
do i=1,LenA(a)
    if(a(i:i).eq.char(9))then
        j=(j-1)/8*8+8+1
    elseif(j.le.160)then
        b(j:j)=a(i:i)

```

```

    j=j+1
  endif
enddo
i=1
j=1
a=' ' ! Pre-processing
iquote=0 ! no ' yet
idquote=0 ! no " yet
j=1
do i=1,LenA(b)
  c=b(i:i)
  if(c.ge.'A'.and.c.le.'Z')c=char(ichar(c)+32)
  if(c.eq.';')goto 15 ! comment
  if(c.eq.'@'.and.i.eq.1)goto 15 ! other procedure includes
  if(c.eq.'"' .and.idquote.eq.0)then
    iquote=1-iquote
    c=' '
  endif
  if(c.eq.'"' .and.iquote .eq.0)idquote=1-idquote
  if(iquote.ne.0.or.idquote.ne.0)c=' '
  if(j.gt.1)then ! (kill multiple spaces)
    if(c.eq.' ' .and.a(j-1:j-1).eq.' ')j=j-1
  endif
  if(c.eq.':')then ! (put space after :)
    if(j.le.160) a(j:j)=':'
    j=j+1
    c=' '
  endif
  if(j.le.160) a(j:j)=c
  j=j+1
enddo

```

```

15 i2=i1
i3=i1
i4=0
igoto=0 ! no goto on line

```

```

if(a.ne.' ' .and.InSub.eq.0..and..not.
& (find(a,'function ',2).or.find(a,'pro ',2)))then ! mainline
  InSub=InSub+1
  nMain=nMain+1
  if(fout)print*, 'Line ',form,' ',b(1:LenA(b))
  if(nMain.gt.1)then
    PRINT*, '***ERROR--TOO MANY MAINLINES***'
    if(fout)WRITE(2,*) '***ERROR--TOO MANY MAINLINES!***'
    if(fout)print*,b
    print*,char(7)
  endif
endif

```

```

i2=i2+1
i3=i3+1
endif

if(find(a,'goto',8+32).or.find(a,'return',1+128))igoto=1

if(find(a,'endif ',2).or.find(a,'endfor ',2)
& .or.find(a,'endelse ',2).or.find(a,'endwhile ',2)
& .or.find(a,'endcase ',2).or.find(a,'endrep ',2))then
i3=i3-1
if(find(a,'begin ',1))i3=i3+1
i4=max(i4,1)
if(i3.lt.InCase)InCase=0
elseif(find(a,'case ',1).or.find(a,'begin ',1))then
InCase=i1
i2=i2+1
i3=i3+1
i4=max(i4,1)
if(find(a,': begin ',0))i4=max(i4,2)
if(find(a,'end ',1))i3=i3-1
elseif(find(a,'end ',2))then
if(i3.gt.0.or.InSub.gt.0)then      ! Problem: IDL end may
i3=i3-1                          ! actually be an endif,
                                ! endelse, etc.
if(i3.eq.0.and.InSub.ne.0)InSub=0
endif
if(i3.lt.InCase)InCase=0
elseif(find(a,'function ',2).or.find(a,'pro ',2))then
if(fout)print*, 'Line ',form, ' ',b(1:LenA(b))
InSub=InSub+1
i2=i2+1
i3=i3+1
if(InSub.ne.1.or.i3.ne.1)then
PRINT*, '***ERROR--INVALID DIAGRAMMING INDEX line',form
if(fout)
& WRITE(2,*)'***ERROR--INVALID DIAGRAMMING INDEX!***'
if(fout)print*,b
print*,char(7)
i3=1
InSub=1
endif
elseif((find(a,': ',0).or.find(a,':',256)).and.
& InCase.ne.0)then      ! simple case instances
i4=max(i4,1)
elseif((find(a,':',0).and.InCase.ne.0))then    !other case instances
ileft=0
iright=0
ileft2=0

```

```

    iringt2=0
    do i=1,icol1
        if(a(i:i).eq.'(')ileft=ileft+1
        if(a(i:i).eq.')')iright=iright+1
        if(a(i:i).eq.'[')ileft2=ileft+1
        if(a(i:i).eq.'])')iright2=iright+1
    enddo
    if(ileft.eq.iright.and.ileft2.eq.iright2.and.icontinue.eq.0)
&    i4=max(i4,1)
    endif

    icontinue=0
    if(find(a,'$ ',0))icontinue=1

    a=' '

    if(i1.lt.0.or.i2.lt.0.or.i3.lt.0.or.i4.lt.0)then
        PRINT*, '***ERROR--INVALID DIAGRAMMING INDEX line',form
        if(fout)WRITE(2,*) '***ERROR--INVALID DIAGRAMMING INDEX!***'
        if(fout)print*,b
        print*,char(7)
        i1=max(i1,0)
        i2=max(i2,0)
        i3=max(i3,0)
        i4=max(i4,0)
    endif

    i2=max(i1,i3)                ! # of nests on current line
    i4=max(i4,iabs(i3-i1))        ! not 0, to flag start or
                                ! end of block

    iBlock=1                     ! For the present version.

    a=' '                         ! Leave space for diagram
    a(12:160)=b                  ! (must match column header)

    LastUse=1                    ! Last usable diagram col
    dowhile(LastUse.lt.160.and.a(LastUse:LastUse).eq.' ')
        LastUse=LastUse+1
    enddo
    LastUse=LastUse-2

    if(igoto.ne.0)a(1:1)='*'      ! Place * next to jumps

    if(i2.gt.0)then              ! Draw one vertical line per
        do i=2,min(i2+1,LastUse) ! nest level.
            a(i:i)=BlockContinue(iBlock)
        enddo

```

```

endif

if(i4.ne.0)then          ! Draw horizontal lines inward
  do i=i2+2,LastUse      ! from above.
    a(i:i)=BlockHoriz(iBlock)
  enddo
endif

do i=0,i4-1              ! May need to replace some
                          ! vertical lines with
  c=      BlockElse(iBlock) ! else symbol
  if(i1+i.lt.i3)c=BlockBegin(iBlock) ! or begin symbol
  if(i1+i.gt.i3)c=BlockEnd (iBlock) ! or end symbol
  j=max(2,min(LastUse,i2+1-i))
  a(j:j)=c
  if(a(j+1:j+1).eq.BlockElse (iBlock)) ! Continue horizontal lines
&    a(j+1:j+1) = BlockElseH (iBlock)
  if(a(j+1:j+1).eq.BlockBegin (iBlock))
&    a(j+1:j+1) = BlockBeginH(iBlock)
  if(a(j+1:j+1).eq.BlockEnd (iBlock))
&    a(j+1:j+1) = BlockEndH (iBlock)
enddo

if(LCol.gt.0.and.a(max(1,LCol+11):160).eq.' ')then ! line #
  if(form(1:1).eq.' ')form(1:1)=BlockContinue(iBlock)
  a(LCol+11:160)=form
endif

n=LenA(a)              ! Output diagrammed line
if(fout) write(2,'(80a1,80a1)')(a(i:i),i=1,n)
if(.not.fout)write(*,'(1x,80a1,80a1)')(a(i:i),i=1,n)

i1=i3
goto 10
99  if(i3.gt.0.or.InSub.ne.0)then
    PRINT*, '***WARNING--SOME NEST LEVELS LEFT HANGING AT END***'
    if(fout)print*,b
    print*,char(7)
  endif
end

end

C-----
logical function find(a,b,icond)      ! find b in a, subject to
                                     ! conditions:
                                     ! icond=sum of the following:
                                     ! 1: Prior, if exists, must
                                     !    be blank
                                     ! 2: Must be first non-blank
                                     ! 4: Prior character, if

```



```

! present, must not be
! alphanumeric.
! 8: Prior character, if
! present, must be blank
! or )
! 16: Prior character, if
! present, must be blank
! or ,
! 32: Next character not
! alphanumeric
! 64: Next character not
! alphabetic
! 128:Next character must be
! blank or (
! 256:1st non-blank, possibly
! except for numeric
! labels

```

c Program by Mitchell R Grunes, ATSC/NRL (grunes@ssdd.nrl.navy.mil).

c Revision date: 8/25/96.

```

character(*) a,b
character*1 c,cNext,c2
common icol,icol1
logical result

```

```

ii=len(a)
jj=len(b)
result=.false.
do i=1,ii-jj+1
  if(a(i:i+jj-1).eq.b)then
    icol1=i                ! icol1=column of item found
    icol =i+jj             ! icol =column after item
                           ! found

```

```

    c=' '
    cNext=' '
    if(icol1.gt.1)c=a(icol1-1:icol1-1)
    if(icol .le.ii)cNext=a(icol:icol)

```

```

    result=.true.
    if(result.and.iand(icond,1).ne.0.and.icol1.gt.1)then
      result=c.eq.' '
    endif

```

```

    if(result.and.iand(icond,2).ne.0.and.icol1.gt.1)then
      result=a(1:icol1-1).eq.' '
    endif

```

```

    if(result.and.iand(icond,4).ne.0)
&    result=(c.lt.'0'.or.c.gt.'9').and.(c.lt.'a'.or.c.gt.'z')

```

```

        if(result.and.iand(icond,8).ne.0)result=c.eq.' '.or.c.eq. ')'
        if(result.and.iand(icond,16).ne.0)result=
&      c.eq.' '.or.c.eq. ','
        if(result.and.iand(icond,32).ne.0)
&      result=(cNext.lt.'0'.or.cNext.gt.'9').and.
&      (cNext.lt.'a'.or.cNext.gt.'z')
        if(result.and.iand(icond,64).ne.0)
&      result=(cNext.lt.'a'.or.cNext.gt.'z')
        if(result.and.iand(icond,128).ne.0)
&      result=cNext.eq.' '.or.cNext.eq. '('
        if(result.and.iand(icond,256).ne.0.and.icol1.gt.1)then
            ii=1
            do iii=1,icol1-1
                c2=a(iii:iii)
                if(c2.ge.'0'.and.c2.le.'9')ii=iii+1
                if(c2.ne.' '.and.(c2.lt.'0'.or.c2.gt.'9'))goto 20
            enddo
20      if(ii.lt.icol1)then
                result=a(ii:icol1-1).eq.' '
            endif
        endif

        find=result
        if(result)return
    endif
enddo
find=result
return
end

```

```

c-----
function LenA(a)                ! Length of string, at
                                ! least 1

```

c Program by Mitchell R Grunes, ATSC/NRL (grunes@ssdd.nrl.navy.mil).
c Revision date: 8/25/96.

```

character*(*) a
n=len(a)
dowhile(n.gt.1.and.a(n:n).eq.' ')
    n=n-1
enddo
LenA=n
end

```

---diagrami.bat-----CUT HERE-----

```

rem          ---diagrami.bat---
rem MS-DOS procedure to diagram an IDL or PV-WAVE language program.

rem by Mitchell R Grunes.

rem I assume that the executable is in directory c:\grunes on
rem your PC.

rem Syntax:
rem  diagrami
rem to be prompted for input parameters.

rem Alternate Syntax:
rem  diagrami filename(s)
rem to append diagram of file(s) into diagram.out

if %1a == a c:\grunes\diagrami
if %1a == a goto quit

echo off
:loop
echo ===== %1 =====
rem Prompt answers: input from %1, output to diagram2.sc (for now),
rem place numbers in column 67, use IBM PC graphics.

echo %1      > diagram.sc
echo diagram2.sc >> diagram.sc
echo 67      >> diagram.sc
echo 1       >> diagram.sc

c:\grunes\diagrami < diagram.sc
type diagram2.sc >> diagram.out
del diagram.sc
del diagram2.sc
shift
if not %1a == a goto loop
:quit
echo Note--This does not delete diagram.out before appending to it.
---diagrami.sh-----CUT HERE-----
#!/bin/csh
#          ---diagrami.sh---
#Unix csh procedure to diagram an IDL or PV-WAVE language program

# by Mitchell R Grunes.

#I assume that the executable and this procedure are in the search path,
# and that this procedure has execute permission.

```

```

#Syntax:
#  diagrami.sh
#to be prompted for input parameters.

#Alternate Syntax:
#  diagrami.sh filename(s)
#to append diagram of file(s) into diagram.out

if (${?noclobber}) then
  unset noclobber
  set  noclobbersave
endif

if %1a == a then
  diagrami
  goto quit
endif

loop:
echo =====-- %1 -----
#Prompt answers: input from %1, output to diagram2.sc (for now),
# place numbers in column 67, don't use IBM PC graphics.

echo %1      > diagram.sc
echo diagram2.sc >> diagram.sc
echo 67      >> diagram.sc
echo 0       >> diagram.sc

diagrami < diagram.sc
cat diagram2.sc >> diagram.out
rm diagram.sc
rm diagram2.sc
shift
if ! (%1a == a) then
  goto loop
endif
quit:
echo Note--This does not delete diagram.out before appending to it.
if (${?noclobbersave}) then
  set  noclobber
  unset noclobbersave
endif
---diagrami.vax-----CUT HERE-----
$!      ---diagrami.vax---
$!VAX VMS procedure to diagram an IDL or PV-WAVE language program
$!
$! by Mitchell R Grunes.
$!

```

```

$! assume that the executable and this procedure are in the search path,
$! and that this procedure has execute permission.
$!
$!Syntax:
$!  @diagrami.vax
$!to be prompted for input parameters.
$!
$!Alternate Syntax:
$!  @diagrami.vax filename(s)
$!to append diagram of file(s) into diagram.out
$
$ if P1 .EQS. ""
$ then
$   define/user sys$input sys$command
$   run diagrami
$   goto quit
$ endif
$
$ write sys$output "=====-- "+P1+" --=====
$
$! Must pre-create diagram.out if does not exist
$ open/append/error=noSkip diagram.out diagram.out
$ goto Skip
$noSkip:
$ open/write diagram.out diagram.out
$Skip:
$ close diagram.out
$
$! Must pre-create diagram2.sc with same file attributes
$ open/write diagram2.sc diagram2.sc
$ close diagram2.sc
$
$ !Prompt answers: input from P1, output to diagram2.sc (for now),
$ ! place numbers in column 67, don't use IBM PC graphics.
$
$ open/write diagram.sc diagram.sc
$ write diagram.sc "$Run diagrami"
$ write diagram.sc P1
$ write diagram.sc "diagram2.sc"
$ write diagram.sc "67"
$ write diagram.sc "0"
$ close diagram.sc
$ @diagram.sc
$ append diagram2.sc diagram.out
$ delete diagram.sc;*
$ delete diagram2.sc;*
$
$ if (P2 .NES. "") then @diagrami.vax 'P2' 'P3' 'P4' 'P5' 'P6' 'P7' 'P8'

```

```
$ write sys$output "Note--This does not delete diagram.out before appending to it."
```

```
$quit:
```

```
-----CUT HERE-----
```

```
-----
```

Mitchell R Grunes, grunes@nrlvax.nrl.navy.mil. Opinions are mine alone.
