Date February 7, 2002 Memo Number STI:02/05
Subject **Sheldon's ANSYS Tips and Tricks: APDL Coding Standards**
Keywords APDL

1. Introduction:

Because of the flexibility and automation APDL (ANSYS Parametric Design Language) provides, many users tend to write input files and macros in addition to using the GUI.

There are currently no recognized 'standards' of writing APDL macros or input files, although there may exist standards within companies on APDL coding. The lack of APDL coding standards may make inheriting someone else's input files or macros more difficult. Also, parameter or component conflicts may also arise when using multiple macros.

This memo hopes to provide some ideas for groups or individuals wishing to develop APDL standards in writing macros or input files.

2. Syntax Highlighting with Text Editors:

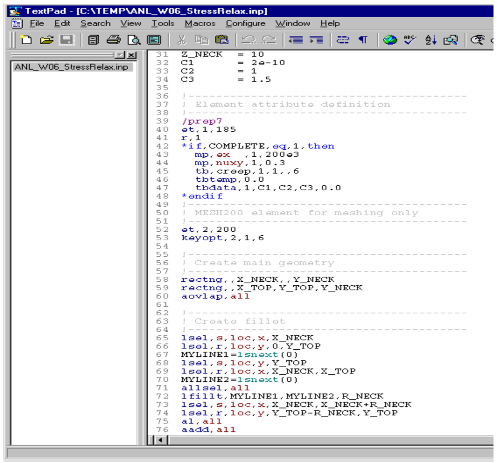
Use of text editors which support syntax highlighting make editing APDL files (input files, macros) much easier. A list of text editors, which support syntax highlighting, is as follows:

- XEmacs (http://www.xemacs.org/)
- NEdit (http://www.nedit.org/)
- VIM (http://www.vim.org/)
- UltraEdit (http://www.ultraedit.com/)
- TextPad (http://www.textpad.com/)

This listing is by no means comprehensive, but above are some editors for UNIX and Windows that the author has used in the past. Some of these editors have ANSYS syntax highlighting contributed by other users, which can be found on the following page: http://ansys.net/ansys/?mycat=tools

The benefits of color syntax highlighting can be seen in the adjacent figure. In this example, regular ANSYS commands are shown in blue, whereas slash-commands (commands preceded by a forward slash, namely graphics settings) and star-commands (commands preceded by an asterisk, usually control statements) are shown purple and light-blue, respectively. Command arguments may also be shown in a separate color as well, such as red. This helps the user distinguish between commands, and it also helps the user verify that commands are typed in correctly (the command does not change color unless it is a 'recognized' ANSYS command). Moreover, comments are shown in light grey, making them less visible, so the user can concentrate on reading the APDL and refer to comments only when necessary. These features help the user become more efficient in reviewing ANSYS input files and macros.

Moreover, many text editors support block indenting by a user-specified number of tabs/spaces and allow saving both DOS and UNIX files (difference of line feed vs. carriage return). The block indents help to make files more readable, and support of DOS/Windows and UNIX files allow for seemless transporting of APDL files across platforms.


TextPad - [C:\TEMP\ANL_W06_StressRelax.inp]
File Edit Search View Tools Macros Configure Window Help
ANL_W06_StressRelax.inp

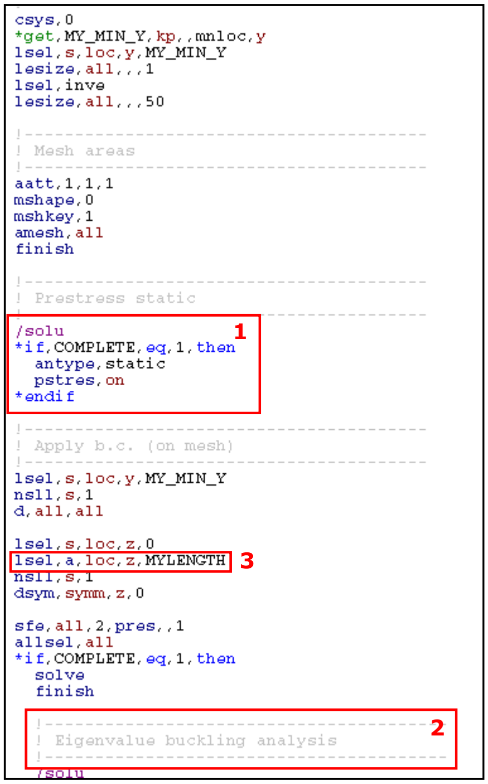
```
Z_NECK   = 10
C1       = 2e-10
C2       = 1
C3       = 1.5

!--------------------------------------------
! Element attribute definition
!--------------------------------------------
/prep7
et,1,185
r,1
*if,COMPLETE,eq,1,then
  mp,ex  ,1,200e3
  mp,nuxy,1,0.3
  tb,creep,1,1,,6
  tbtemp,0.0
  tbdata,1,C1,C2,C3,0.0
*endif
!--------------------------------------------
! MESH200 element for meshing only
!--------------------------------------------
et,2,200
keyopt,2,1,6

!--------------------------------------------
! Create main geometry
!--------------------------------------------
rectng,,X_NECK,,Y_NECK
rectng,,X_TOP,Y_TOP,Y_NECK
aovlap,all

!--------------------------------------------
! Create fillet
!--------------------------------------------
lsel,s,loc,x,X_NECK
lsel,r,loc,y,0,Y_TOP
MYLINE1=lsnext(0)
lsel,s,loc,y,Y_TOP
lsel,r,loc,x,X_NECK,X_TOP
MYLINE2=lsnext(0)
allsel,all
lfillt,MYLINE1,MYLINE2,R_NECK
lsel,s,loc,x,X_NECK,X_NECK+R_NECK
lsel,r,loc,y,Y_TOP-R_NECK,Y_TOP
al,all
aadd,all
```

3. Suggestions on Formatting Style of APDL Macros and Input Files:

Although formatting is sometimes viewed as a cosmetic issue not worth one's consideration, the author finds that using the same format for APDL files within a given group or organization makes sharing files much easier. The following points are ones the author finds quite useful:

1) Use indentation within every set of control statements, such as *IF conditional statements of *DO loops. The number of spaces for an indent should be specified beforehand, such as two spaces. Many text editors can adjust the tab spacing to fit this criterion, convert existing tabs to spaces, and have 'persistent indents.'[1]
2) Precede important and informational sections of code with different 'comment banners.' The author uses a series of minus signs for informational statements, such as
   !-------------------------------------
   whereas a series of asterisks are used for important areas:
   !
3) Use lowercase for all ANSYS commands for easier readability, but use uppercase for APDL parameters and component names. Even if a user copies contents of the ANSYS log file (LGWRITE command or jobname.log file), many text editors allow simple conversion of uppercase to lowercase with a single command.

Despite the fact that some of these points may seem trivial, having a standard formatting scheme is helpful in organizing the APDL file for easier readability. The above suggestions are a few points which the author uses, and it is meant to serve as a simple example of formatting considerations groups may want to consider.



```
csys,0
*get,MY_MIN_Y,kp,,mnloc,y
lsel,s,loc,y,MY_MIN_Y
lesize,all,,,1
lsel,inve
lesize,all,,,50

!----------------------------------------
! Mesh areas
!----------------------------------------
aatt,1,1,1
mshape,0
mshkey,1
amesh,all
finish

!----------------------------------------
! Prestress static
!----------------------------------------
/solu
*if,COMPLETE,eq,1,then
  antype,static
  pstres,on
*endif

!----------------------------------------
! Apply b.c. (on mesh)
!----------------------------------------
lsel,s,loc,y,MY_MIN_Y
nsll,s,1
d,all,all

lsel,s,loc,z,0
lsel,a,loc,z,MYLENGTH
nsll,s,1
dsym,symm,z,0

sfe,all,2,pres,,1
allsel,all
*if,COMPLETE,eq,1,then
  solve
  finish

  !----------------------------------------
  ! Eigenvalue buckling analysis
  !----------------------------------------
  /solu
```

4. Considerations When Writing Macros:

Besides formatting, there are other issues which APDL programmers should agree upon beforehand:

1) Determine how to define parameters in macros and input files. *Parameters in macros are especially important, as multiple users may run any given macro, so parameter name conflicts need to be avoided.* This is not a simple issue since there are many ways in which this can be accomplished, so a group wishing to 'standardize' on APDL coding must agree on how parameters in macros are to be defined. Since ANSYS 6.0 allows for parameter names longer than 32 characters, it is easier to precede any parameter names with the two or three initials of the author's name. For example, "John D. Doe" can use "JDD_" as a prefix for all parameter names used in macros. Although sometimes hard to enforce or tedious for the author, this prevents parameter name conflicts between shared macros or 'nested' macros. A few other points to note:
   - Use of parameter names longer than 8 characters is allowed in ANSYS 6.0 and above, although this would prevent usage of these macros in 5.x versions.
   - The use of a trailing underscore will 'hide' parameters from the users. This is very useful for authors wishing to prevent clutter for the end-user but still retaining parameters for use. *STAT,PRM_ will list these 'hidden' parameters. Note that ANSYS uses parameters beginning with an underscore (also 'hidden' parameters), so this notation should not be used by anyone writing macros.
   - Users can 'clean up' defined parameters with the *DEL command. For arrays, an undocumented option on *DEL will delete arrays 'silently' via *DEL,array,,NOPR. Users cleaning up defined parameters make it much easier for the end-user, although, as noted above, use of trailing underscore parameters ('hidden' parameters) does prevent the end-user from seeing parameters defined but not cleaned up by a macro.

[1] A 'persistent indent' means that, once indented, subsequent carriage returns will keep the number of indents. For example, if a user puts a tab or space in a text editor, then hits the Enter/Return key, the next line will have the same number of indents.

    - ANSYS has ‘local’ parameters which are used only within the macro (they are completely local to the macro, so they will not affect any nested macros). These are scalar parameters defined by AR20 through AR99, providing up to 79 scalar parameters. Note that no such feature currently exists for arrays, tables, or string parameters.
    - Because input files tend to be used for a specific model, the same considerations of macro parameter naming conventions need not be kept for parameters defined by input files. The above considerations should be used for macro parameters (trailing underscore and author’s initials as prefix, such as “JDD_MYPARAMETER_”), and the end-user can use any parameters for input file usage, such as “MYRADIUS.”

2) Comment profusely with “!”. Use of text editors with syntax highlighting will make it easier for those reading macros to ‘ignore’ comments, thus not affecting readability greatly. Well-commented sections help others understand what the input file or macro is doing.
3) After a macro is debugged to ensure that it runs smoothly and without problems, hide the output from the end user to reduce clutter and unnecessary feedback. Use “/NOPR” and “/GOPR” and hide and resume text output, respectively. Use “/COM” or “*MSG” commands to provide the user with informative notes on what the macro is doing.
    - A similar consideration is understanding when “!” and “/COM” should be used for comments. “!” will be comments not printed to screen, so these should be written as frequently as needed to help the APDL programmer understand the inner workings of the input file or macro. “/COM” comments, on the other hand, will be printed to the Output Window/File and should only be used to convey information to the user.
4) Use a standard header which provides a description of the file, defines the required inputs and outputs of the APDL file, and supplies information on the author and date. The required inputs can either be (a) arguments for macros or (b) additional required files for input files. The date information can be used instead of a ‘revision number’ to keep track of any changes to the macro or input file.
    - An additional useful feature but sometimes tedious to implement is a listing of variables defined and released and version(s) of ANSYS the macro was tested on.
    - For macros possibly used outside of the group, a standard ‘disclaimer’ should also be added to prevent any liability for results obtained using one’s macro.
    - This standard header should be included at the beginning of the macro, preferably with “/COM” commands, so it is also echoed to the output window when the user executes the macro, so the user is always aware of these points.

5. Conclusion:

The author hopes that the above discussion provides some ideas for groups or individual users hoping to develop a standardized APDL coding method for their macros and input files. Specifically, (a) readability, (b) definition of parameters, and (c) use of text editors with color syntax highlighting are important in implementing a more consistent framework from which to build complex APDL programs.

Thanks to John Crawford for feedback/comments on this document.

Sheldon Imaoka
ANSYS, Inc.

This document is not being provided in my capacity as an ANSYS employee. I am solely responsible for the content.

Sheldon's ANSYS.NET Tips and Tricks

Sheldon's ANSYS.NET Tips and Tricks is posted about every week at the following URL:

http://ansys.net/ansys/?mycat=search&mytype=Tips&mycategory=Sheldon

Users need to register on ANSYS.NET to download the ANSYS.NET Tips & Tricks.

These tips are written with the latest version of ANSYS in mind (version 6.0, as of the time of this writing). Please remember that, with each new release of ANSYS, new features and methods may be introduced, so please refer to the online help as well as your local ANSYS support distributor to verify that these tips are the most efficient way of doing things.

ANSYS Training

ANSYS, Inc. as well as the ANSYS Support Distributors (ASDs) provide training classes in many different areas of ANSYS, ranging from Introductory classes to CFD to Structural Nonlinearities to Electromagnetics. Information on training classes and schedules at ANSYS, Inc. can be found at:

http://www.ansys.com/services/training/index.htm

New classes current offered include *High-Frequency Electromagnetics* and *Advanced Structural Nonlinearities*. Please contact ANSYS, Inc. or your local ASD for their training schedules and offerings.

XANSYS Mailing List

The XANSYS mailing list has more than 2300 subscribers (as of 9/25/01) with about 40 postings per day. This is a forum for exchanging ideas, providing/receiving assistance from other users, and general discussions related to ANSYS. (Note that it is recommended to contact your local ASD for issues related to technical support) You can sign up by visiting the following URL:

http://groups.yahoo.com/group/xansys

Otherwise, you can also subscribe/unsubscribe by sending an email to the following address:

Post message:	xansys@yahoogroups.com
Subscribe:	xansys-subscribe@yahoogroups.com
Unsubscribe:	xansys-unsubscribe@yahoogroups.com
List owner:	xansys-owner@yahoogroups.com

Because the amount of emails is very large, you can also subscribe in "digest mode" or access the postings via a web browser instead:

Digest mode:	xansys-digest@yahoogroups.com
Web-based:	xansys-nomail@yahoogroups.com http://groups.yahoo.com/group/xansys/messages