Chapter 2: Text User Interface (TUI)

In addition to the graphical user interface described in Graphical User Interface (GUI) (p. 1), the user interface in ANSYS Fluent includes a textual command line interface.

The text interface (TUI) uses, and is written in, a dialect of Lisp called Scheme. Users familiar with Scheme will be able to use the interpretive capabilities of the interface to create customized commands.

2.1. Text Menu System

The text menu system provides a hierarchical interface to the program's underlying procedural interface. Because it is text based, you can easily manipulate its operation with standard text-based tools: input can be saved in files, modified with text editors, and read back in to be executed. Because the text menu system is tightly integrated with the Scheme extension language, it can easily be programmed to provide sophisticated control and customized functionality.

The menu system structure is similar to the directory tree structure of Linux operating systems. When you first start ANSYS Fluent, you are in the "root" menu and the menu prompt is a greater-than symbol.

```
>
```

To generate a listing of the submenus and commands in the current menu, simply press **Enter**. Note that the available submenus and commands will depend on whether you are in meshing mode or solution mode. The submenus and commands that are available from the root menu of the solution mode are as follows:

```
>Enter
adapt/                  mesh/                   surface/
define/                 parallel/               switch-to-meshing-mode
display/                plot/                   views/
exit                    report/
file/                   solve/
```

Note that `switch-to-meshing-mode` is only available for 3D sessions, before you have read a mesh or a case file.

By convention, submenu names end with a `/` to differentiate them from menu commands. To execute a command, just type its name (or an abbreviation). Similarly, to move down into a submenu, enter its name or an abbreviation. When you move into the submenu, the prompt will change to reflect the current menu name.

```
>display

/display> set

/display/set>
```

Release 15.0 - © SAS IP, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

To move back to the previously occupied menu, type `q` or `quit` at the prompt.

```
/display/set> q

/display>
```

You can move directly to a menu by giving its full pathname.

```
/display> /file

/display//file>
```

In the above example, control was passed from `/display` to `/file` without stopping in the root menu. Therefore, when you quit from the `/file` menu, control will be passed directly back to `/display`.

```
/display//file> q

/display>
```

Furthermore, if you execute a command without stopping in any of the menus along the way, control will again be returned to the menu from which you invoked the command.

```
/display> /file start-journal jrnl

 Opening input journal to file "jrnl".

/display>
```

The text menu system provides on-line help for menu commands. The text menu on-line help system is described in Using the Text Interface Help System (p. 38).

To edit the current command, you can position the cursor with the left and right arrow keys, delete with the **Backspace** key, and insert text simply by typing.

For additional information, see the following sections:

2.1.1. Command Abbreviation

To select a menu command, you do not need to type the entire name; you can type an abbreviation that matches the command. The rules for matching a command are as follows: A command name consists of phrases separated by hyphens. A command is matched by matching an initial sequence of its phrases. Matching of hyphens is optional. A phrase is matched by matching an initial sequence of its characters. A character is matched by typing that character.

If an abbreviation matches more than one command, then the command with the greatest number of matched phrases is chosen. If more than one command has the same number of matched phrases, then the first command to appear in the menu is chosen.

For example, each of the following will match the command `set-ambient-color`: `set- ambient-color`, `s-a-c`, `sac`, and `sa`. When abbreviating commands, sometimes your abbreviation will match more than one command. In such cases, the first command is selected. Occasionally, there is an anomaly such as `lint` not matching `lighting-interpolation` because the `li` gets absorbed in `lights-on?` and then the `nt` does not match `interpolation`. This can be resolved by choosing a different abbreviation, such as `liin`, or `l-int`.

Release 15.0 - © SAS IP, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2.1.2. Command Line History

You can use the up and down arrow keys on your keyboard to go through recently used commands that are stored in history. By default, command-history will store only the last ten commands. This can be changed (for example to 15) by using the following command:

```
> (set! *cmd-history-length* 15)
```

Important

Command-history is not available if the ANSYS Fluent application is started with -g options (see Command Line Startup Options in the Getting Started Guide).

Important

The user inputs supplied as the arguments of the TUI command or alias will not be saved in history. By way of illustration, consider the following entry in the TUI:

```
> rc new_file.cas
```

Important

In history, only `rc` (an alias for `read-case`) will be saved, since `new_file.cas` is a user input to the alias-function.

Commands recalled from history can be edited or corrected using the **Backspace** key and the left and right arrow keys.

2.1.3. Scheme Evaluation

If you enter an open parenthesis, `(`, at the menu prompt, then that parenthesis and all characters up to and including the matching closing parenthesis are passed to Scheme to be evaluated, and the result of evaluating the expression is displayed.

```
> (define a 1)

a

> (+ a 2 3 4)

10
```

2.1.4. Aliases

Command aliases can be defined within the menu system. As with the Linux `csh` shell, aliases take precedence over command execution. The following aliases are predefined in Cortex: `error`, `pwd`, `chdir`, `ls`, `.`, and `alias`.

`error`
displays the Scheme object that was the "irritant" in the most recent Scheme error interrupt.

`pwd`
prints the working directory in which all file operations will take place.

Release 15.0 - © SAS IP, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

chdir
will change the working directory.

ls
lists the files in the working directory.

. (period)
prompts you for a journal file name and then run the journal file you have specified.

alias
displays the list of symbols currently aliased.

2.2. Text Prompt System

Commands require various arguments, including numbers, filenames, yes/no responses, character strings, and lists. A uniform interface to this input is provided by the text prompt system. A prompt consists of a prompt string, followed by an optional units string enclosed in parentheses, followed by a default value enclosed in square brackets. The following shows some examples of prompts:

```
filled-mesh? [no] Enter

shrink-factor [0.1] Enter

line-weight [1] Enter

title [""]Enter
```

The default value for a prompt is accepted by pressing **Enter** on the keyboard or typing a `,` (comma).

> **Important**
>
> Note that a comma is not a separator. It is a separate token that indicates a default value. The sequence "`1,2`" results in three values; the number 1 for the first prompt, the default value for the second prompt, and the number 2 for the third prompt.

A short help message can be displayed at any prompt by entering a `?`. (See Using the Text Interface Help System (p. 38).)

To abort a prompt sequence, simply press **Ctrl**+**c**.

For additional information, see the following sections:

2.2.1. Numbers

The most common prompt type is a number. Numbers can be either integers or reals. Valid numbers are, for example, `16`, `-2.4`, `.9E5`, and `+1E-5`. Integers can also be specified in binary, octal, and hexadecimal form. The decimal integer 31 can be entered as `31`, `#b11111`, `#o37`, or `#x1f`. In Scheme,

Release 15.0 - © SAS IP, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

integers are a subset of reals, so you do not need a decimal point to indicate that a number is real; `2` is just as much a real as `2.0`. If you enter a real number at an integer prompt, any fractional part will simply be truncated; `1.9` will become `1`.

2.2.2. Booleans

Some prompts require a yes-or-no response. A yes/no prompt will accept either `yes` or `y` for a positive response, and `no` or `n` for a negative response. Yes/no prompts are used for confirming potentially dangerous actions such as overwriting an existing file, exiting without saving case, data, mesh, etc.

Some prompts require actual Scheme boolean values (true or false). These are entered with the Scheme symbols for true and false, `#t` and `#f`.

2.2.3. Strings

Character strings are entered in double quotes, for example, `"red"`. Plot titles and plot legend titles are examples of character strings. Character strings can include any characters, including blank spaces and punctuation.

2.2.4. Symbols

Symbols are entered *without* quotes. Zone names, surface names, and material names are examples of symbols. Symbols must start with an alphabetical character (that is, a letter), and cannot include any blank spaces or commas.

2.2.5. Filenames

Filenames are actually just character strings. For convenience, filename prompts do not require the string to be surrounded with double quotes. If, for some exceptional reason, a filename contains an embedded space character, then the name must be surrounded with double quotes.

One consequence of this convenience is that filename prompts do not evaluate the response. For example, the sequence

```
> (define fn "valve.ps")

fn

> hc fn
```

will end up writing a picture file with the name `fn`, not `valve.ps`. Since the filename prompt did not evaluate the response, `fn` did not get a chance to evaluate `"valve.ps"` as it would for most other prompts.

2.2.6. Lists

Some functions in ANSYS Fluent require a "list" of objects such as numbers, strings, booleans, etc. A list is a Scheme object that is simply a sequence of objects terminated by the empty list, `'()`. Lists are prompted for an element at a time, and the end of the list is signaled by entering an empty list. This terminating list forms the tail of the prompted list, and can either be empty or can contain values. For convenience, the empty list can be entered as `()` as well as the standard form `'()`. Normally, list prompts save the previous argument list as the default. To modify the list, overwrite the desired elements and terminate the process with an empty list. For example,

Release 15.0 - © SAS IP, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
element(1) [()] 1
element(2) [()] 10
element(3) [()] 100
element(4) [()] Enter
```

creates a list of three numbers: 1, 10, and 100. Subsequently,

```
element(1) [1] Enter
element(2) [10] Enter
element(3) [100] Enter
element(4) [()] 1000
element(5) [()] Enter
```

adds a fourth element. Then

```
element(1) [1] Enter
element(2) [10] Enter
element(3) [100] ()
```

leaves only 1 and 10 in the list. Subsequently entering

```
element(1) [1] ,,'(11 12 13)
```

creates a five element list: 1, 10, 11, 12, and 13. Finally, a single empty list removes all elements

```
element(1) [1] ()
```

A different type of list, namely, a “list-of-scalars” contains pick menu items (and not list items) for which a selection has to be made from listed quantities, which are available at the **Enter** prompt. Hence, a list-of-scalars cannot be entered as a list.

An example of a “list-of-scalars” consists of the following:

Release 15.0 - © SAS IP, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
ASCII scalar(1)>
abs-angular-coordinate  cell-warp              radial-velocity
absolute-pressure       cell-weight            rel-tangential-velocity
adaption-curvature      cell-zone              rel-total-pressure
adaption-function       custom-function-0      rel-velocity-magnitude
adaption-iso-value      density                relative-velocity-angle
adaption-space-gradient density-all            relative-x-velocity
angular-coordinate      dp-dx                  relative-y-velocity
axial-coordinate        dp-dy                  relative-z-velocity
axial-velocity          dp-dz                  tangential-velocity
boundary-cell-dist      dynamic-pressure       total-pressure
boundary-normal-dist    existing-value         velocity-angle
boundary-volume-dist    face-area-magnitude    velocity-magnitude
cell-children           face-handedness        x-coordinate
cell-element-type       face-squish-index      x-face-area
cell-equiangle-skew     interface-wall-zone    x-velocity
cell-equivolume-skew    mass-imbalance         x-velocity-residual
cell-partition          mesh-x-velocity        y-coordinate
cell-refine-level       mesh-y-velocity        y-face-area
cell-squish-index       mesh-z-velocity        y-velocity
cell-surface-area       partition-neighbors    y-velocity-residual
cell-type               pressure               z-coordinate
cell-volume             pressure-coefficient   z-face-area
cell-volume-change      pressure-residual      z-velocity
cell-wall-distance      radial-coordinate      z-velocity-residual

ASCII scalar(1)> pressure

ASCII scalar(2)> z-coordinate

ASCII scalar(3)> cell-volume

ASCII scalar(4)> dynamic-pressure velocity-magnitude x-face-area

ASCII scalar(7)>
```

2.2.7. Evaluation

All responses to prompts (except filenames, see above) are evaluated by the Scheme interpreter before they are used. You can therefore enter any valid Scheme expression as the response to a prompt. For example, to enter a unit vector with one component equal to $1/3$ (without using your calculator),

```
/foo> set-xy
x-component [1.0] (/ 1 3)

y-component [0.0] (sqrt (/ 8 9))
```

or, you could first define a utility function to compute the second component of a unit vector,

```
> (define (unit-y x) (sqrt (- 1.0 (* x x))))

unit-y
/foo> set-xy
```

Release 15.0 - © SAS IP, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
x-component [1.0] (/ 1 3)

y-component [0.0] (unit-y (/ 1 3))
```

2.2.8. Default Value Binding

The default value at any prompt is bound to the Scheme symbol " _" (underscore) so that the default value can form part of a Scheme expression. For example, if you want to decrease a default value so that it is one-third of the original value, you could enter

```
shrink-factor [0.8] (/ _ 3)
```

2.3. Interrupts

The execution of the code can be halted by pressing the **Ctrl**+**c**, at which time the present operation stops at the next recoverable location.

2.4. System Commands

The way you execute system commands with the `!` (bang) shell escape character will be slightly different for Linux and Windows systems.

For additional information, see the following sections:
2.4.1. System Commands for Linux-based Operating Systems
2.4.2. System Commands for Windows Operating Systems

2.4.1. System Commands for Linux-based Operating Systems

If you are running ANSYS Fluent under a Linux-based operating system, all characters following the `!` up to the next newline character will be executed in a subshell. Any further input related to these system commands must be entered in the window in which you started the program, and any screen output will also appear in that window. (Note that if you started ANSYS Fluent remotely, this input and output will be in the window in which you started Cortex.)

```
> !rm junk.*

> !vi script.rp
```

`!pwd` and `!ls` will execute the Linux commands in the directory in which Cortex was started. The screen output will appear in the window in which you started ANSYS Fluent, unless you started it remotely, in which case the output will appear in the window in which you started Cortex. (Note that `!cd` executes in a subshell, so it will not change the working directory either for ANSYS Fluent or for Cortex, and is therefore not useful.) Typing `cd` with no arguments will move you to your home directory in the console.

ANSYS Fluent includes three system command aliases (`pwd`, `ls`, and `chdir`) that will be executed in your working directory with output displayed in the ANSYS Fluent console. Note that these aliases will invoke the corresponding Linux commands with respect to the parent directory of the case file. For example, `pwd` prints the parent directory of the case file in the ANSYS Fluent console, while `!pwd` prints the directory from which you started ANSYS Fluent in the Linux shell window where you started ANSYS Fluent.

Several examples of system commands entered in the console are shown below. The screen output that will appear in the window in which ANSYS Fluent was started (or, if you started the program remotely, in the window in which Cortex was started) follows the examples.

Release 15.0 - © SAS IP, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Example input (in the ANSYS Fluent console):

```
> !pwd

> !ls valve*.*
```

Example output (in the window in which ANSYS Fluent— or Cortex, if you started the program remotely—was started):

```
/home/cfd/run/valve
 valve1.cas  valve1.msh  valve2.cas  valve2.msh
```

2.4.2. System Commands for Windows Operating Systems

If you are running ANSYS Fluent under a Windows operating system, all characters following the `!` up to the next newline character will be executed. The results of a command will appear in the ANSYS Fluent console, or in a separate window if the command starts an external program, such as Notepad.

```
> !del junk.*

> !notepad script.rp
```

`!cd` and `!dir` will execute the DOS commands and the screen output will appear in the ANSYS Fluent console. The `!cd` command with no argument will display the current working directory in the ANSYS Fluent console.

Several examples of system commands entered in the console are shown below.

Example input (in boxes) and output (in the ANSYS Fluent console):

```
  > !cd

p:/cfd/run/valve
 > !dir valve*.*/w

     Volume in drive P is users
     Volume Serial Number is 1234-5678
     Directory of p:/cfd/run/valve
     valve1.cas      valve1.msh      valve2.cas      valve2.msh
        4 File(s)              621,183 bytes
        0 Dir(s)          1,830,088,704 bytes free
```

2.5. Text Menu Input from Character Strings

Often, when writing a Scheme extension function for ANSYS Fluent, it is convenient to be able to include menu commands in the function. This can be done with `ti-menu-load-string`. For example, to open graphics window 2, use

```
(ti-menu-load-string "di ow 2")
```

A Scheme loop that will open windows 1 and 2 and display the front view of the mesh in window 1 and the back view in window 2 is given by

```
(for-each
 (lambda (window view)
   (ti-menu-load-string (format #f "di ow ~a gr view rv ~a"
 window view)))
 '(1 2)
 '(front back))
```

This loop makes use of the `format` function to construct the string used by `menu-load-string`. This simple loop could also be written without using menu commands at all, but you need to know the Scheme functions that get executed by the menu commands to do it:

Release 15.0 - © SAS IP, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
(for-each
 (lambda (window view)
   (cx-open-window window)
   (display-mesh)
   (cx-restore-view view))
 '(1 2) '(front back))
```

String input can also provide an easy way to create aliases within ANSYS Fluent. For example, to create an alias that will display the mesh, you could type the following:

```
(alias 'dg (lambda () (ti-menu-load-string "/di gr")))
```

Then any time you enter `dg` from anywhere in the menu hierarchy, the mesh will be drawn in the active window.

Important

`ti-menu-load-string` evaluates the string argument in the top level menu. It ignores any menu you may be in when you invoke `ti-menu-load-string`.

As a result, the command

```
(ti-menu-load-string "open-window 2 gr") ; incorrect usage
```

will not work even if you type it from within the `display/` menu—the string itself must cause control to enter the `display/` menu, as in

```
(ti-menu-load-string "display open-window 2 mesh")
```

2.6. Using the Text Interface Help System

The text user interface provides context-sensitive on-line help. Within the text menu system, you can obtain a brief description of each of the commands by entering a `?` followed by the command in question.

Example:

```
> ?dis
display/: Enter the display menu.
```

You can also enter a lone `?` to enter "help mode." In this mode, you need only enter the command or menu name to display the help message. To exit the help mode type `q` or `quit` as for a normal menu.

Example:

```
> ?

[help-mode]> di

display/: Enter the display menu.

 [help-mode]> pwd

 pwd: #[alias]
 (LAMBDA ()
   (BEGIN
   (SET! pwd-cmd ((LAMBDA n
  n) 'system (IF (cx-send '(unix?))
    "pwd"
    "cd")))
  (cx-send pwd-cmd)))
```


Release 15.0 - © SAS IP, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
[help-mode]> q
```

To access the help, type a `?` at the prompt when you are prompted for information.

Example:

```
> display/annotate Annotation text [""] ?
 Annotation text [""]
```

Release 15.0 - © SAS IP, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.