ADB04

COLLABORATORS

	TITLE : ADB04	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 27, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ADB04

1.1 AmigaDOS For Beginners - Part 4 - Using Shell & CLI

AMIGADOS FOR BEGINNERS

BY FRANK BUNTON

PART 4 - USING THE SHELL AND CLI WINDOWS

CONTENTS OF THIS ARTICLE

This article has grown rather large but there are some important topics in it which are either essential to using a Shell or CLI window or that make using the window a lot easier. If you find this rather hard going at this stage then please make a note to come back to it after you have read further into the articles in this set. You will, in all likelihood, understand these topics much better after reading them again at a later time.

The topics in this article are:-

Editing Command Lines (Shell)
Shell Only

Editing Command Lines (Old CLI)
Old CLI v1.3 and earlier

Moving Around Command Lines
Shell Only

Old Command History
Shell Only

Getting a blank Line
Shell Only

Searching For Old Commands
Shell Only

Copying and Pasting
Version 2.0 Onwards

Getting A Command Template
All versions CLI and Shell

Redirecting Command Input/Output
All versions CLI and Shell

Redirecting Keyboard Output
All versions CLI and Shell

Pausing Command Output
All versions CLI and Shell

Aborting Command Output
All versions CLI and Shell

Starting a Program From Shell/CLI
All versions CLI and Shell

Enclosing One Command In Another
Version 2.0 Onwards

=== End of File ===

1.2 Part 4 - Using Shell & CLI - Symbols Used in AmigaDOS

SYMBOLS USED IN AMIGADOS (All Versions CLI and Shell)

There are certain symbols, or characters, that have special meaning within AmigaDOS. They are:-

Mathematical Operator Symbols Click here for more information.

```
  *   Multiplication Operator
                *  also miscellaneous symbol
                  /   Division operator
                /  also miscellaneous symbol
                  +   Plus operator
                +  also miscellaneous symbol
                  -   Minus operator
                -  also pattern matching symbol
                  -   Negation
  &   Logical AND
  |   Logical OR
                |  also pattern matching symbol
                  ~   Logical NOT
                ~  also pattern matching symbol
                Wildcard Symbols  Click here for more information

  #   Wildcard
  ?   Wildcard

Pattern Matching Symbols  Click here for more information.

  |   Pattern separator
                |  also mathematical symbol
                  ~   Pattern negation
                ~  also mathematical symbol
                  ()  Pattern enclosers
  []  Pattern enclosers
  %   Null string indicator
  ’   Wildcard selector
  -   Range indicator
                -  also mathematical symbol
                Redirection Symbols

  >
                Redirect Command Output
                  >>
                Redirect & ADD to a File
                  <
                Redirect Command Input
                Miscellaneous Symbols

  :   Path Separator
  :   Move to root directory In Shell/CLI
  /   Path Separator
                /  also mathematical symbol
                  /   Move CD up one level In Shell/CLI
  ""  String delimiters
  ""  Make Assignment to CD
  *
                Refer to Current Shell/CLI window

                *  also mathematical symbol
                  *   Special Code Indicator
                *  also mathematical symbol
                  `   Enclose an AmigaDOS command in String
  +   Add command to RUN command line
```

```
                +  also mathematical symbol
                  ;   Comment indicator in scripts
  .   Comment indicator in scripts
  .   DOT command indicator in scripts


=== End of File ===
```

1.3 Part 4 - Using Shell & CLI - Opening a CLI or Shell Window

```
                OPENING A CLI OR SHELL WINDOW (All Versions CLI and Shell)

Boot up with your workbench disk and double click on its disk icon. If
the SHELL or CLI icon is not in the disk window then open the SYSTEM drawer
(double click on its icon) and look for it there.

Double click on the CLI or SHELL icon and you will get a new window. You
can move it about and/or resize it the same way as any other window. If
```

you have v2+ it should have a close gadget, although it can be customised not to have one (See article Window Specifications). (See later for information on how close a CLI/Shell window without a close gadget)

If you are using v2+ you can also open a new Shell/CLI window by selecting the Workbench menu item "Workbench - Execute Command" and entering in the window that appears "NEWSHELL".

I will discuss the NEWSHELL and NEWCLI commands in a later article but both these commands can be used to create a new Shell or CLI from within an existing Shell or CLI window.

Special Note v1.2

If you have v1.2 and you can't find the CLI icon anywhere, then double click on the Preferences Icon and look at the middle left side of the preferences screen. There is an item there that says "CLI ON OFF" (WB v1.3 has dispensed with this item). Select "ON" (click in that box), select the "SAVE" box (middle right side of screen), then close the System Drawer window, then reopen it. Your CLI icon should now be present. If this preference item is set to "CLI OFF" then the CLI icon will not appear in system window. If the CLI icon is taken out of the system window and put into the disk's main window, then this preferences item will not have any effect on the CLI icon.

=== End of Text ===

1.4 Part 4 - Using Shell & CLI - Closing the CLI/Shell window

CLOSING THE CLI/SHELL WINDOW (All Versions CLI and Shell)

To get rid of a CLI window, just enter:-

```
  > ENDCLI
```

Or, if you have v2+, just click on the "close" gadget in the top left corner of the window (if it is there).

If the window doesn't close, then it could be that some program that was started off from that CLI process is still working away in the background even though you cannot see it. The window will stay open until that other program stops.

I will discuss the ENDCLI and ENDSHELL commands in a later article.
The above comments are just to get you under way.

=== End of Text ===

1.5 Part 4 - Using Shell & CLI - The Shell/CLI Prompt

THE SHELL/CLI PROMPT (All Versions CLI and Shell)

If you are using CLI (not Shell) in v1.3 or earlier you will see in the window:-

```
  1.>
```

If using SHELL in v1.3 or v2.0 onwards you will see something like:-

```
  1.SYS:>
  1.Workbench1.3:>
  1.Workbench2.0:>
```

This is called the "Prompt". It is put there automatically (but it can be customised to your own preference) and any typing you do is to the right of the ">".

The "SYS:" or "Workbenchx.x:" etc. indicates the name of the current directory.

The number 1 tells you that this is CLI process number 1. You will often see the term "CLI Process". A "process" is simply a task that can communicate with AmigaDOS. Thus a CLI or Shell window is a "process" and the 1 says it is process number 1.

If you double click on the CLI or Shell icon again, another window will open and it will show, for example:-

```
  2.Workbenchx.x:>
```

It is CLI window (or process) number 2.

Note - if the first window you open shows a number higher than 1 it means the some other program has been launched, i.e. started up, using a CLI process, maybe while the disk was being booted. When a program is started from CLI (including being started from the startup-sequence on boot up) then it is assigned a CLI process number.

Note - If your window shows, say, process 2 and you open a new window and it shows process 1 then that means that the previous process 1 has been shut down after process 2 was started. When looking for process numbers to allocate, the system takes the lowest number available even if there are other processes running that have higher numbers.

=== End of Text ===

1.6 Part 4 - Using Shell & CLI - Explanation of Key Presses

EXPLANATION OF KEY PRESSES (All Versions CLI and Shell)

In this series of articles, I indicate to press certain key combinations with things like:-

```
   CTRL-K
```

```
   SHIFT-UP

The "CTRL" and "SHIFT" represent the qualifier keys "Ctrl" and "Shift".
They are qualifier keys as they alter, or "qualify" the way the other
keys act. The most common qualifier key is the "SHIFT" key which changes
the alphabetic keys from lower case to upper case.

Where you see something like CTRL-K it means to:-

- hold down the qualifier key (CTRL in this case),
- press the key shown after it (the "K" in this case)
  while still holding down the qualifier key,
- then release both keys

The cursor keys might be represented with things like this in various
texts:-

  UP     or  CursUP    - the UP cursor key
  DOWN   or  CursDOWN  - the DOWN cursor key
  RIGHT  or  CursRIGHT - the RIGHT cursor key
  LEFT   or  CursLEFT  - the LEFT cursor key

=== End of Text ===
```

1.7 Part 4 - Using Shell & CLI - Entering Command Lines

```
                ENTERING COMMAND LINES (All Versions CLI and Shell)

In the article AMIGADOS AND SHELL/CLI we talked about CLI being a "Command
Line Interface" in which you could type lines of AmigaDOS commands.

To enter a command line in a Shell or CLI window you simply type in the
whole line then press the return key. To try it out, type in this line
then press return:-

  > TYPE s:startup-sequence

When you see this sort of thing do not type in the ">" as it represents
the
                prompt
                 that you see in your CLI or Shell window. You just type in
the:-

   TYPE s:startup-sequence

I will not tell you to press the return key. Whenever you see something
like:-

  > TYPE s:startup-sequence

you must assume that the return key must be pressed.

AmigaDOS does not consider the command to have been completed until you
press the return key. The return key is your signal to AmigaDOS that you
have finished doing your thing and that it is AmigaDOS's turn to do
```

something, i.e. to accept the command line and act on it.

You will sometimes find that the command line you type takes up more than one line in the CLI window. This does not matter at all as AmigaDOS considers the command line to be everything typed in from the prompt up until the return key is pressed regardless of whether the command line takes up only a few characters or a number of window lines.

=== End of Text ===

1.8 Part 4 - Using Shell & CLI - Editing command Lines - Old CLI Only

EDITING COMMAND LINES (Old CLI only)

If you are using the old CLI window from v1.x then you will find that you cannot make corrections to the command lines that you type except by using the back space key to delete everything up to the part you want to correct, then retyping the part of the line deleted. This can be very tedious. Nor can you retrieve an old command to reuse it or alter it slightly.

=== End of Text ===

1.9 Part 4 - Using Shell & CLI - Editing command Lines - Shell Only

EDITING COMMAND LINES (SHELL only)

You can use the left and right cursor keys to move the cursor to any character in the command line. Those letters that the cursor passes over will not be deleted. Corrections can thus be made at any point of the line. This can save a lot of typing. Press RETURN at any point on the line to activate the full command line.

There are a number of ways of deleting characters on the command line:-

```
  Delete Key - Deletes one character at the cursor position
  Backspace  - Deletes one character to the left of the cursor position
  CTRL-K     - Deletes all characters from cursor to end of line
  CTRL-Y     - Replace characters removed with CTRL-K (v2+ only)
  CTRL-U     - Deletes all characters from one to left of cursor to start
                 of line
  CTRL-X     - Deletes the entire line but leaves the position within
                 the
                command history
                 unaltered
  CTRL-B     - Deletes the entire line and moves the position within
                 the
                command history
                 to after the last command in the
                 history
  CTRL-W     - Acts only on the WORD that the cursor is on.
```

```
                It deletes that part of the word from the character to the
                left of the cursor back to the start of the word
                ( v2+ only)

=== End of Text ===
```

1.10 Part 4 - Using Shell & CLI - Moving Around Command Lines

```
               MOVING AROUND COMMAND LINES (SHELL Only)

  SHIFT-LEFT  - move cursor to start of command line
  SHIFT-RIGHT - move cursor to end of command line

Version 1.3 note:-

  CTRL-W      - moves the cursor 8 spaces to the right (v1.3 only)

  (Version 2+ note that
               CTRL-W
                deletes the part of the word to the left of
   the cursor

=== End of Text ===
```

1.11 Part 4 - Using Shell & CLI - Old command History

```
               OLD COMMAND HISTORY (SHELL Only)

The system keeps old commands in memory so that you can reuse them later.
This is VERY useful if you are using a long command again, or you have
a command that is very similar to a previous one.

The UP cursor key will display the old commands, cycling through them
upwards towards the top of the list until you reach the first command
used. Finally, a
               blank line
                will be displayed (unless the buffer is full
- see 3 paragraphs below). Once you have gone part or all of the way up the
list, you can use the DOWN cursor key to cycle downwards through the list
to the most recent command at the bottom of the list until you get a blank
line (unless the buffer is full).

When you find the one you want,
               edit it
                if necessary then press return.

You can save an lot of typing in this way.

The memory "buffer" used to hold this command history is limited in size
(2kb). Once the buffer is full the system will start to discard the earliest
```

```
commands at the start, or top, of the list to make room for the new commands
at the end, or bottom, of the list. The number of commands it can hold
will depend on the length of the commands.

In v1.x, using the UP or DOWN cursor keys when the buffer is full will
never give you a
                blank line
                . The complete list will continue to be cycled
ad infinitum.

=== End of Text ===
```

1.12 Part 4 - Using Shell & CLI - Getting a Blank Line

```
                GETTING A BLANK LINE (SHELL Only)

After cycling upwards through old commands, you may want a blank line
to enter a completely new command (especially if the history buffer is
full!).

CTRL-X - will blank the line but leave you at the same point in the

                command history
                . However, any new command entered will go
         to the end of the list.

CTRL-B - (or SHIFT-DOWN) will take you to a blank line at the end of
         the command history.

Of course, with V2+, you can just keep holding down the DOWN or UP cursor
key until you reach the start or end of the
                command history
                .

=== End of Text ===
```

1.13 Part 4 - Using Shell & CLI - Searching For Old Commands

```
SEARCHING FOR OLD COMMANDS (SHELL Only)

If you type in the first few characters of a command then press SHIFT-UP
you will be taken back to the last command line that had those characters
at its start.

For example, enter:-

  > DIR SHIFT-UP

and you will be taken to the last command that started with "DIR".
```

This is one case where you do not press the return key!!

(Don't type in "SHIFT-UP" - don't forget that it means hold down SHIFT
and press the cursor up key. Thus the above means type in DIR then press
the SHIFT and the Cursor Up keys.)

In v2+ only, pressing SHIFT-UP again will continue the search upwards
through the list for items starting with "DIR". For example, it might
find:-

```
  DIR df1:
  DIR df0:
  DIR df0:devs/printers
```

In v1.3, it works a bit differently. If you enter:-

```
  > DIR SHIFT-UP
```

then, if it finds:-

```
  DIR df1:
```

and you continue to press SHIFT-UP it will look for other occurrences
of "DIR df1:" and not for the originally entered "DIR" as v2+ does.

=== End of Text ===

1.14 Part 4 - Using Shell & CLI - Copying and Pasting

COPYING AND PASTING (v2+ Only)

Copying and pasting in Shell windows was introduced with v2+.

Text in any Shell window (if more than one is open) or any Text Editor
window using a console window (i.e. a window similar to the Shell window)
can be copied and pasted into the same or any other window. For example,
copy from one shell and paste into another, or copy from one script file
being edited with the "ED" program and paste into another script file
or into a Shell window.

You can even copy and paste between a Shell window and some programs such
as certain word processors provided that those programs use the same system
for storing the copied or cut text.

Point to the start of the text you wish to copy, hold down the left mouse
button, then drag the pointer to the end of the text to be copied, then
let go the button. The selected text should become highlighted.

Now press RightAMIGA-C. The highlighting will disappear and the text will
be written to a buffer for later use.

Finally, click in the window in which the text is to be pasted and, if
necessary, position the cursor to the desired insertion point. Now press
RightAMIGA-V and the text will appear again.

=== End of Text ===

1.15 Part 4 - Using Shell & CLI - Getting A Command Template

GETTING A COMMAND TEMPLATE

At this stage (if you are a beginner and you are still in the process of reading through the the articles in their proper order) you should not worry too much about Command Templates.

However, if you want to have a quick reminder of how to use a command by looking at its template, you can enter:-

```
  > CommandName ?  (press return)
```

and the template will be displayed with the cursor waiting at the end of the line waiting for you input. For example, if you entered:-

```
  > TYPE ?
```

then the display would be something like:-

```
  FROM/A/M,TO/K,OPT/K,HEX/S,NUMBER/S: #
```

Note - The # represents the Shell/CLI cursor and is not part of the template.

Note - You may get a different template depending on your AmigaDOS version.

Note - If you want to know what all the symbols in the template mean then you can click here.

If you press return without any more input the command will carry on as if you entered the command without arguments. This may give an error message if the command requires at least one argument.

If you wish to enter an argument, then you can do so at the cursor position without reentering the command name. For example:-

```
  > TYPE ?                                              (press return)
  FROM/A/M,TO/K,OPT/K,HEX/S,NUMBER/S: S:startup-sequence    (press return)
```

The TYPE command will now display the contents of the file "S:startup-sequence" in the window.

=== End of Text ===

1.16 Part 4 - Using Shell & CLI - Redirecting Command Input/Output

REDIRECTING COMMAND INPUT/OUTPUT (All Versions of SHELL & CLI)

The symbols > >> < can be used to:-

> Redirect Command Output
CLI and Shell All Versions

>> Redirect to ADD to an Existing File
Shell Only Versions 1.3 & Later

< Redirect Command Input
CLI and Shell All Versions

=== End of Text ===

1.17 Part 4 - Using Shell & CLI - Redirecting Command Output

Redirecting Command Output

Output FROM a command (usually to the process window) can be redirected elsewhere with the symbol '>'.

As an example of output redirection, to copy the contents of the current directory to a file on df1: you would enter:-

```
DIR > df1:filename
```

Instead of the output from DIR appearing in the window, they will be written to the file "df1:filename". You can verify this by entering:-

```
TYPE df1:filename
```

and you will see the directory contents that have been written to "df1:filename".

In v2+, with most commands the redirection can come anywhere in the command line. For example:-

```
DIR Sys: >DF1:filename OPT A
```

```
DIR Sys: OPT A >DF1:filename
```

However, it is best to put it immediately after the command and before any options or arguments. This will ensure that the command line will work with commands that cannot cope with the above type of examples, and with earlier version of AmigaDOS. Thus use this command line instead of the above two examples:-

```
DIR >DF1:filename Sys: OPT A
```

To send the directory of df1: to the printer you could enter:-

```
  DIR > prt: df1:

You can also redirect output from other commands. For example:-

  TYPE > prt: filename

will type the contents of "filename" to the printer instead of to the
output window.


=== End of Text ===
```

1.18 Part 4 - Using Shell & CLI - Redirecting Output To Add

```
Redirection to Add to a File (Shell Only)

If a file already exists, you can add to it with with a double redirection
sign, as in:-

  TYPE >> destination-file FROM source-file

For example:-

  TYPE >ram:mynewfile s:oldfile-1
  TYPE >>ram:mynewfile s:oldfile-2

"Mynewfile" now holds the contents of oldfile-1 and oldfile-2.

If the destination file does NOT exists when you use the ">>" symbol,
the result depends on your AmigaDOS version.

V1.3 will give an error message saying that the file cannot be opened.

V2+ will create the file and all will be O.K.


=== End of Text ===
```

1.19 Part 4 - Using Shell & CLI - Redirecting Command Input

```
                Redirecting Command Input

Input TO a command (usually from the keyboard) can be from elsewhere
(e.g. a disk file) by using the symbol '<'.

This is a lot more uncommon than the
                output redirection
                 and a little
difficult to comprehend. Don't worry if you cannot follow it now. Wait
until you have learned a bit more about AmigaDOS and then come back to it
when a cross reference brings you back here.
```

Some AmigaDOS manuals say that you can input data as the argument for a command by using:-

```
  > CommandName < Disk:Datafile
```

and give the example of having a file called DateFile which contains a date and time, such as:-

```
   Friday 13-Jun-97 14:06:30
```

Then if you use the DATE command in this way:-

```
  > DATE <Disk:DateFile
```

the date and time would be set to the file contents, i.e.:-

```
   Friday 13-Jun-97 14:06:30
```

However, with most commands, this does not work unless you use:-

```
  > DATE <Disk:DateFile ?
```

The use of the '?' at the end of the command line is explained under the heading

Getting A Command Template

.

An example of using the "<" redirection symbol is contained in Article 41 and in the article on Clock commands.

=== End of Text ===

1.20 Part 4 - Using Shell & CLI - Redirecting Keyboard Output

REDIRECTING KEYBOARD OUTPUT (All Versions of SHELL & CLI)

The asterisk character "*" can be used to represent the current Shell window, or "console" as it is sometimes called.

Normally you would expect that pressing keys would make their characters appear on the screen. However, you can redirect the keyboard output elsewhere with:-

```
  COPY * TO df1:filename
```

or

```
  COPY * TO prt:
```

The "TO" is optional. You can use these instead of the above two lines:-

```
  COPY * df1:filename

  COPY * prt:
```

For example, if you used the first of these then started typing the text
you type will be directed to the file "df1:testfile".

Press RETURN whenever you want to end a line

Press CTRL-\ (i.e. CTRL and the BackSlash) to terminate the COPY command
and return output to the screen.

If you try this out, verify that the file has been correctly created and
written with:-

  TYPE df1:testfile

Pressing CTRL-C then RETURN (before CTRL-\ is pressed) will abort the
procedure and remove the destination file from the disk.

To send your keyboard output to your printer, use:-

  COPY * TO prt:

or

  COPY * prt:

Then type your message as if you were using a typewriter! Although, unlike
a typewriter, you have to press the return key or CTRL-\ before any output
appears on paper.

Finish up with the CTRL-\ terminator. This is much better for a short
note than loading up NotePad or a Word processor! Make sure you get your
spelling right. No chance to make corrections as you have with a Word
Processor!!

=== End of Text ===

1.21 Part 4 - Using Shell & CLI - Pausing Command Output

PAUSING COMMAND OUTPUT (All Versions of SHELL & CLI)

You can pause the command output appearing in the window by one of two
methods - mouse or keyboard. Which one you use is entirely a matter of
preference. I find that I mainly use the keyboard as I am already using
that to enter commands.

Keyboard Method

Press any key that produces a character that can be displayed on the screen.
The output remains frozen until you press "BACKSPACE" or "CTRL-X".

I find that the most convenient stop-start combination of keys are the
back slash key "\" and the BACKSPACE as they are next to each other and
easy to press alternately a number of times.

Output will also resume if you press "RETURN" but this will result in

an error message after the output is complete. If you used "\" to pause, then this message will be:-

  \: Unknown command

This is because the system has recorded the "\" and RETURN as a command line.

In v2+ only, you can also pause output with CTRL-S and restart it with CTRL-X. However, I find this very cumbersome. (Some other CTRL-key combinations will also stop and restart output but as they do other things as well I would recommend you stick to the ones mentioned above.)

Mouse Method

You can pause the output of most things by holding down the RIGHT MOUSE BUTTON. Then just release the button to restart output.

This method is more convenient than the keyboard methods if you are already using the mouse.

=== End of Text ===

1.22 Part 4 - Using Shell & CLI - Aborting Command Output

ABORTING COMMAND OUTPUT (All Versions of SHELL & CLI)

Sometimes you will find that the output to the screen or printer is going on beyond a point that you want it to. You can usually abort it with:-

  CTRL-C

Output to the window should stop immediately.

Output to the printer might appear to take quite a while to stop. In fact, the computer stops sending to the printer but, as the printer has a buffer to hold output from the computer, the printer will keep printing until this buffer is empty.

=== End of Text ===

1.23 Part 4 - Using Shell & CLI - Starting a Program From Shell/CLI

STARTING A PROGRAM FROM SHELL/CLI (All Versions of SHELL & CLI)

O.K., so you can now use a Shell/CLI window but how to you get programs going?

First up, you have to make sure that AmigaDOS can find the program. If it is one of the AmigaDOS programs (e.g. a command or a preferences program, etc.) then it is nearly 100% certain that AmigaDOS will know where to find it (see article PATH). However, if it is not one of AmigaDOS's

programs then you will have to tell AmigaDOS which directory it is in.
There are two ways to do this.

Firstly, you can enter the full path and the program name at a Shell/CLI
prompt, as in:-

```
  > DiskName:DirectoryName/ProgramName
```

For example:-

```
  > Games:Shoot-Em-Ups/Alien_Invasion
```

This will tell AmigaDOS to go to the disk called "Games", look in a
directory called "Shoot-Em-Ups" and start up the game called
"Alien_Invasion".

Secondly, and probably the best, is to make the directory that the program
is in the current directory. If you are still going to start up
"Alien_Invasion" then use:-

```
   > CD Games:Shoot-Em-Ups
```

Having done that, you can just enter

```
  > Alien_Invasion
```

The advantage of the second method is that sometimes programs look in
the same directory that they are in for other files and they will not
run if they cannot find them. Making their own directory the current
directory makes sure that they can find the other files.

Entering the program name this way usually means that the program takes
over the Shell/CLI window from which they were started and you cannot
use that window again until the program is exited (unless the program
detaches itself from the CLI process - a few programs do do this.)

If you want to ensure that the Shell/CLI window is left for other usage,
then use the RUN command as in one of these:-

```
  > RUN Games:Shoot-Em-Ups/Alien_Invasion
  > RUN Alien_Invasion
```

The RUN command will be fully explained in the article RUN although,
to use it, you do not need to know much more about it than to enter it
as illustrated above.

=== End of Text ===

1.24 Part 4 - Using Shell & CLI - Enclosing One Command In Another

```
Enclosing One Command In Another Version 2.0 Onwards

It is possible to include one command within another in a command line.
This is done by using the back apostrophe `. For example:-

  > ECHO "This system has `AVAIL TOTAL` bytes of memory available"

In this example, the command:-

  AVAIL TOTAL

will be executed first and its output will be inserted into the total
command line. If the AVAIL command gave a number 7115128 then the command
line would become:-

  ECHO "This system has 7115128 bytes of memory available"

and the output from ECHO would be:-

  This system has 7115128 bytes of memory available

A practical example of this concept is given in the v3.1 Startup-Sequence.


=== End of Text ===
```