Getting Started

Shell environments differ from Graphical User Interfaces in their tone and their style of interaction. When Graphical Interfaces model objects on the screen, users interact with them as they would with physical objects. The user may pick up an object, examine it, and put it away. In contrast, Shell Interfaces are conversational in nature. The user types a command to the computer, and the computer responds with a few words of its own.

The strength of this conversational metaphor is that it naturally develops into a script of actions. For example, a user might type a few commands to do a weekly clean-up of her system. These same commands can be used to construct a clean-up script.

Simple Commands

The simplest commands are single words. If you type "date" in the nShell, it will respond with the current date and time:

```
% date

Friday, April 22, 1994 9:13:52 AM

%
```

A number of commands work this way, as simple requests for information. As a further example, the "ps" command lists programs that are running:

```
% ps

Process Crea Type   Mem Size    Mem Free   Name

------- ---- ---- ---------- ---------- ----

   8192 MACS FNDR       175104         6932 Finder

   8193 hhgg INIT       187392        24434 File Sharing Extension

  8194 pbc2 INIT       114688        25708 Printer Share

  8197 ttxt APPL       540672       461856 SimpleText

 8199 NSHA APPL       425984       322042 nShell™
```

Command Options

Some commands, including "date", have command-line options. These options are selected using the "-" character. As an example, the "-d" option means "date only" and prevents the time from being displayed:

```
% date -d

Friday, April 22, 1994
```

Another option for the date command is "-s", which means "short date":

```
% date -s

4/22/94 9:23:26 AM
```

More than one option may be used at a time. If we wanted the "short date" with "date only", we could add both options or combine them behind one "-":

```
% date -d -s

4/22/94

% date -sd

4/22/94
```

Command Parameters

In some cases we want to pass more data to a command than with an option. Command parameters are general text strings which are passed to the command for processing. The "echo" command provides an example. Echo takes however many parameters it receives and prints the back out:

```
% echo hello there

hello there
```

In this case echo has two parameters, the string "hello" and the string "there". The nShell normally splits parameters at space characters. If you want to put more than one word into a parameter, you can use quotes to protect the spaces. Compare the following:

```
% echo a       b       c       d
```

```
a b c d

% echo "a       b       c       d"

a       b       c       d
```

In the first case four parameters were specified "a", "b", "c" and "d". In the second case a single parameter was given - the string "a b c d".

The ability to protect spaces is important in the Macintosh because the names of folders and files often contain them.

Standard Input

Commands may expect additional data to be provided after the command line. An example is the "notify" command. This command posts an informational dialog. The text for the dialog may be provided on the command line, or follow it as standard input. One advantage of using standard input is that more than one line of text can be entered. To signal the end of standard input, type a ^D (the ctrl and d keys pressed in unison).

Using the command line:

```
% notify "This is a test."
```

Using standard input:

```
% notify
This is a test.
And it has two lines.^D
```

The main advantage of standard input is that a command may be told to collect text from someplace other than the keyboard. Standard input may be collected from a file or from another command. See the "Input/Output Redirection" section of this manual for more information.

Halting Commands

Shell commands may be halted using ^C (the ctrl key and the c key held down in unison). A ^C will halt processing in the frontmost shell window and produce new prompt (%).

The command-. key combination will halt processing in all shells, and produce a new prompt in each.

The Online Manual

It is sometimes difficult to remember what options and parameters a command accepts. To deal with this problem, Shell environments have a traditional form of on-line help called "man pages" (short for Manual Pages). This on-line manual is searched by typing "man <command name>", as in:

```
% man date
```

[I won't repeat the full man page here, try it yourself!]

In some cases a man page has subsections available. In the case of "date" an "examples" page may be printed. The subsection is added to the end of the line:

```
% man date examples
```

While man pages are available for Flow Of Control commands, such keywords (if, then, else, endif, while, until, do, done) must be quoted on the command line:

```
% man "if"
```

What's Next?

The examples given above should be enough to get you started with the nShell. Once you get the basic techniques in order, you may want to look at the more advanced sections on wildcards, scripting, and input/output redirection.

Good Luck.