since this feature is not available on many operating systems. Command editing refers to the ability to retrieve commands that were already issued to the system and then use them again, either reissuing them verbatim or issuing a modified (edited) version of the command. The functionality of command editing can be conveyed with rather straightforward examples; however, if the examples do not illustrate the *motivation* for its use, learners will not appreciate the feature and when it is most useful. Consequently, they will not use the procedures regularly and will probably forget all about them fairly rapidly.

Consider the following example that explains what command editing means, but fails to motivate its use (the example pertains to a modified VMS operating system):

> Suppose you have typed the following sequence of commands into your computer— (11)
>
> $dir
> $finger
> $go .chap
>
> The first command in this sequence produces a listing of the contents of the current directory; the second, a listing of the people currently using the machine, and the last requests that the current directory be changed to a subdirectory called 'chap'.) Now suppose you want to list the contents of the 'chap' subdirectory. Using command editing, you press the up-arrow a few times so that *$go .chap*, then *finger*, then *$dir* appear on the command line. Now you need only press the return key to reissue the *$dir* command for the .chap subdirectory.

The above example is sufficient for explaining the functionality of the command editing procedure, but is poor for motivating its use in that it provides little or no savings in keystrokes over typing a new *dir* command. In such an example, the usefulness of the procedure is obscured and no novice would see the need to spend time learning it. The following example, in contrast, should make much clearer the usefulness of the command editor.

> Suppose, for example, you want to copy a number of files from someone else's account on another system. To copy the first file (called 'draft1.mss'), you must specify a long path to the relevant directory in your friend's account on the other machine. (12)
>
> $copy cmpsyb::[wells.papers.curchap]draft1.mss *.*
>
> Suppose you want to copy another file called 'final.mss' from that same location. One way to do so would be to type a new copy command that would look exactly the same as the command above, except that 'final.mss' would appear in place of 'draft1.mss'. However, retyping the command will require 58 keystrokes for each file you want to copy. And if you enter the command with unnoticed typing mistakes in the path or the filename, you will have to type the entire command again. Command editing saves you all of this retyping. Instead of retyping, you simply 'recall' the last copy command and edit it to change the name of the file. Typing the up-arrow key brings back the last command. Then, by striking the left-arrow key, you can move the cursor leftward to the specific characters in the filename that must be changed. When you are finished changing the name of the file, press RETURN to issue the modified command. By editing and reusing your first copy command, you save nearly 40 keystrokes for each file you have to copy and you reduce the chances of error in retyping the whole command.

Both examples clarify the concept of reissuing a command. By specifying the sequence of keys that must be typed, both examples also instantiate the rules for executing the command-editing procedures. Only the situation example, however, illustrates the conditions under which the command-editing procedure is more desirable than the procedure for issuing a new command. In particular, command editing is worthwhile when you must type a number of long, similar commands. We will return to the issue of learning when to use a procedure in section 4.4 below.

4.2.2. Subdirectories

Consider another example from the same general domain. We have mentioned subdirectories in the course of the preceding example. This construct may also be unfamiliar to many readers of this chapter. Simply stating that a directory can be divided into subdirectories is sufficient for 'explaining' the concept, but it is unlikely that the user will be sufficiently motivated to acquire the cluster of skills needed to make use of such a facility.

The following example, taken from the IBM DOS Manual (Anonymous, 1983), illustrates the concept more fully along with some rationale for the usefulness of subdirectories, but adds little to the reader's sense of how subdirectories might facilitate day-to-day activities on the computer.

> DOS Version 2.00 gives you the ability to better organize your disk by placing groups of related files in their own directories —all on the same disk. For example, let us assume that the XYZ (13)