A Note on the Multics Command Language

A. W. COLIJN

Department of Computer Science, The University of Calgary, Calgary, Alberta, Canada T2N 1N4

KEY WORDS Operating system Command language Recursion Multics Ackermann's function Towers of Hanoi

The Multics operating system,[1,2] currently available on a series of Honeywell machines, was developed starting in 1964 as a joint effort involving Project MAC of M.I.T., Bell Telephone Laboratories, and the computer department of General Electric Company, later taken over by Honeywell Information Systems, Inc. This operating system is very nicely structured, and it is intended for, and very well designed for interactive use. The command language, therefore, is also designed to make interactive use of the computer convenient. This is achieved through a combination of features, including the availability of powerful commands, the availability of standard and user-defined abbreviations, the availability of on-line documentation, and the great consistency in the use of arguments to the commands.

In addition to containing a number of powerful commands for interactive use and immediate execution, the Multics command language is also a reasonably complete programming language, since there is provision in Multics for defining files, called segments in Multics, containing commands; these commands are executed when such a segment is invoked by giving its name as the first argument to the *exec_com* processor. These so-called *exec_com* segments may have parameters, which are passed by value, and they may be called recursively. An extensive set of so-called active functions is provided in the operating system; these active functions allow certain operations, including arithmetic ones, to be performed, and they permit access to many system values and parameters ranging from the time of day and the electronic mail system to such things as the search rules for locating segments in the hierarchical file directory system.

Commands in *exec_com* segments fall into two categories: the regular Multics commands, and commands which can occur only in *exec_com* segments. The latter, which are distinguished by being preceded by an ampersand, include the *&print* command, and the *&if* command, which permits conditional excecution of commands, but which, regrettably, may not be nested.

A weakness in the *exec_com* facility is the fact that there are no facilities for declaring local variables, and even the facilities for declaring and using non-local variables are inconvenient, and very poorly documented. For example, since it is apparently assumed that most *exec_com* processing will be concerned with strings, if the value assigned to a variable by means of the *value$set* command is an arithmetic expression, the command assigns to the variable the string representing the expression (which may be evaluated later using the active function *value*, rather than the value of the expression.

0038-0644/81/070741-04$01.00

Received 12 November 1980