MSPM SECTION BZ.10.00

Identification

The Multics APL Interpreter--Overview

Purpose

This document is a description of the internal operation and method of implementation of the Multics APL interpreter. It is intended to be sufficiently detailed to be adequate background for any system programmer desiring to maintain or modify the interpreter. As a prerequisite to understanding this section, the reader must be acquainted with the APL language (references 1 and 2) and with the usage of the APL command in the Multics environment (references 3 and 4).

References

1. I.B.M. Corporation: APL/360 Primer. File number GH20-0689-1; 1969.
2. Falkoff, A. D.; Iverson, K. E.: APL/360 User's Manual. I.B.M. Corporation; August, 1968.
3. M PM Section ?.
4. MSPM Section ?.

Notation

Since the APL character set differs so markedly from ANSCII, there is some difficulty expressing APL source code in this documentation. This document uses the same escapes which apply to Model 37 Teletype users talking to APL on Multics. See MPM Section or MSPM Section BZ.10.04.

Interpreter Structure

The interpreter itself consists of a single bound segment. It is purely recursive so that it can be invoked from within itself. The interpreter requires for its operation several working segments, named "apl.symbol.?", "apl.value.?", "apl.function.?", and "apl.stack.?" Here, "?" represents a unique identifier created by each invocation of the interpreter. These segments are created in the user's process directory. They are used to hold the current workspace.

Segment "apl.symbol.?" holds the symbol table for the current workspace; segment "apl.value.?" holds all data values; segment "apl.function.?" holds all function definitions; "apl.stack.?" holds the APL state-indicator stack. and

When the current workspace is saved, the information from "apl.symbol.?", "apl.value.?", "apl.function.?", and "apl.stack.?" is compacted into one segment, so that a stored workspace is a single segment. Conversely, upon loading or copying from a saved workspace, the symbol definitions, data values, function definitions, and stack are routed to their proper working segments.

The major components of the interpreter are the parser, the lexical analyzer, the operator routines, and the editor. The principal component is the parser, which dispatches control to other components as required. The central operation loop of the parser consists of reading an input line, parsing it, calling upon other modules to perform the actions implied by the line simultaneously with doing the parse, and finally coming back to read the next line.

The input lines presented to the parser are lexically analyzed by the lexical analyzer. Input lines read during immediate-execution mode (i.e., read from the console, unless the input stream has been diverted) are lexed immediately after read-in and before presentation to the parser. Function lines are kept as text until definition mode is left, then the entire function is lexed at once; during execution of a function, the parser draws from the stored lexed lines.

The operator routines are called upon by the parser to perform most APL operations. The operations are performed during the parse; by the time a statement is completely parsed, it has been completely interpreted.

The editor is given control when the user enters definition mode. The editor treats a function as purely a source string, until the user requests exit from definition mode. At that time, the function header is checked for validity, names are checked for duplication, and all the lines of the function are lexically analyzed. Both the source and the lexed versions are saved in the current workspace. The source is used for future editing (whenever the source changes,

the old lexed version is thrown away) and for printing of error messages. The lexed version is used as input to the parser when the function is executed.

Documentation Organization

This is the arrangement of the various topics covered by the MSPM APL interpreter sections.

BZ.10.00 The Multics APL Interpreter--Overview
- **Purpose**
- **References**
- **Notation**
- **Interpreter Structure**
- **Documentation Organization**

BZ.10.01 APL Parser
- **Purpose**
- **Operation--Initialization**
- **I/O Stream Switching**
- **Operation--Termination**
- **State Indicator Stack**
- **Operation--Main Loop**
- **Implementation of Reductions Analyzer**
- **Interrupt Processing**

BZ.10.02 APL Formal Syntax and Reductions
- **Purpose**
- **Tokens**
- **Basics**
- **Categories**
- **BNF**
- **Reductions Analysis Scheme**
- **Reduction Rules**
- **Type Codes**