Published: 06/18/68

## Identification

An ASCII File Scanner
D. L. Stone

## Purpose

The procedure scan is used to parse ascii files. It was written for the Interim tape_daemon and merge_editor, to allow them to get the arguments from their control files. It expects an ascii segment which is considered to be divided into lines by new_line characters (octal 012). Tokens on a given line are contiguous characters, none of which are delimiters, which are surrounded by delimiters. The delimiters are: space (040), comma (054), horizontal tab (011) and new_line (012). The colon (072) is treated as a special token which does not need to be delimited.

## Call Interface

```
Call  scan$start (p,max_length,error)

     dcl   p ptr,

           (max_length,error)  fixed bin(17);
```

This call causes scan to initialize its static storage, setting the file pointer to "p", the maximum number of characters to be scanned to "max length"; current character position is set to 1; "error" is always returned as 0. The "start" call must be made before any other calls to scan.

```
Call  scan$push (p,max_length,error)
```

The "push" entry causes scan to save the previous file pointer, maximum length and current character position in a pushdown list kept in internal static storage (maximum depth 9) and to switch its current information to the pointer and length given in the call (current character position is set to 1). The only possible error returned is error = 1 - pushdown depth exceeded.

```
Call  scan$pop(error)
```

The "pop" entry causes scan to revert to the previous file at the previous place. All knowledge of the current file is erased. The only error possible is error = 1, attempt to "pop" past the bottom of the stack.

Call scan$next_token(p,n,error)

```
   dcl   n fixed bin (17);
```

"next_token" starts at the character in the current position in the current file and searches for the next string of characters which is:

1. a colon or a new_line character

or 2. a consecutive string of non_delimiters terminated by a delimiter or a colon.

On return, p points to the character string, left adjusted in a word, while n gives the number of characters found. In the case that the string is a new_line character, the current character position is left pointing to the new_line, such that it will be encountered again if "next_token" is called again. (See "next_line_token"). Errors returned are:

error = 1; end of file encountered (as specified by max_length);

error = 2; token of size > 150 characters encountered.

Call scan$next_line_token(p,n,error)

This entry returns the effect of a "next_token" call after advancing the current position to one character beyond the next new_line encountered. Same error possibilities as "next_token".

Call scan$string (p,n,error)

The "string" entry returns (via the pointer "p" and character count "n") the string of characters beginning at the current one and ending at the character before the next new line. The same errors as "next_token" are possible.

Call scan$cur_line(p,n,error)

This entry moves the current character position back to the character following the previous new_line character (or the beginning of the file) and then returns the value of scan$string; the current position is set to the new_line character following the information returned.