

How Long is a String?

Strictly within COBOL, the answer is easy. A character string is declared as PIC X (integer), where the value of integer can be any number from 1 to the documented limit for your COBOL system or the syntactic limit of your chosen COBOL dialect. The compiler is aware of the specified, fixed size and generates appropriate code for MOVE and IF statements based upon the specified size.

When passing strings to subroutines written in, or for use with, other languages, e.g. Micro Focus' own CBL_FILE_* functions, the answer is not so simple. Strings are typically passed "by reference". The string itself is not copied to the subroutine's equivalent of a COBOL linkage section; only the memory address of the first byte is passed to the equivalent of a COBOL "USAGE POINTER" variable. The number of bytes in the string is unknown without further information. For example, a passed string might be a fully qualified UNIX file specification as exemplified below. The called subroutine would "know" the address of the leading "/", but it is difficult to determine where the string ends.

```
/home/users/mydir/subdir/aoiuydfiausydiuiuyiebrtbdyuyetfalueerbbbesyuvveyyfd
```

It is obvious to experienced human intelligence that the file name is too long. The end of the valid file specification must come before the end of memory, but where exactly? The COBOL PIC clause was not passed and there is always something in memory at locations beyond the end of the string.

The C language commonly uses two different means to identify string lengths. Strings are declared as arrays of type "char". The array dimension is the string length. To call a C routine declared that way, the following kind of interface may be documented:

```
01 STRING-VAR          PIC X(1000).
01 STRING-LEN          PIC 9(4) COMP-X VALUE 1000.
```

```
CALL 'CPRG' USING STRING-VAR STRING-LEN
```

In C, a string may also be declared as a pointer to an object of type "array of char". The pointer's value is the address of the first character, but the length is unspecified. This is commonly done in callable subroutines which accept string parameters. The end of the string is marked by a defined "terminator", a character which is not otherwise valid in the string. For example, the operating system's command processor ("shell") uses a null (X'00' or LOW-VALUES) byte to mark the string end:

```
01 NULL-TERMINATED-COMMAND.
   05 COMMAND-STRING          PIC X(1000).
   05 FILLER                  PIC X VALUE LOW-
VALUES.
```

```
MOVE 'ls -l > outfile' TO COMMAND-STRING.
```

* The move will automatically pad the right end of the literal with spaces.

```
CALL 'SYSTEM' USING NULL-TERMINATED-COMMAND.
```

If the required terminator is a space, as it is with CBL_COPY_FILE, modify the VALUE clause of the FILLER appropriately.

The same is true if the passed string is a literal. If, for example, you call CBL_COPY_FILE with a literal as one of the file specifications, there must be a blank space between the string and the closing quote:

```
'space-terminated-literal '
```

```
'unterminated-literal'
```

The content of the memory location one byte to the right of the final "!" in the second example is unknown and unknowable. If it does not happen to be a space, CBL_COPY_FILE will use an unintended file name or report 'file not found'.

In Java, "string" is passed with attributes of length (length of usable text in the current value), capacity (full, dimensioned size of the variable), and pointer (the address of the first character in memory). The Micro Focus supplied file javatypes.cpy shows the definition. Please refer to the "Distributed Computing" manual in your documentation set for specific usage details.