String Format

Lib – Workgroup: 0x10c Standards Committee
RFC X

Malte Schuetze

April 11, 2012

This document presents a general format for strings in libraries intended to be shared across programs and with other users.

1 Introduction

Shared Libraries depend on certain formats being given. Strings, being a common argument type for cross-library calls, should therefore be standardized. Length-prefixed strings are to be used for security and efficiency reasons.

1.1 Terminology

word 16 bits. This is the smallest unit addressable by the DCPU.

character A group of bits representing a glyph or control sequence. A control sequence is a non-printing character that influences the text.

string A sequence of characters. The length of the sequence can vary at runtime.

P-string A string whose length in words is indicated by prefixing a word containing that length to the character string. The prefix word itself MUST NOT be included when determining the length.

C-string A string whose length is indicated by suffixing a NUL character to the character string. The first occurrence of such NUL character terminates the string. The NUL character is the character whose bits are all zero.

library A group of functions used by different programs.

program A sequence of machine instructions for the DCPU.

user Any person using or creating a program.

1.2 Notational Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119

2 Format

A P-string is composed of a 16-bit length prefix and a sequence of n words where n is the value of the prefix.

The prefix word MUST be present and MUST represent the exact number of following words.

Note that the empty P-string consists entirely of a length word containing 0x0000.

```
P-STRING  =  LENGTH BODY
LENGTH    =  n
BODY      =  nWORD
WORD      =  %x0000-ffff
                  ; any 16-bit value
```

3 Rationale

This section is not normative.

The rationale for using P-strings is a simple matter of weighing benefits against disadvantages.

3.1 Benefits

- Accessing the length of the string is O(1) fast.
- Buffer overflows are prevented by being able to allocate enough space ahead of time.

  This also increases the security of programs by preventing arbitrary shell code from being executed.
- The null-character can be used in strings, while using it in a C-string would terminate the string.
- The same format can be used for other data structures such as length-prefixed arrays. This means that functions such as concat must only be implemented once.

3.2 Disadvantages

- Indexing begins at 1 instead of 0.
- Fixed-length strings still require a prefixed length word. The same would be true of fixed-length C-strings. Both formats can be abbreviated by omitting the known prefix/terminator at the cost of ceasing to be a P- or C-string per the definitions given here.

4 Other Notes

- Since the smallest addressable size in the DCPU is a word (16 bits), this effectively allows a maximum string length of 65536 characters without any loss in efficiency.
- Cutting off the beginning of a string is more expensive for P-strings, but cutting off the end is more expensive for C-strings. (Other solutions are possible, but introduce memory leaks.) Therefore, the two arguments negate each other.
- The names P-string and C-string come from Pascal and C, which use the respective format.
- The current implementation of characters in the DCPU effectively leaves the high 8 bits empty. It is therefore possible to store two characters in one word. Such packed strings are outside the scope of this RFC.

5 Security Considerations

Using P-strings reduces the risk of arbitrary shell code being executed by overflowing the input buffer. However, hidden instructions in the string may be executed later when the sections are not cleared and the stack or instruction pointer is moved there. Measures should be taken by the user to ensure this does not happen.