A descriptor based approach to strings

Ulli Hoffmann, Andrew Read

EuroForth 2018

Lots of prior work on Forth strings, e.g.

- Klaus Schliesiek - String Stack, 1986
- Brad Rodriguez – PatternForth, 1989
- Ulli Hoffmann – Stack of Stacks, 2017

Why another strings packages? So we can…

- Try out a new idea well-used in other languages:
  - Array slices
- Further our plans for the “New Synthesis”
  - A strings package to that will fit into the kernel

Array slices: Go example

- Arrays are value types
- Assigning an array makes a copy of the whole array
- Slices are reference type wrappers for arrays section
- Manipulate slices without copying or modifying the array
- Multiple slices may reference the same array
- Also in Algol 68, Fortran 77, even Sinclair BASIC

Why integrate a strings package into the kernel?

- Text processing is a big part of Forth
- Benefits in a more flexible / scalable interpret loop
- We have a killer app in mind – parsing the input stream with regular expressions

Our evaluation criteria

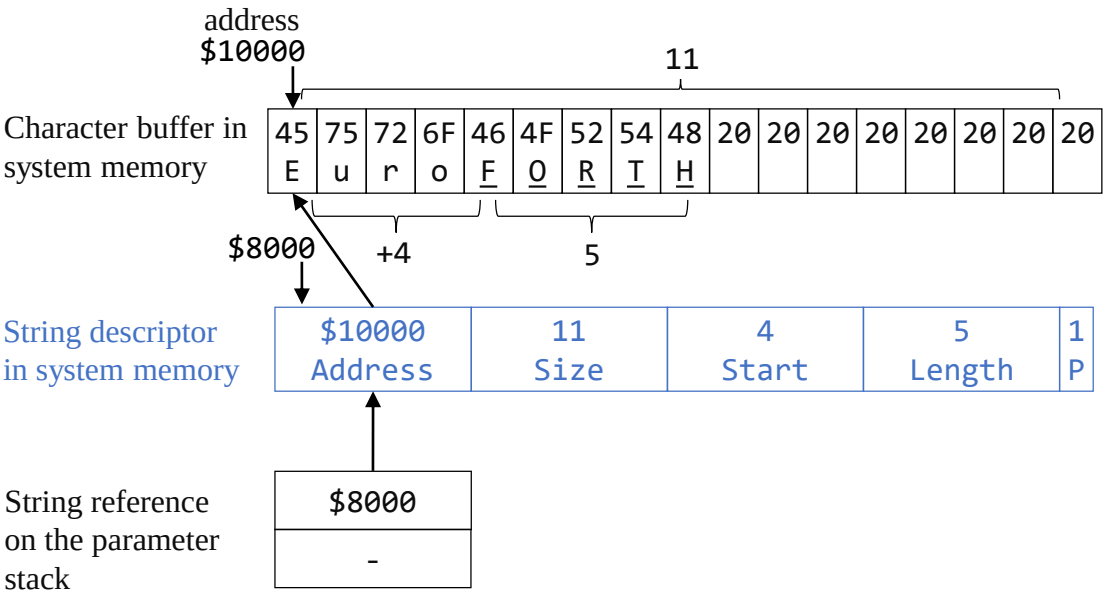
- Develop using an elementary vocabulary
- Cannot rely on string handling facilities in the kernel
- Do not commit the kernel to fixed choices in other areas, such as memory management
- Prefer the package to be small
- Must have good performance – no string copying

Anton Ertl's "Standardize Strings Now!" EuroForth 2013

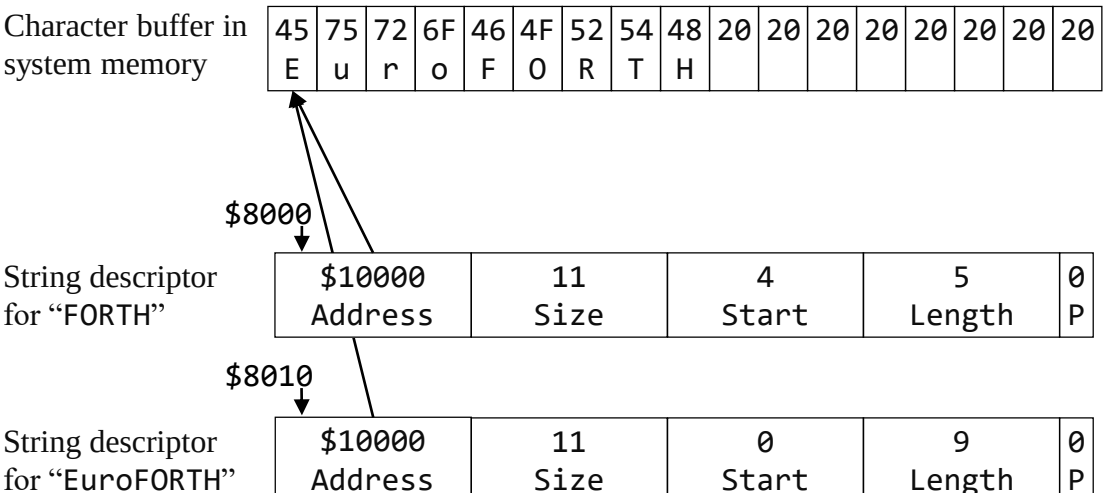
Criteria and challenges for desirable strings :

- Ease of use - convenient stack representation
- Memory management of the character buffer
- Integration with the rest of Forth

Our string descriptors



Multiple descriptors may reference the same character buffer

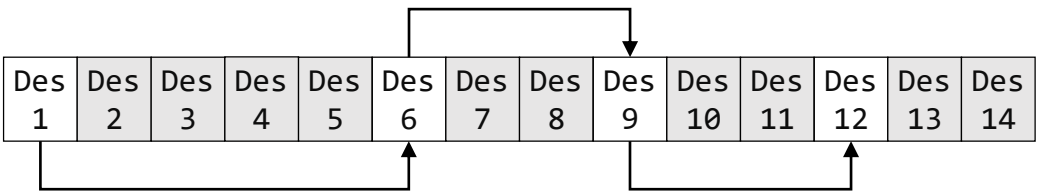


```
10 $initialize
S" Veni vidi01234567890123456789"
\ : $make ( c- addr len size flag -- s$)
9 swap -1 $make
S" vici " dup 0 $make
\ : $+ ( s$ r$ -- s$)
$+
\ : $len ( s$ -- s$ n)
$len CR .
14
\ : $s ( s$ -- c- addr u)
$s CR type
Veni vidi vici
```

Anton's first concern: memory management

- We do not memory manage the character buffer
  - Each kernel will have its own approach to memory management, esp. for kernel level strings
- We do memory manage the descriptors
  - A pool of free descriptors is created at initialization
  - The free descriptors are held in a linked list
  - More descriptors can be added during run-time

Free descriptors are held in a linked list



Key

Unallocated string descriptor

Allocated string descriptor

Anton's second concern: ease of use

`$drop` - remove the descriptor from the stack and recycle

`$dup` - create a new descriptor pointing to the same buffer and having the same length, start and capacity

Our convention: words that consume a stack descriptor automatically recycle ($drop) it

Where this is not convenient, we break Forth convention and leave the parameter on the stack `$len (s$ - s$ n)`

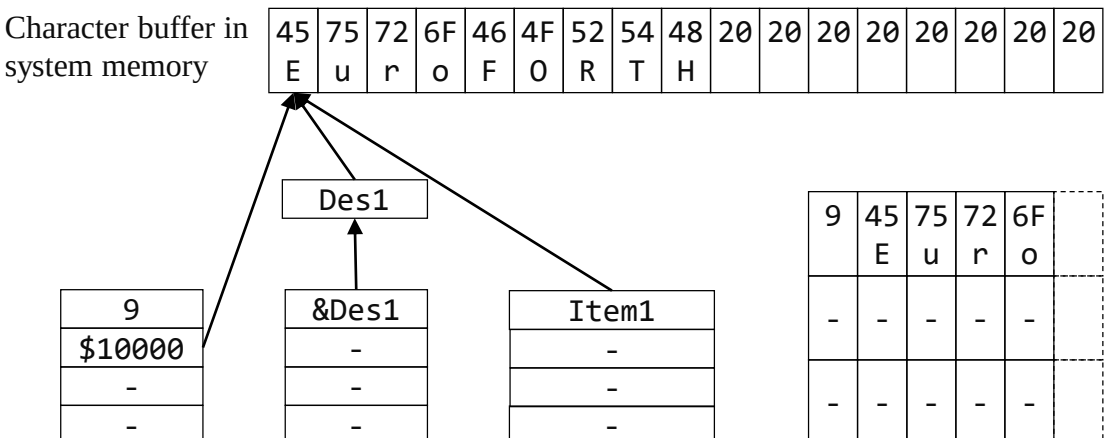
We offer permeant and temporary strings

Our “killer app” – a regular expression matcher

```
: $regex ( s$ r$ -- a$ b$ s$ TRUE | FALSE )
\ Search for regex r$ in string s$ if the regex is found , a$ is the
\ substring before the first match , b$ is the first match
\ s$ ( modified ) is the rest of the string and the TOS is true ;
\ otherwise return false and preserve s$ unmodified
\ r$ is $drop 'ed ( recycled unless defined to be a permanent string )
\ a$ , b$ and s$ all reference portions of the same character data in memory
```

- Based on original C code by Pike and Kernighan
- Less than 250 lines long in Forth (including comments)
- Minimal stack signature thanks to convenient stack representation
- No string copying
- Justifies separate descriptors referencing the same character array

Anton's third concern: integration with the rest of Forth – a reasonable balance between utility and complexity



a) "c-addr u" b) descriptors c) non-copying string stack d) copying string stack

In the paper:

- The full strings package wordlist
- Our extended set of regular expressions
- Consideration of limitations (primarily overhead)
- References

On GitHub:

- Full code, open source
- String package and the regular expression matcher

Thank you!