The akshar package

Vu Van Dung

Version 0.1 — 2020/05/17

Contents

1 Introduction 1

2 User guide 1

3 Implementation 1

Index 2

1 Introduction

When dealing with processing strings in the Devanagari script, normal LaTeX commands usually find some difficulties in distinguishing "normal" characters, like क, and "special" characters, for example ु or ी. Let's consider this example code:

```
\ExplSyntaxOn
\tl_set:Nn \l_tmpa_tl { की}
\tl_count:N \l_tmpa_tl \c_space_token tokens.
\ExplSyntaxOff
```

2 tokens.

The output is 2, but the number of characters in it is only one! The reason is quite simple: the compiler treats ी as a normal character, which it isn't.

To tackle that, this package provides expl3 functions to "convert" a given string, written in the Devanagari script, to a sequence of token lists. each of these token lists is a "true" Devanagari character. You can now do anything you want with this sequence; and this package does provide some front-end macros for some simple actions on the input string.

2 User guide

a

क्ष

b

कौ

c

क्ष्य

d

3 Implementation

```
⟨@@=akshar⟩
⟨*package⟩
```

Declare the package.

```
\RequirePackage{fontspec}
\ProvidesExplPackage {akshar} {2020/05/17} {0.1}
  {Support for syllables in the Devanagari script (JV)}
\tl_const:Nn \c_foo_joining_tl { ◌्}
\tl_const:Nn \c_foo_diacritics_tl {◌ँ◌ं◌ः◌़◌ा◌ि◌ी◌ु◌ू◌ृ◌ॄ◌ॅ◌ॆ◌े◌ै◌ॉ◌ॊ◌ो◌ौ◌ॎ◌ॏ◌ॕ◌ॖ◌ॗ◌ॢ◌ॣ}
\tl_new:N \l_foo_input_tl
\tl_new:N \l_foo_tmp_tl
\bool_new:N \l_foo_prev_joining_bool
\seq_new:N \l_foo_char_seq
\prg_generate_conditional_variant:Nnn \tl_if_in:Nn { No } { TF }
\cs_new:Npn \foo_str_getchar:nn #1 #2
  {
    \seq_clear:N \l_foo_char_seq
    \bool_set_false:N \l_foo_prev_joining_bool
    \tl_set:Nn \l_foo_input_tl {#1}
    \tl_map_variable:NNn \l_foo_input_tl \l_foo_map_tl
      {
        \tl_if_in:NoTF \c_foo_diacritics_tl {\l_foo_map_tl}
          {
            % It is a diacritic.
            \seq_pop_right:NN \l_foo_char_seq \l_foo_tmp_tl
            \seq_put_right:Nx \l_foo_char_seq { \l_foo_tmp_tl \l_foo_map_tl }
          }
          {
            \tl_if_eq:NNTF \l_foo_map_tl \c_foo_joining_tl
              {
                % It is the joining character
                \seq_pop_right:NN \l_foo_char_seq \l_foo_tmp_tl
                \seq_put_right:Nx \l_foo_char_seq
                  { \l_foo_tmp_tl \l_foo_map_tl }
                \bool_set_true:N \l_foo_prev_joining_bool
              }
              {
                % It is a normal character
                \bool_if:NTF \l_foo_prev_joining_bool
                  {
                    % but previously there is a joining character
                    \seq_pop_right:NN \l_foo_char_seq \l_foo_tmp_tl
                    \seq_put_right:Nx \l_foo_char_seq
                      { \l_foo_tmp_tl \l_foo_map_tl }
                    \bool_set_false:N \l_foo_prev_joining_bool
                  }
                  {
                    % Previously: nothing special.
                    \seq_put_right:Nx \l_foo_char_seq { \l_foo_map_tl }
                  }
              }
          }
        % Plus two just to guard against breaking too soon
        \int_compare:nNnT {\seq_count:N \l_foo_char_seq} > {#2 + 2}
          {
            \tl_map_break:
          }
      }
    \seq_item:Nn \l_foo_char_seq {#2}
  }
\NewDocumentCommand \mystrchar {mm}
  {
    \foo_str_getchar:nn {#1} {#2}
  }
⟨/package⟩
```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

B

bool commands:
\bool_if:NTF 37
\bool_new:N 10
\bool_set_false:N 16, 43
\bool_set_true:N 33

C

cs commands:
\cs_new:Npn 13

F

foo commands:
\l_foo_char_seq 11, 15, 23, 24, 30, 31, 40, 41, 47, 52, 57
\c_foo_diacritics_tl 7, 20
\l_foo_input_tl 8, 17, 18
\c_foo_joining_tl 6, 27
\l_foo_map_tl 18, 20, 24, 27, 32, 42, 47
\l_foo_prev_joining_bool 10, 16, 33, 37, 43
\foo_str_getchar:nn 13, 61
\l_foo_tmp_tl 9, 23, 24, 30, 32, 40, 42

I

int commands:
\int_compare:nNnTF 52

M

\mystrchar 59

N

\NewDocumentCommand 59

P

prg commands:
\prg_generate_conditional_variant:Nnn 12
\ProvidesExplPackage 4

R

\RequirePackage 3

S

seq commands:
\seq_clear:N 15
\seq_count:N 52
\seq_item:Nn 57
\seq_new:N 11
\seq_pop_right:NN 23, 30, 40
\seq_put_right:Nn 24, 31, 41, 47

T

tl commands:
\tl_const:Nn 6, 7
\tl_if_eq:NNTF 27
\tl_if_in:Nn 12
\tl_if_in:NnTF 20
\tl_map_break: 54
\tl_map_variable:NNn 18
\tl_new:N 8, 9
\tl_set:Nn 17