Emacs Support for Perl Compatible Regular Expressions (PCRE)

Qingwei Lan (404458762)
University of California, Los Angeles

June 1, 2016

1 Introduction

Emacs currently supports POSIX style regular expressions and the objective of the paper is to outline a method of integrating Perl Compatible Regular Expressions (PCRE) into Emacs.

We want to support this because PCRE supports powerful features that the original POSIX style doesn't support. Integrating PCRE will greatly enhance productivity for Emacs users.

2 Integration Method

Since PCRE and POSIX regex differ in many different aspects, it would be difficult to do regex translation from PCRE to POSIX. From what I figured, it would be easier to simply rewrite the library in Emacs Lisp. Another sugggestion would be to integrate the already existing PCRE C library with the Emacs module system.

2.1 Emacs and Dynamic Loadable Libraries (DDLs)

Since the proper Emacs programming language is Emacs Lisp, a less commonly known language than the traditional C programming language, many issues have been reported that Emacs should support C modules and the topic has been discussed extensively in the GNU Emacs mailing list.

The Emacs Module System provides a way of extending Emacs with dynamically loadable librarie (DDLs), which can be written in C or C++. The great thing about this is that people can write their own extensions in a language other than Emacs Lisp and load into Emacs without recompiling the whole program.

2.2 PCRE C Library

PCRE provides a C library that is simple to use. We can declare regular expressions as simple C strings and compile it into the a native `pcre` object using the API function `pcre_compile` provided.

After we have the `pcre` object we can perform a search for matches on the desired string, which is also represented as a C string using the function `pcre_exec`. The function will return results indicating offsets of the matches.

2.3 Using the `DEFUN` method

The main purpose of this project is to add a new function that allows users to search for PCRE regex within the editor. Emacs allows users to define Emacs funcitons with the `DEFUN` method.

The following example of the `DEFUN` function is found in the file `emacs/src/search.c` of Emacs version 25.

```
DEFUN ("re-search-forward", Fre_search_forward, Sre_search_forward,
    1, 4, "sRE search: ", doc:/* documentation on re-search-forward */)
  (Lisp_Object regex, Lisp_Object bound, Lisp_Object noerror, Lisp_Object count)
{
  return search_command (regex, bound, noerror, count, 1, 1, 0);
}
```

The first argument is the name of the function that will appear in the Emacs editor. The second argument is the actual name of the C function created, typically preceded by an `F`. The next is the name for the C constant structure used for internal use, typically preceded by an `S`. The following two argument specifies the minimum and maximum number of arguments to the Emacs function. The next argument is the interactive definition, which will appear in the Emacs editor when the user uses it. The next argument is a C comment that serves as the documentation of the function. Finally is a list of function argument names.

Then to let the Lisp reader know about the function we would have to call the `DEFSUBR` function, with the actual C function name as the argument. Extending the example above would be to call

```
DEFSUBR(Fre_search_forward);
```

This will let the Lisp reader know about this function and the user of Emacs can call the function. This method seems simple and is perhaps the best way for the project to easily integrate PCRE into Emacs.

2.4 Integration

In order to integrate PCRE into Emacs, we would first have to import the header file for PRCE, then specify `Lisp_Object` wrappers for the function. Then we use the `DEFUN` and `DEFSUBR` functions to let the Lisp reader get the definition for this function. An simple high-level example of what the module might look like would be something like the following.

```
#include "pcre.h"

DEFUN ("pcre-search-forward", Fpcre_search_forward, Sre_search_forward,
    1, 4, "sPCRE search: ", doc:/* documentation on pcre-search-forward */)
  (Lisp_Object regex, Lisp_Object bound, Lisp_Object noerror, Lisp_Object count)
{
  Lisp_Object   retoffset;
  const char    *error;
  pcre          *re;
  int           erroroffset;
  int           rc;


  re = pcre_compile (regex,
                     PCRE_MULTILINE,
                     &error,
                     &erroroffset,
                     0);
  rc = pcre_exec (...);

  // configure the Lisp_Object retoffset
  ...

  return retoffset;
}
```

Then we would have to include the line below for this extension to work properly.

```
DEFSUBR(Fpcre_search_forward);
```

Similar program and function definitions can be done for `pcre-search-backward` and replace methods. The hard part is to build the connection between the PCRE library and the Emacs module system.

3 Conclusion

This project would be hard if we were to rewrite the whole library from scratch or translate PCRE to POSIX regex because of all the details and specifications in PCRE.

However, by choosing to integrate the already existing PCRE library with the GNU Emacs module system, we saved a lot of work and can simply include all the functions in the PCRE library in the Emacs source code. Then we use `DEFUN` and `DEFSUBR` to make the function available to the Lisp reader and the users.