Implementing the GB2312 Encoding in MoarVM

Personal Information

Contact

Nickname: MidCheck
Full name: Xin Jiang
E-mail: MidCheck@foxmail.com
daemon.love.1314@gmail.com
Tel.No.: (+86)187-1056-8391

Education

Xi'an Technological University, Xi'an China

Information Confrontation Technology,
Computer Science and Engineering College
Expected graduation July, 2020

Experience

- I am good at using C language, familiar with C++, Perl 6, Python.
- I have never done open source work before, so this is the first open source experience in my life. There are also no online projects, but several offline projects have been done before.
- I developed a simple translation software in C language at the end of 2016 to help me better read foreign language materials.
- I used the PBC library to help my tutor in the school implement a cryptographic algorithm he designed at the end of last semester, but I can't describe too much here because of the confidentiality issue.
- Can read Java and Common Lisp code.
- Can develop on Linux and Windows.
- Learn about computer networks and cryptography.
- Learned some reverse engineering techniques, including disassembly and decompilation.

About Project

Project Name

Implementing the GB2312 Encoding (Simplified Chinese)

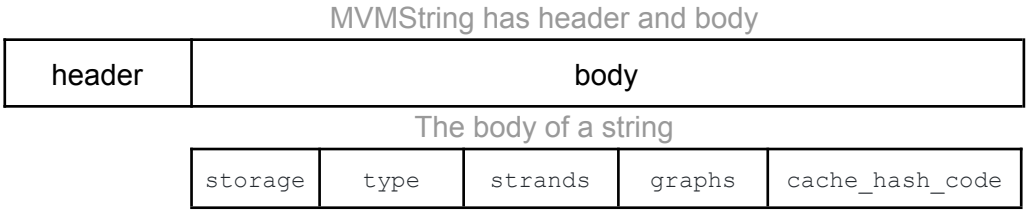
Project Description

- Goal？
  Now Perl6 does not support GB2312 encoding and decoding. Need to implement Perl 6 encoding and decoding function of GB2312 on MoarVM, finally MoarVM can support it well.

- What is the difference between Unicode and GB2312 encoding?
  Each character included in GB2312 is included in Unicode, only the code points used for the same character are different.

- Perl6 used normalization to represent characters visible to the user. Enter the same glyph represented by one code point or two code points, and finally convert to the normalized form G (extended extension of NFC).

- How to represent a string in MoarVM?



Solution

- First we will use the GB2312 code table to convert to the structure we need using other tools. The code table has a total of 94 zones, each zone has 94 characters. In order to be compatible with ASCII, the representation used is EUC-CN. So 0x0000 - 0x007f represents the ASCII Codes, and {1 - 94}{1 - 94} is its Location Codes. So there is at least 94*94*2 + 0x80 = 17800 bytes, which contains unassigned words. The Native Codes can be obtained from the two-byte Location Codes plus 0xA0A0. There is a code table here https://uic.win/charset/show/euc-cn/.

- Next, cache the GB2312 code table in Unicode plaintext into memory. We can use the following format, such as **{ Unicode code point, GB2312 code point}** or **{ GB2312 code point, Graphic Character }**(the Graphic Character is a character that can be printed in the GB2312 code table) or other structure.

    - For the first structure, when we get a character or its Unicode or GB2312 code point, we can match the other two things in the structure of the character.
    - For the second structure, just a pairing of its GB2312 code points and characters.

  If we have other methods that make it faster to find, they can be used here. What I am thinking about now is to use the structure of the hash table or a plain array, with the character as its key or index, its code point as the value, or vice versa.

- Then, according to different bytes, set storage_type to different types, for example, single byte is set to MVMint8, double byte is set to MVMint16, and four bytes are set to MVMint32. For single-byte characters can be stored in blob_8, double-byte can be stored in the lower 16 bits of blob_32, or use any byte, four bytes can be stored in blob_32.

- Finally, fill the rest of it according to the structure of MVMString.

Time Commitment

Because of my school schedule and final exam arrangement, I can't guarantee a fixed time period every day in June. But I can guarantee that in general there are 30 to 40 hours a week to complete this project.

Timeline

Period	Work
May 27 to June 2	Familiar with other related documents, collect information that may be used, such as GB2312 code table.
June 3 to June 19	Write a test version of the program that can be run separately, to achieve the encoding and decoding function of GB2312.
June 20 to June 23	Test the program to fix possible bugs in the beta.
June 29 to July 7	Convert a beta program to a MoarVM type program.
July 8 to July 22	Merge the program into MoarVM and test it.
July 26 to August 3	Optimize the algorithm to improve its performance, then test and fix the bugs that occur.
August 4 to August 11	Complete its technical documentation and instructions for use.

August 12 to August 19	Write a final summary of the product.

Deliverables

In the first stage, we have a beta version of the encoding and decoding capabilities that can be run separately. It can encode a string into a byte array or decode a byte array into a string.

In the second stage, we can get the program as a MoarVM module. It can encode MVMString as a byte array or decode a byte array into an MVMString. At this time, MoarVM should be able to support GB2312.

In the third stage, we expect to get the final relevant documents of the project.

Other information

About me

I really like technology, not just programming. Some things of interest always want to know how it works, and then see if it can create something more interesting. For example, I like music very much in my life, because unique music can sometimes bring me inspiration, often listening to music while programming. However, because curious electronic devices made beautiful sounds, I learned a lot about digital sounds and then met the programming language Chunk.

I also like beautiful things, such as poetry, beautiful paintings, elegant code, and so on. Beautiful things will make me feel happy.
For my technical career, I want to be a good programmer and hacker. Programming is the foundation, so I attach great importance to it, but because there is no guidance, sometimes it will be confusing and little progress.

About Open Source

I am very passionate about open source, and I have been learning and maintaining the spirit of open source since I learned computer technology.
On the one hand, to participate in GSoC in order to improve my technology and understand the real software development process, on the other hand I hope to learn about an open source community by participating in GSoC.

Even if I am not selected this time, it is worthwhile to join the community and meet some friends, and finally the code and energy that can be contributed to this community.