Solving C's biggest flaw — Hemispheric divergence

Ian F.V.G. Hunter

Unaffiliated/Excommunicated, near-antipode of Campbell Island, New Zealand

Abstract

We propose a new domain-specific language (DSL) called "Code for Hemisphere-Unconstrained Master Programs" (CHUMP), which frees software engineers from the traditional constraints of bi-hemispherical coding. We provide several example programs covering core scenarios and discuss future work in this area.

1 Introduction

Traditionally, developing software for both hemispheres has been a difficult endeavor. Usually, developers have to write two separate applications for both hemispheres. This is an obviously inefficient approach with multiple problems:

- The overall size of the codebase is doubled.
- Every single line changes when checking code into source control.
- Double the programmers need to be hired (i.e., one team for C and another for Ↄ).)

In code listings 1 and 2 below, we show a simple test case and the traditional two-system development.

Listing 1: Northern Hemisphere example

```
#include <stdio.h>
int main(){
    printf("Hello World!\n");
    return 0;
}
```

```
#include <stdio.h>
int main(){
    printf("Hello World!\n");
    return 0;
}
```

Listing 2: Southern Hemisphere example

These are compiled with GCC and ↃↃG respectively and two executable binaries are produced for distribution.

2 Proposed Method

We developed a novel C-like Domain Specific Language (DSL) which allows us to contain development for each hemisphere inside a shared file. After a developer writes their program in this DSL, they can target a specific latitude co-ordinate where the program is intended to run.

2.1 Prior Work

Thankfully, most code editors these days will preserve both line endings and character orientation. (i.e., '∀' will be saved as 'A' and vice versa depending on the original orientation of the character). This is a huge step forward as keyboards differ between the hemispheres (See Figures 1 and 2) and even with the work of CHUMP, the overhead of continually switching layouts is impossibly confusing to the average software engineer. Re-ordering of lines and letter order is unfortunately still a NP-impossible research problem, yet to work consistently on all but the most trivial examples.

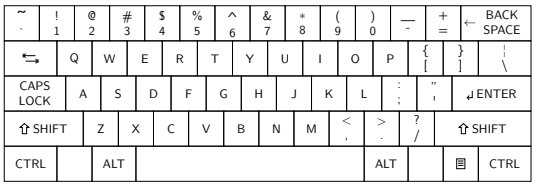

Figure 1: Northern Hemisphere Keyboard

Figure 2: Southern Hemisphere Keyboard

2.2 GCC Preprocessor and comment style

In order to allow both code versions to be written in the one file, there must be some check for certain code to be compiled only when in the target hemisphere. Hence, we place some traditional GCC preprocessing macros like '#if' and '#else'.

However, these are hemisphere-dependent and will fail under ↃↃↃ. These macros be appended with a comment symbol in order for them not be processed. Similar macros for ↃↃↃ are written and corresponding comment symbols so not be processed by GCC.

CHUMP changes the comment style change from '//' as '\\' will not correctly parse when read backwards[1]. In order not to favor one hemisphere over the other, we use the obvious compromise by interpolating the comment styles — '||'.

[1] This is an unfortunate side-effect of text editors saving character orientation!

Listing 3: CombinedHemisphere

```
||                          fidne#
||           CINAP                ||
#if LATITUDE > 0                  ||
||                           esle#
#include <stdio.h>                ||
int main(){                       ||
    printf("Hello World!\n");||
    return 0;                     ||
}                                 ||
#elif LATITUDE < 0                ||
||          0 > EDUTITAL file#
||                               {
||                     ; 0 nruter
||    ;("n\dlroW olleH")ftnirp
||                  }()niam tni
||          <h.oidts> edulcni#
||            0 < EDUTITAL fi#
#else                             ||
||           PANIC                ||
#endif                            ||
```

Of course, || is usually used for a logical OR in C. Another symbol used for this operation in math is '$\vee$'. But, to avoid confusion with '$\wedge$' (logical AND), it was decided to use $\diamond$.

2.3 Shortcomings

It is an unfortunate truth that with every great change, someone will complain. Here are a few messages our first release has received, and our comments:

- "Why not use ¡ƒ elsɹə?"
  - No.
- "Why the name CHUMP and not a derivative name of C, like C++ or C#?"
  - If we wanted to base the name off of a hemisphere-independent name for C, it would look like O which has a high risk of misrepresenting the project as an offshoot of the O programming language[1]. We did mull over being called COↄ, but the pronunciation was an issue.
- "Can you shout me out on your paper?"
  - Sure. Hi Mom.

3 Conclusions & Future Work

We are actively working on a second major version of CHUMP. Please look forward to the following features:

- Support for while-do and ɐlıʞʍ-op loops
- Deprecate "elif" in favour of the more verbose "else if" to avoid conflict with the 'file' system
- "static" to be made default so that code does not move when emailed from one hemisphere to the other.

There are two outstanding limitations of CHUMP, one of which is that CHUMP itself is written in Northern-Hemispheric C and not in CHUMP itself. This has created an undesirable power-dynamic in companies that use CHUMP which we will seek to remove in future versions.

The other issue, which astute readers may have noticed above is the ill-handling of cases at latitude 0. Our working solution is to have several slingshots at the point of zero latitude and launch target platforms across to the other side at such a speed in which no defects will have time to occur.

At some point we hope to have the research funding to expand our work to ↄↄ and ↄↄↄ in order to support users permanently residing at latitude 0 who rely on longitude systems. However, we know there remains an issue for tourists to Station 13010 of Null Island [2] at 0, 0.

References

[1] The O language¶ (no date) The O Language - The O Language 2.0 documentation. Available at: `https://o.readthedocs.io/en/latest/` (Accessed: 29 January 2024).

[2] St. Onge, T. (2016) The geographical oddity of Null Island, The Geographical Oddity of Null Island | Worlds Revealed: Geography & Maps at The Library Of Congress. Available at: `https://web.archive.org/web/20160512183906/http://blogs.loc.gov/maps/2016/04/the-geographical-` (Accessed: 29 January 2024).