Gabriel Gian

# A linguagem C e as vicissitudes da compilação.

Guaxupé, Minas Gerais

2021

Gabriel Gian

# A linguagem C e as vicissitudes da compilação.

Centro Universitário da Fundação Educacional Guaxupé
– UNIFEG

Guaxupé, Minas Gerais
2021

Gabriel Gian
A linguagem C e as vicissitudes da compilação./ Gabriel Gian. – Guaxupé, Minas Gerais, 2021-
13p. : il. (algumas color.) ; 30 cm.

Trabalho – Centro Universitário da Fundação Educacional Guaxupé – UNIFEG, 2021.

1. Linguagens de programação 2. Linguagem de programação C I. UNIFEG

# 1 Introdução

No fim dos anos 70, a AT&T licenciou o sistema operacional Unix desenvolvido nos seus Bell Labs, por Ken Thompson e Dennis Ritchie, no começo daquela mesma década. O desenvolvimento deste sistema, sendo o mesmo praticamente um filho de uma ideia abortada — o sistema de tempo compartilhado Multics — que se mostrou demasiadamente complexo e impraticável em termos de tamanho e tecnologia disponível. O sistema Unix, inicialmente, estava sendo implementado em linguagem assembly para o computador portátil PDP-11, mas o gargalo de complexidade apresentado por este método acabou criando a necessidade de engendrar uma tecnologia que se mostraria ainda mais importante que o próprio Unix.

Era o nascimento da linguagem C.

# 2 Breve história da linguagem C

Desenvolvida por Dennis Ritchie, durante o ciclo de desenvolvimento do Version 4 Unix, a linguagem C nasceu como uma sucessora direta da linguagem *B*, de seu parceiro de desenvolvimento (e Prêmio Turing) Ken Thompson, linguagem que por si própria era uma evolução da sua linguagem anterior, *Bon*. A linguagem buscava ser uma alternativa de alto nível, apresentando um bom grau de abstração em relação ao código de máquina, embora que ainda mantenha um excelente grau de capacidades de manipulação baixo nível — o que torna-a perfeita para o desenvolvimento de sistemas operacionais.

A linguagem C, desde então, foi adotada como uma das linguagens mais amplamente utilizadas, tanto na academia como na indústria, sendo a energia motriz de softwares que vão desde drivers, sistemas embarcados e programas para o processamento veloz de dados requeridos pela matemática computacional, a aplicativos de produtividade, motores de videogames diversos e softwares de controle de aparelhos conectados à dita *Internet das Coisas*.

# 3 "Computação" começa com C

A linguagem C é notória por ser praticamente uma abstração mais legível e com mais implementações de estruturas de dados que uma primitiva linguagem de montagem como a assembly do PDP-11 utilizado por Thompson e Ritchie para a implementação do Unix. A linguagem, sendo inspirada em ALGOL e FORTRAN, apresenta a famosa e influente sintaxe imperativa e procedural com blocos separados por chaves ({}), representando escopos de código como corpos de funções (métodos) e estruturas de controle como blocos *if — else* e *switch* condicionais.

```
i_main:
pushl   %ebpz
movl    %esp, %ebp
subl    $24, %esp
andl    $-16, %esp
movl    $0, %eax
addl    $15, %eax
addl    $15, %eax
shrl    $4, %eax
sall    $4, %eax
movl    %eax, -8(%ebp)
movl    -8(%ebp), %eax
call    __alloca
call    ___main
movl    $0, -4(%ebp)
cmpl    $0, -4(%ebp)
jne L2
movl    $LC0, (%esp)
call    _printf
L2:
movl    $0, %eax
leave
ret
```

```
#include <stdio.h>

int main()
{
    int i = 0;

    if ( i == 0 )
    {
        printf("Hello world!\n");
    }

    return 0;
}
```

Figura 1 – Exemplo de código assembly e o seu equivalente em C. Nota-se a legibilidade impressionante da relação entre as duas — o grande chamativo da linguagem C.

Como se pode ver pelo exemplo acima, a linguagem C é capaz de ser compilada de maneira otimizada (devido aos diversos e maduros compiladores desenvolvidos para implementá-la, como o original *cc* criado por Thompson e Ritchie para o Unix, o moderno *GCC*, do projeto GNU de Richard Stallman, ou a infraestrutura de compilação ainda mais moderna *LLVM*, apoiada pela Apple como concorrente ao GCC, na famosa troca da arquitetura de seus dispositivos de IBM para Intel) para código objeto nativo de praticamente todas as arquiteturas, devido à sua ampla aceitação.

# 4 Futuro

O futuro da linguagem C pode ser discutido por via de diversos pontos de vista. Há pesquisadores e cientistas da computação que acreditam tratar-se de uma tecnologia atemporal, o que de fato pode ser verdade, uma vez que a linguagem C funciona como uma quase perfeita e simples abstração do código de máquina para níveis de código mais legíveis ao ser humano. A ergonomia da linguagem C é indiscutivelmente bela e produtiva. A linguagem C foi capaz de até mesmo chegar ao espaço, uma vez que diversos programas de controle de robôs e satélites são implementados em C devido à sua velocidade graças à otimização eficiente pelos compiladores modernos de C. Até mesmo outras linguagens são implementadas em C, como Ruby e Python, e há ainda implementações de linguagens declarativas mais antigas que o próprio C como a linguagem Lisp, com a implementação CLISP por Bruno Haible e Michael Stoll.

Há, todavia, o ponto de vista contrário, que enxerga a linguagem C como uma tecnologia do passado — suas "dificuldades"sendo enxergadas como empecilhos ineficientes para o programador moderno, acostumado ao gerenciamento de memória automático e às facilidades de depuração e distribuição das linguagens modernas. Com o tempo, o surgimento de linguagens com a pretensão de substituir a linguagem C tem engendrado interessantes candidatos modernos ao posto, tendo a linguagem Rust, ao que tudo indica, a melhor chance de executar tal feito, uma vez que já tem sido usada para implementar e até reescrever (o que não é recomendado pela própria comunidade Rust, em nome do enorme respeito pela linguagem C) aplicações baixo nível originalmente implementadas em C. Sinal maior de que a linguagem Rust está em ascensão neste campo não há senão a notícia de que o kernel Linux tem implementado cada vez mais módulos, drivers e APIs em Rust, seu próprio criador, Linus Torvalds, tendo dito em 2020 que apoia a implementação de partes do código em Rust, embora de maneira parcimoniosa.

# 5 Conclusão

A linguagem C tem percorrido um longo caminho desde a sua concepção nas Bell Labs. Ela possibilitou um salto tecnológico enorme entre os grandes mainframes e microcomputadores monótonos de outrora, o desenvolvimento de sistemas operacionais cada vez mais potentes e úteis, até mesmo de outras linguagens que trouxeram a computação dos grandes centros de tecnologia militar americana às palmas de nossas mãos. É de suma importância reconhecer o valor inestimável que a linguagem possui, mas é também igualmente importante reconhecer que ela deve ser melhorada, não só nominalmente, mas também conceitualmente, mas centenas de linguagens que dela foram engendradas, e que continuarão a carregar seu legado na eterna jornada de desenvolvimento tecnológico da nossa raça.

# 6 Referências bibliográficas


RITCHIE, Dennis. *The development of the C language*. Março 1993. Disponível em: https://dl.acm.org/doi/10.1145/155360.155580. Acesso em 20 março 2021.

JOHNSON, Stephen; RITCHIE, Dennis. (1978). *Portability of C Programs and the UNIX System*. Bell System Tech. J. 57 (6): 2021–2048. DOI: 10.1002/j.1538-7305.1978.tb02141.x. Disponível em: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.138.35. Acesso em 23 março 2021.

*A Brief History of GCC*. Disponível em: https://gcc.gnu.org/wiki/History. Acesso em 23 março 2021.

*The LLVM Compiler Infrastruture*. Disponível em: https://llvm.org. Acesso em 23 março 2021.

LARABEL, Michael. *Linus Torvalds' Initial Comment On Rust Code Prospects Within The Linux Kernel*. Disponível em: https://www.phoronix.com/scan.php?page=news$_i tempx = Linux - Kernel - Rust - PPC64LE$.