.

Homenique Vieira Martins, Rafael Amauri Diniz Augusto

A Linguagem C

Belo Horizonte

2021

Sumário

Lista de tabelas

Introdução

A linguagem C foi criada por Dennis Ritchie e Ken Thompson na década de 1970, enquanto ambos trabalhavam na Bell Labs. A história da linguagem C é fortemente ligada à história de outra linguagem: B. B foi originalmente criada por Thompson como uma versão simplificada da linguagem BCPL que seria usada para facilitar o desenvolvimento de ferramentas para o sistema operacional Unix, que também estava sendo desenvolvido por Thompson e Ritchie na mesma época. Todavia, a linguagem B era muito lenta e não tinha recursos como byte addressability. Por esses motivos, Ritchie começou a se envolver pesadamente no desenvolvimento de B junto com Thompson, e acabou por desenvolver uma nova linguagem: C.

1 História e Cronologia

A versão original de C é conhecida como "K&R C"por programadores. Como a popularidade de C cresceu rapidamente nas décadas de 1970 e 1980, foi criado um comitê para padronizar a linguagem e prover especificações consistentes. Essa versão ficou conhecida como ANSI C, e é a versão mais utilizada da linguagem hoje em dia.

A partir de 1983, quando o padrão ANSI C foi criado, foram adicionadas diversas revisões da linguagem, que adicionam novas funcionalidades e recursos a cada revisão. A última

- Em C99 foram introduzidos suportes para funções inline, suporte a vários tipos novos de dados (como long long int) e suporte para comentários de uma única linha, começando com \\
- Em C11 foram feitas várias novas adições à linguagem, como melhor suporte ao Unicode, multi-threading, tipagem genérica e melhor compatibilidade com C++.
- Em C17 curiosamente não foram introduzidas funcionalidades novas, apenas correções de bugs e outras correções técnicas.

2 Paradigma

Podemos definir paradigma da programação como a forma que uma linguagem é construída, quais são suas regras, suas estruturas e o comportamento esperado. Neste trabalho, iremos discutir sobre o principal paradigma da linguagem C, neste caso, o paradigma Imperativo.

O paradigma imperativo é o mais antigo da história da computação, sendo baseado na estrutura de máquina do cientista Húngaro-americano John Von Neumann. O funcionamento de linguagens imperativas está fortemente ligado ao funcionamento da própria máquina, e por isso linguagens que utilizam essa estratégia tendem a ser mais rápidas, pois o código é uma Transliteração das operações de máquina, ou seja as operações são uma função direta das operações existentes no hardware.

Outros pontos notáveis incluem a execução do código de forma sequencial, com o uso de funções sendo apenas uma ferramenta adicional e não necessariamente obrigatória, salvo o uso da função *main()*.

Linguagens imperativas também são sempre baseadas em comandos, que são *keywords* reservadas da própria linguagem que efetuam operações únicas e servem como instruções para o compilador ou interpretador. Por fim, o paradigma imperativo também consiste no armazenamento de dados alterando as próprias células da memória principal.

Uma linguagem baseada no paradigma imperativo possui como características:

- As variáveis modelam as células de memória
- Os comandos de atribuição, que são baseados nas operações de transferência dos dados e instruções.
- A execução sequencial de instruções.
- A forma iterativa de repetição, que é o método mais eficiente desta arquitetura.

3 Comparações Notáveis

Benchmark

Abaixo pode ser vista uma análise qualitativa da linguagem C em relação às linguagens Java, Python 3 e Node.js. Enquanto Java é uma linguagens executada na JVM e Python 3 e Node são duas linguagens interpretadas, C é uma linguagem compilada e o objetivo deste Benchmark é comparar o tempo de execução e consumo de memória.

Todos os testes foram realizados no seguinte hardware:

Tabela 1 – Configurações

CPU	Intel i5-3330 3.0GHz quad-core
RAM	15.8 GB of RAM
HDD	2TB SATA disk drive
OS	Ubuntu 21.04 x86_64 GNU Linux 5.11.0 − 18−generic

Fonte: <https://benchmarksgame-team.pages.debian.net/benchmarksgame/how-programs-are-measured.html>

Tabela 2 – Resultados do teste busca e escrita em uma árvore binária

Linguagem	Tempo de execução (segundos)	Uso de memória RAM (bytes)
C	1.54	168,832
Java	2.48	1,725,776
Python 3	4.83	462,732
Node.js (Javascript)	7.20	1,282,116

Fonte: <benchmarksgame-team.pages.debian.net/benchmarksgame/performance/binarytrees.html>

Tabela 3 – Resultados do teste de complemento de genoma humano

Linguagem	Tempo de execução (segundos)	Uso de memória RAM (bytes)
C	0.86	712,208
Java	1.53	687,864
Python 3	7.20	1,005,184
Node.js (Javascript)	2.21	1,534,824

Fonte: <https://benchmarksgame-team.pages.debian.net/benchmarksgame/performance/revcomp.html>

4 Linguagens Relacionadas

1. Java

   Similaridades:

   - Sintaxe.
   - Ambas são linguagens imperativas.
   - Permite acesso a ponteiros.
   - Ambas são grandes nomes no mercado de IoT.

   Diferenças:

   - Java compila para bytecode que será usado na JVM. C compila para machine code.
   - C não possui classes, Java possui.
   - C não possui um garbage collector, enquanto Java possui.
   - Por causa da JVM, a portabilidade de Java é bem maior

   Exemplo de código:

```
1  package code;
2
3
4  public class HelloWorld {
5
6      public static void main(String[] args) {
7          System.out.println("Hello World");
8      }
9
10 }
```

2. Python

   Similaridades:

   - Ambas são linguagens com múltiplos paradigmas.
   - Semântica.
   - Possui suporte a tipos dinâmicos e primitivos.

   Diferenças:

   - Python é uma linguagem interpretada.
   - Python não requer uma função main, enquanto C sim.
   - Python não tem ponteiros.
   - Python tem um garbage collector.

   Exemplo de código:

```
1
2  print("Hello World\n")
```

3. C++

   Similaridades:

   - Ambas são multi-paradigma.
   - Ambas permitem manipulação de ponteiros.
   - Ambas não possui garbage collector.
   - Ambas são linguagens compiladas.

   Diferenças:

   - C++ suporta sobrecarga de funções.
   - C++ possui alocação dinâmica na memória.
   - Por ser pesadamente voltada a POO, C++ possui suporte a polimorfismo, herança e encapsulamento.
   - C++ possui suporte direto para exceções.

   Exemplo de código:

```
1 #include <iostream>
2
3
4 int main(int argc, char const *argv[])
5 {
6     std::cout << "Hello World" << std::endl;
7     return 0;
8 }
```

4. Haskell

   Similaridades:

   - Haskell requer uma função main.
   - Ambos aceitam funções como parâmetros de outras funções.
   - Ambas permitem manipulação da memória por ponteiros.
   - Possui suporte a tipos dinâmicos e primitivos.

   Diferenças:

   - Haskell é uma linguagem interpretada.
   - Haskell é uma linguagem funcional.
   - Todos dados em Haskell são imutáveis.
   - Todas funções em Haskell são funções puras.

   Exemplo de código:

```
1 main :: IO ()
2 main = putStrLn "Hello, World!"
```

5 Exemplos de Programas

Exemplo de código em C:

Hello World

```c
#include <stdio.h>

int main()
{
  printf("Hello World!\n");

  return 0;
}
```

Palíndromo

```
%%
%% This is file `.tex',
%% generated with the docstrip utility.
%%
%% The original source files were:
%%
%% fileerr.dtx  (with options: `return')
%%
%% This is a generated file.
%%
%% The source is maintained by the LaTeX Project team and bug
%% reports for it can be opened at https://latex-project.org/bugs/
%% (but please observe conditions on bug reports sent to that address!)
%%
%%
%% Copyright (C) 1993-2021
%% The LaTeX Project and any individual authors listed elsewhere
%% in this file.
%%
%% This file was generated from file(s) of the Standard LaTeX `Tools
   Bundle'.
%%
   ------------------------------------------------------------------------

%%
%% It may be distributed and/or modified under the
%% conditions of the LaTeX Project Public License, either version 1.3c
%% of this license or (at your option) any later version.
%% The latest version of this license is in
%%    https://www.latex-project.org/lppl.txt
%% and version 1.3c or later is part of all distributions of LaTeX
%% version 2005/12/01 or later.
%%
%% This file may only be distributed together with a copy of the LaTeX
%% `Tools Bundle'. You may however distribute the LaTeX `Tools Bundle'
%% without such generated files.
%%
%% The list of all files belonging to the LaTeX `Tools Bundle' is
%% given in the file `manifest.txt'.
%%
```

```
 \message{File ignored}
\endinput
%%
%% End of file `.tex'.
```

Mergesort

```
#include <stdio.h>

void merge_sort(int i, int j, int a[], int aux[]) {
if (j <= i)
{
        return;
  }

int mid = (i + j) / 2;

  merge_sort(i, mid, a, aux);
  merge_sort(mid + 1, j, a, aux);

  int pointer_left  = i;
  int pointer_right = mid + 1;
  int k;

  for(k = i; k <= j; k++)
  {
        if(pointer_left == mid + 1)
        {
aux[k] = a[pointer_right];
              pointer_right++;
        }

else if(pointer_right == j + 1)
        {
          aux[k] = a[pointer_left];
          pointer_left++;
        }
        else if(a[pointer_left] < a[pointer_right])
        {
          aux[k] = a[pointer_left];
          pointer_left++;
        }
        else
        {
          aux[k] = a[pointer_right];
          pointer_right++;
        }
  }

  for (k = i; k <= j; k++)
  {
        a[k] = aux[k];
  }
}


int main()
{
  int array[100], aux[100], n, i, d, swap;
```

```
53
54    printf("Digite numero de elementos no array:\n");
55    scanf("%d", &n);
56
57    printf("Digite %d inteiros\n", n);
58
59    for (i = 0; i < n; i++)
60    scanf("%d", &array[i]);
61
62    merge_sort(0, n - 1, array, aux);
63
64    printf("Printando array ordenado:\n");
65
66    for (i = 0; i < n; i++)
67    {
68          printf("%d\n", array[i]);
69    }
70
71    return 0;
72 }
```

6 Projetos reconhecidos que usam C

Sistemas Operacionais favorecem muito o uso da linguagem C para seu desenvolvimento por causa da proximidade da linguagem com o hardware, o suporte a múltiplas arquiteturas (como x86, x86_64, arm, arm64 e RISC-V),

- Unix, Linux Kernel, MacOS, Windows NT Kernel, FreeBSD Sistemas Operacionais favorecem muito o uso da linguagem C para seu desenvolvimento por causa da proximidade da linguagem com o hardware, o suporte a múltiplas arquiteturas (como x86, x86_64, arm, arm64 e RISC-V), a altíssima qualidade e confiança nos compiladores de C (GCC e Clang principalmente), e alto nível de controle sobre o assembly code que vai ser gerado a partir do código-fonte. Abaixo seguem exemplos de alguns sistemas operacionais que são desenvolvidos com a linguagem C.

- AndroidOS, iOS importam muito do desenvolvimento que foi feito no Linux e no MacOS, respectivamente. A portabilidade de código em C é um dos motivos para essa escolha, já que compilar para outra arquitetura em C muitas vezes envolve apenas adicionar uma flag de compilação.

- Arduino O suporte aos mais diversos tipos de dispositivos é essencial para trabalhar com Arduino, e o suporte de C a diferentes dispositivos provê justamente essa questão de portabilidade e suporte. O hardware dentro do Arduino também é comparativamente fraco quando comparado a um PC ou um celular, então utilizar uma linguagem compilada que minimiza o overhead de runtime e consumo de memória é um ponto importante.

- PostgreSQL, MySQL, Oracle Database Fazer consultas em bancos de dados é uma operação extremamente custosa, e websites e aplicações muitas vezes precisam dos resultados dessas operações para mostrar algum elemento, então velocidade de tempo de execução e minimizar o gasto de memória são os fatores mais importantes na hora de fazer esses programas. Por conta do controle excelente sobre memória que a linguagem C oferece e pela proximidade com o hardware, C é uma excelente opção para o desenvolvimento de gerenciadores de bancos de dados.

- Blender Blender é uma aplicação que também precisa de velocidade e eficiência, já que ela lida com enormes quantidades de dados e fazem muitos cálculos por segundo. Quanto mais eficientes elas.

- Id Tech 1, mais conhecida como Doom Engine

  Criada por John Carmack, a Doom Engine foi uma engine revolucionária pois criava uma ilusão de 3D nunca vista antes, além de estabelecer novos paradigmas para o desenvolvimento de jogos. Usando a eficiência de C, fórmulas simplificadas reduzidos muitos o custos computacionais e criando uma engine leve e eficiente.

Considerações finais

Após este trabalho, o grupo entendeu melhor o enorme peso que a linguagem C tem na história e no mercado de programação. A linguagem possui uma comunidade gigantesca que se mantém unida até quase meio século após sua criação, e isso se deve ao lugar único que a linguagem C ocupa no mercado de desenvolvimento de sistemas altamente eficientes.

Por 49 anos a linguagem C tem se aprimorado e amadurecido além da visão original de seus criadores, além de ter juntado uma das maiores comunidades de linguagens de programação. Essa comunidade transformou C em uma linguagem unificada e padronizada, o que facilita o desenvolvimento nela, promove a longevidade da linguagem e a aprimora.

Por fim, o grupo sente que C desempenha um papel quase insubstituível no mercado de aplicações altamente eficientes e onde gerenciamento eficiente de memória é obrigatório. C é predominante em qualquer desenvolvimento que tenha essas prioridades, e a melhor linguagem a ser utilizada atualmente.

Referências

CAMPOS, A. *RETROCOMPATIBILIDADE: Doom)*. Disponível em: <https://www.youtube.com/watch?v=BV8jmxwSS_E)>. Acesso em: 19 September 2021. Nenhuma citação no texto.

CARMACK, J. *Doom Engine*. Disponível em: <https://github.com/id-Software/DOOM>. Acesso em: 19 September 2021. Nenhuma citação no texto.

DESCONHECIDO. *Blender)*. Disponível em: <https://www.arduino.cc/en/Guide/Introduction)>. Acesso em: 19 September 2021. Nenhuma citação no texto.

DESCONHECIDO. *Blender)*. Disponível em: <https://www.blender.org/)>. Acesso em: 19 September 2021. Nenhuma citação no texto.

DESCONHECIDO. *C language*. Disponível em: <https://en.cppreference.com/w/c/language>. Acesso em: 19 September 2021. Nenhuma citação no texto.

DESCONHECIDO. *C (programming language)*. Disponível em: <https://en.wikipedia.org/wiki/C_(programming_language)>. Acesso em: 19 September 2021. Nenhuma citação no texto.

DESCONHECIDO. *Id Tech)*. Disponível em: <https://en.wikipedia.org/wiki/Id_Tech)>. Acesso em: 19 September 2021. Nenhuma citação no texto.

DESCONHECIDO. *Imperative programming)*. Disponível em: <https://en.wikipedia.org/wiki/Imperative_programming)>. Acesso em: 19 September 2021. Nenhuma citação no texto.

DESCONHECIDO. *Imperative programming: Overview of the oldest programming paradigm)*. Disponível em: <https://www.ionos.com/digitalguide/websites/web-development/imperative-programming/)>. Acesso em: 19 September 2021. Nenhuma citação no texto.

DESCONHECIDO. *macOS*. Disponível em: <https://en.wikipedia.org/wiki/MacOS_version_history>. Acesso em: 19 September 2021. Nenhuma citação no texto.

DESCONHECIDO. *Paradigma imperativo)*. Disponível em: <https://www.inf.pucrs.br/~gustavo/disciplinas/pli/material/paradigmas-aula09.pdf)>. Acesso em: 19 September 2021. Nenhuma citação no texto.

DESCONHECIDO. *Unix)*. Disponível em: <https://en.wikipedia.org/wiki/Unix)>. Acesso em: 19 September 2021. Nenhuma citação no texto.

GOOGLE. *Android*. Disponível em: <https://source.android.com/>. Acesso em: 19 September 2021. Nenhuma citação no texto.

RITCHIE, D. M. *The Development of the C Language**. Disponível em: <http://csapp.cs.cmu.edu/3e/docs/chistory.html>. Acesso em: 19 September 2021. Nenhuma citação no texto.

TORVALDS, L. *Kernel Linux*. Disponível em: <https://github.com/torvalds/linux>. Acesso em: 19 September 2021. Nenhuma citação no texto.

Apêndices

APÊNDICE A – Homenique Vieira Martins

C é uma linguagem muito incrível, nos seus quase 50 anos, suas estrutura e vasta documentação mostra a maturidade que os desenvolvedores foram criando ao longos dos anos no desenvolvimento da linguagem e começa a ficar mais claro como uma linguagem tão velha possui atualizações até nos dias de hoje, continua sendo muito utilizada e porque é um marco tão grande para a computação.

Além disso,estudando para esse trabalho, é notável a detalhada documentação de C, onde no final você aprende ainda mais sobre C.

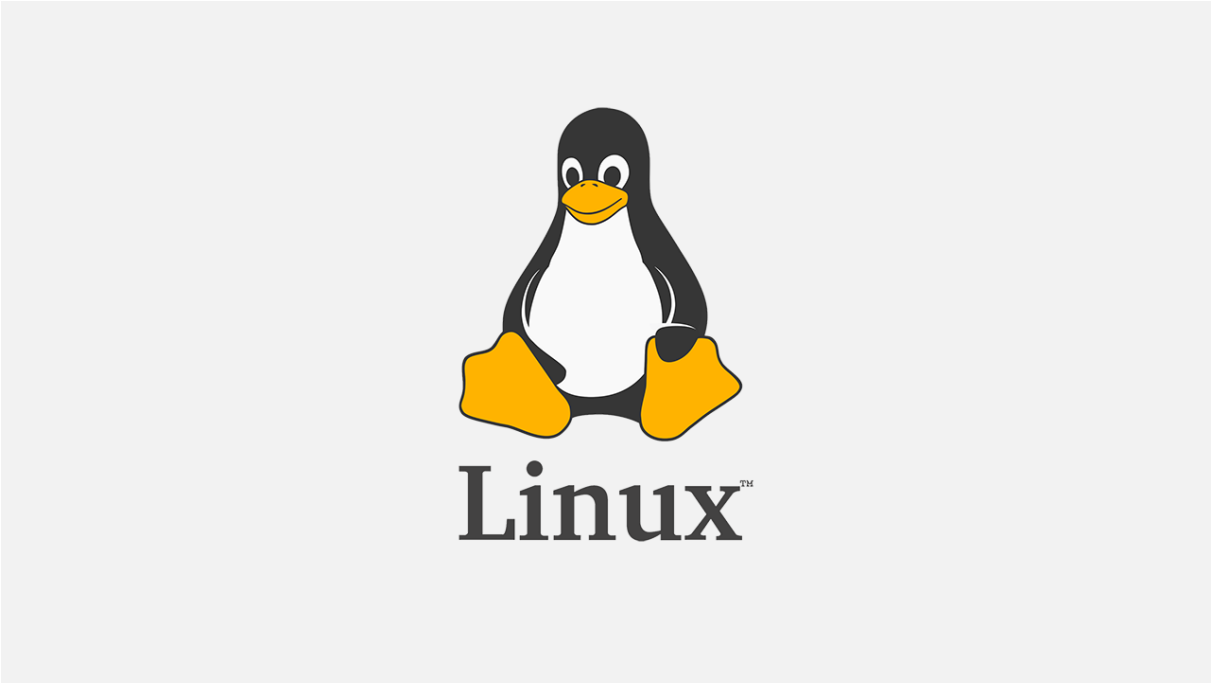
Figura 1 – Doom Engine

Fonte: <https://images.ctfassets.net/rporu91m20dc/7bt6yTABm9pfXTyfbjrqb2/a78a456901b75994349247c260d8b48d/doom--1993--hero-img?q=70&fm=webp>

APÊNDICE B – Rafael Amauri Diniz Augusto

A linguagem C é um marco gigantesco na história da computação. Sem ela, muitas das tecnologias que temos hoje em dia não existiriam ou seriam diferentes ao ponto de serem irreconhecíveis. A presença de C no mercado é gigantesca até 49 anos depois da sua criação, e isso se deve à sua portabilidade e eficiência. C é uma linguagem extremamente próxima de machine-language ao mesmo tempo que tem suporte universal para os mais diversos tipos de arquiteturas, e tudo isso faz ela ser uma linguagem extremamente rápida e portátil.

Figura 2 – Github da Kernel de linux



Fonte: <https://digitalinnovation.one/artigos/conheca-a-historia-do-linux>