Métodos e Técnicas de Programação
.:: Breve histórico e introdução ao C (parte 2) ::.

Prof. Igor Peretta

2018-2

Contents

1 Programação

1.1 Camadas de software

Embora o hardware (parte física de um computador) possa ser resumido em uma máquina de von Neumann, o software possui várias camadas até chegar no usuário:

1. A maioria dos computadores pessoais se inicia com um programa armazenado em memória não-volátil (ROM) chamado BIOS (Basic Input/Output System) que inicia e testa os componentes de hardware da máquina e procura o Sistema Operacional, geralmente o disco rígido (HD);

2. O Sistema Operacional então toma o controle do computador e passa, após seu carregamento na memória volátil (RAM) e sua inicialização, a revezar sua execução com a de outros programas, como se vigiando, controlando e orquestrando todo o processo. Possui as seguintes funções: gerenciamento de processos, gerenciamento de memória, sistema de arquivos, entrada e saída de dados;

3. Através de interfaces, o usuário pode controlar o computador ao enviar comandos ao Sistema Operacional. As interfaces mais conhecidas hoje são a [i.] interface de terminal (CLI), [ii.] a interface textual (se tornou rara hoje em dia), [iii.] a mais comum, a interface gráfica (GUI), [iv.] e as interfaces naturais (voz, gestos, sinais cerebrais, etc);

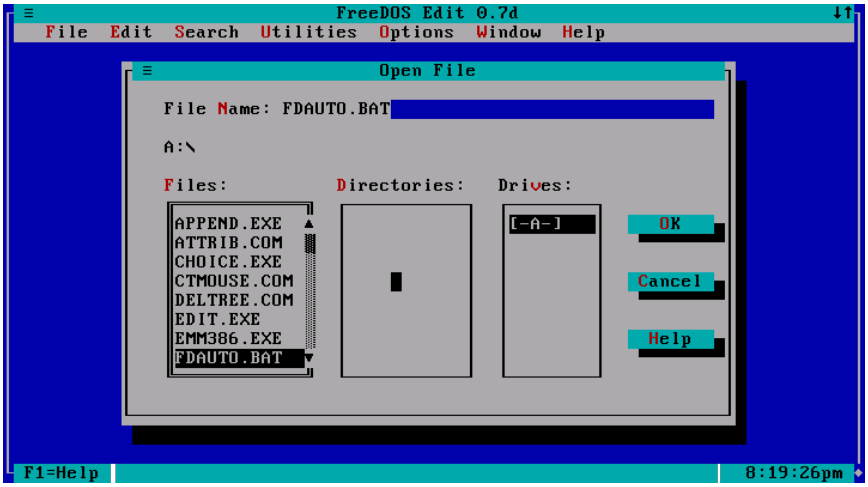

Figure 1: Exemplo de inteface textual

1. Da interface utilizada, o usuário pode chamar os programas dedicados que forem de seu desejo ou de sua necessidade. Daí, o conceito de computação de uso geral.

1.2 Paradigmas de programação

Inicialmente, os computadores eram programados através de código binário com o uso de cartões perfurados. Tal processo era difícil e propício a erros, e é conhecido por código de máquina. Cada processador possui um conjunto de instruções que podem ser diferentes, até mesmo no número de "bits" que utilizam para as mesmas (ex. palavras com 12, 16, **32**, 48, **64** bits).

Para facilitar a programação, foram criadas linguagens de montagem (ASSEMBLY). Substituiram as funções do código de máquina por mnemônicos, endereços de memória absolutos referenciados por identificadores.

O próximo passo foram as linguagens procedimentais (ex. COBOL, FORTRAN, ALGOL). Descrevem passo a passo o procedimento a ser seguido para

Figure 2: Exemplo de interface natural (gestos e voz)

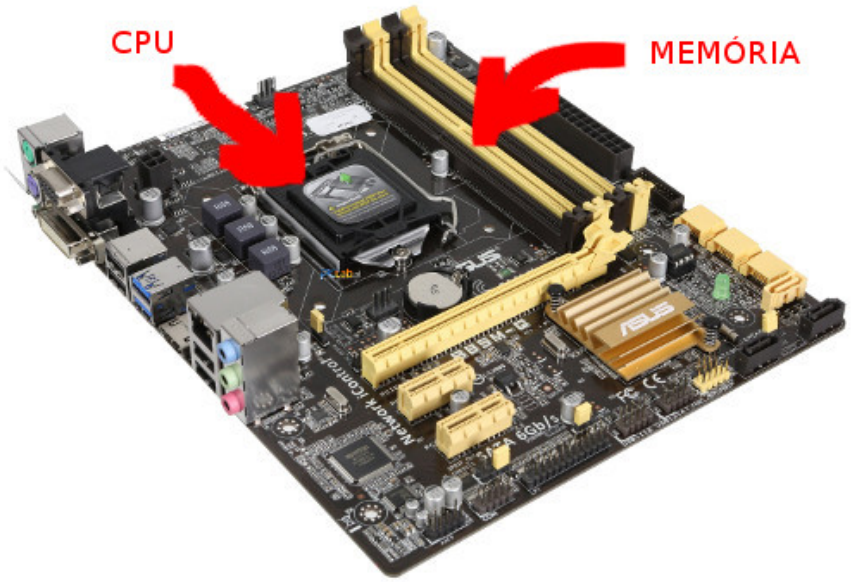

Figure 3: Exemplo de placa mãe

Figure 4: Exemplo de processador, conhecido como CPU

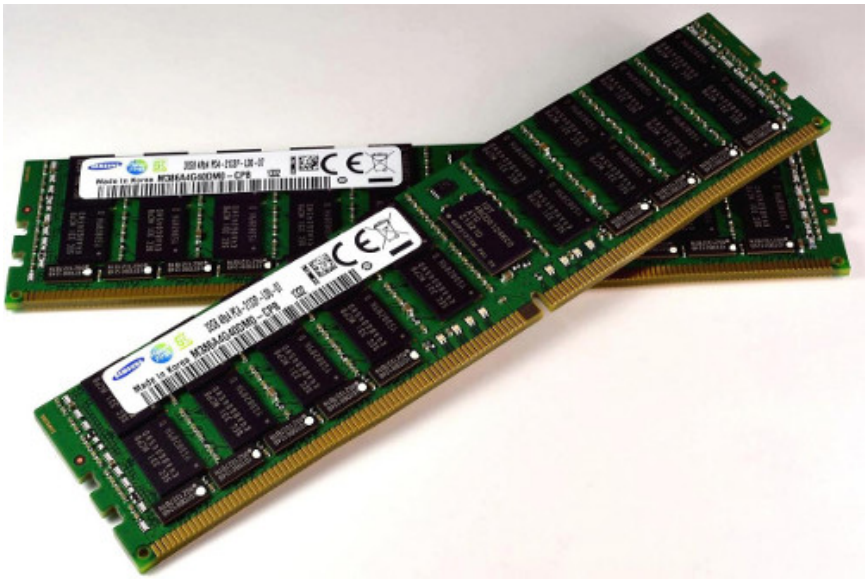

Figure 5: Exemplo de pentes de memória (DDR4)

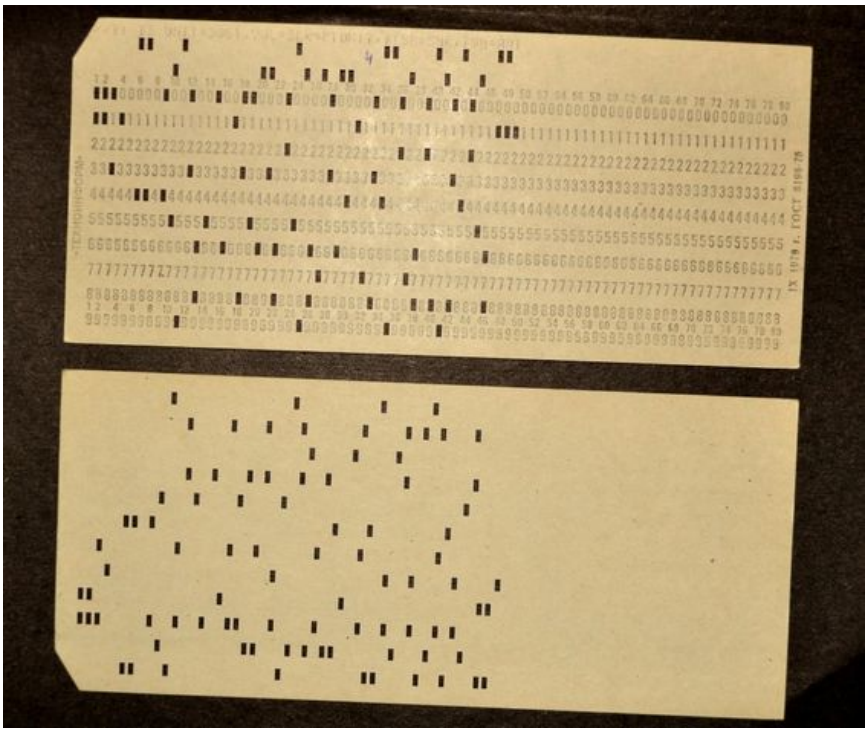

Figure 6: Cartão perfurado para programação

Figure 7: Coleção de cartões perfurados (em torno de 62500), com 5Mb de programa

resolver certo problema e foram desenvolvidas principalmente pra problemas comerciais ou científicos. Sua eficácia e eficiência de solução dependem da experiência, da habilidade e da criatividade do programador.

Após, vieram as linguagens orientadas a objeto. Nelas, os dados e as rotinas para manipulá-los dão mantidos numa unidade "objeto". O programador só pode acessá-los através de subrotinas chamadas "métodos" que permite alterar o funcionamento interno do objeto sem afetar o código que o consome.

Independente desse ramo das linguagens imperativas, baseadas nas linguagens procedimentais, adveio o conceito de programação declarativa. Nessas linguagens, o problema é descrito ao computador, mas não como resolvê-lo. O programa é estruturado como uma coleção de propriedades para se encontrar o resultado esperado e o computador tenta encontrar a solução ao casar todas as propriedades desejadas (ex. SQL, a família das linguagens funcionais e lógicas).

Linguagens de programação funcional usam funções, blocos de código construídos para agir como funções matemáticas. Caracterizam-se pelo desencorajamento da mudança do valor de variáveis através de atribuição e fazem grande uso de recursão.

Na programação lógica são expressados como fórmulas lógicas fatos sobre o domínio do problema e os programas são executados ao se aplicar regras de inferência nas fórmulas até que uma resposta, ou uma coleção delas, é encontrada.

Os computadores modernos são praticamente todos baseados na arquitetura de von Neumann. Isso gera uma extrema compatibilidade com o paradigma imperativo de programação em seus níveis mais essenciais. Todos os outros paradigmas advém da necessidade de abstração do ser humano e do desejo de aumentar o nível de "comunicação" com a máquina, possibilitando grandes computações com o mínimo de código.

1.3 Compiladores

Compilador é um programa de computador (ou grupo de programas) que, a partir de um código fonte escrito em uma linguagem compilada, cria um programa semanticamente equivalente escrita em código objeto. Em geral, um compilador traduz de uma linguagem textual para código de máquina.

Não importa o paradigma ou a linguagem de programação, o fluxo de programa é o mesmo:

1. O código terá entrada em um editor de texto;

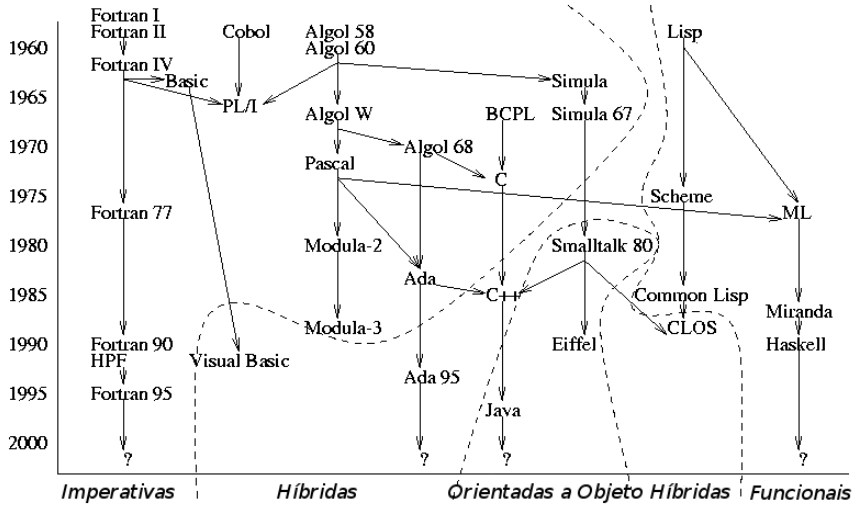

Figure 8: Genealogia das linguagens de programação

2. As bibliotecas e códigos de reuso serão tratados por um "linkador";

3. A gramática e a semântica serão analizados por um compilador ou um interpretador (depende da linguagem de programação) e será convertido em código de montagem (baixo nível, mas ainda independente de hardware);

4. Na parte final, será gerado um executável em código de máquina (dependende de SO e hardware).

2 A linguagem C

C é uma linguagem de programação compilada de propósito geral, estruturada, imperativa, procedural, padronizada pela ISO, criada em 1972, por Dennis Ritchie, no AT&T Bell Labs, para desenvolver o sistema operacional Unix.

A partir de 1990, os compiladores C precisam seguir um padrão mínimo de funcionalidades e definições cuja normatização é conhecida por ISO/IEC 9899:1990, ANSI-C (American National Standards Institute), ou simplesmente C90. Algumas funcionalidades, no entanto, não são cobertas pelas normas e são dependentes de implementação por cada compilador.

Para ser considerada procedural, uma linguagem de programação deve

suportar a especificação de tipos de argumentos, variáveis locais, chamadas recursivas e o uso de procedimentos em módulos distintos de um programa. Ela também pode suportar a distinção entre argumentos de entrada e de saída.

A programação imperativa é um paradigma de programação que descreve a computação como ações, enunciados ou comandos que mudam o estado (variáveis) de um programa.

Programação estruturada é uma forma de programação de computadores que preconiza que todos os programas possíveis podem ser reduzidos a apenas três estruturas: **sequência**, **seleção** e **iteração** (ou repetição) e ao uso extensivo de sub-rotinas.

```
int main(int argc, char** argv) {
    int x, y, z, i;
    x = y = 0;
    if(argc > 2) {
x = (argv[1][i] - 48);
y = (argv[2][i] - 48);
    }
    z = x + y;
    if(z >= 256)
return -1;
    else
return z;
}
```

Figure 9: **Código e execução:** Meu primeiro programa

```
gcc -o exemplo exemplo.c
./exemplo 5 6 || echo $?
objdump -D -b binary -m i8086 exemplo >> exemplo_asm.txt

   ./read_file.c

hexdump exemplo
./read_file exemplo >> exemplo_bin.txt

int main(int argc, char** argv) {
    int x, y, z, i;
    x = y = 0;
```

```
    if(argc > 2) {
i = 0;
while(argv[1][i]) {
    x = x*10 + (argv[1][i] - 48);
    i++;
}
i = 0;
while(argv[2][i]) {
    y = y*10 + (argv[2][i] - 48);
    i++;
}
    }
    z = x + y;
    if(z >= 256)
    return -1;
else
return z;
}
```

Figure 10: **Código e execução:** ampliando utilização

```
#include <stdio.h>
#include <time.h>
int main () { // sequência
  srand(time(0));
  int i, j;
  for(i = 0; i < 5; i++) { // iteração
    j = rand()%2;
    if(j == 0) /* seleção */
      printf("%i X; ",i);
    else
      printf("%i Y; ",i);
  }
  return 0;
}
```

Figure 11: **Código e execução:** as três estruturas de C comandando o fluxo de execução do código

```
0 X; 1 Y; 2 X; 3 X; 4 Y;
```