

Mutirão C

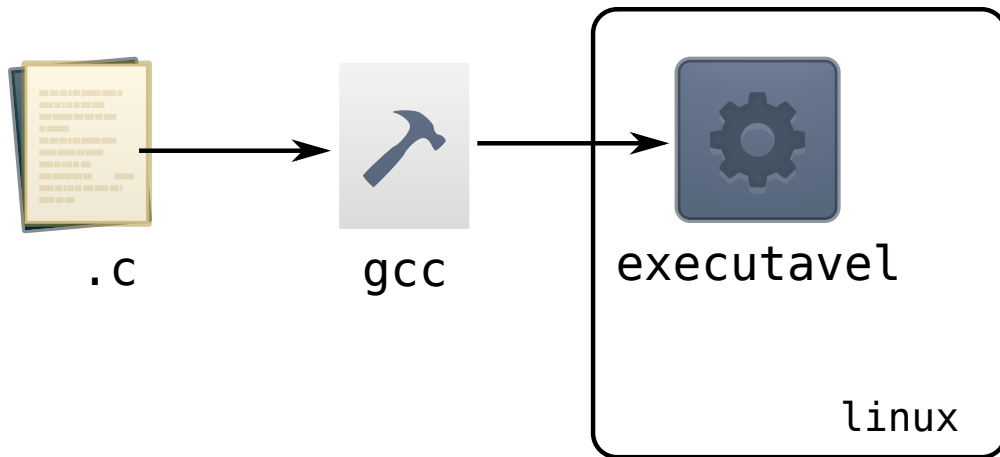
Arrays, Strings e Matrizes

Igor Montagner, Rafael Corsi, Marcelo Hashimoto

Revisão da leitura

Compilação

Compilar um programa significa transformar o código fonte (texto) nas instruções de máquina (bytes) equivalentes.



Específico para

- arquitetura (ARM - embarcados e celulares, x64 - PCs convencionais)
- sistema operacional (Linux, Windows, Android)

Comandos para compilação

Usaremos Ubuntu 20.04 nas disciplinas Sistemas Hardware-Software e Desafios de Programação.

Utilização do `gcc` na linha de comando:

1. `gcc -Wall -pedantic -Og -c arq1.c -o nome_executavel` (para compilar)
2. `./executavel` (para executar)

Vetores unidimensionais

Sintaxe

- Declaração:

```
long vetor[100]; /* Tamanho FIXO */  
long vetor2[] = {1, 2, 3, 4, 5}; /* Inicialização. */
```

- Acesso a elemento:

```
scanf("%ld", &vetor[0]); /* vetor[pos] */
```

- Uso como parâmetro de função:

```
void funcao1(long vetor[], int tamanho);  
void funcao2(long vetor[100]); /* tamanho FIXO */
```

Boas práticas

Evite números mágicos no seu código! Use macros!

```
#define MAX_SIZE 100  
  
long vec[MAX_SIZE];  
  
...  
  
for(i = 0; i < MAX_SIZE; i++) {  
    ...  
}
```

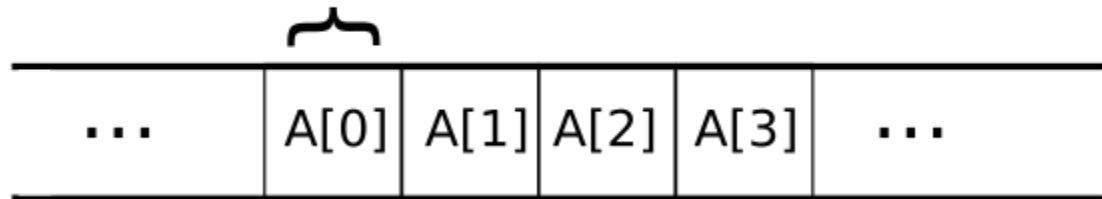
- Por que isto é uma boa ideia?

Organização na memória

Os dados de um vetor são organizados sequencialmente, *sem intervalos*

```
long A[4];
```

8 bytes



Exemplo

```
/* no topo do arquivo*/
#define SZ 5
/* dentro do main*/
long vec[SZ];

for (int i = 0; i < SZ; i++) {
    scanf("%ld", &vec[i]);
}

long s = 0;
for (int i = 0; i < SZ; i++) {
    s += vec[i];
}

printf("Soma eh par: %ld\n", s % 2);
```

Escopo de variáveis "normais"

Qual é o resultado deste código?

```
void soma_um(long n) {  
    n += 1;  
}  
  
int main(int argc, char *argv[]) {  
    long n = 10;  
    soma_um(n);  
    printf("%ld\n", n);  
}
```

Escopo de arrays

Qual é o resultado deste código?

```
void soma_um(long arr[], long pos) {  
    arr[pos]+=1;  
}  
  
int main(int argc, char *argv[]) {  
    long arr[] = {1, 2, 3, 4, 5, 10};  
    soma_um(arr, 5);  
    printf("arr[5] %ld\n", arr[5]);  
}
```

Parâmetros: passagem por valor ou referência

Passagem por valor: `long n` é uma **cópia** do valor passado.

```
void soma_um(long n) {  
    n += 1;  
}
```

Passagem por referência: `long arr[]` é o **mesmo objeto** da função chamadora.

```
void soma_um(long arr[], long pos) {  
    arr[pos] += 1;  
}
```

Atividade 1 - Arrays

Exercícios da primeira seção do Lab 4.

Tempo: 20 minutos

Strings

Codificação

- Cada caractere é representado por um número inteiro de 8 bits (`char`)
- Uma string é um array de `char`
- **Tabela de Codificação:** inteiro → caractere

| Dec | Hex | Oct | Chr | Dec | Hex | Oct | HTML | Chr | Dec | Hex | Oct | HTML | Chr | Dec | Hex | Oct | HTML | Chr |
|-----|-----|-----|---------------------|-----|-----|-----|--------|-------|-----|-----|-----|--------|-----|-----|-----|-----|--------|-----|
| 0 | 0 | 000 | NULL | 32 | 20 | 040 | | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | Start of Header | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | Start of Text | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | End of Text | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | End of Transmission | 36 | 24 | 044 | $ | \$ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | Enquiry | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | Acknowledgment | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | Bell | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | Backspace | 40 | 28 | 050 | (| (| 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | Horizontal Tab | 41 | 29 | 051 |) |) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | Line feed | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | Vertical Tab | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | Form feed | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | Carriage return | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | Shift Out | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | Shift In | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | Data Link Escape | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | Device Control 1 | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | Device Control 2 | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | Device Control 3 | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | Device Control 4 | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | Negative Ack. | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | Synchronous idle | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | End of Trans. Block | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | Cancel | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | End of Medium | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | Substitute | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | Escape | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [| [| 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | File Separator | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | |
| 29 | 1D | 035 | Group Separator | 61 | 3D | 075 | = | = | 93 | 5D | 135 |] |] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | Record Separator | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | Unit Separator | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | | Del |

Sintaxe

Caractere: variável do tipo `char` que contém um inteiro da tabela ASCII.

```
'A' == 65 , '0' == 48
```

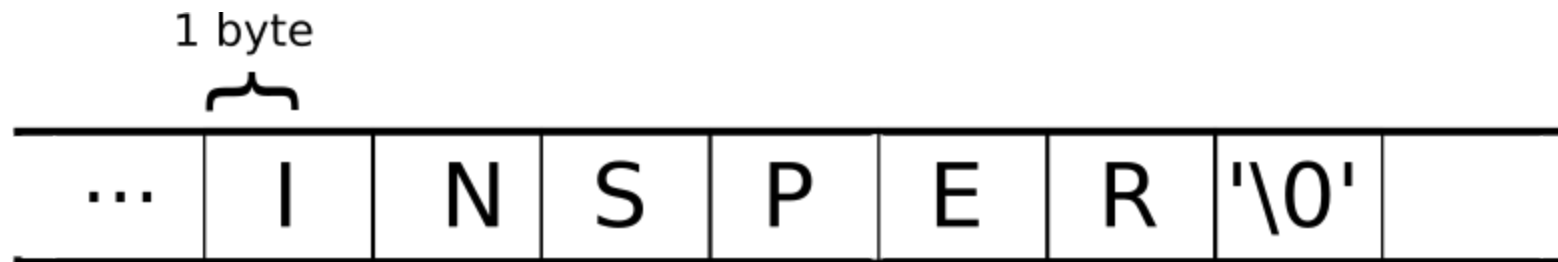
String: vetor de `char` cuja última posição contém o caractere especial `'\0'`.

```
char string[] = "bla bla bla"; /* Inicializado */  
char cmd[100]; /* 99 caracteres, string inválida! */
```

Organização na memória

Os dados de uma string são organizados sequencialmente, *sem intervalos*, sendo que o último elemento é sempre o caractere `'\0'`.

```
char str[] = "INSPER";
```



Tamanho de string

Funciona exatamente como um vetor, com a diferença que **uma string sabe seu tamanho atual, mas não sabe seu tamanho máximo.**

```
int string_length(char str[]) {  
    int n = 0;  
    while (str[n] != '\0') {  
        n++;  
    }  
    return n;  
}
```

A string não precisa ocupar todas as posições do array! Ela acaba no caractere `'\0'`.

Entrada e saída

Impressão:

```
char str[10] = "world!";  
printf("hello %s\n", str);
```

Leitura:

```
char str[10];  
// precisamos passar o tamanho máximo + 1 da nossa string.  
fgets(str, 10, stdin);
```

Atividade 2 - Strings

Exercícios da segunda seção do Lab 4.

Tempo: 20 minutos

Matrizes

Sintaxe

- Declaração:

```
long mat[100][200];  
long mat2[2][3] = {{1, 2, 5}, {3, 4, 6}} ;
```

- Acesso a elemento:

```
printf("%ld\n", mat2[0][2]); /* 5 */
```

- Uso como parâmetro de função:

```
void funcao1(long vetor[100][200]);
```

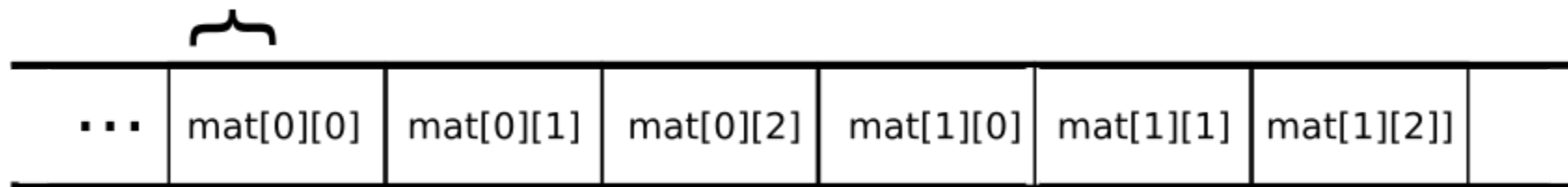
É obrigatório colocar a última dimensão! A primeira é facultativa, mas recomendamos colocar também.

Organização na memória

Os dados de uma matriz são armazenados *linha a linha*

```
long mat[2][3];
```

8 bytes



É preciso conhecer a segunda dimensão para acessar os elementos!

Atividade 3 - Matrizes

Exercícios da terceira seção do Lab 4.

Tempo: 20 minutos

Exercícios

Os exercícios da seção final do Lab 4 são mais avançados e representam o nível esperado de complexidade ao fim do mutirão. Não deixe de fazê-los!