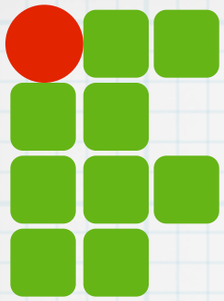
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Algoritmos

ANSI C - Introdução

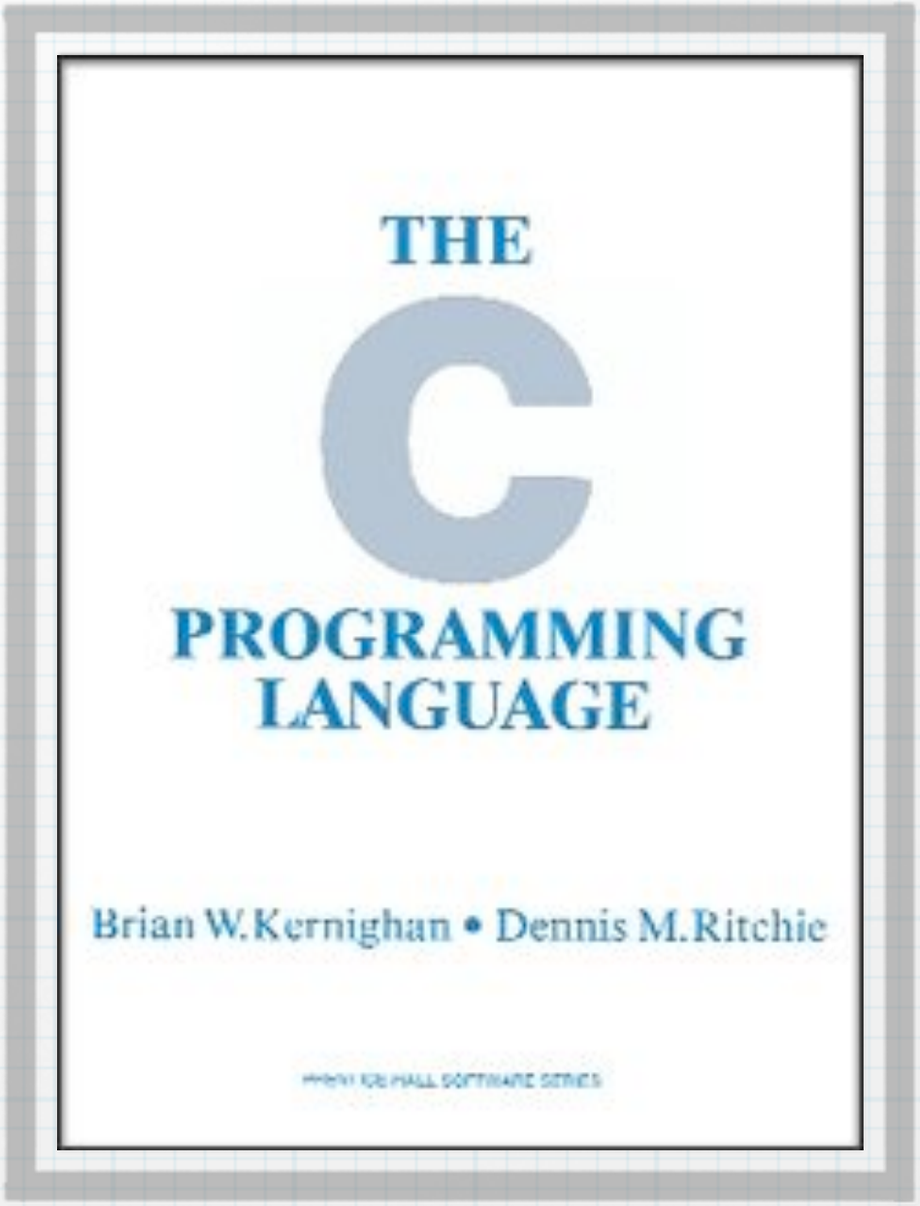


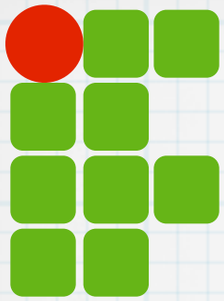
Agenda

* Conceitos básicos
* ANSI C
* Hello World
* Funções em C
* Exercícios

A linguagem C

* Linguagem de propósito geral
* Desenvolvida entre 1969 e 1973
* ANSI C publicado em 1989
  * Tornou-se padrão ISO em 1990
* Usada para desenvolver sistemas operacionais





Linguagem de programação

* Conjunto de palavras (keywords) e regras a serem usadas na descrição de um programa
  * Palavras (keyword)
    * `return`, `while`, `if`, `for`
  * Regras
    * Atribuição: `VAR=EXPR`
    * Condicional: `if (EXPR) then BLOCO end`

Programa de computadores

* Conjunto de instruções que o computador é capaz de processar

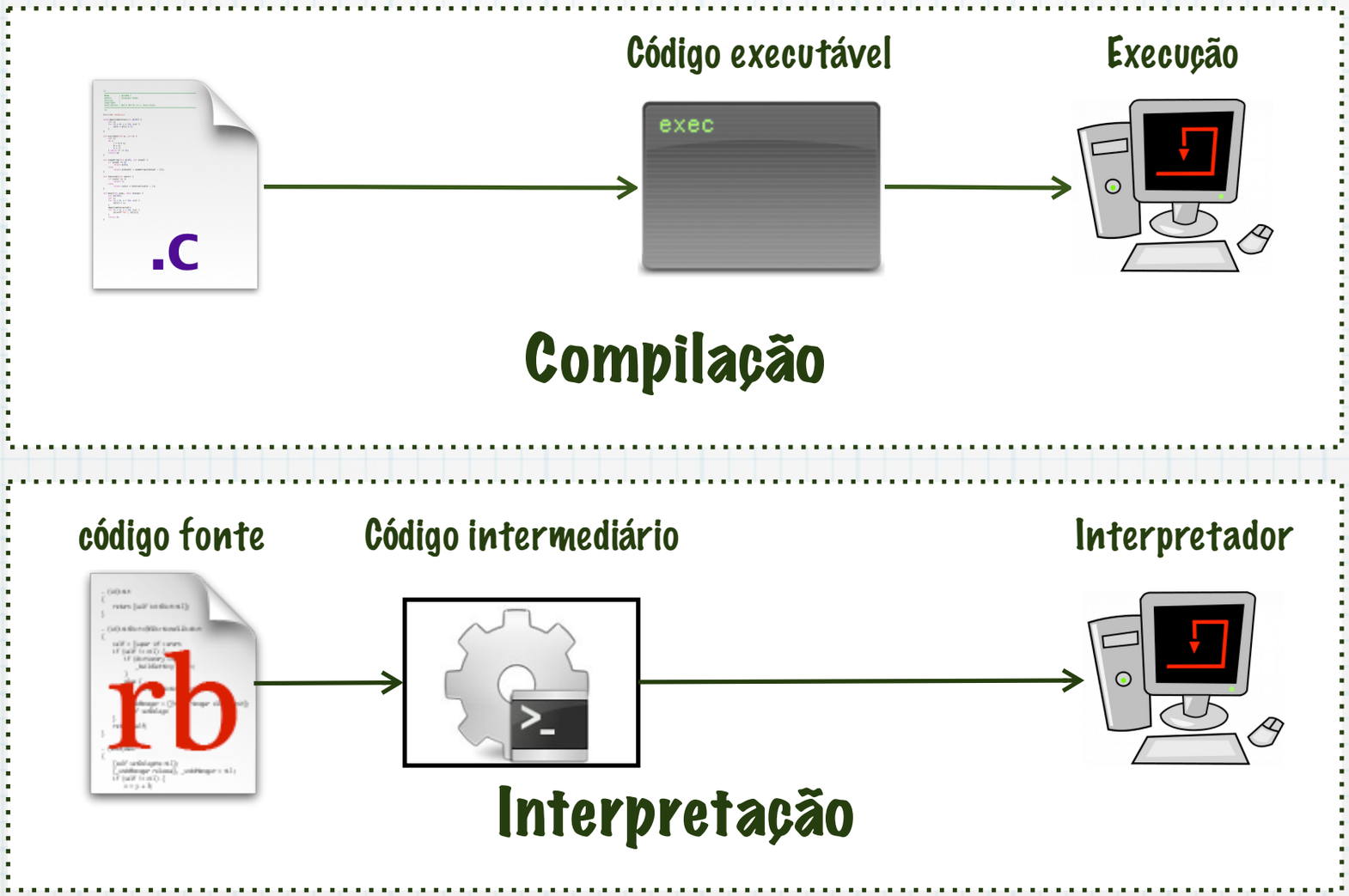
Programa em C

```c
int main(int a, int b){
      int soma,media;
      soma = a+b;
      media = soma/2;
      return media;
   }
```

Compilação e interpretação

- Compilação
  - Traduz o programa em (LM) Linguagem de Máquina
  - Executa-se o programa em LM: Mais rápido
  - Código gerado nativo do processador
- Interpretação
  - Um programa (interpretador) processa as instruções
  - Execução mais lenta
  - Necessário ter interpretador para executar

Compilação e interpretação



A linguagem C

- Um programa em C
  - Conjunto de funções/procedimentos
  - Conjunto de variáveis globais
  - Ponto de início é a função `main`
    - Retorna inteiro
    - Possui dois parâmetros
      - `int`: quantidade de parâmetros
      - `*char[]`: array com os parâmetros (string)

```c
int main(int argc, char ** argv) {
   //Aqui vai o programa
   return 0;
}
```

A linguagem C

- Um programa em C
  - Conjunto de funções/procedimentos
  - Conjunto de variáveis globais
  - Ponto de início é a função `main`
    - Retorna inteiro
    - Possui dois parâmetros
      - `int`: quantidade de parâmetros
      - `*char[]`: array com os parâmetros (string)

Função principal

```
int main(int argc, char ** argv) {
   //Aqui vai o programa
   return 0;
}
```

A linguagem C

- Um programa em C
  - Conjunto de funções/procedimentos
  - Conjunto de variáveis globais
  - Ponto de início é a função `main`
    - Retorna inteiro
    - Possui dois parâmetros
      - `int`: quantidade de parâmetros
      - `*char[]`: array com os parâmetros (string)

Função principal

Parâmetros da função principal

```c
int main(int argc, char ** argv) {
   //Aqui vai o programa
   return 0;
}
```

A linguagem C

- Um programa em C
  - Conjunto de funções/procedimentos
  - Conjunto de variáveis globais
  - Ponto de início é a função `main`
    - Retorna inteiro
    - Possui dois parâmetros
      - `int`: quantidade de parâmetros
      - `*char[]`: array com os parâmetros (string)

Tipo de retorno da função

Função principal

Parâmetros da função principal

```c
int main(int argc, char ** argv) {
    //Aqui vai o programa
    return 0;
}
```

A linguagem C

- Um programa em C
  - Conjunto de funções/procedimentos
  - Conjunto de variáveis globais
  - Ponto de início é a função `main`
- Retorna inteiro
- Possui dois parâmetros
  - `int`: quantidade de parâmetros
  - `*char[]`: array com os parâmetros (string)

Tipo de retorno da função

Função principal

Parâmetros da função principal

```
int main(int argc, char ** argv) {
    //Aqui vai o programa
    return 0;
}
```

Fim da função

Início da função

A linguagem C

- Um programa em C
  - Conjunto de funções/procedimentos
  - Conjunto de variáveis globais
  - Ponto de início é a função `main`
    - Retorna inteiro
    - Possui dois parâmetros
      - `int`: quantidade de parâmetros
      - `*char[]`: array com os parâmetros (string)

```c
int main(int argc, char ** argv) {
  //Aqui vai o programa
  return 0;
}
```

Tipo de retorno da função

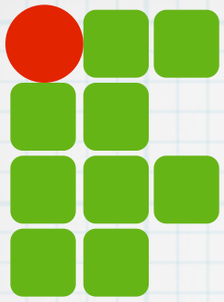
Função principal

Parâmetros da função principal

Fim da função

0 indica que o programa terminou corretamente

Início da função

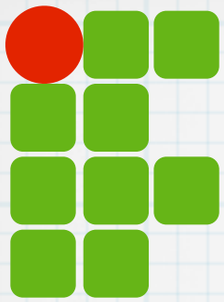


A linguagem C

* Programa "Hello world!"

```c
#include <stdio.h>

int main(int argc, char **argv) {
   printf("Hello World!");
   return 0;
}
```



A linguagem C

* Programa "Hello world!"

Biblioteca para funções de entrada e saída (IO)

```c
#include <stdio.h>

int main(int argc, char **argv) {
   printf("Hello World!");
   return 0;
}
```

A linguagem C

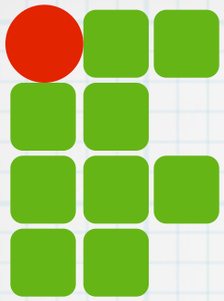
* Programa "Hello world!"

Biblioteca para funções de entrada e saída (IO)

Função para imprimir string formatada

```c
#include <stdio.h>

int main(int argc, char **argv) {
    printf("Hello World!");
    return 0;
}
```

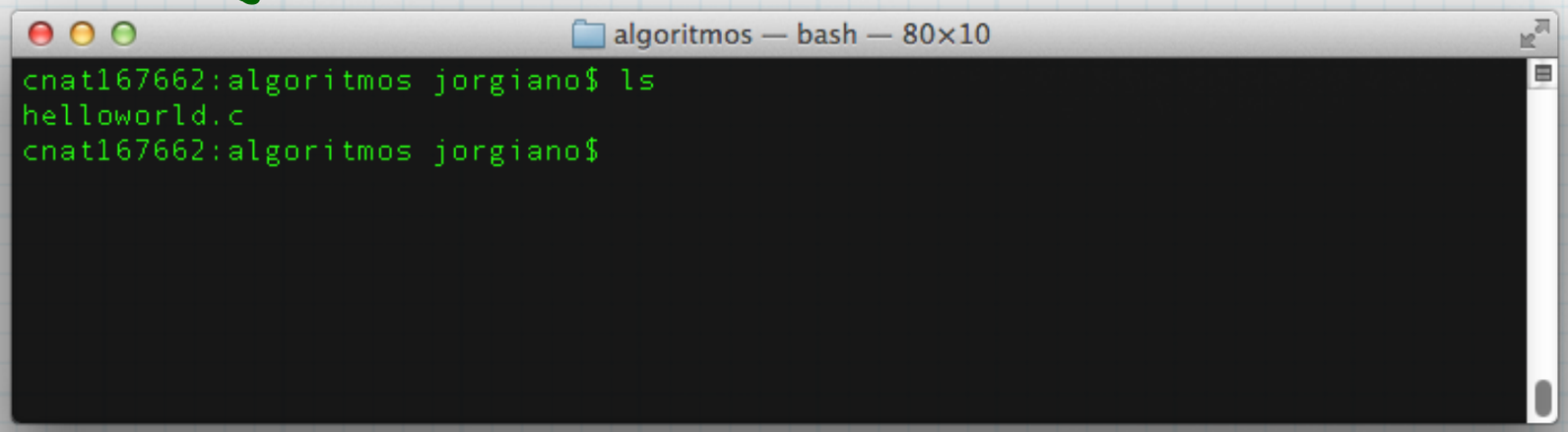


Compilador

* GCC - Gnu Compiler Collection
  * http://gcc.gnu.org
* clang - C Language
  * http://clang.llvm.org
* Existem outros

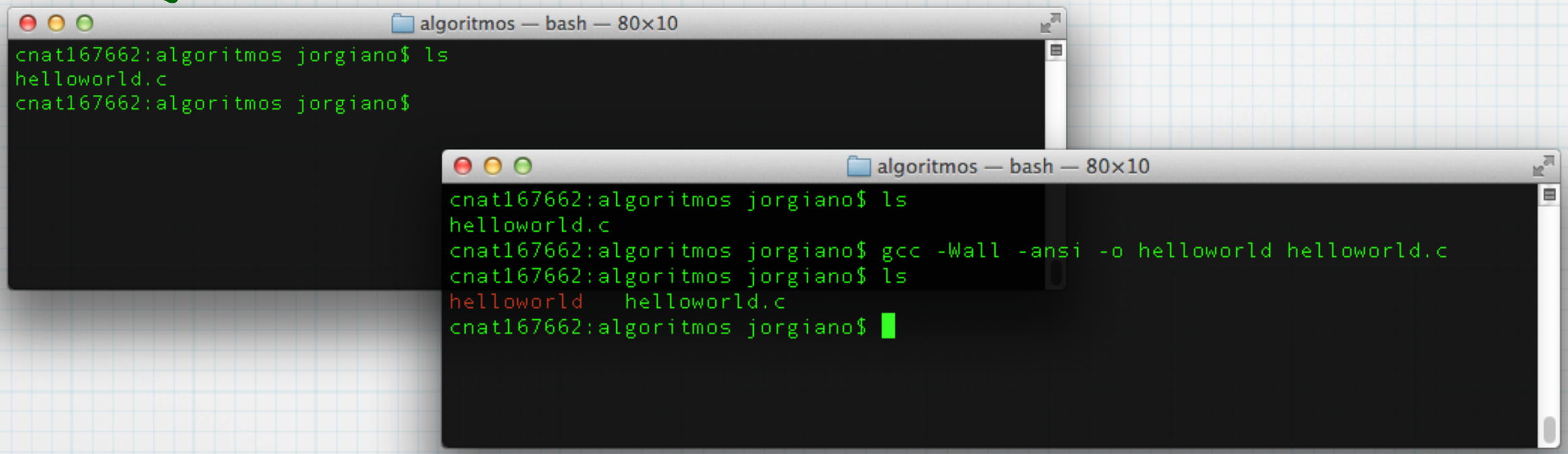
Compilação

* Nome do comando: gcc
* Argumentos
  * `-Wall`: mostra todos os "warnings"
  * `-ansi`: verifica se o código respeita as regras do C ansi
  * `-o EXEC`: gera o executável como nome **EXEC**
* Se não ocorrer erros/warnings o compilador terminará sem mostrar mensagens



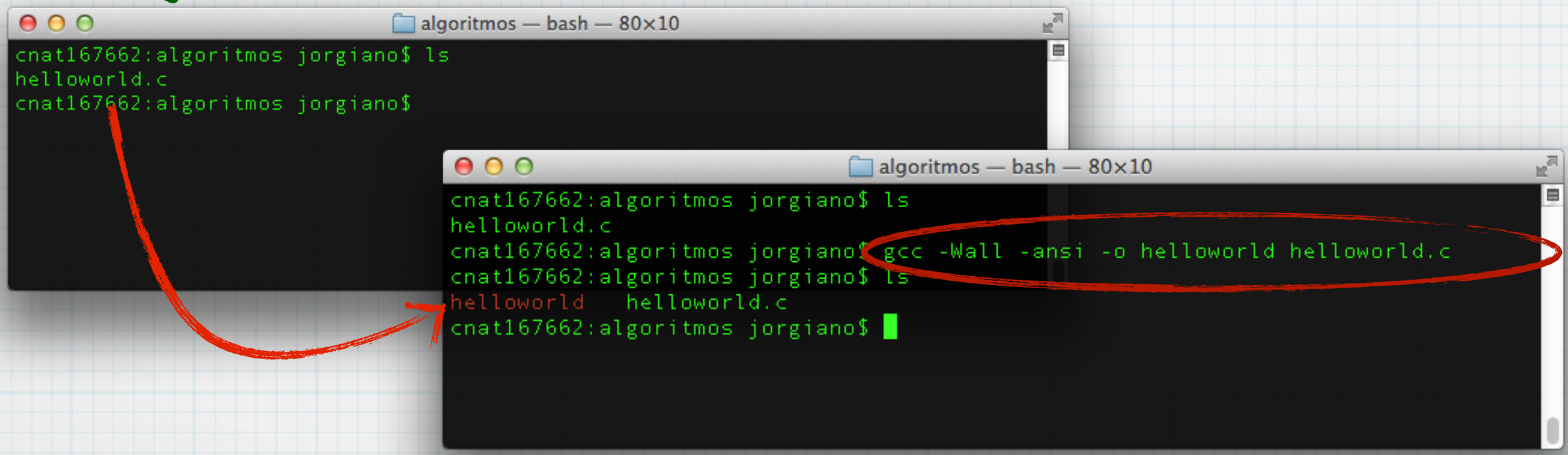
Compilação

* Nome do comando: gcc
* Argumentos
  * `-Wall`: mostra todos os "warnings"
  * `-ansi`: verifica se o código respeita as regras do C ansi
  * `-o EXEC`: gera o executável como nome **EXEC**
* Se não ocorrer erros/warnings o compilador terminará sem mostrar mensagens



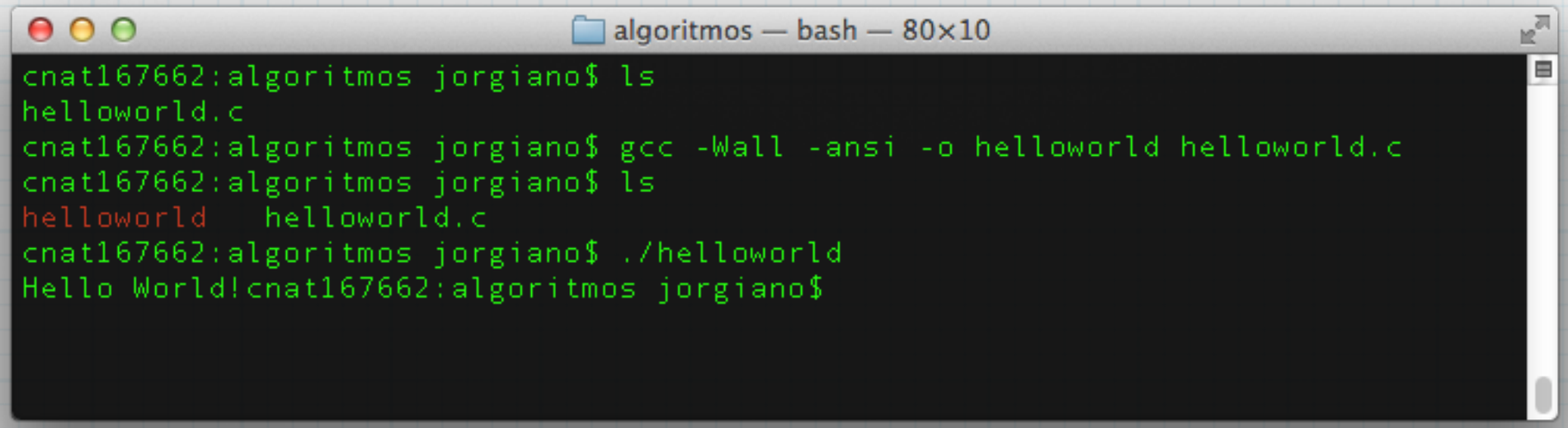
Compilação

* Nome do comando: gcc
* Argumentos
  * `-Wall`: mostra todos os "warnings"
  * `-ansi`: verifica se o código respeita as regras do C ansi
  * `-o EXEC`: gera o executável como nome **EXEC**
* Se não ocorrer erros/warnings o compilador terminará sem mostrar mensagens



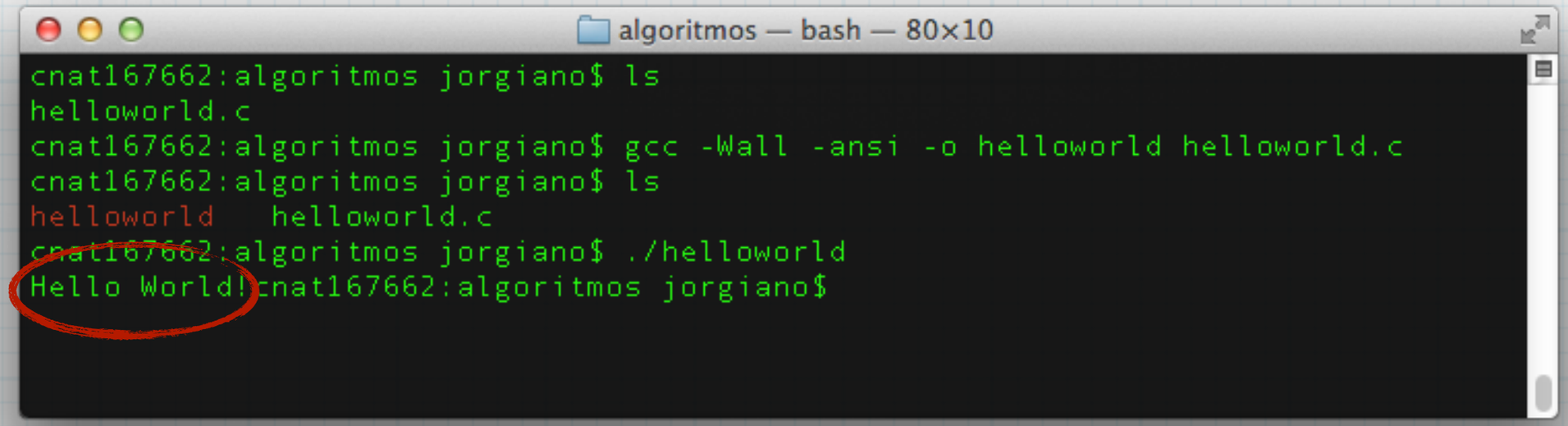
Execução

* Depende do sistema operacional
  * no linux escreve-se o nome do programa na console
* Caminho pode ser absoluto ou relativo

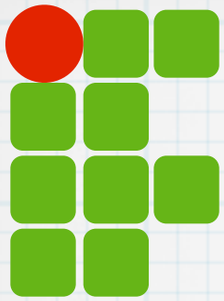


Execução

* Depende do sistema operacional
  * no linux escreve-se o nome do programa na console
* Caminho pode ser absoluto ou relativo

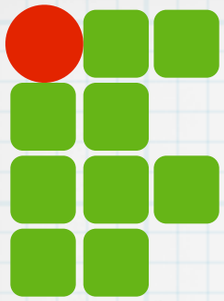


DEMO



Funções em C

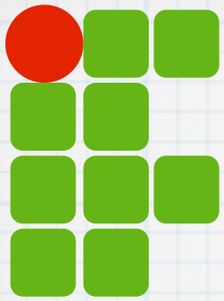
```
int   soma(int a, int b) {
   corpo
}
```



Funções em C

Qualquer tipo

```
int soma(int a, int b) {
   corpo
}
```

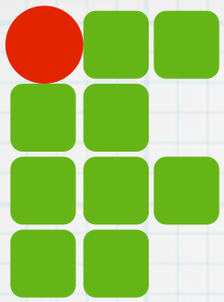


Funções em C

Qualquer tipo

Identificador válido

```
int soma(int a, int b) {
   corpo
}
```



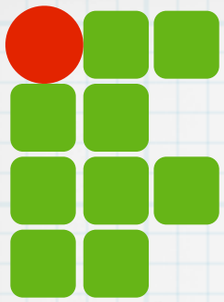
Funções em C

Qualquer tipo

Identificador válido

Lista de identificadores com tipo

```
int soma(int a, int b) {
   corpo
}
```



Funções em C

Qualquer tipo

Identificador válido

Lista de identificadores com tipo

```
int soma(int a, int b) {
  corpo
}
```

Sequencia de declarações e instruções de C

Corpo da função

```c
int soma(int a, int b) {
   int s;
   s = a + b;
   return s;
}
```

Corpo da função

```c
int soma(int a, int b) {
    int s;
    s = a + b;
    return s;
}
```

Sequencia de declarações/ instruções separadas por ponto-e-vírgula

Corpo da função

```c
int soma(int a, int b) {
   int s;
   s = a + b;
   return s;
}
```

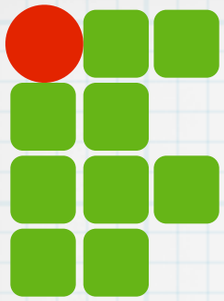
```c
int soma
(int a, int b)
{
   int s;
   s = a + b;
   return s;
}
```

```c
int
soma(int a, int b) {   int s; s = a + b;     return s;}
```

Corpo da função

* Legibilidade
  * Uma instrução por linha
  * Ponto-e-vírgula no final da linha

```c
int soma(int a, int b)
{
   int s;
   s = a + b;
   return s;
}
```



Tipos

* Principais tipos:
  * char
  * short
  * int
  * long
  * float
  * double
* Todos os tipos podem ser:
  * signed
  * unsigned
* Representação binária padrão IEEE
* OBS: String é um array de char

Variáveis

* DEVE ser declarada
  * Declaração define tipo

```c
int soma(int a, int b) {
   int resultado;
   resultado = a + b;
   return resultado;
}
```

Variáveis

* DEVE ser declarada
  * Declaração define tipo

```c
int soma(int a, int b) {
   int resultado;
   resultado = a + b;
   return resultado;
}
```

Variáveis

* DEVE ser declarada
  * Declaração define tipo

Variável resultado é declarada do tipo `int`

```
int soma(int a, int b) {
    int resultado;
    resultado = a + b;
    return resultado;
}
```

Atribuição

* Armazena um valor em uma zona de memória indicada pela variável
  * VAR = EXPR;
* Tipo deve ser compatível
* Variável deve ser declarada
* Exemplo:
  * soma = a+b;

Expressões

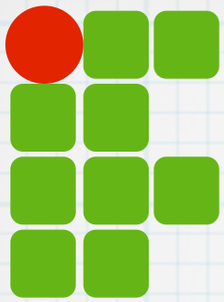
* Lado direito da atribuições
* Operadores dependem do tipo dos operandos
  * + (soma), - (subtração), * (Multiplicação), / (Divisão inteira e real), % (resto da divisão)
* CUIDADO: Divisão inteira diferente da divisão real
  * Observar também tipos das variáveis

10/3 ≠ 10.0/3.0

A função main

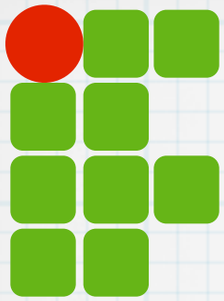
* Todo programa em C começa pela função main
  * retorna um inteiro
  * Possui dois parâmetros
    * inteiro com a quantidade de elementos no array de parâmetros
    * Array de strings

```c
int main(int argc, char ** argv) {
  //Aqui vai o programa
  return 0;
}
```



Entrada e saída

- Ler dados do teclado
  - Função scanf()
    - Ler inteiro: scanf("%d",&a);
    - Ler real: scanf("%f",&x);
    - Ler string: scanf("%s",nome);
- Escrever dados no terminal
  - Função printf
    - printf("Um texto qualquer");
    - printf("a soma de %d e %d é %d",a,b,soma);
    - printf("A média foi de %.2f km/h",media);
  - Importante obeservar tipos das variáveis
- Detalhes do scanf e do printf serão vistos futuramente



Exemplo

```c
#include <stdio.h>

int soma(int a, int b) {
   int resultado;
   resultado = a + b;
   return resultado;
}

int main(int argc, char **argv) {
   int a, b, s;
   scanf("%d", &a);
   scanf("%d", &b);
   s = soma(a, b);
   printf("Soma e %d\n", s);
   return 0;
}
```

Exemplo

```c
#include <stdio.h>

int soma(int a, int b) {
   int resultado;
   resultado = a + b;
   return resultado;
}

int main(int argc, char **argv) {
   int a, b, s;
   scanf("%d", &a);
   scanf("%d", &b);
   s = soma(a, b);
   printf("Soma e %d\n", s);
   return 0;
}
```

Necessário para usar scanf e printf.

Exemplo

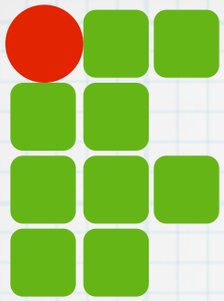
```c
#include <stdio.h>

int soma(int a, int b) {
   int resultado;
   resultado = a + b;
   return resultado;
}

int main(int argc, char **argv) {
   int a, b, s;
   scanf("%d", &a);
   scanf("%d", &b);
   s = soma(a, b);
   printf("Soma e %d\n", s);
   return 0;
}
```

Necessário para usar scanf e printf.

Indica uma quebra de linha



Declaração e uso de função

```c
int soma(int a, int b);

int main(int argc, char **argv) {
   int a, b, s;
   scanf("%d",&a);
   scanf("%d",&b);

   s = soma(a, b);
   printf("Soma e %d\n", s);
   return 0;
}

int soma(int a, int b) {
   int resultado;
   resultado = a + b;
   return resultado;
}
```

Declaração e uso de função

Assinatura pode ser feita separada do corpo da função

```c
int soma(int a, int b);

int main(int argc, char **argv) {
   int a, b, s;
   scanf("%d",&a);
   scanf("%d",&b);

   s = soma(a, b);
   printf("Soma e %d\n", s);
   return 0;
}

int soma(int a, int b) {
   int resultado;
   resultado = a + b;
   return resultado;
}
```

Declaração e uso de função

Assinatura pode ser feita separada do corpo da função

Para que a função possa ser usada, sua assinatura (retorno, param, etc) deve ser conhecida

```c
int soma(int a, int b);

int main(int argc, char **argv) {
   int a, b, s;
   scanf("%d",&a);
   scanf("%d",&b);

   s = soma(a, b);
   printf("Soma e %d\n", s);
   return 0;
}

int soma(int a, int b) {
   int resultado;
   resultado = a + b;
   return resultado;
}
```

Declaração e uso de função

Assinatura pode ser feita separada do corpo da função

Para que a função possa ser usada, sua assinatura (retorno, param, etc) deve ser conhecida

Corpo da função

```c
int soma(int a, int b);

int main(int argc, char **argv) {
   int a, b, s;
   scanf("%d",&a);
   scanf("%d",&b);

   s = soma(a, b);
   printf("Soma e %d\n", s);
   return 0;
}

int soma(int a, int b) {
   int resultado;
   resultado = a + b;
   return resultado;
}
```

DEMO

Dúvidas?