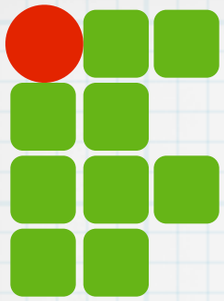
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

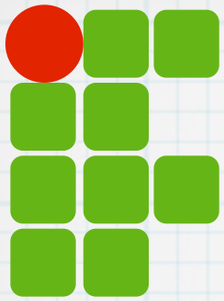
Algoritmos

ANSI C - Strings



Agenda

* O tipo char
* Tabela ASCII
* Strings
* Leitura
* Biblioteca string.h
* A função main em detalhes
* Exercícios



O tipo char

* Representa um caractere
  * 8 bits
* Declaração

```
char letra;
```

* Caractere deve estar entre aspas simples

```
letra = 'A';
```

* %c usado para scanf e printf

```
scanf("%c",&letra);
printf("Voce digitou a letra %c\n", letra);
```

Tabela ASCII

* Cada caractere tem um código
  * Inteiro de 8 bits

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Letra A tem o código 65

Caractere '0' NÃO possui código 0

Tabela ASCII

```c
char letra;
letra = 'A';
printf("O caractere %c tem o código ASCII %d\n", letra, letra);
```

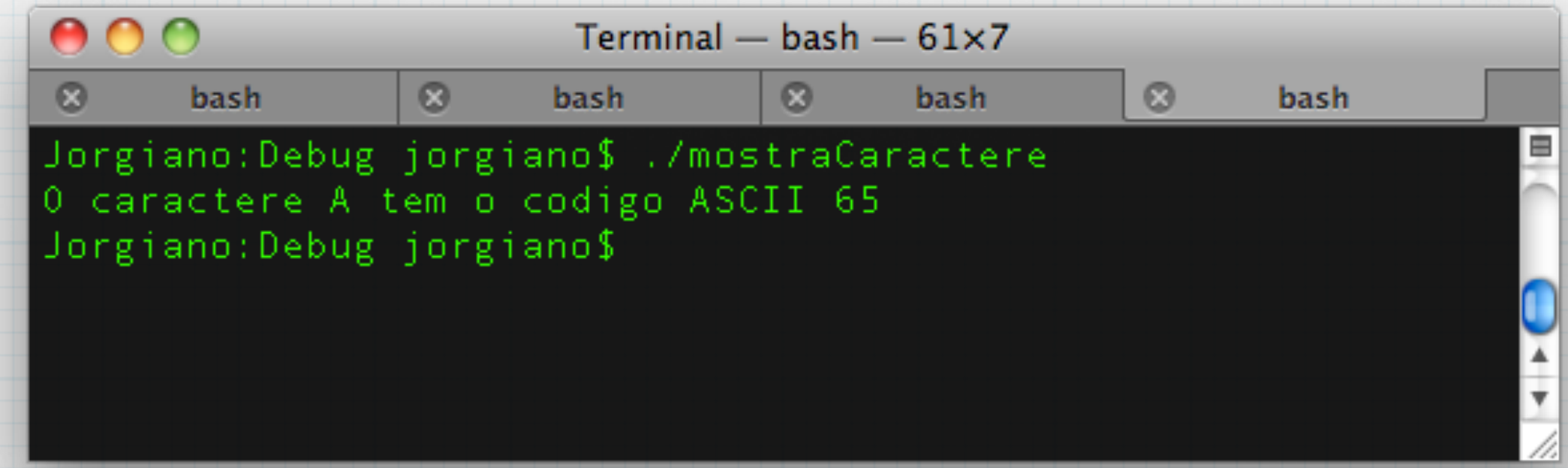


Tabela ASCII

```
char letra;
letra = 'A';
printf("O caractere %c tem o código ASCII %d\n", letra, letra);
```

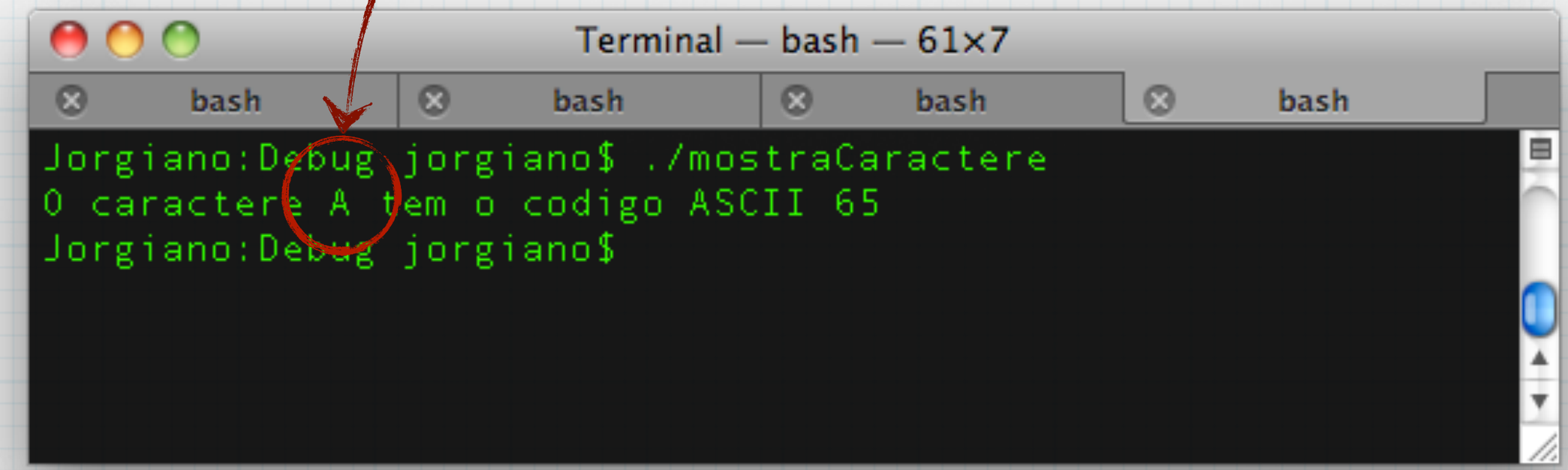


Tabela ASCII

```
char letra;
letra = 'A';
printf("O caractere %c tem o código ASCII %d\n", letra, letra);
```

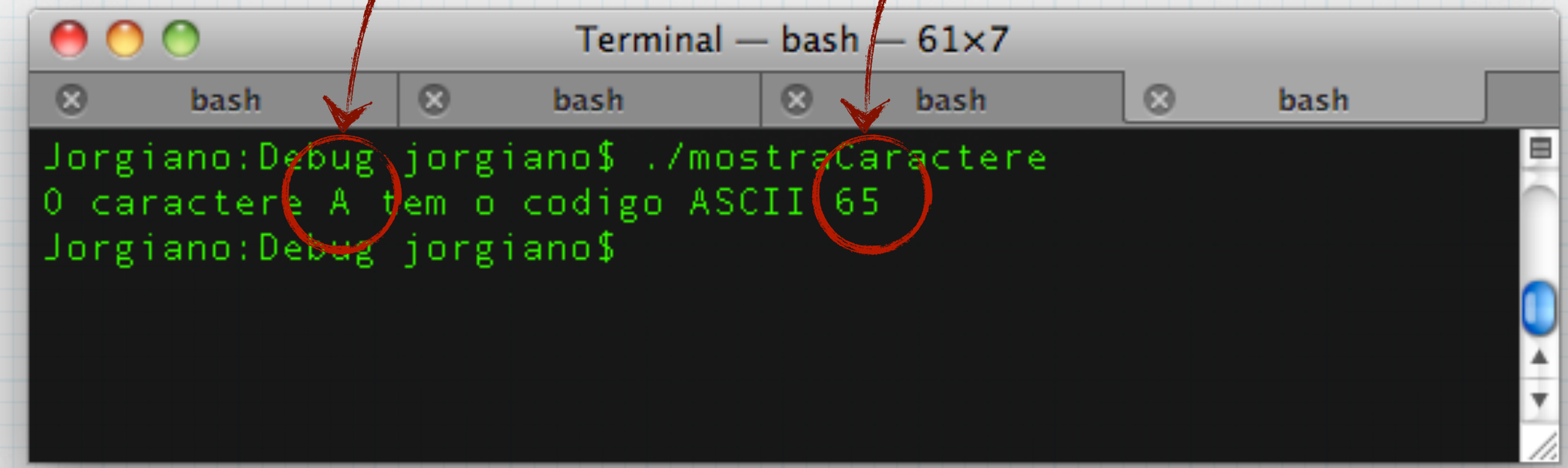
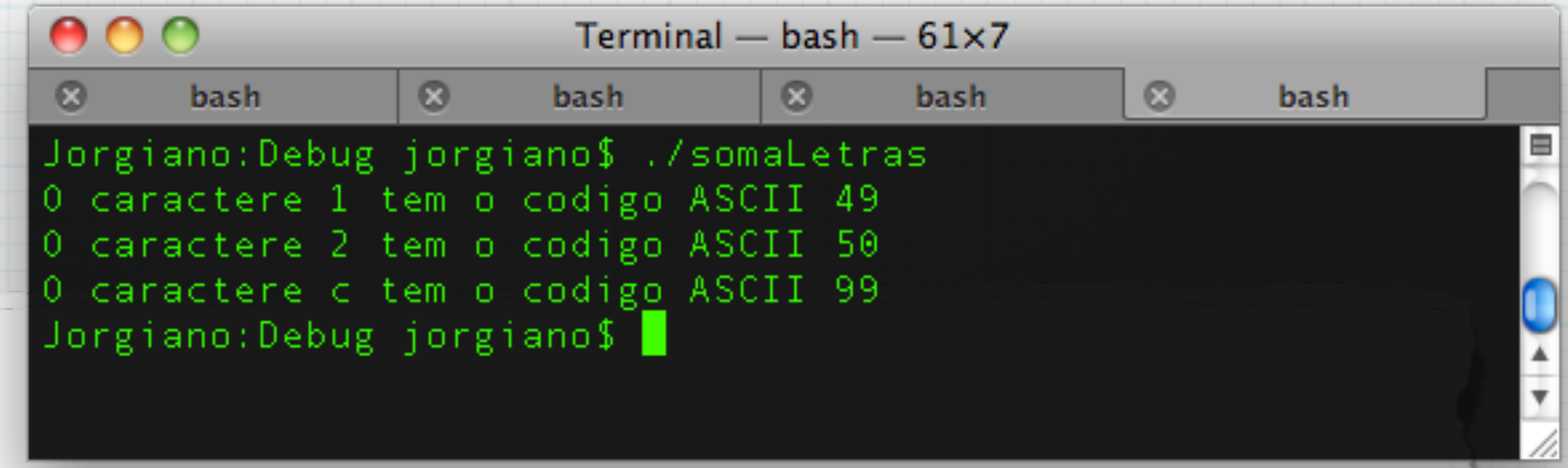


Tabela ASCII

* Operações aritméticas sobre inteiro são válidas
  * 8 BITS




```c
char c1,c2,c3;
c1 = '1';
c2 = '2';
c3 = c1+c2;
printf("O caractere %c tem o codigo ASCII %d\n", c1, c1);
printf("O caractere %c tem o codigo ASCII %d\n", c2, c2);
printf("O caractere %c tem o codigo ASCII %d\n", c3, c3);
```

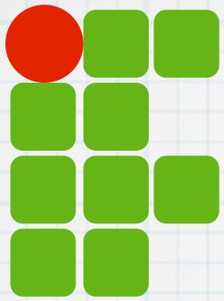


Tabela ASCII

* Converter string para minúscula
  * Verificar cada caractere

```
int main(int argc, char **argv) {
    char c, dif = 'a' - 'A';
    scanf("%c", &c);
    if (c >= 'a' && c <= 'z') {
        c = c - dif;
        printf("Resultado da conversão: %c \n", c);
    } else
        printf("Caractere digitado nao e letra minúscula");
    return 0;
}
```

A	65	a	97
B	66	b	98
C	67	c	99
D	68	d	100
E	69	e	101
F	70	f	102
G	71	g	103
H	72	h	104
I	73	i	105
J	74	j	106
K	75	k	107
L	76	l	108
M	77	m	109
N	78	n	110
O	79	o	111
P	80	p	112
Q	81	q	113
R	82	r	114
S	83	s	115
T	84	t	116
U	85	u	117
V	86	v	118
W	87	w	119
X	88	x	120
Y	89	y	121
Z	90	z	122

Tabela ASCII

* Converter string para minúscula
  * Verificar cada caractere

```c
int main(int argc, char **argv) {
    char c, dif = 'a' - 'A';
    scanf("%c", &c);
    if (c >= 'a' && c <= 'z') {
        c = c - dif;
        printf("Resultado da conversão: %c \n", c);
    } else
        printf("Caractere digitado nao e letra minúscula");
    return 0;
}
```

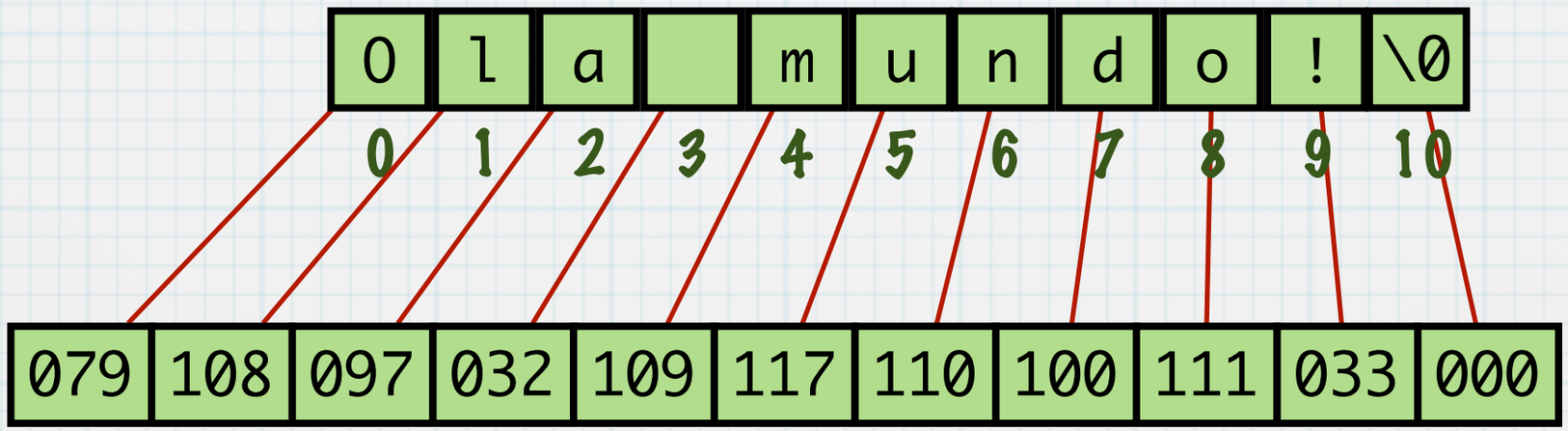
A variável dif armazena a diferença do código de a e A, que é a mesma entre qualquer par maiúscula/minúscula na tabela ASCII. Letras maiúsculas aparecem ANTES das minúsculas

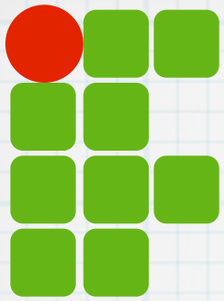
A	65	a	97
B	66	b	98
C	67	c	99
D	68	d	100
E	69	e	101
F	70	f	102
G	71	g	103
H	72	h	104
I	73	i	105
J	74	j	106
K	75	k	107
L	76	l	108
M	77	m	109
N	78	n	110
O	79	o	111
P	80	p	112
Q	81	q	113
R	82	r	114
S	83	s	115
T	84	t	116
U	85	u	117
V	86	v	118
W	87	w	119
X	88	x	120
Y	89	y	121
Z	90	z	122

String

* Uma string (cadeia de caracteres) é um sequência (array) de caracteres que formam um texto
* Terminam com o caractere especial '\0'
  * Ocupa espaço no array
  * O '\0' é o caractere com código ASCII 0 (zero).

```
char mensagem[11]
```





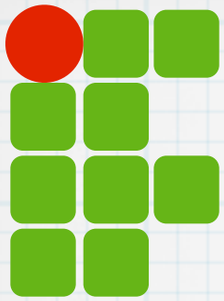
String

```
char mensagem[11]
```

0	l	a		m	u	n	d	o	!	\0
0	1	2	3	4	5	6	7	8	9	10

```
char mensagem[15]
```

0	l	a		m	u	n	d	o	!	\0	?	?	?	?
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14



String

* Declaração e inicialização

```
char mensagem[11] = "Olá mundo!";
```

String

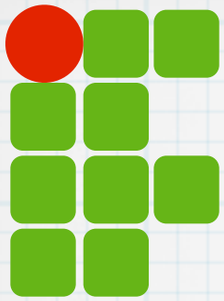
* Declaração e inicialização

```
char mensagem[11] = "Olá mundo!";
```

NÃO funciona para atribuição normal

```
char mensagem[11];
mensagem = "Olá mundo!";
```

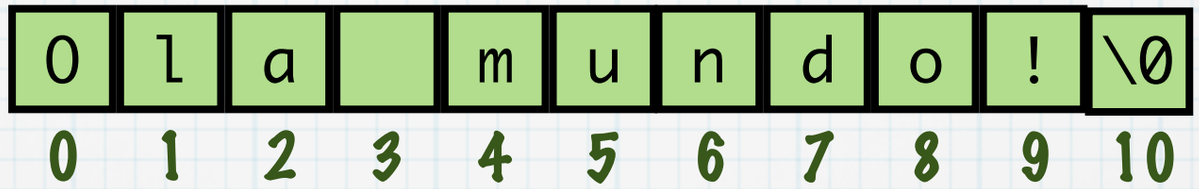
Veremos mais a frente como fazer isso



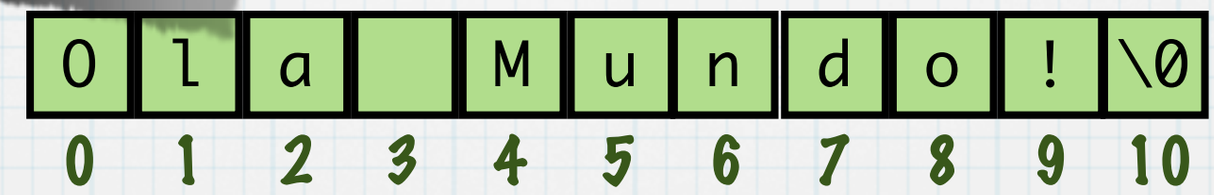
String

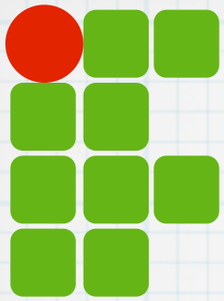
* String == Array

```
char mensagem[11] = "Olá mundo!";
```




```
mensagem[4] = 'M';
```



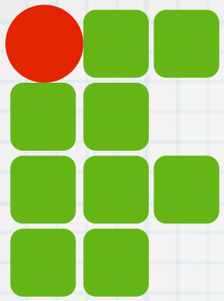


String

* printf
  * Com máscara '%s'

```c
#include <stdio.h>

int main(int argc, char **argv) {
    char mensagem[11] =  "Olá mundo!";
    mensagem[4] = 'M';
    printf ("A mensagem é %s \n",mensagem);
   return 0;
}
```

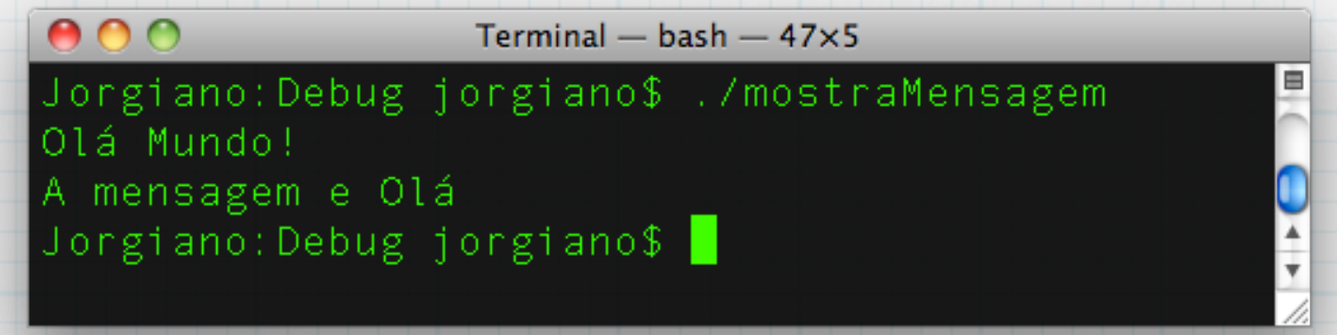


String

* scanf

* Com máscara '%s'

```
int main(int argc, char **argv) {
    char mensagem[80];
    scanf("%s",mensagem);
    printf ("A mensagem é %s \n",mensagem);
   return 0;
}
```



String

* scanf

* Com máscara '%s'

```
int main(int argc, char **argv) {
    char mensagem[80];
    scanf("%s",mensagem);
    printf ("A mensagem é %s \n",mensagem);
  return 0;
}
```

NÃO usamos o &

```
Terminal — bash — 47×5
Jorgiano:Debug jorgiano$ ./mostraMensagem
Olá Mundo!
A mensagem e Olá
Jorgiano:Debug jorgiano$
```

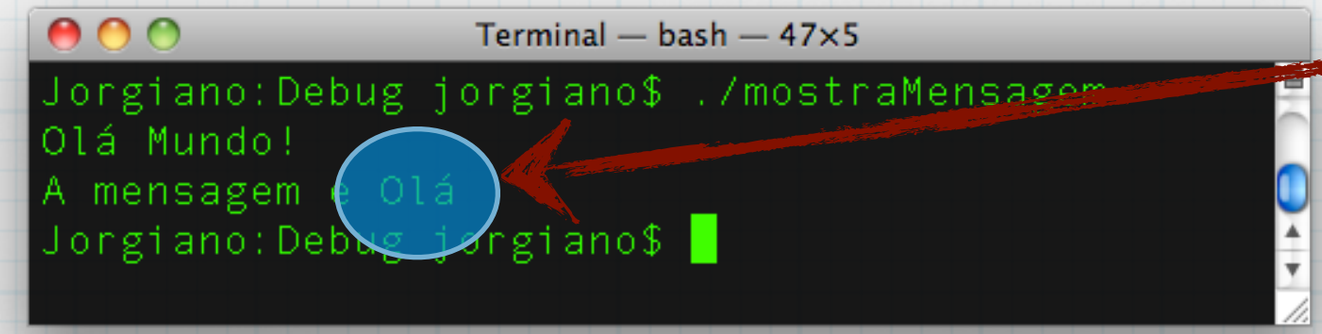
String

* scanf

* Com máscara '%s'

```
int main(int argc, char **argv) {
    char mensagem[80];
    scanf("%s",mensagem);
    printf ("A mensagem é %s \n",mensagem);
  return 0;
}
```

NÃO usamos o &



IMPORTANTE
A leitura da string termina quando encontra um espaço em branco

String

* Comando `gets`
* **CUIDADO**: O tamanho da string deve "caber" no array de carateres
* O gets **NÃO** verifica

```c
#include <stdio.h>
int main(int argc, char **argv) {
     char mensagem[80];
     gets(mensagem);
     printf ("A mensagem e %s \n",mensagem);
    return 0;
}
```

```
Terminal — bash — 52×5
Jorgiano:Debug jorgiano$ ./mostraMensagem
warning: this program uses gets(), which is unsafe.
Olá mundo!
A mensagem e Olá mundo!
Jorgiano:Debug jorgiano$
```

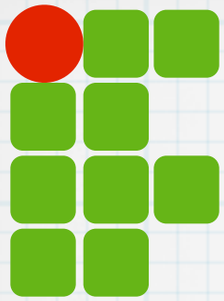
✓

String

* Comando `gets`
* **CUIDADO**: O tamanho da string deve "caber" no array de carateres
* O gets **NÃO** verifica

```c
#include <stdio.h>
int main(int argc, char **argv) {
     char mensagem[80];
     gets(mensagem);
     printf ("A mensagem e %s \n",mensagem);
    return 0;
}
```

```
Terminal — bash — 52×5
Jorgiano:Debug jorgiano$ ./mostraMensagem
warning: this program uses gets(), which is unsafe.
Olá mundo!
A mensagem e Olá mundo!
Jorgiano:Debug jorgiano$
```



Manipulação de strings

* O compilador C vem com uma biblioteca de funções para trabalharmos com string
  * Precisamos incluir o `string.h`
  * `#include "string.h"`
* Algumas funções:
  * `strlen`: tamanho da string
  * `strcpy`: copia uma string
  * `strcat`: concatena strings (junta)
  * `strcmp`: compara duas strings

Manipulação de strings

* `strlen(string)`
  * Retorna o número de caracteres válidos
  * NÃO inclui o '\0' que indica o fim da String
  * para a string "Olá mundo!", o comando `strlen` retorna 10

```
Terminal — bash — 52×5
Jorgiano:Debug jorgiano$ ./olaMundo
A string "Ola mundo!" contem 10 caracteres
Jorgiano:Debug jorgiano$
```

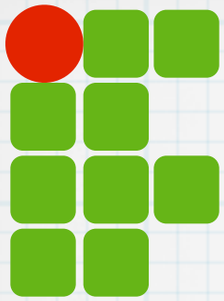
```c
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv) {
    char mensagem[80] = "Ola mundo!";
    int numeroDeCaracteres;
    numeroDeCaracteres = strlen(mensagem);
    printf ("A string \"%s\" contem %d caracteres\n",mensagem,numeroDeCaracteres);
   return 0;
}
```

Manipulação de strings

- `strlen(string)`
  - Retorna o número de caracteres válidos
  - NÃO inclui o '\0' que indica o fim da String
  - para a string "Olá mundo!", o comando `strlen` retorna 10

```
Terminal — bash — 52×5
Jorgiano:Debug jorgiano$ ./olaMundo
A string "Ola mundo!" contem 10 caracteres
Jorgiano:Debug jorgiano$
```

```c
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv) {
    char mensagem[80] = "Ola mundo!";
    int numeroDeCaracteres;
    numeroDeCaracteres = strlen(mensagem);
    printf ("A string \"%s\" contem %d caracteres\n",mensagem,numeroDeCaracteres);
   return 0;
}
```



Manipulação de String

```c
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv) {
    char mensagem[80];
    char letra;
    int i,quantidade;
    printf("Digite a mensagem: ");
    gets(mensagem);
    printf("Digite a letra a ser contada: ");
    scanf("%c",&letra);
    quantidade = 0;
    for (i = 0 ; i < strlen(mensagem) ; i++){
       if (mensagem[i] == letra){
           quantidade++;
       }
    }
    printf ("Quantidade da letra %c na mensagem e %d\n",letra,quantidade);
   return 0;
}
```

Manipulação de String

```c
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv) {
    char mensagem[80];
    char letra;
    int i,quantidade;
    printf("Digite a mensagem: ");
    gets(mensagem);
    printf("Digite a letra a ser contada: ");
    scanf("%c",&letra);
    quantidade = 0;
    for (i = 0 ; i < strlen(mensagem) ; i++){
       if (mensagem[i] == letra){
           quantidade++;
       }
    }
    printf ("Quantidade da letra %c na mensagem e %d\n",letra,quantidade);
   return 0;
}
```

Leitura de UM caractere

Manipulação de String

* strcpy(destino,origem)
  * Copia conteúdo da string origem para a string destino
  * A capacidade da string destino deve ser igual ou maior que a string origem

```c
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv) {
    char mensagem1[80] = "Ola mundo!";
    char mensagem2[20];
    strcpy(mensagem2,mensagem1);
    printf ("mensagem1 = %s\n",mensagem1);
    printf ("mensagem2 = %s\n",mensagem2);
   return 0;
}
```

Manipulação de String

* strcat(destino,origem)
  * strcat ACRESCENTA a string origem no FINAL da string destino
  * A capacidade da string destino deve suportar o novo tamanho (string1+string2)

```
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv) {
    char mensagem1[80] = "Ola ";
    char mensagem2[80] = "mundo!";
    printf ("Mensagem1 = %s\n",mensagem1);
    strcat(mensagem1,mensagem2);
    printf ("Mensagem1 = %s\n",mensagem1);
    printf ("Mensagem2 = %s\n",mensagem2);
   return 0;
}
```

```
Terminal — bash — 47×7
Jorgiano:tmp jorgiano$ ./strcat
Mensagem1 = Ola
Mensagem1 = Ola mundo!
Mensagem2 = mundo!
Jorgiano:tmp jorgiano$
```

Manipulação de String

- strcmp(string1,string2)
  - Este comando compara o conteúdo das duas strings e retorna:
  - 0 (zero): as duas strings são iguais
  - > 0: Strings diferentes, retorna a "distância" entre o primeiro caracter diferente (ver tabela ASCII)
  - < 0: Strings diferentes, retorna a "distância" entre o primeiro caracter diferente (ver tabela ASCII)

string1	string2	Resultado
Ola mundo!	Ola mundo!	0
a	b	-1
b	a	1
a	c	-2
c	a	2
IFRN	ifrn	-32
ifrn	IFRN	32
abc	america	-11
alecrim	abc	10
Joao	Jose	-18
Joao	João	-42

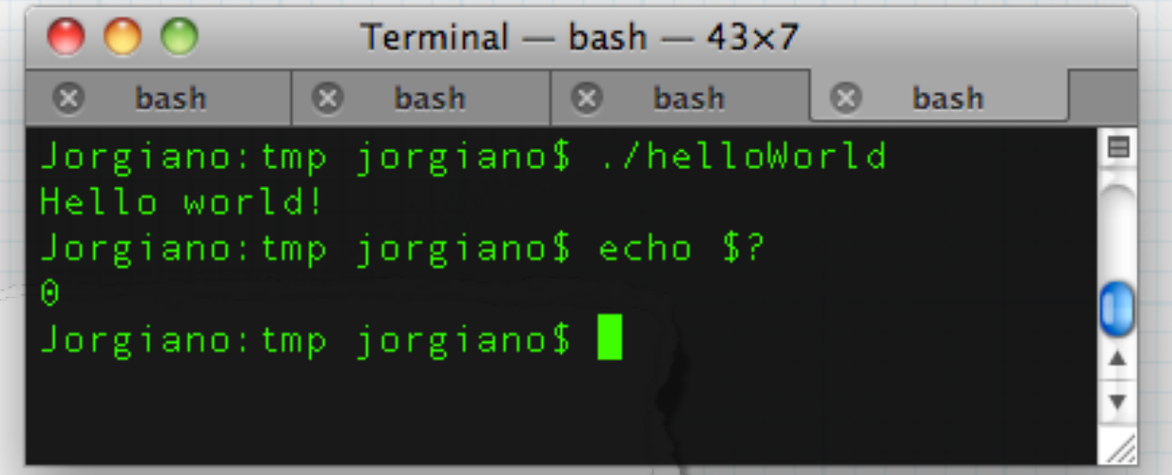
A função main

* Ponto inicial de um programa em C
* Parâmetros podem ser passados
  * lista de strings
* Retorna um valor inteiro ao S.O.

```c
int main(int argc, char **argv) {
    // Programa
  return 0;
}
```

Retorno

* Uma execução de um programa deve retornar um valor inteiro ao S.O.
* O valor 0 (zero) indica que o programa terminou normalmente
* No linux podemos ver o retorno de um programa com a expressão $?



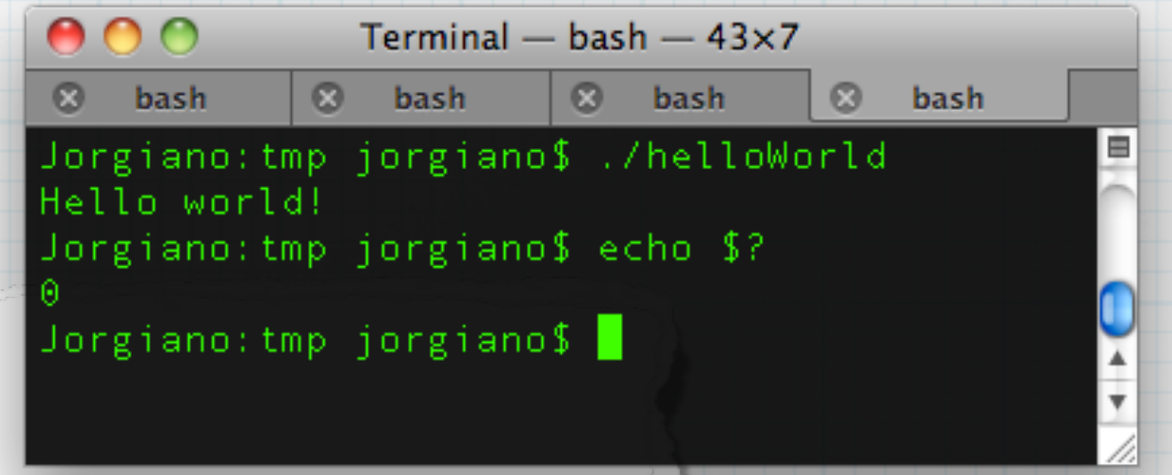

```c
#include <stdio.h>

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

Tipo deve ser `int`

Retorno

* Uma execução de um programa deve retornar um valor inteiro ao S.O.
* O valor 0 (zero) indica que o programa terminou normalmente
* No linux podemos ver o retorno de um programa com a expressão $?



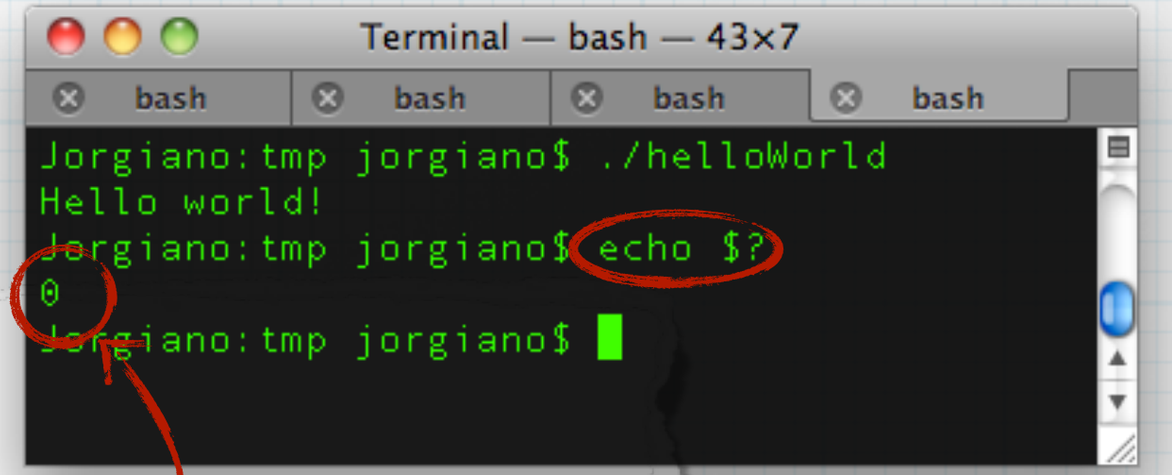

```c
#include <stdio.h>

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

Tipo deve ser `int`

Retorno

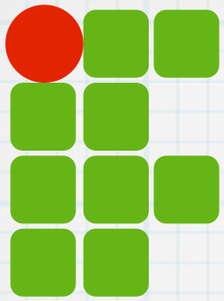
* Uma execução de um programa deve retornar um valor inteiro ao S.O.
* O valor 0 (zero) indica que o programa terminou normalmente
* No linux podemos ver o retorno de um programa com a expressão $?




```c
#include <stdio.h>

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

Tipo deve ser `int`



Parâmetros

* Informações são passadas como strings
  * argc: Parâmetro que indica quantidade de strings passadas
  * argv: array de strings

```
int main(int argc, char **argv)
```

Parâmetros

* Informações são passadas como strings
  * argc: Parâmetro que indica quantidade de strings passadas
  * argv: array de strings

```
int main(int argc, char **argv)
```

A notação ** será explicada mais a frente no curso

Parâmetros

* Listar parâmetros do programa

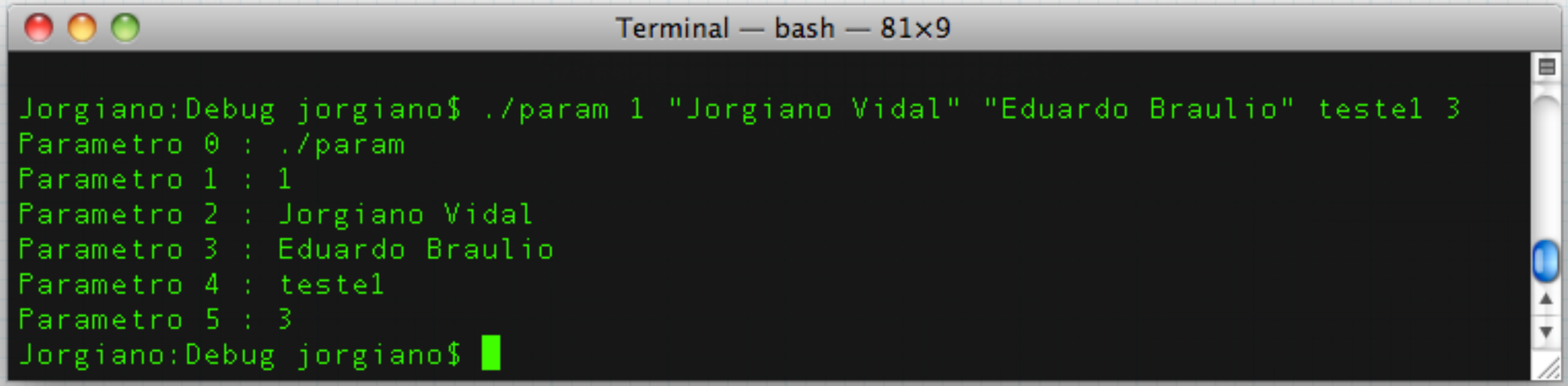
```c
int main(int argc, char **argv) {
    int i;
    for (i = 0 ; i < argc ; i++){
        printf("Parametro %d : %s\n",i,argv[i]);
    }
    return 0;
}
```

```
Terminal — bash — 46×7
Jorgiano:Debug jorgiano$ ./param teste
Parametro 0 : ./param
Parametro 1 : teste
Jorgiano:Debug jorgiano$
```

O primeiro parâmetro é o próprio programa

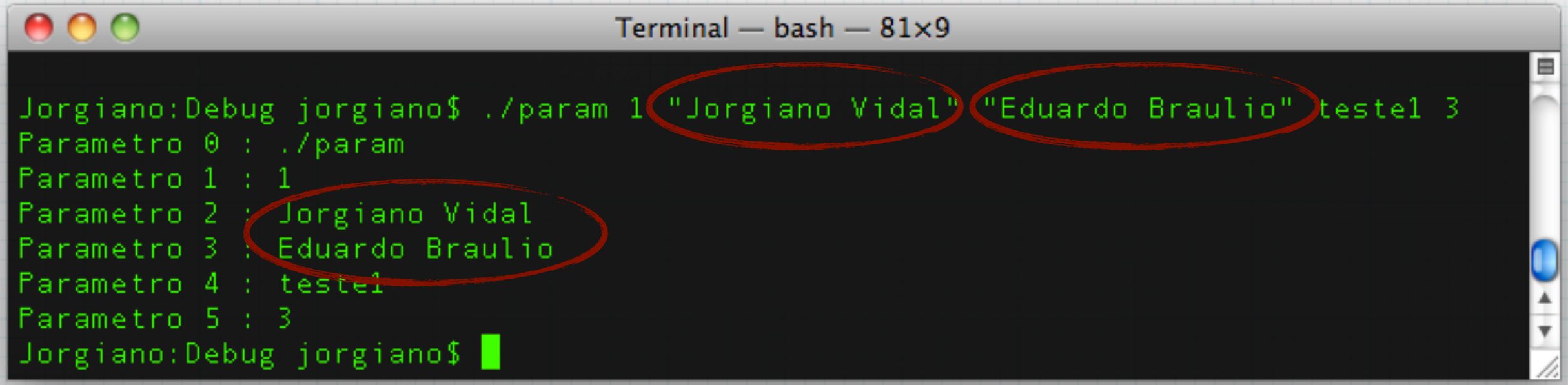
Parâmetros

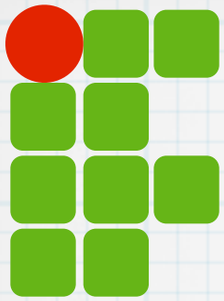
* Espaços separam as strings
  * Como fazer se o espaço fizer parte da string?
  * Coloca entre aspas



Parâmetros

* Espaços separam as strings
  * Como fazer se o espaço fizer parte da string?
  * Coloca entre aspas





Parâmetros

* Dizer se uma string passada é um número

```
int main(int argc, char **argv) {
    int i, eNumero = 0;
    if (argc != 2) {
        printf("Precisa informa um parametro para o programa!\n");
    } else {
        for (i = 0; i < strlen(argv[1]); i++) {
            if (argv[1][i] < '0' || argv[1][i] > '9') {
                eNumero = 1;
            }
        }
        if (eNumero == 0) {
            printf("O parametro e um numero!\n");
        } else {
            printf("O parametro NAO e um numero!\n");
        }
    }
    return 0;
}
```

Dúvidas?

Exercício

* Escrever um programa que recebe duas strings, verifique se ambas são números e mostre a soma dos mesmos
  * DICA
    * Crie funções para:
      * Retornar se uma Strings é um número inteiro
      * Converter uma String em um número