# 3. Primeiros programas

DIM0321

2015.1

# Outline

# Ordem de grandeza e computadores

| Elemento | Latência | Comparação |
|---|---|---|
| Cache L1 | 05 ns | batida coração |
| Predição de ramo errada | 5 ns | bocejo |
| Cache L2 | 7 ns | bocejão |
| Sincronização dado | 25 ns | fazer cafe |
| RAM | 100 ns | lavar dentes |
| Comprimir 1K bytes | 3 µs | novela |
| Enviar 2KB (rede 1GBps) | 20 µs | almoço → fim do dia |
| SSD | 150 µs | fim de semana |
| Ler 1 MB da RAM | 250 µs | fim de semana + feriado |
| Ida/volta num datacenter | 0.5 ms | semana de ferias |
| Ler 1 MB de SSD | 1 ms | natal-reis magos |
| Busca num HD | 10 ms | semestre letivo |
| Ler 1 MB do HD | 20 ms | gravidez |
| Enviar pacote USA→ Europa→ USA | 150 ms | bacharelado |

- Leituras sequênciais

# Bell Labs

- Laboratórios de pesquisa de AT & T (hoje Alcatel-Lucent)
- 8 prêmios Nobel, 2 prêmios Turing (K. Thompson & R. Hamming)

# Unix (1969)

## A filiação

1. CTSS (*Compatible Time-Sharing System*)
2. Multics
3. Unix (ou UNICS) começa em 1969 num PDP-7, a partir dum videogame Space Travel

## Datas

1969 Desenvolvimento

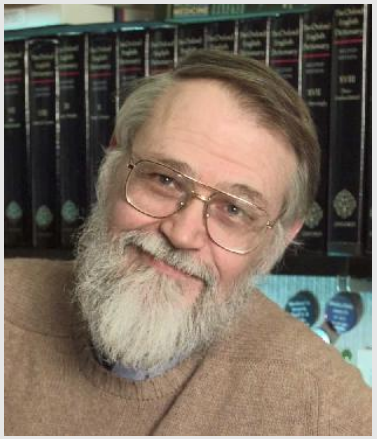
1971 Publicação interna

1973 Publicação externa (Unix v1)

# Pessoas

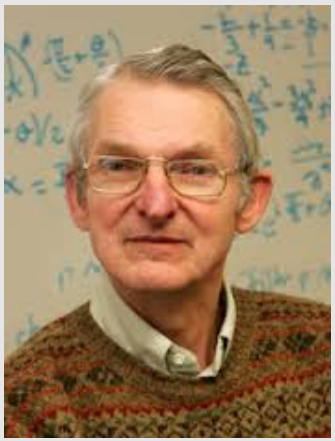
## Dennis Ritchie


## Brian Kernighan



## Ken Thompson

## Doug McIlroy



# Unix moderno (1979)

## Inicialmente

- Primeiro protótipo em 2 dias num PDP-7 (seu celular tem muito mais RAM que ele teve RAM + HD)
- Uso da linguagem B + assembly
- Primeira aplicação: o ancestre de nroff

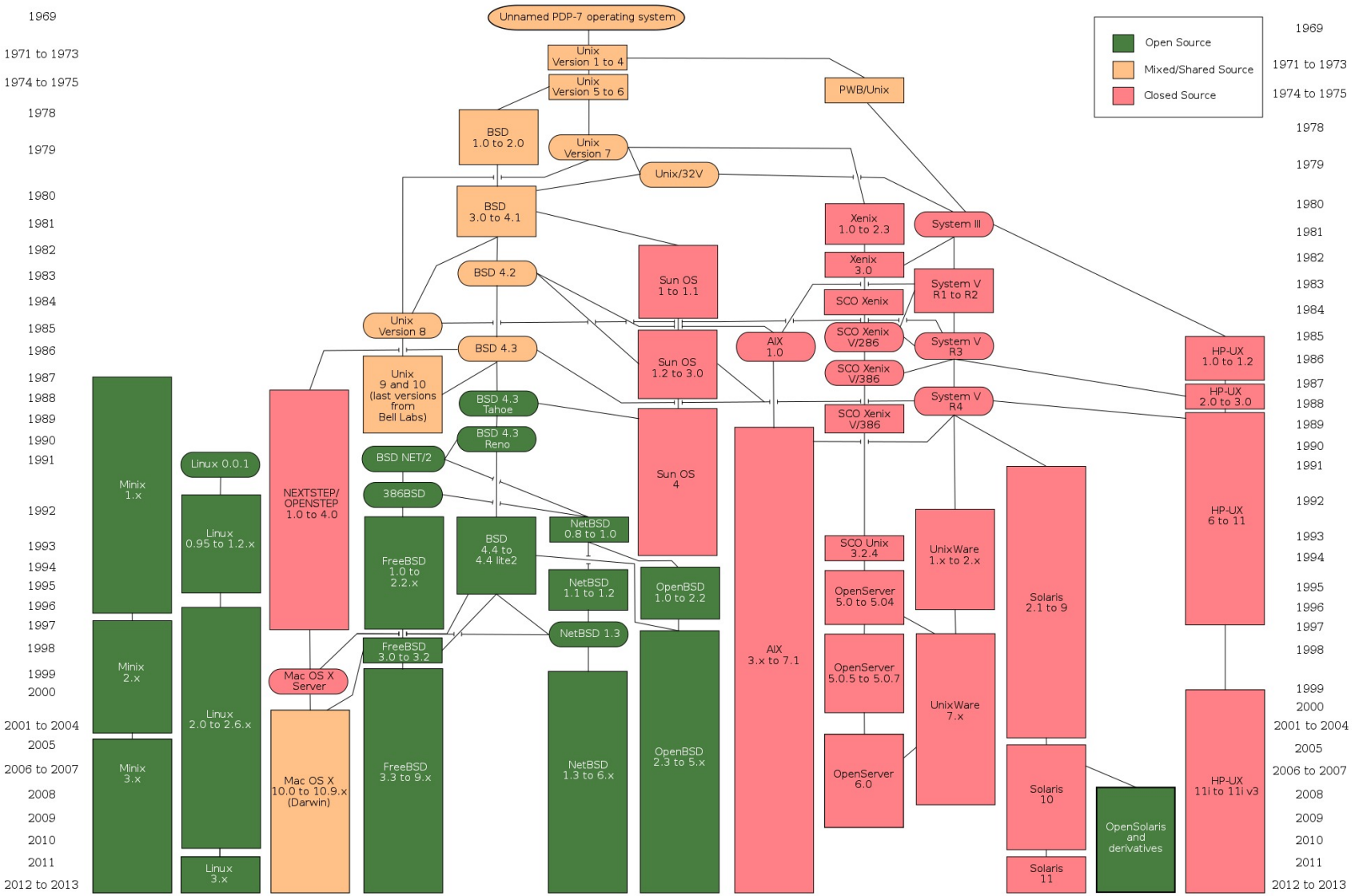
## Depois

- Reescrita em C em 1973
- $>$ 600 instalações em 1974
- Precisava de máquina sem muito poderj
- AT & T não podia torná-lo num produto: cópias eram mandadas a quem queria

1979 Unix v7

1980 DARPA escolha BSD para TCP/IP

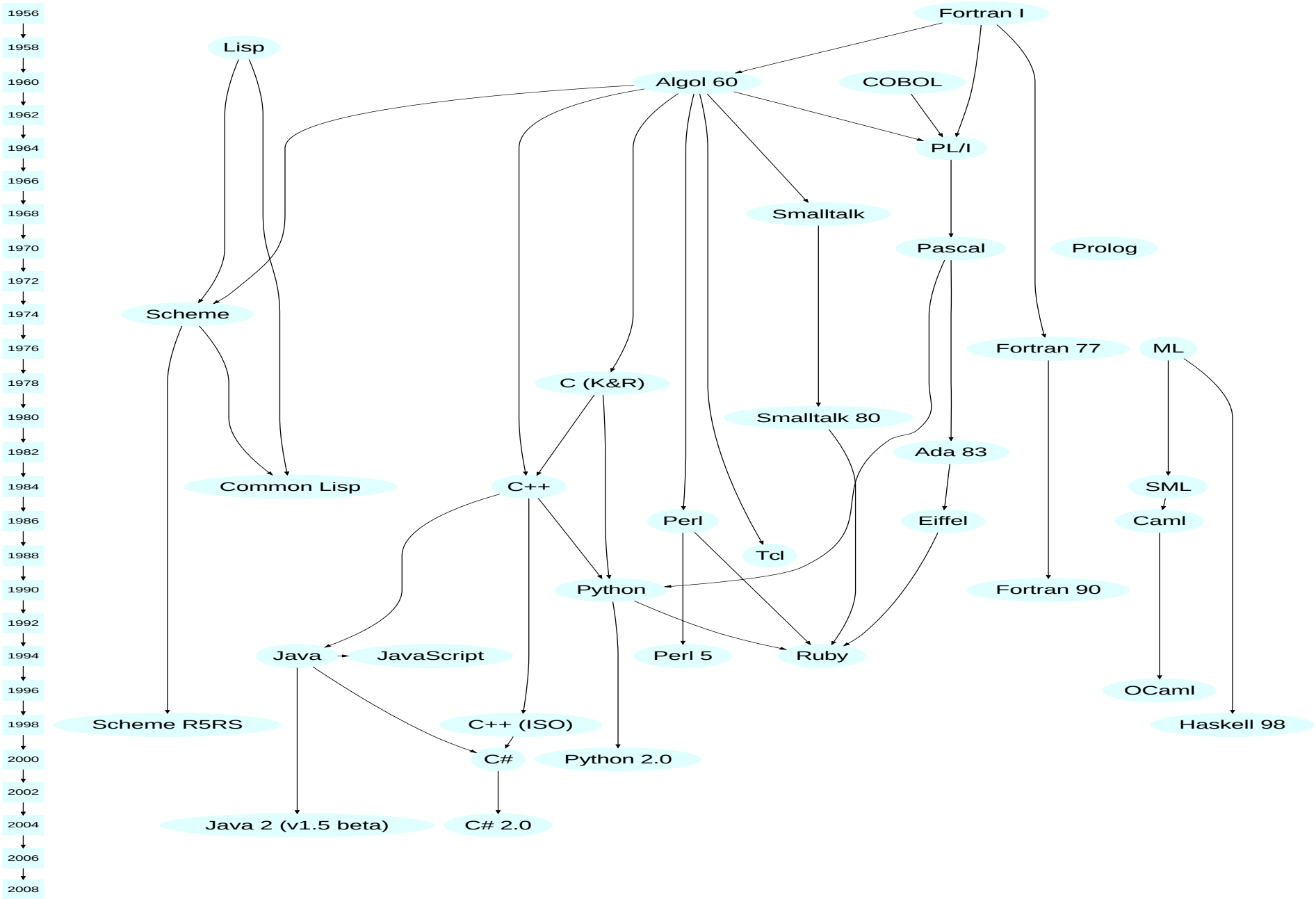
# Descendência de Unix



# C (1972)


- Último padronização C11 (usaremos C99)
- Uso:
  - Sistemas operacionais (Linux, *BSD, Mac OS X, Windows)
  - Software embutido (controles de avião, carros)
  - Primeira/segunda linguagem mais usada em 2014

# Filiação


1956
1958
1960
1962
1964
1966
1968
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
Fortran I
Lisp
Algol 60
COBOL
PL/I
Smalltalk
Pascal
Prolog
Scheme
Fortran 77
ML
C (K&R)
Smalltalk 80
Ada 83
Common Lisp
C++
SML
Perl
Eiffel
Caml
Tcl
Python
Fortran 90
Java
JavaScript
Perl 5
Ruby
OCaml
Scheme R5RS
C++ (ISO)
Haskell 98
C#
Python 2.0
Java 2 (v1.5 beta)
C# 2.0

# GCC (1987)

- *GNU Compiler Collection*

# Clang (2005)

- Projeto concorrente de GCC
- Desenvolvido por Apple
- Baseado na LLVM
- Guerra GNU GPL / BSD
- Boa base para ferramentas de análises de código

# Extensões dos arquivos para C

As extensões padrão dos arquivos em C são

| Extensão | Uso |
|---|---|
| .h | cabeçalhos |
| .c | código fonte |
| .i | código fonte preprocessado |
| .cpp | código fonte (C++) |
| .o | código objeto |

# Documentação (offline)

- Sistemas da família Unix vem acompanhados com bastante documentação
- Acessível sem usar um navegador (offline)

## man

- A documentação da biblioteca padrão da linguagem C está na seção 3 do manual
- `man [secao] <nome>`

```
man man
man 3 printf
```

## info

- Documentação navegável

```
info man
```

## apropos

- Pesquisa das páginas de manual que falam de uma palavra

```
apropos printf
```

# Navegação

## cd

- cd = *change directory*

```
cd dir
cd ..
cd dir2
cd -
cd
```

## pwd

- Qual é o nome da pasta corrente ?

```
pwd
```

# Ler/escrever

## cat

- Concatenar arquivos e escrever na saída padrão (tela)

```
cat file.c
cat file1.c file2.c
```

## echo

- escrever na tela
  - uma mensagem
  - o valore de uma variável do ambiente

```
echo "Hello World!"
echo $SHELL
```

# Descobrir elementos

## ls

- Listar uma pasta

```
ls
ls *.c
```

## less

- Decompôr a saída em páginas visíveis um depois da outra

```
apropos printf | less
```

## file

- Qual é o tipo de um arquivo ?

```
file dir
file file1.c
```

# Manipulação de arquivos

## cp

- Copiar um arquio / uma pasta

```
cp file1.c foo.c
cp -R dir1 dir2
```

## mv

- mover um arquivo/uma pasta

```
mv file1.c plop.c
```

# Manipulação de arquivos

## rm / rmdir

- Remover um arquivo
- Remover uma pasta

```
rm *.c
rm -Rf dir
```

## mkdir

- Criação de uma nova subpasta da pasta corrente

```
mkdir pasta_minha
cd pasta_minha
pwd
```

# O pipe | e a redireção

## |

- O *pipe* (barra) serve para ligar a saída de um comando à entrada de um outro

```
cat file1.c file2.c | less
```

## Redireção

```
cat file1.c file2.c > file3.c
echo "foo" >> file3.c
ls *.c > cfiles.txt
sort < cfiles.txt > sorted_cfiles.txt
```

# Wildcards

- O símbolo * é usado como *wildcard*
- Isso representa "qualquer sequência de caracteres"
- `*.c` = todos os arquivos com extensão ".c"

## Exemplo

```
ls *.c
rm *.c
```

# Completar nomes

O interpretador de comando (*shell*) permite completar os nomes com a tecla `TAB`

| Digite | | Resultado |
|---|---|---|
| les<TAB> | $\rightarrow$ | less |
| apro<TAB> | $\rightarrow$ | apropos |

# Conteúdo geral de uma linguagem de programação

1. Atribuição
2. Declaração de variáveis
3. Sequência
4. Teste / condicional
5. Laços
6. Entrada/saída

# Hello, World!

```c
#include <stdio.h>

int main (void)
{
    printf("Hello, World!\n");
    return 0;
}
```

- 1 : incluir informação da biblioteca padrão (E/S)
- 3 : definição duma função chamada `main`, sem parâmetros
- 4 : chaves `{ }` definem o corpo (i.e. os comandos) da função
- 5 : chamada de função, `\n` representa o caractere de nova linha.

# Conversão de temperaturas

- 3 : comentários

```c
#include <stdio.h>

/* Escreva tabela de conversão
 * graus Celsius / graus Fahreneit
 * de 0 a 100 (de 10 em 10)
 */
int main(int argc, char *argv[])
{
     int fahr, celsius;
     int lower, upper, step;

     lower = 0;
     upper = 100;
     step = 10;

     while (celsius <= upper) {
 fahr = 32 + celsius * 9 / 5;
 printf("%d -> %d\n", celsius, fahr);
 celsius += step;
     }
     return 0;
}
```

# Conversão de temperaturas

- 3 : comentários
- 7 : definição função principal, com parâmetros

```
1  #include <stdio.h>
2
3  /* Escreva tabela de conversão
4   * graus Celsius / graus Fahreneit
5   * de 0 a 100 (de 10 em 10)
6   */
7  int main(int argc, char *argv[])
8  {
9       int fahr, celsius;
10      int lower, upper, step;
11
12      lower = 0;
13      upper = 100;
14      step = 10;
15
16      while (celsius <= upper) {
17   fahr = 32 + celsius * 9 / 5;
18   printf("%d -> %d\n", celsius, fahr);
19   celsius += step;
20      }
21      return 0;
22 }
```

# Conversão de temperaturas

- 3 : comentários
- 7 : definição função principal, com parâmetros
- 9 - 10 : declaração de variáveis

```
1  #include <stdio.h>
2
3  /* Escreva tabela de conversão
4   * graus Celsius / graus Fahreneit
5   * de 0 a 100 (de 10 em 10)
6   */
7  int main(int argc, char *argv[])
8  {
9       int fahr, celsius;
10      int lower, upper, step;
11
12      lower = 0;
13      upper = 100;
14      step = 10;
15
16      while (celsius <= upper) {
17 fahr = 32 + celsius * 9 / 5;
18 printf("%d -> %d\n", celsius, fahr);
19 celsius += step;
20      }
21      return 0;
22 }
```

# Conversão de temperaturas

- 3 : comentários
- 7 : definição função principal, com parâmetros
- 9 - 10 : declaração de variáveis
- 12 - 14 : atribuição, inicialização de variáveis

```
1  #include <stdio.h>
2
3  /* Escreva tabela de conversão
4   * graus Celsius / graus Fahreneit
5   * de 0 a 100 (de 10 em 10)
6   */
7  int main(int argc, char *argv[])
8  {
9       int fahr, celsius;
10      int lower, upper, step;
11
12      lower = 0;
13      upper = 100;
14      step = 10;
15
16      while (celsius <= upper) {
17   fahr = 32 + celsius * 9 / 5;
18   printf("%d -> %d\n", celsius, fahr);
19   celsius += step;
20      }
21      return 0;
22 }
```

# Conversão de temperaturas

- 3 : comentários
- 7 : definição função principal, com parâmetros
- 9 - 10 : declaração de variáveis
- 12 - 14 : atribuição, inicialização de variáveis
- 16 : laço `while`

```
1  #include <stdio.h>
2
3  /* Escreva tabela de conversão
4   * graus Celsius / graus Fahreneit
5   * de 0 a 100 (de 10 em 10)
6   */
7  int main(int argc, char *argv[])
8  {
9       int fahr, celsius;
10      int lower, upper, step;
11
12      lower = 0;
13      upper = 100;
14      step = 10;
15
16      while (celsius <= upper) {
17   fahr = 32 + celsius * 9 / 5;
18   printf("%d -> %d\n", celsius, fahr);
19   celsius += step;
20      }
21      return 0;
22 }
```

# printf

`printf` é a função principal de saída de `C`

- é definida no cabeçalho `<stdio.h>`
- tem 2 tipos de parâmetros formais
  1. Um texto, chamado de **formato**
  2. Uma lista de parâmetros a serem impressos de acordo com a especificação de formato.
- cada instrução de formato % é ligada ao valor correspondente na lista dos parâmetros fornecida

## Exemplo

- `printf("%d", 1)` escreve `1`
- `printf("%.2f", 3.14598)` escreve `3.14`

# Definir, acessar, alterar variáveis

## Definição

- Uma variável possui um tipo de informação que ela guarda
- Declaração: `<tipo> <nome>` ou `<tipo> <nome> = <expressão>`
- Nomes possíveis sequência de letras, números e sublinhado, começando com letra ou sublinhado.
  `[a-zA-z_][a-zA-Z_0-9]*`

## O acesso é feito por o nome da variável, a alteração com =

```c
#include <stdio.h>
int main(void)
{
    int n, m, s;
    m = 14;
    n = 23;
    s = m + n;
    printf("n = %i, m = %i, s = %i", n, m, s);
    return(0);
}
```

# Tipo de dados

## Tipos básicos

| Nome | Tipos | Exemplo de valor | Bytes |
|---|---|---|---|
| Caractere | char | ‘a’, ‘b’, ‘8’, ‘!’ | 1 |
| Inteiro | char, short, int, long | 0, 42, 9234324 | 1 - 8 |
| Ponto flutuante | float, double | 3.14, | 4 - 8 |
| Booleano | bool | true, false (include stdbool.h) | 4 |

## O que é ?

Um tipo de dado define:

- o tamanho de representação em bytes/bits (1/8, 2/16, 4/32, 8/64)
- a representação dos dados usando os bits
- C permite o acesso a essa representação: **tem que conhecê-la**!
- O compilador faz a conversão entre os dados manipulados no seus programas e a representação binária deles.

# Tamanho dos dados

```c
#include <stdio.h>
#include <stdbool.h>

int main(int argc, char *argv[])
{
    printf("char = %u\nshort = %u\nint = %u\nlong = %u\nlonglong = %u\n",
   sizeof(char), sizeof(short), sizeof(int),
   sizeof(long), sizeof(long long) );
    printf("float = %u\ndouble = %u\n", sizeof(float), sizeof(double));
    printf("bool = %u\n", sizeof(bool));
    return 0;
}
```

| | | |
|---|---|---|
| char | = | 1 |
| short | = | 2 |
| int | = | 4 |
| long | = | 8 |
| longlong | = | 8 |
| float | = | 4 |
| double | = | 8 |
| bool | = | 1 |

# Notação posicional

- O valor dos algarismos depende da posição e da base $b$

$$x = a_n b^n + a_{n-1} b^{n-1} + \ldots a_0 b^0 + \ldots + a_{-m} b^{-m}$$

- A base usual é a base 10
  - $13 = 1 * 10^1 + 3 * 10^0$
  - $456 = 4 * 10^2 + 5 * 10^1 + 6 * 10^0$
- A representação do número treze (13) pode ser outra

| Base | Representação | Valor |
|---|---|---|
| 2 | $1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0$ | 1101 |
| 8 | $1 * 8^1 + 5 * 8^0$ | 15 |
| 16 | $13 * 16^0 = d * 16^0$ | d |

# Valores inteiros

## Em C

- O valor interna da variável é **binário**
- O programador pode, dentro do programa, manipulá-la com uma visão/representação:
  - binário (0,1),
  - octal(0-7),
  - decimal(0-9),
  - hexadecimal(0-9a-f)

# Manipulação

## Impressão

- A função `printf` permite a impressão dum valor usando essas vistas diferentes

```c
#include <stdio.h>
int main(void)
{
     int n = 45;
     int k = 0x12;
     int m = 037;
     printf("%x, %o, %d, %d\n", n, k, m, k);
     return 0;
}
```

2d 22 31 18

# Operadores aritméticos

| Operador | Significado | Tipos |
|---|---|---|
| - | subtração, menos unário | todos |
| + | adição, + unário | todos |
| / | divisão | todos |
| * | multiplicação | todos |
| % | resto | todos salvo double ou float |

# Operações sobre inteiros

```c
#include <stdio.h>

int main (void) {
  int a = 55;
  int b = 3;
  printf("%i, %i, %i, %i, %i\n", a + b, a - b, a / b, a * b, a % b);
  return(0);
}
```

58 52 18 165 1

# Exemplos

```c
#include <stdio.h>

int main(void) {
  int fact5;

  fact5 = 1;
  fact5 = 1 * fact5;
  fact5 = 2 * fact5;
  fact5 = 3 * fact5;
  fact5 = 4 * fact5;
  fact5 = 5 * fact5;
  printf("Factorial(5) = %i\n", fact5);
  return(0);
}
```

```
Factorial(5) = 120
```

# Precedências

## Exemplos

Qual é o valor de

1. 2 + 3 * 4
2. 2 - 3 - 4
3. 5 + 6 /2
4. 2 * 8 / 4
5. 6 / 3 / 2
6. 6 - 3 / 2

## No C

- Ordem aritmética, como em matemática $2 + 3 * 4 \rightarrow 2 + (3 + 4)$ (- unário) > (*, /, %) > (+, - binário)
- Operadores de mesma prioridade são associativos ã esquerda $2 - 3 + 4 \rightarrow (2 - 3) + 4$
- O uso de parenteses permite definir a ordem desejada

# Aritmética modulo

- Variáveis têm um valor limite de representação
- O cabeçalho `limits.h` define os limites para sua arquitetura.

## Exemplo

`INT_MAX` ($2^{31} - 1$), `INT_MIN` ($-2^{31}$) são os limites para os inteiros

- `INT_MAX + 1` $\rightarrow$ `INT_MIN`
- a aritmética do computador é uma aritmética **modulo**

```c
#include <stdio.h>
#include <limits.h>
int main(void)
{
    printf("%d, %d", INT_MIN, INT_MAX);
    return 0;
}
```

```
-2147483648    2147483647
```

# Observações estilísticas

- O estilo que eu vou usar é o estilo dito "K&R"
- Uma outra possibilidade é o estilo BSD
- Veja `http://en.wikipedia.org/wiki/Indent_style#Styles`

## K & R

```c
#include <stdio.h>

int main(void)
{
    int i;

    for (i = 1; i < 10; i++) {
printf("%d\n", i);
    }
    return 0;
}
```

## BSD

```c
#include <stdio.h>

int
main(void)
{
int i;

for (i = 1; i < 10; i++) {
printf("%d\n", i);
}
return 0;
}
```

# Referências

- Linha de comando [CLI, CLP]
- Cap. 1 de [KR88]
- Cap. 1 de [KP99]



*The Command Line Crash Course*, `http://cli.learncodethehardway.org/book/`.


*Linux Command*, `http://linuxcommand.org/`.


Brian W. Kernighan and Rob Pike, *The Practice of Programming*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.


Brian W. Kernighan and Dennis M. Ritchie, *The (ANSI) C Programming Language*, 2nd ed., Prentice Hall Professional Technical Reference, 1988.

# Perguntas ?


http://dimap.ufrn.br/~richard/dim0321