Linguagem de Programação II

Conceitos de Suporte

Universidade do Estado do Rio de Janeiro-UERJ
Instituto de Matemática e Estatística-IME
Ciência da Computação
Professor: Alexandre Sztajnberg

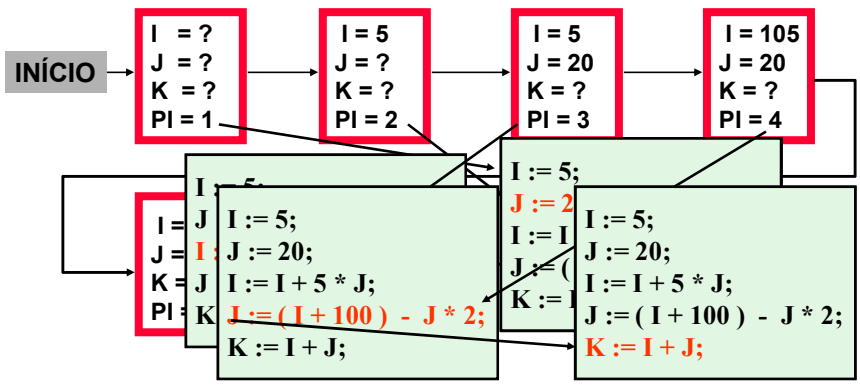
Programa versus Processo

- "processo é um programa em execução"
- o processo possui um estado e um contexto ("é um programa com alma")
- o desempenho previsto para o programa é teórico
- o desempenho de um processo é relativo ao ambiente que o hospeda e o executa
  - HW, SO
  - outros processos

Programa pascal

```
PROGRAM SIMPLES;
LAL 1,2,3,4,5;
VAR I,J,K : INTEGER;
BEGIN
   1: I := 5;
   2: J := 20;
   3: I := I + 5 * J;
   4: J := (I + 100) - J * 2;
   5: K := I + J;
END.
```

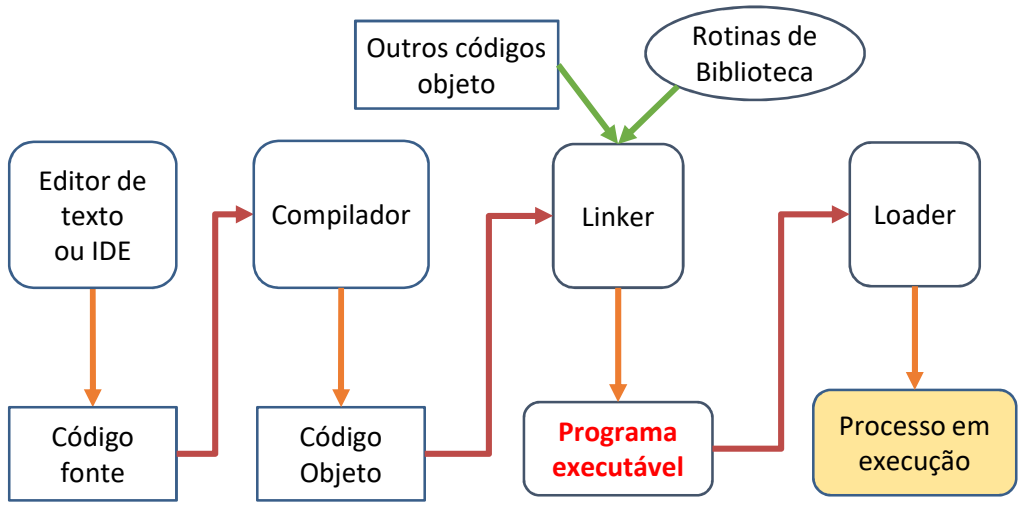
Programa em Execução: Processo



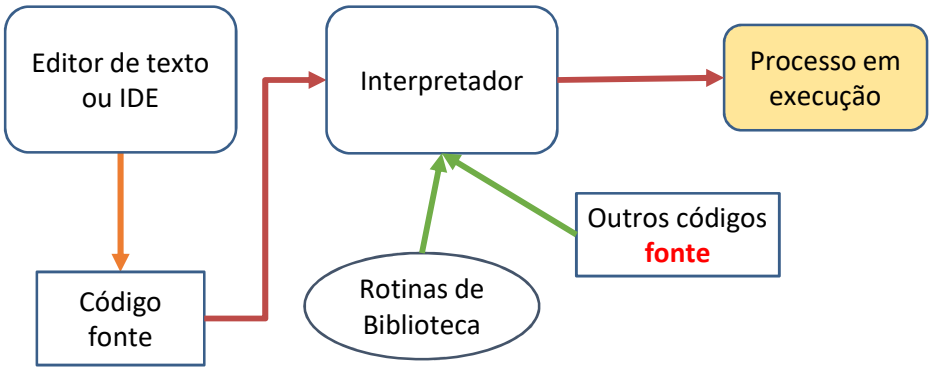
Linguagem de programação

- Linguagem em alto nível, possui estruturas
- Sintaxe entendível (também) por humanos
- Exemplos
  - Python: `print("Hello world")`
  - Java: `System.out.println("Hello world")`
  - JavaScript: `console.log("Hello world")`
  - PHP
  - C e C++

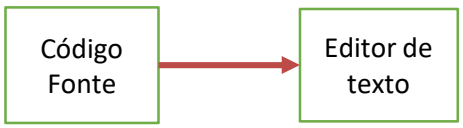
Processo de desenvolvimento com compilador



Processo de desenvolvimento com interpretador



Editor de Texto

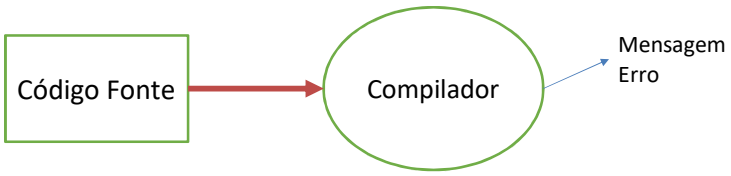


- Sublime text
- Aton
- Visual Studio Code
- Notepad ++

IDE

- Eclipse
- NetBeans
- PyCharm
- IntelliJ IDEA

Compilação



- Tradução de um código fonte para código mais próximo do processador
  - Código de máquina
  - Código intermediário ou linguagem-alvo (*target*)
- O compilador pode realizar os seguintes passos para fazer a tradução
  - Pré-processamento
  - Análise sintáxica
  - Análise semântica
  - Optimização de código
  - Geração de código-objeto

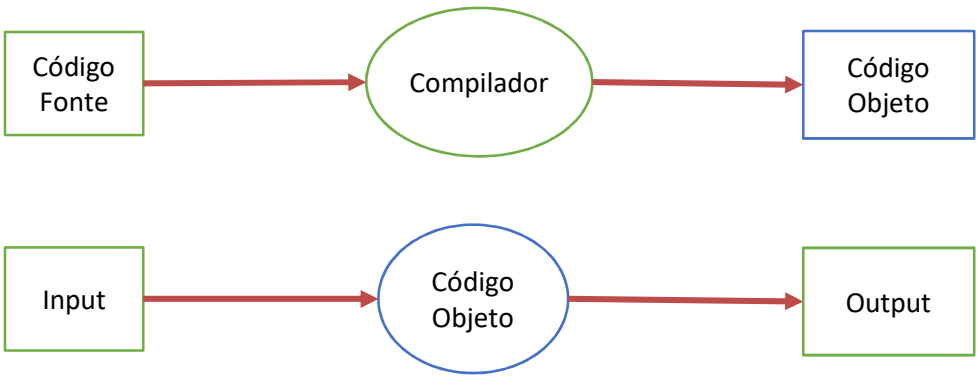
Do Programa-fonte para o Código Executável

```
program gcd(input, output);
var i, j: integer;
begin
  read(i, j);
  while i <> j do
     if i > j then i := i - j;
     else j := j - i;
  writeln(i)
end.
```

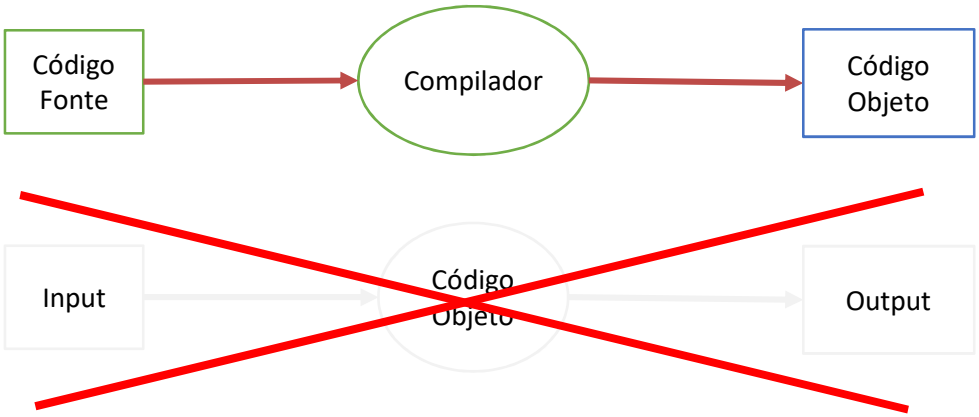
Compilação

```
27bdffd0 afbf0014 0c1002a8 00000000 0c1002a8 afa2001c 8fa4001c
00401825 10820008 0064082a 10200003 00000000 10000002 00832023
00641823 1483fffa 0064082a 0c1002b2 00000000 8fbf0014 27bd0020
03e00008 00001025
```

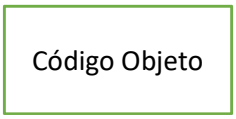
Compilação



Compilação



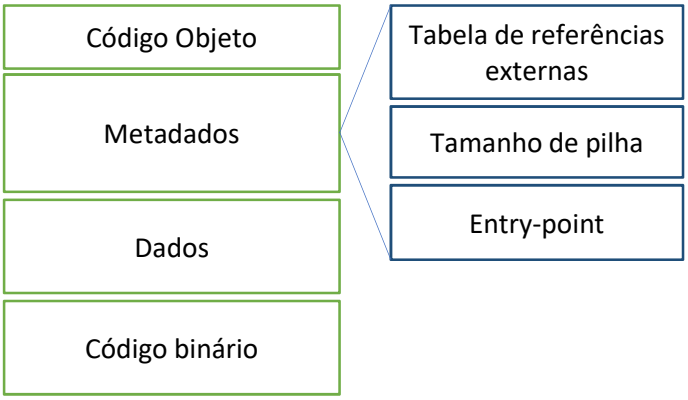
Código objeto



- O código objeto (ou código-objeto, ví as duas formas) é gerado pelo compilador no processo da tradução do código fonte
- O código objeto só é criado se não houver erro de sintaxe no código fonte que será traduzido
- O código objeto salvo em um arquivo e **contém uma ESTRUTURA com vários campos**
- Um dos campos é, justamente, a tradução do programa para código de máquina (ou em "linguagem-alvo, ou *target program*).

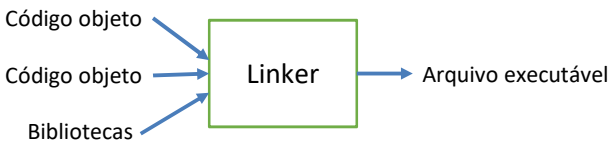
- **Importante**
  - A estrutura, a organização desta estrutura, é dependente do Sistema Operacional
  - O código de máquina gerado na tradução é dependente do processador
  - O código objeto tem referências (endereços) incompletas, cruzadas, que precisam ser resolvidas ...

Estrutura do Código Objeto



- **Common Object File Format** (**COFF**) is a format for executable, object code, and shared library computer files used on Unix systems. It was introduced in Unix System V
- **Relocatable Object Module Format (OMF)** is an object file format used primarily for software intended to run on Intel 80x86 microprocessors.
- **Executable and Linkable Format** (**ELF**, formerly named **Extensible Linking Format**), is a common standard file format for executable files, object code, shared libraries, and core dumps.

Linker

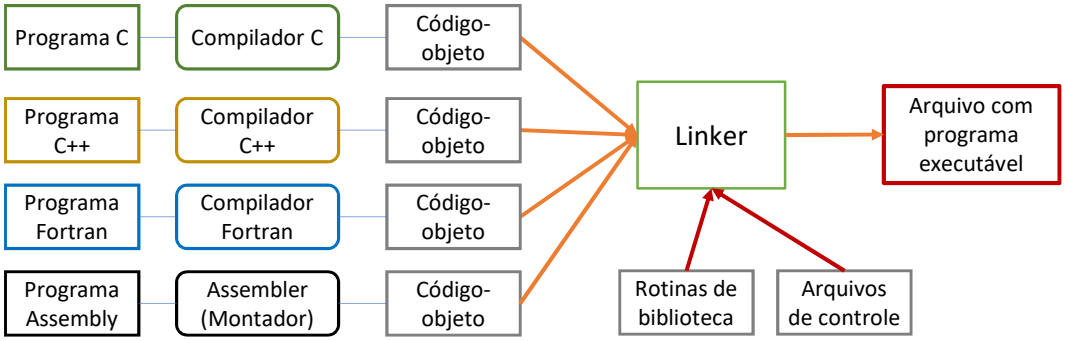


- O código objeto, via de regra, utiliza rotinas e variáveis externas
  - De outras partes do programa que você está desenvolvendo, ou seja, outros código-objeto
  - De bibliotecas
  - ... E de DLLs ou Shared Objects .so (mas isso é outro assunto ...)

  ... Então como gerar o código executável?

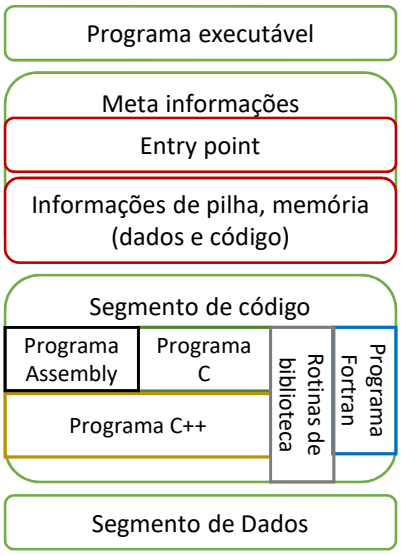
- Linker, ligador ou link-editor é responsável por
  - Cruzar as referências de bibliotecas padrão e bibliotecas compartilhadas ao código objeto
  - Cruzar um ou mais códigos objetos
  - Organizar num segmento de dados todos os códigos de máquina, agora com as referências cruzadas já resolvidas
  - Gerar o código executável e salvá-lo num arquivo
    - O Executável também é uma estrutura com vários campos.

Linker



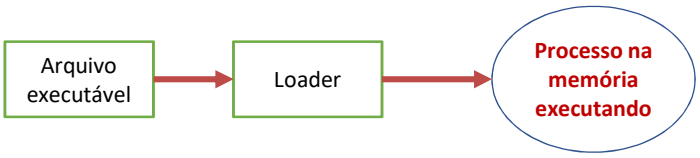
- Posso desenvolver um programa com módulos programados em linguagens diferentes?

Programa executável



- Um arquivo executável é também uma estrutura
- Segmento de código contém uma "colagem" dos códigos ligados
- Para onde aponta o *entry point*?

Loader



- o *loader* (carregador) é uma rotina/módulo do sistema operacional que
  - Lê o arquivo executável do disco (ou outro dispositivo de persistência, inclusive a rede)
  - Interpreta as meta-informações do arquivo executável e prepara o ambiente de execução
  - Aloca memória para código, dados e pilha
  - Aciona o SO para criar os identificadores (PID, por exemplo) e alocar estruturas de controle do novo processo
  - Chama o despachante para "passar a bola" (o controle do processador) para o novo processo

O novo processo começa, então, a execução. Começa de onde???

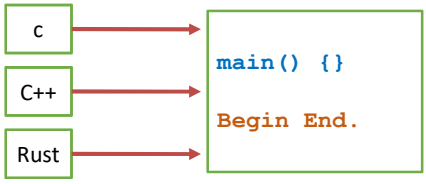
Outras funções do Loader

- Validar o programa para requisitos de memória, permissões, etc.
- Copiar os arquivos necessários, como a imagem do programa ou as bibliotecas necessárias, do disco para a memória.
- Copiar os argumentos de linha de comando necessários para a pilha
- Ligar o ponto de partida do programa e ligue qualquer outra biblioteca necessária
- Inicializar os registros
- Saltar para o ponto de partida do programa na memória

Entry-Point

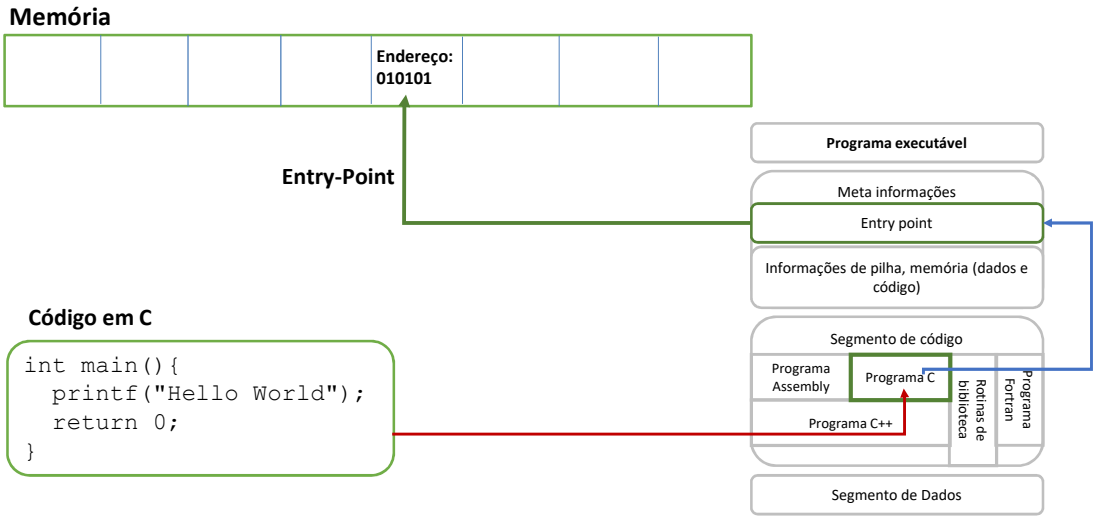
- O entry-point é um endereço de memória que indica onde o programa, função ou módulo começa
- Lembre-se que o código do programa pode ser composto por vários módulos/rotinas cuja origem pode ser de códigos fonte escritos em linguagens de programação distintas
- O entry-point não precisa necessariamente estar no endereço 0 do segmento de código, ou no primeiro endereço do working-set da memória do processo recém criado
- Ele também pode apontar para um ponto de entrada de uma biblioteca ou wrapper

Entry-Point



- Em algumas linguagens de programação o entry-point é identificado pelo procedimento/método *main* indicando que é a partir deste ponto no código fonte que o programa começa
- O código fonte ao ser traduzido em código executável irá fazer uma sinalização que informará o endereço de memória referente ao ponto de início(método *main*)

Entry-Point



Variáveis de Ambiente

Importante: o *loader* precisa, primeiro, carregar o arquivo executável de disco. E se ele não souber onde o arquivo está ou se não consegue encontrar o mesmo?

- A maioria dos sistemas operacionais usam o conceito de variável de ambiente
- Uma variável de ambiente é simplesmente uma dupla chave / valor
- Um programa, incluindo um *script*, pode usar uma ou mais variáveis de ambiente como quiser
  - Toda linguagem de programação permite criar e ler variáveis de ambiente
  - O formato específico de como o valor da variável é organizado é arbitrário
- O valor de uma variável de ambiente pode ser, por exemplo
  - A sequencia de diretórios pelos quais o loader deve pesquisar a existência de um arquivo executável
  - o editor padrão que deve ser usado
  - configurações locais do sistema
- As variáveis de ambiente podem ser criadas e editadas por utilitários ou comandos do sistema operacional

Usos das Variáveis de Ambiente

- Modo de execução, informações do sistema, localização de diretórios específicos
- Nomes de domínio
- URL / URI da API
- Endereços de correio de grupo, como os de marketing, suporte, vendas
- Nomes de contas de serviço
- Variáveis específicas de programas
  - **JAVAPATH**
  - **CLASSPATH (lembrem-se dessa!)**

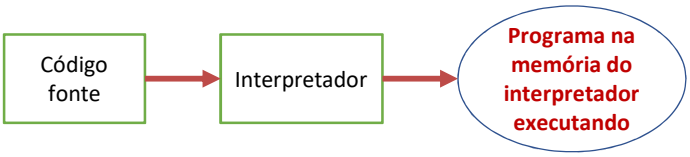
```
Microsoft Windows [versão 10.0.19041.804]
C:\Users\alexszt>set
ANDROID_HOME=C:\Android\Sdk
APPDATA=C:\Users\alexszt\AppData\Roaming
NUMBER_OF_PROCESSORS=4
Path=C:\Program Files (x86)\Common
Files\Oracle\Java\javapath;C:\ProgramData\Oracle\Java\java
path;C:\WINDOWS\system32;C:\WINDOWS;C:\Program Files\Java\
jdk1.8.0_131\bin;[...] C:\Program Files\nodejs\;C:\Program
Files\Git\cmd;C:\Users\alexszt\AppData\Local\Microsoft\Win
dowsApps;C:\Program Files (x86)\Nmap;c:\src\flutter\bin;
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.MSC
PROCESSOR_ARCHITECTURE=AMD64
PROMPT=$P$G
TEMP=C:\Users\alexszt\AppData\Local\Temp

Linux
PATH=/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/sbin:/usr/lo
cal/bin
CLASSPATH=./:/home/alexszt/exec-pac/app/:/home/alexszt/
exec-pac/excp/:/home/alexszt/exec-pac/lib/
```

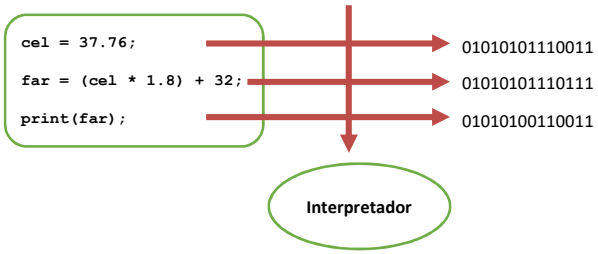
Variável de ambiente PATH

- A maioria dos sistemas operacionais (Windows, MacOS, Linux, Android) usa uma variável de ambiente chamada PATH
  - PATH=<Sequencia de diretórios>
  - A forma de separação dos itens da sequencia é dependente do sistema
  - O *loader* segue esta sequencia para e, em cada diretório verifica se o arquivo contendo o programa executável solicitado ali se encontra
    - Caso positivo, usa o arquivo para efetivamente carregar o programa e criar o processo para executar
    - Caso negativo, passa para o próximo
    - E se não encontrar?
- Alguns aplicativos incluem seus diretórios de controle ou de arquivos executáveis no PATH durante a instalação
- Qualquer aplicação pode ler a variável PATH também
- Diferenças entre o Windows e o Linux
  - Além da diferença da separação dos itens da sequencia dos diretórios: diretório corrente – “**./**”

Interpretação



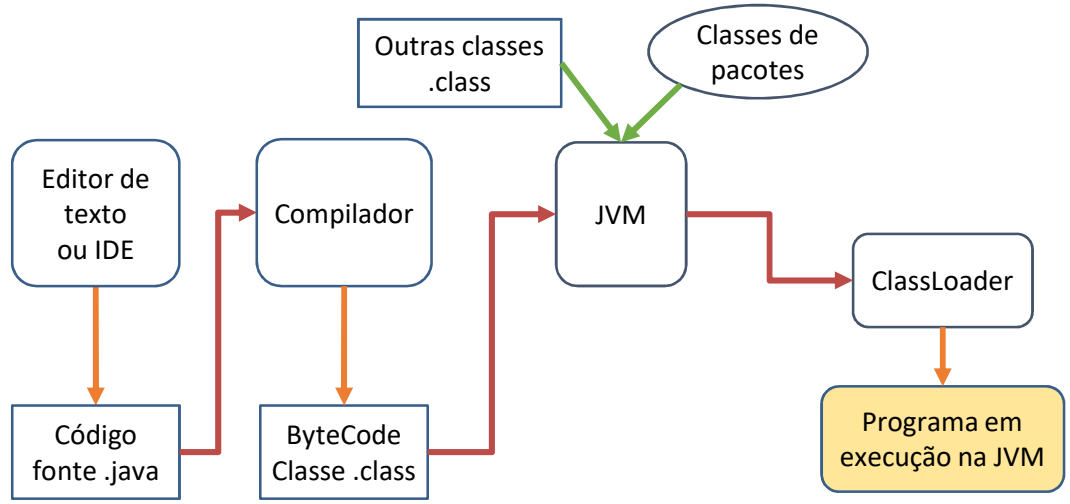
- Os interpretadores são programas
  - Fazem a tradução do código fonte, linha por linha, em tempo de execução
  - A cada tradução, código de máquina (ou em linguagem alvo) é gerada e executada
- Os interpretadores precisam ser chamados todas as vezes que o programa é executado
  - O código fonte é carregado como parâmetro de entrada
  - Nova tradução é realizada a cada execução do programa



Compilação versus interpretação

- Vantagens?
- Desvantagens?

Processo de desenvolvimento em Java



Compilação em Java

- Em Java a compilação ocorre de forma diferente, o compilador traduz o código fonte em um código binário intermediário, denominado ByteCode
- O ByteCode
  - Padrão do Java
  - Todas as JVMs são preparadas para interpretar o mesmo ByteCode

Interpretação em Java

- ByteCode é interpretado e executado por uma JVM(Java Virtual Machine, Máquina Virtual Java)
- A interpretação é um dos processos que ocorrem na JVM, convertendo o ByteCode em operações de processador, e executando as operações
- Como o ByteCode é uma linguagem binária, este passo é relativamente eficiente

Exercícios

- Na nossa VM, Linux, identifique:
  - As suas variáveis de ambiente
  - Altere os arquivos de inicialização para incorporar os diretórios necessários para você usar o Java sem precisar informar o diretório completo
  - Identifique e explore o compilador C, C++ e java
  - Explore as ferramentas de inspeção de código objeto e executável
  - Explore as ferramentas de ligação
  - Explore as ferramentas de biblioteca e “archive”
  - Explore os editores de texto disponíveis: vi e mcedit
  - Faça o curso básico de Linux da Cisco Academy