

# Computadores de Programação (DCC/UFRJ)

## Aula 20: Ligação e carga de programas

Prof. Paulo Aguiar

1 Ligação e carga de programas

2 Arquivos objeto

3 Referências bibliográficas

# Ligação e carga de programas

**Ligação** é o processo de coletar e combinar vários pedaços de código e dados em um único arquivo que pode ser **carregado** (copiado) na memória e executado

**Carga** é o processo de copiar o programa na memória e transferir o controle da execução para ele

# Tempo de ligação

A **ligação** pode ser feita:

- 1 **durante a compilação:** quando o código fonte é traduzido para linguagem de montagem (pelo compilador)
- 2 **durante a carga:** quando o programa é carregado na memória (pelo carregador)
- 3 **durante a execução:** quando o programa está sendo executado

# Vantagem do processo de ligação

Permite a **compilação em partes**: não requer que as aplicações sejam constituídas de um único arquivo fonte

Quando apenas uma parte do código é alterada, permite que apenas esta parte seja recompilada e ligada com os módulos já compilados, evitando a recompilação de todo o código da aplicação

# Importância de entender o processo de ligação

## Depuração e manutenção de grandes programas

- Normalmente acontecem erros como módulos faltantes, bibliotecas faltando, uso de bibliotecas com versões incompatíveis
- Evita a recompilação de todo o código para a atualização parcial de bibliotecas, que é um processo frequente

## Compreensão de conceitos fundamentais

- Ajuda a compreender a diferença em definir uma variável ou função com o atributo `static` ou as implicações entre uma variável ser local ou global
- Permite entender os códigos objetos executáveis e seu impacto na carga e execução de programas, e no uso de memória virtual e paginação

# Controladores de compilação

- A maioria dos sistemas de compilação provê um **controlador de compilação** que invoca:
  - 1 pre-processador
  - 2 compilador
  - 3 montador e
  - 4 ligador
- ... de forma transparente para o programador

## Exemplo: o sistema de compilação GNU

- No **sistema de compilação GNU**, o controlador de compilação é o **gcc**
- (*Ver exemplos nos arquivos de código em anexo (ex1-estatica)*)





# O controlador de compilação gcc

- Quando invocamos o GCC, ele faz (por default) todas as etapas de construção de um programa executável: pre-processamento, compilação, montagem e ligação
- As opções do GCC permitem parar o processo de construção em estágios intermediários, por ex., a opção **-c** diz para não chamar o **ligador**

# Carga e execução dos programas

Quando o programa executável é chamado no **shell** (ou ativado por dois cliques em uma janela), o programa shell (ou o tratador de eventos da janela) chama uma função do Sistema Operacional – “*loader*” – que copia o código e dados do arquivo executável para a memória e transfere o controle do processador para a primeira instrução do programa

# Compiladores e Montadores versus Ligador

Códigos objeto referenciam símbolos e consistem de blocos de dados e código, com as instruções, variáveis globais inicializadas e variáveis não inicializadas em diferentes seções

Compiladores e montadores fazem a maior parte do trabalho que requer conhecimento sobre a **máquina alvo** e geram código e dados começando no endereço 0

O ligador realoca e concatena os blocos, decidindo sobre localização em tempo de execução e modificando localizações nos blocos de código e de dados

# Arquivos objeto

Podem ser de três formas:

- 1 **arquivo objeto realocável**
- 2 **arquivo objeto executável**
- 3 **arquivo objeto compartilhável**

# Arquivo objeto realocável

- Contém **código binário e dados** em formato que permite ser combinado com o conteúdo de outros arquivos objeto realocáveis
- O processo de combinação ocorre em **tempo de compilação** e cria um **arquivo objeto executável**

# Arquivo objeto executável e compartilhável

## Arquivo objeto executável

Contém **código binário e dados** em formato que permite ser copiado para a memória diretamente, e executado (gerado por ligador)

## Arquivo objeto compartilhável

Tipo especial de **arquivo objeto realocável** que pode ser carregado na memória e ligado dinamicamente, em **tempo de carga ou de execução** (gerado por compilador e montador)

# Estrutura geral dos arquivos objeto

- Arquivos objeto são coleções de blocos de bytes
- Alguns blocos contêm **código**, outros **dados** e outras estruturas de dados que servem de apoio para o ligador e para o carregador



## Módulo objeto e arquivo objeto

- Um **módulo objeto** é uma sequência de bytes
  - Um **arquivo objeto** é um módulo objeto armazenado no disco como um arquivo
- 
- O **formato do arquivo objeto** depende do sistema operacional
  - ELF (*Executable and Linkable Format*) é o formato de arquivo objeto do Linux e de versões mais modernas de Unix

# Formato de arquivo objeto ELF

## Cabeçalho de arquivo ELF

- 1 Tamanho da palavra e ordenação de bytes do sistema que gerou o arquivo
- 2 Tamanho do cabeçalho
- 3 Tipo do arquivo objeto (realocável, executável, compartilhável)
- 4 Tipo de máquina (ex., IA32)
- 5 Offset da tabela de cabeçalho de seção (descreve as seções do arquivo objeto)
- 6 Tamanho e número de entradas na tabela de cabeçalho de seção

(executar: `readelf -h main.o`)

# Seções de um arquivo objeto ELF

- 1 **.text**: código de máquina do programa compilado
- 2 **.rodata**: dados de leitura apenas  
(ex., formato de string para printf e tabelas de desvio de switch)
- 3 **.data**: variáveis globais inicializadas
- 4 **.bss**: variáveis globais não-inicializadas
- 5 **.symtab**: tabela de símbolos  
(funções e variáveis globais definidas/referenciadas no programa)
- 6 **.rel.text**: localizações a ser modificadas na combinação com outros arquivos  
(i.e., instrução que chama uma função externa ou referencia uma variável global)
- 7 **.rel.data**: informações de realocação para variáveis globais definidas ou referenciadas pelo módulo
- 8 **.debug**: tabela de símbolo de depuração (se compilado com a opção -g)
- 9 **.line**: mapeamento de linhas entre código fonte e instruções de máquina  
(se compilado com a opção -g)
- 10 **.strtab**: tabela de string

(Executar: `readelf -s main.o`)

## .data versus .bss

- A distinção entre variável **global inicializada** (.data) e variável **global não-inicializada** (.bss) é feita por questão de eficiência do uso do espaço de memória: **variáveis não inicializadas não precisam ocupar espaço no disco**

# Referências bibliográficas

- *Computer Systems—A Programmer's Perspective* (**Cap. 7**)