DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

UNIVERSIDADE ESTADUAL DE CAMPINAS

Programação de Sistemas: Uma Introdução

Ivan Luiz Marques Ricarte

`http://www.dca.fee.unicamp.br/~ricarte/`

2004

Impressão gerada em 2 de março de 2004

Capítulo 1

Software de sistema

Nas últimas décadas, o computador deixou de ser uma ferramenta cara e exclusiva dos grandes centros de pesquisas e passou a ser presença constante no cotidiano de todos. Apesar dessa "quase onipresença," para muitos ainda há uma aura de mistério no que se refere ao funcionamento e à capacidade funcional de computadores.

Um computador tem essencialmente duas componentes, *hardware* e *software*. *Hardware* é o termo coletivo que representa todo o conjunto de circuitos que permite o funcionamento do computador. *Software* é o termo coletivo expressando o conjunto de programas que, juntamente com o *hardware*, permite a operação de computadores. Para a maior parte dos sistemas computacionais modernos, o *hardware* é genérico, sendo que o *software* é que determina quais são as funcionalidades do sistema oferecidas aos usuários finais.

O principal objetivo deste texto é apresentar a parte do *software* que não está associado especificamente a alguma aplicação, mas que deve estar presente em qualquer sistema para permitir o desenvolvimento e execução dessas aplicações.

Um computador nada mais faz do que executar **programas**. Um programa é simplesmente uma seqüência de instruções definida por um programador. Assim, um programa pode ser comparado a uma receita indicando os passos elementares que devem ser seguidos para desempenhar uma tarefa. Cada instrução é executada no computador por seu principal componente, o processador ou CPU (do inglês para unidade central de processamento).

Cada processador entende uma linguagem própria, que reflete as instruções básicas que ele pode executar. O conjunto dessas instruções constitui a **linguagem de máquina** do processador. Cada instrução da linguagem de máquina é representada por uma seqüência de bits distinta, que deve ser decodificada pela unidade de controle da CPU para determinar que ações devem ser desempenhadas para a execução da instrução.

Claramente, seria extremamente desconfortável se programadores tivessem que desenvolver cada um de seus programas diretamente em linguagem de máquina. Programadores não trabalham diretamente com a representação binária das instruções de um processador. Existe uma descrição simbólica para as instruções do processador — a linguagem *assembly* do processador — que pode ser facilmente mapeada para

sua linguagem de máquina.

No entanto, mesmo *assembly* é raramente utilizada pela maior parte dos programadores, que trabalha com **linguagens de programação de alto nível**. Em linguagens de alto nível, as instruções são expressas usando palavras, ou termos, que se aproximam daquelas usadas na linguagem humana, de forma a facilitar para os programadores a expressão e compreensão das tarefas que o programa deve executar. Várias linguagens de alto nível estão disponíveis para diversas máquinas distintas. Essa independência de um processador específico é uma outra vantagem no desenvolvimento de programas em linguagens de alto nível em relação ao uso de *assembly*. Em geral, cada linguagem de alto nível teve uma motivação para ser desenvolvida. BASIC foi desenvolvida para ensinar princípios de programação; Pascal, para o ensino de programação estruturada; FORTRAN, para aplicações em computação científica; lisp e Prolog, para aplicações em Inteligência Artificial; Java, para o desenvolvimento de *software* embarcado e distribuído; e a linguagem C, para a programação de sistemas.

Um programa, desde sua criação em uma linguagem de alto nível, é manipulado por um grande conjunto de outros programas que traduzem seu código para linguagem de máquina e controlam sua execução no computador. Este conjunto de programas recebe a denominação genérica de ***software* de sistema** e é o objeto de estudo deste texto. No desenvolvimento de programas, o *software* de sistema é extensamente utilizado, com as várias etapas inter-relacionadas para a criação e execução de um programa (Figura 1.1). Tipicamente, esse relacionamento dá-se de forma transparente para o programador.

Programas são usualmente descritos em linguagens de alto nível. O **compilador** é o programa do sistema que traduz um programa descrito através de uma linguagem de alto nível específica para um programa equivalente em linguagem *assembly*. Esse processo de tradução é denominado compilação.

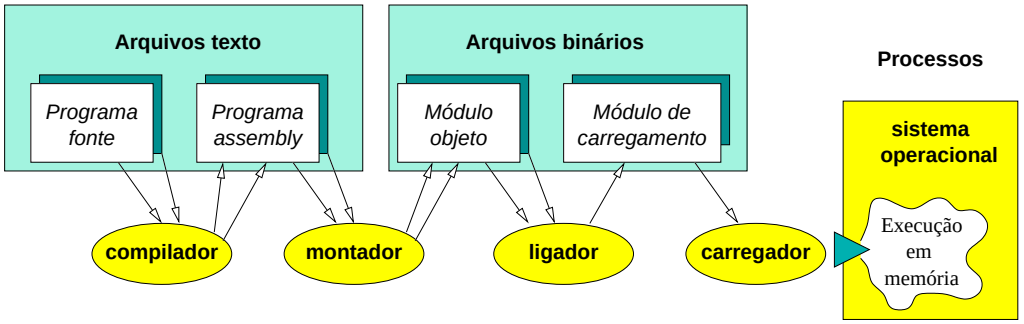
O **montador** (*assembler*) é o programa do sistema responsável por traduzir um programa *assembly* para o código de máquina. Esse processo de tradução de um programa-fonte *assembly* para um programa em código de máquina é denominado montagem; o resultado da montagem é um **módulo objeto** contendo pelo menos o código binário que será posteriormente executado.

Programas complexos raramente são descritos através de um único arquivo-fonte, mas sim organizados em módulos objetos interrelacionados. Tais módulos podem agregar funcionalidades da aplicação sendo desenvolvida ou recursos comuns do sistema que devem ser integrados à aplicação. O programa do sistema **ligador** é o responsável por interligar os diversos módulos de um programa para gerar o programa que será posteriormente carregado para a memória. Essa etapa de preparação de um programa para sua execução é denominada ligação.

Para que um programa possa ser executado, seu código de máquina deve estar presente na memória. O **carregador** é o programa do sistema responsável por transferir o código de máquina de um módulo objeto para a memória e encaminhar o início de sua execução. O processo de transferir o conteúdo de um módulo ob-

jeto para a memória principal é denominado carregamento. A execução de qualquer programa deve ser precedida por seu carregamento.

Figura 1.1 Etapas para criação e execução de programa.



A execução de cada programa se dá sob o controle do **sistema operacional**. A um programa em execução dá-se o nome de **processo**. Além das instruções do programa, um processo necessita de todo um conjunto de informações adicionais para o controle de sua execução. O estado corrente dessas informações associadas a cada programa em execução constitui o **estado do processo**. O sistema operacional é o responsável por gerenciar cada processo no computador, estabelecendo como será realizada sua execução. Ele também atua como um programa supervisor que estabelece uma camada de controle entre o *hardware* do computador e as aplicações de usuários. Uma de suas funções é estabelecer uma interface de *software* uniforme entre o computador, outros programas do sistema e programas de aplicação de usuários. Outra função fundamental de um sistema operacional é gerenciar os recursos de um computador de forma a promover sua eficiente utilização. Exemplos de sistemas operacionais são MS-DOS, Windows NT, OS/2, Linux e Solaris — estes dois implementações do sistema operacional Unix.

Os capítulos a seguir descrevem cada uma das etapas apresentadas como elipses na Figura 1.1. Onde há exemplos com programas existentes, a preferência foi dada sempre à utilização de *software* disponível gratuitamente. Assim, os exemplos foram criados em sistema operacional Linux e com o compilador Gnu C++ (`g++`). No entanto, afora os detalhes das linhas de comando, os conceitos demonstrados independem dessa opção e os mesmos exemplos podem ser reproduzidos em outros ambientes operacionais.

As características de processos e o funcionamento de sistemas operacionais, embora objeto de estudo do *software* de sistema, não são tratados neste texto.

1.1 Exercícios

1.1 Além de compiladores, outro tipo de programa tradutor para a execução de programas é o **interpretador**. Descreva como funciona um interpretador e apresente para ele um esquema similar àquele da Figura 1.1.

1.2 Utilize uma máquina com sistema operacional linux para descobrir que programas de sistema indicados na Figura 1.1 (compiladores, montadores, ligadores e carregadores) existem instalados. Para cada um deles, indique o nome do programa e a versão disponível.

1.3 Nos primeiros computadores, programas eram definidos através de conexões em painéis ou perfurações em fitas de papel. Descubra quem propôs o conceito de programa armazenado e quais os argumentos utilizados na época em favor dessa proposta. Esses argumentos todos são ainda válidos?