

TECNÓLOGO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PROGRAMAÇÃO DE COMPUTADORES I

Aula 01: Conceitos Iniciais

Observações:

- 1) *apresentar o plano de ensino da disciplina aos alunos.*
- 2) *efetuar o cadastro dos alunos no SETA.*

Organização e Arquitetura de Computadores / Lógica Digital (Primeira Parte)

O conteúdo da primeira parte da aula tem por objetivo apresentar uma visão geral sobre a Organização e Arquitetura de Computadores, e é baseado no livro “Princípios Básicos de Arquitetura e Organização de Computadores” – Linda Null e Julia Labur.

Introdução

Entender o que faz um sistema de computação funcionar é necessário para que possamos pensar em otimizá-lo, corrigi-lo, ou apenas, utilizá-lo adequadamente.

O que é Organização de Computadores?

Para compreendermos o funcionamento de um computador, faz-se necessária a familiarização com os diversos circuitos eletrônicos e componentes, que ajustados, criam um sistema computacional. Isso é feito através do estudo da organização de computadores. A organização de computadores trata das questões de controle (como o computador é controlado), tipos de memória; compreende todos os aspectos físicos do sistema de computação.

O que é Arquitetura de Computadores?

O estudo da arquitetura de computadores tem seu foco na estrutura e o comportamento do sistema de computação. A arquitetura de computadores inclui elementos como: processador (conjuntos de instruções, tipos de dados, quantidade e tipos de registradores) memória (modos de endereçamento, métodos de acesso à memória principal) e diversos mecanismos de E/S. A arquitetura de um sistema afeta diretamente a execução lógica de programas. O estudo da arquitetura de computadores nos ajuda a compreender como é projetado um computador.

Principais Componentes de um Computador

No nível mais básico, um computador é um dispositivo que consiste de três partes:

1. Um processador para interpretar e executar programas;
2. Uma memória para armazenar os dados e as instruções dos programas;

3. Um mecanismo para transferir dados vindos do mundo externo e enviar dados para o mundo externo (E/S);

O funcionamento de um computador pode ser análogo a muitos comportamentos humanos: imagine um estudante sentado em sua cadeira representa os três componentes de um computador – o cérebro do estudante é o processador, as anotações que estão sendo feitas representam a memória e a caneta usada para tomar notas é o mecanismo de E/S.

Terminologia de Medidas na Computação

Na computação para que grandezas possam ser representadas, com intuito de informar o quão grande ou quão rápido algo é, são usadas unidades de medida. Nos anos 60, alguém decidiu que, como as potências de 2 eram similares às potências de 10, os mesmos nomes de prefixos poderiam ser usados para ambas. Portanto, 2^{10} é próximo a 10^3 , e, assim, “kilo” é usado para referir-se a ambos. Por exemplo: se o seu modem antiquado transmite a 28,8 kb/s, então ele transmite 28.800 bits por segundo (ou $28,8 \times 10^3$).

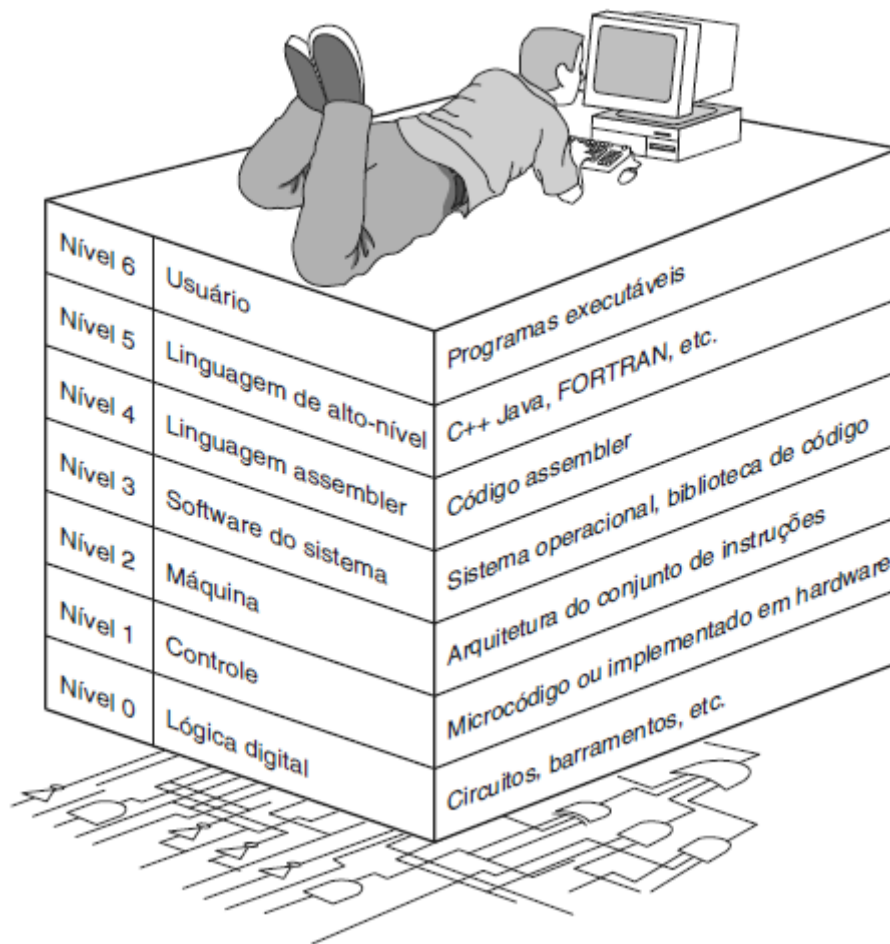
Importante notar que o uso da letra minúscula “k” para representar 10^3 e da letra minúscula “b” para se referir a bits. Uma letra maiúscula “K” é usada para se referir ao prefixo de potência de 2, ou 1024. Note que a letra maiúscula “B” se refere a byte. Se um disco contém 1MB, então ele contém 2^{20} bytes (ou um megabyte) de informação.

TABELA 1.1 Prefixos comuns associados com arquitetura e organização de computadores

Pref.	Símb.	Potência de 10	Potência de 2	Pref.	Símb.	Potência de 10	Potência de 2
Kilo	K	1 mil = 10^3	$2^{10} = 1024$	Mili	m	1 milésimo = 10^{-3}	2^{-10}
Mega	M	1 milhão = 10^6	2^{20}	Micro	μ	1 milionésimo = 10^{-6}	2^{-20}
Giga	G	1 bilhão = 10^9	2^{30}	Nano	n	1 billionésimo = 10^{-9}	2^{-30}
Tera	T	1 trilhão = 10^{12}	2^{40}	Pico	p	1 trillionésimo = 10^{-12}	2^{-40}
Peta	P	1 quadrilhão = 10^{15}	2^{50}	Femto	f	1 quadrillionésimo = 10^{-15}	2^{-50}
Exa	E	1 quintilhão = 10^{18}	2^{60}	Atto	a	1 quintillionésimo = 10^{-18}	2^{-60}
Zetta	Z	1 sextilhão = 10^{21}	2^{70}	Zepto	z	1 sextillionésimo = 10^{-21}	2^{-70}
Yotta	Y	1 setilhão = 10^{24}	2^{80}	Yocto	y	1 setillionésimo = 10^{-24}	2^{-80}

Hierarquia de Níveis

A organização de um sistema computacional pode ser abordada em camadas. Podemos imaginar a máquina como sendo construída em uma hierarquia de níveis, onde cada nível tem uma função específica.



Hierarquia de Níveis

- **Nível 6:** Nível do Usuário, composto pelas aplicações, é o nível com o qual todos estamos familiarizados. Neste nível executamos programas como processadores de textos, pacotes gráficos ou jogos, sendo os níveis inferiores praticamente invisíveis ao nível do usuário.
- **Nível 5:** Nível das Linguagens de Alto Nível, como C, C++, FORTRAN, essas linguagens devem ser traduzidas (usando um compilador ou um interpretador) para uma linguagem que a máquina possa entender. Linguagens compiladas são traduzidas para a linguagem simbólica do nível abaixo.
- **Nível 4:** Nível da Linguagem Simbólica, são a tradução das linguagens compiladas de alto nível para linguagem simbólica, que em níveis inferiores, será diretamente traduzida para linguagem de máquina. Esta tradução ocorre um a um, ou seja, uma instrução em linguagem simbólica é traduzida exatamente para uma instrução em linguagem de máquina (zeros e uns).
- **Nível 3:** Nível do Software de Sistema, trata das instruções de operação do sistema. Este nível é responsável por multiprogramação, proteção de memória, sincronização de processos e várias outras funções importantes. Muitas vezes as instruções traduzidas da linguagem simbólica para a linguagem de máquina são passadas por meio deste nível sem modificação.
- **Nível 2:** Arquitetura do Conjunto de Instruções (ISA), ou Nível de Máquina, consiste da linguagem de máquina reconhecida pela arquitetura particular do sistema de computação, podem ser executados diretamente pelos circuitos eletrônicos sem quaisquer interpretadores, tradutores ou compiladores.

- **Nível 1:** Nível de Controle, é onde a unidade de controle se assegura de que as instruções sejam decodificadas e executadas adequadamente e de que os dados sejam movidos para onde e quando eles devem ser. A unidade de controle interpreta as instruções de máquina passadas para ela, uma de cada vez, fazendo com que as ações requeridas sejam executadas.
- **Nível 0:** Nível de Lógica Digital, onde encontramos os componentes físicos do sistema de computação: as portas e os fios. Estes são os blocos básicos de construção, as implementações da lógica matemática, que são comuns a todos os sistemas de computação.



Prática: Lógica Digital – Nível 0 (Software Digital Works – Windows)
http://www.gileduardo.com.br/IFPR/II/downloads/ii_pratica01.pdf

Arquitetura Von-Neumann

Nas primeiras máquinas de computação eletrônicas, programar era sinônimo de conectar fios a pinos, sendo assim trocar a programação, ou seja, o programa que uma máquina devia executar era uma tarefa quase impossível, porém com a proposta de Arquitetura do matemático Von Neumann, foi possível criar computadores com programas armazenados, que ficaram conhecidos como sistemas de Von Neumann.

Arquitetura Von Neumann consiste em três sistemas de hardware:

- Uma unidade central de processamento (UCP), como unidade de controle, formada pela unidade lógica aritmética (ULA), registradores (pequenas memórias) e um contador de programa;
- Um sistema de memória principal, que armazena programas que controlam a execução do computador;
- Um sistema de Entrada e Saída.

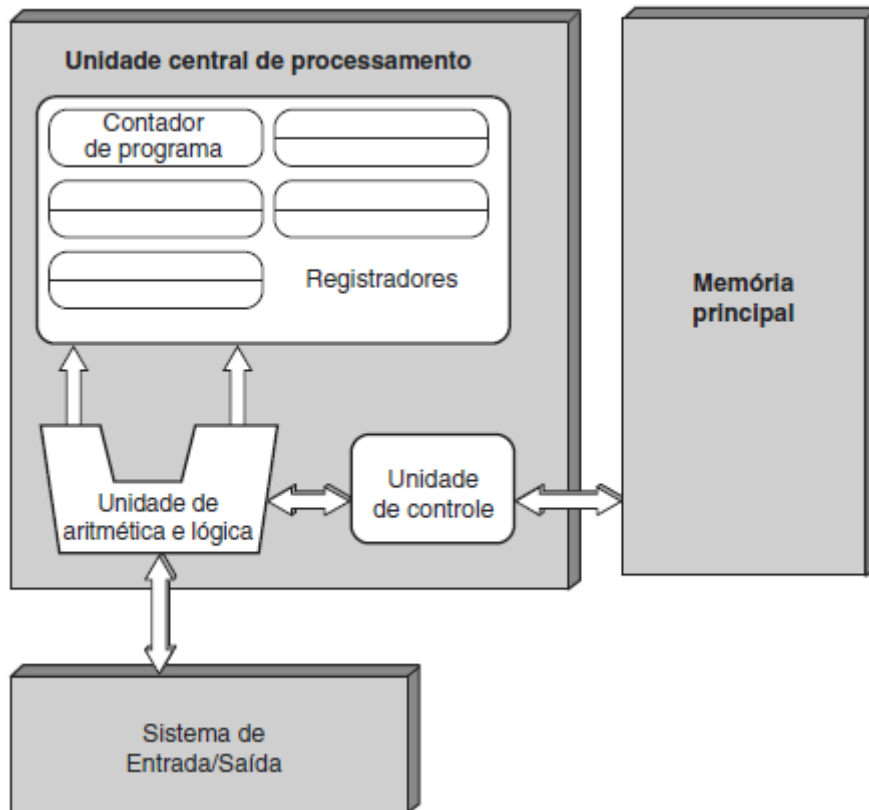
Seu funcionamento agrega os seguintes tópicos:

- Capacidade de executar processamento sequencial de instruções;
- Possui um único caminho, físico ou lógico, entre o sistema de memória principal e a unidade de controle da UCP, forçando a alternância entre ciclos de instrução e execução.

Um ciclo de instrução (ou ciclo de busca → decodifica → executa) consiste em:

1. A unidade de controle busca a próxima instrução do programa na memória usando o contador de programa para determinar onde a instrução está localizada.
2. A instrução é decodificada para uma linguagem que a ULA possa entender.

3. O operando de dados requerido para executar a instrução é carregado da memória e colocado em registradores dentro da UCP.
4. A ULA executa a instrução e coloca os resultados em registradores ou na memória.



Arquitetura Von Neumann



Prática: Máquina / Instruções – Nível 1 e 2 (Software SAVON – Windows)
http://www.gileduardo.com.br/IFPR/II/downloads/ii_pratica01.pdf

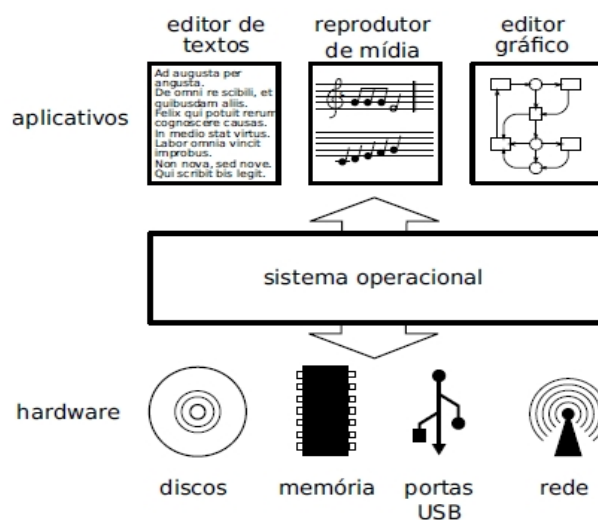
Sistema Operacional / Gerência de Tarefas (Segunda Parte)

O conteúdo da segunda parte da aula tem por objetivo apresentar uma visão geral sobre Sistemas Operacionais e Terminal Linux, e é baseado no livro do **Prof. Dr. Carlos Alberto Maziero**, disponível no link: <http://wiki.inf.ufpr.br/maziero/doku.php>

Introdução

Um sistema de computação é constituído basicamente por hardware e software, existindo uma grande distância entre os circuitos eletrônicos e dispositivos de hardware e os programas em software. Os circuitos são complexos, acessados através de interfaces de baixo nível (geralmente usando as portas de entrada/saída do processador) e muitas vezes suas características e seu comportamento dependem da tecnologia usada em sua construção. Por exemplo, a forma de acesso em baixo nível, a discos rígidos IDE, difere da forma de acesso a discos SCSI ou leitores de CD. Essa grande diversidade pode ser um grande problema para o desenvolvedor de aplicativos. Com isso torna-se desejável oferecer aos programas uma forma de acesso homogênea aos dispositivos físicos, que possibilite abstrair as diferenças tecnológicas entre eles.

O sistema operacional é uma camada de software que opera entre o hardware e os programas aplicativos voltados ao usuário final. O sistema operacional é uma estrutura de software ampla, muitas vezes complexa que incorpora aspectos de baixo nível (como *drivers* de dispositivos e gerência de memória física) e de alto nível (como programas utilitários e a própria interface gráfica). A figura a seguir ilustra a arquitetura geral de um sistema de computação típico. Nela, podemos observar elementos de hardware, o sistema operacional e alguns programas aplicativos.



Estrutura de um Sistema de Computação Típico

Abstração de Recursos

Acessar os recursos de hardware de um sistema de computação pode ser uma tarefa complexa, devido às características específicas de cada dispositivo físico e a complexidade de suas interfaces. Por exemplo, a seqüência a seguir apresenta os

principais passos envolvidos na abertura de um arquivo (operação open) em um leitor de disquete:

1. Verificar se os parâmetros informados estão corretos (nome do arquivo, identificador do leitor de disquete, buffer de leitura, etc);
2. Verificar se o leitor de disquetes está disponível;
3. Verificar se o leitor contém um disquete;
4. Ligar o motor do leitor e aguardar atingir a velocidade de rotação correta;
5. Posicionar a cabeça de leitura sobre a trilha onde está a tabela de diretório;
6. Ler a tabela de diretório e localizar o arquivo ou subdiretório desejado;
7. Mover a cabeça de leitura para a posição do bloco inicial do arquivo;
8. Ler o bloco inicial do arquivo e depositá-lo em um buffer de memória.

Assim, o sistema operacional deve definir interfaces abstratas para os recursos do hardware, visando atender os seguintes objetivos:

- Prover interfaces de acesso aos dispositivos mais simples que as interfaces de baixo nível, com intuito de simplificar a construção de programas aplicativos. Por exemplo: para ler dados de um disco rígido, uma aplicação usa um conceito chamado arquivo, que fornece uma visão abstrata do disco rígido.
- Tornar os aplicativos independentes do hardware. Ao definir uma interface abstrata de acesso a um dispositivo de hardware, o sistema operacional desacopla o hardware dos aplicativos e permite que ambos evoluam de forma mais autônoma. Por exemplo, o código de um editor de textos não deve ser dependente da tecnologia de discos rígidos utilizada no sistema.
- Definir interfaces de acesso homogêneas para dispositivos com tecnologias distintas. Através de suas abstrações, o sistema operacional permite aos aplicativos usar a mesma interface para dispositivos diversos. Por exemplo, um aplicativo acessa dados em disco através de arquivos e diretórios, sem precisar se preocupar com a estrutura real de armazenamento dos dados, que podem estar em um disquete, um disco IDE, uma máquina fotográfica digital conectada à porta USB, um CD ou mesmo um disco remoto, compartilhado através da rede.

Gerência de Recursos

Os aplicativos utilizam o hardware para executar suas funções: ler e armazenar dados, editar e imprimir documentos, navegar na Internet, tocar música, etc. Com diversas atividades em execução, e diversos possíveis acessos simultâneos ao hardware, conflitos podem surgir. Cabe ao sistema operacional definir políticas para gerenciar o uso dos recursos de hardware pelos aplicativos, e resolver eventuais disputas e conflitos. Exemplos:

- O uso desse processador deve ser distribuído entre os aplicativos presentes no sistema, de forma que cada um deles possa executar na velocidade adequada para cumprir suas funções sem prejudicar os outros.
- Ataques de negação de serviço (DoS – *Denial of Service*). Consistem em usar diversas técnicas para forçar um servidor de rede a dedicar seus recursos a

atender um determinado usuário, em detrimento dos demais. Por exemplo, ao abrir milhares conexões simultâneas em um servidor de e-mail.



Prática: Sistema Operacional Linux – Comandos Básicos / Repositório (Linux)
http://www.gileduardo.com.br/ifpr/pci/downloads/pc_pratica01_02.pdf

Gerência de Tarefas

Uma tarefa é definida como sendo a execução de um fluxo sequencial de instruções, construído para atender uma finalidade específica: realizar um cálculo complexo, a edição de um gráfico, a formatação de um disco, etc. Assim, a execução de uma sequência de instruções em linguagem de máquina, normalmente gerada pela compilação de um programa escrito em uma linguagem qualquer, é denominada “tarefa” ou “atividade” (do inglês *task*).

É importante ressaltar as diferenças entre os conceitos de tarefa e de programa:

- Um programa é um conjunto de uma ou mais sequências de instruções escritas para resolver um problema específico, constituindo assim uma aplicação ou utilitário. O programa representa um conceito estático, sem um estado interno definido (que represente uma situação específica da execução) e sem interações com outras entidades (o usuário ou outros programas). Por exemplo, os arquivos `C:\Windows\notepad.exe` e `/usr/bin/nano` são programas de edição de texto.
- Uma tarefa é a execução, pelo processador, das sequências de instruções definidas em um programa para realizar seu objetivo. Trata-se de um conceito dinâmico, que possui um estado interno bem definido a cada instante (os valores das variáveis internas e a posição atual da execução) e interage com outras entidades: o usuário, os periféricos e/ou outras tarefas. Tarefas podem ser implementadas de várias formas, como processos ou threads, como veremos posteriormente.

Sistemas Operacionais e Tarefas

Em um computador, o processador tem executar todas as tarefas submetidas pelos usuários. Essas tarefas geralmente têm comportamento, duração e importância distintas. Cabe ao sistema operacional organizar as tarefas para executá-las e decidir em que ordem fazê-lo.

Sistemas mono-tarefa

Os primeiros sistemas de computação, nos anos 40, executavam apenas uma tarefa de cada vez. Nestes sistemas, cada programa binário era carregado do disco para a memória e executado até sua conclusão. Os dados de entrada da tarefa eram carregados na memória juntamente com a mesma e os resultados obtidos no

processamento eram descarregados de volta no disco após a conclusão da tarefa. Todas as operações de transferência de código e dados entre o disco e a memória eram coordenadas por um operador humano.

Nesse método de processamento de tarefas é possível delinear um diagrama de estados para cada tarefa executada pelo sistema, representado na figura a seguir.

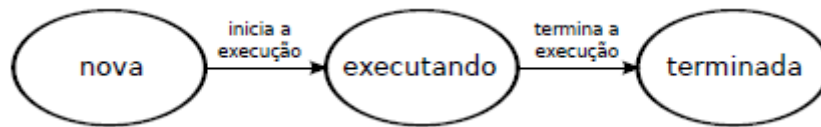


Diagrama de estados de uma tarefa em um sistema mono-tarefa

Com a evolução do hardware, as tarefas de carga e descarga de código entre memória e disco, coordenadas por um operador humano, passaram a se tornar críticas: mais tempo era perdido nesses procedimentos manuais que no processamento da tarefa em si. Para resolver esse problema foi construído um programa monitor, que era carregado na memória no início da operação do sistema com a função de coordenar a execução dos demais programas. O programa monitor executava continuamente os seguintes passos sobre uma fila de programas para execução, armazenados no disco:

1. Carregar um programa do disco para a memória;
2. Carregar os dados de entrada do disco para a memória;
3. Transferir a execução para o programa recém-carregado;
4. Aguardar o término da execução do programa;
5. Escrever os resultados gerados pelo programa no disco.

Percebe-se claramente que a função do monitor é gerenciar uma fila de programas a executar, mantida no disco. Na medida em que os programas são executados pelo processador, novos programas podem ser inseridos na fila pelo operador do sistema. Além de coordenar a execução dos demais programas, o monitor também colocava à disposição destes uma biblioteca de funções para simplificar o acesso aos dispositivos de hardware (teclado, leitora de cartões, disco, etc). Assim, o monitor de sistema constitui o precursor dos sistemas operacionais.

Sistemas multi-tarefa

O uso do programa monitor agilizou o uso do processador, mas outros problemas persistiam. Como a velocidade de processamento era muito maior que a velocidade de comunicação com os dispositivos de entrada e saída¹, o processador ficava ocioso durante os períodos de transferência de informação entre disco e memória. Se a operação de entrada/saída envolvia fitas magnéticas, o processador podia ficar vários minutos parado, esperando. O custo dos computadores era elevado demais (e sua capacidade de processamento muito baixa) para permitir deixá-los ociosos por tanto tempo.

A solução encontrada para resolver esse problema foi permitir ao processador suspender a execução da tarefa que espera dados externos e passar a executar outra tarefa. Mais tarde, quando os dados de que necessita estiverem disponíveis, a tarefa

suspensa pode ser retomada no ponto onde parou. Para tal, é necessário ter mais memória (para poder carregar mais de um programa ao mesmo tempo) e definir procedimentos para suspender uma tarefa e retomá-la mais tarde. O ato de retirar o processador de uma tarefa é denominado preempção.

Sistemas que implementam esse conceito são chamados sistemas preemptivos. A adoção da preempção levou a sistemas mais produtivos (e complexos), nos quais várias tarefas podiam estar em andamento simultaneamente: uma estava ativa e as demais suspensas, esperando dados externos ou outras condições. Sistemas que suportavam essa funcionalidade foram denominados monitores multi-tarefas. O diagrama de estados da figura a seguir ilustra o comportamento de uma tarefa em um sistema desse tipo.

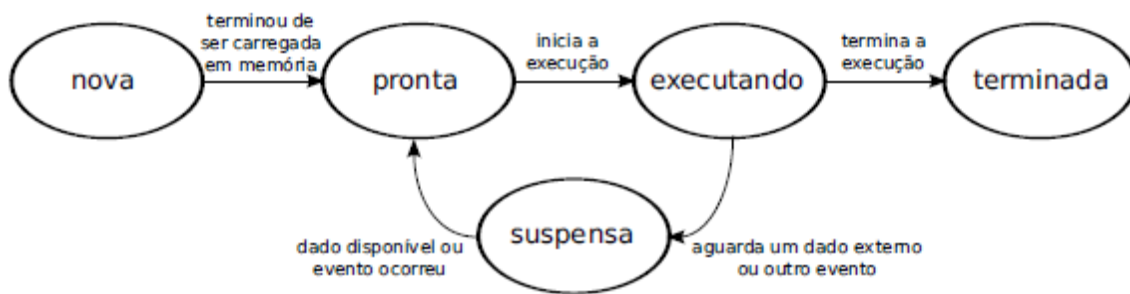


Diagrama de estados de uma tarefa em um sistema multi-tarefa



Prática: Gerenciamento de Tarefas – Comando “Kill” (Linux)

http://www.gileduardo.com.br/ifpr/pci/downloads/pc_pratica01_02.pdf

Prática: Cálculo de π com Threads (Linux)

http://www.gileduardo.com.br/ifpr/pci/downloads/pc_pratica01_02.pdf