Fundação Presidente Antônio Carlos de
Conselheiro Lafaiete
Engenharia de Computação

LINUX SERVER

Uma análise sobre o sistema

Paulo Henrique de Oliveira Rodrigues

Conselheiro Lafaiete, 22 de setembro de 2020

Paulo Henrique de Oliveira Rodrigues

LINUX SERVER
Uma análise sobre o sistema

Trabalho apresentado na Fundação Presidente Antônio Carlos (FUPAC) - Conselheiro Lafaiete , como um dos pré-requisitos para a aprovação na disciplina de Sistemas Operacionais no curso de Bacharel em Engenharia de Computação.

Conselheiro Lafaiete
22 de setembro de 2020

Agradecimentos

Agradeço a Deus por me iluminar nos momentos difíceis, dando força na longa caminhada.

Em especial minha esposa, por me apoiar incondicionalmente e por sempre confiar em meu potencial me incentivando a fazer sempre o melhor e a meu filho pelas risadas calorosas que me renovam sempre o ânimo e me lembram dos meus objetivos.

Ao meu orientador Alex Vitorino por sua paciência e sempre propondo novos desafios que são de grande contribuição em relação ao meu aprendizado e com ensinamentos importantes para consolidação deste trabalho.

Enfim em todos que acreditaram no meu sonho.

"A mão queimada ensina melhor.
Depois disso o conselho sobre o fogo
chega ao coração."– Gandalf - O Cinzento
(J.R.R. Tolkien)

Resumo

O objetivo deste presente trabalho é a consolidação dos conhecimentos adquiridos durante a realização da disciplina de Sistemas Operacionais, dando enfoque aos sistemas baseados em *Linux* utilizados em servidores, salientando suas utilizações e o seu mercado de atuação. O *Linux* é um sistema operacional que vive em um crescimento contínuo e é amplamente usado ele está tanto em sensores a supercomputadores, e podemos vê-lo sendo usados em espaçonaves, automóveis, *smartphones*, relógios e muitos outros dispositivos em nossa vida cotidiana.

Em especial o sistema *Linux* é um sistema de código aberto o que significa que é possível executá-lo para qualquer propósito, estudar seu funcionamento e modificá-lo se assim desejar, ou realizar cópias para terceiros dando total liberdade para sua comunidade.

Ele também opera a maior parte da Internet, todos os 500 maiores supercomputadores do mundo e as bolsas de valores do mundo. Estes funcionam em uma variação do *Linux* preparada para um grande fluxo de tratamento de dados, podendo rodar vários serviços simultaneamente, esta versão é a *Linux* para servidores ou *Linux Server*.

Palavras-chaves: Linux, Servidores, Sistema Operacional.

Lista de ilustrações

Figura 1 – Onde o sistema operacional se encaixa. [1] 1
Figura 2 – Um *pipeline* com três estágios. [1] 5
Figura 3 – Uma *CPU* superescalar. [1] 6
Figura 4 – Uma hierarquia de memória típica. Os números são apenas aproximações. [1] 6
Figura 5 – As memorias cache em um processador moderno. 7
Figura 6 – Do processador a RAM. 7

Lista de abreviaturas e siglas

ARM	Acorn RISC Machine
CPU	Central Process Unit
EEPROM	Electrically Erasable PROM
E/S	Entrada e saída
FreeBSD	Free OS descended from the Berkeley Software Distribution
GNU	GNU's Not Unix
GNU GPL	GNU General Public License
GUI	Graphical User Interface
MIT	Massachusetts Institute of Technology
MULTICS	MULTiplexed Information and Computing Service
OS	Operating System
OS X	Operating Systems number 10
PC	Personal Computer
PROM	Programmable Read Only Memory
RAM	Random Access Memory
ROM	Read Only Memory
SO	Sistema Operacional
X86	Processadores baseados no Intel 8086

Sumário

1 INTRODUÇÃO 1

2 SISTEMA OPERACIONA 4
2.1 Revisão sobre hardware de computadores 5
2.1.1 Processadores 5
2.1.2 Memória 6
2.1.3 Discos 8
2.1.4 Dispositivos de E/S 8
2.1.5 Barramentos 8
2.2 Estrutura de sistemas operacionais 9
2.3 O zoológico dos sistemas operacionais 10

3 PROCESSOS E THREADS 11
3.1 Processos 11
3.2 Threads 11
3.3 Comunicação entre processos 11
3.4 Escalonamento 11

Referências 12

1 Introdução

Um computador moderno consiste em um emaranhado de peças que contém um ou mais processadores, alguma memória principal, alguma memória secundaria, interfaces de rede diversos periféricos como impressoras, teclado, mouse, monitor e vários outros dispositivos de entrada e saída. Podemos dizer que este é um sistema complexo, para realizar a desafiadora maratona que é compreender como cada parte funciona e gerenciar com maestria esses componentes é um grande desafio [1].
Para isso os computadores modernos são equipados com um SO esse dispositivo de *software* tem a função de fornecer uma plataforma simples e limpa para o usuário de forma a ajuda-lo nas entradas e saídas de dados. Em uma visão simplista podemos ver na figura 1 onde o SO se encontra em relação entre *hardware* e o usuário [1].

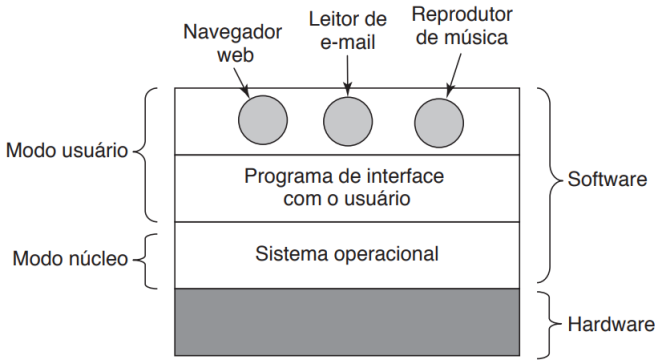

Fig. 1. Onde o sistema operacional se encaixa. [1]

Inicialmente a ideia que temos de um SO é a visão que temos dos ditos sistemas operativos que temos conhecimento que podem ser *Windows*, *Linux*, *FreeBSD*, ou *OS X* mas normalmente a forma de interagir com diretamente com o sistema é através de terminais comumente conhecidos como shell (interpretadores de comando) isto quando baseado em texto ou *GUI (Graphical User Interface)* quando em modo gráfico [1].
Um sistema operacional é projetado para ocultar as particularidades de *hardware* (ditas "de baixo nível") e assim criar uma máquina abstrata que fornece às aplicações serviços compreensíveis ao usuário (ditas "de alto nível") [2].
Assim o sistema trabalha em dois estados o modo núcleo e o modo usuário. Sendo que no modo núcleo (também chamado modo supervisor ou *Kernel mode*). O sistema tem acesso completo aos recursos seja de ao *hardware* ou *software* e pode executar qualquer instrução que a máquina for capaz de executar [1], [3].
Quando o sistema está em modo *kernel* é considerado que as execuções são de uma fonte confiável e, portanto, pode executar quaisquer instruções e fazer referência a quaisquer endereços de memória (ou seja, locais na memória). O *kernel* tem total controle sobre o

sistema e trata todos os outros *softwares* como programas não confiáveis, assim todas as operações em modo usuário que necessitem alterar o sistema solicitam ao uso do *kernel* por meio de uma chamada de sistema para executar instruções privilegiadas, como criação de processos ou operações de entrada / saída [1], [3].

Neste trabalho iremos trabalhar com o sistemas baseados em *Linux* para servidores mas é importante entender um pouco da evolução desse sistema.

Em meados da década de 60 uma iniciativa conjunta do *MIT*, da *Bell Labs* e da *General Electric* decidiram embarcar no desenvolvimento de um "computador utilitário", isto é, uma máquina que daria suporte a algumas centenas de usuários simultâneos em pouco tempo nasce o projeto MULTICS (Serviço de Computação e Informação Multiplexada) [1]. O MULTICS foi projetado para ser um sucesso com suporte para centenas de usuários em uma máquina apenas um pouco mais poderosa do que um PC baseado no 386 da *Intel*. Mas transformá lo em um produto final de fácil comercialização não foi amarga realidade [1].
A *Bell Labs* abandonou o projeto, e a *General Electric* abandonou completamente o negócio dos computadores. Entretanto, o *MIT* persistiu e finalmente colocou o MULTICS para funcionar.E foi instalado por mais ou menos 80 empresas e universidades importantes mundo afora [1].
Um dos cientistas da *Bell Labs* que havia trabalhado no projeto MULTICS, Ken Thompson, decidiu escrever uma versão despojada e para um usuário do MULTICS. Esse trabalho mais tarde desenvolveu-se no sistema operacional UNIX, que se tornou popular no mundo acadêmico, em agências do governo e em muitas empresas [1].
Em 1987, Andrew Tanenbaum lançou um pequeno clone do UNIX, chamado MINIX, para fins educacionais. Em termos funcionais, o MINIX é muito similar ao UNIX [1].
Em 1991 Linus Torvalds começou um projeto inicialmente um emulador de terminal que era utilizado para acessar os servidores em UNIX da universidade Helsinki. Ele escreveu o código para especificamente para o *hardware* que utilizava um computador com um processador 80386 ele realizou o desenvolvimento no minix usando o *GNU C compiler* [4], [5], [6].
O *Linux* também é distribuído sob uma licença de código aberto. O código aberto segue estes locatários principais:

- A liberdade de executar o programa, para qualquer propósito.
- A liberdade de estudar como o programa funciona e alterá-lo para que ele faça o que você deseja.
- A liberdade de redistribuir cópias para que você possa ajudar seu vizinho.
- A liberdade de distribuir cópias de suas versões modificadas para terceiros.

Esses pontos são cruciais para entender a ideia por trás do *Linux*. O *Linux* se transformou em um sistema de fácil acesso. Com a grande liberdade de se poder modificar o sistema ele proporcionou a criação de diversas distribuições uma vez que qualquer usuário pode criar uma que atenda a suas necessidades [7].
Nos próximos sessões iremos discutir sobre o sistema e aprofundar no *Linux* para assim entender, suas funcionalidade, modo de funcionamento e quais suas principais atuações.

2 Sistema Operacional

Mas afinal o que é um sistema operacional?

É difícil entender o que é um sistema operacional pois talvez a única coisa que podemos afirmar é que é um *software* que trabalhe em modo núcleo (e por vezes até isso não é uma completa verdade), para tal precisamos entender que o SO tem duas funções bases que é fornecer uma abstração do *hardware* para programadores de aplicativos e usuários em geral, e o gerenciamento dos recursos de *hardware*.
Em geral o conjunto de instruções, estrutura de barramento, E/S e a organização de memória é complexo e varia dependendo da arquitetura das peças utilizadas no computador. Por esse motivo o SO fornece uma camada de abstração para os aplicativos que se utilizam dos *hardwares* do computador possam criar, escrever e ler arquivos, sem ter de lidar com os detalhes complexos de como o *hardware* realmente funciona.
Como dito anteriormente um computador moderno consiste em um emaranhado de peça e a função do sistema operacional é fornecer uma alocação ordenada e controlada para todas elas além que o SO moderno permite que múltiplas aplicações estejam em memória e sejam executados “ao mesmo tempo”.
Precisamos entender que o sistema não faz execução de todos os recursos ao mesmo tempo isso seria insano de se imaginar mais ele utiliza de filas de processos e de um nivelamento de urgências para definir o que será processado e quando será processado, ele ainda define as utilizações em nível de memória para definir prioridades e utilizando delas para alterar o tempo de acesso ao processador, desta forma mesmo com vários processos “abertos” a execução se dá em filas multiplexadas.
Ainda devemos entender que se o sistema possuir vários usuários a necessidade de gerenciar e proteger a memória, dispositivos de E/S e outros recursos é ainda maior, pois cada usuário precisa ter acesso aos recursos de *hardwares* e compartilhar de informações salvas como (arquivos, bancos de dados etc.) e as ações de um usuário pode influenciar ao uso do outro.
Podemos dizer então que a função primordial do SO é manter controle, isso seja sobre como conceder acesso a recursos requisitados, sobre quais programas estão usando qual recurso, sobre como conceder recursos requisitados, contabilizar o seu uso, assim como mediar requisições conflitantes de diferentes programas e usuários.
O termo *hardware* foi muito utilizado mais qual seriam os *hardwares* de um computador ou uma estrutura basica dos mesmos e como o sistema operacional se utiliza deles para em sua execução.

2.1 Revisão sobre hardware de computadores

2.1.1 Processadores

A *CPU* (*Central Process Unit* - no portugues Unidade Central de Processamento) muitas vezes é considerado o cérebro do computador ela é responsável por receber as instruções da memória e processá las, decodificando-as e executando todos os processos em fila de execução num ciclo que é repetido até que o programa termine [1].
Assim os programas são executados. Cada *CPU* é presa a sua arquitetura assim não conseguindo decodificar instruções de arquiteturas diferentes como *ARM* para X86 ou vice e versa. A velocidade de cálculo das *CPU's* é muito maior que o tempo para executar uma instrução registradores internos são utilizados para armazenamento de variáveis e resultados temporários [1].
O SO deve conhecer absolutamente todos os processos destinados ao processador e também todos os registradores gerados pois quando a *CPU* realiza uma multiplexação ela interrompe o programa em execução para recomeçar outro. Assim faz-se necessário que o SO tem que salvar todos os registradores de maneira que eles possam ser restaurados quando o programa for executado mais tarde. Muitas *CPU's* modernas têm recursos para executar mais de uma instrução ao mesmo tempo [1].
Por exemplo, uma *CPU* pode ter unidades de busca, decodificação e execução separadas, assim enquanto ela está executando a instrução não, poderia também estar decodificando a instrução n + 1 e buscando a instrução n + 2. Uma organização com essas características é chamada de *pipeline* como na figura 2 abaixo [1].

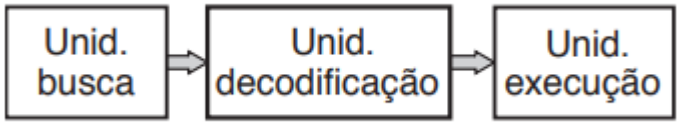

Fig. 2. Um *pipeline* com três estágios. [1]

Ainda mais avançada que um projeto de pipeline é uma *CPU* superescalar, mostrada na Figura 3. Nesse projeto, unidades múltiplas de execução estão presentes. Uma unidade para aritmética de números inteiros, por exemplo, uma unidade para aritmética de ponto flutuante e uma para operações booleanas. Duas ou mais instruções são buscadas ao mesmo tempo, decodificadas e jogadas em um *buffer* de instrução até que possam ser executadas. Tão logo uma unidade de execução fica disponível, ela procura no buffer de instrução para ver se há uma instrução que ela pode executar e, se assim for, ela remove a instrução do *buffer* e a executa [1].

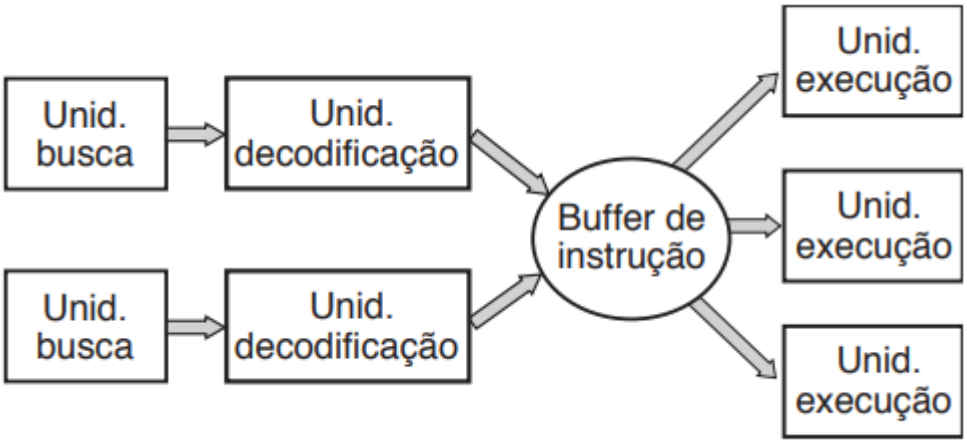

Fig. 3. Uma *CPU* superescalar. [1]

Um *thread* é um tipo de processo leve, mais para o SO cada *thread* é como uma *CPU* separada, considere um sistema com duas *CPU's* efetivas, cada uma com dois threads. O sistema operacional verá isso como quatro *CPU's* chamamos isso de *multithreading* [1].

2.1.2 Memória

A memória é a capacidade de adquirir, armazenar e recuperar informação. essa é uma definição que serve tanto para o meio biológico como para o meio artificial, podemos dizer que a memória é tão importante quanto o processador e que é um dos principais componentes do computador. Com a função de armazenar a informação antes dela seguir para o processador a memória segue uma topologia que define sua utilização pelo SO, as camadas superiores têm uma velocidade mais alta, capacidade menor e um custo maior, o que se altera nas camadas inferiores que têm velocidade mais baixa, capacidade maior e um menor custo.

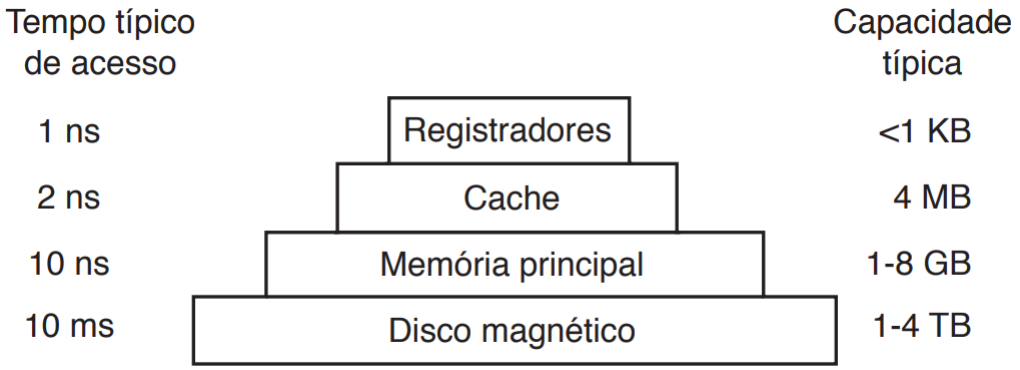

Fig. 4. Uma hierarquia de memória típica. Os números são apenas aproximações. [1]

Na camada superior temos os registradores que são feitos do mesmo material do processador e possuem velocidade similar a deles, posteriormente temos o Cache que é principalmente controlada pelo hardware e pode se separar em até três níveis cada nível mais lento e maior que o anterior. O cache é erroneamente confundido com o buffer do processador mas basicamente ele tem por função servir como um espaço para informação de rápido acesso por exemplo o cache L1 é dividido em memória de instrução e memória para dados. Com isso, o processador vai direto à memória de instrução, se estiver buscando uma instrução, ou vai direto à memória de dados, se estiver buscando um dado o cache L2 possui uma capacidade de armazenamento maior e é um pouco mais lento que o L1 ele serve para armazenar as informações antes delas irem para o cache L1 e assim sucessivamente o cache L3 diferente dos anteriores normalmente se apresenta fora do núcleo do processador e é compartilhado pelos núcleos como podemos ver na figura 5 .

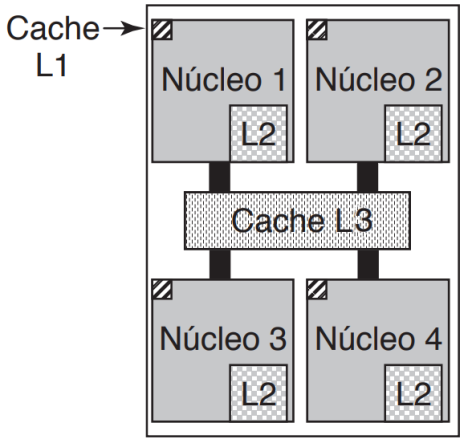

Fig. 5. As memorias cache em um processador moderno.

Logo em seguida vem a memória RAM (Random Access Memory — em portugues memória de acesso aleatório)Todas as requisições da CPU que não podem ser atendidas pela cache vão para a memória principal. Essa memória é responsável por realizar a leitura dos conteúdos quando requeridos, ou seja, de forma não-sequencial e guardar a informação de forma rápida para que seja lida cache assim como manter informações da cache.

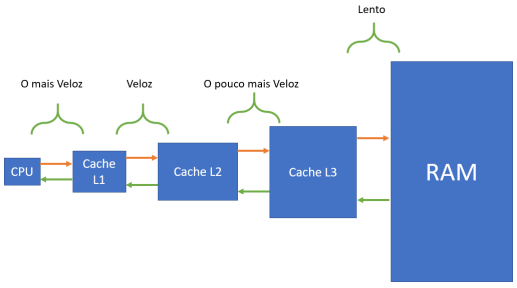

Fig. 6. Do processador a RAM.

A ROM (Read Only Memory — em portugues memória somente de leitura) é programada na fábrica e não pode ser modificada depois. Ela é rápida e barata. Em alguns computadores, o carregador (bootstrap loader) usado para inicializar o computador está contido na ROM.
A EEPROM (Electrically Erasable PROM — em portugues ROM eletricamente apagável) e a memória flash também são não voláteis, mas, diferentemente da ROM, podem ser apagadas e reescritas. No entanto, escrevê-las leva muito mais tempo do que escrever em RAM, então elas são usadas da mesma maneira que a ROM, apenas com a característica adicional de que é possível agora corrigir erros nos programas que elas armazenam mediante sua regravação.

2.1.3 Discos

2.1.4 Dispositivos de E/S

2.1.5 Barramentos

asd

2.2 Estrutura de sistemas operacionais

asd

2.3 O zoológico dos sistemas operacionais

asd

3 Processos e threads

3.1 Processos

3.2 Threads

3.3 Comunicação entre processos

3.4 Escalonamento

Referências

[1] A. S. Tanenbaum and H. Bos, *Sistemas operacionais modernos*. 4ª edição, 2016.

[2] D. Comer, *Operating system design : the xinu approach, linksys version*. Boca Raton: CRC Press-Taylor and Francis, 2012.

[3] T. L. I. Project, "Kernel mode definition," 2007.

[4] L. Torvalds, ""what would you like to see most in minix?"," 1991.

[5] M. Magazine., "The choice of a gnu generation an interview with linus torvalds," 1993.

[6] L. Torvalds and D. Diamond, *Just for Fun: The Story of an Accidental Revolutionary*. HarperCollins, 2002.

[7] T. L. Foundation, "What is linux?," 2020.