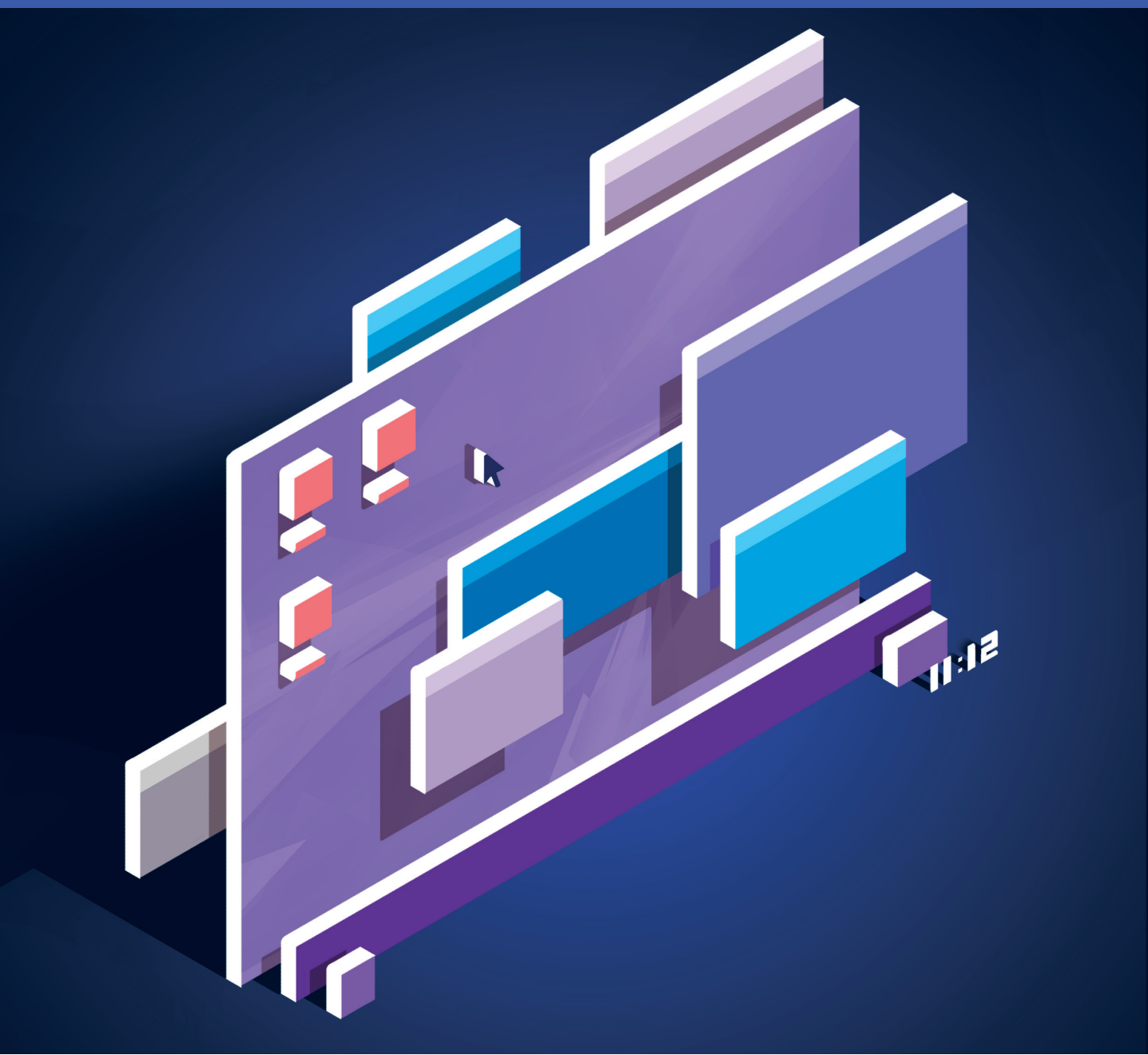
SISTEMAS OPERACIONAIS

AUTORES
Guilherme Bernardino da Cunha
Evandro Preuss
Ricardo Tombesi Macedo



LICENCIATURA EM COMPUTAÇÃO

SISTEMAS OPERACIONAIS

AUTORES
Guilherme Bernardino da Cunha
Evandro Preuss
Ricardo Tombesi Macedo

UNIVERSIDADE FEDERAL DE SANTA MARIA

Santa Maria | RS
2017

©Núcleo de Tecnologia Educacional – NTE.
Este caderno foi elaborado pelo Núcleo de Tecnologia Educacional da Universidade Federal de Santa Maria para os cursos da UAB.

PRESIDENTE DA REPÚBLICA FEDERATIVA DO BRASIL
Michel Temer

MINISTRO DA EDUCAÇÃO
Mendonça Filho

PRESIDENTE DA CAPES
Abilio A. Baeta Neves

UNIVERSIDADE FEDERAL DE SANTA MARIA

REITOR
Paulo Afonso Burmann

VICE-REITOR
Paulo Bayard Dias Gonçalves

PRÓ-REITOR DE PLANEJAMENTO
Frank Leonardo Casado

PRÓ-REITOR DE GRADUAÇÃO
Martha Bohrer Adaime

COORDENADOR DE PLANEJAMENTO ACADÊMICO E DE EDUCAÇÃO A DISTÂNCIA
Jerônimo Siqueira Tybusch

COORDENADOR DO CURSO DE LICENCIATURA EM COMPUTAÇÃO
Sidnei Renato Silveira

NÚCLEO DE TECNOLOGIA EDUCACIONAL

DIRETOR DO NTE
Paulo Roberto Colusso

COORDENADOR UAB
Reisoli Bender Filho

COORDENADOR ADJUNTO UAB
Paulo Roberto Colusso

NÚCLEO DE TECNOLOGIA EDUCACIONAL

DIRETOR DO NTE
Paulo Roberto Colusso

ELABORAÇÃO DO CONTEÚDO
Ricardo Tombesi Macedo, Mateus Pelloso,
Evandro Preuss, Fábio Parreira

REVISÃO LINGUÍSTICA
Camila Marchesan Cargnelutti
Maurício Sena

APOIO PEDAGÓGICO
Carmem Eloísa Berlote Brenner
Caroline da Silva dos Santos
Keila de Oliveira Urrutia

EQUIPE DE DESIGN
Carlo Pozzobon de Moraes – Diagramação
Juliana Segalla – Diagramação
Matheus Tanuri Pascotini – Capa e Ilustração

PROJETO GRÁFICO
Ana Letícia Oliveira do Amaral

CC BY SA

C972s Cunha, Guilherme Bernardino da
Sistemas operacionais [recurso eletrônico] / Guilherme Bernardino da Cunha, Evandro Preuss, Ricardo Tombesi Macedo. – 1. ed. – Santa Maria, RS : UFSM, NTE, 2017.
1 e-book

Este caderno foi elaborado pelo Núcleo de Tecnologia Educacional da Universidade Federal de Santa Maria para os cursos da UAB
Acima do título: Licenciatura em computação
ISBN 978-85-8341-218-2

1. Computação 2. Sistemas operacionais (Computadores) I. Preuss, Evandro II. Macedo, Ricardo Tombesi III. Universidade Federal de Santa Maria. Núcleo de Tecnologia Educacional IV. Universidade Aberta do Brasil V. Título.

CDU 004.451

Ficha catalográfica elaborada por Alenir Goularte - CRB-10/990
Biblioteca Central da UFSM

Ministério da Educação

CAPES UNIVERSIDADE ABERTA DO BRASIL Universidade Federal de Santa Maria 1960 PROGRAD UFSM NÚCLEO DE TECNOLOGIA EDUCACIONAL

APRESENTAÇÃO

A disciplina de Sistemas Operacionais objetiva capacitar os alunos quanto aos conhecimentos necessários para utilizar um computador, de modo a entender o funcionamento e a interface entre o hardware e o software proporcionada pelo Sistema Operacional (SO).

Neste material você tomará conhecimento a respeito do modo de funcionamento dos computadores e do SO e a forma como este controla o hardware no mesmo instante que esconde a complexidade do mesmo, oferecendo para o usuário uma visão completamente acessível dos recursos computacionais.

Para alcançar o objetivo proposto, este material foi organizado em um conjunto de unidades, cada uma delas sendo cuidadosamente elaborada pelos autores para proporcionar um aprendizado efetivo e agradável, buscando aliar os conceitos teóricos com as operações diárias com o uso dos SOs Windows e Linux.

A primeira unidade introduz a disciplina, apresentando o princípio de funcionamento dos computadores e a sua evolução, desde a sua criação até os dias de hoje, além do surgimento dos Sistemas Operacionais. A segunda unidade apresenta uma visão genérica de como o SO funciona. A terceira unidade descreve em detalhes o gerenciador de processos. A unidade quatro detalha como a memória é gerenciada pelos SOs. A quinta unidade aborda o gerenciamento dos dispositivos de entrada e saída e a unidade seis apresenta os sistemas de arquivos.

A equipe de professores autores deste material acredita em você e deseja sucesso na sua formação profissional!

ENTENDA OS ÍCONES

ATENÇÃO: faz uma chamada ao leitor sobre um assunto, abordado no texto, que merece destaque pela relevância.

INTERATIVIDADE: aponta recursos disponíveis na internet (sites, vídeos, jogos, artigos, objetos de aprendizagem) que auxiliam na compreensão do conteúdo da disciplina.

SAIBA MAIS: traz sugestões de conhecimentos relacionados ao tema abordado, facilitando a aprendizagem do aluno.

TERMO DO GLOSSÁRIO: indica definição mais detalhada de um termo, palavra ou expressão utilizada no texto.

SUMÁRIO

APRESENTAÇÃO ·5

▷ **UNIDADE 1 – FUNDAMENTOS ·10**

Introdução ·11
1.1 Fundamentos de sistemas operacionais ·15
1.2 Principais sistemas operacionais ·27
1.3 Interface dos sistemas operacionais ·30

▷ **UNIDADE 2 – ESTRUTURA DO SISTEMA OPERACIONAL**

Introdução ·44
2.1 Como funciona um sistema operacional ·45
2.2 Funções do núcleo do sistema operacional ·48
2.3 Serviços do sistema operacional ·50
2.4 Recursos e ambiente operacional ·52
2.5 Tipos de sistemas operacionais ·54

▷ **UNIDADE 3 – GERÊNCIA DE PROCESSOS**

Introdução ·58
3.1 Gerência de processos ·60
3.2 Estrutura de um processo ·62
3.3 Estados dos processos ·67
3.4 Tipos de processos ·69
3.5 Escalonamento ·72
3.6 Esudos de caso ·74

▷ **UNIDADE 4 – GERÊNCIA DE MEMÓRIA**

Introdução ·93
4.1 Gerência de memória ·94
4.2 Memória Virtual ·98

▷ **UNIDADE 5 – GERÊNCIA DE ENTRADA E SAÍDA** ·106

Introdução ·107
5.1 Gerência de dispositivos ·108
5.2 Discos magnéticos ·117
5.3 Gerência de entrada e saída no Windows ·119
5.4 Gerência de entrada e saída no Linux ·122

▷ **UNIDADE 6 – SISTEMA DE ARQUIVOS** ·127

Introdução ·128
6.1 Sistema de arquivos ·130

▷ **CONSIDERAÇÕES FINAIS** ·141

▷ **REFERÊNCIAS** ·142

▷ **ATIVIDADES** ·143

▷ **APRESENTAÇÃO DOS PROFESSORES AUTORES** ·147

1

FUNDAMENTOS

INTRODUÇÃO

Nesta unidade vamos apresentar os conceitos sobre Sistemas Operacionais, seus objetivos, funcionamento e um breve histórico de sua evolução e sua classificação, os diferentes tipos de interface, além de outros conceitos importantes para entendimento da área. Para entendermos a função de um Sistema Operacional (SO) precisamos entender, inicialmente, como é o funcionamento de um computador e como a evolução dos computadores fez surgir a necessidade de um Sistema Operacional.

Um computador nada mais é do que uma máquina de calcular composta por um processador, também chamado de CPU (abreviação do termo em inglês da Unidade Central de Processamento – Central Process Unit), por uma memória (RAM Random Access Memory) e por dispositivos de entrada de dados (tais como o teclado, mouse e microfone) e de dispositivos de saída, tais como o monitor e a impressora; ou ainda por dispositivos que fazem entrada e saída ao mesmo tempo, como as placas de rede e as unidades de armazenamento como o **Hard Disk (HD)**, **Digital VersatileDisc(DVD)** e **Solid-State Drive (SSD)**.

TERMO DO GLOSSÁRIO: Disco Rígido ou HD (abreviatura de hard disk drive): faz parte da memória secundária, não-volátil, onde são armazenados dados.

DVD: Disco ótico digital adaptado à multimídia e ao vídeo digital, com capacidade de armazenamento superior ao CD (CompactDisc)

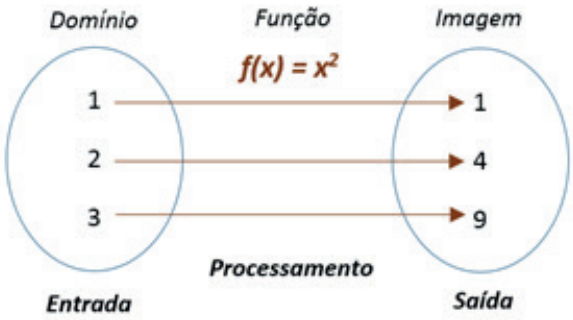
SSD: tecnologia de armazenamento considerada a evolução do HD. Não possui partes móveis e é construído com um circuito integrado semicondutor, diferentemente dos sistemas magnéticos, como o HD.

Processar os dados quando um programa é executado significa obter os dados de entrada, aplicar um processamento e gerar uma saída. O processamento de dados envolve realizar inúmeros cálculos pelo processador e realizar inúmeras movimentações de dados entre o processador e a memória e, também, entre os dispositivos de entrada e saída.

Um computador implementa um modelo matemático, semelhante ao conceito de domínio, função e imagem, como mostra a Figura 1. Neste modelo, tem-se um conjunto domínio com os valores de entrada, uma função f(x) que vai aplicar um

cálculo sobre esses valores de entrada e gerar um conjunto imagem, com os resultados da função f(x). Na ilustração tem-se um conjunto domínio (entrada) com os valores 1, 2 e 3; uma função f(x) = x2, que, a partir de cada um dos valores de entrada do conjunto domínio, aplica o respectivo cálculo (processamento), elevando o valor de x ao quadrado nesse nosso exemplo e gerando os valores do conjunto imagem (saída) com os dados processados.

FIGURA 1 – Entrada, processamento e saída



FONTE: NTE, 2017.

Processar os dados, envolve obter os valores de uma entrada, como por exemplo, por meio de um teclado ou de um mouse, aplicar um cálculo e visualizar o resultado da saída em um monitor ou em uma impressora. Isso, porém, seria o resultado apenas de uma aplicação de uma única função. O processamento de dados de um programa de computador envolve bilhões de cálculos e, para implementá-los, seriam necessários bilhões de componentes. Ainda assim, o computador faria sempre o mesmo tipo de cálculo, ou seja, teria sempre o mesmo programa, de forma semelhante ao que temos em nossas calculadoras portáteis. Uma calculadora consegue calcular, de forma muito rápida e precisa, mas somente aqueles cálculos básicos para os quais ela foi implementada.

Para fazer computação propriamente dita, é necessário entender outro conceito: o conceito de programa armazenado em memória. Esse modelo permitiu construir computadores genéricos, cujo programa é armazenado na memória e, a cada ciclo, uma nova instrução é buscada na memória, decodificada e executada sobre os dados de entrada, gerando novos dados de saída, que podem ser enviados aos dispositivos de saída ou transformando-se nos valores de entrada da instrução do ciclo seguinte. Nesse modelo, um programa de computador é formado pelas sequências de instruções para o processador executar as operações de cálculo ou movimentação de dados. Assim, um programa implementa uma infinidade de funções f(x), que obtêm os dados de entrada e geram os dados de saída.

Para se ter uma idéia do que isso representa, quando estamos assistindo um vídeo de um clipe de música no computador com duração de 3 minutos, o computador busca do dispositivo de entrada (que pode ser o disco rígido ou a placa de rede do computador) uma sequência de aproximadamente 500.000.000 (quinhentos milhões) de bits, formados de números 0 e 1 e aplica milhares de cálculos, podendo chegar a aproximadamente 5 bilhões de cálculos por segundo, para transformar

esses zeros e uns em uma imagem em movimento e envia-os para a placa de vídeo, ao mesmo tempo em que o áudio é enviado para a placa de som.

Nesse exemplo, os dados de entrada são os zeros e uns (dados binários) que estão no arquivo de vídeo, provavelmente em um formato **.mp4** com aproximadamente 50MB de tamanho e a entrada é o disco rígido, ou o pendrive, onde este arquivo está armazenado. O processamento é realizado pelo programa que vai executar esse arquivo multimídia, por exemplo, o VLC Media Player ou o Windows Media Player, que contém uma sequência de instruções capazes de transformar os bits de entrada em outros bits para serem enviados para a placa de vídeo e ainda outros bits para serem enviados simultaneamente para a placa de áudio. As saídas são as imagens mostradas no monitor e o áudio reproduzido nas caixas de som ou no fone de ouvido.

TERMO DO GLOSSÁRIO: MP4: MPEG-4 Part 14, um padrão de áudio e vídeo, parte da especificação MPEG-4 desenvolvido pela ISSO/IEC 14496-14. Similar ao popular formato AVI.

O que difere um computador de uma calculadora é o fato dele ter um programa armazenado em memória que determina o que será realizado e o computador faz o processamento dos dados de acordo com o programa, que pode ser alterado. Em uma calculadora nós não podemos adicionar uma nova função nela, ao passo que, em um computador, basta trocar o programa. Um MP3 Player também executa um arquivo de áudio no formato .mp3, mas ele só faz isso. Não dá para trocar o programa do MP3 player; dá para trocar a música, mas não o programa. Se ele só executa áudio no formato .mp3 ele não vai reproduzir um arquivo .wma; já em um computador, basta trocar o programa que ele executa qualquer coisa para a qual foi programado.

E onde entra o SO nessa história? O SO se encarrega de esconder os detalhes do hardware para facilitar a vida do programador e do usuário. Para executar um programa o usuário só precisa clicar ou tocar com o dedo se a tela for touchscreen (tela sensível ao toque) sobre o ícone do programa. Toda essa facilidade é dada pelo SO. Ao executar o programa, o usuário vai precisar informar onde está o arquivo multimídia a ser executado e o SO oferece uma visão do sistema de arquivos de forma organizada, com unidade e pastas nas quais o usuário coloca seus arquivos e documentos. Depois de informar o nome do arquivo, o programa simplesmente vai pedir para o SO carregar o arquivo com um determinado nome e o programa não precisa ter a menor idéia se os dados estão em um disco rígido, em um SSD, em um pendrive e como se faz para buscar estes dados do arquivo de entrada, pois desses detalhes o SO está encarregado. Durante a execução do programa, milhares de instruções são enviadas para o processador e este vai processando os dados e exibindo o áudio e o vídeo, sem se preocupar em saber como deve ser a sequência de zeros e uns que devem ser enviadas para a placa de vídeo e para a placa de som, muito menos se preocupar com o tipo da placa de vídeo e da placa de som instalado.

Essa é a função do SO: a de proporcionar que os mais diversos programas utilizem os mais diferentes componentes do hardware de modo seguro e padro-

nizado, independente da marca e modelo instalado, fornecendo uma interface amigável para que o usuário possa usar um computador sem a necessidade de ser um especialista.

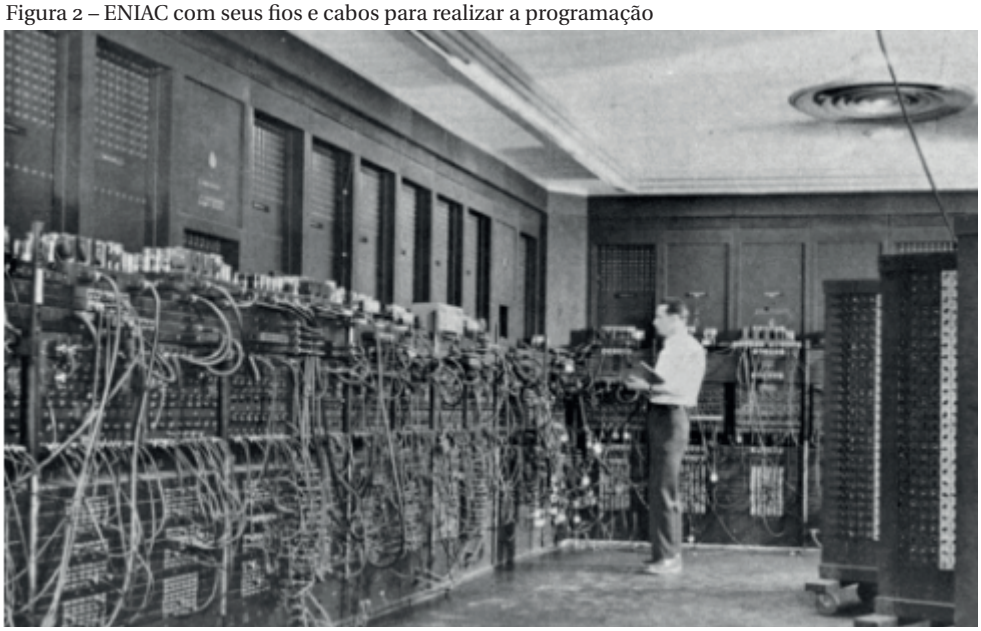
1.1
FUNDAMENTOS DE SISTEMAS OPERACIONAIS

Nos primeiros computadores criados nas décadas de 1940 e 1950, os programas eram desenvolvidos diretamente em linguagem de máquina e exigiam, dos programadores, um altíssimo conhecimento do **hardware** do computador para seu desenvolvimento. Com isso, os programas eram extremamente complexos, pois a maior parte das instruções eram para fazer o hardware funcionar. Além disso, somente um programa poderia ser executado por vez.

TERMO DO GLOSSÁRIO: Hardware: é a parte física de um computador, formado pelos componentes elétricos, eletrônicos, chips, e circuitos, sendo a máquina física propriamente dita.

No início da computação centenas de fios eram interconectados e centenas ou milhares de chaves eram posicionadas para fazer a programação de um computador. Ao final do processamento algumas luzes no painel eram acesas e os programadores (na época, geralmente engenheiros) faziam as leituras dessas luzes e realizavam alguns cálculos para interpretar os resultados. A Figura 2 apresenta o ENIAC, que foi o primeiro computador eletrônico de propósito geral, com seus painéis com fios e chaves de programação.

Figura 2 – ENIAC com seus fios e cabos para realizar a programação

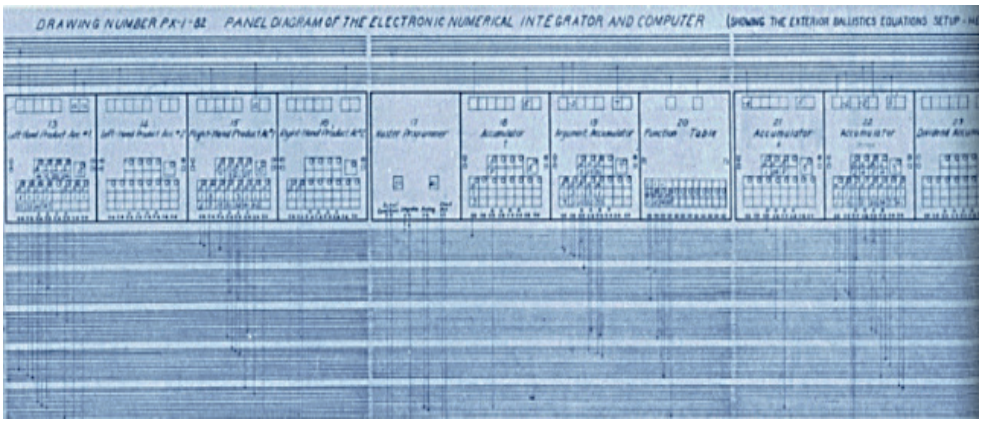


Fonte: Wikipédia. Disponível em: https://en.wikipedia.org/wiki/ENIAC#/media/File:Eniac.jpg

Uma das peculiaridades que distinguiram o **ENIAC** de todos os computadores posteriores foi a forma como as instruções eram configuradas na máquina. Os engenheiros exerciam o papel de programadores, determinando quais módulos deveriam ser interligados e em que ordem. Um grupo de técnicos treinados de posse de esquemas de ligação realizava a conexão dos módulos, de forma que a máquina pudesse ser energizada e os cálculos realizados. Nesta época o computador não era programável, pois seu hardware era modificado para cada tipo de problema diferente, o que representava uma tarefa complexa e delicada. Para realizar a programação eram usados gráficos de programação, mostrados na Figura 3, com a representação da fiação e conexões necessárias para configurar um programa. (STALLINGS,2002).

INTERATIVIDADE: Você pode visualizar o ENIAC em operação no vídeo ENIAC thefirst general purposeelectroniccomputer,disponível em https://www.youtube.com/watch?v=9QTHZNNzWCo

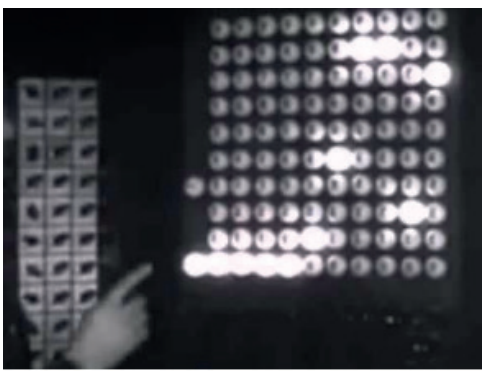
Figura 3 - Gráfico de programação ENIAC que representa a fiação para configurar uma equação balística.



Fonte: Columbia University. Disponível em: http://www.columbia.edu/cu/computinghistory/eniac--program.gif

Após os cálculos, o resultado era apresentado em um painel de luzes, mostrado na Figura 4, cujos resultados deviam ser interpretados por um especialista.

Figura 4 – Painel de luzes do ENIAC que mostra o resultado do cálculo



Fonte: ENIAC. Disponível em: https://i.ytimg.com/vi/k4riLQeVH4w/hqdefault.jpg

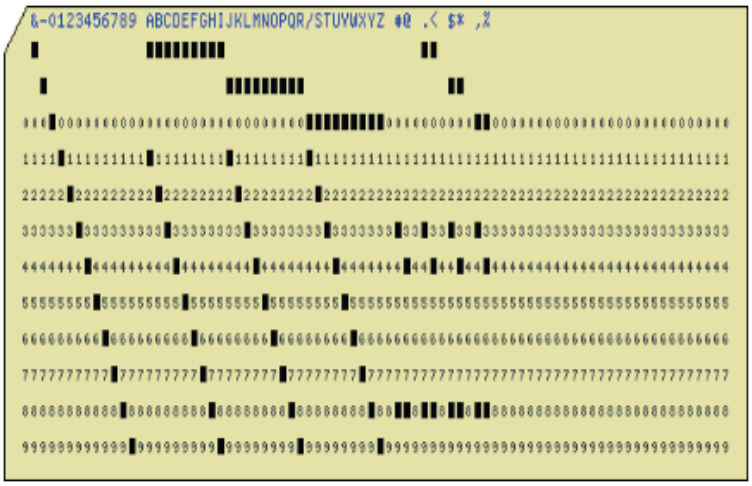
Os primeiros computadores eram enormes e de difícil operação, pois eram grandes máquinas de calcular, fabricados com circuitos baseados em relés, dispositivos eletromecânicos e válvulas eletrônicas. Apresentavam constantes problemas de funcionamento devido à queima desses componentes, exigindo que todo trabalho fosse recomeçado. Devido ao custo altíssimo de produção e operação, seu uso era restrito a órgãos militares e governamentais e grandes universidades (JANDL, 2004).

Os computadores passaram a ser desenvolvidos baseados no modelo de máquina genérica proposto pelo matemático Von Neumann, formado por um bloco de memória, um bloco de processamento e um bloco de dispositivos de entrada e saída. Neste modelo, a sequência de passos necessária para a execução de um programa é armazenada na própria memória do computador e pode resolver diferentes tipos de problemas sem necessidade de alterar o hardware. Este conceito de programa armazenado na memória e computador programável é utilizado até hoje.

Na década de 1950, a fabricação do transistor permitiu o desenvolvimento de computadores menores e mais confiáveis, iniciando a comercialização para grandes empresas e órgãos governamentais. Em 1951 foi criado o oUnivac-I (*Universal Automatic Computer*), que foi o primeiro computador comercial, e em 1953 a IBM lançou o IBM 701, seu primeiro computador digital. A programação desses computadores ainda exigia um alto grau de conhecimento sobre o funcionamento dos mesmos, e dos seus circuitos e dispositivos, de modo que o uso dos mesmos era restrito a especialistas (JANDL, 2004).

Como a possibilidade de programar o computador ao invés de conectar fios e cabos, foi desenvolvido um mecanismo no qual os programas eram codificados em cartões perfurados, mostrados na Figura 5. A Figura 6 apresenta uma máquina para codificar os cartões, que posteriormente eram lidos e interpretados pelo computador para a execução do programa.

Figura 5 – Cartão perfurado



Fonte: Wikipédia. Disponível em: https://en.wikipedia.org/wiki/Keypunch#/media/File:IBM_026_card_code.png

Figura 6 – Máquina de imprimir cartões perfurados

Fonte: Wikipédia. Disponível em: https://en.wikipedia.org/wiki/Keypunch#/media/File:IBM26.jpg

Os programas e os dados para serem processados eram organizados em tarefas, que eram lidas por uma máquina leitora de cartões perfurados e os dados eram gravados numa fita magnética. A fita magnética era levada para o computador, que interpretava a tarefa e realizava o processamento, gravando em outra fita magnética os resultados de cada tarefa. Esta fita de saída era lida por outra máquina que fazia a impressão das listagens e relatórios (JANDL, 2004).

Esses primeiros computadores utilizavam cartões perfurados como entrada e muitas páginas de impressão de dados como saída. Desta forma, era necessário um enorme conhecimento dos programadores e técnicos da época para compreender o que os incontáveis caracteres significavam. Assim, a programação que inicialmente era realizada totalmente baseada em ligar e desligar chaves e conectar e desconectar cabos, passou a ser realizada por meio da codificação de programas, que continuam fazendo com que chaves e circuitos sejam ligados ou desligados, porém de forma eletrônica, pois um programa de computador é um conjunto de instruções para o processador sob a forma de código binário (uma sequência de zeros e uns), que fazem com que milhares de chaves e circuitos sejam ligados e desligados para realizar os cálculos e computar os resultados.

Ainda assim, a programação era uma tarefa complexa, pois cada programador precisava conhecer todos os detalhes do hardware do computador. Com o passar do tempo, os fabricantes de computadores passaram a desenvolver um programa que controlasse o hardware do computador e esse programa passou a fornecer as funcionalidades do hardware para os demais programas. Assim, ao desenvolver um programa de computador, o programador simplesmente fazia, de forma bem mais simples, uma requisição a esse programa para acessar as funcionalidades do hardware. Com isso, o programador passou a não se preocupar mais com o funcionamento do hardware e pôde focar na programação específica do seu aplicativo.

Esse **software** que gerencia o hardware ficou encarregado de todo o serviço complexo. Esse software, responsável por gerenciar o hardware, é chamado de **Sistema Operacional (SO)**. Um SO é, portanto, um software que habilita as aplicações a interagir com o hardware de um computador (DEITEL, 2005).

TERMO DO GLOSSÁRIO: Software: é uma sequência de instruções escritas para serem executadas por um computador com o objetivo de executar tarefas específicas. São os programas de computador.

Sistema operacional: é um programa ou um conjunto de programas que habilita as aplicações a interagir com o hardware de um computador, gerenciando os recursos e oferecendo uma interface amigável para o usuário.

A utilização de um sistema operacional permitiu o uso e programação do computador de forma mais segura, confiável e coerente, pois a gerência dos recursos do hardware deixou de ser preocupação do programador. Isso também permitiu que vários programas fossem executados simultaneamente, como um lote de tarefas, dando origem ao termo processamento em lote (batch processing). Os sistemas em lote impulsionaram o uso comercial dos computadores na década de 1950, quando foi desenvolvida a primeira linguagem de programação de alto nível: o IBM FORTRAN (*Formula Translator*), em 1957 (JANDL, 2004).

Na década de 1960, os computadores e sistemas operacionais foram ficando cada vez mais robustos e complexos, exigindo uma gerência cada vez mais sofisticada e proporcionando uma interface e forma de utilização mais facilitada, para que os mesmos pudessem ser utilizados não apenas por especialistas. Nesta década surgiu a linguagem de programação COBOL (*Commom Business Oriented Language*), para auxiliar o desenvolvimento de sistemas comerciais. Em 1961 iniciou a comercialização dos primeiros circuitos integrados.

Na década de 1960, buscando uma utilização mais eficiente e segura dos computadores, os sistemas operacionais foram se tornando cada vez mais complexos, gerenciando os recursos do computador de forma cada vez mais sofisticada. Ao mesmo tempo em que os sistemas operacionais deixavam o uso mais eficiente e seguro do computador, buscavam-se alternativas para que pessoas menos especializadas pudessem utilizar o computador, concentrando-se em suas verdadeiras tarefas e ampliando as possibilidades de uso dos mesmos (JANDL, 2004).

A tecnologia continuava avançando cada vez mais e foi desenvolvido o Teleimpressor, que exibia as imagens em uma tela de televisão, evitando assim uma quantidade enorme de impressões. Em 1963 a DEC introduziu o uso de terminais de vídeo (JANDL, 2004). O monitor só exibia as imagens dos códigos, com números e letras. Na verdade, um monitor de computador é formado por uma infinidade de pequenas "luzes" que se acendem e apagam, formando as imagens. Assim, um hardware foi desenvolvido para converter os resultados dos cálculos do computador em imagens que representavam os números e letras necessárias para entender a saída dos programas. Conceitualmente, o computador continua igual, mostrando o resultado por meio de luzes que acendem e apagam, porém tão pequenas que

conseguem formar outras imagens. Para se ter uma idéia de como são formadas as imagens em uma tela, um notebook convencional tem uma tela de LED com 1366 pixels de largura por 768 **pixels** de altura, o que dá exatamente 1.049.088 pixels; como cada pixel é formado por uma combinação de 3 cores básicas (***Red Green Blue - RGB***): vermelho, verde e azul, a tela de um notebook tem 3.147.264 luzes que se acendem e apagam para formar as imagens que vemos em nossos monitores.

TERMO DO GLOSSÁRIO: Pixel: ponto luminoso do monitor de vídeo que, juntamente com outros pontos, forma as imagens na tela.

No sistema RGB as cores são obtidas por meio da mistura das 3 cores primárias (vermelho, verde e azul), cada uma em uma escala que varia de 0 a 255.

Um dos importantes avanços ocorridos na década de 60 foi a possibilidade de execução simultânea de vários programas, chamada de **multiprogramação**, que permitia a execução de várias tarefas simultaneamente. Quando um programa está em execução, o mesmo está carregado na memória principal do computador (memória RAM) e cada instrução do programa faz com que seja executada uma operação do processador ou uma movimentação de dados de ou para um dispositivo de entrada e saída. Desta forma, um programa está usando, em um determinado momento, ou o processador ou um dispositivo de saída. Com a multiprogramação, enquanto um recurso está em uso por um programa, outro programa pode utilizar, no mesmo instante o outro recurso.

TERMO DO GLOSSÁRIO: Multiprogramação: é quando vários programas ou jobs estão na memória principal simultaneamente com o processador sendo alternado de um job para outro job, executando-os enquanto os dispositivos periféricos são mantidos em uso quase constante (DEITEL, 2005).

A multiprogramação permitiu o uso mais eficiente dos computadores, por meio da divisão da memória para cada tarefa, permitindo que mais de uma tarefa seja carregada para a memória e executada. Em meados da década de 60, os primeiros sistemas operacionais foram desenvolvidos seguindo a evolução da tecnologia da época, porém cada tipo de computador possuía um SO específico, o que implicava na incompatibilidade entre diferentes computadores. Um dos SOs mais importantes da época foi o CTSS, criado pelo MIT, sendo lançado em 1961 para o computador IBM 7090 (JANDL, 2004).

Com a evolução do desenvolvimento do software, surgiram os sistemas que permitiam o uso simultâneo do computador por diversos usuários, onde o processador é compartilhado. A ideia central destes sistemas era dividir o poder computacional de um computador entre seus vários usuários, dando a impressão de que o computador estava integralmente disponível para cada usuário, mesmo que isto não fosse verdade. Nestes sistemas, denominados de **sistemas de tempo compartilhado** (*time-sharing systems*), o tempo do processador era dividido em pequenos intervalos

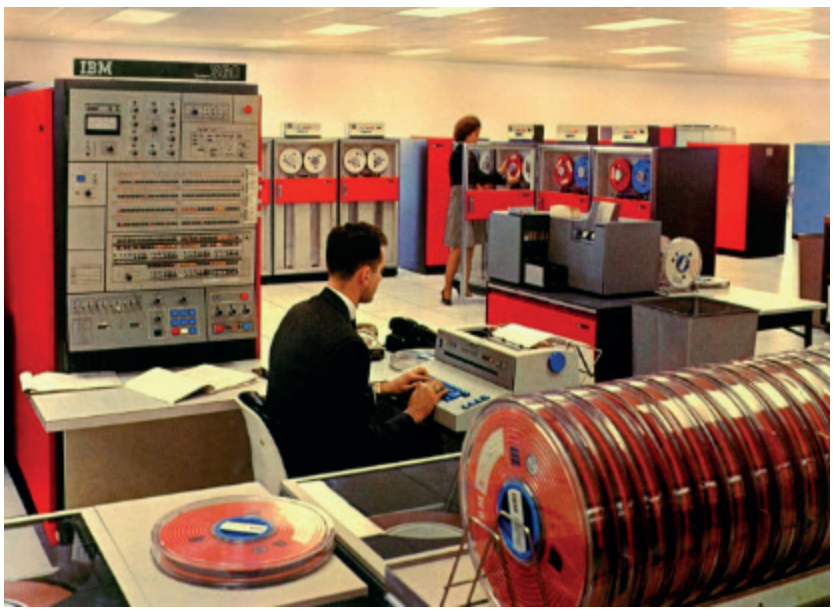
denominados fatia de tempo (time-slice). Tais fatias de tempo eram distribuídas de forma sequencial entre os processos em execução, de modo que a espera entre os intervalos fosse imperceptível para os usuários. Cada processo utiliza tantas fatias de tempo quanto necessário, mas apenas uma a cada ciclo, retornando para ele novamente no ciclo seguinte. Com isto, a capacidade de processamento do computador era dividida proporcionalmente entre os processos (JANDL, 2004).

> TERMO DO GLOSSÁRIO: Sistemas de Tempo Compartilhado: são sistemas que permitem que vários programas estejam executando simultaneamente num único computadoralternando o uso de CPU e dos dispositivos de entrada e saída para cada um dos programas de forma imperceptível ao usuário.

Para implementar a técnica de tempo compartilhado foi desenvolvido o mecanismo de interrupção, que permite parar a execução de um programa para tratar de uma outra rotina, retornando à execução prévia, assim que o tratamento da interrupção for concluído. Para a multiprogramação também foram criados os mecanismos de identificação de usuário e proteção com senha (user id e password) e das rotinas de início (login ou logon) e término (logout) de sessão de trabalho (JANDL, 2004).

Na década de 1970 houve um aumento na complexidade e sofisticação dos sistemas computacionais, com a utilização de circuitos integrados no desenvolvimento de computadores comerciais e a criação de famílias de computadores compatíveis, iniciadas com o lançamento do IBM System/360, que inaugurou uma nova era na computação e a expansão de sua utilização. Neste período surgiram os **mainframes** (computador principal) e se popularizaram os centros de processamento de dados (CPDs) (JANDL, 2004). A Figura 7 apresenta um mainframe IBM 360 da década de 70.

Figura 7 – Mainframe IBM 360 (década de 1970)



Fonte: IBM. Disponível em: http://www.ibm360.info/ibm_360-50.jpg

Visando contornar o problema da incompatibilidade de SOs de máquinas distintas, um grupo da AT&T criou o Unix em 1969, sendo o primeiro SO moderno da computação. Vários outros SOs foram influenciados de alguma maneira pelo Unix. Sua primeira versão foi escrita em linguagem ***Assembly***, sendo posteriormente reescrito na linguagem de programação C no ano de 1973, cuja linguagem ainda é utilizada até os dias de hoje. Este sistema popularizou conceitos importantes para a computação, principalmente os de portabilidade, multiusuário, multitarefas e compartilhamento de recursos.

SAIBA MAIS: Assembly é uma linguagem de programação de baixo nível cujo código-fonte é formado por mnemônicos que representam as instruções de linguagem de máquina.

Durante a década de 70, o Unix foi distribuído gratuitamente, inclusive seu código-fonte, para universidades e órgãos governamentais norte-americanos, o que conferiu muita popularidade a este sistema. Sua interface era totalmente em modo texto, sem interface gráfica. Em 1977 foi lançado o BSD, SO fortemente baseado no Unix, focado principalmente para a execução em computadores específicos de alto desempenho, como o VAX, que foi uma referência de hardware na época.

Tanto o Unix quanto o BSD, em suas primeiras versões, foram desenvolvidos para o uso em computadores de grande porte, normalmente em universidades. Porém, alguns jovens programadores tiveram uma ideia considerada absurda para a época: criar computadores e SOs para que pessoas comuns pudessem usar os computadores. Um dos primeiros a pensar desta forma foi Steve Jobs, fundador da Apple, sempre com o principal foco de criar computadores para serem usados no dia-a-dia, com SOs fáceis de serem utilizados.

Em 1971 surgiu o primeiro microprocessador comercial integrado em um único **chip**, o Intel 4004. Neste período começaram a surgir pequenos computadores destinados ao uso pessoal ou doméstico, começando pelos microcomputadores da Xerox em Palo Alto, em 1973, o Altair 8800 destinado ao consumo em massa em 1974, o IBM 5100 em 1975 e o sucesso de vendas do Apple 2 lançado em 1977. Começaram também a ser desenvolvidas as primeiras aplicações comerciais dos microcomputadores.

TERMO DO GLOSSÁRIO: Chip: é uma pequena lâmina miniaturizada (geralmente de silício), usada na construção de semicondutores, capaz de realizar funções complexas.

Pela primeira vez, um computador possuía um teclado fácil de ser utilizado, com uma pequena televisão adaptada como monitor e não eram necessários conhecimentos avançados de computação para utilizá-lo. Steve Jobs criou o seu SO do zero, sem se basear inicialmente no Unix. Nos anos seguintes, os modelos Apple II e Apple III foram lançados no mercado, sendo um sucesso de vendas. Suas interfaces gráficas eram bastante primitivas se comparadas com o padrão usado atualmente, mas eram avançadíssimas para a época. O Apple II foi lançado a um custo de US$ 1300 na época, equivalente atualmente a US$ 5000. A Figura 8 apresenta um

computador Apple 2.

Figura 8 – Apple II (1976)

Fonte: Wikipédia. Disponível em: https://en.wikipedia.org/wiki/File:Apple_II_IMG_4212.jpg

Em meados de 1979, Steve Jobs tomou conhecimento sobre o desenvolvimento de um computador totalmente inovador pela Xerox Parc. Em uma visita a esta empresa, ele ficou deslumbrado com Xerox Alto, um Personal Computer (**PC** ou computador pessoal) que possuía uma interface gráfica – GraphicalUser Interface (GUI) totalmente revolucionária. Pouco tempo depois, a Apple lançou o Lisa, aproveitando todas as ideias gráficas do computador, fazendo um enorme sucesso, porém cada versão possuía um SO distinto. Em 1994 o Apple Machintosh foi lançado com muito sucesso, introduzindo o conceito de desktop, utilizando ícones e pastas para representar programas e arquivos do modo como conhecemos hoje, com o inovador sistema chamado MAC OS.

INTERATIVIDADE: O filme Pirates of Silicon Valley (Piratas da Informática ou Piratas do Vale do Silício) apresenta uma versão dramatizada do nascimento da era da informática doméstica, desde o primeiro PC, através da histórica rivalidade entre a Apple e seu Macintosh e a Microsoft, indo desde o Altair 8800 da empresa MITS, passando pelo MS-DOS, pelo IBM PC e terminando no Microsoft Windows.

Em 1980, William (Bill) Gates e Paul Allen, fundadores da Microsoft, compraram o sistema QDOS (Quick and Dirty Operating System) de Tim Paterson e passaram a chamá-lo de DOS (Disk Operating System), vendendo licenças deste SO à IBM. O DOS vendeu inúmeras cópias, como o SO padrão para os computadores pessoais desenvolvidos pela IBM e possibilitou que a Microsoft se transformasse na maior empresa de software do mundo.

Na década de 80 surgiu a geração de microcomputadores, com os sistemas desktop, a partir do lançamento do IBM-PC (Personal Computer) em 1981 e o de-

senvolvimento de uma enorme indústria de hardware, software e serviços originadas nestes eventos. Nestes computadores pessoais, o SO apresentava uma interface basicamente em modo texto, permitindo a execução de um único programa por vez. A Figura 9 apresenta um computador IBM-PC.

Figura 9 – IBM-PC

Fonte: Wikipédia. Disponível em: https://upload.wikimedia.org/wikipedia/commons/thumb/6/69/IBM_PC_5150.jpg/280px-IBM_PC_5150.jpg

Nesta década surgiram, também, empresas especializadas em arquiteturas de alto desempenho e são desenvolvidos os primeiros **supercomputadores** tais como os modelos 1-A ouX-MP da CrayResearch, que era vendido a US$ 15milhões. A Figura 10 apresenta um supercomputador **Cray X-MP.**

Figura 10 – Supercomputador Cray X-MP

Fonte: Computer History. Disponível em: http://archive.computerhistory.org/resources/still-image/Cray/cray_x-mp-4.1985.102635318.lg.jpg

TERMO DO GLOSSÁRIO: Supercomputadores são computadores com um altíssimo poder de processamento, utilizados para cálculos científicos, simulações, previsões meteorológicas e aplicações gráficas. Normalmente executam um único programa por vez, que exige várias horas de processamento.

SAIBA MAIS: Um supercomputador Cray X-MP de 1982 custava US$ 15 milhões e era 10000 vezes mais rápido que um computador desktop da época, porém, o desempenho de um computador desktop atual, que custa aproximadamente US$ 1 mil é cerca de 100 vezes mais rápido que o Cray X-MP de 1982.

Os assistentes pessoais chamados de PDAs, handhelds, ou palmtop, que eram dispositivos portáteis com funções de processamento de dados começaram a aparecer a partir de 1984. Eles não eram integrados a telefones, eram apenas aparelhos multimídia com aplicativos para funções de agenda e sistema de escritório elementar.

A década de 1990 se caracterizou pelo uso das estações de trabalho (workstations) e pela computação pessoal. Os microcomputadores, com poder de processamento maior do que os mainframes da década de 1970, se popularizam. Os supercomputadores passam a ser fabricados quase que artesanalmente. Os notebooks, palmtops e PDAs (*Personal Digital Assistents*) passaram a ser usados pela ampla maioria dos profissionais. Estes aparelhos portáteis possuíam versões simplificadas de sistemas operacionais, que permitiam que uma gama maior de serviços e funcionalidades fosse oferecida. Além disso, a eletrônica embarcada começa a ser tornar presente em automóveis e utensílios domésticos, além das tradicionais aplicações industriais. Surge uma tendência de downsizing, que tira o poder computacional dos mainframes e transfere para as estações de trabalho.

A Internet se popularizou e com ela o uso de sistema de correio eletrônico (e-mail), grupos de discussão, educação a distância, aplicações multimídia e comércio eletrônico. Para atender essa demanda e complexidade e necessidade de interligação de todos esses equipamentos, milhões de linhas de código são necessárias para a criação de um sistema operacional. No começo da década de 1990, o estudante de computação finlandês Linus Torvalds impulsionou o desenvolvimento do sistema operacional Linux por uma comunidade de programadores do mundo todo e popularizou o SO baseado em software livre.

O aumento do poder computacional, aliado à significativa redução do tamanho dos componentes e dos seus custos, foram um atrativo à integração dos computadores com celulares. Sendo assim, começaram a surgir os smartphones, que são computadores de bolso e com todas as funcionalidades de um telefone. O primeiro produto a combinar as funcionalidades de celular com PDA foi o IBM Simon lançado em 1993, com tela de toque e conseguia acessar e-mails. A Figura 11 apresenta o Simon, o primeiro smartphone.

Figura 11 – IBM Simon – o primeiro smartphone

Fonte: Wikipédia. Disponível em: https://pt.wikipedia.org/wiki/Smartphone#/media/File:IBM_Simon_Personal_Communicator.png

Na década de 2000 a supercomputação se torna realidade e os sistemas distribuídos se expandem com as técnicas de computação em grid e o emprego de clusters (agrupamentos) de computadores geograficamente dispersos e interligados, através da Internet ou conectados através de redes locais de alto desempenho. O desenvolvimento e miniaturização de novos componentes eletrônicos, aliados à evolução das redes digitais de telefonia móvel, permitiram que muitos milhões de usuários passassem a utilizar os smartphones e se popularizaram os notebooks.

A partir de 2010 se popularizaram os tablets e os smartphones, que passaram a ter um alto poder computacional, e a computação móvel passou a dominar. Nesse período se populariza a **computação em nuvem** (*cloudcomputing*), na qual os dados e aplicações passam a ser armazenados em servidores espalhados pelo mundo.

> TERMO DO GLOSSÁRIO: Computação em nuvem (cloudcomputing) refere-se à utilização da memória e da capacidade de armazenamento e cálculo de computadores e servidores compartilhados e interligados por meio da Internet, baseado no princípio da computação em grade (cluster).

Com o passar do tempo, os SOs evoluíram de forma bastante significativa, permitindo que diferentes componentes de computador, até mesmo de diferentes fabricantes, pudessem ser adicionados ao mesmo. Além disso, passaram a oferecer uma interface muito mais rica e intuitiva fornecendo uma interface gráfica com a qual podemos interagir por meio do mouse e das telas touchscreen, além de técnicas como reconhecimento de voz.

1.2 PRINCIPAIS SISTEMAS OPERACIONAIS

Os principais SOs para computadores desktop e notebooks atualmente são:

a) **Windows 10**: é um SO proprietário, desenvolvido pela Microsoft, lançado em 2014, apresenta um design com características de versões anteriores para manter a compatibilidade e atender o gosto de usuários destas versões anteriores; sua interface foi desenvolvida para telas sensíveis ao toque e agrada tanto em telas touchscreen quanto em desktops. É o SO mais popular, especialmente em países onde computadores PC e compatíveis são a maioria;

b) **Linux**: o SO Linux é uma coleção de software livre ou código aberto, criada por indivíduos, grupos e organizações de todo o mundo, incluindo o núcleo Linux. O núcleo (Kernel) Linux foi originalmente escrito por Linus Torvalds, do Departamento de Ciência da Computação da Universidade de Helsinki, Finlândia, com a ajuda de vários programadores voluntários por meio da Internet. É distribuído por meio de comunidades e equipes que preparam pacotes de distribuição com os programas e assistentes necessários para realizar a instalação, dentre as quais se destacam as distribuições: Ubuntu, Mint, Debian, Fedora, SUSE e OpenSUSE. Pode ser instalado praticamente em qualquer computador;

c) **MacOSX**: é um sistema proprietário baseado no kernel do Unix, fabricado e vendido pela Apple, destinado exclusivamente aos computadores e notebooks Mac e que combina a experiência adquirida com a tradicional GUI desenvolvida para as versões anteriores do Mac OS com um estável e comprovado núcleo. Assim, o MacOS é uma combinação de um moderno e confiável SO com uma renovada interface gráfica do usuário (GUI) chamada Aqua.

Os principais SOs para smartphones e tablets são:

a) **Android**: é baseado no núcleo Linux e desenvolvido pela Google, tendo sido projetado para dispositivos móveis como smartphones e tablets; com interface específica para TV (AndroidTV), carro (Android Auto) e relógio de pulso (AndroidWear). Utiliza-se da tela sensível ao toque permitindo ao usuário manipular os objetos virtuais e também de um teclado virtual. Apesar de ser principalmente utilizado em dispositivos com tela sensível ao toque, também é utilizado em consoles de videogames, câmeras digitais, computadores e outros dispositivos eletrônicos. O Android é o SO móvel mais utilizado do mundo. Disponibiliza uma loja de aplicativos, a Google Play, com mais de 1 milhão de aplicativos disponíveis, baixados mais de 50 bilhões de vezes. Mais de 70% dos programadores para sistemas móveis desenvolvem seus aplicativos para o Android. O código do SO é disponibilizado pela

Google sob licença de código aberto, apesar de que a maior parte dos dispositivos é lançada com uma combinação de software livre e software proprietário;

b) **iOS**: é um sistema operacional móvel criado e desenvolvido pela Apple exclusivamente para seu hardware. É o SO que atualmente alimenta os dispositivos móveis da empresa, incluindo o iPhone, iPad e iPod Touch. É o segundo SO móvel mais popular do mundo, após o Android. A interface do usuário iOS é baseada em manipulação direta, usando gestos multi-touch. Os elementos de controle da interface consistem em controles deslizantes, interruptores e botões. A interação com o SO inclui gestos como swipe, toque, pitada e pitada reversa, todos com definições específicas no contexto do SO iOS e sua interface multi-touch. Os acelerômetros internos são usados por algumas aplicações para responder a agitação do dispositivo. Apresenta funções de acessibilidade completas, permitindo que usuários com deficiência visual ou auditiva usem corretamente o iOS;

c) **Windows Phone**: é um SO para smartphones, desenvolvido pela Microsoft; apresenta características de interface semelhantes às do Windows para desktop. Disponibiliza as principais aplicações do mercado em loja de aplicativos, além de alguns aplicativos e jogos exclusivos. O sistema ainda conta com ferramentas como a Cortana, assistente de voz pessoal, a integração nativa com o Microsoft Office e a sincronização com o Windows em computadores.

A Figura 12 apresenta a identidade visual dos principais SOs e distribuições Linux atuais.

Figura 12 – Identidade visual dos SOs e distribuições Linux atuais

Fonte: Autores, 2017.

Dentre as principais diferenças entre eles está o fato de que o Linux e o Android são baseados em **software livre** ao passo que Windows, MacOS e iOS são **softwares proprietários** e que necessitam de licença para sua utilização.

TERMO DO GLOSSÁRIO: Software Livre: é um programa de computador que pode ser executado, copiado, modificado e redistribuído gratuitamente. Os usuários possuem livre acesso ao código-fonte do software e fazem alterações conforme as suas necessidades.

Software proprietário ou não livre é programa de computador no qual a empresa que o desenvolveu detém seus direitos de uso, edição ou redistribuição sobre o mesmo e vende ou aluga uma licença para uso do mesmo.

Atualmente esses sistemas são atualizações de suas versões originais, que foram derivados ou tiveram grande influência no seu desenvolvimento a partir de um dos sistemas: Unix, lançado em 1969; DOS, lançado em 1980 ou OS/2 lançado em 1987, conforme mostra o quadro 1, onde os SOs em negrito são os que estão disponíveis atualmente. Neste quadro podemos observar, por exemplo, que o sistema MacOS é derivado do Rhapsody, que por sua vez é derivado do NeXTSTEP, que é derivado do BSD, que é uma derivação do Unix.

Quadro 1 – Principais sistemas operacionais e suas versões e ano de lançamento

<table>
<tr><td rowspan="11">1969
UNIX</td><td rowspan="3">1977
BSD</td><td rowspan="2">1989
NeXTSTEP</td><td rowspan="2">1996
Rhapsody</td><td colspan="2">2001 MacOS X 10.0 Cheetah
2001 MacOS X 10.1 Puma
2002 MacOS X 10.2 Jaguar
2003 MacOS X 10.3 Panther</td></tr>
<tr><td>2004 MacOS X 10.4 Tiger</td><td rowspan="3">2008 iOS 1
2008 iOS 2
2009 iOS 3
2010 iOS 4
2011 iOS 5
2012 iOS 6
2013 iOS 7
2014 iOS 8
2015 iOS 9
2016 iOS 10
2017 iOS 11</td></tr>
<tr><td>1993 FreeBSD</td><td></td><td>2005 MacOS X 10.5 Leopard
2009 MacOS X 10.6 Snow Leopard
2010 MacOS X 10.7 Lion
2012 MacOS X 10.8 Mountain Lion
2013 MacOS X 10.9 Mavericks
2014 MacOS X 10.10 Yosemite
2015 MacOS X 10.11 El Capitan
2016 MacOS X 10.12 Sierra
2017 MacOS X 10.13 High Sierra</td></tr>
<tr><td rowspan="8">1991
Linux</td><td>1993
Debian</td><td colspan="3">2008 Android 1.0
2009 Android 1.5 Cupcake
2009 Android 1.6 Donut
2009 Android 2.0/2.1 Eclair
2010 Android 2.2 Froyo
2010 Android 2.3 Gingerbread
2011 Android 3.0/3.1/3.2 Honeycomb
2011 Android 4.0 Ice Cream Sandwich
2012 Android 4.1/4.2/4.3 JJelly Bean
2014 Android 4.4 KitKat
2014 Android 5.0 Lolipop
2015 Android 6.0 Marshmallow
2016 Android 7.0 Nongat</td></tr>
<tr><td>2017
Debian 8.8</td><td colspan="2">2004 Ubuntu
2017 Ubuntu 17.10</td><td>2006 Linux Mint
2016 Linux Mint 18 Sarah</td></tr>
<tr><td colspan="4">1993 Slackware
2016 Slackware 14</td></tr>
<tr><td colspan="2">1994 SUSE
2016 SUSE 12.2</td><td colspan="2">2006 OpenSUSE
2014 Open SUSE 13..2</td></tr>
<tr><td colspan="4">1995 RedHat</td></tr>
<tr><td colspan="4">2003 Fedora
2017 Fedora 26</td></tr>
<tr><td colspan="4">2011 Chrome OS</td></tr>
<tr><td rowspan="3">1980
DOS</td><td colspan="4">1981 IBM PC DOS</td></tr>
<tr><td colspan="2">1981 MS DOS</td><td colspan="2">1985 Windows 1.0
1987 Windows 2.0
1993 Windows 3.0
1992 Windows 3.1 e 3.11
1995 Windows 95
1998 Windows 98</td></tr>
<tr><td colspan="2">1999 Windows SE</td><td>2000 Windows ME</td></tr>
<tr><td rowspan="4">1987
OS/2</td><td colspan="4">1994 OS/2 Warp</td></tr>
<tr><td rowspan="2">1993 Windows NT 3.1*
*não é derivado do OS/2, mas teve uma forte influência</td><td colspan="3">2000 Windows 2000
2001 Windows XP
2006 Windows Vista
2009 Windows 7
2000 Windows Mobile
2010 Windows Phone</td></tr>
<tr><td colspan="3">2012 Windows 8
2013 Xbox OS</td></tr>
<tr><td>1996 Windows NT 4.0</td><td colspan="3">2013 Windows 8.1
2014 Windows 10</td></tr>
</table>

Fonte: Autores, 2017.

1.3
INTERFACES DOS SISTEMAS OPERACIONAIS

Atualmente os SOs possuem efeitos gráficos cada vez mais realistas e impressionantes. Os sistemas modernos utilizam uma interface gráfica do usuário (GUI – Graphical User Interface) para permitir ao usuário interagir com o mesmo, clicando em suas opções, selecionando-as, gerando comandos e executando as tarefas no computador, por meio de várias janelas de programas diferentes sendo exibidos simultaneamente e com menus, botões e ícones representativos.

Mas nem sempre foi assim. As primeiras interfaces disponibilizavam apenas um **terminal** ou um **prompt de comando** onde o usuário escrevia os comandos que deveriam ser executados. Apesar disso parecer primitivo, ainda hoje todos os SOs disponibilizam ainda essa forma de interação com o mesmo, pois isso facilita as atividades de gerenciamento e configuração de computadores tais como os usados como servidores em empresas.

A primeira interface foi desenvolvida foi no ano de 1970, pela Xerox Palo Alto ResearchCenter (PARC), e o primeiro computador a usar uma interface gráfica foi o Xerox Alto em 1973, que não era comercializável, mas de uso restrito a pesquisadores e universidades. A Figura 13 apresenta a primeira interface gráfica Xerox Alto.

Figura 13 – Interface gráfica Xerox Alto

Fonte: Wikipédia. Disponível em: https://upload.wikimedia.org/wikipedia/en/1/10/Alto_Neptune_Filemanager.gif

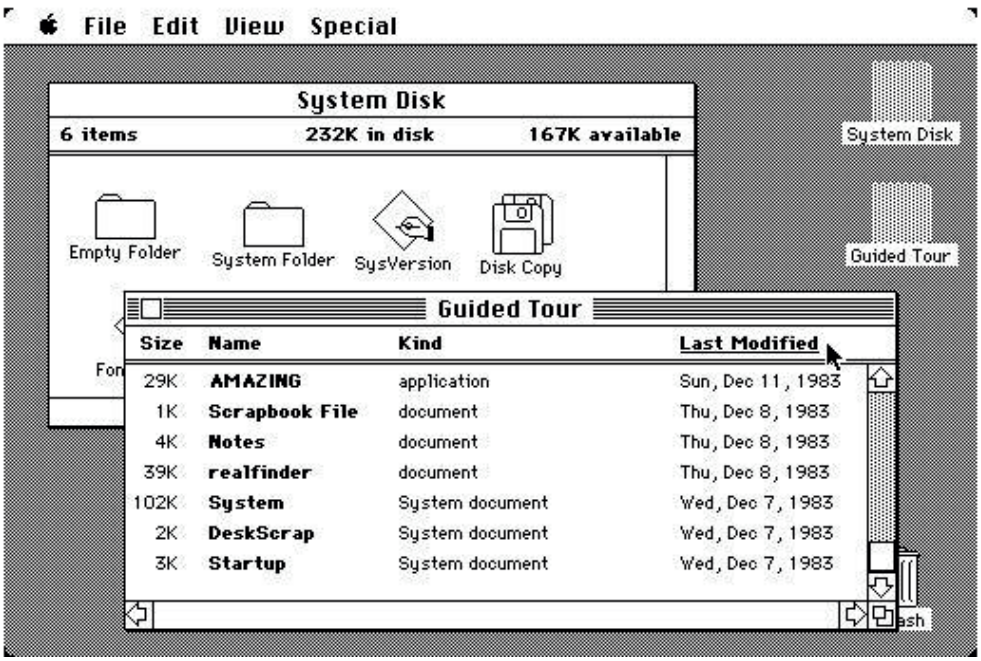
As interfaces com os usuários dos primeiros computadores pessoais no início dos anos 1980 apresentavam normalmente textos e interações por meio de comandos escritos através de um Prompt de comando, como mostra a Figura 14.

Figura 14 – Interface em modo texto dos primeiros computadores pessoais

Fonte: Old Computers. Disponível em: http://oldcomputers.net/pics/media-master.jpg

Com o lançamento dos primeiros computadores Macintosh, da Apple, as interfaces gráficas começaram a se popularizar, pois além de um visual agradável os comandos passaram a ser mais intuitivos, especialmente pelo uso do mouse, como mostra a Figura 15.

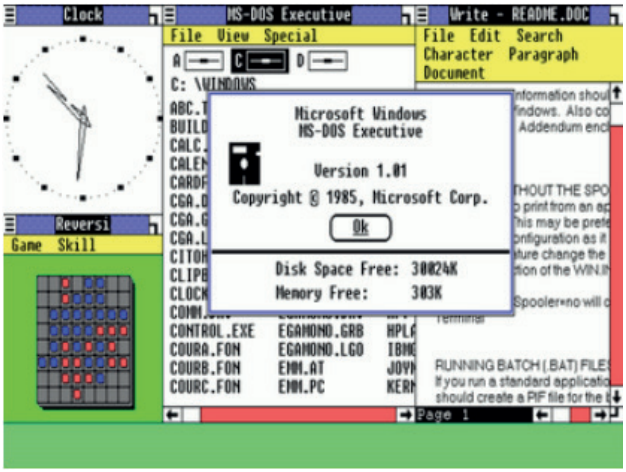
Figura 15 – Interface gráfica dos primeiros computadores Macintosh



Fonte: History Computer. Disponível em: http://history-computer.com/ModernComputer/Personal/images/MacGUI.jpg

Com o lançamento da primeira versão do Windows, em 1985, a Microsoft passou a utilizar a interface gráfica em seu novo SO. A Figura 16 apresenta a interface gráfica do Windows 1.0.

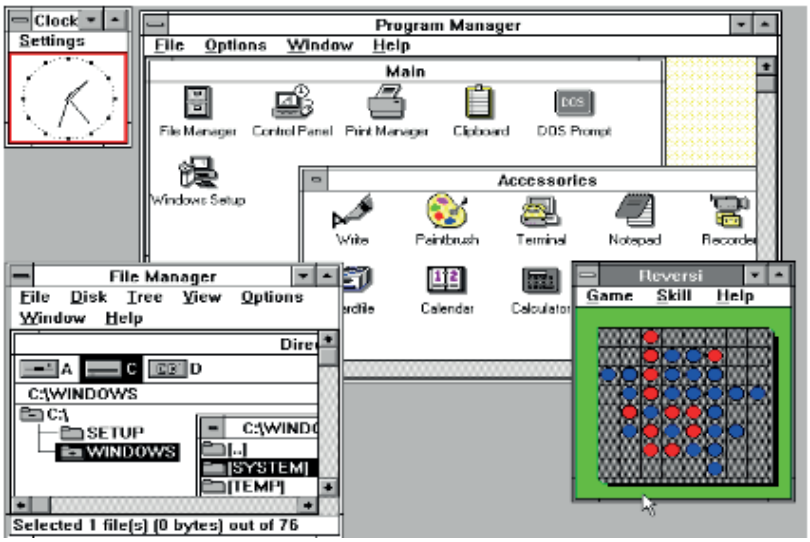
Figura 16 – Interface gráfica do Windows 1.0



Fonte: Wikipédia. Disponível em: https://upload.wikimedia.org/wikipedia/en/4/4e/Windows1.0.png

A versão 2.0 do Windows praticamente não alterou a interface gráfica, tendo uma alteração significativa no lançamento da versão 3.0 do Windows, em 1990, como mostra a Figura 17.

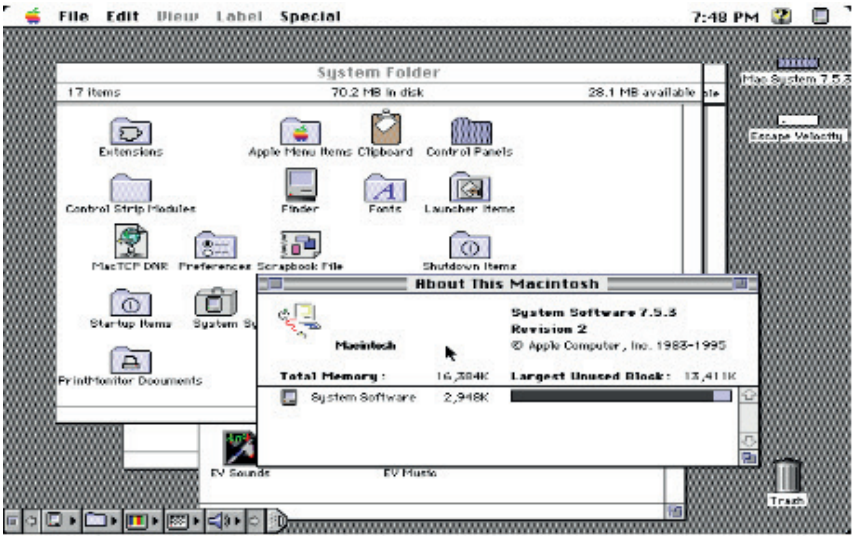
Figura 17 – Interface gráfica do Windows 3.0



Fonte: Wikipédia. Disponível em: https://en.wikipedia.org/wiki/Windows_3.0#/media/File:Windows_3.0_workspace.png

A versão 7.0 do MacOS foi a primeira GUI MacOS com suporte a cores. As cores cinza, azul e amarelo foram adicionadas aos ícones, desenvolvida em 1991. A Figura 18 apresenta a interface do MacOS System 7.

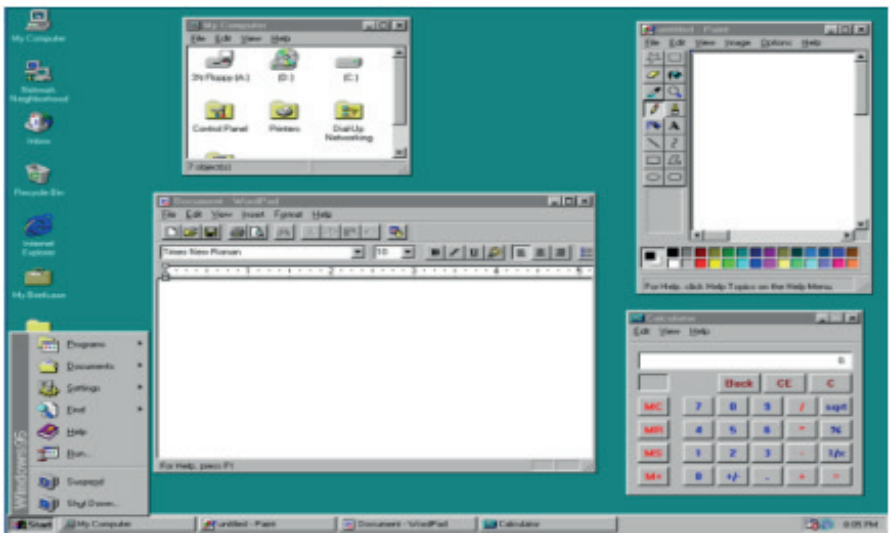
Figura 18 – Interface gráfica do MacOS System 7



Fonte: Wikipédia. Disponível em: http://apple.wikia.com/wiki/File:Macintosh_System_7.5.3_screenshot.png

A interface do Windows 95 foi completamente redesenhada, como mostra a Figura 19.

Figura 19 – Interface do Windows 95



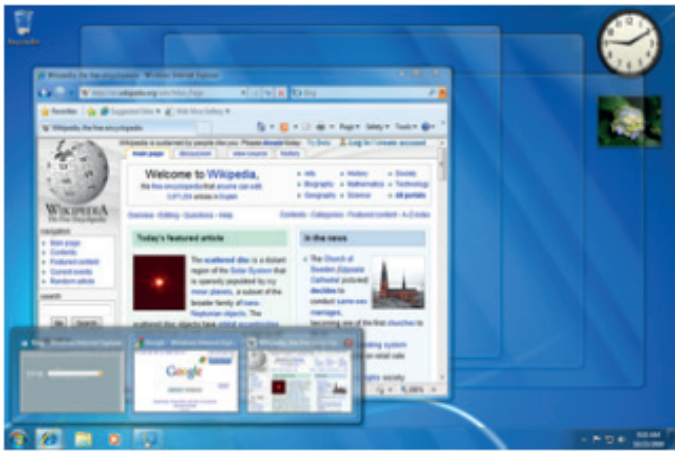
Fonte: Wikipédia. Disponível em: https://upload.wikimedia.org/wikipedia/pt/9/90/Am_windows95_desktop.png

A interface do Windows sofreu pequenas alterações e atualizações nas versões seguintes. As Figuras 20 a 23 apresentam as interfaces das versões Windows Vista, 7, 8 e 10.

Figura 20 – Windows Vista

Fonte: Wikipédia. Disponível em: https://pt.wikipedia.org/wiki/Windows_Vista#/media/File:Windows_Vista.png

Figura 21 – Windows 7



Fonte: Wikipédia. Disponível em: https://upload.wikimedia.org/wikipedia/en/3/31/Windows_7_Aero_Peak.png

Figura 22– Windows 8

Fonte: Wikipédia. Disponível em: ttps://pt.wikipedia.org/wiki/Windows_8#/media/File:Ecr%C3%A3_Inicial_do_Windows_8.png

Figura 23 – Windows 10

Fonte: Wikipédia. Disponível em: https://upload.wikimedia.org/wikipedia/en/3/32/Windows_10_%28version_1703%29.png

Os computadores da Apple com o SO MacOS X sempre foram destaque em termos de qualidade, beleza e facilidade de uso. As Figuras 24 e 25 apresentam a interface do SOMacOS X.

Figura 24 – MacOS X 10.0

Fonte: Wikipédia. Disponível em: https://en.wikipedia.org/wiki/MacOS#/media/File:MacOSX10-oscreenshot.png

Figura 25 – MacOS X 10.11 El Capitan

Fonte: Wikipédia. Disponível em: https://en.wikipedia.org/wiki/MacOS#/media/File:El_capitan_desktop_display.png

De forma semelhante, as interfaces gráficas dos SOs baseados em Linux também seguiram as evoluções em termos de beleza e funcionalidade, porém, de forma diferente do Windows e do Mac OS, a interface gráfica do Linux não faz parte do seu kernel e pode ser alterada ou customizada. Existem várias interfaces gráficas disponíveis, mas podem ser destacadas:

a) **KDE:** a interface KDE é bastante famosa por ser bastante similar à interface do Windows e, por isso, é indicada aos iniciantes no Linux que buscam uma familiaridade com o sistema da Microsoft. O KDE é uma das interfaces livres mais elegantes e modernas que existem e permite que o usuário adicione aplicativos (widgets) na área de trabalho. A desvantagem dessa interface é que ela exige mais recursos, como memória e processamento;

b) **GNOME**: é uma das interfaces gráficas mais famosas, criada em 1997 e fazia muito sucesso, sendo padrão em muitas distribuições Linux, tais como o Ubuntu, Fedora, Debian, RedHateCentOS, entre outras, porém perdeu um pouco de espaço quando mudou completamente sua interface, e adotou novos ares de modernidade, o que gerou muita desconfiança, pois era reconhecido por ser muito leve, mas com a mudança passou a exigir mais processador e memória e além disso, não é possível customizá-lo como a KDE;

c) **Unity**: Adotada como interface padrão na distribuição Ubuntu, a Unity conquistou muitos adeptos com o passar do tempo, tornando-se uma boa opção ao GNOME 3, e ganhou diversas funcionalidades desde então, com uma interface elegante e com efeitos interessantes. Também exige considerável recurso de memória e processamento e não há tanta liberdade para personalização da interface, mas é

possível adicionar alguns aplicativos e mudar a aparência das janelas e algumas cores do sistema e ícones;

d) **Cinnamon:** Adotada como uma das alternativas de interface para a distribuição Linux Mint, é baseada no GNOME 3, mas é customizada para ser mais similar ao GNOME 2 e é uma alternativa ao GNOME 3, sendo um pouco mais leve, com efeitos bonitos e altamente customizável, com os chamados Desklets, que são uma espécie de widgets, assim como na KDE;

e) **MATEe**: esta interface surgiu alguns meses após o lançamento das primeiras versões do GNOME 3, como uma alternativa a essa nova interface e como uma continuação da interface GNOME 2. Atualmente, MATE é uma interface bem estável, indicada para quem quer uma interface mais familiar e amigável, e para quem gostava do GNOME 2. Ela é geralmente mais leve que GNOME 3, Cinnamon e Unity, mas ainda requer uma quantidade considerável de memória;

f) **XFCE**: O XFCE é uma interface interessante e leve, com o objetivo de ser uma interface para ser executada em computadores com menos recursos. As distribuições que apresentam a interface geralmente vêm com programas que consomem pouca memória RAM e executam em qualquer processador. Sua interface é bem similar ao GNOME 2. Também é altamente customizável, sendo possível acrescentar barras novas, novos itens nas barras e alterar a aparência do sistema e os ícones. Apresenta poucos efeitos visuais em algumas distribuições, o que pode tornar a interface XFCE um pouco mais feia em relação às interfaces KDE, GNOME, Unity ou Cinammon;

g) **LXDE**: O LXDE é uma interface bastante leve, desenvolvida com foco nos computadores com poucos recursos de memória e processador. A interface é tão leve que pode ser executada em computadores com menos de 256MB de memória RAM e com processadores com menos de 500Mhz. A interface é semelhante ao GNOME 2 e não possui grandes efeitos de tela, justamente para priorizar a velocidade, estabilidade e leveza da interface. Entretanto é altamente customizável;

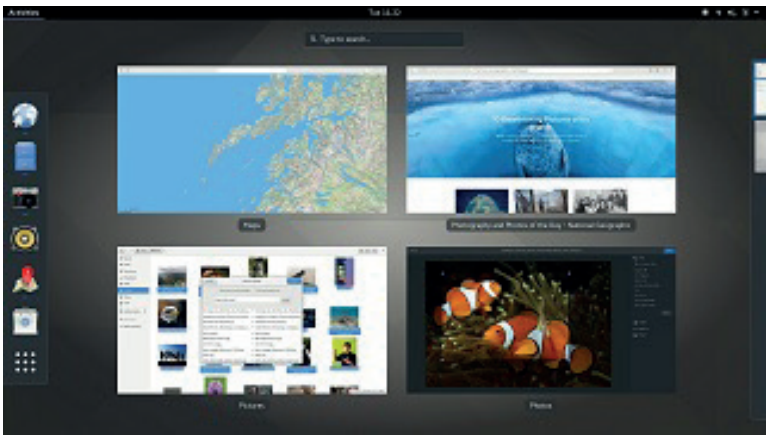
h) **OpenBox**: O OpenBox é uma interface gráfica muito leve, que pode ser incorporada ao GNOME ou ao KDE e é altamente customizável, mas pode ser um pouco complicada pois é diferente dos padrões atuais das interfaces. Sua maior vantagem é poder ser executada em computadores muito antigos, tais como um 486DX com 16 MB de RAM.

As Figuras 26 a 33 apresentam algumas das principais interfaces gráficas para o SO Linux.

Figura 26 – KDE

Fonte: KDE. Disponível em: https://www.kde.org/announcements/4.4/images/general-desktop.jpg

Figura 27 – Gnome 3



Fonte: Gnome. Disponível em: https://www.gnome.org/gnome-3/00

Figura 28 – Unity

Fonte: Wikipédia. Disponível em: https://pt.wikipedia.org/wiki/Linux#/media/File:Ubuntu_15.10_with_Firefox_and_Nautilus_open.png

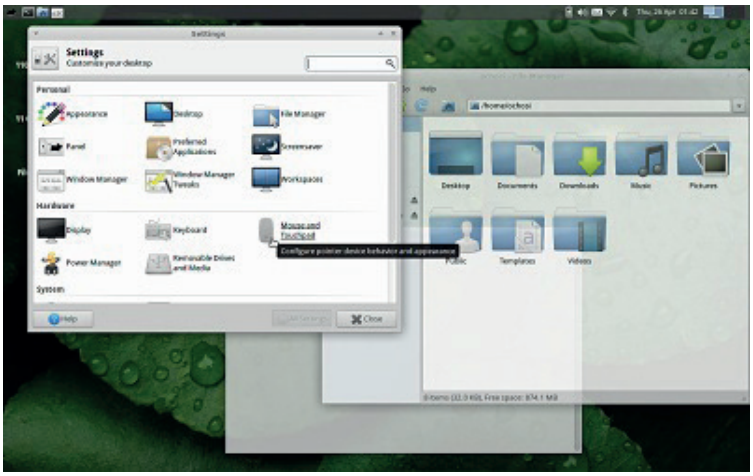
Figura 29 – Cinnamom

Fonte: Cleuber. Disponível em: http://cleuber.com.br/wp-content/uploads/2015/08/linuxmint-cinnamon.png

Figura 30 – Mate

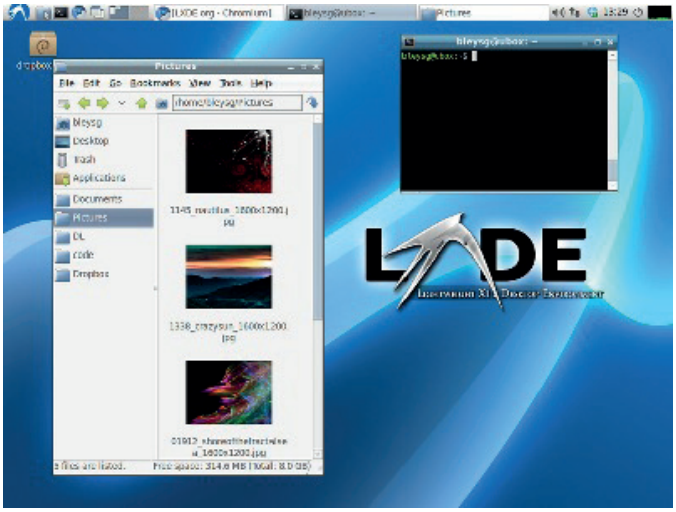
Fonte: LME Linux. Disponível em: http://lmelinux.net/wp-content/uploads/sites/2/2014/11/linux-mint171mate.png

Figura 31 – XFCE



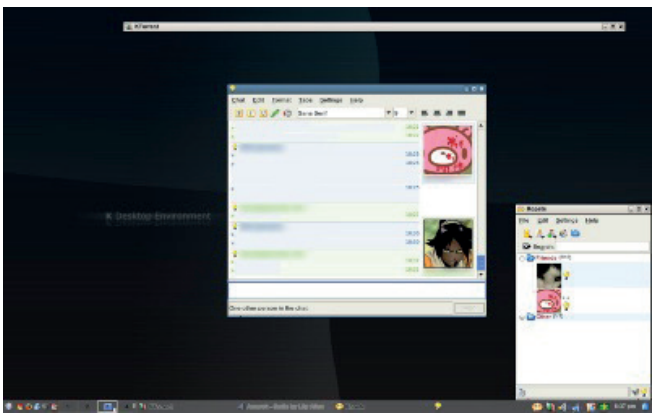
Fonte: Wikipédia. Disponível em: https://pt.wikipedia.org/wiki/Xfce#/media/File:XFCE-4.10-Desktop.png

Figura 32 – LXDE



Fonte: Wikipédia. Disponível em: https://wiki.lxde.org/en/File:LXDE_desktop_full.png

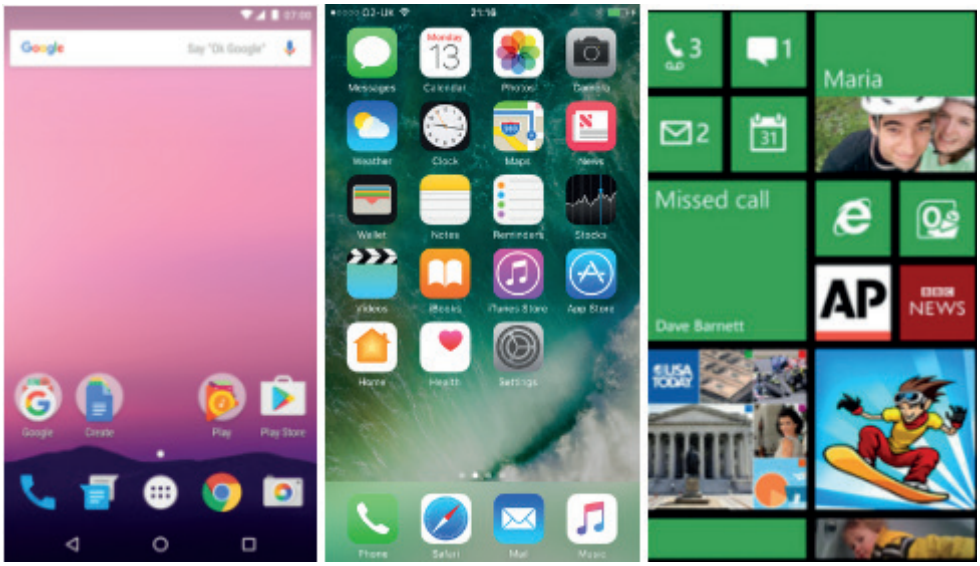
Figura 33 – OpenBox



Fonte: Wikipédia. Disponível em: http://openbox.org/wiki/File:KDE-Openbox.png

Os smartphones também utilizam interfaces amigáveis e com excelentes funcionalidades, utilizando os conceitos de telas touchscreen, que fazem com o que o toque dos dedos permita um acesso de maneira natural e intuitiva. A Figura 34 apresenta as interfaces dos sistemas Android, iOS e Windows Phone.

Figura 34 – Interface do Android, iOS e Windows Phone



Fonte: CDN. Disponível em: http://cdn.bmwblog.com/wp-content/uploads/Device-Info-Windows-Phone.jpeg

2

ESTRUTURA DO SISTEMA OPERACIONAL

INTRODUÇÃO

Agora que você conhece a história da evolução dos computadores e dos SOs, é importante lembrar que um SO não é um programa de computador que controla o hardware pois, se assim fosse, ele seria um "programa operacional". Um SO é formado por um conjunto de programas que gerenciam o hardware e existem inúmeras atividades de gerenciamento que ele proporciona, tanto para os programas quanto para o usuário, sempre com a intenção de facilitar o uso, mas sem abrir mão da confiabilidade e segurança.

Talvez você esteja curioso sobre como o SO faz esse controle? Como eu interajo com o SO e tiro proveito dessas facilidades? Por que eu preciso me preocupar com isso? Caso você tenha estas dúvidas, fique tranquilo, elas serão esclarecidas nesta unidade.

Para responder estas questões, esta unidade foi organizada em seções, onde cada uma delas explica uma parte do funcionamento do SO, suas funções e seus tipos.

2.1
COMO FUNCIONA UM SISTEMA OPERACIONAL

Quando um computador é utilizado, dá a impressão de que é o usuário que está comandando o que vai acontecer, porque o mouse é movimentado na tela, o programa escolhido é executado, o site desejado é acessado, a música escolhida é executada, porém, na verdade quem está realizando todas essas tarefas é o SO do computador, que está interpretando os comandos do usuário e fazendo com que o hardware do computador faça tudo o que for necessário para executar os programas que o usuário precisa.

O SO é o primeiro programa a ser carregado no computador e é responsável por fazer o computador funcionar e permitir a execução dos programas que o usuário necessita. Sem o SO, um computador é inútil para um usuário e somente um especialista com altíssimo conhecimento conseguiria programar um computador para realizar alguma tarefa útil se o mesmo não tivesse um SO. A função de um SO é gerenciar e controlar o hardware e permitir que os mais diferentes programas possam ser executados, gerenciando esses programas e suas interações e mantendo a integridade dos dados.

Nos últimos anos, os SOs passaram a fazer parte de outros dispositivos eletrônicos, como smartphones, roteadores de **Wi-Fi** e Smart TVs. Os computadores usados nesses pequenos dispositivos ficaram tão poderosos que agora podem executar um SO e aplicativos.

TERMO DO GLOSSÁRIO: Wi-Fi: a expressão tornou-se um sinônimo para a tecnologia IEEE 802.11, que permite a conexão entre diversos dispositivos sem fio.

É importante ressaltar que nem todos os dispositivos eletrônicos e computadores possuem SOs. O computador (microcontrolador) que controla o forno de micro-ondas da cozinha, por exemplo, não precisa de um SO, pois ele tem um conjunto bem específico de tarefas para executar, com uma entrada muito direta baseada em um teclado numerado e alguns botões pré-definidos e possui também um hardware simples, que nunca terá alterações. Nesse caso, disponibilizar um SO aumentaria os custos do equipamento e não seria útil, a não ser que esse forno oferecesse diferentes módulos adicionais, como acesso à Internet, para você poder programar e ligar o forno remotamente, por exemplo, deixando uma pizza no forno e, quando saísse do trabalho, poderia ligá-lo por meio de um comando enviado pela Internet, para que, quando chegasse em casa a pizza estivesse pronta. Se este não é o caso, o computador em um forno de micro-ondas tem apenas um único programa de controle que fica o tempo todo esperando um botão ser pressionado para ligar o forno e desligá-lo depois do tempo pré-definido.

Todos os computadores de mesa (desktops) e notebooks possuem SOs, dentre

os quais se destacam o Windows, Linux, MacOS e Chrome OS, da mesma forma que todos os smartphones possuem um SO, dentre os quais os principais são o Android, iOS ou Windows Phone. Até mesmo aqueles celulares bem simples possuem um SO, pois se existe mais de um programa que pode ser usado ao mesmo tempo, como rádio e câmera fotográfica, é necessário um sistema para controlar o seu uso para receber uma ligação telefônica e retornar ao aplicativo que estava em uso.

Em qualquer dispositivo que possua um SO, geralmente há uma maneira de alterar as configurações, instalar aplicativos e atualizar o sistema. Uma das razões pelas quais se justifica utilizar um SO é para que possam ser realizadas alterações e novas instalações em um mesmo dispositivo. Isso quer dizer que em um computador podem ser instalados, além de novos programas, novas placas, novos discos rígidos, novos módulos de memória e até trocar ou atualizar o SO.

Basicamente um SO tem duas funções:

a) **Gerenciar os recursos de hardware**: esses recursos incluem o processador, a memória, o espaço em disco e os dispositivos de entrada e saída, como teclado, tela e conexões de rede além do gerenciamento de energia e bateria, buscando seu uso eficiente e garantindo a integridade e a segurança dos dados armazenados;

b) **Servir de interface entre o hardware e o software com o usuário**: ele fornece uma maneira transparente, estável e consistente para os aplicativos interagirem com o hardware sem ter que conhecer e se preocupar com todos os detalhes do hardware e estender essa funcionalidade por meio de uma interface com o usuário, que efetivamente é quem vai usar os programas e recursos do hardware.

O gerenciamento dos recursos de hardware e software é muito importante, pois vários programas e dispositivos de entrada e saída competem pelo uso do processador (CPU) e da memória de um computador. Nesta tarefa, o SO desempenha o papel para garantir que cada aplicativo obtenha os recursos necessários para sua tarefa, sem prejudicar as demais aplicações.

A segunda tarefa, fornecer uma interface para o usuário usar os aplicativos de forma amigável e consistente, permite que o usuário adquira ou desenvolva programas de acordo com as suas necessidades, e que esses programas possam ser instalados em diferentes computadores, com diferentes configurações, desde que seja para o mesmo SO, pois a função deste é justamente esconder os detalhes e diferenças do hardware e oferecer uma interface padronizada para o usuário e para os programas. É o SO que padroniza a forma como será a interface das aplicações, de modo que todas as aplicações de um SO possuem janelas, menus, botões e caixas de texto com aparência e localização semelhantes. E quem desenvolve essas aplicações não precisa programar esses componentes, apenas fazer uso deles.

Um SO pode garantir que os aplicativos continuem sendo executados mesmo quando ocorrem atualizações de hardware, pois é o SO e não o aplicativo que é o responsável por gerenciar o hardware e a distribuição de seus recursos. Um dos desafios enfrentados pelos desenvolvedores de SOs é manter seus sistemas flexíveis o suficiente para permitir a execução com o hardware dos milhares de fornecedores

que fabricam equipamentos de informática. Os sistemas atuais podem acomodar milhares de impressoras diferentes, unidades de disco e periféricos especiais em qualquer combinação possível.

Quando ligamos um computador, o primeiro programa que é executado geralmente é um conjunto de instruções mantidas na memória somente de leitura do computador (*ReadOnlyMemoryROM*). Este código examina o realiza um autoteste (POST) no hardware do sistema para verificar se está instalado e funcionando o processador (CPU), a memória RAM e a placa de vídeo. Uma vez que o POST tenha concluído com sucesso, o software carregado na ROM, também chamado de BIOS ou UEFI nos computadores mais recentes, começará a ativar as unidades de disco do computador. Na BIOS ou UEFI, está configurada a ordem de ativação dos discos ou unidades, e quando o computador ativa a unidade, no primeiro setor da unidade MBR (Master Boot Record) ou GPT (*GUID PartitionTable*), ele encontra e executa um pequeno programa que carrega o SO. Esse programa encarregado de carregar o SO para a memória é chamado de bootstraploader. Depois de carregado, é o SO que passa a controlar o hardware.

Nos sistemas modernos existem o chamado bootloader de segundo estágio. Por exemplo, o GRUB, é um programa capaz de fornecer ao usuário uma interface na qual ele pode escolher múltiplos boots, ou seja, o usuário pode fazer a escolha de qual SO vai ser inicializado, no caso de ter instalado Windows e Linux, por exemplo, em um mesmo computador.

O processo de boot é um termo abreviado para bootstrap. Assim, é comum associar todo o processo descrito ao bootstrap (ou ao bootstraploader). O importante a ressaltar é que o bootstrap (boot em sentido amplo) não faz parte do SO, mas o bootloader (sendo mais específico) varia de acordo com cada sistema.

O carregador do bootstrap (bootstraploader) é um pequeno programa que possui uma única função: ele carrega o SO na memória e permite que ele comece a operar. Na forma mais básica, o carregador do bootstrap configura os pequenos programas de drivers que interagem e controlam os vários subsistemas de hardware do computador. Ele configura as divisões da memória que possuem o SO, informações e aplicativos do usuário, reconhece o sistema de arquivos da unidade onde está o SO e estabelece as estruturas de dados e controle necessários e, em seguida, passa o controle do computador para o sistema operacional.

Um SO é um conjunto de programas capaz de oferecer, da forma mais transparente possível, os recursos de um sistema computacional aos seus usuários, controlando e organizando o uso dos recursos de maneira que se obtenha um sistema eficiente e seguro.

Os SOs são uma camada de softwareque ”envolve” os componentes de hardware de um computador, intermediando as interações entre estes componentes e os programas dos usuários. Assim, podemos considerar os SOs como sendo uma extensão do próprio computador ou como gerenciadores dos recursos existentes neste computador.

Ao invés de lidar com a complexidade do hardware, o SO oferece a funcionalidade disponível no hardware por meio de uma interface de programação (ApplicationProgram Interface API) orientada à operação de cada tipo de recurso, proporcionando, além da transparência, um isolamento do hardware das aplicações.

2.2
FUNÇÕES DO NÚCLEO DO SISTEMA OPERACIONAL

As tarefas do núcleo (kernel) do SO se enquadram em cinco categorias:

a) **Interface de programação de aplicativos**: As interfaces de programação de aplicativos (APIs) permitem que os programadores de aplicativos usem funções do computador e do SO sem a necessidade de conhecer os detalhes do hardware e os detalhes da operação da CPU. Como o SO esconde os detalhes do hardware para o programador, é por meio da API que um programa tem acesso aos recursos de hardware, tais como unidades de armazenamento e acesso à rede, além de utilizar as rotinas de manipulação das janelas, menus, ícones e botões;

b) **Interface de usuário**: Assim como a API fornece uma maneira consistente para que os aplicativos usem os recursos do sistema de computador, uma interface de usuário (User Interface UI) oferece uma estrutura para a interação entre um usuário e o computador, especialmente com a interface gráfica do usuário (GUI), nativas no Windows e no MacOS e incorporadas ao Linux. Existem outras interfaces de usuário, algumas gráficas e outras não, para outros SOs. O Unix, por exemplo, possui interfaces de usuário chamadas shells, que apresentam uma interface de usuário mais flexível e poderosa do que a interface baseada em texto nativa do SO. Programas como o Korn Shell e o C Shell são interfaces baseadas em texto que adicionam funcionalidades importantes, mas cujo principal objetivo é o de facilitar o uso para manipular as funções do SO. Há também interfaces de usuário gráficas, como X-Windows, Gnome e KDE (vistas na unidade anterior), que tornam o Unix e o Linux mais parecido com os computadores Windows e Macintosh do ponto de vista do usuário. A interface do usuário é um programa, ou um conjunto de programas, que executa como uma camada acima do próprio SO. A diferença como essa camada está vinculada ao núcleo do SO diferencia os sistemas atuais e influencia de maneira significativa o seu desempenho. No Windows e no MacOS a interface gráfica executa no mesmo nível do SO, enquanto que no Linux ele é apenas um programa, como outro qualquer, executando sobre o núcleo, o que gera uma certa perda no desempenho das aplicações gráficas neste sistema;

c) **Gerenciamento do processador**: o gerenciamento de processador ou de processos é a atividade na qual o SO é responsável por atender cada um dos processos, que são os programas em execução, com uma determinada fatia de tempo da CPU. Na prática, todos os processos que estão em execução são colocados em uma fila pelo SO e cada um deles recebe de 20 a 100 milissegundos para executar na CPU. Durante esse tempo, ele pode executar até aproximadamente 40 milhões de instruções ou cálculos e, depois disso, vai para o final da fila, passando a executar o processo que estava na frente na fila. Essa alternância entre os processos que estão

em execução é tão rápida que não percebemos e temos a impressão que todos estão sendo executados ao mesmo tempo, quando na verdade estão sendo alternados de forma muito rápida;

d) **Gerenciamento da memória**: o gerenciamento de memória é a atividade na qual o SO gerencia quais partes da memória serão destinadas a quais processos, garantindo que nenhum processo invada a área de memória de outro processo. Além disso, para proporcionar que mais memória seja usada, o SO carrega para a memória principal somente o que está em uso, armazenando o restante temporariamente no disco rígido, em um processo denominado swaping, carregando e retirando da memória sempre que necessário. Alguns SOs como o Mac OS e o Windows 10 também realizam um processo de compactação da memória das partes que não estão em uso;

e) **Gerenciamento de dispositivos de entrada e saída**: o gerenciamento de Entrada e Saída é encarregado de realizar a comunicação do SO com dispositivos de entrada e saída, por meio dos drivers de dispositivos, fornecendo uma forma padronizada para os programas solicitarem os acessos a esses dispositivos. Os drivers, por sua vez, enviam os dados para os dispositivos propriamente ditos. Devido a isso, quando compramos uma impressora nova, ela só funciona corretamente depois de instalarmos os drivers adequados. O programa envia o comando para o SO realizar a impressão. Porém, como o SO não conhece os detalhes de como a impressora funciona, ele envia a requisição para o driver da impressora, que é um programa que "entende" o que o SO vai pedir e traduz com os comandos adequados para a impressora. Além disso, o SO também precisa gerenciar o armazenamento dos dados nos discos rígidos (HD), pendrives e unidades de estado sólido (SSD), por meio de um sistema de arquivos.

A Figura 35 apresenta a estrutura do SO como intermediário entre o hardware, os programas (aplicativos) e o usuário.

Figura 35 – Estrutura do Sistema Operacional

Fonte: Autores, 2017.

2.3
SERVIÇOS DO SISTEMA OPERACIONAL

Outro enfoque que pode ser dado é considerar o SO como um prestador de serviços, pois ele fornece um ambiente para a execução de programas, ou seja, fornece certos serviços aos programas e aos usuários destes programas. Estes serviços podem ser:

a) **Execução de programa**: o SO deve ser capaz de carregar um programa na memória e executar esse programa, além de ser capaz de encerrar a sua execução, de forma normal ou anormal, com indicação de erro;

b) **Operação de Entrada e Saída**: o SO deve fornecer os meios para realizar as operações de entrada e saída, pois certamente um programa em execução irá precisar de E/S;

c) **Manipulação do sistema de arquivos**: o SO deve fornecer as funcionalidades de acesso ao sistema de arquivos, pois os programas precisam ler e gravar dados, criando e excluindo arquivos nas unidades de armazenamento;

d) **Comunicação**: o SO precisa fornecer mecanismos de comunicação entre processos, pois existem várias situações nas quais um processo precisa trocar informações com outro processo. Essa comunicação pode ser com processos em um mesmo computador ou entre processos que estão executando em diferentes sistemas de computação interconectados por uma rede;

e) **Detecção de erros**: o SO precisa ficar monitorando constantemente a ocorrência de erros, que podem ocorrer no hardware da CPU e da memória, em dispositivos de E/S ou no programa de usuário. Para cada tipo de erro o SO deve tomar a medida adequada para garantir o funcionamento correto e consistente e não simplesmente travar;

f) **Alocação de recursos**: os recursos como a memória e os dispositivos de E/S devem ser alocados de maneira coerente e eficiente para cada um dos processos que estão em execução simultânea;

g) **Contabilização**: é preciso manter um registro dos usuários que utilizam os recursos do computador, em que quantidade e quais recursos. Este registro pode ser usado para contabilização e para realocação mais eficiente de cada recurso;

h) **Proteção**: quando vários processos independentes estão sendo executados ao mesmo tempo, um processo não poderá interferir na área de outro processo ou do próprio SO. A proteção visa garantir que todo acesso aos recursos do sistema seja controlado.

Os programas solicitam serviços ao SO por meio das chamadas de sistema. Através de parâmetros, o programa informa exatamente o que necessita e o SO se encarrega de executar a tarefa e devolver o retorno.

Por exemplo, qualquer função de uma linguagem de programação que escreva algo na tela, exige uma chamada de sistema. A lista de serviços do SO é transformada em uma lista de chamadas de sistema, cujas descrições dessas chamadas são transcritas nos manuais de um SO.

A parte do SO responsável por implementar as chamadas de sistema é normalmente chamada de núcleo ou kernel. Os principais componentes do kernel de qualquer SO envolvem as operações de gerência de processador, a gerência de memória, o sistema de arquivos e a gerência de entrada e saída.

Devido à complexidade interna de um **kernel** completo, alguns SOs são implementados em camadas com um **micronúcleo** ou **microkernel** implementando os serviços mais básicos de comunicação e os demais serviços são implementados sobre esse microkernel.

O Mac OS X e o Windows são implementados com a ideia de microkernel, com um micronúcleo, porém com uma forte influência da organização em camadas, porém, com os demais serviços do SO também implementados no contexto do micronúcleo, ao contrário da idéia do microkernel que prevê que todos os demais gerenciamentos são programas convencionais executando sobre o microkernel.

Ao contrário do microkernel, existem os sistemas **monolíticos** onde todas as funções de um núcleo de SO são implementadas no núcleo. O Linux é um exemplo desse tipo de sistema.

No contexto do usuário, os programas de sistema, também chamados de utilitários, são programas normais executados fora do kernel do SO. Eles utilizam as mesmas chamadas de sistema disponíveis aos demais programas e implementam as tarefas básicas para a utilização do sistema e muitas vezes são confundidos com o próprio SO. Exemplos desses programas são os utilitários para manipulação de arquivos, programas de sistema para a obtenção de informações a respeito do sistema, tais como data, hora ou quais usuários estão utilizando o computador no momento, a relação de programas em execução e os recursos que os mesmos estão consumindo, um interpretador de comandos e, em alguns sistemas como Windows, Mac OS X, Android e iOs, uma interface gráfica integrada ao SO.

2.4
RECURSOS E AMBIENTE OPERACIONAL

A parte física do computador (hardware) determina suas capacidades brutas de processamento e seus limites. Todos os elementos funcionais do computador são considerados recursos do sistema computacional. Os recursos podem ser o teclado, mouse, monitores de vídeo, placas de rede ou comunicação, impressoras, scanners, unidades de disco rígido, unidades leitoras/gravadoras de CD, DVDs etc.

O SO aparece como uma camada sobre o hardware e **firmware**, envolvendo-os e escondendo seus detalhes e servindo de interface para eles. Desta forma o SO, por meio de sua interface, define uma nova máquina que é a combinação de um certo hardware com este SO. O conjunto de hardware e SO é usualmente chamado de **plataforma** ou **ambiente operacional**.

TERMO DO GLOSSÁRIO: Firmware: é um conjunto de instruções operacionais que são programadas diretamente no hardware de equipamentos e dispositivos eletrônicos e fazem com que os mesmos funcionem. São armazenados diretamente num circuito integrado (chip) de memória de hardware, como uma ROM, EEPROM ou em memória flash, para permitir atualização.

O ambiente operacional é distinto do hardware, pois o hardware do computador por si só não é capaz de copiar um determinado arquivo de uma unidade de disco rígido para uma unidade de pendrive, por exemplo. Para realizar esta cópia, uma série procedimentos devem ser executados, desde o acionamento das unidades utilizadas, passando pela localização das partes do arquivo que vai ser copiado e das áreas disponíveis no pendrive de destino, até a transferência efetivada dados.

Por meio da interface do SO, ações como esta podem ser possíveis através de um comando fornecido através de um console (Prompt de comando ou terminal) ou através de uma interface gráfica com uma operação visual, frequentemente denominada arrastar e soltar (draganddrop), realizadas com o mouse.

Enquanto o hardware isolado não permite a realização direta da tarefa de copiar um arquivo, por meio do SO temos a impressão de que o computador se tornou capaz de realizar a tarefa desejada através do comando dado, ou da operação visual efetuada, sem a necessidade do conhecimento de detalhes de como a tarefa é verdadeiramente realizada. O hardware é realmente o realizador da tarefa, enquanto o SO apenas intermedia esta operação através de sua interface.

A disponibilização de uma interface gráfica para o usuário facilita o uso por parte de usuários com menos conhecimentos técnicos da área, enquanto que a utilização em modo de comando exige um alto grau de conhecimento, pois cada comando exige uma variação grande de parâmetros que um usuário convencional tem difi-

culdade de usar e há um risco maior da ocorrência de erros. Com essas interfaces amigáveis, o SO transforma o hardware de um computador em um determinado ambiente operacional, de forma transparente.

2.5
TIPOS DE SISTEMAS OPERACIONAIS

Existem inúmeros tipos de SOs, mas eles podem ser categorizados de acordo com os tipos de aplicativos e usuários que eles suportam. Os principais são:

a) **Sistema operacional de tempo real**: sistemas operacionais de tempo real são usados para controlar máquinas, instrumentos científicos e sistemas industriais. Um sistema desse tipo normalmente possui muito pouca capacidade de interface do usuário e nenhum utilitário de usuário final, uma vez que o sistema atenderá uma única aplicação e não pode ser interrompido. O detalhe mais importante é que esse sistema precisa gerenciar os recursos do computador para que uma determinada operação seja executada precisamente na mesma quantidade de tempo, sempre que ocorrer;

b) **Monousuário, monotarefa**: este tipo de SO foi projetado para gerenciar o computador para que um único usuário possa, efetivamente, fazer uma coisa por vez. O DOS nos primeiros computadores pessoais e o Palm OS para PDAs é um bom exemplo de um SO de uma única tarefa;

c) **Monousuário, multitarefa**: este é o tipo de SO que a maioria das pessoas usa em seus computadores desktop e smartphones atualmente. O Windows da Microsoft e as plataformas MacOS da Apple são exemplos de SOs que permitem que um único usuário tenha vários programas em operação ao mesmo tempo;

d) **Multiusuário**: um SO multiusuário permite que vários usuários aproveitem simultaneamente os recursos do computador. O SO deve garantir que os recursos usados pelos diferentes usuários sejam equilibrados e que cada um dos programas, que eles estão usando, tenha recursos suficientes e separados, para que um problema com um usuário não afete os demais e o funcionamento do sistema. Nos SOs Unixe as versões para servidores do Windows e Linux são exemplos de SOs multiusuários, apesar de que as versões domésticas do Windows e do MacOS atualmente podem ser usadas por mais de um usuário ao mesmo tempo;

e) **Distribuídos**: um SO distribuído faz uso de diferentes computadores interconectados por uma rede de comunicação, fazendo com que os mesmos funcionem como se fossem um só. A computação em nuvem é baseada nesse modelo de sistema, onde os servidores, tais como os do Google e do Facebook estão espalhados pelo mundo, mas enxergamos como se tivesse um só.

Os principais aspectos que diferem os tipos de SOs se referem ao propósito para o qual o mesmo foi desenvolvido. Assim, o tipo de um SO é classificado de acordo com os seguintes aspectos:

a) **Interatividade**: é o aspecto que considera se o usuário utiliza diretamente o sistema computacional, podendo receber as respostas deste, sem intermediação e dentro de intervalos de tempo razoáveis;

b) **Tempo de resposta**: é uma medida de interatividade, que representa o intervalo de tempo decorrido entre um pedido ou solicitação de processamento e a resposta produzida pelo sistema. Tempos de resposta da ordem de alguns milissegundos a no máximo alguns poucos segundos configuram sistemas interativos;

c) **Tempo de reação**: é outra medida de interatividade, a qual considera o tempo decorrido entre a solicitação de uma ação e seu efetivo processamento;

d) **Produtividade *(throughput)***: é uma medida de trabalho relativa do sistema, expressa usualmente em tarefas completas por unidade de tempo, ou seja, é uma medida que relaciona o trabalho efetivamente produzido e o tempo utilizado para realização deste trabalho. A produtividade não deve ser confundida com o desempenho bruto do processador do sistema e sua capacidade de processamento, pois depende da arquitetura do sistema e do SO o quanto desta capacidade é efetivamente convertida em trabalho útil e o quanto é despendida nas tarefas de controle e gerência do próprio sistema computacional.

3

GERÊNCIA DE PROCESSOS

INTRODUÇÃO

Diversas tarefas cotidianas executadas por meio do nosso computador estão relacionadas com a gerência de processos. Todavia, talvez você nunca tenha se perguntado o real motivo da existência deste conceito. Para responder esta questão, devemos nos lembrar que os primeiros computadores projetados para operarem usando um SO eram monousuário e monotarefa. Isto quer dizer que apenas um usuário por vez poderia usar o computador e que as tarefas eram executadas uma de cada vez. Como talvez você já tenha percebido, esta abordagem gerava diversos problemas, sendo o principal deles a ociosidade da CPU. Este fato ocorria quando o programa precisava executar operações de entrada/saída, por exemplo, a leitura ou escrita de dados em disco. Esta operação não demanda a execução de operações lógicas e aritméticas desempenhadas pela CPU, ocasionando a ociosidade deste componente.

Para entendermos a gravidade deste problema, podemos fazer uma analogia com o mundo empresarial. Este tipo de ambiente envolve um grande número de funcionários, todos eles contratados para desempenhar um conjunto pré-determinado de tarefas. Vale a pena salientar que cada um destes funcionários possui um custo associado para sua manutenção, por exemplo, o seu salário e a taxa de contribuição do Instituto Nacional de Seguridade Social (INSS). Do ponto de vista do empregador, cada empregado deve passar a maior parte do seu tempo trabalhando para gerar lucro para empresa, caso contrário, a manutenção deste funcionário gerará um prejuízo para a empresa.

O mesmo acontece no mundo dos computadores. Cada componente do computador possui um custo de aquisição, sendo o processador, ou CPU, um dos componentes mais caros. Assim, da mesma forma como acontece no mundo empresarial, precisamos que os componentes do computador, em especial a CPU, passe a maior parte do tempo ocupada para obtermos a melhor relação de custo/benefício do nosso ambiente computacional. Entretanto, os primeiros SOs projetados para operarem como monotarefa e monousuário não se preocupavam com estas questões.

Para contornar estas limitações, foram projetados os SOs com suporte a multiusuários e multitarefas. Diferentemente dos seus predecessores, estes sistemas tinham como objetivo manter a CPU ocupada a maior parte do tempo, visando proporcionar um melhor desempenho para seus usuários. Todavia, surgiu um desafio a ser solucionado, como gerenciar a utilização da CPU? Pois, do ponto de vista deste componente, não existia distinção entre as tarefas que lhe eram delegadas. A

solução para este questionamento, foi basicamente associar um programa com um processo. Neste momento você deve estar se perguntando: mas o que exatamente é um processo? Esta pergunta pode ser respondida por meio de uma perspectiva geral ou completa.

Por meio da perspectiva geral, podemos dizer que um processo consiste em um programa em execução. Utilizando a perspectiva completa, podemos definir o processo como o conjunto de informações necessárias para que o SO implemente a gerência da utilização do processador.

A implementação desta nova abordagem impactou na criação de um novo termo, a concorrência de processos. A concorrência de processos caracteriza a disputa entre os processos para acessar os recursos da CPU. A gerência de processos controla a concorrência dos processos e consiste em uma das principais funções do SO. Ela possibilita alocar recursos, compartilhar dados, trocar informações e sincronizar suas execuções. O SO implementa a concorrência entre processos, controlando como a CPU é acessada. Assim, o conceito de processo é essencial para implementar o conceito de concorrência de processos.

Quando estamos utilizando um computador com um ou mais programas ao mesmo tempo, muitas vezes, o mesmo não apresenta o desempenho que desejamos. Neste contexto, nesta unidade serão apresentados os conceitos para poder monitorar os recursos que cada programa utiliza e verificar se a lentidão é decorrente do uso excessivo de CPU por parte de um programa, ou se é por uso excessivo de memória. Se for devido a uma dessas razões, deve-se pensar em trocar o processador ou adicionar mais memória no computador.

Sempre é bom lembrar que, quando usamos um computador e, se tudo estiver funcionando como queremos, nem precisamos nos lembrar que existe um SO por baixo proporcionando tudo isso. Entretanto, se o computador estiver apresentando lentidão e travamentos, precisamos conhecer os detalhes para saber qual parte está sobrecarregada.

Nesta unidade você aprenderá os principais conceitos sobre a gerência de processos. Visando proporcionar um aprendizado mais eficaz, sempre que possível os conceitos sobre a gerência de processos serão demonstrados usando o **simulador** de sistemas operacionais (**SOSim**), o SO Windows ou o SO Linux. As subunidades a seguir apresentarão a estrutura, os estados e os tipos de processos. Na sequência são apresentados os estudos de caso sobre a gerência de processos usando os SOs Windows e Linux.

TERMO DO GLOSSÁRIO: A simulação consiste em uma técnica usada para caracterizar as principais propriedades de um ambiente e representá-las através de um software (JAIN, 1991).

INTERATIVIDADE: Para obter o simulador SOSim acesse o site www.training.com.br/sosim/

3.1
GERÊNCIA DE PROCESSOS

A gerência de processos, ou como também é chamada de gerenciamento do processador, envolve a parte do SO responsável por escolher qual processo ou programa vai utilizar o processador e por quanto tempo. A gerência de processos envolve dois problemas relacionados:

- Garantir que cada processo e aplicativo recebam tempo do processador para funcionar corretamente;
- Usar tantos ciclos de processador quanto possível para processamento efetivo.

Um processo ou task é um programa em execução. O aplicativo que um usuário executa, como um processador de texto, uma planilha ou um jogo, é, de fato, um processo, porém esse aplicativo pode fazer com que outros processos sejam executados para realizar tarefas como comunicações com outros dispositivos ou outros computadores.

Por exemplo, quando acessamos uma página web o browser, que é um processo, faz um pedido para o servidor web para enviar a página que estamos acessando; ao receber o código HTML, começa a exibir a página e, para cada imagem que deve compô-la, cria um novo subprocesso, chamado de thread, para fazer a requisição da imagem que vai compor a página. Desta forma, enquanto a página ainda está sendo carregada, vários subprocessos (threads) vão trazendo, simultaneamente, as imagens que compõem a página web. Assim, o browser é composto de um processo principal e vários outros subprocessos, criados para atender necessidades momentâneas e que são executados simultaneamente.

Na verdade, simultaneamente só podem ser executados tantos processos quanto o número de núcleos (cores) que o processador possui, normalmente 2 ou 4. Os demais processos, são escalonados de forma tão rápida, que nem percebemos que eles estão sendo alternados na sua execução. Cada processo executa aproximadamente de 28 a 100 milissegundos.

Um processo, então, é um software que realiza alguma ação e pode ser controlado - por um usuário, por outras aplicações ou pelo SO. São os processos, e não os aplicativos, que o SO controla e faz o agendamento e escalonamento para a execução pela CPU. Em um sistema de tarefas únicas, o cronograma é direto. O SO permite que o aplicativo comece a funcionar, suspendendo a execução apenas o tempo suficiente para lidar com interrupções e entrada do usuário.

As interrupções são sinais especiais enviados pelo hardware ou software para a CPU. É como se alguma parte do computador subitamente levantasse a mão para chamar a atenção da CPU durante a execução do programa. Às vezes, o SO agendará a prioridade dos processos para que as interrupções sejam mascaradas - ou seja,

o SO ignorará as interrupções de algumas fontes para que um trabalho específico possa ser concluído o mais rápido possível. Existem algumas interrupções, como as de problemas de erro ou problemas de memória, que são tão importantes que não podem ser ignoradas. Essas interrupções não-mascaráveis (NMIs) devem ser tratadas imediatamente, independentemente das outras tarefas em mãos.

A fim de dar a impressão de que vários processos estão sendo executados simultaneamente, o SO deve alternar entre diferentes processos, milhares de vezes por segundo. Veja como acontece:

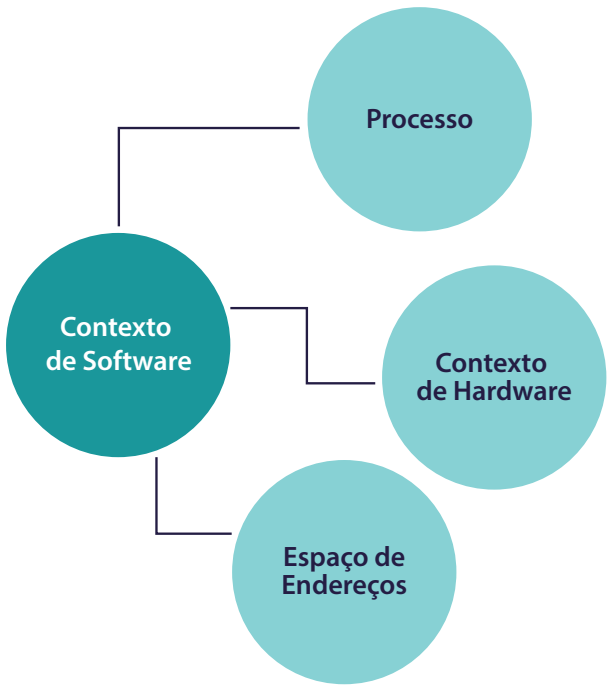
- Um processo ocupa uma certa quantidade de memória RAM. Ele também faz uso de registros, pilhas e filas dentro da CPU e do espaço de memória do SO;
- Quando dois processos são multitarefa, o SO atribui um certo número de ciclos de execução da CPU a cada programa;
- Após esse número de ciclos, o SO faz cópias de todos os registradores, pilhas e filas usadas pelos processos e observa o ponto em que o processo pausou em sua execução;
- Em seguida, carrega todos os registros, pilhas e filas usados pelo segundo processo e permite um certo número de ciclos de CPU;
- Quando esses são completos, ele faz cópias de todos os registros, pilhas e filas usadas pelo segundo programa e carrega o primeiro programa.

3.2
ESTRUTURA DE UM PROCESSO

Um processo é formado por três partes, o **contexto de hardware**, o **contexto de software** e o **espaço de endereçamento**. Quando o processo está em execução, o contexto de hardware está armazenado nos registradores do processador. No momento que o processo perde a utilização da CPU, o sistema salva as informações no contexto de hardware do processo. O contexto de hardware é fundamental para a implementação de sistemas multiprogramáveis, pois os processos podem ser interrompidos e restaurados. A figura 36 ilustra a estrutura do processo.

SAIBA MAIS: O contexto de hardware armazena o conteúdo dos registradores de uso específico, por exemplo, o program counter (PC), stack pointer (SP) e o registrador de status.

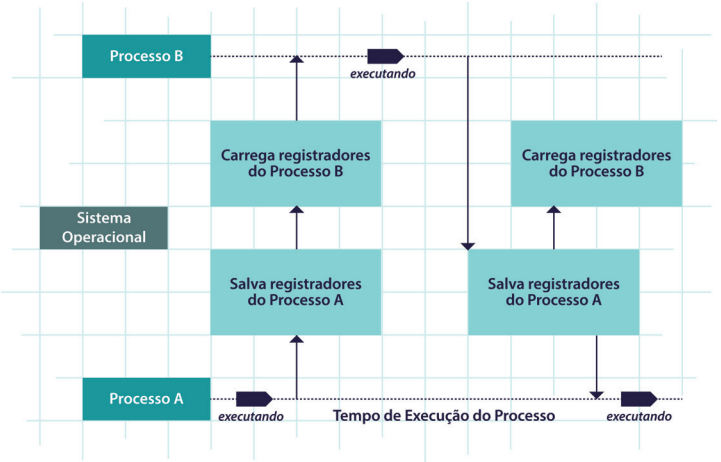
Figura 36 - Estrutura de um Processo



Fonte: Autores, 2017.

O SO gerencia as mudanças de contexto para implementar a concorrência ao salvar o conteúdo dos registradores do processo que está deixando a CPU e carrega os valores referentes ao novo processo que será executado. A figura 37 ilustra um exemplo deste conceito considerando dois processos.

Figura 37 -Mudança de Contexto

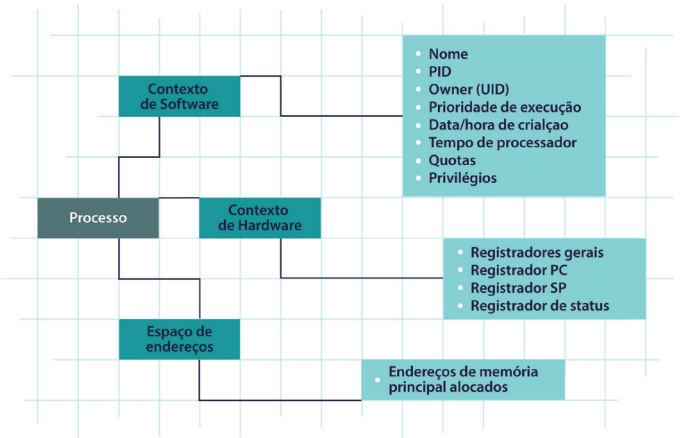


Fonte: Autores, 2017.

O **contexto de software** especifica limites e características dos recursos que podem ser alocados pelo processo, tais como o número máximo de arquivos abertos simultaneamente pelo processo, a prioridade de execução, o tamanho do buffer para operações de E/S. Muitas destas características são determinadas na criação do processo e outras são determinadas durante a existência do processo. A figura 38 ilustra a estrutura do processo considerando as informações específicas para o contexto de software, o contexto de hardware e o espaço de endereçamento.

ATENÇÃO: O contexto de software é composto por três grupos de informação sobre o processo: identificação, quotas e privilégios

Figura 38 - Características da Estrutura de um Processo



Fonte: Autores, 2017.

O grupo de identificação descreve o identifier (ID), o dono e UserIdentifier (UID). O ID consiste em um número inteiro único que possibilita referenciar o processo para consultar seu contexto e alterar suas características. O dono, ou owner, des-

creve o responsável pelo processo. O UID consiste no ID do usuário. Através da combinação do ID, UID e dono, podem ser implementados modelos de segurança para acessar os processos.

O grupo de quotas descreve os limites de cada recurso que um processo pode usar. Quando este recurso se esgota o processo pode ficar lento, pode ser interrompido ou até mesmo não ser executado. Dentre os exemplos de quotas podemos citar o tamanho máximo de memória principal e secundária alocada por processo e o número máximo de operações de E/S pendentes.

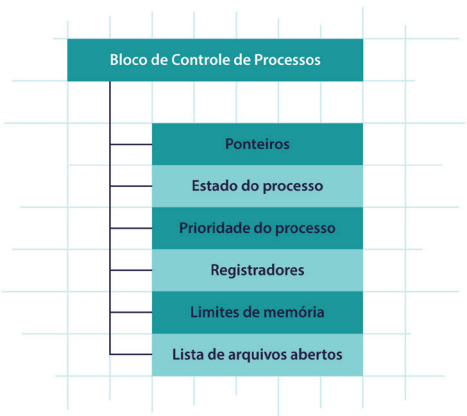
O grupo de privilégio define as ações que um processo pode fazer. Este grupo pode ser dividido em dois tipos: (1) o que pode ser feito em relação a si mesmo; e (2) o que pode ser feito em relação aos demais processos. Exemplos de ações que um processo pode fazer em relação a si mesmo consiste na possibilidade de alteração de suas características, prioridade de execução e limites alocados na memória principal e secundária. Exemplos de ações que um processo pode executar em relação aos demais consistem na possibilidade de alterar as características dos demais processos do SO.

O espaço de endereçamento consiste na área de memória pertencente ao processo onde as instruções e dados do programa são armazenados para execução. Cada processo possui seu próprio espaço de endereçamento, o qual deve ser protegido do acesso dos demais processos.

Para implementar o processo no SO, torna-se necessário empregar uma **estrutura de dados** para representar o contexto de software, o contexto de hardware e o espaço de endereçamento. A estrutura de dados usada para estas situações consiste no **bloco de controle de processos (PCB)**, também chamado de tabela de processos. Os PCBs dos processos residem na memória principal e o SO determina o espaço dos PCBs, armazenando nessa tabela todos os dados e recursos necessários para o gerenciamento dos processos. A figura 39 ilustra a estrutura de dados da PCB.

ATENÇÃO: A partir do PCB o sistema operacional mantém todas as informações sobre o contexto de hardware, o contexto de software e espaço de endereçamento

Figura 39 - Estrutura de Dados do Bloco de Controle de Processos

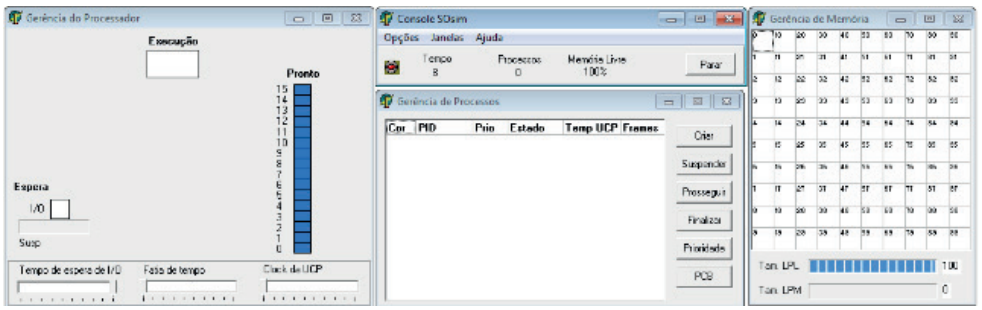


Fonte: Adaptação de Machado, 2014.

Os processos desempenham um papel fundamental para gerenciar a concorrência aos recursos físicos de um sistema computacional. Esta concorrência é implementada em termos de software e baseia-se nas informações associadas com cada processo. Como foi visto anteriormente, as informações sobre os processos são organizadas em três grupos, o contexto de hardware, o contexto de software e espaço de endereçamento. Para representar estas informações é utilizado o bloco de controle de processos PCB.

Podemos usar o simulador SOSim para exemplificar a aplicação do contexto de hardware e software. Para fazer isto, primeiramente execute o simulador clicando duas vezes sobre o arquivo executável principal. Após a inicialização, você verá que quatro janelas serão abertas na inicialização. A figura 40 ilustra esta etapa.

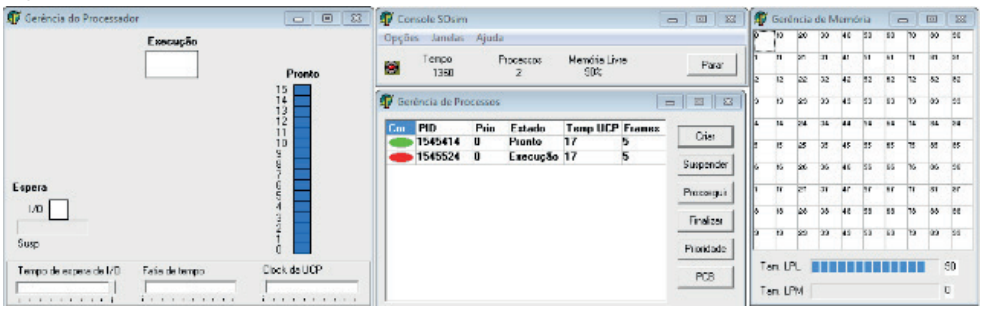
Figura 40 - Tela Inicial do Simulador de Sistemas Operacionais SOSim



Fonte: Autores, 2017.

Em seguida, crie dois processos. Para isto escolha a janela de gerência de processos, clique em Criar, deixe as opções previamente carregadas e mude o número de processos para 2. Por enquanto não se preocupe em entender todas essas opções, no decorrer desta unidade estudaremos cada uma delas. Após esta configuração clique em OK. Seguindo estes passos você deverá receber uma saída similar à ilustrada na figura.

Figura 41 - Criação de Processos no Simulador SOSim



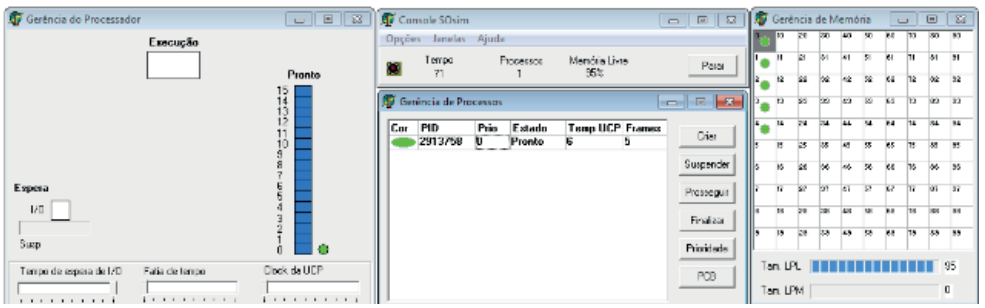
Fonte: Autores, 2017.

Na Figura 41 observamos que existem informações dinâmicas sobre os processos, ou seja, que variam constantemente, e outras estáticas, isto é, permanecem inalteradas. Dentre as informações dinâmicas encontramos o estado e o tempo de UCP (ou CPU). As informações estáticas consistem no PID, na prioridade e frames. As informações dinâmicas, neste caso, pertencem ao contexto de software. Apesar

de não variar neste exemplo, a prioridade também pertence ao contexto de software. As informações referentes ao contexto de hardware, neste caso, consistem ao campo Frame.

O simulador SOSim também permite visualizar as informações da PCB. Para ver estas informações, primeiramente você precisa criar um processo. Para isto reinicie o simulador e crie um novo processo. A figura 42 mostra a tela do simulador após a criação do processo.

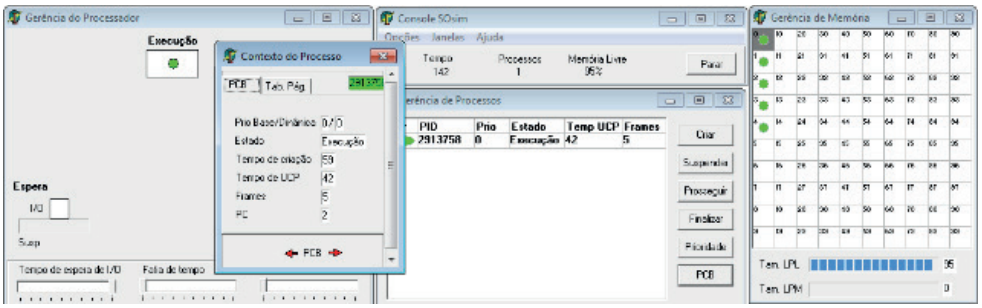
Figura 42 - Passo 1 para Obtenção das Informações de PCB do Processo



Fonte: Autores, 2017.

Na sequência, clique na opção PCB na janela de Gerência de Processos. A figura 43 ilustra a tela do simulador após a execução deste passo. Após seguir este procedimento, será apresentada uma nova janela descrevendo o contexto do processo. Nesta janela, escolha a opção PCB. Dentre as opções desta aba será apresentada a faixa de prioridade base e dinâmica, o estado, o tempo de criação, o tempo de CPU, os frames e o PC.

Figura 43 - Passo 2 para Obtenção das Informações de PCB do Processo



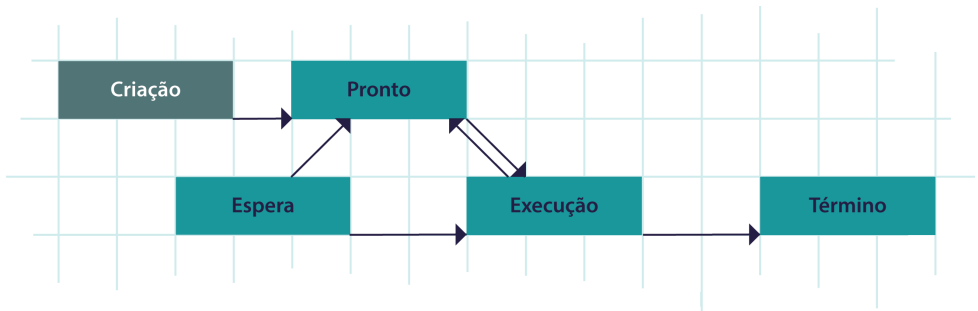
Fonte: Autores, 2017.

3.3
ESTADOS DO PROCESSOS

Os **processos** passam por diferentes estados ao longo do seu processamento. O estado execução ocorre quando o processo está em andamento. O estado pronto ocorre quando o processo aguarda para ser executado. O estado de espera acontece quando o processo aguarda algum evento externo, ou por algum recurso, para ser executado. Além destes três estados, o sistema operacional pode adicionar o estado de criação e término. O estado de criação ocorre quando o SO criou um novo PCB, porém não pode colocá-lo como pronto. O estado de término ocorre quando o SO necessita desalocar os recursos associados com um processo e finalizá-lo. A figura 44 ilustra as possíveis mudanças de estado dos processos.

ATENÇÃO: Um processo pode encontrar-se em três estados: execução, pronto ou espera.

Figura 44 - Mudanças de Estado do Processo



Fonte: Autores, 2017.

Os estados de um processo podem mudar ao longo do seu processamento devido à ocorrência de eventos. Estes eventos podem ser classificados como voluntários e involuntários. Os eventos **voluntários** são gerados pelo próprio processo. Os eventos **involuntários** são gerados pelo SO. Existem quatro mudanças de estado dos processos, sendo eles: **Criação para Pronto; Pronto para Execução; Execução para Espera; Execução para Pronto; Execução para Término**; e **Espera para Pronto**. Após a Criação do processo ele passa para o estado Pronto. Em seguida, o processo passa de Pronto para Execução para ser processado pela CPU. Depois disto, o processo pode passar do estado em Execução para um dos estados: Espera, Pronto ou Término. Um detalhe importante é que, mesmo quando for configurado para o estado de Término, o processo será de fato finalizado quando ele for processado, ou seja, quando ele entrar em execução. Por meio destas mudanças de estados os processos concorrem pela utilização dos recursos físicos do sistema distribuído e proporcionam ao usuário uma ilusão de que todos os seus programas são executados ao mesmo tempo, enquanto que na verdade todos estão compartilhando os recursos.

Os processos podem compartilhar recursos em termos de tempo e espaço. No compartilhamento em termo de tempo, cada processo do sistema acessa uma fatia

de tempo que pode variar de acordo com sua prioridade. Por exemplo, o compartilhamento por recursos de processamento. No compartilhamento em termos de espaço, cada processo possui acesso a uma parte de uma área de armazenamento. Por exemplo, o compartilhamento da memória principal, onde cada processo fica armazenado em uma parte deste recurso.

Para ilustrar estes conceitos, repita a simulação anterior e observe as variações entre os estados dos processos. Explore as opções Suspender, Prosseguir, Finalizar e Prioridade.

3.4
TIPOS DE PROCESSOS

Os processos podem ser classificados de acordo com a forma como utilizam o processador. Seguindo esta taxonomia, temos dois tipos principais, os processos **CPU-bound** e I/O-bound. Nos processos CPU-bound, a principal característica consiste em fazer poucas operações de leitura e escrita. Estes processos são comumente encontrados em aplicações científicas e estão estreitamente associados com a execução de cálculos complexos.

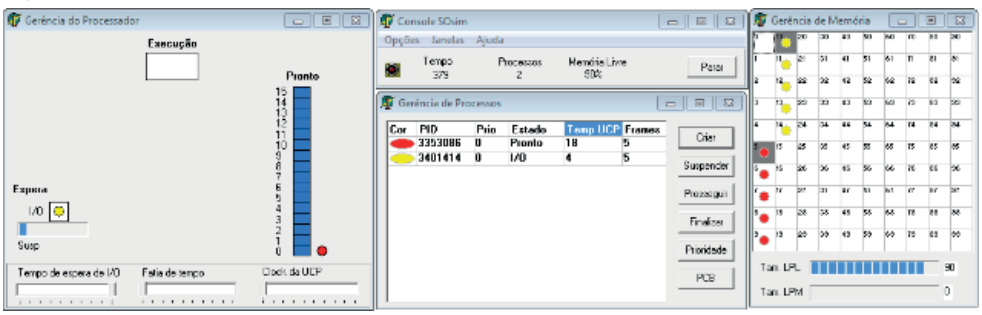
SAIBA MAIS: Os processos CPU-bound têm a maior parte das instruções exigem processamento da CPU e passam a maior parte do tempo nos estados em execução ou pronto (na fila de espera para usar a CPU).

Em contrapartida, os processos **I/O-bound** realizam um elevado número de operações de entrada/saída, por exemplo, lendo e escrevendo dados em discos e imprimindo resultados para os usuários. Os processos I/O-bound são normalmente encontrados em aplicações comerciais, por exemplo, os sistemas de controle de estoque ou sistemas de pagamentos.

SAIBA MAIS: Os processos I/O-bound são aqueles que têm a maior parte das instruções que exigem operações de entrada e saída e passam a maior parte do tempo em estado de espera (bloqueados), aguardando a operação de entrada e saída ficar pronta.

Primeiramente, reinicie o simulador. Em seguida, crie um processo do tipo CPU-Bound usando a janela gerência de processos. Clique em Criar e selecione o tipo de processo CPU-Bound. Na sequência, siga as mesmas instruções para criar outro processo, mas selecione o tipo I/O-Bound. Seguindo estes passos você terá como resultado algo semelhante ao ilustrado na figura 45.

Figura 45 - Simulação Envolvendo Processos CPU-Bound e I/O Bound



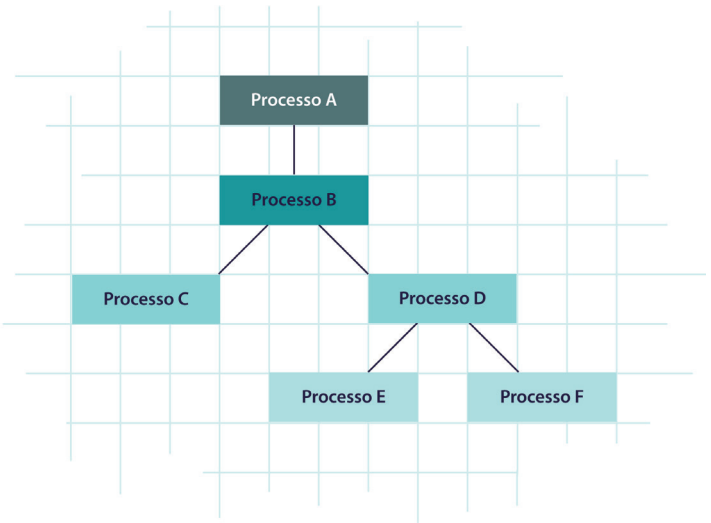
Fonte: Autores, 2017.

Outra forma de classificar os processos consiste na maneira como eles são executados. Seguindo esta abordagem, podemos agrupar os processos como foreground e

background. Os processos foreground permitem a comunicação direta do usuário com o processo durante seu processamento. Por exemplo, quando um usuário executa um programa diretamente por meio de um terminal, geralmente existe a possibilidade de interagir com o processo. Os processos background não possibilitam a comunicação com o usuário durante seu processamento. No decorrer desta unidade será apresentada uma atividade prática sobre estes conceitos, o que facilitará o entendimento dos mesmos.

Existem três tipos de processos: **processos independentes**, **subprocessos** e **threads**. Os **processos independentes** consistem na maneira mais simples de implementar a concorrência e cada processo possui seu próprio contexto de hardware, contexto de software e espaço de endereçamento, impactando no seu custo de implementação. Os **subprocessos** são criados dentro de uma estrutura hierárquica, sendo que o processo criador é denominado "processo-pai" e o subprocesso "processo-filho". As principais características do subprocessos são: o processo-filho pode criar outros; quando o processo-pai é eliminado, os subprocessos também são; os subprocessos podem compartilhar quotas com o processo-pai. A figura 46 mostra a hierarquia de processos. Nesta ilustração podemos perceber que o processo A consiste no processo-pai de B. O processo B atua como processo-pai de C e D. O processo D consiste no processo-pai de E e F..

Figura 46 - Hierarquia de Processos



Fonte: Adaptação de Stuart, 2011.

No entanto, tanto os processos independentes quanto os subprocessos demandam o consumo de diversos recursos do sistema, pois sempre que um processo é criado, o sistema deve alocar um contexto de hardware, um contexto de software e o espaço de endereçamento, consumindo tempo de CPU. Para contornar este desafio surgiram os **threads**. Por meio deste compartilhamento, o tempo gasto em criação, eliminação e troca de contexto é reduzido, gerando uma economia nos recursos do sistema como um todo. Os threads contribuem positivamente para o desempenho de um sistema distribuído.

ATENÇÃO: Cada thread possui seu próprio contexto de hardware, mas compartilha o contexto de software e espaço de endereçamento com outros threads do processo.

3.5 ESCALONAMENTO

O **escalonamento de processos** ou escalonamento do processador é a atividade do SO responsável por escolher qual processo vai ser executado pelo processador (CPU) e por quanto tempo (alguns milissegundos) ele vai ficar executando. A parte do SO que faz esta escolha é chamada de escalonador e o algoritmo que ele usa para fazer essa escolha é o **algoritmo de escalonamento** (TANNENBAUM, 2016).

Todos os processos (programas em execução) que estão ativos ficam aguardando em uma fila até que o SO escolha um deles para utilizar a CPU por um período determinado, denominado fatia de tempo (time-slice), normalmente de 20 a 30 milissegundos, ou até que executem uma operação de entrada e saída antes do término da sua fatia de tempo. Ao terminar a fatia de tempo, o mesmo é colocado no final da fila e o processo que está na frente da fila passa a executar. Se durante a execução ele necessita de uma operação de entrada e saída, o mesmo é retirado da execução e colocado na fila de processos bloqueados, que estão aguardando uma operação de entrada e saída; quando a operação de entrada e saída está concluída, o processo é colocado na fila de processos que estão aguardando a execução.

Quando um processo faz uma requisição de uma operação de entrada e saída, imediatamente ele perde o processador e fica na fila de processos bloqueados, porque uma operação de entrada e saída demora mais do que a fatia de tempo restante que ele teria disponível para usar a CPU, e como ele depende da operação de E/S, não há motivo para ele ficar aguardando e deixar a CPU ociosa.

A forma como o escalonamento é realizado é responsável pela produtividade e eficiência atingidas por um sistema computacional e envolve política de tratamento dos processos que permita obter os melhores resultados possíveis. A política de um escalonador deve levar em conta os seguintes objetivos (TANNENBAUM, 2016):

a) **Justiça**: dar a cada processo uma porção justa da CPU;

b) **Aplicação da política**: verificar se a política estabelecida está sendo cumprida;

c) **Equilíbrio**: manter ocupadas todas as partes do sistema;

d) **Vazão (throughput)**: maximizar o número de tarefas executadas;

e) **Utilização de CPU**: manter a CPU ocupada o tempo todo;

f) **Tempo de resposta**: responder rapidamente às requisições;

g) **Proporcionalidade**: satisfazer às expectativas dos usuários;

h) **Previsibilidade**: evitar a degradação da qualidade.

Um algoritmo de escalonamento pode ser **não preemptivo** quando o processador designado para um certo processo não pode ser retirado deste até que o processo seja finalizado. Já um algoritmo de escalonamento é considerado **preemptivo** quando o processador designado para um processo pode ser retirado deste em favor de um outro processo. Os sistemas modernos utilizam algoritmo preemptivo, determinando uma fatia de tempo máximo de alguns milissegundos para cada processo utilizar a CPU.

Há diferentes tipos de algoritmos de escalonamento, mas o principal modelo utilizado nos SOs atuais é baseado em **múltiplas filas, com prioridades**. Neste modelo, os processos são classificados de acordo com sua característica, se são I/O Bound ou CPU Bound e colocados em diferentes filas com prioridades diferentes. Sempre que tiver um processo pronto para ser executado em uma fila com prioridade maior ele será executado. Não havendo processos nas filas de prioridades mais altas, os processos de menor prioridade são executados. Ao final de cada ciclo de execução, os processos são reavaliados para reenquadrá-los nas devidas filas e prioridades.

As prioridades de execução também podem ser ajustadas pelo usuário ou pelo programa. Por exemplo, um programa antivírus pode ser programado para executar com uma prioridade baixa, para que ele seja executado quando os recursos estiverem ociosos, em segundo plano, não atrapalhando e degradando a utilização do computador enquanto ele atua; mas também pode ser programado para executar com prioridade alta, exigindo todos os recursos para ele realizar a tarefa de verificação, praticamente inviabilizando o uso de outros programas enquanto ele não completa a tarefa.

O Windows, por exemplo, implementa 32 níveis de prioridades, em 7 classes de prioridades:

- Ociosa (Idle);
- Baixa (Lowest);
- Abaixo do Normal (Bellow Normal);
- Normal;
- Acima do Normal (Above Normal);
- Alta (Highest);
- Tempo crítico (Time Critical).

O escalonador Linux também provê a existência de prioridades dinâmicas, monitorando o comportamento de um processo e ajustando dinamicamente sua prioridade. Os processos que ocuparam o processador durante um período de tempo maior têm sua prioridade reduzida. De forma análoga, aqueles que estão há mais tempo sem executar recebem um aumento na sua prioridade. O escalonador possui 140 níveis de prioridade (quanto menor o número, maior é a prioridade). Prioridades de 1 a 100 são para processos de tempo real; de 101 a 140 para os demais processos de usuário (interativos ou não interativos). Nos níveis de prioridade 101 a 140, os processos recebem fatias de tempo de 20 milissegundos.

3.6
ESTUDOS DE CASO

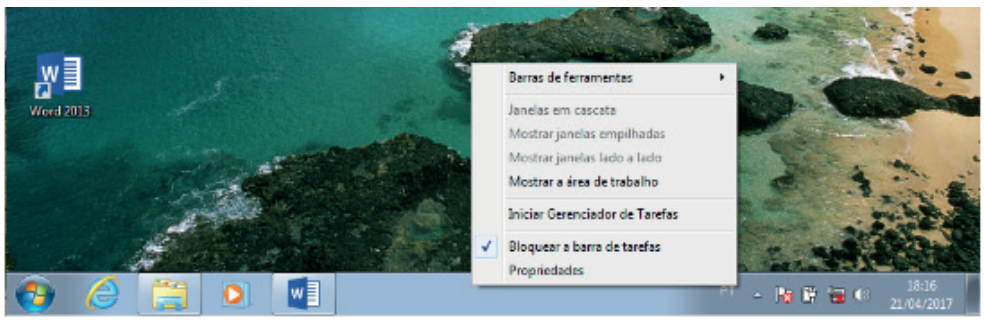
Esta seção apresenta dois estudos de caso. A Subseção 3.6.1 descreve o estudo de caso no SO Windows. A Subseção 3.6.2 apresenta o estudo de caso no SO Linux.

3.6.1 Gerência de Processos no Sistema Operacional Windows

Primeiramente analisaremos a gerência de processos no SO Windows, pois este consiste no SO mais utilizado por usuários comuns. Existe uma grande possibilidade que você esteja utilizando este sistema atualmente ou conheça apenas ele. Devido ao grande número de usuários usando este sistema, ficará fácil seguir os procedimentos de gerenciamento de processos neste sistema.

A ferramenta do SO Windows que implementa o gerenciamento de processos consiste no gerenciador de tarefas. Mas você pode estar se perguntando: Como eu faço para acessar esta ferramenta? A resposta para esta questão pode ser respondida de três formas. A forma convencional compreende os seguintes passos: Clicar com o botão direito do mouse na barra **Iniciar** e selecionar a opção "Iniciar Gerenciador de Tarefas". A figura 47 apresenta como iniciar a ferramenta por meio desta opção.

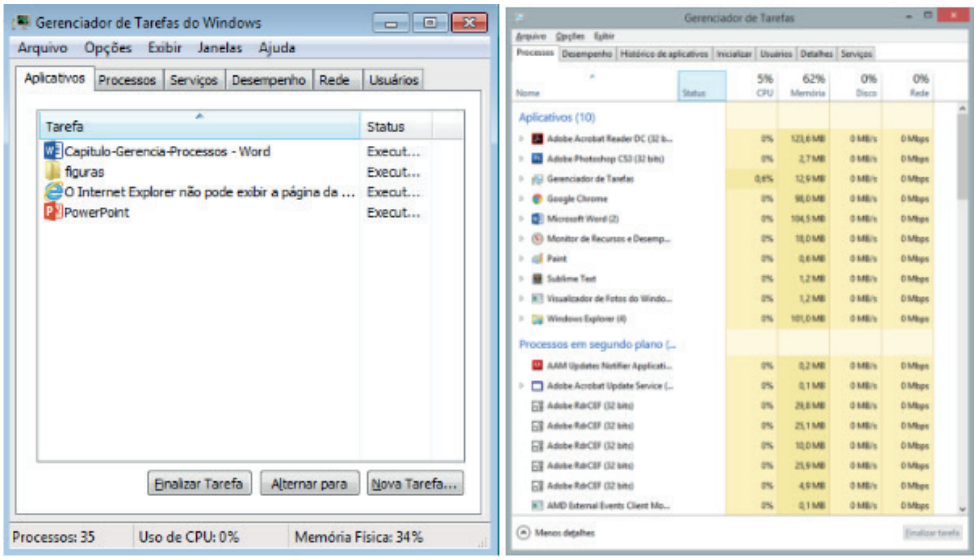
Figura 47 - Iniciando o Gerenciador de Tarefas no Windows



Fonte: Autores, 2017.

A segunda forma consiste em apenas pressionar as teclas Ctrl + Alt + Delete, Ctrl+ (Shift) + Esc ou ainda, clicar no botão Iniciar, em seguida escolha a opção Executar, digite "taskmgr" e pressionar a tecla Enter. Seguindo qualquer uma destas três opções, a tela ilustrada na figura deve ser aberta dependendo se o seu Windows for a versão 7, 8 ou 10.

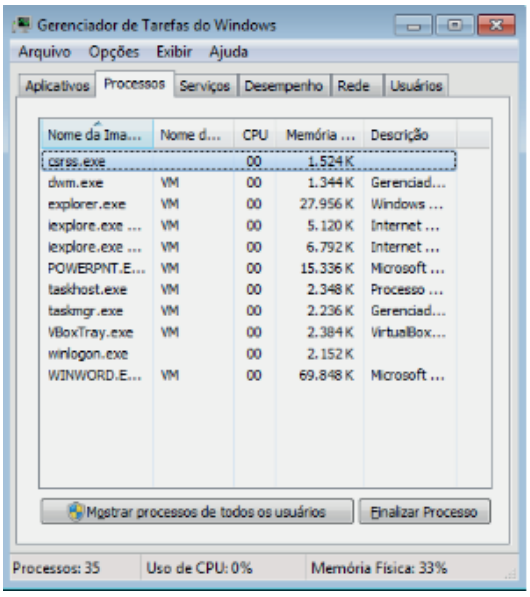
Figura 48 - Passo 1 - Gerência de Tarefas no Windows



Fonte: Autores, 2017.

As principais funcionalidades da ferramenta Gerenciador de Tarefas relacionadas com a gerência de processos são encontradas nas abas Processos e Desempenho. A aba Processos apresenta todos os processos que estão em execução. A figura exemplifica um tipo de saída encontrado ao clicar nesta aba. Analisando esta saída, veremos uma lista de processos. Você pode clicar com o botão direito em cima de um destes processos para ver as opções de interação com os processos. Algumas das principais opções disponíveis são: finalizar processo, abrir local do arquivo, definir prioridade e definir afinidade.

Figura 49 - Passo 2 - Gerência de Processos no Windows

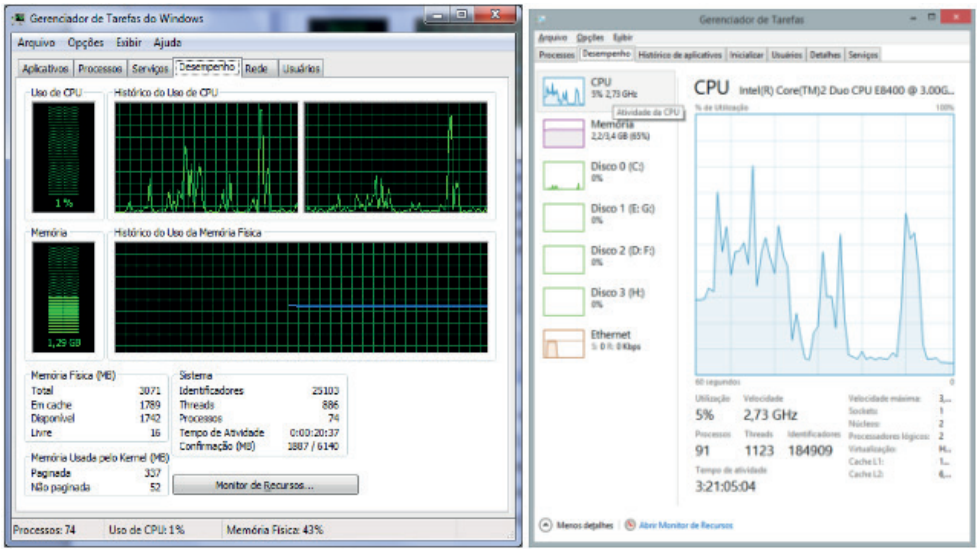


Fonte: Autores, 2017.

Ao escolher uma destas opções, diferentes eventos podem acontecer nos processos. Ao escolher a opção finalizar processo, o estado do processo escolhido será alterado para término e consequentemente será encerrada a sua execução. Escolhendo a opção abrir local do arquivo, será aberta a pasta onde o arquivo executável se encontra. A opção definir prioridade possibilita ao usuário alterar prioridade de um processo para Tempo real, Alta, Acima do normal, Normal, Abaixo do normal e Baixa. Por fim, a opção definir afinidade permite ao usuário expressar sua preferência sobre quais processadores serão usados para realizar o processo em questão.

A aba **Desempenho** possibilita ao usuário monitorar a utilização dos principais recursos do computador. A figura apresenta o resultado da saída desta aba, para a versão Windows 7 e Windows 8 e 10. Por meio desta aba, você poderá obter informações sobre a utilização da memória e CPU do seu computador. Alguns exemplos dos tipos de informações mais específicas sobre a gerência de processos que podem ser obtidas nesta aba consistem no número de processos e threads.

Figura 50 - Passo 3 - Gerência de Processos no Windows



Fonte: Autores, 2017.

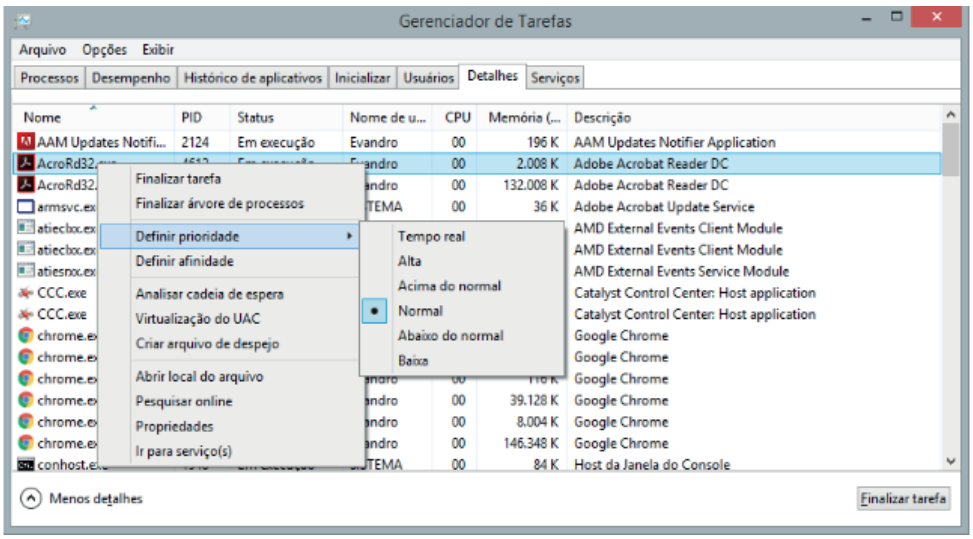
Na Figura 50, na tela da esquerda (Windows 7), podemos visualizar como está o uso da CPU e da memória do computador, onde podemos ver que a CPU está com 1% de uso no momento, com 74 processos em execução e utilizando 43% da memória RAM instalada. Analisando mais detalhadamente, podemos ver que o kernel do SO está usando 337Mb de memória paginada mais 52MB de memória não paginada, ou seja, o kernel do Windows 7 está usando 389MB de memória. Também podemos ver que estão em execução 74 processos com 886 threads. Na imagem da direita (Windows 8 e 10) podemos identificar que a CPU está com 5% de uso, com 91 processos e 1123 threads em execução e que estamos usando 65% da memória instalada. Também podemos ver que o computador está ligado há 20:37s na imagem da esquerda e há 3 dias, 21:05:04s na imagem da direita. Também podemos ver na tela da direita que o processador instalado é um Intel Core 2 Duo E8400 operando a 3Ghz e que o mesmo possui 2 núcleos (e 2 processadores lógicos).

Como nosso objetivo nessa unidade é a gerência de processos, não vamos nos

preocupar com as informações relacionadas à memória, que veremos na próxima unidade. Cada processo tem pelo menos um thread de execução, mas normalmente eles possuem mais de um thread. Por exemplo, quando acessamos uma página web com o Google Chrome, temos pelo menos um processo em execução (alguns aplicativos criam mais de um processo por questões de otimização e segurança) que vai carregar uma página web. Quando iniciamos o carregamento enquanto está vindo o texto da página web, várias threads adicionais são criadas para fazerem os downloads das imagens. Desta forma, quando termina de carregar a página, as imagens já estão praticamente prontas para serem visualizadas. A função do thread é justamente esta: realizar parte do trabalho de forma assíncrona e simultânea a outra tarefa.

Na aba **Detalhes** do gerenciador de tarefas do Windows 8 e 10, ou diretamente na aba Processo do gerenciador de tarefas do Windows 7, podemos clicar sobre um processo e enviar sinais para o mesmo, mudando sua prioridade ou encerrando sua execução (Finalizar tarefa), como mostra a Figura 51. O recurso de **Finalizar tarefa** ou **Finalizar árvore de processos** é bastante útil quando um programa "trava" e para de responder. Já a mudança de prioridades é extremamente delicada e exige grandes conhecimentos técnicos e pode deixar o sistema instável.

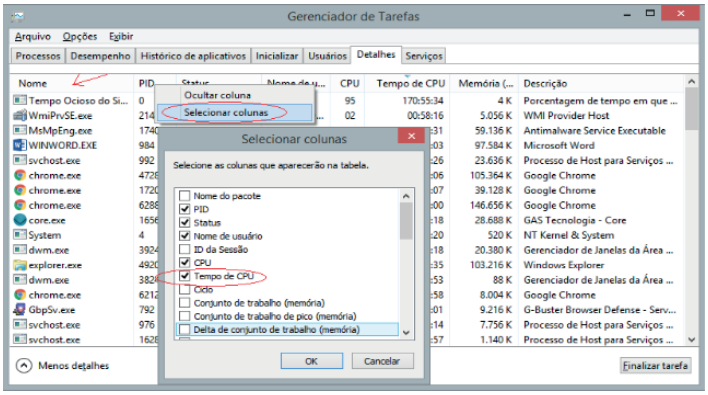
Figura 51 - Alterando um processo



Fonte: Autores, 2017.

Nesta aba também é possível escolher as colunas com as informações que nos interessam, como mostra a Figura 52, bastando clicar com o botão direito do mouse sobre os nomes das colunas e escolhendo: Selecionar colunas, marcando as colunas que nos interessam, como no caso da Figura 52 que vamos marcar a coluna com o tempo de CPU de cada processo.

Figura 52 - Alterando um processo

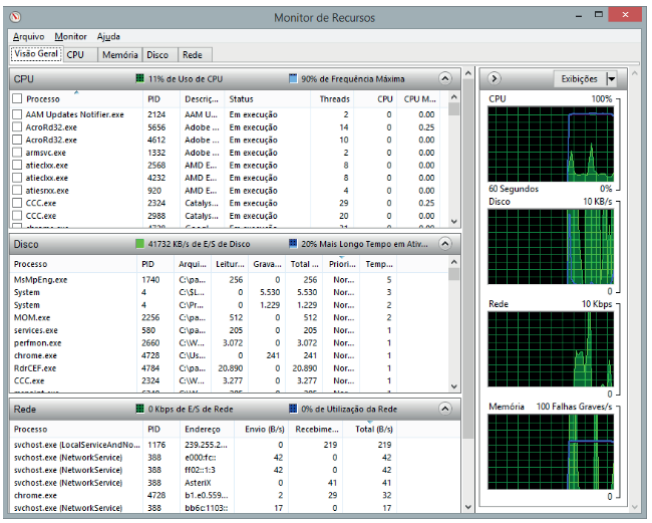


Fonte: Autores, 2017.

Na Figura 52 pudemos ver que o processo que mais utilizou CPU é o Tempo Ocioso do Sistema, ou seja, não é um processo em si, mas é o que guarda a contabilidade do tempo em que a CPU não foi usada por nenhum outro processo, e marca 170 horas, 55 minutos e 34 segundos de ociosidade. Como visto na Figura 50 que o computador estava ligado há 3 dias, 21:05:04s, e como o mesmo usa um processador de 2 núcleos, o tempo total de CPU que estava disponível era de 186 horas, 10 minutos e 8 segundos (o dobro do tempo de atividade, por ter 2 núcleos) e destes, 170:55:34s estiveram ociosos, ou seja, não tinha nenhum programa utilizando a CPU. O programa que mais utilizou CPU foi o WmiPrvSe.exe (WMI Provider Host) que utilizou a CPU por 58:16.

Ainda na aba Desempenho, podemos clicar no botão Monitor de Recursos para abrir uma janela como a mostrada na Figura 53, onde é possível visualizar o uso da CPU, Disco, Rede e Memória, com o detalhe de cada processo que está usando o recurso. Clicando em cada aba correspondente podemos encontrar mais detalhes sobre cada um dos recursos.

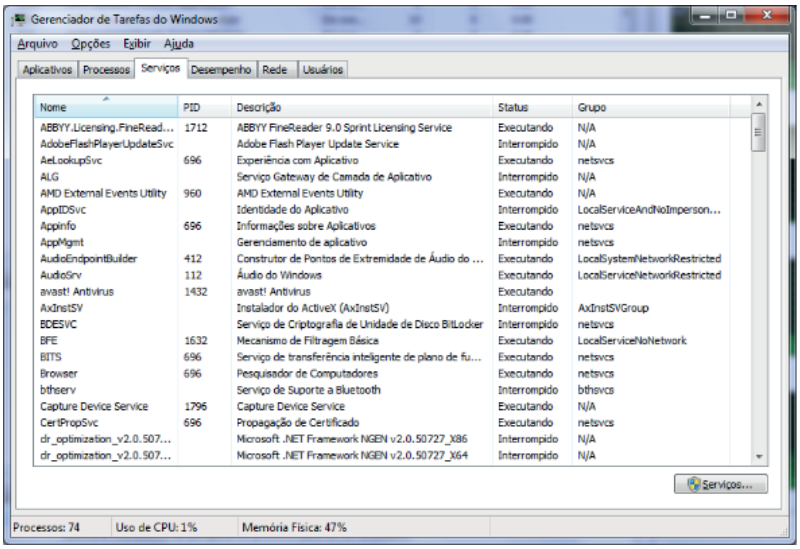
Figura 53 - Monitor de Recursos do Windows



Fonte: Autores, 2017.

Clicando sobre a Aba **CPU** podemos visualizar detalhes sobre o número de threads de cada processo em execução e informações sobre os serviços que estão em execução. Os serviços são processos que executam em segundo plano, ou seja, não possuem interface para que possamos interagir com eles, e executam tarefas que complementam as funções do SO. Nós podemos visualizar os serviços em execução por meio da aba Serviços do gerenciador de tarefas do Windows, como mostra a Figura 54.

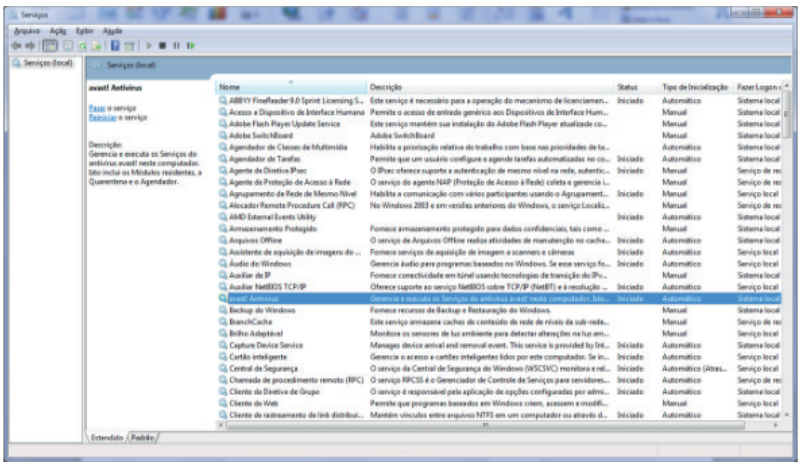
Figura 54 – Serviços do Windows



Fonte: Autores, 2017.

Os serviços que estão em execução possuem um identificador de processo (PID) e seu status aparece como: Executando. Clicando no botão Serviços... dessa aba, é aberta a janela dos serviços do Painel de Controle do Windows, conforme Figura 55.

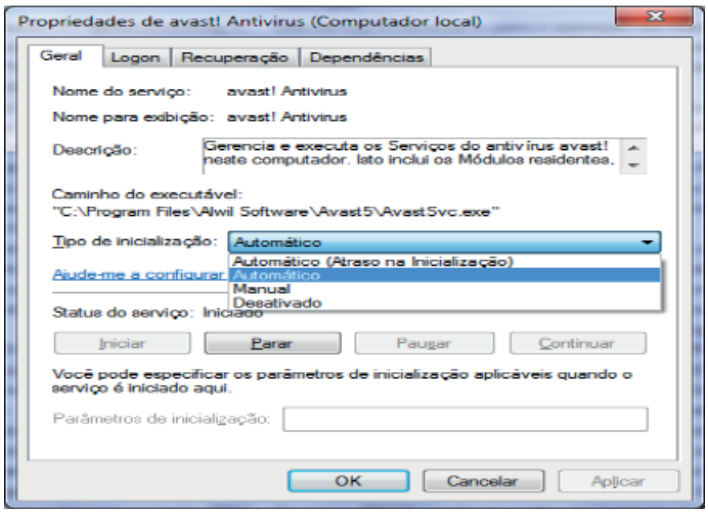
Figura 55 - Configuração dos Serviços do Windows



Fonte: Autores, 2017.

Clicando sobre o serviço, aparece a descrição do mesmo e as opções **Parar** o serviço e **Reiniciar** o serviço (se ele estiver executando – status iniciado) ou **Iniciar** o serviço (se ele estiver parado). Dando um duplo-clique sobre o serviço abre-se uma janela onde é possível configurar o tipo de inicialização do serviço: Automático, Manual ou Desativado. Se for escolhido automático o serviço iniciará junto com o SO; se for escolhido manual o serviço só iniciará quando o usuário clicar em **Iniciar Serviço** ou utilizar um programa que inicia o serviço; e desativado faz com que o serviço não seja iniciado em hipótese alguma, como mostra a Figura 56.

Figura 56 - Configuração dos Serviços do Windows



Fonte: Autores, 2017.

Vários programas são iniciados automaticamente no Windows e podem ser configurados na aba Inicializar, do gerenciador de tarefas do Windows 8 e 10, ou por meio do programa **msconfig.exe** no Windows 7, onde é possível habilitar e desabilitar os programas que serão iniciados automaticamente quando ligamos o computador, conforme mostra a Figura 57.

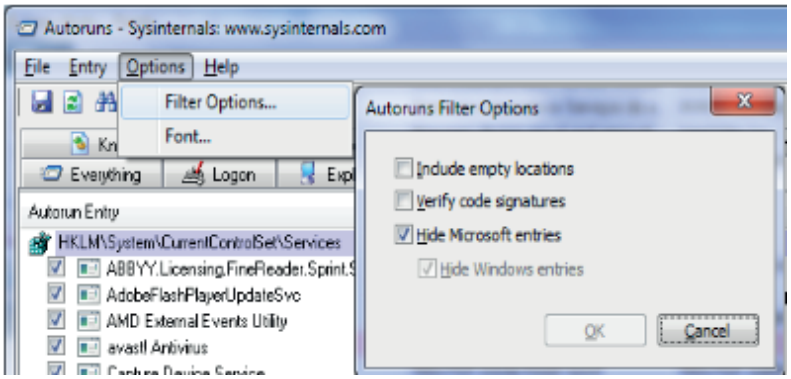
Figura 57 - Configuração dos Programas que inicializarão automaticamente

Fonte: Autores, 2017

Existem ainda outros programas de monitoramento e configuração do SO, que não acompanham o mesmo, mas que podem ser utilizados, dentre os quais se destacam o Autoruns e o Process Explorer da SysInternals.

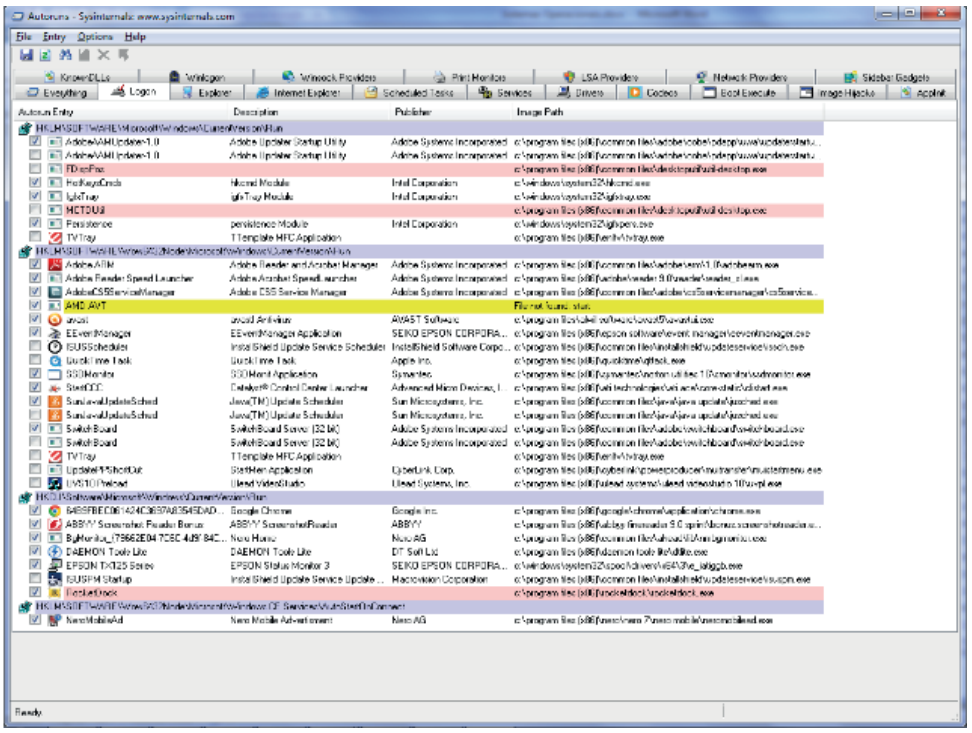
O Autoruns é um programa que compõe o suite da SysInternals e permite visualizar todos os processos que são inicializados automaticamente no nosso computador. A aba **Logon** apresenta todos os processos que inicializarão junto com o SO e, ao usá-lo, é importante certificar-se que a opção de filtro está escondendo as entradas de processos e serviços da Microsoft e do SO, para evitar desabilitar serviços e processos essenciais para o bom funcionamento do sistema: **Hide Microsoft Entries/ Hide Windows Entriesn**, como mostram as Figuras 58 e 59.

Figura 58 o Configuração dos Autoruns para esconder as entradas da Microsoft



Fonte: Autores, 2017.

Figura 59 - Autoruns exibindo os programas que são inicializados

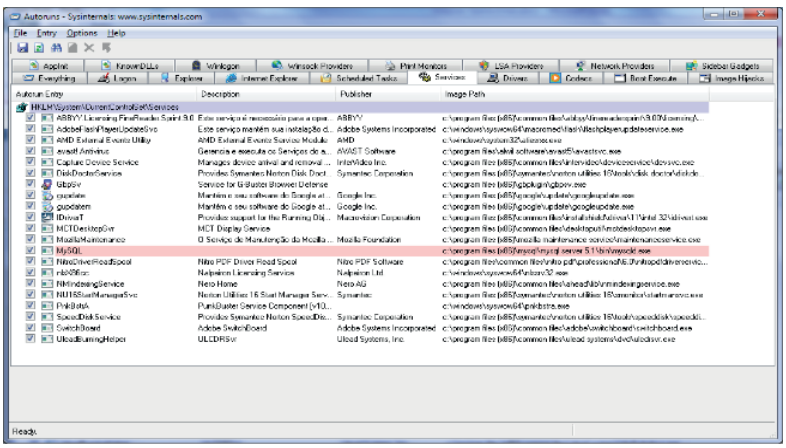


Fonte: Autores, 2017.

Para que um processo não inicialize, pode-se excluir o processo da lista clicando sobre o mesmo e pressionando a tecla [Delete] ou clicando com o botão direito e escolhendo a opção **Delete**. Esta exclusão não permite voltar atrás e desfazer a exclusão, portanto é muito perigoso fazer isso. O mais seguro é desmarcar a caixa de escolha (checkbox) do processo, pois desta forma o mesmo não é inicializado. Se for necessário iniciar o processo, basta retornar para esta tela, marcar novamente o processo e reinicializar o computador. Observe que na tela da Figura 59 várias caixas estão desmarcadas, significando que esses processos não são iniciados, poupando memória e melhorando o desempenho do SO.

Clicando sobre a aba Services são apresentados todos os serviços que não fazem parte do SO e que estão instalados, e provavelmente em execução, como mostra a Figura 60.

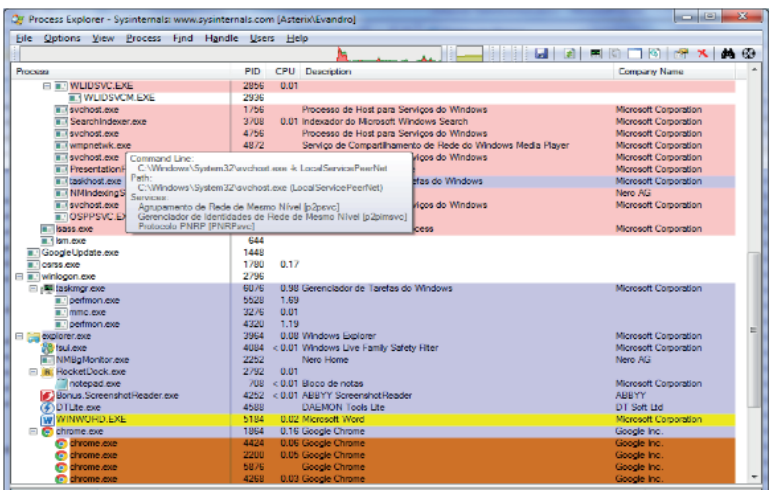
Figura 60 - Serviços que serão inicializados automaticamente



Fonte: Autores, 2017.

O Process Explorer também faz parte do suite da SysInternals e permite visualizar os processos em execução de acordo com a hierarquia, ou seja, cada processo fica vinculado ao processo que o criou, como mostra a Figura 61.

Figura 61 - Process Explorer da SysInternals



Fonte: Autores, 2017.

Esta subseção descreveu um estudo de caso sobre a gerência de processos usando o SO Windows. Vimos que os usuários interagem com o SO para customizar estas funções por meio da ferramenta **Gerenciador de Tarefas**. Percebemos que o usuário consegue alterar um conjunto limitados de opções através desta ferramenta gráfica.

3.6.2 Gerência de Processos no Sistema Operacional Linux

Esta seção descreve um estudo de caso da gerência de processos considerando o SO Linux. O principal motivo de realizarmos este estudo deve-se ao tipo de licença do Linux. Apenas para salientar os benefícios, podemos fazer uma comparação da licença do Linux com a do Windows. O SO Windows é comercializado por meio de uma licença de software proprietária e que restringe o acesso ao seu código fonte. Em contrapartida, a licença do Linux permite o acesso, o estudo e a modificação do código fonte do SO. Para fins de estudos, esta particularidade do sistema Linux proporciona vários benefícios aos estudantes da área de Informática, pois aspectos técnicos podem ser analisados e compreendidos nos mínimos detalhes. Outro motivo para estudar o Linux, consiste na sua vasta aplicação em servidores de alta disponibilidade. Muitas grandes empresas, especializadas em prestar serviços para um grande número de usuários e provedores de serviços de Internet, utilizam este SO, oferecendo vagas de trabalho para profissionais do ramo da Informática capacitados a operarem o sistema Linux.

Este estudo será conduzido considerando o uso da distribuição Linux Ubuntu. Uma distribuição Linux consiste na combinação de um conjunto de softwares para o usuário com o núcleo do sistema. O núcleo do Linux, também conhecido como kernel, implementa todas as funções do SO, incluindo a gerência de processos. A distribuição Ubuntu foi escolhida para este estudo por se tratar de uma das distribuições mais utilizadas por usuários comuns, ou seja, você mesmo pode acessar o **site** e instalar, no seu computador, o Ubuntu para **praticar.** Esta seção não abordará a instalação e configuração do **Ubuntu**. A seção está organizada como segue: primeiramente serão apresentadas as informações básicas sobre a gerência de processos. Na sequência serão descritos comandos para visualização dos processos. Por fim, serão descritos comandos para interagir com os processos.

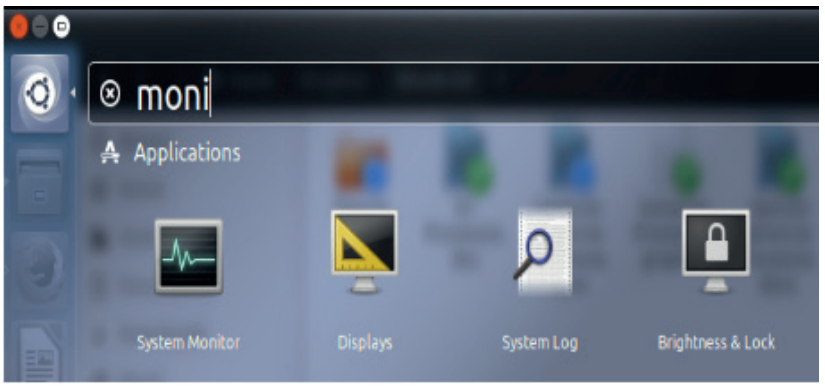
INTERATIVIDADE: www.ubuntu.com/download/desktop

SAIBA MAIS: Caso você queira aprender mais sobre a instalação do Ubuntu, acesse o site www.ubuntuforum-br.org.

Se você quiser um tutorial detalhado sobre a instalação do Ubuntu, acesse o link www.techtudo.com.br/dicas-e-tutoriais/noticias/2016/01/como-instalar-o-ubuntu.amp

Assim como acontece no sistema operacional Windows, o Ubuntu Linux fornece uma interface simples para usuários leigos obterem informações sobre o gerenciamento de processos. Basicamente, este procedimento pode ser realizado em quatro passos. O primeiro passo consiste em acessar a ferramenta de monitoramento dos recursos do sistema. Para isto, basta que você clique no botão principal e pesquise por monitor para abrir a ferramenta System Monitor ou Monitor do Sistema. A figura ilustra como executar este passo. Vale a pena salientar que este procedimento de busca por programas pode ser utilizado no Ubuntu para encontrar qualquer programa instalado.

Figura 62 - Passo 1 - Gerência de Processos no Linux



Fonte: Autores, 2017.

Ao abrir a ferramenta, você visualizará três opções principais na parte superior da ferramenta, sendo elas: Processes, Resources e File Systems. A opção Processes apresenta as informações sobre os processos do sistema. A figura 63 mostra o resultado da ferramenta ao escolher esta opção. Ao usar esta ferramenta, percebemos que as informações fornecidas são muito similares àquelas obtidas por meio do gerenciamento de tarefas do Windows.

Figura 63 - Passo 2 - Gerência de Processos no Linux

Process Name	User	% CPU	ID	Memory	Priority
at-spi2-registryd	ricardo	0	2174	544,0 KiB	Normal
at-spi-bus-launcher	ricardo	0	2122	828,0 KiB	Normal
bamfdaemon	ricardo	0	1881	8,4 MiB	Normal
compiz	ricardo	1	2145	262,0 MiB	Normal
dbus-daemon	ricardo	0	1646	1,8 MiB	Normal
dbus-daemon	ricardo	0	2129	484,0 KiB	Normal
dconf-service	ricardo	0	1687	708,0 KiB	Normal
deja-dup-monitor	ricardo	0	2746	3,0 MiB	Normal
dropbox	ricardo	0	2335	193,5 MiB	Normal
evolution-addressbook-factor	ricardo	0	2383	2,9 MiB	Normal
evolution-addressbook-factor	ricardo	0	2404	3,0 MiB	Normal
evolution-calendar-factory	ricardo	0	2339	6,4 MiB	Normal
evolution-calendar-factory-sul	ricardo	0	2374	4,9 MiB	Normal
evolution-calendar-factory-sul	ricardo	0	2385	4,7 MiB	Normal
evolution-source-registry	ricardo	0	2263	6,9 MiB	Normal
firefox	ricardo	5	3610	644,7 MiB	Normal
gconfd-2	ricardo	0	3093	512,0 KiB	Normal
gedit	ricardo	0	3782	19,4 MiB	Normal

Fonte: Autores, 2017.

Ao escolher a opção Resources, a ferramenta fornece informações sobre a utilização do conjunto de processadores do computador, da memória e da rede. A figura ilustra esta opção. Vale salientar que as informações, sobre a utilização da memória, agregam dados de monitoramento da memória swap. A memória swap consiste em uma área de troca entre a memória principal e a memória secundária em situações que a memória principal não possui mais capacidade disponível. Por meio do monitoramento de rede, diagnósticos simples sobre o consumo da largura de banda no computador podem ser obtidos.

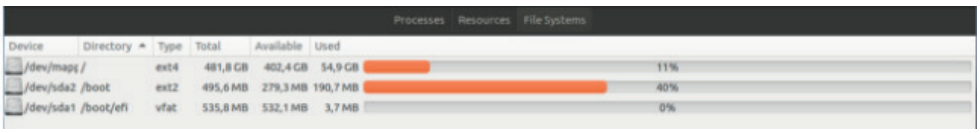
Ao selecionar um destes processos e clicar com o botão direito, existem diversas opções para interagir com os processos. As principais opções disponíveis nesta aba consistem no **mapa de memória, mudar a prioridade, parar, continuar e matar o processo**. O **mapa de memória** informa detalhes técnicos sobre as informações que o processo armazena em memória. A opção **mudar a prioridade** possibilita alterar as opções de muito baixa até muito alta, ou customizar a prioridade do processo. As opções **parar, continuar** e **matar**, permitem alterar os estados dos processos. Por meio destas opções o usuário pode interagir com os processos no Ubuntu.

Figura 64 - Passo 3 - Gerência de Processos no Linux

Fonte: Autores, 2017.

A opção File Systems disponibiliza informações sobre o uso do sistema de arquivos. Por meio desta opção você poderá monitorar informações sobre a utilização das partições do computador. Usando esta funcionalidade você consegue verificar se o disco rígido está muito sobrecarregado de arquivos. A Figura 65 mostra o resultado desta opção.

Figura 65 - Passo 4 - Gerência de Processos no Linux



Fonte: Autores, 2017.

Nesta subseção vimos que a distribuição Linux Ubuntu fornece uma ferramenta gráfica semelhante ao gerenciador de tarefas do Windows, capaz de prestar funcionalidades relacionadas com o gerenciamento de processos. No Ubuntu, esta ferramenta consiste no System Monitor. Por meio desta ferramenta torna-se possível visualizar os processos e interagir com os processos. Esta ferramenta proporciona ao usuário interagir com os processos usando uma interface gráfica.

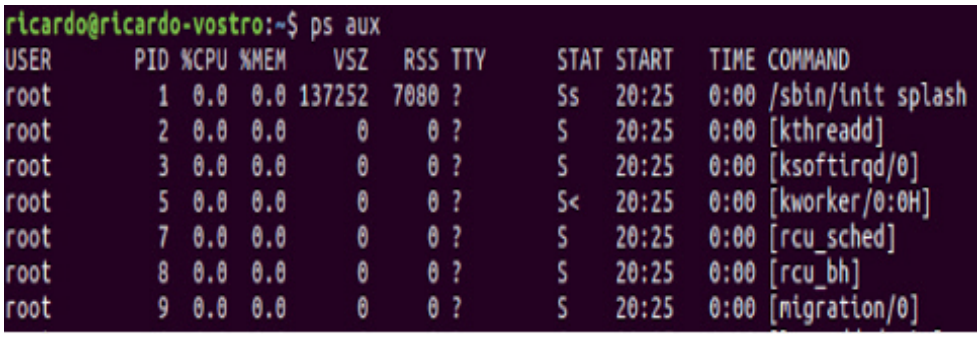
Comandos para Visualização de Processos

Devido ao fato de que o SO Linux possui seu código de fonte aberto, temos a opção de explorar com mais detalhes o funcionamento da gerência de processos. Uma forma intermediária de fazer isto consiste em explorar a utilização de comandos. Podemos classificar os principais comandos do SO Linux relacionados com a gerência de processos em dois grupos: os comandos para visualização e para interação com os processos. Os comandos ps e pstree pertencem ao grupo de visualização. Na sequência, cada um destes comandos será explicado e veremos sua utilização prática. Antes de iniciarmos a discussão técnica sobre estes comandos, precisamos aprender como podemos executá-los. As diferentes variações do SO Linux possuem uma característica em comum, todas proporcionam a utilização de pelo menos uma ferramenta de terminal shell. A principal função desta ferramenta consiste em interpretar e executar comandos, proporcionando aos usuários mais experientes uma interface avançada de interação do usuário com o SO. Como nosso estudo de caso será realizado usando a distribuição Linux Ubuntu, focaremos no terminal desta versão. Para acessar esta ferramenta, basta clicar na barra de busca de programas e digitar terminal. Após este passo, aparecerá o ícone do terminal. Clique neste ícone para iniciar a ferramenta.

Agora que conseguimos acessar o terminal, focaremos nossos esforços em testar os diferentes grupos de processos. Primeiramente estudaremos o grupo de processos de visualização, ou seja, os comandos ps e pstree. Quando executado, o comando ps gera uma lista com todos os processos em execução e seus atributos. Como nossa intenção consiste em testar este comando, digite no terminal o comando ps aux. Passando por parâmetro a opção **aux**, o comando ps mostrará todos os processos de todos os usuários, fornecendo uma lista completa dos processos existentes no SO. A figura mostra o resultado da execução deste comando.

SAIBA MAIS: Caso você queria saber mais sobre as opções de um comando, digite no terminal man nome_do_comando. Por exemplo, digite manps para ver e acessar o manual do sistema operacional sobre o comando ps. O mesmo princípio pode ser seguido para outros comandos.

Figura 66 - Resultado do Comando PS AUX

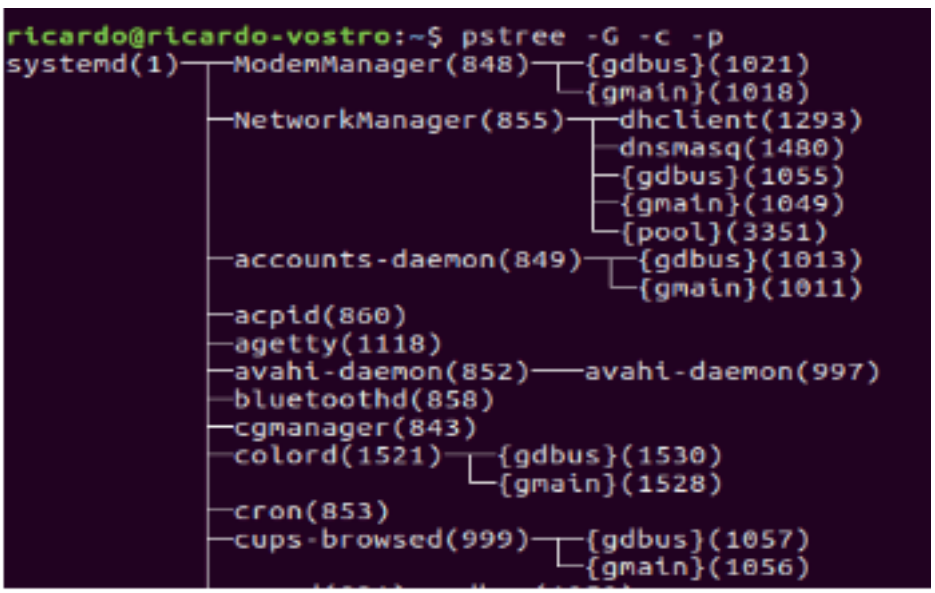


Fonte: Autores, 2017

O resultado do comando ps ilustrado na figura é representado em formato tabular. Cada coluna desta tabela apresenta informações importantes sobre os processos. Cada linha desta tabela corresponde a um processo em execução no sistema operacional. A coluna USER descreve qual usuário é o dono do processo. A coluna PID retrata a identificação do processo. No sistema Linux, cada processo possui um número de identificação único. O primeiro processo iniciado no SO Linux consiste no init, e seu PID sempre será 1. Para o restante dos processos este número é incrementado ao passo que novos processos são executados. As colunas %CPU e %MEM detalham o percentual de uso de CPU e memória usada por um processo, sendo muito útil para identificar processos que estão consumindo muitos recursos do sistema operacional. Por fim, a coluna COMMAND mostra o nome do processo.

Outro comando para visualização de processos consiste no pstree. O principal diferencial deste comando consiste em proporcionar uma visão em forma de árvore para descrever a relação dos processos. Por meio deste comando, podemos observar a hierarquia dos processos no SO Linux e distinguir processos pais dos processos filhos. A figura mostra o resultado da execução deste comando.

Figura 67 - Resultado do Comando PSTREE



Fonte: Autores, 2017.

A gerência de processos no SO Linux apresenta peculiaridades relacionadas com a utilização de variáveis de ambiente, diretório (ou pasta) de trabalho e temporizadores. Neste SO, cada processo herda do processo pai algumas variáveis de ambiente, responsáveis por armazenar valores que podem ser importantes para o processo em execução. Além disto, os processos também são associados a um diretório de trabalho. Neste diretório, os processos podem fazer leitura e escrita em disco. O núcleo do sistema Linux mantém temporizadores, ou seja, registros da hora da criação dos processos e o seu consumo de CPU durante a sua execução.

Nesta subseção estudamos os comandos para visualizar os processos. Os principais comandos estudados foram ps e pstree. Esta subseção explicou cada um destes comandos e forneceu exemplos de como utilizá-los.

Comandos para Interagir com Processos

A interação do usuário com os processos no SO Linux ocorre principalmente por meio dos comandos top, kill, killall, bg, fg, nice e renice. O comando top exibe os processos em execução no sistema obedecendo uma taxa de atualização. Por meio deste comando, o usuário pode monitorar como os processos utilizam os recursos do computador, tais como a CPU e a memória. Além disto, este comando possibilita ao usuário interagir com os processos enviando sinais, os quais serão detalhados no decorrer da seção. A Figura 68 apresenta o resultado da execução deste comando.

Figura 68 - Execução do Comando top

```
top - 21:11:38 up 46 min,  1 user,  load average: 0,19, 0,30, 0,20
Tasks: 237 total,   1 running, 236 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3,0 us,  1,4 sy,  0,0 ni, 95,5 id,  0,1 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem :  8084828 total,  3028980 free,  3523160 used,  1532688 buff/cache
KiB Swap:  8298492 total,  8298492 free,        0 used.  4229320 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1113 root      20   0  395308 113632  63892 S   6,0  1,4   0:59.02 Xorg
 3100 ricardo   20   0 4925732 565556 480904 S   4,3  7,0   0:40.04 VirtualBox
 2145 ricardo   20   0 1715416 211560  88848 S   3,3  2,6   0:35.96 compiz
 2458 ricardo   20   0 1552144  98880  50400 S   2,7  1,2   0:12.05 rhythmbox
 2167 ricardo    9 -11  778948  16208  12544 S   2,0  0,2   0:07.06 pulseaudio
 1772 ricardo   20   0   89320   5396   4836 S   0,3  0,1   0:01.44 xbrlapi
 2071 ricardo   20   0  484884  30856  25348 S   0,3  0,4   0:00.23 ibus-ui-gtk3
 2335 ricardo   20   0 3196684 228652  57124 S   0,3  2,8   0:13.57 dropbox
 3159 ricardo   20   0  810420  36788  28112 S   0,3  0,5   0:00.86 gnome-terminal-
 3419 ricardo   20   0   49232   4012   3256 R   0,3  0,0   0:00.02 top
    1 root      20   0  137252   7080   5132 S   0,0  0,1   0:01.00 systemd
```

Fonte: Autores, 2017.

O comando kill permite ao usuário enviar sinais para os processos. A Tabela 1 apresenta os principais tipos de sinais deste sistema. Para enviar um sinal por meio do comando kill, torna-se necessário conhecer previamente o PID do processo. Para obter esta informação você pode usar o comando ps, conforme mostrado previamente. A título de exemplo, considere que obtemos o número 9000 como PID do processo que desejamos enviar o sinal HUP. Neste caso, bastaria digitarmos o comando kill –HUP 9000.

Tabela 1 - Principais Tipos de Sinais do Sistema Operacional Linux

SINAL	VALOR NUMÉRICO	AÇÃO
HUP	1H	ang-Up ou desligamento. Este sinal é usado quando o usuário desconecta de uma sessão ou fecha um terminal. Ele também é utilizado por processos servidores para invocar a releitura do arquivo de configuração.
INIT	2	Interrompe o processo. Ele é enviado automaticamente quando um processo é abortado com as teclas *Ctrl+c*.
KILL	9T	ermina o processo incondicionalmente de forma rápida e drástica. Este tipo de sinal pode deixar arquivos abertos e bases de dados corrompidos. Deve ser utilizado caso o processo pare de responder ou em uma emergência.
TERM	15	Termina o processo de forma elegante, possibilitando que ele feche arquivos e execute suas rotinas de fim de execução.
TSTP	18	Termina a execução para continuar depois. Este sinal é enviado automaticamente quando utilizamos as teclas *Ctrl+z*. É utilizado para colocar o processo em segundo plano.

Fonte: Ribeiro, 2004.

Diferentemente do comando kill, o comando killall não necessita do conhecimento prévio do PID do processo. Ao invés do PID, o usuário deve passar como parâmetro o nome do processo. No entanto, o comando killall possibilita apenas enviar o sinal de término do processo. A título de exemplo, suponha uma situação hipotética onde tenhamos a necessidade de terminar o processo do navegador Web Firefox usando o comando killall. Neste caso, poderíamos abrir o terminal e digitar: *killallfirefox*.

Os comandos fg e bg são responsáveis por colocar o processo em *foreground* e *background*, respectivamente. Para exemplificar o uso destes comandos, abra um terminal e execute o comando gedit. Como resultado será aberto o programa *Gedit*, uma espécie de bloco de notas desenvolvido para o SO Linux. Observe que ao executar este programa, o terminal ficará impossibilitado de executar novos comandos e serão apresentadas mensagens internas do sistema operacional. Neste momento, o processo do Gedit está em *foreground*. Para passar o processo para background, primeiramente precisamos suspendê-lo. Isto pode ser feito ao pressionar as teclas Ctrl+z. Ao colocar o processo como suspenso, você verá que ele ficará impossibilitado de ser usado. Neste caso, você pode digitar o comando bg para colocá-lo em background ou fg para colocá-lo em foreground.

Os comandos *nice* e *renice* possibilitam ao usuário alterar a prioridade de execução de um processo, impactando assim na urgência que o mesmo é escalonado pelo **processador**. Quanto maior a prioridade do processo, mais tempo de CPU este processo terá. Mudar a prioridade da execução de um processo pode ser útil para proporcionar que usuários avançados configurem o sistema operacional para valorizar processos considerados mais importantes.

ATENÇÃO: Os processos no sistema operacional Linux compreendem a faixa de valores de -20 até 19. Um processo com prioridade -20 representa o nível mais alto, enquanto que 19 representa o menor nível de prioridade.

A diferença entre estes comandos está na forma como eles se relacionam com os processos. O comando nice é usado para configurar a prioridade de um processo no momento da sua criação. Por exemplo, ao executar o comando nice *-12 firefox*, você está iniciando o navegador *Web Firefox* com prioridade 12. Para executá-lo com o maior nível de prioridade, você precisa digitar o comando nice *–20 firefox*. O comando renice permite alterar a prioridade do processo durante a sua execução. Para isto, você precisa previamente saber o PID do processo a ser modificada a prioridade. Tendo esta informação, você pode digitar, por exemplo, o comando *renice17 –p 1134* para alterar para 17 a prioridade do processo com PID 1134. Por meio do renice, a prioridade negativa pode ser passada diretamente por parâmetro. Por exemplo, digitando *renice -3 –p 1134*, o processo com PID 1134 executará com prioridade -3.

Nesta subseção estudamos os comandos para interagir com os processos. Os principais comandos estudados foram *top, kill, killall, bg, fg, nice e renice*. Esta subseção explicou cada um destes comandos e forneceu exemplos de como utilizá-los.

4

GERÊNCIA DE MEMÓRIA

INTRODUÇÃO

Um computador é uma máquina formada por um processador, pelos dispositivos de entrada e saída e pela memória. O que tornou os computadores tão populares e poderosos é a capacidade de armazenar os programas e os dados na memória. Uma CPU somente executa as instruções dos programas que estão armazenadas em memória e, para cada instrução, faz a execução da mesma, que pode ser uma movimentação de dados da memória para a CPU, uma movimentação de dados da CPU para a memória, um cálculo de um dado que já está na CPU ou a movimentação de um dado de um dispositivo de E/S para a CPU ou da CPU para o dispositivo de E/S. Desta forma, boa parte das instruções envolvem a memória.

Com a evolução dos computadores e SOs, vários programas podem estar em execução simultaneamente, porém, cada programa deve ter sua própria área de memória. Para garantir a segurança e integridade dos dados, um programa não pode invadir a área de memória de outro programa e é o SO quem deve garantir isso. Da mesma forma, a memória de um processo deve ser dividida e protegida de modo que as instruções dos programas não possam ser alteradas durante sua execução e na área de dados dos programas não pode ter execução de instruções, para evitar ataques de vírus e programas maliciosos.

Além disso, às vezes precisamos executar programas que consomem mais memória física do que temos instaladas em nosso computador e o SO se encarrega de gerenciar essa memória para carregar na memória RAM somente o que está sendo executado ou referenciado, mantendo no disco, em uma área reservada à memória virtual ou swap, o restante das informações.

Quem garante esse gerenciamento da memória para proteger os programas e seus dados e permite utilizar mais memória do que a fisicamente instalada é a parte do SO responsável pela gerência de memória. Nesta unidade você aprenderá os principais conceitos sobre a gerência de memória.

4.1
GERÊNCIA DE MEMÓRIA

O gerenciamento de memória é necessário porque a memória primária (ou memória principal - RAM), que é um dos elementos básicos da arquitetura computacional atual, tem a velocidade de suas operações de leitura e escrita consideravelmente mais baixas, do que a velocidade dos processadores, o que é conhecido como gargalo de Von Neumann. Como todo processo utiliza memória primária, ao gerenciarmos memória estamos, indiretamente, gerenciando os processos.

A gerência de memória é a parte do SO responsável por garantir que cada programa tenha uma área de memória para o seu código e os seus dados, garantidas a segurança e integridade para a sua execução.

Em um computador atual, vários processos estão em execução simultaneamente e, para isso, precisam estar alocados na memória. Em um sistema computacional o armazenamento de dados ocorre hierarquicamente, ou seja, em diversos níveis de acordo com os tipos de dispositivos, devido a quatro fatores básicos:

- Tempo de acesso;
- Velocidade de operação;
- Custo por unidade de armazenamento;
- Capacidade de armazenamento.

Devido às dificuldades tecnológicas, associadas à construção de dispositivos eficientes de memória e seu custo, o armazenamento de dados em um sistema computacional apresenta a seguinte organização:

a) **Armazenamento interno:** são posições de memória disponíveis internamente no processador, como os registradores e seu **cache** interno;

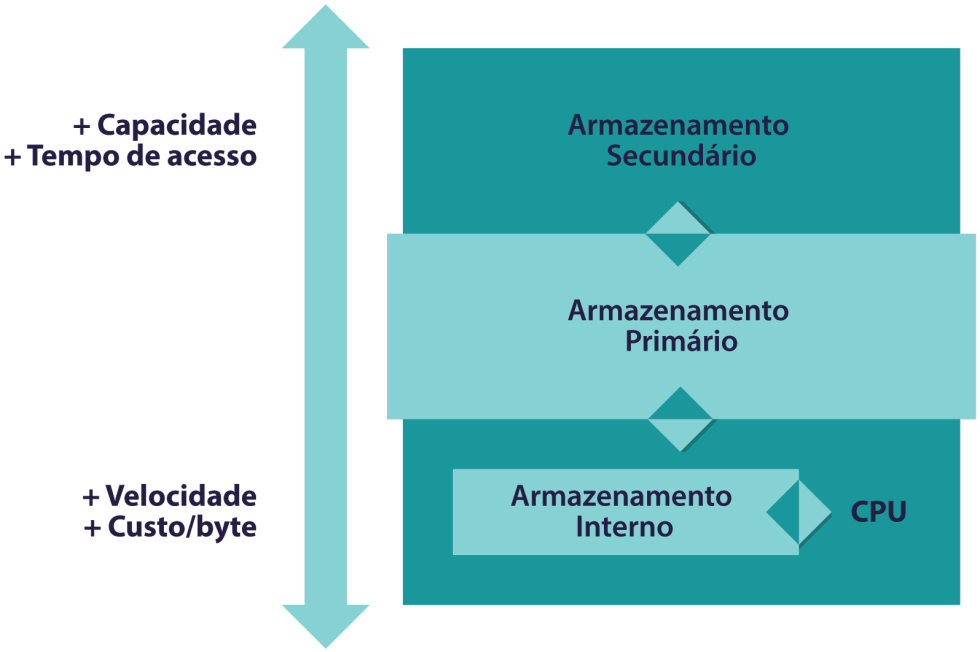
b) **Armazenamento primário**: são posições de memória externa, diretamente acessíveis pelo processador, como os circuitos do tipo RAM, EEPROM, EPROM, PROM ou ROM;

c) **Armazenamento secundário**: são as posições de memória externa que não podem ser acessadas diretamente pelo processador, devendo ser movidas para o armazenamento primário antes de sua utilização, como as unidades de disco rígido e mídias removíveis.

TERMO DO GLOSSÁRIO: Cache: é um dispositivo de memória com acesso rápido que serve de intermediário entre um operador de um processo e o dispositivo de armazenamento ao qual esse operador acessa e que guarda temporariamente os dados acessados. Em um novo acesso, os dados são obtidos da própria cache ao invés de acessar o dispositivo de armazenamento.

O armazenamento interno é aquele que possui as maiores velocidades de acesso e as mais altas taxas de transferência de dados, embora sendo os mais caros. Por este motivo as quantidades e memória deste tipo são menores. Em contrapartida, os dispositivos de armazenamento secundário são os de maior capacidade e de melhor relação custo por byte, mas significativamente mais lentos. A memória primária representa um caso intermediário, onde a velocidade e tempo de acesso são adequados à operação, mas ainda não com o nível desejado. A Figura 69 apresenta a organização da memória em níveis.

Figura 69 – Organização da memória em níveis



Fonte: Autores, 2017.

O modelo apresentado na Figura 69 pode ser percebido nos computadores atuais, onde temos no topo do armazenamento secundário os discos rígidos, com capacidade média de 1 TeraByte (TB), custando aproximadamente R$ 200,00 e com taxa de transferência de dados de 300MB/s; para o armazenamento primário temos memória RAM DDR4 com capacidade de 4 GigaBytes (GB), custando aproximadamente R$ 200,00 e com taxa de transferência de dados de 16GB/s (16 GigaBytes por segundo); para comprar um processador com 6MB de cache adicional paga-se aproximadamente R$ 200,00, com uma taxa de transferência de 64GB/s. Ou seja, os 3 dispositivos têm o mesmo custo, porém a quantidade de bytes armazenados é 1TB, 4GB e 6MB respectivamente, e as taxas de transferência dos dados (velocidade) são 300MB/s, 16GB/s e 64GB/s.

Com a evolução dos computadores, a atual organização conta com outros elementos adicionados para otimizar a performance do sistema e ainda assim reduzir seu custo, como mostra a Figura 70.

Figura 70 - Organização de Armazenamento

Fonte: Autores, 2017.

Os registradores são implementados em número limitado, devido ao seu custo, e utilizados para armazenar temporariamente os dados dentro do processador enquanto estão sendo manipulados. Os caches interno e externo, devido à suas maiores velocidades, são usados para manter uma pequena porção do programa e dos dados mais usados que pode, assim, ser acessada mais rapidamente do que na memória principal, aumentando o desempenho do sistema (DEITEL, 2005).

A memória primária armazena os programas e dados em execução no sistema. Os dispositivos de armazenamento secundário são usados para armazenamento dos dados de forma permanente, embora também possam ser usados para expandir as capacidades da memória primária. O cache de disco é utilizado para acelerar a operação das unidades de disco, podendo, esta técnica, ser utilizada para outros tipos de periféricos.

A necessidade de manter múltiplos programas ativos na memória do sistema, obriga o SO a controlar como esta memória é utilizada por estes vários programas. Como a memória é um recurso caro, cuja administração influencia diretamente a eficiência e desempenho de um sistema computacional, o SO precisa considerar as três estratégias para sua utilização:

Estratégias de busca (fetch): as estratégias de busca preocupam-se em determinar qual o próximo bloco de programa ou dados que deve ser transferido da memória secundária para a memória primária. Usualmente se utilizam estratégias de demanda, ou seja, são transferidos os blocos determinados como necessários para a continuação do processamento;

Estratégias de posicionamento (placement): são as estratégias relacionadas com a determinação das regiões da memória primária (física) que serão efetivamente utilizados pelos programas e dados, ou seja, pela determinação do espaço de endereçamento utilizado;

Estratégias de reposição ou substituição (replacement): são as estratégias preocupadas em determinar qual bloco será enviado à memória secundária para disponibilização de espaço na memória principal para execução de outros programas, ou seja, determinam quais blocos de memória serão substituídos por outros.

Todo SO possui alguma estratégia de busca e alguma estratégia básica de posicionamento. O aumento da sofisticação dos sistemas computacionais exige a utilização de estratégias de busca e posicionamento mais sofisticadas e, para maximizar-se as capacidades dos sistemas computacionais, são necessárias as estratégias de reposição.

4.2
MEMÓRIA VIRTUAL

A memória virtual é uma técnica que os SOs utilizam para gerenciar a memória que permite que os processos (programas) consigam endereçar muito mais memória do que a fisicamente disponível (DEITEL, 2005). A ideia básica da memória virtual é que o tamanho combinado do programa e seus dados podem exceder a quantidade de memória física disponível para o mesmo (TANNENBAUM, 2016).

A memória virtual disponibiliza para os programas uma quantidade de memória que é a soma da memória primária (RAM) com uma área de troca de dados em disco, denominada área de **swap**. O **swapping** é uma técnica que retira os dados que não estão em uso da memória primária (RAM) e armazena temporariamente em uma área de troca do disco rígido e transfere do disco rígido para a memória RAM, sempre que for referenciada ou necessária.

A memória virtual pode ser implementada basicamente por meio dos seguintes mecanismos:

a) **Paginação**: técnica em que o espaço de endereçamento virtual é dividido em blocos, denominados páginas, de tamanho e posição fixas, geralmente com um tamanho de 4kB, aos quais se associa um número. O SO efetua um mapeamento das unidades de alocação em endereços de memória, determinando também quais estão presentes na memória física e quais estão nos arquivos de troca;

b) **Segmentação**: técnica em que o espaço de endereçamento virtual é dividido em blocos de tamanho fixo ou variável, definidos por um início e um tamanho de acordo com o segmento, que pode ser um conjunto de variáveis ou um conjunto de rotinas de um programa, cuja posição também pode ser fixa ou variável. O SO mapeia estes blocos em endereços de memória, efetuando um controle de quais blocos estão presentes na memória física e quais estão nos arquivos de troca.

Estas duas técnicas só podem ser implementadas se o processador dispuser de mecanismos de deslocamento dos endereços referenciados pelo programa para as regiões de memória que efetivamente serão usadas, com um sofisticado mecanismo de endereçamento de memória mantido pelo SO e implementados como uma unidade de gerenciamento de memória ou Memory Management Unit (MMU).

Além dos mecanismos de paginação ou segmentação, a memória virtual exige a disponibilização de espaço nos dispositivos de armazenamento secundário para a criação de um ou mais arquivos de troca – os **swapfiles**.

A memória total de um sistema é, portanto, a soma de sua memória física (de tamanho fixo) com a memória virtual do sistema. O tamanho da memória virtual do sistema é definido por, basicamente, o menor valor dentre os seguintes:

- Capacidade de endereçamento do processador,
- Capacidade de administração de endereços do sistema operacional e
- Capacidade de armazenamento dos dispositivos de armazenamento secundário (unidades de disco).

Para implementar a memória virtual, a maioria dos SOs modernos utiliza as técnicas de paginação combinadas com a segmentação e os sistemas híbridos que utilizam a paginação combinada com segmentação, e são implementadas tabelas para realizar a conversão de endereços virtuais em endereços físicos. Estas tabelas, suportadas diretamente pelo hardware do sistema e mantidas pelo SO são, juntamente com os mecanismos de conversão de endereços, o ponto central destes esquemas de organização de memória.

Neste modelo de gerenciamento, o endereço virtual é composto de duas partes, um número da página virtual e um deslocamento dentro da página. O número da página virtual é usado como índice em uma tabela de páginas, ou seja, é somado ao endereço de base da tabela de páginas, mantido em um registrador qualquer do processador, obtendo-se uma referência para uma entrada da tabela que contêm o endereço real da moldura de página desejada. Somando-se o deslocamento, contido no endereço virtual, ao endereço da moldura de página, obtido da tabela de páginas, obtêm-se o endereço real completo.

Os mecanismos de memória virtual baseiam-se no fato de que os processos são armazenados em várias páginas da memória e que algumas destas páginas podem estar armazenadas em arquivos especiais denominados arquivos de troca (swap file). Quando um processo necessita acessar uma porção de seu código cuja página não esteja na memória real, ocorre um **pagefault**, ou seja, é detectada a falta de uma página que deverá ser carregada novamente para a memória principal.

As operações de substituição de páginas são lentas, pois envolvem uma interrupção do processamento e o acesso à memória secundária, ou seja, necessitam acessar dispositivos de entrada e saída, muito mais lentos do que a memória primária.

Se a página substituída for necessária em breve ou for preciso um número muito grande de substituições para execução dos programas ativos, ocorrerá a **hiperpaginação** (**hiperpaging** ou **thrashing**) ou seja, uma degradação significativa da performance do sistema, devido ao excesso de operações de troca de páginas, normalmente quando os programas estão exigindo bem mais memória física do que a instalada.

Desta forma, o SO precisa prover um mecanismo capaz de identificar quais páginas da memória não estão em uso (ou não serão necessárias logo) e movê-las para a área de swap do disco. O grande problema consiste, então, em determinar qual página será substituída ou copiada para os arquivos de troca, como forma de liberar espaço para aquela página que se tornou necessária. Um algoritmo ótimo de troca de páginas deveria ser capaz de identificar qual página não mais será utilizada ou estabelecer aquela que demorará mais a ser novamente utilizada, minimizando a ocorrência de **pagefaults** e, com isto, as operações de troca de páginas. Como não é possível realizar tal previsão, outras formas de se definir qual página será substituída são empregadas (DEITEL, 2005).

Os algoritmos que tratam deste problema são aqueles que implementam as estratégias de substituição (*replacementstrategies*) e são denominados algoritmos de troca ou algoritmos de substituição de páginas. Os algoritmos de substituição de páginas mais comuns são os que implementam uma fila simples ou utilizam alguma técnica mais complexa marcando as páginas que foram referenciadas ou modificadas nos últimos instantes, removendo aquelas que estão há mais tempo

sem serem utilizadas.

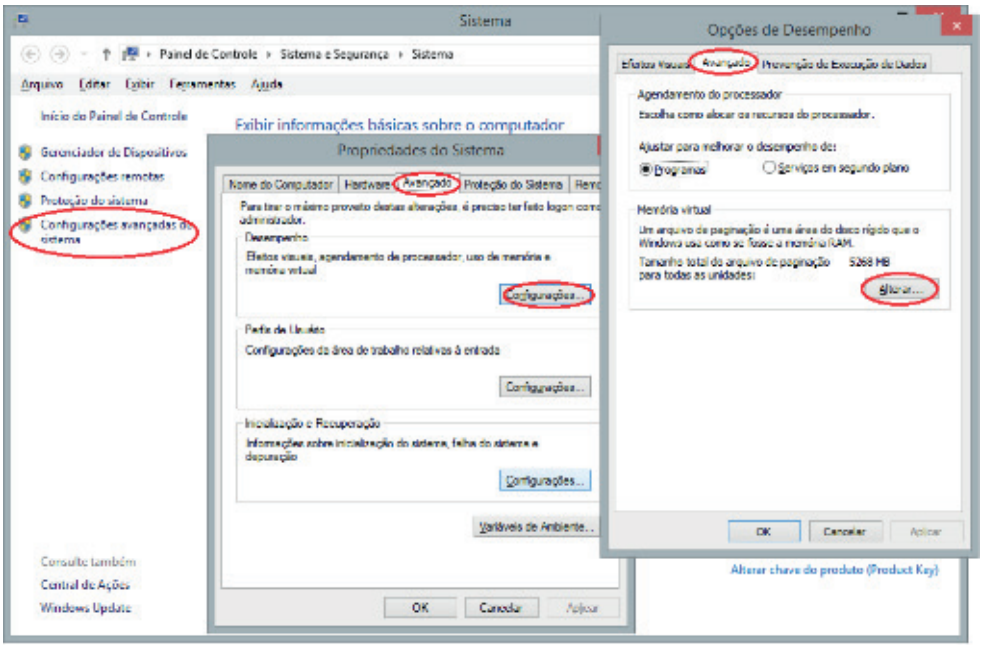
A técnica de página é utilizada para facilitar a movimentação de pequenas páginas (4Kb) entre a memória RAM e o arquivo do disco rígido, porém esta técnica gera um grande número de **pagefaults**, pois as páginas não têm o menor contexto com as demais. Para contornar essa deficiência, é associada a ela a técnica de segmentação, que divide o programa em segmentos cujos blocos têm grande relação. Assim, o algoritmo de substituição de página carrega e retira, cada vez, todas as páginas ou a maioria das páginas de um mesmo segmento de um programa.

4.2.1 Gerência de Memória no Windows

O Windows implementa um gerenciamento de memória com suporte à memória virtual, que é transparente ao usuário, mas é possível gerenciá-la conforme a necessidade. O arquivo de paginação da memória virtual no Windows pode ser encontrado na unidade C com o nome "pagefile.sys", normalmente com 2 ou 3GB de tamanho.

Para gerenciar e alterar as configurações do arquivo de paginação do Windows, entre no **Painel de Controle** e, em seguida, em **Sistema** ou clique com o botão direito do mouse sobre o ícone do **Meu Computador**, no gerenciador de arquivos e escolha a opção **Propriedades** ou, ainda, acesse diretamente, pressionando simultaneamente as teclas Windows (•) e Pause/Break. No menu do canto esquerdo, na parte inferior, clique em **Configurações avançadas do sistema**. Clique na aba **Avançado**, nas opções de Desempenho, clique novamente em **Avançado** e, na opção de Memória Virtual, pode-se alterar o gerenciamento da mesma, clicando em **Alterar**, como mostra a Figura 71.

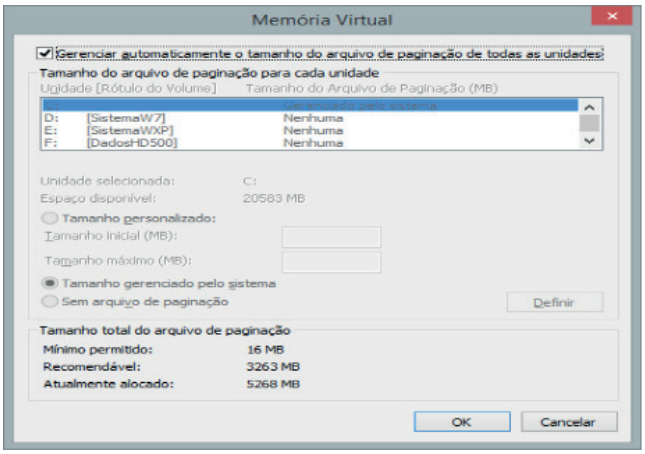
Figura 71 - Configuração da Memória Virtual no Windows



Fonte: Autores, 2017.

Isso vai abrir uma caixa onde é possível alterar a configuração, como mostra a Figura 72. Nela é possível escolher o seu tamanho, disco onde está localizado e até mesmo deixar sem arquivo de paginação, ou seja, sem memória virtual. Na janela apresentada, as configurações estão sendo gerenciadas de forma automática pelo Windows, que é o recomendado, mas pode-se alterar manualmente e personalizar.

Figura 72 - Configuração do arquivo de paginação do Windows



Fonte: Autores, 2017.

Para usuários que costumam abrir vários aplicativos simultaneamente, que consomem bastante memória ao mesmo tempo em uma máquina com pouca memória RAM (2 ou 4GB), aumentar esse tamanho do arquivo de paginação pode melhorar um pouco o desempenho, porém atribuir um valor muito alto, como 20 GB, não é uma boa opção, já que o Windows terá que fazer muitas trocas (swap), o que pode gerar uma degradação no sistema ainda maior.

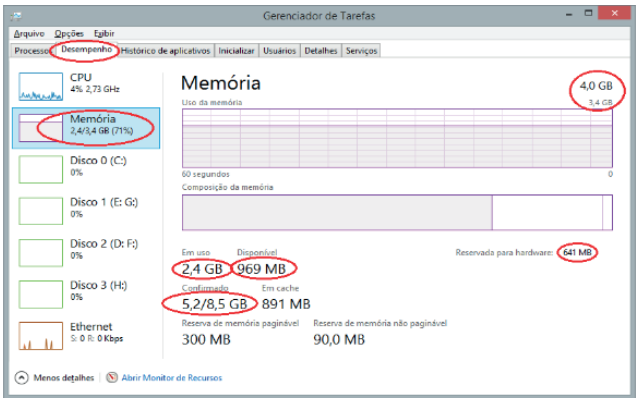
Em computadores que possuem mais de um disco rígido, mudar o arquivo de paginação para um outro HD é uma boa opção, pois aliviará o acesso ao disco na unidade em que está o SO e utilizará uma que estaria ociosa.

Mesmo em máquinas que possuem uma quantidade grande de memória RAM, como 16GB, desabilitar a memória virtual pode causar problemas se os programas em uso precisarem mais do que a quantidade de memória instalada. Se o computador estiver bastante lento e o LED da atividade do disco rígido estiver sempre aceso, indica que o sistema está fazendo swapping excessivo e seria interessante adicionar mais memória RAM no computador para melhorar o desempenho, pois nestes casos a memória RAM está sobrecarregada, gerando excessivos pagefault e o Windows está paginando demais, degradando o desempenho.

Para monitorar o uso de memória pelos programas, abra o **Gerenciador de Tarefas** do Windows e clique em **Desempenho** =>**Memória** e você visualizará, como mostra a Figura 73, a quantidade de memória instalada (4GB), a quantidade disponível para o sistema (Uso de memória: 3,4GB), a memória em uso (2,4GB – 71%), a quantidade disponível (969MB), a quantidade reservada para o hardware (641MB) e o tamanho de confirmação 5.2GB/8.5GB, que quer dizer que todos os programas em execução, neste momento, estão usando 5.2GB da memória virtual e destes, 2,4GB estão carregados na memória RAM. O restante está no disco, que

pode ainda receber mais 3,3GB pois o seu limite é 8.5 GB. Porém o detalhe é que neste caso, estamos utilizando 5.2GB de memória e só temos 3.4GB de memória RAM física para serem usados, ou seja, o sistema está fazendo swap e utilizando a memória virtual. Outro detalhe a ser observado nesta figura, é que existem 4GB instalados, porém só estão disponíveis 3,4 GB para o sistema, pois 641MB estão reservados para o hardware, provavelmente para a placa de vídeo onboard, que utiliza a memória do próprio computador.

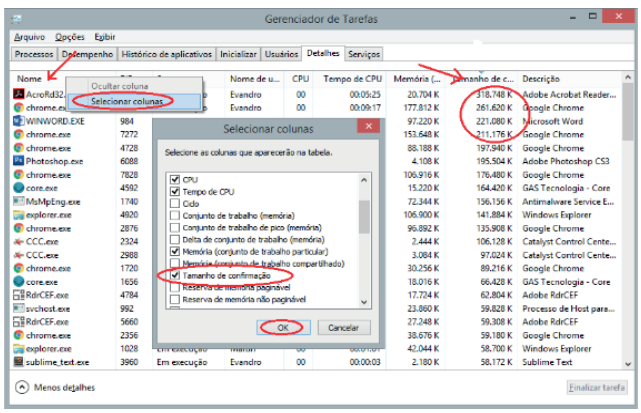
Figura 73 - Informações do uso de memória do Gerenciador de Tarefas do Windows



Fonte: Autores, 2017.

Para saber exatamente quanto cada programa está utilizando de memória e qual programa está consumindo mais memória, você deve clicar na aba **Detalhes** do Gerenciador de Tarefas do Windows, conforme mostra a Figura 74. Clique com o botão direito do mouse sobre o nome das colunas e escolha: **Selecionar colunas**; marque a opção **Tamanho de confirmação**; clique sobre o título da coluna Tamanho de confirmação para ordenar os dados em ordem decrescente e temos a relação em ordem dos programas que mais utilizam memória. Na Figura 74 é possível observar que os programas que mais estão usando memória no momento são: o Adobe Acrobat Reader (318,7MB), uma janela do Chrome (361,6MB), o Microsoft Word (221MB) e outra janela do Chrome (211,1MB). Observe que cada janela do Chrome é um processo independente.

Figura 74 - Informações dos programas que mais utilizam memória no Windows



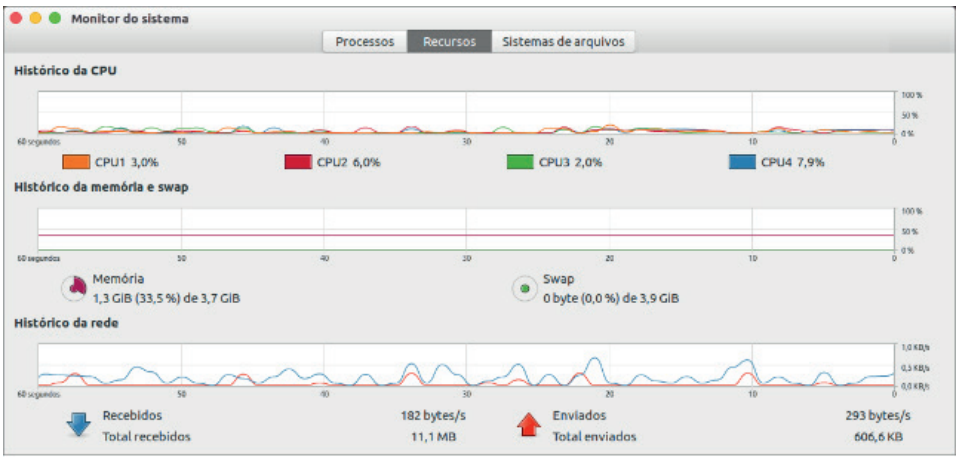
Fonte: Autores, 2017.

Adicionalmente pode-se ainda escolher outras colunas, como o Conjunto de trabalho de pico, que mostra o pico de uso de memória ou a coluna falhas de página, que mostra a quantidade de pagefaults, ou seja, a quantidade de páginas que o sistema tentou acessar e precisou que o sistema fizesse um swap, carregando a página do disco, do arquivo de paginação para a memória RAM. Se este número estiver aumentando, é sinal que o programa está exigindo muito swap.

4.2.2 Gerência de Memória no Linux

Para visualizar dados sobre o uso de memória no Linux, basta acessar a ferramenta de monitoramento dos recursos do sistema, clicando no botão principal e pesquisando por monitor para abrir a ferramenta **System Monitor** ou **Monitor do Sistema**, como mostra a Figura 75, onde é possível visualizar que estão em uso 1,3GB de 3,17GB de memória RAM disponíveis, enquanto que o arquivo de swap não está sendo usado e tem 3,9GB disponíveis.

Figura 75 – Monitor do Sistema

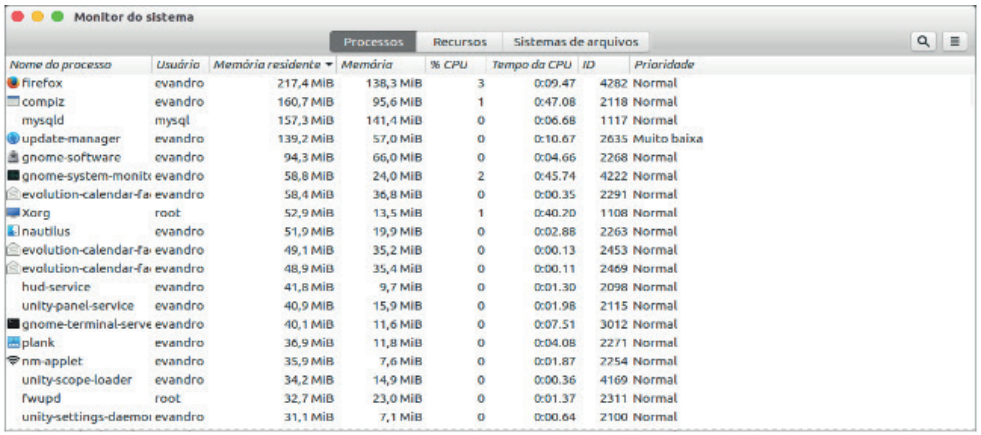


Fonte: Autores, 2017.

A memória virtual no Linux é o total do espaço de endereço reservado pelo SO para o processo para o seu código, dados e pilha. O tamanho da memória virtual relatado pelo Linux inclui todo o código, dados e espaço de pilha reservado para uso por um processo, em memória física e espaço de swap. Este valor inclui o espaço utilizado por todas as bibliotecas compartilhadas usadas pelo processo. O tamanho da memória residente, de um processo exibido pelo Linux, inclui apenas a quantidade de memória física do processo e as bibliotecas compartilhadas, que são referenciadas e em uso em um determinado momento.

Para visualizar os detalhes do uso de memória pelos processos é necessário clicar na aba Processos. A Figura 76 apresenta os processos e os recursos de memória utilizados. Clicando com o botão direito do mouse é possível escolher as colunas relativas ao uso de memória. Na figura, podemos visualizar o Firefox consumindo 138MB de memória e com memória residente (processo + bibliotecas compartilhadas) de 217,4MB.

Figura 76 - Detalhes do uso de recursos pelos processos no Linux

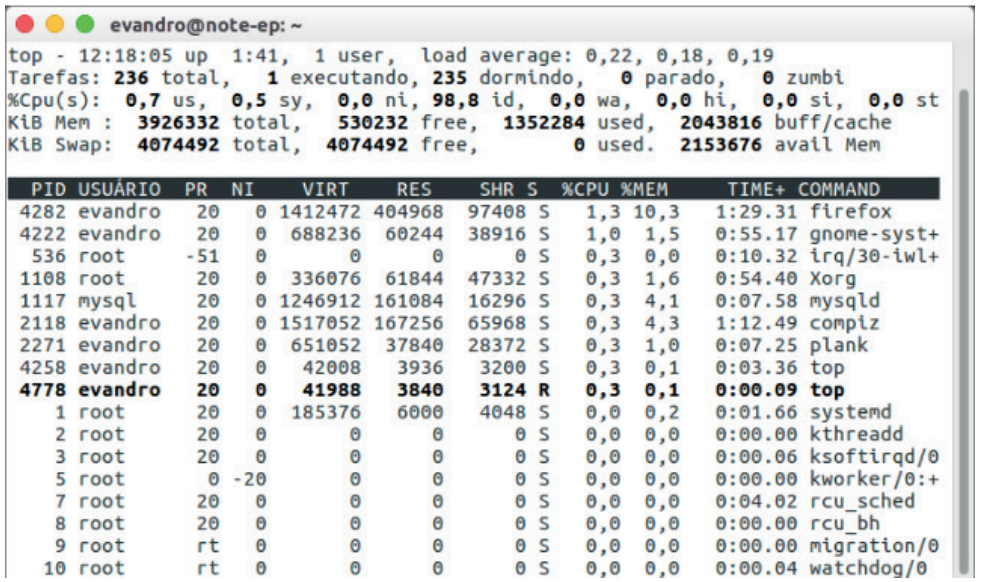


Nome do processo	Usuário	Memória residente	Memória	% CPU	Tempo da CPU	ID	Prioridade
firefox	evandro	217,4 MiB	138,3 MiB	3	0:09.47	4282	Normal
compiz	evandro	160,7 MiB	95,6 MiB	1	0:47.08	2118	Normal
mysqld	mysql	157,3 MiB	141,4 MiB	0	0:06.68	1117	Normal
update-manager	evandro	139,2 MiB	57,0 MiB	0	0:10.67	2635	Muito baixa
gnome-software	evandro	94,3 MiB	66,0 MiB	0	0:04.66	2268	Normal
gnome-system-monit	evandro	58,8 MiB	24,0 MiB	2	0:45.74	4222	Normal
evolution-calendar-fa	evandro	58,4 MiB	36,8 MiB	0	0:00.35	2291	Normal
Xorg	root	52,9 MiB	13,5 MiB	1	0:40.20	1108	Normal
nautilus	evandro	51,9 MiB	19,9 MiB	0	0:02.88	2263	Normal
evolution-calendar-fa	evandro	49,1 MiB	35,2 MiB	0	0:00.13	2453	Normal
evolution-calendar-fa	evandro	48,9 MiB	35,4 MiB	0	0:00.11	2469	Normal
hud-service	evandro	41,8 MiB	9,7 MiB	0	0:01.30	2098	Normal
unity-panel-service	evandro	40,9 MiB	15,9 MiB	0	0:01.98	2115	Normal
gnome-terminal-serve	evandro	40,1 MiB	11,6 MiB	0	0:07.51	3012	Normal
plank	evandro	36,9 MiB	11,8 MiB	0	0:04.08	2271	Normal
nm-applet	evandro	35,9 MiB	7,6 MiB	0	0:01.87	2254	Normal
unity-scope-loader	evandro	34,2 MiB	14,9 MiB	0	0:00.36	4169	Normal
fwupd	root	32,7 MiB	23,0 MiB	0	0:01.37	2311	Normal
unity-settings-daemo	evandro	31,1 MiB	7,1 MiB	0	0:00.64	2100	Normal

Fonte: Autores, 2017.

A visualização do uso de memória no Linux também pode ser obtida a partir do comando top, que exibe os processos em execução no sistema. A figura apresenta o resultado da execução deste comando, onde pode-se observar a quantidade total de memória KiB Mem: 3,9GB (3926332total), a quantidade livre: 530MB (530232 free), a quantidade em uso: 1,3 GB (1352284 used) a quantidade disponível para swap: 4 GB (4074495 total) e a quantidade livre para swap: 4 GB (4074495 total). Também é possível visualizar no topo dos processos o Firefox consumindo 10,3% da memória do sistema (404968 bytes de memória residente), sendo o processo que mais está usando memória nesse momento.

Figura 77 - Execução do Comando top




```
top - 12:18:05 up  1:41,  1 user,  load average: 0,22, 0,18, 0,19
Tarefas: 236 total,   1 executando, 235 dormindo,   0 parado,   0 zumbi
%Cpu(s):  0,7 us,  0,5 sy,  0,0 ni, 98,8 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem :  3926332 total,   530232 free,  1352284 used,  2043816 buff/cache
KiB Swap:  4074492 total,  4074492 free,        0 used.  2153676 avail Mem

  PID USUÁRIO   PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 4282 evandro   20   0 1412472 404968  97408 S   1,3 10,3   1:29.31 firefox
 4222 evandro   20   0  688236  60244  38916 S   1,0  1,5   0:55.17 gnome-syst+
  536 root     -51   0       0      0      0 S   0,3  0,0   0:10.32 irq/30-iwl+
 1108 root      20   0  336076  61844  47332 S   0,3  1,6   0:54.40 Xorg
 1117 mysql     20   0 1246912 161084  16296 S   0,3  4,1   0:07.58 mysqld
 2118 evandro   20   0 1517052 167256  65968 S   0,3  4,3   1:12.49 compiz
 2271 evandro   20   0  651052  37840  28372 S   0,3  1,0   0:07.25 plank
 4258 evandro   20   0   42008   3936   3200 S   0,3  0,1   0:03.36 top
 4778 evandro   20   0   41988   3840   3124 R   0,3  0,1   0:00.09 top
    1 root      20   0  185376   6000   4048 S   0,0  0,2   0:01.66 systemd
    2 root      20   0       0      0      0 S   0,0  0,0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S   0,0  0,0   0:00.06 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 kworker/0:+
    7 root      20   0       0      0      0 S   0,0  0,0   0:04.02 rcu_sched
    8 root      20   0       0      0      0 S   0,0  0,0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0,0  0,0   0:00.00 migration/0
   10 root      rt   0       0      0      0 S   0,0  0,0   0:00.04 watchdog/0
```

Fonte: autores, 2017.

5

GERÊNCIA DE ENTRADA E SAÍDA

INTRODUÇÃO

O controle da entrada e saída (E/S) de dados dos dispositivos é uma das funções principais de um SO, pois ele precisa promover o compartilhamento seguro dos recursos e permitir o seu uso de modo que os processos não tenham acesso direto aos dispositivos de entrada e saída, mas sim, por meio dos serviços implementados pelas chamadas de sistema, que permitem ler e escrever dados nos dispositivos.

Os dispositivos são os componentes que realizam alguma operação de entrada ou saída de dados, como o teclado, o mouse, o monitor, a placa de rede, a placa de vídeo, o disco rígido ou um pendrive, por exemplo. Para que cada um desses dispositivos funcione, é necessário um driver que deve ser instalado para que o SO possa se comunicar com esse dispositivo. O responsável pelo desenvolvimento do driver, normalmente é o próprio fabricante do dispositivo.

O controle de E/S é implementado por meio de uma estrutura de camadas de software e hardware que busca oferecer uma interface simples e confiável para o usuário e para a aplicação, escondendo das camadas superiores os detalhes das camadas inferiores que envolvem o hardware dos dispositivos.

A maior parte da gerência de E/S trabalha independentemente de dispositivo, permitindo a comunicação dos processos com qualquer tipo de periférico, proporcionando maior flexibilidade, confiabilidade e facilidade de programação e uso.

O sistema deve tornar a tarefa de gerenciamento de E/S o mais simples possível, garantindo a independência de dispositivos, pois o sistema deve se comunicar com qualquer tipo de dispositivo conectado ao computador, de modo que o usuário acesse os dispositivos sem se preocupar com os detalhes dos mesmos. A comunicação é feita por meio das bibliotecas e chamadas de sistema, através de parâmetros, escondendo do programador características associadas à programação de cada dispositivo.

5.1
GERÊNCIA DE DISPOSITIVOS

A função do SO nas operações de entrada e saída (E/S ou I/O) envolve prover uma interface para utilização dos dispositivos, de modo que a sintaxe das operações seja independente dos dispositivos, emitir comandos, atender as interrupções geradas e tratar os erros nas operações desses dispositivos.

As principais funções do gerenciamento de E/S são:

- Realizar funções que são comuns a todos os dispositivos;
- Criar uma unidade lógica de informações independente de dispositivos e repassá-la para os níveis superiores;
- Implementar todo o mecanismo de proteção de acesso aos dispositivos;
- Mapear o nome do dispositivo com seu respectivo driver, para que as camadas superiores acessem o dispositivo através deste nome;
- Controlar os dispositivos de Entrada/Saída de forma segura e confiável obtendo um maior compartilhamento;
- Realizar a bufferização, permitindo reduzir o número de operações de Entrada/Saída, utilizando-se de uma área de memória intermediária chamada ***buffer***.

Neste modelo, os aspectos e funções específicas de cada dispositivo ficam a cargo dos devicedrivers e normalmente, os erros são tratados nas camadas mais próximas ao hardware, sendo que alguns podem ser tratados independente de dispositivos.

O princípio básico de funcionamento dos dispositivos de E/S é a implementação de interrupções. Um pedido de interrupção (*IRQ – InterruptRequest*) é a forma pela qual os componentes de hardware requisitam a CPU para executarem o código do driver relacionado. Um IRQ é a sinalização de um pedido de interrupção de hardware.

Durante a execução de um programa ou processo, algumas operações podem estar sendo executadas nos dispositivos de E/S e elas não podem esperar, como por exemplo, os caracteres digitados no teclado e os movimentos do mouse, que precisam ser interpretados rapidamente pelo sistema, mesmo quando o computador está executando tarefas pesadas. Outro exemplo, de operações que precisam de atenção semelhante, são as transferências de dados recebidos através da placa de rede, operações de leitura e escrita nos HDs, dentre outras.

A placa de rede, por exemplo, não pode ficar simplesmente esperando que chegue a sua vez de ser atendida, pois isso retardaria a transmissão dos pacotes e poderia ocasionar a perda de dados vindos pela rede.

É neste momento que entram as requisições de interrupção por meio dos endereços de IRQ. Ao ser avisado por meio de qualquer um desses canais, o processador imediatamente para de executar o processo corrente, salva o contexto para continuar depois e dá atenção ao dispositivo, executando o código do driver associado a cada interrupção. O código do tratamento da interrupção vai atender o dispositivo vinculado a ele, realizando a tarefa necessária. Ao concluir o tratamento

da interrupção, o processador volta ao trabalho que foi interrompido, continuando do ponto em que estava, como se nada tivesse acontecido.

Cada endereço é uma espécie de sinal, que pode ser acionado a qualquer momento. Se não fossem as interrupções, o processador não seria capaz de ler as teclas digitadas no teclado ou os cliques do mouse e a transmissão de dados através da rede pararia toda vez que você abrisse qualquer programa.

Quando um programa de usuário emite uma chamada ao sistema relacionada aos dispositivos de E/S, esta é encaminhada ao driver apropriado. Para evitar que a CPU fique ocupada interrogando se o dispositivo terminou a operação de E/S (espera ociosa), o driver pede ao dispositivo que lhe sinalize quando isso ocorrer. Dessa forma, o SO poderá executar outras tarefas enquanto o programa que o chamou pedindo o serviço se encontra bloqueado. Ao final da operação, o controlador do dispositivo gera uma interrupção ao chip controlador de interrupção, para sinalizar à CPU que a tarefa foi finalizada.

Caso nenhuma outra interrupção esteja pendente ou em tratamento e nenhum outro dispositivo fez uma requisição, de maior prioridade no mesmo momento, a interrupção é tratada imediatamente. Caso contrário, ela é ignorada, e o dispositivo continuará emitindo sinal de interrupção. Após executar uma instrução, a CPU verifica as linhas de interrupções para saber se alguma interrupção foi sinalizada. Caso tenha sido sinalizada uma interrupção, o hardware grava os registradores da CPU, na pilha do programa que estava em execução, e carrega o contador de programa com o endereço da rotina referente à interrupção sinalizada.

Os dispositivos de E/S apresentam uma grande variedade, com diferentes velocidades e taxas de transferência de dados e podem ser divididos em um modo genérico como dispositivos de bloco e caractere.

Um dispositivo de bloco armazena as informações em blocos de tamanho fixo, cada qual com seu endereço. Cada bloco pode ser lido ou escrito, de maneira independente uns dos outros. Os dispositivos de caractere não utilizam estrutura de blocos nem posicionamento. Os discos e unidades de armazenamento são exemplos de dispositivos de bloco.

No dispositivo de caractere ele simplesmente recebe e manipula um fluxo de caracteres. O teclado, mouse e impressora são exemplos de dispositivos deste tipo.

Além desses dois modelos, os relógios (timers) não são dispositivos de blocos nem de caracteres. Os relógios só causam interrupções pré-programadas com um determinado tempo.

5.1.1 Módulos de I/O

O modelo de arquitetura de Von Neumann define o computador como o conjunto de processador, memória e dispositivos de entrada e saída, interconectados por meio de vários barramentos especializados: o barramento de dados, o barramento de endereços e o barramento de controle. Porém, diferentemente do que se poderia imaginar, os dispositivos periféricos não são conectados diretamente a tais barramentos, mas, sim, a módulos de I/O, que por sua vez são ligados aos barramentos do sistema. As razões para a conexão ser dessa maneira são várias, dentre elas:

- Existem inúmeros tipos de periféricos, com diferentes formas de operação, sendo impraticável implantar no computador uma lógica que permitisse a operação com todos ou mesmo uma grande parte destes dispositivos;
- A velocidade de operação dos dispositivos periféricos é bem menor que a da memória ou do processador.

Desta forma é bem mais conveniente implementar módulos de I/O que atuem como conexões mais genéricas para os diferentes dispositivos periféricos, permitindo o uso de estruturas padronizadas para ligação com a memória e processador e, por esse motivo, os módulos de I/O são chamados de interfaces.

A conexão com o barramento do processador geralmente é uma interface de alto nível, ou seja, uma conexão baseada em comandos, mais adequada à operação com um processador. Já a conexão com o periférico é normalmente uma interface de baixo nível, baseada em sinais elétricos e um protocolo dedicado próprio, que exige tratamento especializado. Desta forma, as particularidades de cada tipo de periférico ficam isoladas do sistema principal, facilitando o desenvolvimento dos programas que utilizarão estes dispositivos.

Esta estrutura permite que um único módulo de I/O controle mais de um periférico, geralmente do mesmo tipo, como por exemplo os controladores de unidades de disco SATA que podem conectar vários discos rígidos, SSDs e drives de CD, DVD e **BluRay**.

TERMO DO GLOSSÁRIO: Cache: BluRay: é um formato de disco ótico para vídeo e áudio de alta definição e armazenamento de dados de alta densidade.

Outra estrutura possível para a implementação dos módulos de I/O são os canais de I/O (I/O channels). Os canais de I/O são sistemas computacionais de propósito especiais destinados ao tratamento de entrada e saída de forma independente do processador do sistema computacional (DEITEL, 2005).

Esta forma de implementação opera com múltiplos barramentos de alta velocidade, podendo acessar o armazenamento primário de forma independente, proporcionando grande desempenho, pois as operações de I/O são realizadas paralelamente ao processamento.

5.1.2 Operação de Módulos de I/O

De acordo com a forma como a CPU se envolve na transferência dos dados, os módulos de I/O podem ser implementados de três maneiras básicas:

- I/O programado;
- I/O com Interrupções;
- I/O com Acesso Direto à Memória (DirectMemory Access - DMA).

I/O Programado

No modo I/O programado, os dados são trocados diretamente entre o processador e o módulo de I/O e o processador deve executar um programa que verifique o estado do módulo de I/O, preparando-o para a operação, enviando o comando que deve ser executado e aguardando o resultado do comando para então efetuar a transferência entre o módulo de I/O e algum registrador do processador. Neste modo, o processador não pode executar outras tarefas enquanto aguarda a operação do módulo de I/O, e deverá permanecer em espera durante toda a operação executada no periférico.

O módulo de I/O se relaciona com o processador por meio de comandos ou códigos de controle transformados em sinais enviados ao módulo, que indica qual operação deverá ser realizada. Normalmente os comandos do módulo de I/O têm equivalência direta com as instruções do processador, facilitando sua operação integrada.

Como é possível para um módulo de I/O controlar mais de um dispositivo periférico, são associados endereços para cada um destes periféricos, de forma a permitir sua operação individualizada. Existem duas formas para a interpretação destes endereços por parte dos módulos de I/O quando há o compartilhamento de barramentos do sistema:

a) **Mapeada em Memória**: o módulo de I/O opera dentro do espaço de endereçamento da memória, usando um conjunto de endereços reservados. Assim o processador trata os registradores de status e dados do módulo de I/O como posições normais de memória utilizando operações de leitura e escrita;

b)**Mapeada em I/O**: existe um espaço de endereçamento independente para os dispositivos de I/O, nos quais as portas de I/O passam a ser acessíveis apenas por operações especiais de I/O. Utilizando esta forma há uma maior simplicidade e a disponibilidade de um maior conjunto de instruções, embora haja um reduzido espaço em memória devido à reserva de endereços para portas de I/O. Com esse modo, há uma maior segurança nas operações e um maior espaço de endereçamento. Os microcomputadores IBM PC compatíveis são um exemplo desta utilização, no caso, para enviar-se dados ao monitor de vídeo padrão do sistema devem ser utilizados os endereços de I/O na faixa de 03D0h a03DFh, enquanto que o acesso às placas de som tipo SoundBlaster devem usar o endereço 0220h.

I/O com interrupções

Para contornar o problema da espera ociosa do processador por operações nos dispositivos periféricos, pode ser utilizado o mecanismo das interrupções, ou seja, o I/O por meio de interrupções. A interrupção permite que um dispositivo ganhe a atenção imediata do processador para concluir sua tarefa (DEITEL, 2005).

Assim, quando o processador envia um comando para o módulo de I/O, o mesmo pode passar a executar outra tarefa, sem a necessidade de monitorar o módulo acionado. Quando a operação for concluída, o módulo de I/O interrompe o processador, acionando uma interrupção para requisitar a troca de dados com o processador. O processador executa então a troca de dados, liberando o módulo de I/O e retomando o processamento anterior (STALLINGS, 2002).

A operação de I/O com interrupções envolve:

a) O processador envia um comando ao módulo de I/O, que a realiza de modo independente;

b) O processador passa a executar outra tarefa ou um outro processo;

c) Ao finalizar a operação requisitada, o módulo de I/O sinaliza uma interrupção para o processador;

d) Ao término da instrução em execução, o processador verifica a sinalização de uma interrupção, identificando o dispositivo que a originou e sinaliza enviando um sinal de reconhecimento;

e) O processador salva o contexto da tarefa atual na pilha, preservando assim o conteúdo do contador de programa e demais registradora;

f) O processador carrega o endereço da rotina de tratamento da interrupção detectada. Tais endereços são armazenados em uma região pré-determinada da memória, denominada vetor de interrupções;

g) A rotina de tratamento da interrupção é executada e os dados são lidos do módulo de I/O para um registrador do processador e depois transferidos para uma área de memória apropriada;

h) Finalizada a rotina de tratamento da interrupção, o processador restaura o contexto do processo que foi interrompido;

i) O processador retorna ao processamento da tarefa no ponto em que foi interrompida.

A operação de dispositivos de I/O utilizando interrupções permite que o processador continue executando enquanto o módulo de I/O realiza a operação solicitada, melhorando o desempenho do sistema.

I/O com Acesso Direto à Memória (DMA)

As técnicas de I/O programado e I/O com interrupções possuem a desvantagem de terem uma limitação da velocidade de transferência de dados, pois o processador fica responsável por transferir os dados do módulo para a memória.

Para solucionar este problema foi desenvolvida uma outra técnica denominada I/O com acesso direto à memória ou DMA (Direct Memory Access). Esta técnica utiliza uma única interrupção para efetuar a transferência de um bloco de dados diretamente entre o periférico e a memória primária, sem o envolvimento do processador, reduzindo a quantidade de operações necessárias e otimizando o processo.

Para isso, é necessária a existência de um módulo adicional, denominado controlador de DMA, cuja operação é (STALLINGS, 2002):

1. O processador envia comando de leitura ou escrita para controlador de DMA;
2. O processador continua seu trabalho enquanto DMA efetua a transferência de dados do dispositivo de I/O;
3. Ao final da operação, o controlador de DMA aciona uma interrupção para sinalizar o término da operação;
4. O processador pode executar a rotina de tratamento da interrupção, processando os dados lidos ou produzindo novos dados para serem escritos.

Este método é relativamente mais rápido do que o I/O programado ou I/O via interrupções pois utiliza apenas uma única interrupção, liberando o processador para executar outras tarefas e a transferência dos dados ocorre em bloco diretamente entre o periférico e a memória.

5.1.3 Controladores de Dispositivos

As unidades de I/O constituem-se de um componente eletrônico e um mecânico. O elemento eletrônico é chamado de controlador de dispositivo ou adaptador. Nos computadores pessoais, o controlador de dispositivo aparece sob a forma de uma placa de circuito impresso, com um conector que pode ser plugado a outros dispositivos.

Cada controlador de dispositivo tem seus registradores. Esses registradores são usados para se comunicar com a CPU. Por meio da escrita nesses registradores do controlador, o SO pode comandar o dispositivo. Além dos registradores, os dispositivos contêm *buffers*, nos quais o SO lê e escreve.

A CPU comunica-se com os registradores do controlador e com os *buffers* do dispositivo por meio dos registradores, associados a um número de porta de I/O. Usando uma instrução, a CPU pode ler o registrador do controlador e armazenar o resultado no seu registrador. A mesma pode escrever o conteúdo do registrador da CPU para o registrador de controle.

5.1.4 Drivers do dispositivo

Cada controlador tem alguns registradores do dispositivo, utilizados para dar comandos. O número de registradores do dispositivo e a natureza dos comandos variam de dispositivos para dispositivos. Por exemplo, um driver de mouse deve aceitar informações do mouse, tais como o quanto se moveu e qual botão foi pressionado. Por outro lado, o driver do disco deve saber sobre o setor, trilhas, cilindros e cabeçotes. Como esses drivers serão bastante diferentes, cada dispositivo de I/O, ligado ao computador, precisa de algum código especifico do dispositivo para controlá-lo.

Esse código que controla os dispositivos é chamado de driver do dispositivo e normalmente é desenvolvido pelo fabricante, juntamente com o dispositivo. Como cada SO precisa de seus próprios drivers dos dispositivos, os fabricantes desenvolvem e fornecem esses drivers para os sistemas operacionais mais populares, pois sem os mesmos os dispositivos ficam inoperantes. Os drivers devem ser instalados no SO e passam a fazer parte dos mesmos.

Na maioria dos SOs, especialmente no Linux, os device drivers mais importantes e os mais genéricos, que são utilizados largamente pelo computador, normalmente são compilados e fazem parte do núcleo do sistema (kernel) e os utilizados com menos frequência, ou que não exigem uma demanda muito grande, são carregados como módulos. Os drivers que não estão disponíveis inicialmente com o SO, que são drivers disponibilizados por terceiros, são módulos que são incorporados posteriormente ao SO.

5.1.5 Dispositivos de Entrada/Saída

Os dispositivos de entrada e saída são os responsáveis pela comunicação entre o computador e o mundo externo, e podem ser somente para entrada de dados, tais como o teclado e o mouse; somente para saída de dados, como a placa de vídeo e as impressoras; ou para entrada e saída de dados, como as placas de rede, discos e unidades de armazenamento.

Os dispositivos podem ser Estruturados ou Não Estruturados. Os dispositivos estruturados armazenam as informações em blocos de tamanho fixo, cada um com um endereço, cujo tamanho do bloco pode variar entre 128 e 1.024 bytes e cujos blocos podem ser lidos ou gravados de forma independentes. Os discos são exemplos de dispositivos estruturados.

Os dispositivos estruturados podem ser de:

a) Acesso Direto: os blocos de dados podem ser recuperados diretamente por meio de um endereço, como por exemplo, nos discos magnéticos, em que pode ser lido um setor qualquer do disco;

b) Acesso sequencial: para acessar um bloco de dados, o dispositivo deve percorrer sequencialmente o meio de armazenamento à procura do bloco, exemplo, fita magnética, CD, DVD.

Os dispositivos não-Estruturados podem enviar ou receber uma sequência de

caracteres sem estar estruturado no formato de um bloco. Esta sequência de caracteres não é endereçável, não podendo haver operações de acesso ao dado após a transmissão, como por exemplo em terminais, impressoras e interfaces de rede. Esses dispositivos de caractere são tão rápidos quanto possível para escrever e ler dados do seu hardware e, portanto, não necessitam de um buffer.

Já um dispositivo de bloco, devido à sua característica de acesso aleatório, necessita de um buffer, pois quando um programa utiliza um dispositivo de bloco, as operações de escrita e leitura estão sendo realizadas no buffer do dispositivo e esse, por sua vez, se encarrega de fazer a mudança fisicamente, no momento adequado.

5.2
DISCOS MAGNÉTICOS

Pela importância do uso dos discos magnéticos como principal dispositivo de memória secundária, utilizados também para implementar a memória virtual, o acesso aos dados nos mesmos merece uma atenção especial, principalmente os aspectos como desempenho e segurança devem ser considerados.

O Disco é composto de diversos discos sobrepostos, unidos por um mesmo eixo vertical, girando em velocidade constante. Cada disco compõe-se de trilhas concêntricas, que estão divididas em setores. As trilhas dos diferentes discos que ocupam a mesma posição vertical formam um cilindro. Para cada superfície de um disco existe uma cabeça de leitura e gravação. O conjunto de cabeças é preso a um braço que se movimenta entre os vários cilindros no sentido radial.

O tempo necessário para ler/gravar um bloco de dados é dado em função de três fatores:

a) **Tempo de busca (seek)**: tempo gasto para mover o braço até o cilindro onde o bloco se encontra;

b) **Tempo de latência**: tempo de espera até que o setor desejado se posicione sob a cabeça de leitura/gravação.

c) **Tempo de transferência**: tempo necessário para ler ou gravar o bloco para o setor.

Em muitos casos, o fator que leva mais tempo é o tempo de busca, sendo necessárias algumas estratégias para minimizar esse tempo de busca.

Os SOs empregam uma técnica baseada em um **algoritmo de escalonamento do braço do disco**, que implementa uma técnica de buffer das operações de leitura e escrita, guardando a operação para ser efetivada somente quando, o braço do disco, estiver mais próximo do setor desejado. Neste modelo, as requisições são enfileiradas e são atendidas na ordem da movimentação do braço do disco e não na ordem das requisições.

Outras técnicas também são implementadas pelos SOs, como copiar parte dos dados do disco que estão sendo constantemente requisitados para a memória principal, implementando uma cache de disco, para poder minimizar os tempos de procura e latência. Um exemplo disso acontece quando copiamos algumas fotos de um cartão de memória para uma unidade de disco. Na primeira operação, os dados dos arquivos são lidos, um de cada vez do cartão de memória e gravados em uma pasta (diretório) no disco rígido. Se resolvermos copiar para outra pasta, ou para outro disco, para fazer uma cópia adicional, os dados não são mais lidos do cartão de memória e são copiados de forma muito mais rápida pois os dados já estavam na memória cache de disco, na memória principal. Porém, se a quantidade de fotos que queremos copiar é muito grande, não é feita uma cópia em cache e os dados têm que ser novamente lidos da unidade de origem.

Outra técnica que pode ser usada nos disco é a implementação de RAID (Redundant Arrayof Inexpensive Disks), que envolve a técnica de conectar dois ou mais discos para que os mesmos trabalhem como se fosse um só, aumentando o desempenho ou aumentando a segurança, no caso da ocorrência de problemas com um deles.

A técnica RAID pode ser implementado com os seguintes níveis:

a) **Nível 0**: também chamado de "Striping" ou "Fracionamento", este nível é usado quando se deseja melhorar a performance do computador, pois nele os dados são divididos em segmentos, que têm um tamanho específico de bloco, e divididos entre os discos, proporcionando uma grande velocidade tanto na gravação quanto na leitura de informações, pois os dados são fragmentados e gravados simultaneamente em cada um dos discos. Assim, quanto mais discos forem usados, mais rápida é a distribuição dos dados. Esse nível não aceita falhas, já que cada parte dos dados está em um disco, todos os discos são necessários para a leitura dos dados;

b) **Nível 1**: também chamado de "*Mirroring*" ou "*Espelhamento*", este nível é usado quando se deseja garantir a integridade dos dados, pois nele os dados são clonados para outro dispositivo. Deste modo, se um computador com dois discos rígidos utilizar este nível, ao gravar dados em um disco automaticamente será gravado no outro disco, tornando-os cópias idênticas. Se houver falha em um disco ou perda de dados, basta recuperar o setor defeituoso copiando os arquivos contidos do outro disco;

c) **Nível 3**: para utilizar esse nível, pelo menos 3 discos são necessários, pois os dados são divididos entre dois discos ou mais discos e o terceiro fica responsável pelas informações de paridade. Assim, todos os bytes têm sua paridade gravada neste disco extra, permitindo identificar onde o erro ocorreu, garantindo a integridade dos dados. Em caso de recuperação, a sua montagem é via software;

d) **Nível 4**: utiliza a mesma técnica do nível 3, porém em caso de falhas os dados são reconstruídos em tempo real por meio da utilização da paridade, calculada a partir dos outros discos, sendo que cada um pode ser acessado de forma independente;

e) **Nível 5**: oferece mais desempenho em relação ao nível 4, pois a paridade encontra-se distribuída nos discos, fazendo com que a gravação seja mais rápida, já que não é necessário acessar um único disco a cada gravação. Esse é o nível de RAID mais utilizado;

f) **Nível 6**: semelhante ao nível 5, porém esse modelo usa o dobro de bits de paridade, garantindo a integridade para o caso de até dois discos rígidos falharem ao mesmo tempo. RAID 6 pode ser utilizado para sistemas de missão-crítica, onde a confiabilidade dos dados é essencial;

g) **Nível 0+1**: é o nível mais caro de ser implementado, pois exige no mínimo 4 discos rígidos, esse nível implementa rapidez e integridade dos dados pois ele

combina os níveis 0 e 1 onde os dados são divididos entre os discos para melhorar o rendimento, mas também utilizam outros discos para duplicar as informações.

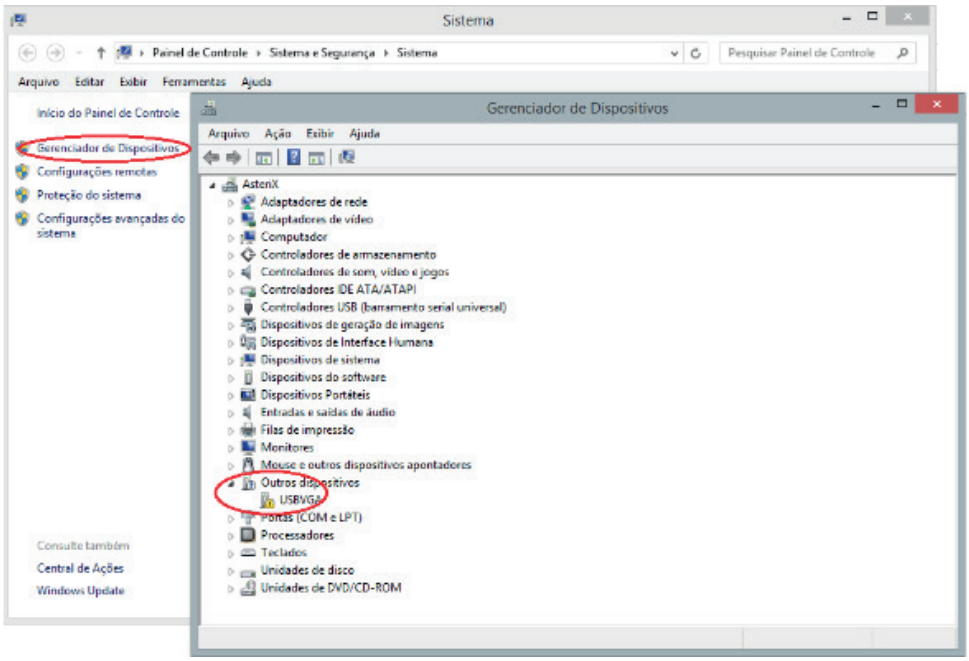
5.3
GERÊNCIA DE ENTRADA E SAÍDA NO WINDOWS

O Windows implementa a gerência de entrada e saída de forma transparente e automática para o usuário, que precisa apenas instalar o driver do dispositivo para ter acesso a todos os recursos que o mesmo proporciona, sem a necessidade de nenhuma configuração de endereçamento de memória, IRQ, canal de I/O ou DMA.

O Windows disponibiliza um painel com todas as informações do hardware e dos recursos do dispositivo, para visualização se o mesmo está funcionando ou não.

Para verificar as configurações dos dispositivos de entrada e saída do Windows, acesse o **Painel de Controle** e, em seguida, clique em **Sistema** ou clique com o botão direito do mouse sobre o ícone do Meu Computador no gerenciador de arquivos e escolha a opção **Propriedades** ou, ainda, acesse diretamente pressionando simultaneamente as teclas Windows(•) e Pause/Break. No menu do canto esquerdo, na parte inferior, clique em **Gerenciador de Dispositivos**, Uma lista com todos os dispositivos cujos respectivos drivers tenham sido instalados será exibida. Caso algum driver não tenha sido adequadamente instalado, um triângulo amarelo de alerta será exibido, como mostra a Figura 78.

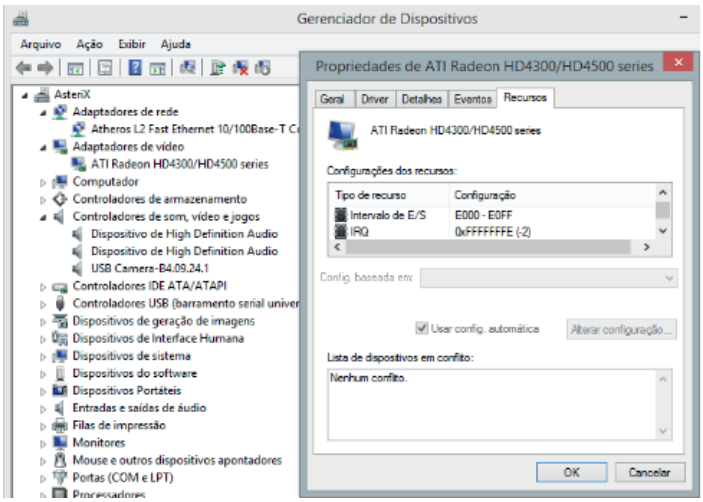
Figura 78 – Gerenciador de Dispositivos



Fonte: Os Autores, 2017

Clicando sobre os dispositivos é possível visualizar os detalhes do mesmo, como as informações do driver dos recursos de hardware em uso como mostra a Figura 79, onde é possível visualizar os recursos do intervalo de E/S em uso (E000 - E0FF) e a interrupção (IRQ -2).

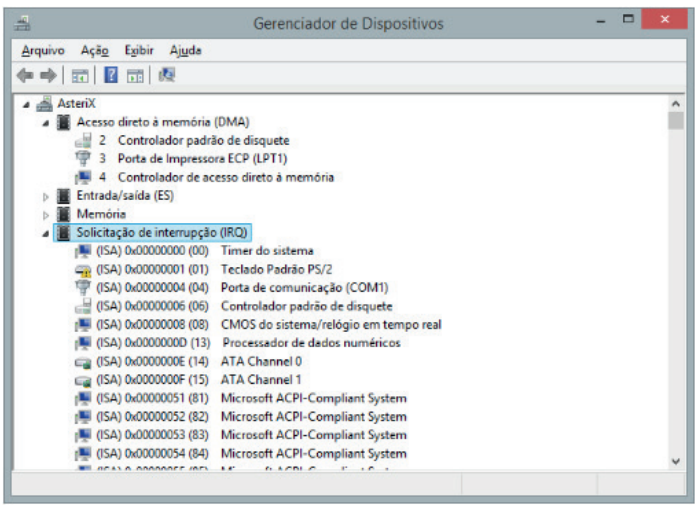
Figura 79 – Propriedades dos dispositivos



Fonte: Autores, 2017.

No Windows também é possível ver todos os dispositivos de acordo com o recurso de hardware em uso, como mostra a Figura 80, bastando clicar no menu **Exibir =>Recursos por tipo**. Na figura podemos visualizar os dispositivos mais antigos que usam DMA (disquete e impressora paralela LPT1) e os dispositivos que usam IRQ: Timer do sistema (IRQ 00), teclado padrão PS/2 (IRQ 01 – com problema no driver indicado pelo triângulo amarelo), Porta de comunicação COM1 (IRQ 04), dentre outros.

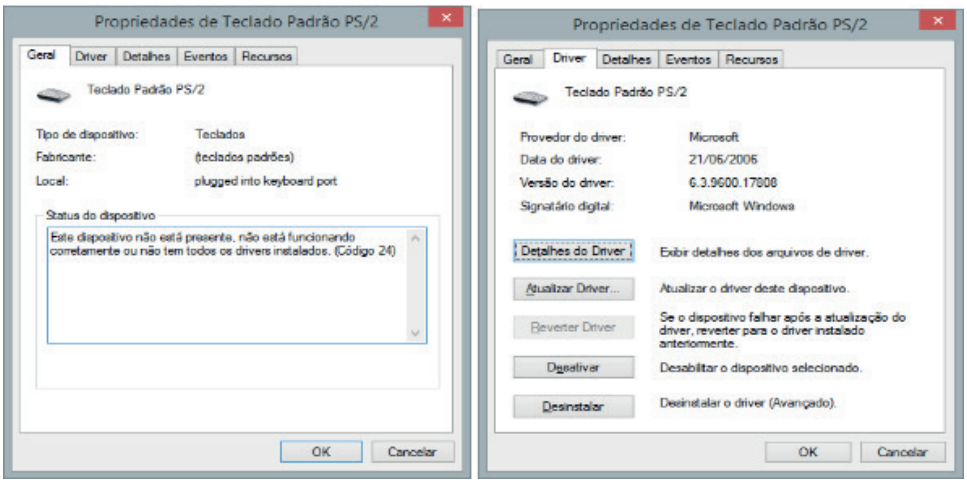
Figura 80 – Propriedades dos recursos por tipo



Fonte: Autores, 2017.

O caminho para corrigir um erro de driver do dispositivo com problema, é dar um duplo clique sobre o dispositivo ou clicar com o botão direito sobre ele e acessar suas propriedades, como mostra a Figura 81. Clicando sobre a aba ***Driver*** e no botão ***Atualizar Driver*** é possível atualizá-lo.

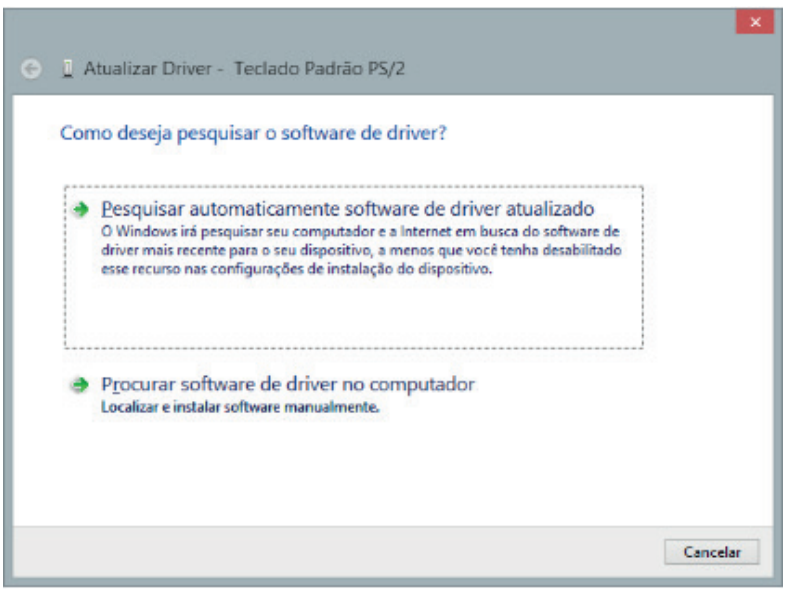
Figura 81 – Propriedades de um dispositivo com problema



Fonte: Autores, 2017.

Solicitando a atualização, abrirá uma janela na qual é possível escolher se a atualização será pesquisando os driver,s de forma automática, ou procurando os mesmos em uma pasta, ou em um CD de instalação, como mostra a Figura 82. Nem sempre os dados são encontrados de forma automática e, nesses casos, é necessário o CD de instalação ou os arquivos oriundos de um *download* do fabricante do dispositivo.

Figura 82 – Propriedades de um dispositivo com problema



Fonte: Autores, 2017.

5.4 GERÊNCIA DE ENTRADA E SAÍDA NO LINUX

No Linux, todos os dispositivos são tratados como se fossem um arquivo. Ao enviar ou ler os dados para esse arquivo, a comunicação e os bytes, são repassados imediatamente ao driver do dispositivo. Ao passo que, em outros SOs, tais como o Windows, a representação dos dispositivos de armazenamento é feita por letras de unidades de disco e a localização dos demais dispositivos e forma como eles são controlados pelo sistema é ocultado, no SO Linux, todos os dispositivos têm uma correspondência no diretório **/dev** exceto as interfaces de rede, que são de fato arquivos.

Enviar alguns bytes para um desses arquivos do **/dev** fará com que esses bytes sejam enviados para o dispositivo. Quem provê essa funcionalidade é, em parte, o SO e o seu driver de dispositivo. O SO descobre qual driver controla determinado dispositivo, pois a instalação do driver cria uma representação com números de identificação do dispositivo sob a forma de arquivo para cada dispositivo, e esses números são utilizados como a identidade virtual do hardware. Junto com o tipo de dispositivo o SO é capaz de identificar qual driver está controlando o dispositivo e faz a ligação entre os dois.

O papel de um módulo (driver) é estender a funcionalidade do kernel. Ainda mais, o papel do dispositivo driver é prover mecanismos, e não a política. Por exemplo, o driver de floppy é livre de políticas: seu papel é apenas mostrar o disquete como um contínuo array de blocos de dados. Níveis mais altos provêem as políticas, tais como quem pode acessar o driver de floppy, se o driver é acessado diretamente ou via um filesystem e se os usuários podem montar o filesystem no driver. Um driver é flexível, oferecendo acesso ao hardware sem adicionar restrições.

Um módulo (driver) no Linux é executado no espaço do kernel enquanto que os aplicativos são executados no espaço do usuário. No Linux, o kernel é executado no mais alto nível, chamado de **modo supervisor**, onde tudo é permitido. Os aplicativos são executadas em um nível mais baixo, chamado **modo usuário**, onde o processador controla o acesso direto ao hardware e o acesso não autorizado à memória.

Uma das boas características do Linux é a habilidade de oferecer um conjunto de facilidades oferecido pelo kernel, como a possibilidade de adicionar código ao kernel em tempo de execução por meio dos **módulos**.

O Linux prevê três tipos de dispositivos fundamentais: charmodule, blockmodule e networkmodule, mas existem ainda outros tipos de módulos drivers. Cada dispositivo USB é controlado por um módulo USB que trabalha com o subsistema USB, porém o próprio dispositivo é usado no sistema como um char device (se for uma porta serial USB), um blockdevice (se for um leitor de cartão de memória USB) ou um networkdevice (se for uma interface USB Ethernet ou Wireless).

Os arquivos de dispositivos se referem aos dispositivos (físicos ou virtuais) do sistema, como o disco rígido, a placa de vídeo, o monitor, ou teclado. Um exemplo

de dispositivo virtual é o teclado, representado pelo **/dev/console**.

Um arquivo de dispositivo está associado com um dispositivo de hardware, como por exemplo, um HD com o **/dev/sda** ou com uma partição do disco, como por exemplo, /dev/sda2. É importante distinguir os **arquivos de dispostivos** dos **arquivos de drivers**. Os arquivos de dispositivos são simplesmente pontos de encontro usados para comunicação do SO com os drivers. Eles não são os drivers em si. Um driver de dispositivo, também chamado de módulo de dispositivo, cuida dos detalhes específicos de gerenciamento direto do dispositivo de hardware. A Figura 83 apresenta um exemplo dos arquivos de dispositivos, obtido a partir de uma listagem dos dispositivos com o comando **ls –l/dev**

Figura 83 - Exemplos de arquivos de dispositivos

Nome	Tipo	Major	Minor	Descrição
/dev/console	char	5	1	Console (terminais não-gráficos)
/dev/hda	block	3	0	Primeiro disco IDE
/dev/hda2	block	3	2	Segunda partição primária, primeiro disco IDE
/dev/hda3	block	3	67	Terceira partição primária, segundo disco IDE
/dev/lp1	char	6	1	Impressora paralela
/dev/null	char	1	3	Nulldevice
/dev/rtc	char	10	135	Clock de tempo-real
/dev/sda	block	3	64	Primeiro disco SATA
/dev/sda1	block	3	2	Primera partição, primeiro disco SATA
/dev/sdb	block	8	0	Segundo disco SATA
/dev/ttyp0	char	3	0	Terminal
/dev/ttyS0	char	4	64	Primeira porta serial

Fonte: Autores, 2017.

Os arquivos de dispositivo vêm com duas identificações básicas:

a) O tipo do arquivo de dispositivo: se é de bloco (HD ou CD-ROM), de caracter (para hardware do tipo placas de som, teclado e mouse) ou interface de rede (etho);

b) Um par de números: o principal e o secundário (“major number” e “minornumber“). O primeiro identifica o tipo do dispositivo e o segundo número identifica o dispositivo específico dentre um grupo de dispositivos que compartilham o mesmo número principal e o mesmo controlado. Por exemplo, os discos gerenciados pelo mesmo controlador de disco têm o mesmo número principal e diferentes números secundários. Os dispositivos do tipo bloco e do tipo char têm numerações independentes, de modo que o dispositivo de bloco (3,0) é diferente do dispositivo char (3,0).

Os arquivos de dispositivos em /dev possuem essas identificações. Estes números de dispositivo principal e secundário identificam o dispositivo para o kernel, onde este utiliza esses números para associar referências do arquivo de dispositivo ao driver correspondente. O número de dispositivo principal identifica o driver com o qual o arquivo está associado (ou seja, o tipo de dispositivo). O número de dispositivo secundário identifica qual a instância em particular de um dado tipo de dispositivo deve ser tratada.

O núcleo (kernel) do Linux contém drivers de dispositivos que gerenciam a interação com os dispositivos e o kernel é independente dos dispositivos. A existência de uma camada de driver ajuda a manter o Linux independente dos dispositivos.

Os drivers de dispositivos fazem parte do kernel e, portanto, não são processos de usuários, porém um driver pode ser acessado tanto pelo kernel quanto pelo processo do usuário, por meio de arquivos de dispositivos especiais que residem no diretório **/dev**. O kernel transforma operações nesses arquivos em chamadas para o código do driver. Somente drivers de dispositivos para uso com Linux podem ser instalados em um sistema Linux. Os drivers para outros SOs, como Windows, por exemplo, não funcionarão no Linux.

5.4.1 Montar um dispositivo

Para acessar os dados que estão em um dispositivo no Linux é necessário aplicar o conceito de montar o dispositivo, de modo que, quando se coloca um CD ou pendrive no computador, por exemplo, é necessário montar este dispositivo para acessá-lo, por meio do comando mount. A sintaxe básica do comando mount é:

mount /dev/nome_do_dispositivoponto_de_montagem

De forma semelhante é usado um comando para "desmontar" o mapeamento do dispositivo, e sua sintaxe básica é:

umount /dev/nome_do_dispositivo

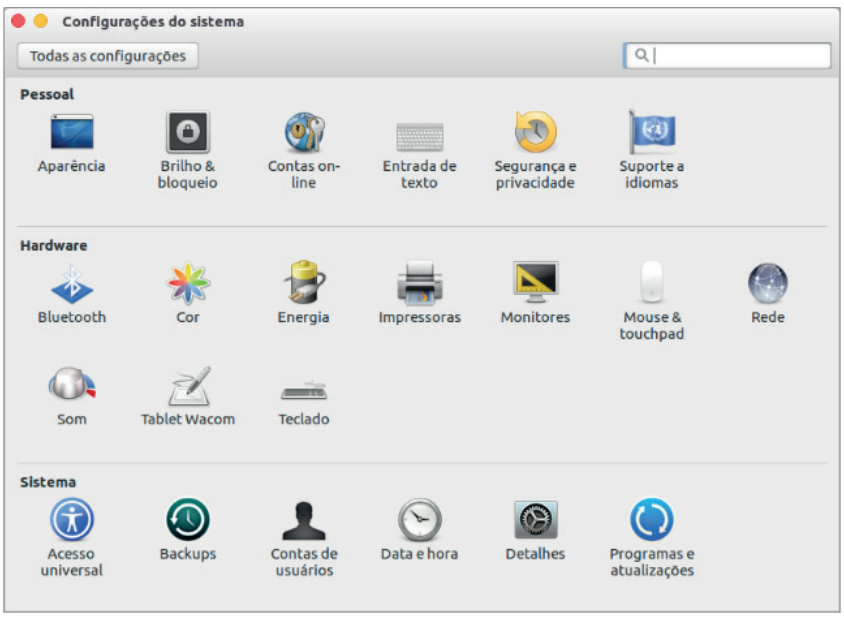
Por exemplo, para montar o pendrive na pasta /media/fulano/pendrive o comando é:

 # mount /dev/sdb1 /media/fulano/pendrive
 # umount /dev/sdb1

Atualmente, praticamente todas as interfaces gráficas do Linux montam e desmontam automaticamente as mídias removíveis no momento da inserção e da retirada dos dispositivos.

Além dos comandos através do terminal ou console do Linux, é possível gerenciar e configurar os dispositivos de E/S no Linux por meio do aplicativo Configurações do Sistema, exibido na Figura 84.

Figura 84 – Configurações do Sistema no Linux



Fonte: Autores, 2017.

6

SISTEMA DE ARQUIVOS

INTRODUÇÃO

O sistema de arquivos é uma padronização e um conjunto de estruturas usados em todos os dispositivos de armazenamento de dados como discos rígidos, SSDs, pendrives e chips de memória flash, para permitir que os dados sejam gravados e lidos de forma adequada e segura, e determinar um nome e uma localização para os mesmos dentro do dispositivo. O sistema de arquivos também controla as permissões de acesso de cada arquivo, bem como gerencia a utilização de cada bloco da área de armazenamento do dispositivo.

O sistema de arquivos é uma das partes mais visíveis de um SO, pois a manipulação de arquivos é uma atividade frequentemente realizada pelos usuários. Somente com um sistema de arquivo é que os dados armazenados podem ser localizados e lidos pelos computadores e smartphones com Windows, Linux, iOS ou Android. Na prática, um sistema de arquivo (file system) é um conjunto de estruturas lógicas feitas via software, que permite ao SO ter acesso e controlar os dados gravados no disco.

Os dados são armazenados como uma sequência de bytes nas unidades de armazenamento sob a forma de arquivos, e a maneira com que os arquivos são armazenados e manipulados dentro de uma unidade de armazenamento varia de acordo com o SO. Cada SO, nativamente, manipula um sistema de arquivos diferente com as suas peculiaridades, como limitações, qualidade, velocidade, gerenciamento de espaço, dentre outras características.

O sistema de arquivos é o responsável por definir como os bytes que compõem um arquivo serão armazenados na unidade de armazenamento e a forma como o SO terá acesso aos dados. Além do sistema de arquivos nativos, os SOs conseguem acessar os sistemas de arquivos de mídias removíveis e alguns nativos de outros sistemas.

O sistema de arquivos é um conjunto de estruturas lógicas e de rotinas que permitem ao SO acessar os dados das unidades de armazenamento, como o disco rígido, o SSD e as mídias removíveis. Existem diferentes sistemas de arquivos, que vão desde sistemas simples como o FAT16, utilizados em cartões de memória, até sistemas como o NTFS, EXT4 e ReiserFS, que implementam recursos mais avançados e seguros.

Nesta unidade você vai conhecer o que é o sistema de arquivos e a estrutura onde ele é armazenado e a diferença entre os diferentes sistemas. Atualmente, o

sistema de arquivos utilizado pelo Windows é o NTFS, o Linux utiliza nativamente o sistema de arquivos Ext4 e os pendrives podem utilizar o sistema FAT32.

6.1
SISTEMA DE ARQUIVOS

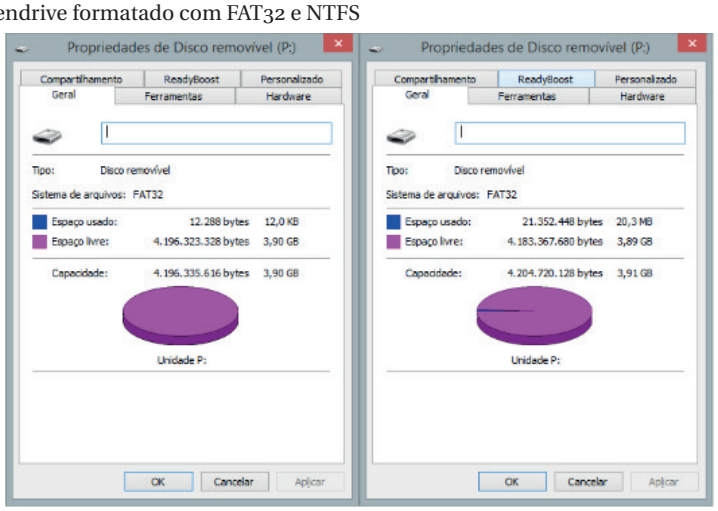
Na maioria das vezes, um disco, um pendrive ou um SSD é dividido em pequenas porções chamadas setores. Em cada setor cabem 512 bytes de informação. O sistema de arquivos utiliza alguns desses setores para criar uma estrutura de controle para gerenciar esses setores. Ele agrupa esses setores em blocos, normalmente de 4 Kbytes, para facilitar o gerenciamento e armazena nessa área de controle as informações dos diretórios (pastas) e arquivos, como seu nome, a data de criação e de modificação, alguns atributos, tais como se é somente leitura, oculto ou um arquivo de sistema, o tamanho e o bloco do disco (formado por alguns setores) onde inicia a área de dados.

Como os dados são constantemente alterados, a sequência de blocos onde ficam o restante dos dados nem sempre é sequencial e, por isso, o sistema de arquivos mantém em sua estrutura de controle, uma lista com a sequência de blocos onde se encontram os dados de cada arquivo. O sistema de arquivos é essa estrutura de controle, capaz de gravar e ler os bytes gravados em um dispositivo de armazenamento.

Quando compramos um disco rígido ou um SSD, a primeira coisa que precisamos fazer é a formatação do mesmo. A formatação nada mais é do que criar um sistema de arquivos no dispositivo. Quando compramos um pendrive, ele já vem formatado, ou seja, já vem com um sistema de arquivos, normalmente o sistema FAT32, que é reconhecido por todos os SOs e também pelos dispositivos de mídia reprodutores de arquivos no formato .mp3.

Se formatarmos um pendrive, podemos escolher o sistema de arquivos que colocaremos nele. Se formatarmos, por exemplo, um pendrive de 4GB, podemos escolher se o mesmo será formatado com FAT32, que é o padrão ou se formataremos com NTFS. A Figura 85 apresenta as propriedades de um mesmo pendrive de 4GB formatado com os sistemas de arquivos FAT32 e com NTFS.

Figura 85 – Pendrive formatado com FAT32 e NTFS



Fonte: Autores, 2017.

Na Figura 85 é possível visualizar que a capacidade, o espaço usado e o espaço livre em ambos os sistemas são diferentes, apesar de ser o mesmo pendrive. É essa utilização de alguns setores para armazenar a estrutura de dados de controle que é o sistema de arquivos. Observe que inicialmente o sistema com FAT32 já utiliza 12Kb enquanto que o NTFS já utiliza 20,3MB, sem ter nenhum programa armazenado no pendrive, e essa quantia necessária para o controle aumenta à medida que criamos novas pastas (diretórios) e armazenamos os arquivos.

Por isso que, quando compramos um pendrive de 4 GB, por exemplo, não conseguimos armazenar nele um arquivo de 4GB, porque o próprio sistema de arquivos ocupa um espaço para fazer o gerenciamento dos demais arquivos, da mesma forma que quando compramos um disco rígido de 500GB temos bem menos de 480GB efetivamente disponíveis.

É importante notar também que 1 KB não representa 1.000 bytes, mas sim 1.024, assim como 1 MB não representa 1.000.000 de bytes, mas sim 1.048.576 e 1 GB representa 1.073.741.824 bytes.

O Windows reconhece os sistemas de arquivos: FAT16, FAT32 e NTFS, além dos sistemas ISO9660 E UDF de CDs, DVDs e BDs. O FAT16 é o mais antigo, usado desde os tempos do MS-DOS, enquanto o NTFS é o mais complexo e atual. Por outro lado, há uma variedade muito grande de sistemas de arquivos diferentes no Linux e em outros sistemas Unix, que incluem o Ext3, Ext4, ReiserFS, Reiser4 XFS, JFS, ZFS e muitos outros.

Os dispositivos de armazenamento secundário são utilizados para armazenar dados de forma permanente, ou seja, de forma confiável e íntegra, mesmo quando o sistema computacional permanece desligado por longos períodos de tempo. Para que o armazenamento possa ser realizado de modo permanente, é necessária uma adequada organização destes dados nos dispositivos destinados ao armazenamento secundário, tais como unidades de disco, pendrives, unidades de fita magnética eDVDs, dentre outros.

Como tudo mais em um sistema computacional, o SO controla as operações sobre os arquivos, organizando seu armazenamento no que chamamos de sistema de arquivos (file system). Um sistema de arquivos normalmente contém (DEITEL, 2005):

• **Métodos de acesso**: forma com que os dados são armazenados nos arquivos;

• **Gerenciamento de arquivos**: conjunto de mecanismos de armazenamento, referência, compartilhamento e segurança;

• **Mecanismos de integridade**: que asseguram que os dados de um arquivo permanecem íntegros.

• Um sistema de arquivos deve permitir, que (DEITEL, 2005):

• Os usuários possam criar, modificar e eliminar arquivos, bem como realizar sua duplicação ou a transferência de dados entre arquivos;

• Seja possível o compartilhamento de arquivos por meio de mecanismos controlados e bem definidos;

• As operações de backup e sua restauração sejam suportadas;

• Seja possível a adoção ou implementação de procedimentos de proteção e segurança;

• Que a interface seja amigável e consistente, admitindo a realização de operações

apenas por meio dos nomes simbólicos dos arquivos, garantindo independência do dispositivo utilizado.

6.1.1 Arquivos

Os arquivos podem ser entendidos como uma sequência de bytes não interpretadas pelo sistema, dependendo-se de aplicações apropriadas para sua correta utilização, sendo um poderoso mecanismo de abstração que permite ao usuário e seus programas utilizarem dados armazenados dentro do sistema computacional. É por meio da manipulação dos arquivos que são realizadas as operações de escrita e leitura de dados de forma transparente, evitando que sejam conhecidos detalhes do funcionamento, com que estas operações tratam e armazenam a informação (TANNENBAUM, 2016).

A organização dos arquivos consiste no modo como os dados estão internamente armazenados, podendo ter uma estrutura de acordo com a função do tipo de informação contida no arquivo. A forma mais básica de organização é por meio de uma sequência não estruturadas de bytes, cuja organização e interpretação é de inteira responsabilidade da aplicação.

Muitos sistemas operacionais utilizam a extensão do nome do arquivo como uma referência para o tipo de seu conteúdo. Particularmente, no SO Windows, podem ser associadas aplicações a determinadas extensões, auxiliando o usuário.

Genericamente arquivos comuns, utilizados pelos usuários para o armazenamento de informações, podem ser considerados como do tipo texto, quando seu conteúdo pode ser visualizado diretamente ou do tipo binário, quando seu conteúdo precisa ser interpretado por uma aplicação para poder ser visualizado. Os arquivos executáveis também são arquivos binários, cuja execução é realizada diretamente pelo SO.

6.1.2 Diretórios ou Pastas

Além dos arquivos comuns (ou documentos), os sistemas de arquivos geralmente mantêm arquivos especiais denominados **diretórios** (ou pastas) que contêm partes da estrutura do sistema de arquivos. Em sistemas Unix ainda existem arquivos especiais utilizados para modelar certos dispositivos periféricos, que permitem o acesso a terminais, impressoras e rede no primeiro caso, discos, fitas e outros dispositivos de armazenamento secundário no segundo.

Um diretório nada mais é que um arquivo com uma estrutura de controle dos demais arquivos, mantido pelo sistema de arquivos, que contêm uma lista de arquivos e possivelmente outros diretórios. Desta forma é possível criar-se estruturas hierárquicas de arquivos, onde os diretórios existentes dentro de outros são denominados **subdiretórios (ou subpastas)**.

O que motivou a criação dos diretórios é o fato de que, torna-se difícil a organização de um número grande de arquivos armazenados, em um determinado dispositivo, sem qualquer forma de divisão. A criação de diretórios, permite que os arquivos sejam divididos e organizados, de acordo com um critério escolhido, possibilitando seu acesso e localização de forma mais simples.

6.1.3 Operações com Arquivos e Diretórios

O SO, por meio de seu sistema de arquivos, deve prover um conjunto de operações para manipulação dos arquivos e de seu conteúdo e também dos diretórios. Do ponto de vista dos arquivos, visto como unidades, devem ser oferecidas as seguintes operações (DEITEL, 2005), (TANNENBAUM, 2016):

a) **Abertura (open)**: prepara arquivo para uso;

b) **Fechamento (close)**: encerra o uso do arquivo evitando sua alteração;

c) **Criação (create)**: cria um novo arquivo;

d) **Eliminação (erase, delete ou destroy)**: remove um arquivo;

e) **Renomeação (rename)**: troca o nome de um arquivo;

f) **Cópia (copy)**: copia o conteúdo de um arquivo para outro;

g) **Exibição (type ou list)**: exibe o conteúdo de um arquivo;

h) **Catalogação (cat ou dir)**: lista os arquivos existentes em um determinado diretório ou unidade;

i) **Modificação de atributos (get ou set)**: obtêm ou modifica os atributos de um arquivo;

Do ponto de vista do conteúdo e manipulação dos dados armazenados nos arquivos, devem também existir operações para:

a) **Leitura (read)**: possibilita a leitura de dados contidos nos arquivos;

b) **Escrita (write)**: efetua a gravação de dados em um arquivo;

c) **Pesquisa (seek ou find)**: para determinar uma posição para escrita ou leitura em um arquivo;

d) **Anexação (append)**: para adição de novos dados em um arquivo existente;

e) **Inserção (insert)**: para inclusão de um dado em um arquivo;

f) **Atualização (update)**: para modificação de um dado existente em um arquivo;

g) **Eliminação (delete)**: para remoção de um dado existente em um arquivo.

Como muitos sistemas de arquivos suportam diretórios, operações específicas devem ser providas para sua utilização (TANNENBAUM, 2016):

a) **Criação (create)**: efetua a criação e preparo de um novo diretório;

b) **Remoção (delete ou remove)**: elimina um diretório e opcionalmente seu conteúdo;

c) **Abertura (open)**: operação que permite a leitura de um diretório;

d) **Fechamento (close)**: operação que encerra o uso de um dado diretório;

e) **Leitura (read)**: permite a leitura do conteúdo de um diretório, ou seja, sua catalogação ou listagem;

f) **Pesquisa (search)**: para possibilitar a localização de determinados arquivo em um diretório;

g) **Renomeação (rename)**: troca o nome de um diretório;

h) **Navegação (traverse)**: permite a navegação entre diretórios do sistema.

Ainda podem ser possíveis outras operações sobre arquivos e diretórios, como inclusão ou remoção de ligações (links e atalhos), obtenção de informações sobre arquivos ou diretórios, associação com aplicativos, dentre outras. Todas as operações suportadas pelo SO para manipulação de arquivos e diretórios estão disponíveis para utilização em programas por meio das chamadas do sistema (system calls).

6.1.4 Partições

A forma de armazenamento consiste em dispor os dados em setores ou blocos consecutivos nas unidades de disco. Porém, como os arquivos são constantemente modificados e eliminados, a alocação não consegue manter um arquivo de forma sequencial. Este fato provoca o que se chama de fragmentação, quando começam a surgir setores livres entre os setores ocupados.

Se o arquivo aumenta de tamanho, um novo setor, provavelmente bem longe do último ocupado, será alocado para dar continuidade nos dados do arquivo, deixando o arquivo fragmentado. Se o tamanho do arquivo for reduzido, um espaço no sistema de arquivos ficará ocioso e gerará uma fragmentação do sistema de arquivos. Normalmente os SOs oferecem algum tipo de utilitário que realiza a desfragmentação do sistema de arquivos, movendo os arquivos de modos que todos os blocos de cada arquivo fiquem de modo contíguo e que não haja espaço vazio entre eles.

Para instalar um sistema de arquivos em um disco ou unidade de armazenamento é necessário realizar o particionamento do disco, criando as partições, onde cada partição é independente das outras, ou seja, cada partição pode ter o seu próprio sistema de arquivos. Desta forma, podemos, por exemplo, instalar o Linux em uma

partição e o Windows em outra partição, em um mesmo disco rígido.

A tabela, onde são armazenadas as informações sobre as partições, fica no primeiro setor do disco e chama-se MBR (Master Boot Record). O MBR é um padrão antigo que define a estrutura das partições, mantendo a informação sobre como as partições lógicas estão organizadas no disco. Usando esta estrutura, há algumas limitações: um disco pode dividir-se em um máximo de 4 partições primárias ou em até 3 partições primárias e 1 partição estendida. Uma partição estendida pode dividir-se em múltiplas partições lógicas. Cada partição primária e lógica tem a sua própria unidade.

Como o MBR usa 32 bits para guardar informações das partições, cada partição apenas pode ter um tamanho máximo de 2 TB e toda a informação das partições é guardada apenas em um único local, o MBR.

Os novos sistemas computacionais permitem o uso de GPT (Tabela de Partição Guid), que é uma nova forma de realizar o particionamento de discos rígidos. O GPT traz muitos benefícios comparativamente ao tradicional MBR (Master boot record), entre os quais se destacam:

- Suporte para partições bem maiores do que 2 TB;
- Suporte para até 128 partições primárias;
- Melhor estrutura/organização ao nível das partições;
- Possui mecanismos para detecção de dados e partições corrompidas;
- Aumenta a probabilidade de recuperação de dados, na existência de setores do disco danificados, uma vez que tem um cabeçalho secundário (Secondary GPT Header) que funciona como backup da tabela de partições;
- Mantém a compatibilidade com o MBR, para sistemas legados.
- Depois de realizado o particionamento do disco é necessário realizar a formatação da partição, escolhendo um sistema de arquivos para a mesma, de acordo com o SO que vai ser usado.

6.1.5 Tipos de Sistemas de Arquivos

Os principais tipos de Sistemas de arquivos, de acordo com os SOs mais usuais e formato nativo de arquivos, são:

- Apple Macintosh:

a) **HFS - Hierarchical File System**: é um sistema de arquivos desenvolvido pela Apple Computer para uso em computadores rodando o Mac OS, também pode ser referenciado como HFS Standard e Mac OS Standard, sendo que o seu sucessor HFS Plus também é chamado HFS Extended ou Mac OS Extended;

- Unix (FreeBSD, OpenBSD, Linux, Solaris, etc.):

b) **UFS**: o Unix File System (UFS), é um sistema de arquivos usado por muitos sistemas operacionais Unix e assemelhados. Também é conhecido como Berkeley Fast File System, BSD Fast File System ou simplesmente FFS. É o sistema original do tipo i-nodes;

c) **Ext2**: o Ext2 (second extended file system) é um sistema de arquivos para dispositivos de blocos (disco rígido, disquete, pendrive). O Ext2 foi projetado e implementado para corrigir as deficiências do Ext e prover um sistema que respeitasse a semântica do Unix;

d) **Ext3**: o Ext3 (third extended file system) é um sistema de arquivos desenvolvido para o Linux, que acrescenta o recurso de journaling ao Ext2. O journaling é o controle de transações que permite desfazer uma operação que falhou e voltar ao último estado seguro do sistema;

e) **Ext4**: o Ext4 é um sistema de arquivos do Linux desenvolvido para ser o sucessor do Ext3 a partir de 2006. As novas funcionalidades propostas são: alocação tardia (delay ed allocation); marcas temporais com maior resolução (nano segundos) verificação de integridade do journal (jornal check sums) e suporte para tamanhos maiores de volumes e arquivos;

f) **SWAP**: sistema de arquivo para o arquivo de paginação da memória virtual do Linux;

g) **Reiser** - Criado por Hans Reiser e inicialmente mantido pela empresa Namesys, o ReiserFS foi o primeiro sistema de arquivos com suporte a "journaling" incluído no kernel do Linux;

h) **HPFS**: o HPFS é o sistema de arquivos utilizado pelo OS/2 da IBM, com recursos que se aproximam muito dos permitidos pelo NTFS como nome de arquivos com até 254 caracteres incluindo espaços, partições de até 512 GB e unidades de alocação de 512 bytes. Embora muito eficiente, este sistema de arquivos caiu em desuso juntamente com o OS/2, sendo suportado atualmente somente pelo Linux;

i) **JFS**: o Journaling File System é um sistema de arquivos desenvolvido pela IBM, disponível em licença open-source, com o intuito de ser utilizado nos "UNIXes" que a IBM vendia. Além de possuir journal, ele permite que as partições do sistema sejam redimensionadas sem que seja necessário desligar o computador. O sistema de arquivos JFS também usa a estrutura i-node para armazenar a localização dos blocos de cada arquivo nas estruturas físicas do disco, a versão JFS2 armazena esses i-nodes em uma árvore binária para acelerar o acesso a essas informações;

j) **XFS**: o XFS é um sistema de arquivos inicialmente desenvolvido pela Silicon Graphics Inc. para o seu sistema operacional IRIX. Posteriormente teve seu código fonte liberado e foi adaptado para funcionar no Linux. Usa alocação dinâmica de i-nodes e journaling;

- MS-DOS/Microsoft Windows

k) **FAT16 ou FAT**: sistema básico usado no MS-DOS da década de 90 e nos sistemas Windows 1.X ou superior. Suporta somente nomes com 8 caracteres e extensão de 3 caracteres;

l) **FAT32**: usado nos sistemas MS-DOS 7.1 e Windows 95 ou superior. Tem suporte a nomes longos e é largamente usado em pendrives e cartões de memória, devido à simplicidade da sua estrutura. Apresenta uma limitação de que o maior arquivo não pode exceder 4GB;

m) **NTFS**: usado no Windows NT ou superior, apresenta um algo grau de confiabilidade com controle de permissões de usuários, suporte à criptografia e compactação;

• CDs/DVDs/BDs (BluRay Disc)

n) **ISO9660**: um padrão internacional que especifica como os dados são fisicamente gravados no CD (setor por setor) em uma árvore de diretório (até oito níveis de diretórios). CD-ROMs criados de acordo com as especificações ISO 9660 Nível 1 podem ser lidos em vários SOs, como Macintosh, DOS, Windows 3.x, Windows 98/ NT/2000/XP e Linux;

o) **Joliet**: Um padrão desenvolvido pela Microsoft para o Windows 95 que permite que CDs sejam gravados usando nomes de arquivo longos. O padrão Joliet permite usar nomes de arquivos com até 64 caracteres de comprimento, incluindo espaços, e também permite usar caracteres Unicode para um melhor suporte internacional. O padrão Joliet grava o nome de arquivo padrão DOS associado (8 + 3 caracteres) para cada arquivo de modo que o CD possa ser lido em sistemas DOS ou em versões anteriores do Windows;

p) **RockRidge** é uma extensão que permite nomes longos para o ISO9660 file system, usado com o Linux;

q) **El Torito**: pode ser usado para produzir CD-ROMsbootáveis. Para esta característica funcionar, a BIOS precisa suportar o boot;

r) **UDF**: Universal Disk Format é um padrão desenvolvido pela Optical Storage Technology Association (OSTA) e é um sistema de arquivos usado com tecnologias de disco óptico, tal como DVD e BluRay Disc. Este padrão é particularmente útil na gravação de arquivos muito grandes (maiores que 1 GB). O sistema de arquivos UDF permite uma maior flexibilidade em comparação com os sistemas Joliet ou ISO 9660.

Diferentes SOs podem usar diferentes sistemas de arquivos. Atualmente, o NTFS (*New Technology File System*) é o sistema de arquivos padrão do Windows, enquanto o Ext4 é o mais usado no Linux.

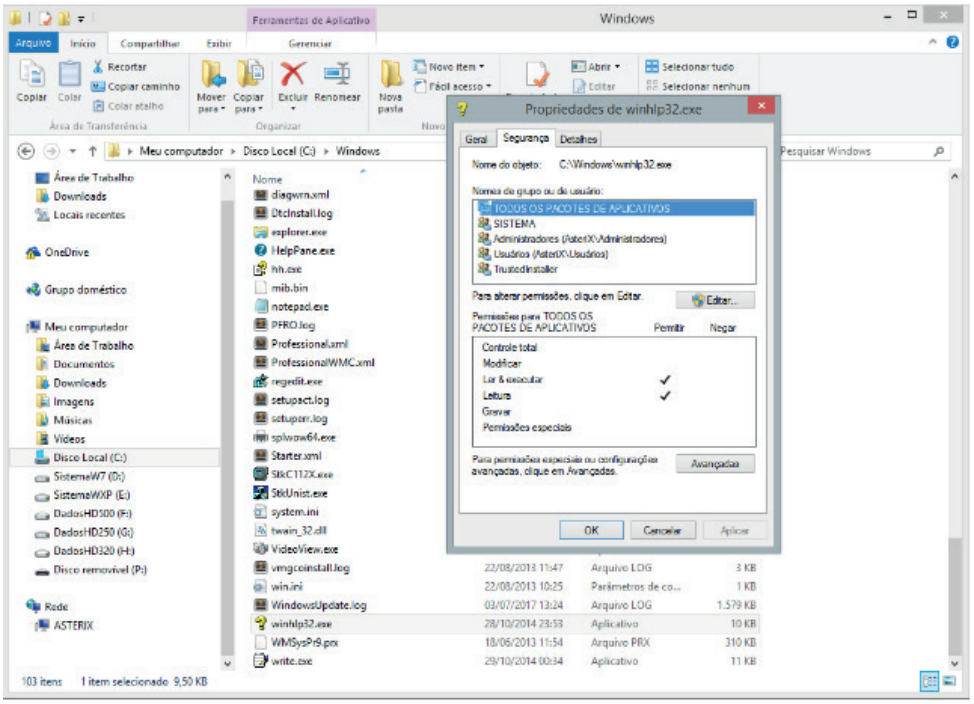
6.1.6 Sistema de arquivos do Windows

O NTFS - New Technology File System é um sistema de arquivos que surgiu juntamente com o lançamento do Windows NT. Por apresentar excelente desempenho e confiabilidade, foi adotado nos SOsWindows.

O NTFS possui várias características, em caso de falhas, por exemplo, quando o computador tem um desligamento repentino, ele tem a capacidade de reverter os dados para a condição anterior ao problema. O NTFS também possui a característica de suportar uma replicação de dados, controle de permissões de acesso e possui muita eficiência no trabalho com grandes arquivos e também unidades de discos bastante cheias. Suporta criptografia, indexação e compactação.

Cada unidade ou volume de disco é identificada com uma letra, começando por padrão com C. Armazena as estruturas de controle do sistema de arquivos em duas áreas distintas do disco para garantir a integridade, no caso de falha em uma área do disco. A Figura 86 apresenta o sistema de arquivos do Windows, com o detalhe das propriedades de um arquivo e suas permissões.

Figura 86 – Sistema de Arquivos do Windows



Fonte: Autores, 2017.

6.1.7 Sistema de arquivos do Linux

O Linux é um SO que dá suporte a diversos sistemas de arquivos. Atualmente, o sistema de arquivos mais utilizado é o ext4 desenvolvido para ser o sucessor do ext3 a partir de 2006.

O Ext4 apresenta inúmeros novos aprimoramentos quanto a desempenho, escalabilidade e confiabilidade e suporta sistemas de arquivos de até 1 exabyte (um

bilhão de Gigabytes). Apresenta uma performance considerada muito boa, principalmente para o seu uso em máquinas de grande porte, superando a capacidade de armazenamento do EXT3.

O novo sistema de journaling do EXT4 melhora a quantidade de arquivos e processos suportados pelo Linux. Graças à alocação tardia, existe uma considerável redução de fragmentação, por ele melhorar nas decisões do sistema sobre o arquivo levando em consideração o tamanho do mesmo, porém isso apresenta riscos de ocorrência de perda de dados em caso de desligamentos abruptos.

No Linux, um diretório (corresponde ao conceito de pasta do Windows) pode ter outros diretórios (subdiretórios) ou arquivos. O diretório acima, onde um subdiretório foi criado é chamado de **diretório pai**.

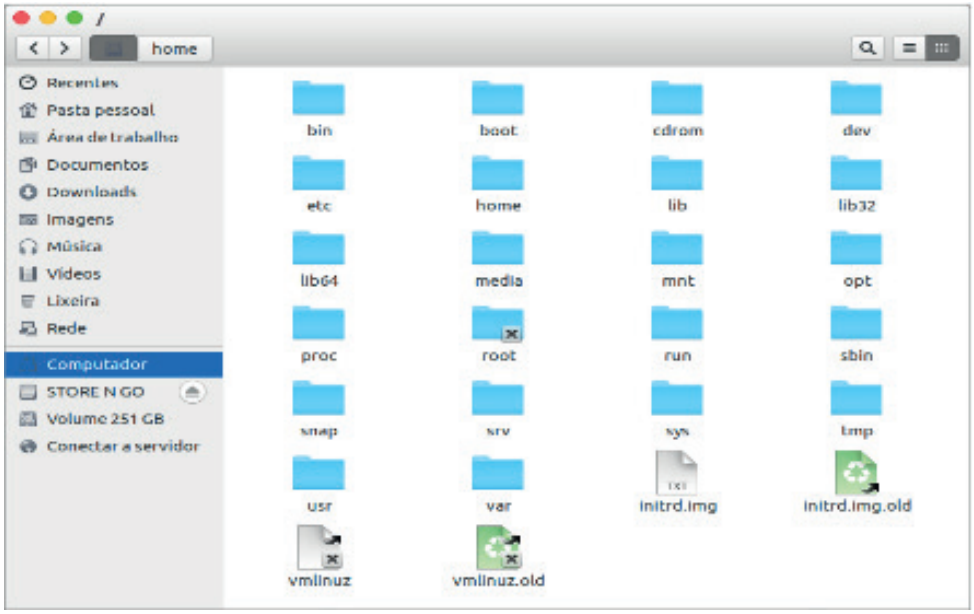
O Linux implementa um sistema de arquivos baseado nos *i-nodes* do Unix, onde há uma tabela na unidade de armazenamento onde são controlados os nomes, permissões e atributos de cada arquivo ou diretório, bem como os endereços dos blocos iniciais de cada arquivo ou diretório. Os demais endereços das partes de cada arquivo ou diretório são gravados espalhados pelo disco, próximos aos blocos dos respectivos arquivos e diretórios.

O diretório raiz do Linux (ou diretório /) é o diretório com maior hierarquia entre todos os diretórios do sistema. Isto significa que todos os diretórios do Linux ficam abaixo deste diretório. A seguir são apresentados alguns exemplos de diretórios que ficam abaixo do diretório raiz:

- **bin** - diretório com os comandos disponíveis para os usuários (não privilegiados);
- **boot** - diretório com os arquivos do boot de inicialização;
- **dev** - diretório com as definições dos dispositivos de entrada/saída;
- **etc** - diretório com os arquivos de configuração do sistema;
- **home** - diretório que armazena os diretórios dos usuários do sistema;
- **lib** - diretório com as bibliotecas e módulos (carregáveis) do sistema;
- **mnt** - diretório usado para montagem de partições;
- **proc** - diretório com informações sobre os processos do sistema;
- **root** - diretório home do root (administrador do sistema);
- **sbin** - diretório com os aplicativos usados na administração do sistema;
- **tmp** - diretório com arquivos temporários;
- **usr** - diretório com aplicativos e arquivos utilizados pelos usuários como, por exemplo, o sistema de janelas X, jogos, bibliotecas compartilhadas, programas de usuários e de administração, etc;
- **var** - diretório com arquivos de dados variáveis (spool, logs, etc).

Convém também observar que é possível colocar os subdiretórios do diretório raiz em partições separadas. O objetivo é facilitar a manutenção do sistema e aumentar a segurança dos dados. Portanto, a distribuição do diretório raiz em várias partições é uma escolha pessoal do administrador do sistema. Normalmente, é sugerido que os seguintes diretórios possuam uma partição própria: **/home, /opt, /tmp, /usr e /usr/local**. A Figura 87 apresenta o sistema de arquivos do Linux.

Figura 87 – Sistema de Arquivos do Linux



Fonte: Autores, 2017.

CONSIDERAÇÕES FINAIS

Este material apresentou os principais conceitos sobre os sistemas operacionais. Através deste material você obteve uma visão geral de como funciona um computador e de como sistema operacional facilita o seu uso, gerenciando os recursos de hardware e fornecendo uma interface amigável que permite o seu uso de forma intuitiva. O material também apresentou a evolução dos computadores e dos diferentes tipos de sistemas operacionais, desde os primeiros computadores criados até os modernos smartphones atuais. Também estudamos, de forma mais detalhada, como funciona um sistema operacional, como o computador é inicializado e as principais funções de um sistema operacional. Estudamos o propósito de função de gerenciamento que o sistema operacional desempenha, envolvendo a gerência de processos, fornecendo detalhes sobre seu modo de operação e descrevendo como podemos acessar as funções de gerenciamento relacionadas aos programas que estão em execução. Analisamos também a forma como o sistema operacional gerencia a memória e protege os programas, para que um programa não invada a área de memória de outro programa ou usuário. Analisamos a forma como os dispositivos de entrada e saída são gerenciados e também os principais sistemas de arquivos, nos principais sistemas operacionais utilizados atualmente.

REFERÊNCIAS

DEITEL, H. M.; DEITEL, P. J.; CHOFFNES, D. R. **Sistemas Operacionais.** São Paulo: Pearson Prentice Hall, 2005

JAIN, R. **The Art of Computer System Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling.** New York – NY: Wiley Computer Publishing, John Wiley & Sons, 1991.

JANDL JR., P. **Notas sobre Sistemas Operacionais**. Apostila, In: FACULDADE de Viçosa. Viçosa: FVMG, 2004. Disponível em <http://correio.fdvmg.edu.br/downloads/DET424/SistemasOperacionais-peter-jandl-jr.pdf>. Acesso em 20 mai. 2017.

MACHADO, F, B.; MAIA, P, L. **Arquitetura de Sistemas Operacionais**. Rio de Janeiro: LTC, 2014.

RIBEIRO, U. **Certificação Linux**. Rio de Janeiro: Axcel Books do Brasil, 2004.

STALLINGS, W. **Arquitetura e Organização de Computadores: projeto para o desempenho.** São Paulo: Prentice Hall, 2002.

STUART, B. L. **Princípios de sistemas operacionais: projetos e aplicações.** São Paulo: Cengage Learning, 2011.

TANENBAUM, A. S. **Sistemas Operacionais Modernos**. Tradução de Jorge Ritter, revisão técnica de Raphael Y. de Camargo. São Paulo: Pearson Education do Brasil, 2016.

ATIVIDADES

Unidade 1 - Fundamentos

1) Como eram os computadores e os sistema operacionais nas décadas de 1940 e 1950?

2) Por que foi necessário criar um sistema operacional para os computadores?

3) O que difere um computador de uma calculadora?

4) O que é a multiprogramação?

5) Quais são os principais sistemas operacionais atualmente?

6) Como é a interface do sistema operacional?

7) O que difere o Windows do Linux?

Unidade 2 - Estrutura do Sistema Operacional

1) Quais as funções de um sistema operacional?

2) Quais as funções do núcleo de um sistema operacional?

3) Como é o processo de boot e de carregamento do sistema operacional para a memória?

4) Quais são os serviços que um sistema operacional disponibiliza?

5) O que é o núcleo ou kernel de um sistema operacional?

6) O que é uma plataforma ou ambiente operacional?

7) De acordo com os tipos de aplicativos e usuários suportados, quais são os principais tipos de sistemas operacionais?

Unidade 3 - Gerência de Processos

1) Defina o conceito de processo.

2) Por que o conceito de processo é tão importante no projeto de sistemas multiprogramáveis?

3) É possível que um programa execute no contexto de um processo e não execute no contexto de um outro? Por quê?

4) Quais partes compõem um processo?

5) O que é o contexto de hardware de um processo e como é a implementação da troca de contexto?

6) Qual é a função do contexto de software? Exemplifique cada grupo de informação.

7) O que é espaço de endereçamento de um processo?

8) Como o sistema operacional implementa o conceito de processo? Qual a estrutura de dados indicada para organizar os diversos processos na memória principal?

9) Defina os cinco estados possíveis de um processo.

10) Dê um exemplo que apresente todas as mudanças de estado de um processo, juntamente com o evento associado a cada mudança.

11) Diferencie processos multithreads, subprocessos e processos independentes.

12) Explique a diferença entre processos foreground e background.

13) Dê exemplos de aplicações CPU-Bound e I/O-Bound.

14) Justifique com um exemplo a frase "o sinal está para o processo, assim como as interrupções e exceções estão para o sistema operacional".

15) Liste o nome de 5 processos que estão executando no Windows.

Unidade 4 - Gerência de Memória

1) Qual é a função da gerência de memória?

2) O que é a memória virtual?

3) O que é o swap e a técnica de swapping?

4) O que são as pagefaults e por que elas ocorrem?

5) O que é a hiperpaginação e como ela pode ser evitada?

6) Acesse um computador e descreva: quanto de memória tem instalada, quanto está em uso da memória real, quanto é o tamanho da memória virtual, quanto está em uso da memória virtual.

Unidade 5 - Gerência de Entrada e Saída

1) O que envolve a gerência de Entrada e Saída?

2) O que são as interrupções?

3) O que acontece com o processador quando ocorre uma interrupção?

4) O que são os módulos de I/O?

5) O que são os controladores de dispositivos?

6) O que são os drivers do dispositivo?

7) O que é a técnica de RAID de discos?

Unidade 6 - Sistema de Arquivos

1) O que é um sistema de arquivos?

2) O que é um arquivo?

3) O que é um diretório?

4) Quais as principais operações sobre arquivos e diretórios oferecidas por um sistema operacional?

6) O que é a formatação de uma unidade de armazenamento?

7) O que é o sistema FAT32?

8) O que é o sistema Ext4?

9) O que é o sistema NTFS?

10) Para formatar um pendrive e gravar algumas músicas no formato .mp3, para ser ouvido no Windows, Linux, Mac e também no rádio do carro, qual o sistema de arquivos deve ser usado?

11) Para formatar um pendrive de 16GB, para gravar um único arquivo .iso de instalação de um jogo para Windows, com tamanho de 5,2GB, qual o melhor sistema de arquivos?

APRESENTAÇÃO DOS PROFESSORES AUTORES

Este material foi elaborado por professores altamente qualificados na área da computação, integrantes do Departamento de Tecnologia da Informação da Universidade Federal de Santa Maria – Campus Frederico Westphalen/RS (UFSM/FW):

Prof. Guilherme Bernardino da Cunha: Bacharel em Ciência da Computação pelo Unitri (Centro Universitário do Triângulo), Especialista em Produção de Material Didático para EaD pela Universidade Federal do Amazonas (UFAM), Mestre e Doutor em Engenharia Elétrica pela Universidade Federal de Uberlândia (UFU). Suas principais áreas de interesse são: Processamento Gráfico, Informática aplicada à Saúde, Inteligência Artificial, Software Básico e Linguagens de Programação. Atualmente é Professor Adjunto do Departamento de Tecnologia da Informação da UFSM/FW.

Ricardo Tombesi Macedo é Professor Adjunto da UFSM/FW, tendo obtido o título de Doutor em Ciência da Computação pela Universidade Federal do Paraná (UFPR) com período sanduíche na Université de La Rochelle na França, o título de Mestrado em Engenharia da Produção pela UFSM e de Bacharel em Ciência da Computação pela Universidade de Cruz Alta (UNICRUZ).

Evandro Preuss possui graduação em Ciência da Computação e em Ciências Contábeis pela Universidade Regional Integrada do Alto Uruguai e das Missões (URI), Mestrado em Informática pela Pontifícia Universidade Católica do Rio Grande do Sul (PUC-RS) e atualmente é Professor Assistente da UFSM/FW.