

Fundamentos de Sistemas Operacionais

Fundamentos de sistemas operacionais

Juliana Schiavetto Dauricio

© 2015 por Editora e Distribuidora Educacional S.A

Todos os direitos reservados. Nenhuma parte desta publicação poderá ser reproduzida ou transmitida de qualquer modo ou por qualquer outro meio, eletrônico ou mecânico, incluindo fotocópia, gravação ou qualquer outro tipo de sistema de armazenamento e transmissão de informação, sem prévia autorização, por escrito, da Editora e Distribuidora Educacional S.A.

Presidente: Rodrigo Galindo

Vice-Presidente Acadêmico de Graduação: Rui Fava

Gerente Sênior de Editoração e Disponibilização de Material Didático:

Emanuel Santana

Gerente de Revisão: Cristiane Lisandra Danna

Coordenação de Produção: André Augusto de Andrade Ramos

Coordenação de Disponibilização: Daniel Roggeri Rosa

Editoração e Diagramação: eGTB Editora

Dados Internacionais de Catalogação na Publicação (CIP)

Dauricio, Juliana Schiavetto
D243f Fundamentos de sistemas operacionais / Juliana
Schiavetto Dauricio. – Londrina : Editora e Distribuidora
Educacional S.A., 2015.
256 p.

ISBN 978-85-8482-233-1

1. Sistemas operacionais (Computadores). I. Título.

CDD 005.43

2015

Editora e Distribuidora Educacional S.A

Avenida Paris, 675 – Parque Residencial João Piza

CEP: 86041-100 – Londrina – PR

e-mail: editora.educacional@kroton.com.br

Homepage: <http://www.kroton.com.br/>

Sumário

Unidade 1 Introdução aos sistemas operacionais	7
Seção 1.1 - Definição, conceitos e histórico dos sistemas operacionais	9
Seção 1.2 - Tipos de sistemas operacionais: monoprogramáveis, multiprogramáveis e multiprocessamento	27
Seção 1.3 - Características dos sistemas operacionais multiprogramáveis	41
Seção 1.4 - Exemplos de sistemas operacionais: Unix e Windows	55
Unidade 2 Processos e <i>threads</i>	71
Seção 2.1 - Introdução a processos: modelo, criação, término, hierarquia, estados, implementação e <i>threads</i>	73
Seção 2.2 - Comunicação entre processos e problemas clássicos de comunicação entre processos	87
Seção 2.3 - Introdução ao escalonamento: conceitos, tipos e escalonamento de <i>threads</i>	101
Seção 2.4 - Algoritmos de escalonamento: características, políticas, tipos e exemplos	115
Unidade 3 Sistema de arquivos	131
Seção 3.1 - Arquivos: atribuição de nomes, estrutura, tipos, acesso, atributos e operações	133
Seção 3.2 - Diretórios: diretórios simples, sistemas de diretório hierárquico, nomes de caminho e operações	149
Seção 3.3 - Introdução à implementação do sistema de arquivos. Virtualização do sistema de arquivos e registro	163
Seção 3.4 - Introdução à segurança e mecanismos de proteção	177
Unidade 4 Gerenciamento de dispositivos	193
Seção 4.1 - Gerenciamento de memória: conceitos, tipos, características e virtualização	195

Seção 4.2 - <i>Swapping</i> : conceitos, tipos e suas características _____	209
Seção 4.3 - Memória virtual: conceitos, paginação, segmentação e virtualização _____	221
Seção 4.4 - Gerenciamento de dispositivos de entrada e saída: conceitos, rotinas, tipos e suas características _____	235

Palavras do autor

Caro(a) aluno(a), bem-vindo à disciplina de Fundamentos de Sistemas Operacionais. Nesta unidade curricular, você será apresentado aos principais tópicos de sistemas operacionais, tais como: o contexto histórico e as responsabilidades desse *software* para o bom desempenho da máquina, inclusive no que tange ao gerenciamento de aplicativos, programas e arquivos. Com isso, ele também se torna responsável, por alocar e distribuir os recursos de memória e processamento de forma a otimizar e melhorar o desempenho da máquina.

A fim de evidenciar a importância de estudos desta unidade curricular, fica ainda a sugestão para que você visite quantas vezes achar necessário o material didático. O seu material é composto pelo livro didático, que apresenta os principais tópicos que deverão ser estudados para desenvolver os conhecimentos necessários para a sua aplicação; além desse, você também pode contar com a orientação das atividades apresentadas nas webaulas e, ainda, os momentos de orientação, mediação, explicação e interação que ocorrem no decorrer das aulas. Participe atentamente das atividades! A estrutura de seu livro didático contempla quadro unidades de ensino. São elas:

- **Introdução aos sistemas operacionais:** apresenta todo o contexto histórico desses *softwares*.
- **Processos e threads:** compreende os modelos de processos e o modo de compartilhamento de recursos.
- **Sistemas de arquivos:** contempla alguns fatores, como alocação, virtualização, segurança e mecanismos de proteção.
- **Gerenciamento de dispositivos:** expõe as formas de se entender as rotinas e as características desses processos.

Prezado estudante, mantenha uma rotina de estudos que lhe possibilite se dedicar aos processos de leitura, participação e realização das atividades propostas. É de extrema importância para que você obtenha sucesso tanto em construção e desenvolvimento de aprendizagem quanto em sua aplicação. Desde já, desejo a você bons estudos!

INTRODUÇÃO AOS SISTEMAS OPERACIONAIS

Convite ao estudo

Para você obter um bom nível de desempenho na disciplina, de forma a permitir, contribuir e colaborar para com o seu processo de ensino-aprendizagem, será preciso desenvolver algumas competências. Essas são classificadas em geral e técnicas. Conheça-as a seguir:

- **Competência geral:** o aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional, bem como ter conhecimento sobre como se dá o compartilhamento de recursos e a sua gerência.

- **Competência de fundamentos de área:** conhecer a evolução dos sistemas operacionais e suas respectivas especificidades; conhecer e saber identificar os principais processos e como ocorre o compartilhamento de recursos; conhecer como se dá a gerência de processos e de armazenamento de arquivos; conhecer e saber gerenciar os dispositivos de entrada e saída.

Além dessas, conheça também os objetivos específicos de aprendizagem aos quais o contato com esta unidade curricular pretende atender. São eles: saber fazer a manipulação das informações do sistema operacional, ter conhecimento sobre as principais funcionalidades e como gerenciá-las, além de saber analisar a utilização de recursos e promover a sua otimização.

Agora, você pode se perguntar: Em que momento de minha carreira precisarei aplicar tais conhecimentos? Quais são os principais problemas pertinentes ao uso dos sistemas operacionais? Como esses precisam evoluir para acompanhar as necessidades de gerenciamento, processamento e armazenamento de uma máquina, seja ela um computador pessoal ou mesmo

um aparelho *mobile*?. Todas essas são situações que podem permear os seus pensamentos ou mesmo o seu dia a dia profissional. Então, o conhecimento sobre essa importante ferramenta do ambiente computacional poderá facilitar algumas atividades, como a escolha de programas, aplicativos e dispositivos. Saber de sua compatibilidade e portabilidade também é bastante útil para a vida profissional, independente de sua área de atuação; esses são conhecimentos valiosos para a sua rotina. Já tentou baixar um aplicativo que foi desenvolvido para IOS (Apple-IPhone) em um sistema operacional Android (Google – demais marcas de aparelhos celulares)? Isso não será possível, mas por quê? Pois é, atualmente, há a necessidade de se conhecer alguns princípios e premissas necessárias até mesmo para o uso correto e mais potencializado do seu aparelho celular, por exemplo. Se você conhecer o sistema operacional desse equipamento, alguns problemas poderão ser evitados.

Então, pensando nesse contexto, o seu desafio, a partir desse ponto, já passa a ser encontrar a melhor solução no que tange à escolha de um sistema operacional, para uma empresa de pequeno porte, no início de suas atividades, inclusive de automatização de algumas rotinas. Sua maior atenção deverá ser encontrar, dentre os sistemas operacionais disponíveis no mercado, um que seja adequado à sua realidade profissional e que de fato esteja dentro de seu orçamento. Então, desde já, bons estudos e práticas!

Seção 1.1

Definição, conceitos e histórico dos sistemas operacionais

Diálogo aberto

Olá, aluno! Vamos, agora, iniciar os estudos em sistemas operacionais. Você, primeiramente, será apresentado a um breve levantamento histórico, o que o levará a compreender como surgiram e de que forma têm evoluído até os dias atuais. Nesse sentido, cabe resgatar de que forma o mercado de trabalho exige esse conhecimento.

A proposta é apresentar, avaliar e escolher, um sistema operacional que atenda às necessidades de uma empresa de consultoria acadêmica, de pequeno porte, cujo modelo de negócios é baseado em orientação escolar. Ela apresenta grande potencial no mercado e está no início de suas atividades. Em parceria com as escolas da cidade, precisará imediatamente implantar um sistema operacional que permita a instalação dos aplicativos para envio e recebimento de materiais e informações sobre a evolução dos alunos nas disciplinas indicadas.

A empresa, ainda, mantém-se conectada à internet todo o tempo, além de ter de compartilhar recursos, tais como o acesso à rede de computadores interna cabeada e sem fio; duas impressoras e uma máquina de fazer cópias de documentos.

Esses recursos e dispositivos precisam ser compartilhados com os professores e demais colaboradores da empresa. Além disso, há um *software* que é utilizado nas aulas, que sintetiza as informações trabalhadas e gera um relatório de aula, baseado nas informações que o próprio aluno compreendeu e registrou.

Perceba que vários são os seus desafios! Pergunte-se: "Qual sistema operacional consegue atender à necessidade de processamento, armazenamento e compartilhamento de recursos de que essa empresa de consultoria necessita?". Sua primeira entrega será pautada em:

- 1º: verificar qual é o sistema operacional utilizado na empresa atualmente, de forma a entender o seu funcionamento e, com isso, ter fundamentos para a escolha

do próximo sistema operacional que a consultoria utilizará. Estamos partindo da premissa de que a empresa utilize ainda o MS-DOS e realize apenas algumas ações no sistema.

Observe que não é necessário ter conhecimento prévio do parque tecnológico da empresa, no entanto saber identificar, de acordo com o seu segmento de mercado, o que ela precisa em termos de tecnologias, *softwares* e *hardwares* sempre será um diferencial.

Não pode faltar

Vamos conhecer como os pesquisadores, professores e estudiosos da área definem um sistema operacional? No decorrer de sua leitura, você será apresentado a algumas referências, tais como a de Machado e Maia (2013, p. 3), que definem um sistema operacional como: “[...] um conjunto de rotinas executado pelo processador, de forma semelhante aos programas dos usuários. Sua principal função é controlar o funcionamento de um computador, gerenciando a utilização e o compartilhamento dos seus diversos recursos, como processadores, memórias e dispositivos de entrada e saída”. Os sistemas operacionais têm, basicamente, duas funções:

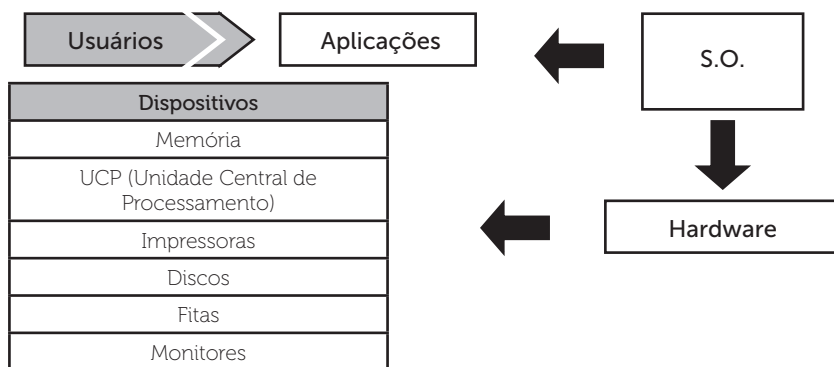
- facilitar o acesso a recursos do sistema;
- organizar o compartilhamento de recursos de forma a garantir a sua proteção.



Assimile

Observe a figura a seguir, que representa essas funcionalidades principais:

Figura 1.1 | Funções do sistema operacional



Fonte: Adaptado de Machado e Maia (2013, p. 4)

O que a Figura 1.1 indica é a interação entre o usuário e o *hardware*, sendo essa facilitada através da interface dos sistemas operacionais.



Refleta

“Cabe, então, ao sistema operacional servir de interface entre os usuários e os recursos disponíveis no sistema computacional, tornando esta comunicação transparente, além de permitir um trabalho mais eficiente e com menores chances de erros” (MACHADO; MAIA, 2013, p. 4).

Quanto à facilidade de acesso, o sistema operacional assume um papel de interface entre os usuários e o *hardware*, algo que é próprio dos sistemas operacionais, e a isso atribui-se o nome de máquina virtual.

Os sistemas operacionais também são responsáveis por gerenciar os recursos das máquinas. Isso significa que, mesmo sendo processos concorrentes, como vários usuários compartilhando uma impressora, ou, ainda, um usuário usando em seu computador um editor de texto, a internet e também uma calculadora, esse uso pode acontecer simultaneamente porque o sistema operacional está gerenciando os recursos de processamento e armazenamento de forma apropriada para que o desempenho da máquina permaneça estável.

Se analisarmos as funcionalidades, é possível dizer que os sistemas operacionais trabalham em camadas para a realização de suas tarefas: os usuários interagem com as aplicações, que interagem com o sistema operacional; esse, por sua vez, se comunica com os dispositivos de *hardware*. Veja a figura a seguir, que exhibe esse modelo de comunicação em camadas que o sistema operacional pode exercer:

Figura 1.2 | Máquina de camadas

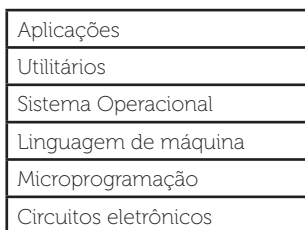
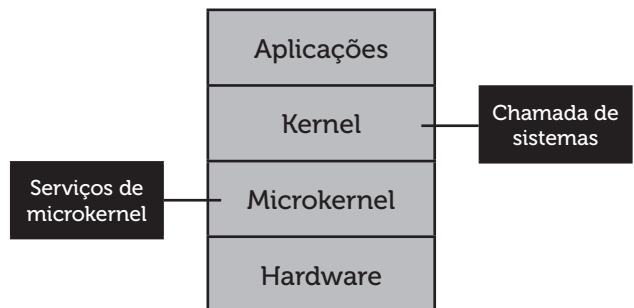


Figura 1.3 | Organização do sistema de Kernel e Microkernel



Fonte: Machado e Maia (2013, p. 6)

Fonte: Adaptado de Oliveira et al. (2010, p. 26)

Para que os comandos possam ser interpretados pelas máquinas, que só processam informações em linguagem binária, ou seja, 0 e 1, todos os comandos, ações e operações exercidas precisam ser codificados, para que a máquina possa processar e exibir o resultado dessa ação. O programa do sistema operacional responsável por essa tarefa é o interpretador de comandos (OLIVEIRA et al., 2010). Assim que o usuário inicia sua sessão de trabalho, o interpretador recebe esses comandos e faz uma chamada de sistema, sendo o núcleo do sistema operacional, também chamado de Kernel. Ele é composto por um processador, memória, sistema de arquivos e é também responsável pela gerência dos dispositivos de entrada e saída. Veja como funciona na Figura 1.3 esse mecanismo. Kernel é o responsável, portanto, pelas chamadas do sistema, e o microkernel pelo gerenciamento dos serviços.

Agora que você já conheceu como é organizado o sistema computacional e o sistema operacional, vamos realizar um breve levantamento acerca da evolução dos sistemas operacionais.

O primeiro sistema operacional desenvolvido em 1953, conhecido como “monitor”, foi criado por uma equipe da General Motors que utilizava o computador IBM701. Esse sistema operacional, segundo Machado e Maia (2013), foi reescrito para o IBM704. Em 1950, as principais linguagens de programação, conhecidas como de “alto nível”, como FORTRAN e COBOL, deram espaço para que outras linguagens de programação fossem desenvolvidas, de forma a manter o foco sobre a interface entre usuários e máquinas, e não apenas à comunicação entre máquinas de processar e armazenar. Com isso, os sistemas operacionais também evoluíram, objetivando, principalmente, facilitar os processos de codificação, submissão, execução e depuração de *softwares*.

Quadro 1.1 | Desenvolvimento embrionário

Sistemas Operacionais década 1950	Descrição
Monitor – 1953	Primeiro sistema operacional.
SOS (Share Operating System) FMS (Fortran Monitor System) IBSYS	Incorporou rotinas de processos de entrada e saída de dados conhecidos como IOCS-Input Output Control System. Usados em máquinas IBM.
Atlas	Desenvolvido pela Universidade de Manchester, que implementou esquema de paginação e hierarquia entre memória principal e secundária.

Fonte: Machado e Maia (2013)

Em 1960, chegou a vez dos circuitos integrados, que permitiram a adesão de sistemas computadorizados pelas empresas. O primeiro motivo foi a diminuição dos equipamentos; o segundo, a conseqüente redução de custos. No entanto, os *softwares* precisavam acompanhar essas evoluções também, tornando-se mais acessíveis e amigáveis aos usuários. Com isso, essa década proporcionou avanços em sistemas operacionais, técnicas de programação, multiprocessamento, *time-sharing* e memória virtual.

Quadro 1.2 | Evolução dos sistemas operacionais II

Sistemas Operacionais década 1960	Descrição
MCP (Master Control Program) – 1963	Feito para o computador B-5000 com sistema de multiprocessamento assimétrico, que permitia multiprogramação e uso de memória virtual.
System/360 – 1964	Lançado pela IBM, era compatível com máquinas de tamanhos e configurações diferentes, o que permitia permanecer com aplicativos já existentes, mesmo alterando o porte da máquina.
OS/360	Lançado pela IBM. Totalmente compatível com o System/360, porém, adotou-se esse nome de sistema operacional para as máquinas que ofereciam melhores configurações ou diferenciadas. Oferecia tempo de resposta menor para os usuários.
CTSS (Compatible Time-Sharing System)	Um dos primeiros sistemas operacionais com tempo de processamento compartilhado. Desenvolvido pelo MIT (Massachusetts Institute of Technology) para o computador IBM7094. Podia compartilhar processamento com até 32 usuários. Base para o MULTICS.
MULTICS (Multiplexed Information and Computing Service) – 1965	Foi desenvolvido para os computadores GE modelo 645. Tinha também implementadas memória virtual, paginação e multiprogramação.
PDP – 8 da Digital Equipment Corp (DEC)	Esse computador foi considerado revolucionário por ser de baixo custo e de pequeno porte. Ken Thompson, do projeto do Multics, usou este computador para trabalhar no desenvolvimento do sistema operacional Unix.

Fonte: Adaptado de Machado e Maia (2013)

A partir de agora, serão apresentados os sistemas operacionais das décadas de 1970 a 2010:

Quadro 1.3 | Evolução dos sistemas operacionais III

Sistemas Operacionais década 1970	Descrição
PDP-11 da Digital Equipment Corp (DEC)	Acompanhou uma tendência de miniaturização.
VAX/ VMS (Virtual Memory System)	Esse já foi desenvolvido para computadores de 32 bits.
CP/M (Control Program Monitor) – Digital Research	Esse era o sistema operacional dos primeiros microcomputadores.

Fonte: Adaptado de Machado e Maia, (2013)

Paralelamente a tais evoluções, os mecanismos de processamento de informações também foram aprimorados. Em 1971, a Intel lançou o seu primeiro microprocessador, o Intel 4004, e, três anos depois, o Intel 8080. Outra empresa, a Zilog, lançou nessa mesma época o Z80. Essa foi uma década muito importante, pois Steve Jobs e Steve Wozniak desenvolveram o Apple II de 8 bits e a empresa Microsoft foi fundada. Nessa década, iniciou-se a popularização das redes de computadores nas empresas da internet. Acompanhando também essa evolução, a arquitetura de computadores

passou a trabalhar com o conceito de multiprocessadores.

Quadro 1.4 | Evolução dos sistemas operacionais IV

Sistemas Operacionais década 1980	Descrição
DOS (Disk Operating System)	Comercializado e desenvolvido pela Microsoft.
Unix (Bekeryley Software Distribution – BSD)	Nessa década, esse sistema operacional foi bastante estudado e melhorado na universidade de Berkeley – Califórnia. Esse sistema já contava com a programação compatível com a comunicação em rede através dos protocolos TCP/IP.
SunOs	Em 1982, a Sun Microsystems foi fundada e lançou esse sistema operacional, que, em seguida, passou a ser o Sun Solaris.
Microsoft Windows	Interface gráfica.
OS/2	Interface gráfica.
Mac OS	Integrou a interface gráfica.
Novell Netware	Sistema operacional para gerenciamento de redes de computadores.
Microsoft LAN Manager	Sistema operacional para gerenciamento de redes de computadores.
PLURIX – 1982 a 1986	Desenvolvido pelo Núcleo de Computação Eletrônica (NCE) da Universidade Federal do Rio de Janeiro. Utilizado no computador Pegasus do NCE.
TROPIX	Evolução do PLURIX – multiusuário e multitarefa, era distribuído em versões gratuitas, seguindo a filosofia Unix.

Fonte: Adaptado de Machado e Maia (2013)

Muitos avanços tecnológicos, tanto em *hardware* e *software* como em redes de computadores, marcaram os anos 1990. Os protocolos TCP/IP tiveram de ser padronizados, para que os acessos em rede e o seu controle pudessem ser realizados e otimizados pelos sistemas operacionais. Outro fator importante é a arquitetura de redes cliente/servidor, que permitiu o desenvolvimento de sistemas operacionais dedicados, ou seja, que passaram a controlar as demandas de integração com os servidores web (Apache), com os bancos de dados (MySQL), e, ainda, com os serviços de correios eletrônicos (SendMail) e administração de arquivos. Os sistemas operacionais SunSolaris, IBM-AIX e Unix HP-UX consolidaram-se no mercado corporativo. Confira, a seguir, um breve levantamento dos principais sistemas operacionais dessa década:

Quadro 1.5 | Evolução dos sistemas operacionais V

Sistemas Operacionais década 1990	Descrição
Microsoft Windows	Predominaram o mercado nessa década.
Unix	Foi fortalecido como sistema operacional que fornecia maior segurança.

Linux	Linus Torvalds (1991), em conjunto com outros desenvolvedores, melhorou o Kernel desse sistema operacional, que é atualmente bastante utilizado no setor público e nas faculdades e universidades.
Windows NT	Lançado em 1993 pela Microsoft, veio para substituir as versões do DOS e os anteriores do Windows.

Fonte: Adaptado de Machado e Maia (2013)

A década de 2000 também é um marco em desenvolvimento de *softwares*, *hardwares* e das telecomunicações e o seu crescimento, junto à ascensão dos *notebooks*, *netbooks* e, também, os celulares. Com esse desenvolvimento, a evolução da internet e suas inúmeras possibilidades de realização de transações comerciais disparou com os sistemas de busca como Google e ainda a consolidação das redes sociais. Basicamente, os mesmos sistemas operacionais da década de 1990 foram aprimorados e se tornaram, quanto à interface, ainda mais intuitivos e inteligentes. Passaram a ter procedimentos de atualização e correção de erros a partir da conexão com a internet, realizando verificações automáticas.

Em 2010, a novidade passou a ser a computação em nuvem, ou *cloud computing*, como é conhecida. Houve, também, uma vasta popularização de equipamentos celulares, *tablets* e *smartphones* em geral, o que corroborou com o surgimento de sistemas operacionais, como o Android da gigante Google, que veio para ficar e concorrer com o sistema operacional IOS de outra importante empresa, a Apple. Os principais sistemas operacionais para *smartphones* têm sido:

Quadro 1.6 | Evolução dos sistemas operacionais VI

Sistemas Operacionais década 2010	Descrição
Symbian OS	Desenvolvido e utilizado predominantemente nos celulares da Nokia. É multitarefa, eficiente em aplicações em tempo real e estável.
Windows Mobile	Integrado ao Microsoft Exchange Server, sincroniza e-mails e arquivos de computadores pessoais. Boa integração com dispositivos Microsoft, como Xbox 360 e o Zune.
Android	Desenvolvido pelo consórcio entre 47 empresas lideradas pelo grupo Google. É um sistema aberto. Suporta diversos padrões, como bluetooth, EDGE/ GSM, 3G, 4G e wi-fi.
Blackberry OS	Desenvolvida por Research In Motion (RIM). Pioneira em smartphones corporativos. Suporta sincronização com Microsoft Exchange, Lotus Domino/ Novell Group Wise, e serviços como e-mail, calendário, tarefas, notas e contatos, quando usados com o Blackberry Enterprise Server.
iOS	Apple, fundamento no Mac OS X. Divide-se em três partes: machine level software, que visa atender às demandas de ações dos usuários; o system level software, que responde às funções principais do sistema; e user level software, que visa atingir o nível de gerenciamento de ações de um único usuário. Versão 4.0 incorpora funções multitarefa. Interface amigável.

Fonte: Adaptado de Machado e Maia (2013)

É importante reforçar as responsabilidades dos sistemas operacionais: gerenciar os recursos, ou seja, conceder ou negar acessos aos recursos de *hardware* que são solicitados pelas aplicações; prover serviço que facilite a realização das tarefas; e, por fim, ser a interface entre a aplicação e o *software*. Dessa forma, seguem as áreas pelas quais os S.O. são responsáveis: processos, memória, dispositivos de entrada e saída, sistemas de arquivos, segurança, redes e as interfaces com o usuário mencionado (STUART, 2011).

Aproveite para assistir à webaula deste material e aprenda ainda mais!



Vocabulário

Idem: a palavra "idem" significa que aquela informação foi baseada nas pesquisas do autor supramencionado no(s) parágrafo(s) anterior.

Multiprocessamento: ocorre quando o sistema operacional executa mais de um processo simultaneamente.

Time-sharing: significa tempo compartilhado. Quando há algum tempo ocioso entre um processo e outro, o sistema operacional executa outro processo de forma que se possa alternar entre vários processos, transmitindo ao usuário a impressão de que os processos estão sendo executados simultaneamente. No entanto, eles compartilham o tempo de execução.

Memória virtual: conceito utilizado para indicar os processos armazenados temporariamente no disco rígido do computador, no caso de a memória RAM (*Random Access Memory*) estar totalmente ocupada com o armazenamento temporário das informações que estão sendo mostradas na tela do usuário.

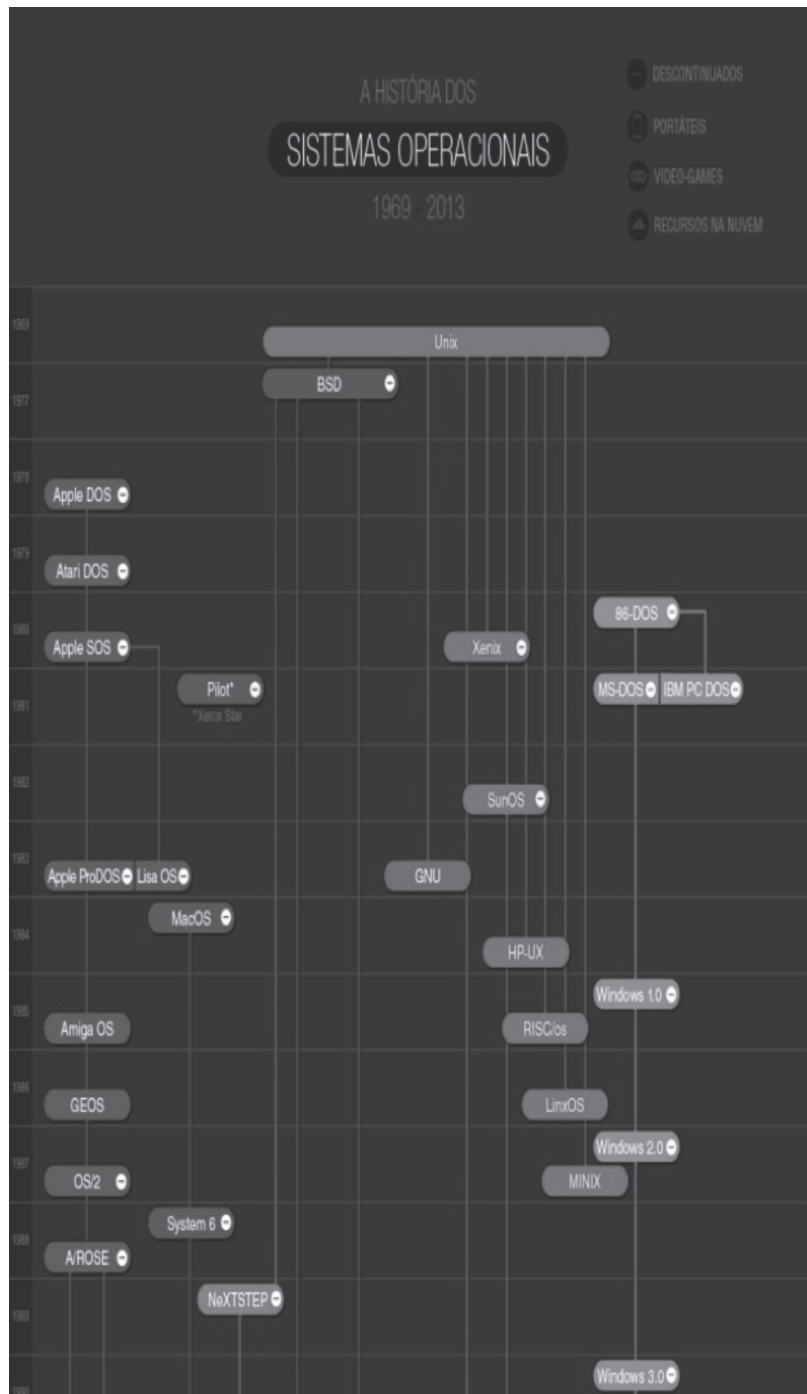


Faça você mesmo

1. Acesse a referência www.tecmundo.com.br e confira as principais evoluções em sistemas operacionais, exemplificadas neste infográfico.
2. Faça um levantamento das máquinas, ou seja, do *hardware* que era utilizado cada um desses sistemas operacionais.
3. Um filme bastante recomendado é **Piratas do Vale do Silício**. Você saberá um pouco mais sobre as empresas Apple e Microsoft, bem como sobre a evolução do *hardware* e dos sistemas operacionais desenvolvidos por essas empresas!



Exemplificando





Fonte: Disponível em: <www.tecmundo.com.br>. Acesso em: 16 jul. 2015



Pesquise mais

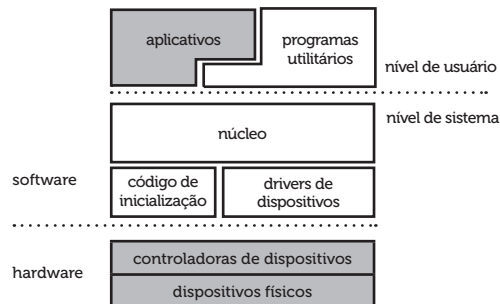
Saiba mais sobre o impacto de troca de versões de sistemas operacionais no gerenciamento de recursos. Disponível em: <<http://support.hp.com/br-pt/document/c01910697>>. Acesso em: 17 jun. 2015.

Sem medo de errar

Aplicação dos procedimentos de atuação convenientes à SP.

Agora, você terá de apresentar aos colegas da consultoria uma visão geral de funcionamento dos sistemas operacionais. Com base nessa situação, analise a figura a seguir e descreva de que forma poderá explicar ao seu público-alvo a funcionalidade de um sistema operacional:

Figura 1.4 | Estrutura de um sistema operacional



Fonte: Maziero (2013, p. 9)

Primeiramente, como observado na figura 1.4, a estrutura de um sistema operacional é analisada em forma de camadas, ou módulos que contêm uma responsabilidade específica para executar as responsabilidades desse *software*. A considerar que o sistema operacional é a interface entre o usuário, o *hardware* e os *softwares* ali instalados, é necessário que se compreenda:

- quais são os dispositivos que o S.O. precisará gerenciar;
- que tipo de operações são realizadas para que sejam inicializados os *softwares* que interpretam os comandos e que *drivers* são ativados nesse processamento;
- quais são os códigos de inicialização e que processos estão envolvidos;
- explicar, em nível de usuário, quais são as finalidades de um sistema utilitário

(que executa ações que o sistema operacional não contempla), e, ainda, quais são os aplicativos e a sua eficiência de acordo com o sistema operacional;

- assistir ao vídeo que explica a evolução dos sistemas operacionais para facilitar como se deu a evolução e como eles também podem evoluir quanto à escolha do seu e a importância disso para a modernização dos processos que estão dispostos a implementar. Disponível em: <<https://www.youtube.com/watch?v=tV3xeB8Pt2I>>. Acesso em: 17 jun. 2015.



Atenção!

Uma das funções primordiais de um sistema operacional é verificar os dispositivos, carregar a memória e executar os programas de acordo com as respectivas solicitações dos usuários.



Lembre-se

Você viu alguns dos sistemas operacionais que marcaram o desenvolvimento desses e de sua capacidade de gerenciamento de recursos e integração com outros softwares e dispositivos. Agora, você precisa fazer um levantamento sobre o MS-DOS, um sistema operacional precursor, que a Microsoft lançou, como observado no material. Portanto, não se esqueça também de que o modo de funcionamento desse S.O. contempla e prioriza o uso de comandos. Acesse o prompt do MS-DOS de seu computador e conheça alguns de seus comandos.

Avançando na prática

Pratique mais!

Instrução

Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.

Definição, conceitos e histórico dos sistemas operacionais

1. Competência de fundamentos de área

Conhecer a evolução dos sistemas operacionais e suas respectivas especificidades; conhecer e saber identificar os principais processos e como ocorre o compartilhamento de recursos; conhecer como se dá a gerência de processos e de armazenamento de arquivos; conhecer e saber gerenciar os dispositivos de entrada e saída.

2. Objetivos de aprendizagem	Saber fazer a manipulação das informações do sistema operacional; ter conhecimento sobre as principais funcionalidades e como gerenciá-las, além de saber analisar a utilização de recursos e promover a sua otimização.																						
3. Conteúdos relacionados	Definição, conceitos e histórico dos sistemas operacionais.																						
4. Descrição da SP	Compreenda as principais evoluções do MS-DOS (<i>Microsoft-Disk Operating System</i>), um dos sistemas operacionais mais utilizados na década de 1980 e que, até os dias de hoje, permanece como base dos sistemas operacionais Windows, da Microsoft. Suponha que a consultoria ainda gerencie muitos arquivos e configurações de seu computador pelo MS-DOS. Então, conheça algumas das funcionalidades desse sistema operacional e compreenda quais são as ações que eles efetuam no sistema operacional:																						
5. Resolução da SP	<p>Quadro 1.7 Descrição dos comandos internos mais utilizados</p> <table> <tr> <th>Comandos internos</th><th>Descrição</th></tr> <tr> <td>CD</td><td>Comando usado para acessar diretórios, pastas e arquivos. Identifica o diretório atual.</td></tr> <tr> <td>DIR</td><td>Esse comando lista todos os arquivos de um diretório.</td></tr> <tr> <td>PATHY</td><td>Define a sequência, o caminho de comandos.</td></tr> <tr> <td>TYPE</td><td>Exibe o conteúdo de um arquivo de texto.</td></tr> <tr> <td>DEL</td><td>Comando que exclui arquivos.</td></tr> <tr> <td>REN</td><td>Renomear arquivo.</td></tr> <tr> <td>MD</td><td>Cria um novo diretório.</td></tr> <tr> <td>RD</td><td>Remove arquivo de um diretório.</td></tr> <tr> <td>CLS</td><td>Limpa a tela.</td></tr> <tr> <td>DATE</td><td>Exibe e define a data.</td></tr> </table>	Comandos internos	Descrição	CD	Comando usado para acessar diretórios, pastas e arquivos. Identifica o diretório atual.	DIR	Esse comando lista todos os arquivos de um diretório.	PATHY	Define a sequência, o caminho de comandos.	TYPE	Exibe o conteúdo de um arquivo de texto.	DEL	Comando que exclui arquivos.	REN	Renomear arquivo.	MD	Cria um novo diretório.	RD	Remove arquivo de um diretório.	CLS	Limpa a tela.	DATE	Exibe e define a data.
Comandos internos	Descrição																						
CD	Comando usado para acessar diretórios, pastas e arquivos. Identifica o diretório atual.																						
DIR	Esse comando lista todos os arquivos de um diretório.																						
PATHY	Define a sequência, o caminho de comandos.																						
TYPE	Exibe o conteúdo de um arquivo de texto.																						
DEL	Comando que exclui arquivos.																						
REN	Renomear arquivo.																						
MD	Cria um novo diretório.																						
RD	Remove arquivo de um diretório.																						
CLS	Limpa a tela.																						
DATE	Exibe e define a data.																						

Quadro 1.8 | Descrição dos comandos externos mais utilizados

Comandos externos	Descrição
ATTRIB	Exibe ou define propriedades de arquivos.
BACKUP	Cria cópias de segurança.
FORMAT	Permite formatar um diretório.
RESTORE	Restaura arquivos de segurança criados pelo backup.
FASTOPEN	Reduz o tempo de abertura de arquivos e diretórios.
FDISK	Permite configurar um disco rígido.
JOIN	Permite a configuração de um disco rígido para ser usado no DOS.
XCOPY	Copia arquivos e diretórios e pastas.
LABEL	Permite criar, alterar, excluir o nome de volume de disco.
MODE	Permite definir quantidade de caracteres por linha, velocidade de transmissão de dados, ajustes de tela e configurar dispositivos de impressão.



Faça você mesmo

Teste os comandos! Mas cuidado: como você viu, alguns podem excluir arquivos e até pastas e diretórios inteiros! Então, com o auxílio de seu mediador, professor e colegas, realize essa atividade e veja os resultados.



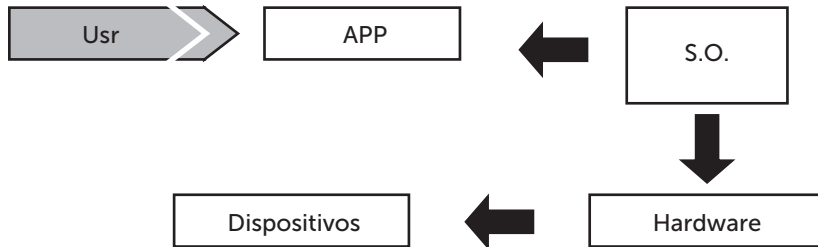
Lembre-se

“Sem um sistema operacional, um usuário para interagir com um computador deveria conhecer profundamente diversos detalhes sobre hardware do equipamento, o que tornaria o seu trabalho lento e com grandes possibilidades de erros. As duas principais funções são: ‘facilidade de acesso aos recursos do sistema’ e ‘compartilhamento de recursos de forma organizada e protegida’” (MACHADO; MAIA, 2005, p. 2).

Faça valer a pena!

1. Analise a figura e responda:

Figura 1.5 | Funções do sistema operacional



Fonte: Adaptado de Machado e Maia (2013, p. 4)

- A figura representa a interface que o sistema operacional realiza entre *softwares* e *hardwares*.
- A figura apresenta a codificação da linguagem de máquina para os dispositivos.
- A figura evidencia que o usuário precisa saber linhas de comando das aplicações.
- A figura mostra a interface da aplicação.
- A figura evidencia a importância do sistema operacional para gerar relatórios.

2. Defina o que é um sistema operacional.

3. O conceito de Máquina Virtual está presente em qual das alternativas a seguir?

- Integra um computador detalhes sobre *hardware* e *software*.
- É a estrutura de um sistema operacional que é analisada em forma de camadas.
- São códigos de inicialização e que processos do sistema operacional.
- Essa tem a função de simular e assumir um papel de interface entre os usuários e o *hardware*, algo que é próprio dos sistemas operacionais.
- Visa integrar o nível de gerenciamento de ações de um único usuário.

4. Assinale a alternativa que apresenta os sistemas operacionais da década de 1960:

- a) Symbian OS, Windows Mobile, Android, Blackberry OS, iOS.
- b) Microsoft Windows, Unix, Linux, Windows NT.
- c) Monitor, SOS, FMS, IBSYS, Atlas.
- d) PDP- 11, VAX/ VMS, CP/M.
- e) MCP, System/360, OS/360, CTSS, MULTICS, PDP- 8.

5. Associe, no quadro a seguir, o sistema operacional à sua respectiva descrição.

Sistemas Operacionais década 1990	Descrição
a) Microsoft Windows	() Linus Torvalds (1991), em conjunto com outros desenvolvedores, melhorou o Kernel desse sistema operacional, que é atualmente bastante utilizado no setor público e nas faculdades e universidades.
b) Unix	() Lançado em 1993 pela Microsoft, veio para substituir as versões do DOS e os anteriores do Windows.
c) Linux	() Predominou no mercado nessa década.
d) Windows NT	() Foi fortalecido como sistema operacional que fornecia maior segurança.

6. Quais são os principais sistemas operacionais da década de 2010 e seus respectivos fornecedores?

7. Identifique, dentre as afirmações, as verdadeiras ou falsas e assinale a alternativa correspondente:

I - O primeiro sistema operacional desenvolvido em 1953, conhecido como "monitor", foi criado por uma equipe da General Motors que utilizava o computador IBM701. Esse sistema operacional, segundo Machado e Maia (2013), foi reescrito para o IBM704.

II - Em 1971, a Intel lançou o seu primeiro microprocessador, o Intel 4004, e, três anos depois, o Intel 8080.

III - CP/M (*Control Program Monitor*) – *Digital Research*: Esse era o sistema operacional dos primeiros microcomputadores.

Está correto o que se afirma em:

- a) I, II, III.
- b) Apenas I.
- c) II e III.
- d) Apenas II.
- e) I e III.

Seção 1.2

Tipos de sistemas operacionais: monoprogramáveis, multiprogramáveis e multiprocessamento

Diálogo aberto

Após conhecer como se deu o desenvolvimento dos sistemas operacionais e qual a relação que têm com o desenvolvimento de *hardware*, a partir de agora, você terá contato com os tipos de sistemas operacionais e também saberá identificá-los de acordo com as características de cada um. Iniciamos a apresentação dos sistemas operacionais chamados de monotarefa ou monoprogramáveis; em seguida, são descritos os sistemas multiprogramáveis. Na sequência, são apresentados os conceitos inerentes aos sistemas operacionais conhecidos como multiprocessamento e a sua diferença com os demais.

Além desses, você ainda verá uma breve descrição do que são os sistemas distribuídos e qual a sua relação com os sistemas operacionais da atualidade.

Nesse contexto, ao aproximar a teoria com a sua aplicação em um contexto profissional, conforme sugerido, vamos elaborar uma análise que vise diferenciar os tipos de sistemas operacionais e, principalmente, quais são as características de *hardware* e *software* que podem ser adicionados e executados nesses sistemas. Afinal, como se dá essa relação entre o *hardware* e o *software*?

Com isso, você precisa saber como se deu a evolução dos sistemas operacionais e de que forma estão organizados. Esses estudos proporcionarão conhecer quais são suas respectivas especificidades. É importante salientar que, quando se trata de sistemas operacionais, as suas duas funcionalidades básicas são:

- facilitar o acesso aos recursos do sistemas;
- compartilhar esses recursos.

Uma das visões recomendadas para melhorar a compreensão do modo de funcionamento dos sistemas operacionais é dividir o sistema computacional em

camadas. Nesse sentido, serão descritas as camadas mais comumente analisadas.

Por exemplo, compreender que há, além do *hardware* e do *software*, outros níveis a considerar, inclusive em nível de gerenciamento dos pacotes de serviços de compartilhamento de recursos, de gerenciamento e alocação de memória, e, ainda, de como funcionam os sistemas de arquivos. Aqui, você terá contato com os fundamentos para a realização desses estudos. Desde já, bons estudos e práticas a você!

Não pode faltar

O desenvolvimento dos sistemas operacionais está diretamente ligado ao desenvolvimento dos computadores. Muitas pessoas e esforços foram empreendidos até que se obtivesse esse nível de organização por camadas. Ao se realizar um breve levantamento histórico, é possível destacar os seguintes marcos desse processo:

Quadro 1.9 | Evolução dos computadores

Ano	Descrição	Pesquisador
1642	Máquina de somar.	Blaise Pasqual.
1673	Acumulador – máquina de somar e multiplicar.	Gottfried Leibniz.
1820	Máquina com as operações: adição, subtração, multiplicação e divisão.	Charles Colmar.
1822	Máquina para calcular equações polinomiais.	Charles Babbage.
1833	Máquina analítica que foi criada para executar qualquer tipo de operação. Mudou o conceito de unidade central de processamento, memória e controle de dispositivos.	<ul style="list-style-type: none">• Charles Babbage (hardware). Foi considerado, por isso, o pai do computador;• Ada Byron (software): programava as instruções da máquina.
1854	Criou a lógica booleana, que é a base para a computação digital.	George Boole.
1890	Mecanismo com cartões perfurados para ajudar no processamento do censo nos EUA.	Herman Hollerith.
1896	Criou a empresa Tabulating Machine Company. Hollerite tornou-se sinônimo de máquinas de cartão perfurado.	Herman Hollerith.
1924	Tabulating Machine Company, de Herman Hollerith, transformou-se na empresa IBM.	Herman Hollerith.
1930	Várias iniciativas para se criar máquinas de calcular.	Konrade Suze (Alemanha - Z-I); John Vincent Atanasoft e Clifford Berry (ABC – primeiro computador eletrônico).

1936	Máquina de Turing: processamento de símbolos. Algoritmo para processamento de sequências de ações.	Alan Turing.
------	--	--------------

Fonte: Maziero (2013, p. 9)

A partir da década de 1940, seguindo a lógica binária de George Boole, foram desenvolvidas ações que precisariam ser executadas por sistemas que operassem em dispositivos como relés e válvulas, usados nos computadores dessa década. Nessa época, houve uma aceleração de alguns processos relacionados à automatização de processos manuais, em função da Segunda Guerra Mundial, o que impulsionou as pesquisas e o desenvolvimento dos computadores eletromecânicos. Esses eram muito grandes e a base de válvulas. O seu foco principal era melhorar o tempo de processamento de cálculos, então ainda eram chamados de calculadoras.

Alan Turing, em 1943, na Inglaterra, trabalhava na máquina de decifrar conhecida como Colossos, que tinha por objetivo traduzir e interpretar as mensagens alemãs chamadas de Enigma. Os aliados conseguiram, a partir dessa máquina, uma considerável vantagem, principalmente para interpretar a mensagem que indicava o chamado "Dia D". Portanto, esse invento já se mostrou desde o princípio importante: fosse para otimização do tempo, ou mesmo para a tomada de decisões (MACHADO; MAIA, 2013).



Pesquise mais

Leia mais informações sobre a evolução dos computadores, dos sistemas operacionais e também sobre Alan Turing:

<<http://producao.virtual.ufpb.br/books/camyle/introducao-a-computacao-livro/livro/livro.chunked/ch01s02.html>>. Acesso em: 8 jun. 2015.

Enquanto isso, nos Estados Unidos, o professor Howard Haiken, de Harvard, construiu o primeiro computador eletromecânico chamado de Mark I. Essa pesquisa foi incentivada pela IBM e o invento seguia os mesmos princípios da máquina analítica de calcular de Babbage.

Depois do Mark I, o ENIAC foi desenvolvido com tecnologia digital e eletrônica, sendo a sua criação atribuída aos engenheiros J. Presper Eckert e John W. Mauchly na Universidade da Pensilvânia. Foi bastante utilizado para realizar cálculos balísticos e para o projeto da bomba de hidrogênio. Ficou em operação por 9 anos, de 1946 a 1955 (MACHADO; MAIA, 2013). Como você pode ver, as pesquisas foram motivadas, principalmente, para auxiliar os militares na elaboração de suas estratégias.

Nesse projeto, do ENIAC, também estava o professor John Von Neumann. Já ouviu

falar? É claro que sim! A partir do modelo elaborado por Neumann é que as máquinas são desenvolvidas até hoje. Ele propôs um modelo de armazenamento central tanto para instruções quanto para dados. Por esse motivo, tornou-se um modelo da arquitetura computacional até os dias de hoje, o que uniu em uma única máquina os princípios computacionais de Turing e de Babbage. Neumann ainda desenvolveu os computadores Manchester Mark I, ORDVAC e ELLIAC, na Universidade de Illinois em Pinceton, no Instituto de Estudos Avançados (IAS).

Primeiro computador com o conceito de armazenamento central, ou, como conhecido, “programa armazenado”, é o EDSAC (*Eletronic Delay Storage Calculator*) do professor Maurício Wilkes em Cambridge (1949 – Inglaterra). Há outros ainda, como o EDVAC (*Eletronic Discrete Variable Automatic Computer*) na Pensilvânia (MACHADO; MAIA, 2013).

Após a década de 1950, as válvulas dos computadores foram substituídas por transistores. Nessa década, também, o armazenamento por fita magnética marcou o acesso mais rápido aos dados.



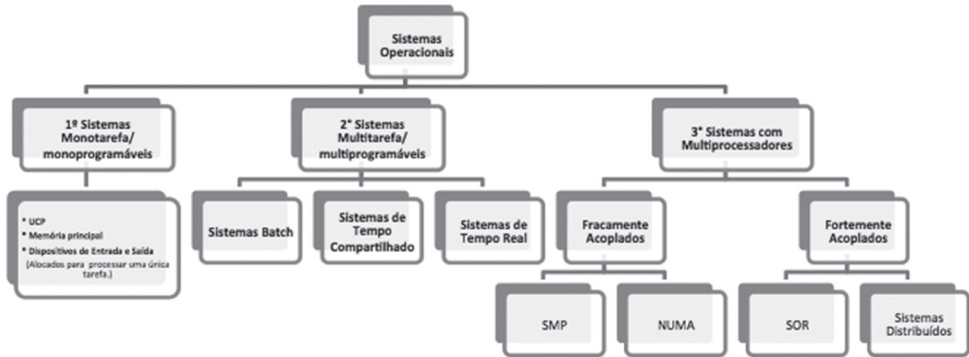
Faça você mesmo

Leia e aprofunde os seus conhecimentos em: <<http://www.fazano.pro.br/port51.html>>. Acesso em: 8 jun. 2015.

Nesse período também surgiu o conceito de processamento em lote, conhecido como “batch”, uma vez que, após a execução de todos os programas, como as informações haviam sido gravadas nas fitas magnéticas, elas eram lidas e impressas (MACHADO; MAIA, 2013), diferentemente dos que eram processados um a um pelo operador do computador.

Apenas em 1953, ficou conhecido como o primeiro sistema operacional, o “monitor”. O sistema operacional monitor foi desenvolvido por uma equipe da General Motors, que utilizava o computador IBM701. Esse sistema operacional, segundo Machado e Maia (2013), foi reescrito para o IBM704. Nas décadas de 1960 e 1970, respectivamente, com o uso dos primeiros computadores utilizados por governos, faculdades e algumas empresas, seguidos pelos computadores pessoais, utilizavam um conceito conhecido como sistema monotarefa, isto é, esses computadores eram desenvolvidos para executar apenas um tipo de tarefa, com todos os recursos da máquina dedicados a esse processamento. A evolução se deu da seguinte forma, conforme ilustrado na Figura 1.6:

Figura 1.6 | Tipos de sistemas operacionais



Fonte: Adaptado de Machado e Maia (2013, p. 16)



Refleta

“Os tipos de sistemas operacionais e sua evolução estão relacionados diretamente com a evolução do hardware e das aplicações por ele suportadas. Muitos termos inicialmente introduzidos para definir conceitos e técnicas foram substituídos por outros, na tentativa de refletir uma nova maneira de interação ou processamento. Isto fica muito claro quando tratamos da unidade de execução do processador. Inicialmente, os termos programa ou job eram os mais utilizados, depois surgiu o conceito de processo e subprocesso e, posteriormente, o conceito de thread” (MACHADO; MAIA, 2013, p. 15).

Como mencionado, os sistemas operacionais monotarefa utilizavam todos os recursos da máquina para que pudessem processar uma única tarefa. Essa característica fazia com que o processador ficasse ocioso, por exemplo, se o usuário estivesse realizando a simples tarefa de digitar um dado. Além disso, esse tipo de sistema operacional não utilizava todos os recursos de memória, se o programa não ocupasse todo o espaço existente. Quanto aos periféricos, por não haver a preocupação de compartilhar dispositivos de entrada e saída, ficavam também dedicados a um único usuário.

O segundo marco da evolução dos sistemas operacionais foi pautado no compartilhamento de recursos e na possibilidade de se trabalhar com mais de um aplicativo, ou mesmo programa, sendo processados ao mesmo tempo. Com isso, uma das preocupações ou responsabilidades dos sistemas operacionais passou a ser o gerenciamento de processamento, memória e o compartilhamento de recursos.

Nesse sentido, os sistemas multitarefa também foram classificados em monousuário e multiusuário. Sistemas multiprogramáveis monousuário eram

utilizados por apenas um usuário, o que permitia que ele realizasse várias tarefas ao mesmo tempo, como editar um texto, usar a internet, imprimir um documento. Exemplos desses são os computadores pessoais e ainda as estações de trabalho. Já os sistemas multiprogramáveis multiusuário requerem o compartilhamento de recursos como dispositivos de entrada e saída, entre outras características. Por esse motivo, os sistemas multiusuários foram classificados em três tipos, conforme ilustrado na Figura 1.6: *batch*, de tempo compartilhado e de tempo real. Confira no quadro a seguir a descrição de cada um deles:

Quadro 1.10 | Classificação de sistemas operacionais multiprogramáveis multiusuário

Tipo de sistemas operacionais multiprogramáveis multiusuário	Descrição
Batch	<ul style="list-style-type: none"> • Implementados na década de 1960; • programas/ Jobs: processados com cartões perfurados. • armazenados em fita ou disco e esperavam o processamento quando houvesse espaço suficiente de memória para esse processo. • não exigem interação do usuário. • entradas e saídas acontecem por memórias secundárias como disco ou fita. • exemplos: processamento de cálculos, ordenações, backups entre outros.
Tempo compartilhado	<ul style="list-style-type: none"> • Conhecidos como time-sharing ou on-line; • divide o tempo do processador em intervalos: time-slice ou fatias de tempo. • cada usuário possui o seu ambiente de trabalho próprio. • permite a interação dos usuários pelos dispositivos de entrada.
Tempo real	<ul style="list-style-type: none"> • Conhecidos como sistemas "real-time", tempo real, ou seja, que precisam dedicar toda a sua capacidade de processamento para executar uma determinada tarefa, até que uma outra seja priorizada; • destacam-se ainda quanto à sua complexidade. • exemplos de uso: máquina de lavar, controles de processos em refinarias de petróleo, controle de tráfego aéreo, usinas termoeletricas entre outros.

Fonte: Adaptado de Machado e Maia (2013, p. 18-20)

Vamos a mais informações sobre os sistemas multiusuários? Siga em frente!



Pesquise mais

Veja a evolução em processamento por cartões e fitas perfuradas. Disponível em: <http://www.memoriainfo.furg.br/index.php?option=com_content&view=article&id=31:cartoes&Itemid=28>. Acesso em: 18 jun. 2015.

Os sistemas de processamento em lote ou *batch* eram recomendados, pois utilizam o processador de forma otimizada e aproveitam todo o seu potencial, mas

o tempo de resposta é maior, justamente porque depende do tamanho do arquivo a processar, ou da quantidade de tarefas a realizar. Atualmente, não há a necessidade desse tipo de processamento.

Já os sistemas de tempo compartilhado dividem o processamento das tarefas por fatia de tempo. Se uma delas não for suficiente para que o processo seja concluído, o próprio sistema operacional interromperá essa execução e o processo aguardará a alocação de uma outra fatia de tempo que seja capaz de processar aquela tarefa sem interrupções. O usuário pode interagir com o sistema operacional através de comandos.

Os sistemas de tempo real diferem dos de tempo compartilhado quanto ao tempo requerido para processamento. Sendo que, nesse tipo de sistema, o processador é utilizado pelo tempo necessário à execução do programa. Não há, portanto, o conceito de fatia de tempo como nos sistemas de tempo compartilhado.



Assimile

"Enquanto em sistemas de tempo compartilhado o tempo de processamento pode variar sem comprometer as aplicações em execução, nos sistemas de tempo real os tempos de processamento devem estar dentro de limites rígidos, que devem ser obedecidos, caso contrário poderão ocorrer problemas irreparáveis" (MACHADO; MAIA, 2013, p. 19).

Conheça, também, o modo como funcionam os sistemas com múltiplos processadores. Vamos lá! Esses utilizam duas ou mais UCPs (Unidade Central de Processamento) que trabalham em conjunto. Isso significa que uma máquina pode executar vários programas simultaneamente e, além disso, que o seu processamento pode ser dividido entre os processadores. Desse modo, esses sistemas são muito utilizados para processamento de imagens e desenvolvimento aeroespacial. Esse tipo de sistema apresenta as seguintes vantagens:

- escalabilidade: termo utilizado para definir a capacidade de ampliar o potencial de processamento de dados pelo computador, através do uso de vários processadores;
- disponibilidade: a disponibilidade aqui sugerida é referente à possibilidade de manter o processo em execução, mesmo no caso de falhas. Isso significa apenas que pode ser que o processamento ocorra de forma um pouco mais lenta, no entanto os processos não deixarão de ser executados;
- balanceamento de carga: isso se dá pela capacidade de distribuição de processamento de acordo com os processadores disponíveis. O balanceamento de carga melhorar o desempenho da máquina.

Uma das características dos sistemas operacionais que trabalham com múltiplos processadores é o modo como acontece a comunicação entre as UCPs. Além disso, também são considerados o nível de compartilhamento de recursos de memória e dos dispositivos de entrada e saída. Por esse motivo, segundo Machado e Maia (2013), esses sistemas são classificados em:

- fortemente acoplados: nesse tipo de sistema, há vários processadores compartilhando uma única memória física, e os dispositivos de entrada e saída são gerenciados por um único sistema operacional. Também, são conhecidos como multiprocessadores. Exemplos de sistemas operacionais fortemente acoplados são o Unix e o Windows.



Exemplificando

Quadro 1.11 | Tipo de sistemas fortemente acoplados

Tipo	Descrição
SMP (Symmetric Multiprocessors)	Processos que levam o mesmo tempo de acesso à memória principal pelos processadores, ou seja, tempo uniforme de processamento.
NUMA (Non-Uniform Memory Access)	Trabalha com o conceito de conjunto de processos que compartilham processadores e memória principal, sendo o tempo de execução de acordo com cada conjunto. Os conjuntos de processos se interligam por uma rede de interconexão que, portanto, varia em função de sua localização.

Fonte: Adaptado de Machado e Maia (2013)

- Fracamente acoplados: possuem dois ou mais sistemas interconectados em rede, sendo que cada sistema opera de forma independente com o seu próprio sistema operacional e gerenciamento de recursos de processamento (UCPs), memória e dispositivos. Cada sistema pode ter mais de um processador. São chamados, por esses motivos, de multicomputadores.

Esse tipo de sistema era bastante utilizado na década de 1980, quando várias máquinas poderiam estar interligadas por uma linha de comunicação serial dedicada ou mesmo por uma linha telefônica pública, no entanto, o processamento das informações era realizado por um terminal central que continha o sistema capaz de processar a informação e devolvê-la ao terminal solicitante através das linhas de comunicação, trabalhando de forma centralizada (MACHADO; MAIA, 2013).



Exemplificando

Quadro 1.12 | Tipos de sistemas fracamente acoplados

Tipo de sistema fracamente acoplado	Descrição
Sistemas operacionais de rede (SOR)	Permitem que um servidor compartilhe recursos com outros computadores, servidores e dispositivos de entrada e saída cadastrados nessa rede. Ex.: redes locais (LAN).
Sistemas distribuídos	Tratam os sistemas computacionais que estão interligados como se fossem um único sistema. Permitem, inclusive, que o processamento de uma aplicação seja realizado de forma dividida. Cada parte dessa aplicação poderá ser executada por um servidor diferente. No entanto, para o usuário é como se a aplicação estivesse sendo executada em apenas um computador de forma centralizada. Ex.: Clusters de bancos de dados ou buscas na web, em que há vários servidores interligados por uma rede de alto desempenho, não importando quais são os hosts ou servidores cadastrados, e sim que compartilhem os recursos de processamento e tragam a resposta solicitada pelo usuário. Fonte: Machado e Maia (2013).

Fonte: Adaptado de Machado e Maia (2013)

Com esse conceito de interconexão e a evolução das telecomunicações, surgiram as redes de computadores. Com isso, cada sistema interligado poderia trabalhar de forma independente e oferecer serviços aos demais servidores ou hosts, o que fez com que o processamento das informações deixasse de ser realizado de forma centralizada, o que levou ao conceito de sistemas distribuídos. Bons estudos!

Sem medo de errar

Aplicação dos procedimentos de atuação convenientes a SP.

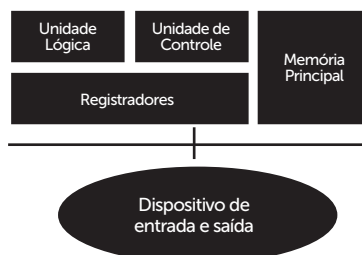
Com o intuito de apresentar os tipos de sistemas operacionais e explicar o seu modo de funcionamento, essa atividade o auxiliará na oferta desse produto à consultoria que está recebendo o seu auxílio. Nesse sentido, você precisa explicar brevemente por que não é viável manter apenas o sistema operacional MS-DOS como principal gerenciador de recursos, pois, atualmente, há a necessidade de trabalho com vários aplicativos e programas simultaneamente, principalmente no que tange ao uso eficiente dos recursos de processamento e armazenamento. Explique o funcionamento desse sistema computacional. Observe a figura e os componentes que precisam ser considerados para a escolha de controle das funcionalidades de um

sistema operacional e explique a sua importância para a eficiência da máquina em termos de processamento.

- Processador: gerencia o sistema computacional;
- unidade de controle (UC): gerencia as atividades dos componentes do computador como gravação de dados e localização de instruções;
- unidade lógica e aritmética (ULA): realiza operações lógicas e aritméticas;
- registradores: armazenam dados temporariamente;
- controlador de instruções (CI): contém o endereço da próxima instrução para o processador executar;
- apontador da pilha (AP) ou *stack pointer* (SP): refere-se às instruções que estão no topo da pilha de execução. Contém o seu endereço na memória;
- registrador de instruções (RI): armazena a instrução que será decodificada pelo processador.
- registrador de *status* ou *program status word* (PSW): armazena informações sobre os processos em execução;
- ciclo de busca e instruções do processador:
 1. Busca na memória principal o endereço CI e armazena RI.
 2. Atualiza o CI com o endereço da próxima instrução.
 3. Decodifica a instrução do RI.
 4. Busca operando em memória.
 5. Busca instrução decodificada e reinicia o processo.

Fonte: Machado e Maia (2013)

Figura 1.7 | Unidades funcionais de um sistema computacional



Fonte: Adaptado de Machado e Maia (2013, p. 23)



Atenção!

Assista ao vídeo e compreenda a evolução dos computadores e dos sistemas operacionais. Disponível em: <https://www.youtube.com/watch?v=Sx1Z_MGwDS8>. Acesso em: 22 jun. 2015.



Lembre-se

"O processador, também denominado unidade central de processamento (UCP), gerencia todo o sistema computacional controlando as operações realizadas por cada unidade funcional. A principal função do processador é controlar e executar instruções presentes na memória principal, por meio de operações básicas como somar, subtrair, comparar e movimentar dados" (MACHADO; MAIA, 2013, p. 24).

Avançando na prática

Pratique mais!	
Instrução Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.	
"Tipos de sistemas operacionais: monoprogamáveis, multiprogamáveis e multiprocessamento"	
1. Competência de fundamentos de área	Conhecer a evolução dos sistemas operacionais e suas respectivas especificidades; conhecer e saber identificar os principais processos e como ocorre o compartilhamento de recursos; conhecer como se dá a gerência de processos e de armazenamento de arquivos; conhecer e saber gerenciar os dispositivos de entrada e saída.
2. Objetivos de aprendizagem	Saber fazer a manipulação das informações do sistema operacional; ter conhecimento sobre as principais funcionalidades e como gerenciá-las, além de saber analisar a utilização de recursos e promover a sua otimização.
3. Conteúdos relacionados	Tipos de sistemas operacionais: monoprogamáveis, multiprogamáveis e multiprocessamento.
4. Descrição da SP	Agora que você sabe a importância dos sistemas operacionais, a sua evolução e como funciona o processamento das instruções, faça um levantamento do modo de armazenamento de dados. Estamos falando da memória principal. Explique o modo de funcionamento desse componente do sistema computacional, que é controlado pelo sistema operacional.

5. Resolução da SP

Para tal, você precisa:

1. Definir o que é a memória principal e quais são os seus componentes.
2. Realize um levantamento sobre os tipos de memória: secundária, ROM, RAM, cache.
3. Assista ao vídeo recomendado: <<https://www.youtube.com/watch?v=KMVMhMGab9I>>. Acesso em: 23 jun. 2015. Vale a pena o exercício! O vídeo mostra os elementos, os componentes de *hardware*.
4. Associe tais componentes ao gerenciamento de dispositivos e como se dá a alocação de memória para atender essas ações.



Lembre-se

"A memória principal, primária ou real é o local onde são armazenados instruções e dados. A memória é composta por unidades de acesso chamadas células, sendo cada célula composta por um determinado número de bits. O bit é a unidade básica de memória, podendo assumir o valor lógico 0 ou 1" (MACHADO; MAIA, 2013, p. 25)



Faça você mesmo

Aprofunde os seus estudos e faça um levantamento sobre o papel das memórias: secundária, RAM, ROM, Cache. Veja também a relevância em *hardware* dos dispositivos de entrada e saída.

Faça valer a pena!

1. Assinale a alternativa que contém os tipos de sistemas operacionais:
 - a) Redes, memória principal, multiprocessamento e distribuídos.
 - b) Monoprogramáveis, multitarefa, redes, processadores.
 - c) Monotarefa/monoprogramáveis, multitarefa/multiprogramáveis, multiprocessadores, sistemas distribuídos.
 - d) Monotarefa, multiprogramáveis, redes, barramentos.
 - e) Sistemas distribuídos, registradores, processadores, sistemas de arquivos.
2. Defina sistemas monotarefas e cite um exemplo.

3. Leia o trecho de texto a seguir e assinale a alternativa que contém os conceitos que completam as lacunas:

I. Os sistemas _____ requerem o compartilhamento de recursos como dispositivos de entrada e saída entre outras características.

II. Os sistemas _____ eram utilizados por apenas um usuário, o que permitia que ele realizasse várias tarefas ao mesmo tempo.

- a) Multiprogramáveis monousuário/multiprogramáveis multiusuário.
- b) Distribuídos/multiprocessadores.
- c) Fortemente acoplados/fracamente acoplados.
- d) Monotarefa/fracamente acoplados.
- e) Fortemente acoplados/multiusuário.

4. Defina sistemas fortemente acoplados e fracamente acoplados.

5. De acordo com o quadro abaixo, temos representados que tipos de sistemas operacionais especificamente? Assinale a alternativa correspondente:

Tipo	Descrição
SMP (Symmetric Multiprocessors)	Processos que levam o mesmo tempo de acesso à memória principal pelos processadores, ou seja, tempo uniforme de processamento.
NUMA (Non- Uniform Memory Access)	Trabalha com o conceito de conjunto de processos que compartilham processadores e memória principal, sendo o tempo de execução de acordo com cada conjunto. Os conjuntos de processos se interligam por uma rede de interconexão que, portanto, varia em função de sua localização.

- a) Monotarefa.
- b) Fortemente acoplados.
- c) Fracamente acoplados.
- d) Distribuídos.
- e) Multitarefa.

6. O exemplo mencionado no texto, conforme o trecho abaixo, indica que está sendo utilizado para esse tipo de operação qual tipo de sistema?

“Ex.: Clusters de bancos de dados ou buscas na web, em que há vários servidores interligados por uma rede de alto desempenho. Não importando quais são os hosts ou servidores cadastrados e sim que compartilhem os recursos de processamento e tragam a resposta solicitada pelo usuário” (MACHADO; MAIA, 2013).

- a) Monotarefa e multitarefa.
- b) Fortemente acoplado.
- c) Multiprogramados.
- d) Multiprocessador: fracamente acoplado: distribuído.
- e) Redes de computadores.

7. Dentre os sistemas abaixo relacionados, a qual pode ser atribuído o exemplo das redes locais?

Tipo de sistema fracamente acoplado	Descrição
Sistemas operacionais de rede (SOR)	Permitem que um servidor compartilhe recursos com outros computadores, servidores e dispositivos de entrada e saída cadastrados nessa rede.
Sistemas distribuídos	Tratam os sistemas computacionais que estão interligados como se fossem um único sistema. Permitem, inclusive, que o processamento de uma aplicação seja realizado de forma dividida. Cada parte dessa aplicação poderá ser executada por um servidor diferente. No entanto, para o usuário é como se a aplicação estivesse sendo executada em apenas um computador de forma centralizada. Ex.: clusters de bancos de dados ou buscas na web, em que há vários servidores interligados por uma rede de alto desempenho. Não importando quais são os hosts ou servidores cadastrados e sim que compartilhem os recursos de processamento e tragam a resposta solicitada pelo usuário (MACHADO; MAIA, 2013).

- a) SOR.
- b) Distribuídos.
- c) Multiprogramados.
- d) Multitarefa.
- e) Monotarefa.

Seção 1.3

Características dos sistemas operacionais multiprogramáveis

Diálogo aberto

Olá, aluno! A fim de proporcionar maior entendimento sobre os sistemas operacionais multiprogramáveis, faremos, a princípio, uma revisão dos recursos gerenciáveis pelo sistema e a forma como esses interagem. Isso facilitará o processo seguinte de aprendizagem dos sistemas multiprogramáveis.

Você já estudou alguns pontos importantes, como as funções do processador, e também pode investigar um pouco mais sobre a alocação de memória. Reveja, no Quadro 1.13, alguns itens estudados e como esses funcionam.

Lembre-se de que os recursos de processamento e de memória são essenciais para o bom funcionamento do sistema operacional:

Quadro 1.13 | Recursos gerenciáveis pelo sistema operacional I

Recursos	Descrição
Memória principal	Acesso pelo endereço que é registrado pelo MAR (<i>memory address register</i>). Através desse, a unidade de controle, saberá onde alocar os dados e a sua disponibilidade (endereço 0 ao endereço $2^n - 1$).
Memória cache	Utilizada para minimizar a diferença entre a velocidade do processador e a da leitura dos dados na memória principal. Ela é pequena, porém muito veloz, pois armazena uma parte do conteúdo da memória principal e é acessada primeiro pelo processador. Quanto menor a cache, mais rápido será o acesso ao dado. Volátil.
Memória secundária	Não volátil e armazena programas e dados a um custo baixo de processamento e alta capacidade de armazenamento.

Fonte: Adaptado de Machado e Maia (2013)

Para auxiliar na proposta de sistema operacional que será apresentada à empresa com que estamos trabalhando, agora o objetivo está em relacionar os conteúdos necessários para realizar a gerência do processador e explicar de que forma o processador trata as informações de instruções que são interrompidas e como ocorre o

tratamento das exceções. Além desses, outros fatores são importantes como entender as operações de entrada e saída e, ainda, como acontece e o que é reentrância. Verifique se os sistemas operacionais multiprogramáveis atendem às necessidades de *softwares*, *hardwares* e compartilhamento de recursos na consultoria. Siga em frente!

Não pode faltar

Além dos recursos de memória, você também precisa conhecer quais são os demais itens envolvidos no processo de gerenciamento de recursos pelo qual o sistema operacional é responsável. Veja, no quadro abaixo, outros elementos que interagem com o usuário, *hardware* e *software*:

Quadro 1.14 | Recursos gerenciáveis pelo sistema operacional II

Recursos	Descrição
Dispositivos de entrada e saída	Viabilizam a comunicação entre o sistema computacional e o ambiente externo. Dividem-se em duas categorias: como memória secundária e os que fazem interface usuário-máquina.
Barramento	Realizam a comunicação de dados entre condutores, processadores, memórias e dispositivos de E/S.
Pipelining	Técnica que permite ao processador executar várias instruções paralelamente e em estágios distintos. Divide uma tarefa em subtarefas para a execução em sequência. Enquanto uma instrução é executada, essa função permite que ele busque a próxima e, assim, otimize o tempo de processamento de modo geral.
Arquiteturas RISC e CISC	Processadores interpretarão os dados e instruções escritas em linguagem de máquina, que são especificados de acordo com a sua arquitetura. RISC (Reduced Instruction Set Computer) contém poucas instruções de máquina, que são executadas diretamente pelo hardware, trabalhando diretamente com registradores. CISC (Complex Instruction Set Computers) e possuem instruções complexas que precisam ser interpretadas por microprogramas que já vêm gravados na memória ROM. Possui pequeno número de registradores.
Software	Conjunto de programas para realizar a interface entre usuários e hardwares. Também podem ser chamados de utilitários.
Tradutor	Têm por função codificar instruções em linguagem de máquina para que possam ser interpretadas no processador (código-fonte). O produto do tradutor é um módulo objeto que pode ser montador, que traduz um programa-fonte para linguagem de máquina ou, compilador é um utilitário que gera a partir de um programa-fonte um programa não executável em linguagem de máquina. O compilador opera integrado ao sistema operacional.
Interpretador	É um tradutor que não produz um módulo objeto. Traduz e executa imediatamente as instruções (Ex.: Basic e Perl).

Linker	Gera, a partir de um módulo-objeto, um executável para permitir a identificação de subtarefas que o tradutor não conseguiu identificar. Ele também faz a relocação de memória.
Loader	Carrega na memória principal o programa que será executado.
Depurador	Permite ao usuário acompanhar a sequência de instruções em execução.

Fonte: Adaptado de Machado e Maia (2013)

Todos os elementos de *hardware* e ações de *software* acima mencionados exercem um papel fundamental para o bom funcionamento da sua estação de trabalho (computador). Nesse contexto, vamos agora aprofundar os conhecimentos nos sistemas operacionais multiprogramáveis.

Os sistemas operacionais multiprogramáveis surgiram em função da necessidade de executar e utilizar mais de um programa ao mesmo tempo e, ainda, manter um nível de processamento eficiente. Os sistemas monoprogramáveis executavam apenas um programa por vez, dedicando toda capacidade do processador a essa atividade, o que tornava o tempo de espera longo, inclusive com relação aos dispositivos de entrada e saída. Enquanto um acesso à memória era realizado, o processador ficava ocioso, como se o usuário estivesse digitando um texto ou mesmo trabalhando com cálculos.

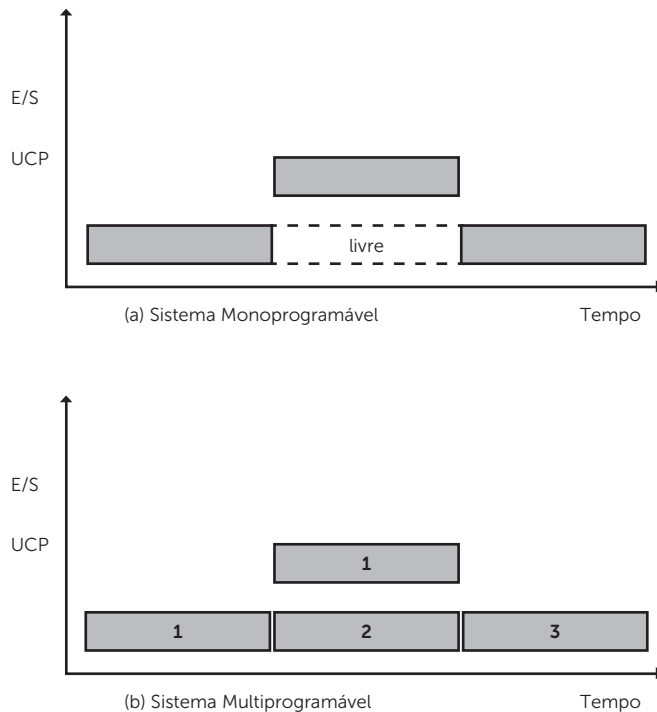
Quando se trata de sistemas operacionais multiprogramáveis, vários programas podem ser instalados e executados de forma que os processos se tornam concorrentes, ou seja, sequencialmente executados. Quando há a solicitação de uma tarefa de entrada e saída, os programas revezam o processador.

A característica da concorrência nesses sistemas se dá pelo fato de não existir queda em nível de processamento, enquanto há a alteração de execução de uma tarefa, ou seja, o seu estado deve ser idêntico ao anterior, independente do tipo de processo ou tarefa que está sendo realizado. Isso significa que o computador executará imediatamente a instrução seguinte àquela que foi interrompida. Dessa forma, mantém-se o nível de processamento sem perdas notáveis ao usuário em termos de tempo e execução de tarefas (MACHADO; MAIA, 2013).



Exemplificando

Você sabe como funciona o processamento da informação em um sistema operacional monoprogramável e em um sistema multiprogramável? Veja, a seguir, um exemplo desse procedimento:



Fonte: Machado e Maia (2013, p. 37)

Suponha que o programa deve ler um arquivo e executar 100 instruções de código por registro lido. As figuras acima representam esse processo em execução.

A figura "a" representa a leitura de um registro em um sistema operacional monoprogramável e o processador fica ocioso durante esse processo. Na figura "b", que representa os sistemas operacionais multiprogramáveis, não há a ociosidade do processador enquanto é realizada a leitura do arquivo, o que otimiza o processamento da informação e ainda o uso da memória.

Um exemplo comum é o compartilhamento de impressoras. Pode haver concomitantemente três equipamentos, como: um computador, uma impressora e ainda se considerar o trabalho de acesso à memória ou a uma memória secundária como um pen drive. Não há a necessidade de espera da realização de leitura ou do processamento do registro para que depois seja realizada uma impressão. É possível abrir um arquivo, enviar um outro para impressão e, ainda, executar uma operação em um navegador de internet, ao mesmo tempo sem que sejam esgotados os recursos do processador ou perda de eficiência e ainda ociosidade de memória.



Assimile

Observe que a Tabela 1.1 traz informações acerca da eficiência e custo de processamento nos sistemas monoprogramáveis e multiprogramáveis.

Tabela 1.1 | Comparação entre Monoprogramação e Multiprogramação

Serviço	Monoprogramação	Multiprogramação
Utilização da UCP	17%	33%
Utilização de memória	30%	67%
Utilização de disco	33%	67%
Utilização de impressora	33%	67%
Tempo total de processamento	30 min	15 min
Taxa de throughput	6 prog./ hora	12 prog./ hora

Fonte: Adaptado de Machado e Maia (2013, p. 38)

Note que nos sistemas multiprogramáveis há a ocupação praticamente total de memória enquanto é processada uma tarefa. Isso significa que não há a necessidade de se esperar o encerramento de um processo para que se inicie o outro, uma vez que o processador já pode buscar a informação previamente alocada em memória e, assim que encerrar uma tarefa, já estará preparado para iniciar o processamento de uma outra.

Outras características dos sistemas multiprogramáveis que precisamos compreender são os conceitos de:

- interrupções e exceções;
- operações de entrada e saída;
- *buffering*;
- *spooling*;
- reentrância.

Primeiramente, vamos compreender o que é uma interrupção. Ela não depende de um processo em execução, e sim ocorre em função de um evento externo ao programa que está em uso. Isso torna possível a implementação de concorrência entre os processos, que é a característica principal dos sistemas multiprogramáveis, sincronizando as tarefas e sua execução com as operações dos usuários e também o controle dos dispositivos (MACHADO; MAIA, 2013). Uma interrupção ocorre de

forma assíncrona, isso porque não está vinculada à execução de um programa que identifique o início e fim de cada agrupamento de bits.



Pesquise mais

Este vídeo aponta algumas características de sistemas operacionais. <<https://www.youtube.com/watch?v=5XFFzlyPl4g>>. Acesso em: 24 jun. 2015.

Um exemplo de interrupção ocorre quando um dispositivo de entrada ou saída encerra uma tarefa, e o processador, por sua vez, interrompe a execução daquela instrução do programa para executar as instruções de encerramento da operação sinalizada.

Com isso, a unidade de controle é acionada para verificar o que houve e iniciar a rotina de tratamento de interrupção. As instruções que forem executadas para esse tratamento de interrupção devem ser armazenadas em um registrador para que, ao retornar à execução do programa, seja possível restaurar aquelas informações e dar continuidade ao processo interrompido.

Como as instruções de tratamento ficam guardadas nos registradores, isso facilita o acesso à informação caso aquele evento volte a ocorrer e, com isso, acionar a rotina apropriada para realizar o desvio do fluxo de processamento de forma mais rápida. Há a necessidade de um controlador de pedidos de interrupção.



Refleta

"Sistemas operacionais podem ser vistos como um conjunto de rotinas executadas de forma concorrente e ordenada (Pinkert, 1990). A possibilidade de o processador executar instruções ao mesmo tempo que outras operações, como, por exemplo, operações de entrada e saída, permite que diversas tarefas sejam executadas concorrentemente pelo sistema. O conceito de concorrência é o princípio básico para o projeto e a implementação dos sistemas multiprogramáveis" (MACHADO; MAIA, 2013, p. 36).

Exceção é diretamente ligada ao programa, ou seja, é um evento ocorrido em função do processamento do programa e, por isso, também, síncrona. Um exemplo comum é o de *overflow*, que ocorre quando há uma divisão por zero e não foi previsto um tratamento no código-fonte do programa. Com isso, o sistema operacional entende que uma instrução do programa gerou um erro lógico ao ser executada, e esse problema ocorrerá todas as vezes em que o programa for executado, portanto

a solução é prever esse tipo de erro e incluir o tratamento das exceções no próprio programa.

Operações de entrada e saída eram controladas por um conjunto de instruções de entrada e saída, nos primeiros sistemas computacionais. Então, foi desenvolvido o controlador ou interface, que realiza essas operações de reconhecer os comandos e solicitações advindas dos dispositivos e que precisam se comunicar com o *hardware* e com o *software*. Sendo assim, o processador não se comunicava mais diretamente com o *hardware* e com o *software*, e sim o controlador ou interface. São dois os tipos de controladores: E/S controlada por programa e E/S controlada por interrupção.

No controlador por programa, o processador ficava aguardando e testando o estado dos dispositivos de entrada e saída até terminar a operação de E/S. Essa ação do processador é conhecida como *busy wait*, e esse tipo de controlador deixava o processador ocioso e, por esse motivo, é considerado menos eficiente do que o controlador por interrupção. Como uma evolução, assim que se iniciasse a transferência dos dados, o processador era liberado e faria a verificação de tempos em tempos para saber o estado dos dispositivos de entrada e saída, o que ficou conhecido como *pooling*.

O mecanismo de controle denominado E/S, controlado por interrupção, consiste na liberação do processador para executar outras tarefas, assim que ele realiza a execução de um comando de leitura e gravação. Então, o controlador recebe as instruções e armazena, nos registradores ou em memória, aquela informação e sinaliza na sequência o processador para que inicie o processo de tratamento da interrupção. Nesse momento, a rotina responsável é acionada e transfere os dados do registrador do controlador para a memória principal. O processador retorna ao processamento das instruções do programa, quando é encerrada a transferência das informações de tratamento, liberando agora o controlador para outra operação (MACHADO; MAIA, 2013). Isso permite a execução de várias operações de entrada e saída simultaneamente. No entanto, isso pode gerar uma sobrecarga no processador, o que reduz a sua eficiência.

Para tratar a possível perda de eficiência do processador, no caso de esse realizar muitas intervenções de controle de E/S, foi desenvolvida a técnica DMA (*Direct Memory Access*), que permite a transferência de dados diretamente da memória principal para os dispositivos de E/S, e vice-versa, sem que o processador participe dessa operação. A área da memória que é utilizada para realizar essa operação é chamada de *buffer* de entrada e saída. A função do processador, nesse caso, é apenas a de informar ao controlador a localização, qual o dispositivo de E/S e a posição inicial da memória que deverá receber os dados de leitura, a sua gravação ou mesmo o tamanho do bloco de dados.



Assimile

“No momento em que uma transferência de dados através da técnica de DMA é realizada, o controlador deve assumir, momentaneamente, o controle do barramento. Como a utilização do barramento é exclusiva de um dispositivo, o processador deve suspender o acesso ao bus, temporariamente, durante a operação de transferência. Este procedimento não gera uma interrupção, e o processador pode realizar tarefas, desde que sem a utilização do barramento, como um acesso à memória cache” (MACHADO; MAIA, 2013, p. 42).

Atualmente, quase não há a intervenção da unidade de processamento central (UCP), pois, nas novas arquiteturas, há um processador de entrada e saída, que otimiza o tempo e uso de recursos pelo computador.

Mas, além desses, como mencionado anteriormente, há também a técnica de *buffering*. Ela é responsável por fazer a transmissão dos dados dos dispositivos de entrada e saída para a memória principal, a partir do uso de registradores para fazer esse transporte.

Com isso, o dado será sempre transferido primeiramente ao *buffer*, que permitirá o acesso à informação, que deverá ser imediatamente processada. Isso faz com que os dispositivos de E/S sejam liberados para receber novas instruções e que seja reduzido o problema de diferença de processamento, leitura e gravação de novas instruções de E/S, bem como de sua execução. O *buffer* ainda permite que existam vários registros armazenados e ainda não lidos, e esses podem variar em tamanho de acordo com o tipo de informação que deverá ser lida pelo processador.

Semelhante ao processo de *buffering*, a técnica de *spooling* (*simultaneous peripheral operation on-line*), introduzida em 1950 com o intuito de aumentar a possibilidade de trabalho com processos concorrentes, trouxe a possibilidade de armazenar um conjunto de instruções ou Jobs, em fita magnética para serem processados. Essa técnica era realizada sequencialmente cada job armazenado, o que diminui o tempo de processamento e busca por cada instrução que deve ser processada. A saída desse tipo de processamento é o armazenamento da informação em outra fita magnética, ou outra área do disco rígido. Essa foi a base para o processamento batch (MACHADO; MAIA, 2013).



Faça você mesmo

1. Faça o *download* do simulador de processos de um sistema operacional SOSim. Recomendação de acesso: <http://www.training.com.br/sosim/index_v12.htm>. Acesso em: 25 jun. 2015.

2. Assista ao vídeo de demonstração do SOSim. Disponível em: <https://www.youtube.com/watch?v=6KUeMwXmU_A>. Acesso em: 2 jul. 2015.
3. Inicie seus estudos nessa ferramenta, criando processos e verificando as estatísticas.
4. Faça as anotações das estatísticas e investigue esses conceitos. Faça as pesquisas em grupos e compartilhem suas conclusões.



Vocabulário

Throughput: refere-se à quantidade de dados que são processados e ao tempo que levou para essa transferência acontecer. É aplicável tanto em transferências em disco rígido quanto em redes de computadores.

Sem medo de errar

Relacionar os conteúdos necessários para realizar a gerência do processador e explicar de que forma o processador trata as informações de instruções que são interrompidas e como ocorre o tratamento das exceções. Essa informação será importante para que a empresa possa tomar a decisão de compra do sistema operacional. Então, vamos à explicação de como é importante a escolha de um sistema operacional que seja multiprogramável!

Há dois tipos de tratamento de interrupção: o vetor de interrupção e um registrador de *status*. O vetor de interrupção tem como objetivo guardar o endereço em que está o conjunto de instruções que foram executadas para tratar o evento. Já o registrador de *status* armazena qual foi o tipo de evento ocorrido e, então, para cada tipo de evento, há a sua respectiva rotina de tratamento.

A seguir, estão relacionados os processos que ocorrem para tratar a interrupção. De acordo com Machado e Maia (2013), são:

1. Processador recebe sinalização de ocorrência do evento.
2. Processador encerra a execução da instrução que está efetuando no momento e interrompe o processamento das instruções daquele determinado programa.
3. Os registradores do tipo PC, ou seja, de contagem de instruções, são acionados para guardar tais instruções.
4. Processador verifica a qual rotina o evento está associado e busca no registrador

a informação para execução.

5. O tratamento de interrupções é salvo e entra na pilha de controle do programa.

6. A rotina de tratamento é executada.

7. Em seguida, as informações que foram salvas nos registradores de uso geral são restauradas, para que o processador continue a execução das instruções do programa que foi interrompido, exatamente do ponto que parou.

Esses podem ser considerados fatores fundamentais na escolha de um sistema operacional, pois não prejudicam o processamento de informações que estejam sendo executadas paralelamente em outros programas.



Atenção!

A seguir, está representado na ilustração como acontece o processo de tratamento de possíveis interrupções do sistema e de que forma ele entende e trata essa informação:

Figura 1.8 | Interrupções e exceções



Fonte: Machado e Maia (2013, p. 39)



Lembre-se

“Para cada tipo de interrupção há uma rotina de tratamento associada, para a qual o fluxo de execução deve ser desviado. A identificação do tipo de evento ocorrido é fundamental para determinar o endereço da rotina de tratamento. No momento da ocorrência de uma interrupção, o processador deve saber para qual rotina de tratamento deve ser desviado o fluxo de execução” (MACHADO; MAIA, 2013, p. 39).

Avançando na prática

Pratique mais!	
Instrução Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.	
Reentrância	
1. Competência de fundamentos de área	Conhecer a evolução dos sistemas operacionais e suas respectivas especificidades; conhecer e saber identificar os principais processos e como ocorre o compartilhamento de recursos; conhecer como se dá a gerência de processos e de armazenamento de arquivos; conhecer e saber gerenciar os dispositivos de entrada e saída.
2. Objetivos de aprendizagem	Saber fazer a manipulação das informações do sistema operacional; ter conhecimento sobre as principais funcionalidades e como gerenciá-las, além de saber analisar a utilização de recursos e promover a sua otimização.
3. Conteúdos relacionados	Características dos sistemas operacionais multiprogramáveis.
4. Descrição da SP	De acordo com as necessidades de compartilhamento de recursos que a consultoria educacional precisará, é correto afirmar que um sistema multiprogramável facilitará e proporcionará o bom gerenciamento e compartilhamento de recursos que eles detêm para a execução de suas necessidades em termos de rotinas sistêmicas. Nesse sentido, é necessário que se compreenda como ocorre o procedimento de reentrância. Explique o que é e como acontece.
5. Resolução da SP	<p>A considerar que vários usuários podem utilizar os aplicativos simultaneamente, como seria se cada um deles tivesse de instalar o seu próprio <i>software</i> ou sistema operacional novamente?</p> <p>Haveria vários arquivos idênticos em um mesmo computador ou servidor, o que torna essa possibilidade completamente inviável.</p> <p>Então, a técnica de reentrância permite que um programa seja compartilhado entre vários usuários com uma única cópia salva em memória, como no caso dos editores de texto, por exemplo.</p> <p>Ou, ainda, caso alguma sub-rotina do sistema operacional seja executada por vários usuários compartilhando recursos de forma concorrente, e isso acontecendo de forma segura com a identificação de solicitação do serviço realizado. Também, infere em trabalhar com vários níveis de prioridade de execução.</p>



Lembre-se

“Reentrância é a capacidade de um código executável (código reentrante) ser compartilhado por diversos usuários, exigindo que apenas uma cópia

do programa esteja na memória. A reentrância permite que cada usuário possa estar em um ponto diferente do código reentrante, manipulando dados próprios, exclusivo de cada usuário" (MACHADO; MAIA, 2013, p. 45).



Faça você mesmo

Observe no setor de xerox o uso dos aplicativos e *softwares*. A impressora terá de imprimir diversas solicitações de forma concorrente, sem prejudicar a alocação de memória e processamento da(s) máquina(s) a que está interligada.

Faça valer a pena!

1. Explique o funcionamento do controlador por interrupção.
2. Assinale a alternativa que traz o mecanismo de funcionamento do controlador por programa:
 - a) Realiza a contagem de instruções.
 - b) Faz a transmissão dos dados dos dispositivos de entrada e saída para a memória principal.
 - c) Os dados são processados e ao mesmo tempo realiza a essa transferência destes para o registrador.
 - d) É a capacidade de um código executável ser compartilhado por diversos usuários.
 - e) Processador aguarda e testa o estado dos dispositivos até terminar a operação de E/S.
3. Explique como funciona o mecanismo de interrupção por modo assíncrono.
4. Assinale a alternativa correspondente ao mecanismo de exceção síncrona:
 - a) A reentrância permite que cada usuário possa estar em um ponto diferente do código reentrante, manipulando dados próprios, exclusivos de cada usuário.
 - b) É diretamente ligada ao programa, ou seja, é um evento ocorrido em função do processamento do programa e, por isso, também, síncrona.

- c) Permite a leitura e gravação de um registro.
- d) Realiza o transporte de dados dos dispositivos de E/S para a memória.
- e) Faz o processamento de informações do *buffer*.

5. Das afirmações a seguir, qual(ais) delas traz(em) o conceito do modo de funcionamento da técnica de acesso direto à memória DMA?

I - Permite a transferência de dados diretamente da memória principal para os dispositivos de E/S e vice-versa, sem que o processador participe desta operação.

II - A área da memória que é utilizada para realizar essa operação é chamada de *buffer* de entrada e saída.

III - A função do processador, nesse caso, é apenas a de informar ao controlador a localização, qual o dispositivo de E/S e a posição inicial da memória que deverá receber os dados de leitura, a sua gravação ou mesmo o tamanho do bloco de dados.

- a) Apenas I.
- b) Apenas II.
- c) I, II e III.
- d) Apenas III.
- e) I e III.

6. Associe na tabela o conceito à sua respectiva definição:

Conceito	Definição
a) Buffering	() Permite que um programa seja compartilhado entre vários usuários com uma única cópia salva em memória.
b) Spooling	() Responsável por fazer a transmissão dos dados dos dispositivos de entrada e saída para a memória principal, a partir do uso de registradores
c) Reentrância	() Base para o processamento batch aumenta a possibilidade de trabalho com processos concorrentes.

7. Complete a frase com as palavras da alternativa correta:

"Há dois tipos de _____: o _____ e um

_____. O _____ tem como objetivo guardar o endereço em que está o conjunto de instruções que foram executadas para tratar o evento. Já o _____ armazena qual foi o tipo de evento ocorrido e, então, para cada tipo de evento há a sua respectiva rotina de tratamento.”

- a) Vetor de interrupção/ registrador de *status*/ vetor de interrupção/ registrador de *status*/ tratamento de interrupção.
- b) Tratamento de interrupção/ registrador de *status*/ vetor de interrupção/ registrador de *status*/ vetor de interrupção.
- c) Tratamento de interrupção/ registrador de *status*/ vetor de interrupção/ vetor de interrupção/ registrador de *status*.
- d) Registrador de *status*/ tratamento de interrupção/ vetor de interrupção/ vetor de interrupção/ registrador de *status*.
- e) Tratamento de interrupção/ vetor de interrupção/ registrador de *status*/ vetor de interrupção/ registrador de *status*.

Seção 1.4

Exemplos de sistemas operacionais: Unix e Windows

Diálogo aberto

Olá, aluno! Bem-vindo a mais esta seção de autoestudos! Você, agora, é convidado a conhecer a evolução de dois grandes sistemas operacionais multiprogramáveis: o Unix e o Windows. Para tal, você já deve ter compreendido quais são os tipos de sistemas operacionais e, ainda, saber identificar as características dos sistemas multiprogramáveis.

Retomando a situação real, em que precisamos oferecer uma proposta em sistema operacional que atenda às necessidades de uso de sistemas computadorizados para a consultoria, vamos, a partir de agora, apresentar os modelos UNIX e Windows e as suas principais diferenças.

Iniciaremos a seção de conteúdos com a apresentação do Windows, seu histórico, características, como realiza o gerenciamento de recursos, controla processos e *threads* e também uma breve descrição de seu sistema de arquivos, conceitos esses que serão melhor estudados na próxima unidade de ensino. Essas informações também serão apresentadas para a definição e descrição do Unix. Dessa forma, você poderá compreender as semelhanças e diferenças entre eles.

Em função da complexidade desses sistemas e sua importância para as rotinas profissionais e pessoais, vamos compreender, também, de que forma eles colaboraram para com o surgimento de outros sistemas operacionais. Então, a intenção aqui não é de destacar apenas esses dois sistemas, e sim mostrar como esses têm participação importante no desenvolvimento dos sistemas operacionais e como são norteadores de sua evolução. Não obstante aos dois modelos apresentados, porém trabalhado em outro contexto de desenvolvimento tecnológico e de inovações, o iOS da Apple também merece destaque. Acesse o *link* a seguir para saber mais sobre a evolução do iOS (9): <<http://www.tecmundo.com.br/apple/81232-resumo-conferencia-apple-wwdc-2015-confira-destaques-video.htm>> (Acesso em: 20 jun. 2015). Aproveito para sugerir que você conheça como estão situados no mercado *mobile* os sucessores desses sistemas operacionais antes dedicados apenas aos serviços para *desktops*, e,

a considerar o caráter de adaptação e inovação que incorporam, é importante que conheça e investigue mais sobre a sua evolução e as tendências. Segue uma tabela que representa o mercado de sistemas operacionais mobile em plena expansão:

Tabela 1.2 | Fatia de mercado dos sistemas operacionais mobile

Milhões	2009	2010	2011	2012	2013	2014	ΔAno	Market Share
Android	12,0	69,6	243,5	500,1	794,0	1059,4	33,4%	81,4%
iPhone OS	20,3	46,8	93,1	135,9	153,7	192,2	25,0%	14,8%
W.Phone	14,7	12,2	9,0	17,5	34,1	35,4	3,9%	2,7%
Blackberry	34,5	47,5	51,1	32,5	19,4	6,1	(68,4%)	0,5%
Symbian	80,0	109,9	81,5	23,9	1,7	-	-	-
linux	6,4	5,2	14,5	13,0	3,9	-	-	-
Other OSs	3,4	5,7	1,8	2,4	12,6	8,1	(36,1%)	0,6%
TOTAL	172,3	296,9	494,5	725,3	1.019,4	1.301	27,6%	100,0%

Fonte: IDC.
Nota: Android - Google; iPhone OS - Apple; Blackberry - RIM;
Symbian - Nokia; W.Phone - Microsoft.

Desde já, recomendamos a você que explore as funcionalidades dos sistemas operacionais modernos e saiba como foi possível chegar a tal possibilidade. Então, bons estudos!

Não pode faltar

Vamos acompanhar, no Quadro 1.15, a evolução dos sistemas operacionais Windows e Unix. Em seguida, serão descritas algumas especificidades de cada um:

Quadro 1.15 | Perspectiva de evolução dos sistemas operacionais Windows e Unix I

Ano/ Versão Windows	Descrição	Ano/ Versão Unix	Descrição
1981 – MS-DOS	16 bits, monoprogramável, monousuário, interface de linha de comando. Desenvolvido com base nos sistemas operacionais CP-M e Unix.	1969 – UNICS (Uniplexed Information and Computing Service). 1969 – UNIX	Criado para facilitar o trabalho dos desenvolvedores de software. Chamado primeiramente de UNICS e depois de UNIX. Desenvolvido em Assembly. Criado e liberado pela Bell Labs, subsidiária da AT&T.
1985 – MS Windows	interface gráfica. MS DOS como núcleo do sistema operacional, bem como as versões 3.0, 95, 98 e ME.	1973	Desenvolvido em C.

1988 – Windows NT (New Technology)	Projeto conduzido por David Cutler, que participou do desenvolvimento de outros S. O como: PDP/ RSX, VAX/ VMS, OS2 e LAN Manager.	1974 1980 1982 – SVR4	<p>Foi adotado como plataforma padrão pelas universidades. A universidade de Berkeley desenvolveu algumas versões próprias com melhoramentos, como: memória virtual, shell, Fast File System, sockets e compatibilidade aos protocolos TCP/IP.</p> <p>Berkeley lançou a versão FreeBSD.</p> <p>AT&T iniciou a comercialização do Unix. A versão SVR4 é a de maior importância. Liberou um conjunto de especificações – System V Interface Definition.</p>
1990 – Windows 3.0	Interface gráfica incorporada ao MS-DOS. Processador Intel 80386, 16 cores e o conceito de Gerenciador de Programas e Painel de Controle.	1990 1991	<p>O IEEE (Institute of Electrical and Electronics Engineers) com o comitê POSIX (Portable Operating System Unix), Surge o padrão IEEE 1003.1, que estabelece um conjunto de bibliotecas padrão para UNIX.</p> <p>Linus Torvalds inicia o desenvolvimento do Linux, com base no Minix de Andrew Tanenbaum da Universidade de Vrije-Holanda. Minix. Ele não segue as especificações do Unix e, por esse motivo, é considerado apenas semelhante. É desenvolvido em C e Assembly, tendo evoluído a partir da colaboração de vários desenvolvedores.</p>
1993 – Windows NT Desktop e para Servidores	32 bits, multitarefa preemptiva, multithread, memória virtual, multiprocessadores simétricos. Não tem MS-DOS como núcleo do S.O, mas ainda é compatível. Interface gráfica.	1993	Unix é vendido para a Novell, que lança o UnixWare. Novell transfere os direitos da marca Unix para o consórcio X/ OPEN.
1995- Windows 95	Incorporou o Menu Iniciar e a Barra de Tarefas e opções de Enviar para com interligação direta ao Outlook.	1995- UNIX95	X/OPEN lança o UNIX95 e cria uma especificação única para UNIX. X/OPEN passa a se chamar The Open Group e traz a versão 3 de especificações UNIX, chamada de POSIX.

1998- Windows 98	Novidade foi a interligação com a internet com a versão 4 do Internet Explorer. Sistema de arquivos FAT- 32.	1995 – atualmente	Atualmente, esse sistema operacional é comercializado por diversos fabricantes, tais como: Sun Microsystems (SunOS e Solaris), HP (HP-UX), IBM (AIX) e ainda Compaq (Compaq Unix).
---------------------	--	----------------------	--

Fonte: Adaptado de Machado e Maia (2013, p. 2-45)

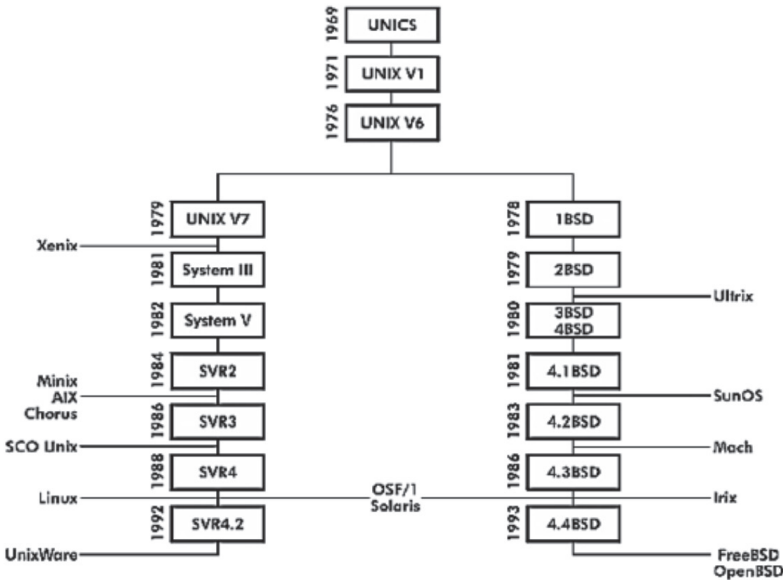


Refleta

“O Unix foi inicialmente desenvolvido em Assembly para um microcomputador PDP-7 da Digital. Para torná-lo mais fácil de ser portado para outras plataformas, Thompson desenvolveu uma linguagem de alto nível chamada B e reescreveu o código do sistema nessa nova linguagem. Em função das limitações da linguagem B, Thompson e Dennis Ritchie, também da Bell Labs, desenvolveram a linguagem C, na qual o Unix seria reescrito e, posteriormente, portado para um minicomputador PDP-11 em 1973” (MACHADO; MAIA, 2013, p. 18).

Veja a Figura 1.8, que traz as evoluções do Unix:

Figura 1.8 | Evolução do sistema operacional Unix



Fonte: Machado e Maia (2013, p. 24)



Assimile

Para Unix:

“Um processo no Unix é formado por duas estruturas de dados: a estrutura do processo (proc structure) e a área do usuário (user area ou u area). A estrutura do processo, que contém o seu contexto de software, deve ficar sempre residente na memória principal, enquanto a área do usuário pode ser retirada da memória, sendo necessária apenas quando o processo é executado” (MACHADO; MAIA, 2013, p. 24).



Pesquise mais

Este vídeo traz algumas diferenças e vantagens entre os sistemas operacionais: Windows, Mac e Linux. Disponível em: <<https://www.youtube.com/watch?v=wETMvJEoO6I>>. Acesso em: 24 jun. 2015.

Algumas dúvidas devem ter surgido, por exemplo: o que é 16, 32 ou 64 bits? O que isso significa? Refere-se à quantidade de endereços que o processador consegue registrar as informações. Então, a considerar que os processadores também tiveram de evoluir, passaram de 16 bits, ou seja, da capacidade de processamento de registros na ordem de 65.000 endereços que poderia acessar mais rapidamente para realizar a execução dele. Sendo assim, se um computador possui um processador de 32 bits, estamos falando de 232 (4.294.967.295) de endereços para que possa guardar informações para processar de forma mais rápida. Da mesma forma, se esse for de 64 bits, temos a possibilidade de guardar 264 de endereços de registros, um total de 18.446.744.073.709.551.616 endereços distintos.



Refleta

“O MS Windows possui mais de 40 milhões de linhas de código escritas em sua maioria em Linguagem C, porém com alguns módulos desenvolvidos em C++ e Assembly. O sistema é estruturado combinando o modelo de camadas e o modelo cliente-servidor. Embora não seja totalmente orientado a objetos, o MS Windows representa seus recursos internos como objetos” (MACHADO; MAIA, 2013, p. 4).

Como os computadores interpretam apenas informações binárias, é preciso saber quantificar essa possibilidade de processamento de registros. Cada processo recebe um número equivalente àquele registro, que funciona como se fosse um índice que

tem por função mostrar o que se procura e onde. A esse índice atribui-se o nome de *handle*.

Outra questão interessante para que você fique por dentro das evoluções mencionadas e saiba identificar essa melhoria: o que é multitarefa preemptiva? Em sistemas operacionais, atribui-se esse nome à ação de repartir o tempo e recursos do sistema para processar uma tarefa. Se o tempo for ultrapassado, o sistema automaticamente abortará a tarefa e iniciará, do mesmo modo, o processamento de outra. No entanto, ao retomar o processamento da tarefa abortada, irá continuar do ponto que parou o processamento daquele registro. Ou, ainda, se uma tarefa é classificada com alta prioridade, o sistema garante que nem todos os seus recursos estão voltados àquele processamento (MAYER, s.d).

Vale ainda trazer os conceitos relacionados à evolução do modo de processamento dos registros em um computador. Iniciamos com a visão de processamento de uma tarefa por vez, ou seja, em que todos os recursos da máquina ficavam dedicados à execução de uma determinada sequência de instruções ou tarefas. Como exemplo, podemos relacionar o processamento em lote (*batch*). O conceito de processamento mudou quando se tornou possível compartilhar recursos de processador e memória de forma concorrente e simultânea para executar o que passou a se chamar: processo. Com isso, foi necessário estabelecer um mecanismo de controle de estados do processo, que recebeu o nome de *thread*. Ela controla os estados do processo que deverá ser executado: criação, espera, execução, transição, pronto, *standby* e terminado. Você estudará, em seguida, com mais detalhes, esses mecanismos (MACHADO; MAIA, 2013, p. 16; 30). No entanto, há informações no livro de Machado e Maia, indicado na literatura básica desta disciplina, que detalham esses conceitos e arquitetura tanto para Windows quanto para Unix. Então, siga em frente!

Outro conceito importante que pode ser elencado com a evolução dos sistemas operacionais, processadores e meios de armazenamento é como acontece o controle e a organização dos arquivos e diretórios. Estamos falando dos sistemas de arquivos. Existem quatro tipos de sistemas de arquivos para Windows: CDFS (*CD-ROM File System*, que suporta formatos de CD e DVD), UDF (*Universal Disk Format* – CD e DVD), FAT (*File Allocation Table*), desenvolvido inicialmente para o MS-DOS e depois no Windows, com FAT16 e FAT32. Além desses, o NTFS (*NT File System*) utiliza esquema de organização de arquivos em estrutura de dados conhecida como árvore-B, e também oferece maior segurança. No Unix, não há uma definição de um tipo de sistema de arquivo especificamente porque esse trabalha de forma hierárquica nos diretórios. Então, é possível, com isso, criar vários diretórios e arquivos que, na verdade, estão distribuídos entre as máquinas que compartilham recursos remotamente, o que torna viável uma implementação de sistema de arquivos que suporte o trabalho remoto. Sendo assim, o Unix tem os seguintes sistemas de arquivos remotos: NFS (*Network File System*), RFS (*Remote File System*) e *Andrew File System* (AFS) (MACHADO; MAIA,

2013, p. 18 e 32).

Nesse contexto, cabe trazer a evolução do Windows que teve maiores modificações nos anos 2000. Veja o Quadro 1.16 que apresenta tais perspectivas:

Quadro 1.16 | Perspectiva de evolução dos sistemas operacionais Windows

Ano/ Versão Windows	Descrição
2000 – Windows 2000	Por acompanhar o legado DOS-Windows, teve o acréscimo da função plug-and-play. Adicionou também o Active Directory (A. D.), voltado à administração de redes, segurança e administração de acessos possibilitados pela centralização da administração da rede no A.D.
2001 – Windows XP	Mesma arquitetura do Windows 2000. Ajuda e sistema disponíveis em 25 idiomas. Mais estável e intuitivo. Suporte a acesso remoto. Sistema de arquivos criptografado. Projetado para processadores de 64 bits.
2003 – Windows Server	Suporte a processadores de 32 ou 64 bits. Escalonamento preemptivo. Disponível em seis versões, sendo que a diferença entre elas consiste na quantidade de processadores que suporta, capacidade de memória, quantidade de conexão em rede.
2007 – Windows Vista (Desktop e Servidores)	Destaque ao sistema de segurança. Dados de usuários ficam encriptados no disco rígido. Melhorias no Media Player, que oferecia suporte à tv, fotografias e edição de vídeo.
2009 – Windows 7	Maior atenção aos serviços de conexão em redes sem fio em função do aumento de vendas de notebooks. Ajustes automáticos de configurações de impressoras e arquivos. Modo de trabalhos em janela. Novas funcionalidades em touch.
2012 – Windows 8	Integração de funcionalidades com os dispositivos móveis. Serviços de armazenamento em nuvem.
2015 – Windows 10	Lançamento previsto para 29 de julho de 2015. Novo navegador de internet: Edge. Possibilidade de criar áreas de trabalho virtuais. Possibilita ajustes de tela com o toque. Mais aplicativos internos e o One Drive para backup e disponibilização.

Fonte: Adaptado de Machado e Maia (2013, p. 2-45). Informações adicionais em TechTudo: <<http://www.techtudo.com.br/artigos/noticia/2012/05/a-evolucao-do-windows.html>>. Acesso em: 1º jul. 2015.



Faça você mesmo

No *link* a seguir, você tem informações sobre o desenvolvimento da Microsoft e também muitas informações sobre o mesmo movimento na empresa Apple. Pesquise, leia e entenda! Acesse o **site Roughly Drafted**: <<http://www.roughlydrafted.com/2007/09/10/office-wars-3-how-microsoft-got-its-office-monopoly/>>. Acesso em: 1 jul. 2015.

Mas, afinal, o que é *Plug-and-Play*? É a função do sistema operacional de reconhecer a entrada ou instalação de um novo dispositivo e disparar imediatamente o seu reconhecimento para todas as outras funcionalidades do sistema, o que torna possível o uso, ou ainda o compartilhamento desse com outros computadores em redes ponto a ponto (*Peer-to-peer*) (MICROSOFT, 2015).

Você também identificou outro recurso importante que foi adicionado nas versões a partir de 2000, o *Active Directory*. Ele oferece um serviço de diretório que contém as características ou atributos da(s) rede(s) em que a(s) máquina(s) está(ão) interligada(s) (MICROSOFT, 2015).

Estudaremos em outra unidade o escalonamento preemptivo, mas vamos conhecer e entender como é que funciona esse mecanismo. Cada processo tem um tempo pré-determinado de execução. Então, quando esse tempo acaba, esse processo é suspenso e imediatamente é alocado outro que esteja na pilha para execução. O sistema de escalonamento preemptivo inserido nos sistemas operacionais tem por função criar uma lista de processos que já estão prontos. Trabalha com um algoritmo (*Round-Robin*, entre outros) de controle de fila que exhibe os processos prontos e libera o registro para receber outro processo a ser executado. Observe o Quadro 1.17 e veja algumas características técnicas que distinguem os dois modelos de arquiteturas e mecanismo de funcionamento dos sistemas operacionais multiprogramáveis Windows e Unix:



Exemplificando

Confira a seguir algumas informações que apresentam as principais características de cada um dos sistemas operacionais Unix e Windows na versão NT:

Quadro 1.17 | Características do Unix e Windows

Característica	Unix	Windows NT	Comentários
Sistema de Arquivo	UFS – Unix File System	NT File System (NTFS) ou File Allocation Table (FAT)	FAT é mais compatível com outros sistemas operacionais. Apresenta mais problemas de segurança e funcionalidade limitadas. NTFS é comparável ao sistema de arquivo do Unix.

Sistema de Arquivo de Rede	Network File System (NFS)	Server Message Block (SMB), Common Internet File System (CIFS)	Esse é um problema de integração. Possui utilitários e características diferentes, o que torna os sistemas incompatíveis nesse aspecto. Como solução, é sugerido o software Samba.
Senhas	/etc/passwd, /etc/shadow, NIS ou NIS+	via Registry	No NT, infos dos usuários estão no Registry. No Unix, em um simples arquivo ASCII.
Usuário Principal	root	administrator	Essa conta tem total controle sobre o sistema de arquivo.
Multitarefa	Excelente	Modesta	Ambos podem executar múltiplas tarefas ao mesmo tempo.
Multiusuário	Sim	Não	NT é originalmente monousuário, a menos que se utilizem outros softwares, como: WinFrame da Citrix.

Fonte: Figueiredo (1999)

É preciso pensar na compatibilidade entre os sistemas operacionais ao adotar ou desenvolver uma solução em *software*. Tal aspecto deve ser levado em consideração, pois esse *software* pode ter sido desenvolvido para uma versão específica.



Assimile

Para Windows

“O MS Windows oferece diversas APIs relacionadas à gerência de E/S, sendo a maior parte ligada ao subsistema gráfico, chamado Graphics Device Interface (GDI). É um conjunto de rotinas que permite a uma aplicação manipular dispositivos gráficos, como monitores e impressoras, independente do tipo do dispositivo físico, funcionando como uma interface entre a aplicação e os drivers dos dispositivos. O GDI oferece funções de gerenciamento de janelas, menus, caixas de diálogo, cores, desenhos, textos, bitmaps, ícones e clipboard” (MACHADO; MAIA, 2013, p. 16).

Sem medo de errar

Aplicação dos procedimentos de atuação convenientes à SP

A fim de apresentar as versões mais recentes dos sistemas operacionais Windows e Unix como uma possibilidade que a empresa de consultoria em educação pode utilizar, você, que já fez um levantamento prévio de suas versões desde o lançamento dele no mercado, agora terá de criar um quadro com as seguintes descrições: características técnicas/Como pode te ajudar. E Valor da Licença. Crie outro quadro, com as mesmas definições sobre o Unix, com o que há de mais novo neste sistema operacional.

Quadro 1.18 | Características Windows 10

Características técnicas	Como pode te ajudar	Valor da licença
<p>MS Windows 10 (todas as funcionalidades melhoradas)</p> <p>As especificações são:</p> <ol style="list-style-type: none"> 1. Processor: 1 gigahertz (GHz) or faster processor or SoC 2. RAM: 1 gigabyte (GB) for 32-bit or 2 GB for 64-bit 3. Hard disk space: 16 GB for 32-bit OS 20 GB for 64-bit OS 4. Graphics card: DirectX 9 or later with WDDM 1.0 driver 5. Display: 800x600 6. Compatível a telas mult-touch. 	<p>Compatível com todas as versões anteriores.</p> <p>Retorna o Menu Iniciar.</p> <p>Mais estável e seguro do que as versões anteriores.</p> <p>Sua construção e avaliação conta com a participação de uma comunidade do Windows Insider.</p> <p>Proporciona melhor experiência em navegação web.</p> <p>Capacidade de exibição em tela de até quatro aplicações utilizadas simultaneamente.</p>	<ul style="list-style-type: none"> • Atualizações free para quem tiver as respectivas licenças genuínas do Windows 7 Service Pack 1 e 8.1 atualizado; • o valor médio de lançamento está estimado em R\$350,00 a licença.

Fonte: Microsoft (2015)

Quadro 1.19 | Características versão Apache 2.0

Características técnicas	Como pode te ajudar	Valor da licença
<ol style="list-style-type: none"> 1. Apache versão 2.0 incorpora o OMI. 2. OMI (Open Management Infrastructure) <ol style="list-style-type: none"> a. Padrões CIM / WBEM DMTF. b. desenvolvido em C; c. compatível praticamente com todas as versões Unix e Linux. 	<ul style="list-style-type: none"> • Alto desempenho e portabilidade; • pilha de gerenciamento de processos compatível com outros S.O; • permite gerenciamento em nuvem; • gerenciamento de memória e dispositivos de armazenamento; • gerenciamento de servidor, dispositivos de rede e de E/S. 	<ul style="list-style-type: none"> • Free- distribuído por Apache. • Para ver a licença, acesse: http://httpd.apache.org/docs/2.2/pt-br/license.html.

Fonte: Apache.Org (2014)



Atenção!

Para maiores informações, acesse: <<https://collaboration.opengroup.org/omi/>>. (Acesso em: 30 jul. 2015). E veja outras funcionalidades deste *open source*.



Lembre-se

Para mais informações, acesse: <<http://www.apache.org/>>. Acesso em: 30 jul. 2015.

Conheça também a nova versão do AIX (IBM): <<http://www-03.ibm.com/systems/br/power/software/aix/>>. Acesso em: 30 jul. 2015.

Avançando na prática

Pratique mais!

Instrução

Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.

Exemplos de sistemas operacionais: Unix E Windows

1. Competência de fundamentos de área	Conhecer a evolução dos sistemas operacionais e suas respectivas especificidades; conhecer e saber identificar os principais processos e como ocorre o compartilhamento de recursos; conhecer como se dá a gerência de processos e de armazenamento de arquivos; conhecer e saber gerenciar os dispositivos de entrada e saída.
2. Objetivos de aprendizagem	Saber fazer a manipulação das informações do sistema operacional e ter conhecimento sobre as principais funcionalidades e como gerenciá-las, além de saber analisar a utilização de recursos e promover a sua otimização.
3. Conteúdos relacionados	Exemplos de sistemas operacionais: Unix E Windows.
4. Descrição da SP	Suponha que a consultoria queira uma explicação da estrutura do Unix, pois já conhecem o Windows e gostariam de conhecer melhor esse "novo" sistema operacional. Com base nessa necessidade apresentada, ofereça, de forma clara e concisa, as informações de que eles necessitam:

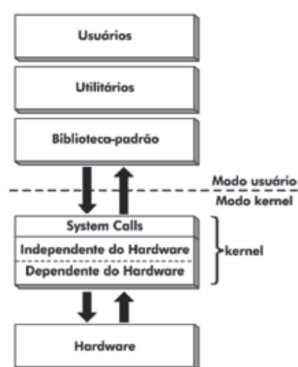
5. Resolução da SP

Kernel: tem por função controlar os recursos de hardware e as chamadas de sistema que acessam os serviços de gerência de memória virtual, processador, sistemas de arquivos e E/S. Unix trabalha com a chamada de rotinas que pertencem à biblioteca padrão.

Utilitários: representam os aplicativos (editores, planilhas etc.) acessados pelos usuários. Faz a interface do sistema com o usuário. Todos os comandos digitados são interpretados pelo shell.

Há três tipos de shell: Bourne Shell (sh está em todas as versões de Unix; CShell (csh para BSD) e Kron Shell (ksh para o System v) (MACHADO; MAIA, 2013).

Figura 1.9 | Estrutura do Unix



Fonte: Machado e Maia (2013, p. 25)



Lembre-se

No Unix:

“Os processos existentes no sistema são organizados em um vetor, chamado tabela de processos, onde cada elemento representa uma estrutura do processo. O tamanho desse vetor é pré-definido e limita o número máximo de processos no sistema. A estrutura do processo, por sua vez, possui um ponteiro para a área do usuário” (MACHADO; MAIA, 2013, p. 24).



Faça você mesmo

Conheça os principais comandos e utilitários do Unix: 1. <<http://www.inf.ufpr.br/roberto/ci064/labUnix.html>>. (Acesso em: 30 jul. 2015) 2. <<https://www.ime.usp.br/~ueda/ldoc/rb.html#metacs>>. Acesso em: 2 jul. 2015.

Faça valer a pena!

1. Assinale a alternativa que traz uma característica do Windows:

- a) Um dos distribuidores: Apache.
- b) Free: Open/Source.
- c) Menu Iniciar.
- d) Biblioteca padrão.
- e) Bourne Shell.

2. Assinale a alternativa que traz uma característica do Unix:

- a) *Software* Proprietário.
- b) Atualizações *free* para quem tiver as respectivas licenças genuínas do Windows 7 Service Pack 1 e 8.1 atualizado.
- c) Navegador Edge.
- d) Índice de controle de registros handle.
- e) Incorpora o OMI.

3. Dentre as afirmações abaixo, assinale a alternativa que apresenta a sequência verdadeira:

I - Sistema de Arquivos do Windows: NFS (*Network File System*).

II - Sistema de arquivos do Windows: FAT (32/ 64).

III - Sistema de arquivos do Unix: FAT 32.

- a) F, V, F.
- b) V, V, V.
- c) F, V, V.
- d) V, F, F.
- e) V, V, F.

4. Ao considerar a importância da administração dos recursos, tanto da máquina quanto da rede a qual está interligada, descreva qual é o objetivo do Active Director do Windows.

5. Quais são os *status* gerados por *threads* para controle de processos?

6. Associe no quadro e assinale a alternativa correspondente:

Ano/ Versão	Descrição
a) 1981 – MS-DOS	() Projeto conduzido por David Cutler, que participou do desenvolvimento de outros S. O., como: PDP/ RSX, VAX/VMS, OS2 e LAN Manager.
b) 1988 – Windows NT (New Technology)	() 16 bits, monoprogramável, monousuário, interface de linha de comando. Desenvolvido com base nos sistemas operacionais CP-M e Unix.
c) 1974 – Unix	() Foi adotado como plataforma padrão pelas universidades. A Universidade de Berkeley desenvolveu algumas versões próprias com melhoramentos como: memória virtual, shell, Fast File System, sockets e compatibilidade aos protocolos TCP/IP.

- a) a, b, c.
- b) b, a, c.
- c) c, b, a.
- d) b, c, a.
- e) a, c, b.

7. Dentre as afirmações abaixo, assinale a alternativa que apresenta a sequência correta:

I - Primeira versão do Unix foi desenvolvida em Assembly.

II - Subsistema gráfico do MS Windows: *Graphics Device Interface* (GDI).

III - No Unix, os processos são organizados em um vetor, o que limita a quantidade de processos no sistema.

- a. V, V, V.
- b. F, F, F.
- c. V, F, V.
- d. F, F, V.
- e. V, V, F.

Referências

APACHE.Org. Disponível em: <<http://httpd.apache.org/docs/2.2/pt-br/license.html>>. Acesso em: 2 jul. 2015.

FIGUEIREDO, Antônio. CCUEC, Unicamp. **Integrando o Windows NT e Unix**. 1999. Disponível em: <<http://www.ccuec.unicamp.br/revista/infotec/admsis/admsis2-1.html>>. Acesso em: 1º jul. 2015.

MAZIERO, Carlos A. **Sistemas operacionais: conceitos e mecanismos**. Universidade Tecnológica Federal do Paraná (UTFP), 2013. Disponível em: <<http://dainf.ct.utfpr.edu.br/~maziero/lib/exe/fetch.php/so:so-livro.pdf>>. Acesso em: 17 jun. 2015.

MACHADO, Francis B.; MAIA, Luiz P. **Material suplementar do livro de arquitetura de sistemas operacionais**. 5. ed. Rio de Janeiro: LTC, 2013.

MACHADO, Francis B.; MAIA, Luiz P. **Arquitetura de sistemas operacionais**. 5. ed. Rio de Janeiro: LTC, 2013.

MAYER, José Francisco M. **Bate Byte**. Uma comparação prática entre sistema Multitarefa e Multithread: Unix e NT. Disponível em: <<http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=1469>>. Acesso em: 1º jul. 2015.

MICROSOFT. **Suporte**. Disponível em: <<https://support.microsoft.com/pt-br/kb/323713/pt-br>>. Acesso em: 2 jul. 2015.

MONTEIRO, M. A. **Introdução à organização de computadores**. 4. ed. Rio de Janeiro: LTC, 2005.

STUART, Brian L. **Princípios de sistemas operacionais: projetos e aplicações**. São Paulo: Cengage Learning, 2010.

TECMUNDO. **Resumo da conferência Apple WWDC 2015**: confira os destaques [vídeo]. Disponível em: <<http://www.tecmundo.com.br/apple/81232-resumo-conferencia-apple-wwdc-2015-confira-destaques-video.htm>>. Acesso em: 20 jul. 2015.

PROCESSOS E *THREADS*

Convite ao estudo

Olá, aluno! Vamos começar a trabalhar com os conceitos e práticas relacionados a processos e *threads*. Você já notou que uma das principais funções dos sistemas operacionais é controlar o processamento de informações de modo que cada uma das etapas, desde a criação de um processo até o seu encerramento, possa ser devidamente registrada e processada, além de garantir a sua continuidade em casos de interrupção e exceções. E, então, já percebeu o quanto isso é importante para o bom funcionamento de sua máquina ou estação de trabalho? A partir de agora, você é convidado a compartilhar desse momento de estudos! Para que você possa desenvolver algumas competências básicas para o uso e trabalho com os sistemas operacionais, vamos lembrar quais são elas:

- **Competência geral:** o aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional, bem como ter conhecimento sobre como se dá o compartilhamento de recursos e a sua gerência.

- **Competências técnicas:** conhecer a evolução dos sistemas operacionais e suas respectivas especificidades; conhecer e saber identificar os principais processos e como ocorre o compartilhamento de recursos; conhecer como se dá a gerência de processos e de armazenamento de arquivos; conhecer e saber gerenciar os dispositivos de entrada e saída.

Além dessas, você também precisa lembrar os objetivos específicos desta disciplina: saber manipular as informações do sistema operacional, ter conhecimento sobre as principais funcionalidades e como gerenciá-las, além de saber analisar a utilização de recursos e promover a sua otimização. Desse

modo, fica a recomendação de estudos e leituras frequentes de seu livro didático, resolução dos exercícios propostos e o acompanhamento da webaula.

O seu desafio nesta unidade de ensino é gerenciar rotinas e processos, criar, excluir e executar comandos para o processamento de dados de forma a otimizar o sistema operacional que interage com o ERP. Se você tivesse que realizar esse serviço em uma clínica médica que acabou de adquirir um módulo de sistema integrado de gestão, como realizaria essa tarefa sem afetar as múltiplas operações em andamento? Desde já, bons estudos e práticas a você!

Seção 2.1

Introdução a processos: modelo, criação, término, hierarquia, estados, implementação e *threads*

Diálogo aberto

Vamos iniciar esta seção de autoestudos conhecendo a estrutura de um sistema operacional. Ele é carregado todas as vezes em que o computador for ligado ou reiniciado. Esse procedimento se chama ativação de sistema ou *boot* (MACHADO; MAIA, 2013).

O sistema operacional é composto basicamente por um conjunto de rotinas que conhecemos como núcleo do sistema, também chamado de kernel, que tem por função realizar o controle e tratamento de interrupções e exceções, criar e eliminar processos e *threads*, sincronizar a comunicação entre eles, bem como escalonar e controlá-los. Desse modo, também é de responsabilidade desse conjunto de rotinas gerenciar memória, sistemas de arquivos, dispositivos de E/S, permitir suporte a redes locais e distribuídas, realizar a contabilização das ações do sistema e também a sua auditoria e segurança (MACHADO; MAIA, 2013).

Para cada uma das rotinas que o sistema executará, há um mecanismo de controle de chamadas de sistema, o *system call*, que pode ser explícito ou implícito. No explícito, há uma instrução de qual chamada deverá ser executada no próprio programa, através da implementação de uma função que carrega os seus respectivos parâmetros. Já na implícita, há a inserção de um comando da linguagem de programação. O *system call* é responsável por verificar os parâmetros da solicitação e enviar a sua respectiva resposta com o estado do processo, no caso, concluído, ou se houve algum erro e precise retornar à pilha de processos. Além disso, também é preciso conhecer as linguagens de comando, pois essas são importantes ferramentas para a criação de arquivos de comandos, também chamados de *batch* ou *shell scripts*. Esses arquivos têm por função viabilizar a automatização de algumas tarefas do sistema operacional que fazem a gerência do sistema. O *shell* é responsável por interpretar esses comandos.

A arquitetura do kernel pode ser monolítica, em camadas, máquina virtual ou, ainda, ser do tipo microkernel. Confira no Quadro 2.1 cada uma das respectivas descrições:

Quadro 2.1 | Tipos de arquitetura de núcleo

Tipo de arquitetura do núcleo	Descrição
Monolítica (MS- DOS e Unix)	Módulos executados separadamente mas que compõem um único executável.
Camadas (MULTICS e OpenVMS)	Devido à complexidade do sistema, ele é dividido em níveis e suas funções só podem ser utilizadas por camadas superiores (usuário, supervisor,executivo e kernel).
Máquina Virtual (VM)	Faz o intermédio entre o hardware e o sistema operacional. Oferece todos os serviços do SO. Pode haver várias máquinas virtuais em uma única máquina.
Microkernel	Menor e mais simples. Trata serviços por processos que oferece funções específicas.

Fonte: Adaptado de Machado e Maia (2013, p. 53- 58)

Tendo em mente que você precisa definir quais são as configurações necessárias para se implementar um sistema de gestão integrado em uma clínica médica, identifique tais características e descreva como acontece essa gestão de processos no sistema operacional indicado. Dessa forma, considere as configurações técnicas do servidor e das estações de trabalho. Boas práticas!

Não pode faltar

Quando estudamos o comportamento dos processos em um sistema operacional, temos de ter em mente que este, mesmo que a máquina contenha uma única unidade central de processamento, poderá criar várias CPUs virtuais. Isso se dá porque um único processo pode gerar outros processos filhos e dividir recursos de processamento. Nesse sentido, é necessário separar o modo como tais informações serão tratadas sob o contexto do *hardware*, do *software* e também do armazenamento de informações (MACHADO; MAIA, 2013).

Figura 2.1 | Modos de tratamento de dados em processos

Hardware	Software	Armazenamento
Os dados do processo ficam armazenados nos registradores (status, PC e SP).	Há a especificação de recursos e suas limitações para que possam ser alocados os processos. Nome, PID, Owner (usr), prioridade, data/ hora de criação, tempo de processador, quotas e privilégios.	Refere-se à área de memória que será alocado o processo para que possa ser executado.

Fonte: Adaptado de Machado e Maia (2013)

Como descrito, há a separação de contexto para que seja realizado o devido processamento do sinal enviado. Desse modo, o sistema operacional se torna responsável por realizar esse controle. A possibilidade de recuperar o processo a partir do ponto de interrupção é essencial para que exista a concorrência. No momento em que cessa um deles, a UCP é imediatamente ocupada por outro que será executado, substituindo o contexto de *hardware* de um processo pelo de outro.

Em contexto de *hardware*, os dados são tratados de acordo com o seu estado de processamento e armazenado no respectivo registro responsável por armazenar aquela determinada informação.

No contexto de *software*, as informações que o sistema operacional deve controlar referem-se à quantidade de arquivos que poderão ser abertos concomitantemente e quais processos detêm prioridade de execução, tamanho do *buffer* de E/S, por exemplo. Considera a identificação do processo e de quem o criou (PID – *Process Identification* e *Owner*), quotas (limites de recursos alocados) e privilégios (pode alterar desde o *status* do processo como o de outros, se este tiver um privilégio de controle de processos de administrador de sistema, por exemplo). Então, durante a criação do processo, há a especificação dos recursos que serão necessários.

No contexto do armazenamento, cada processo possui um endereço específico na memória (MACHADO; MAIA, 2013).



Refleta

A ideia principal é que um processo constitui uma atividade. Ele possui programa, entrada, saída e um estado. Um único processador pode ser compartilhado entre os vários processos, com algum algoritmo de escalonamento usado para determinar quando parar o trabalho sobre um processo e servir outro (TANEMBAUM, 2009, p. 51).

Nos sistemas operacionais multiprogramáveis, os processos não devem receber de forma dedicada todos os recursos da máquina. Com isso, os processos são divididos em estados: execução (*running*), pronto (*ready*) e espera (*wait*). O processo está em execução enquanto é processado pela UCP, sendo que os processos revezam o tempo de processamento controlado pelo sistema operacional. Quando o processo se encontra no estado de pronto, quer dizer que o processo está aguardando para ser processado, enquanto o estado de espera acontece quando o processo aguarda um recurso para continuar o processamento ou, ainda, aguarda o tratamento de um evento para que possa prosseguir. Os processos em espera são organizados no sistema em listas encadeadas e de acordo com o tipo de evento ocorrido. Quando recebem os recursos necessários, mudam para o estado de pronto (MACHADO; MAIA, 2013).

Quando se fala em mudança de estado, é preciso saber que essa só acontecerá se tiver algum evento que interfira na execução do processo. Então, um processo pode sair do estado de pronto e entrar em execução, bem como sair do estado de execução e entrar em espera, ou, ainda, do estado de espera passar para o estado de pronto novamente, e de execução para o estado de pronto (MACHADO; MAIA, 2013).

Além das características estudadas até o momento, um processo também pode representar apenas ações do sistema operacional e por esse motivo chama-se “processo de sistema operacional”. Os serviços que o sistema operacional implementa através dos processos são: auditorias e segurança, serviços de rede, contabilização do uso de recursos e de erros, gerência de impressão, de processos em lote tipo *batch*, a temporização de processos, a comunicação entre eventos e, ainda, a interface de comandos (*shell*).

Os processos do sistema operacional que administram a comunicação entre os eventos e a sua sincronização ocorrem através do envio de sinais. Esses sinais são bits que compõem o bloco de controle de processos também conhecido pela sigla PCB (*Process Control Block*) e que podem ficar em modo de espera até que o processo seja escalonado. Se o processo for eliminado, será acionado o bit correspondente ao evento e ele será excluído apenas quando for entrar em execução, com isso conclui-se que os sinais respondem diretamente aos processos (MACHADO; MAIA, 2013).

Além desses aspectos, quando se cria (*new*) um processo, é preciso também informar o seu término (*exit*) ou encerramento. Desse modo, a partir do momento de sua criação, o sistema operacional começa a gerenciá-lo.



Assimile

Há quatro eventos principais que fazem com que processos sejam criados:

1. Início do sistema.
2. Execução de uma chamada de sistema de criação de processo por um processo em execução.
3. Uma requisição do usuário para criar um novo processo.
4. Início de uma tarefa em lote (*batch/job*) (TANEMBAUM, 2009, p. 52).

Para que o sistema operacional possa controlar todas essas informações, há uma estrutura de dado chamada bloco de controle de processo ou PCB (*Process Control Block*), que trabalha da seguinte forma: faz a leitura dos ponteiros, que justamente têm por função apontar para o endereço de memória em que se encontram os registros,

lê o estado do processo, identifica, verifica a prioridade, verifica a informação dos registradores, limites de memória e cria uma lista de arquivos que ainda estão abertos para serem executados.

Mas quando um processo pode ser encerrado? Veja, no exemplo a seguir, o modo como acontece o término de um processo:



Exemplificando

Um processo pode ser encerrado quando:

- a) há a saída normal ou voluntária do processo;
- b) há a saída por erro, que também é voluntária;
- c) ocorreu algum erro considerado fatal para a continuidade de execução do processo, neste caso involuntário;
- d) quando ocorre o cancelamento de um processo por uma solicitação de outro processo, que também é involuntário (MACHADO; MAIA, 2013).



Faça você mesmo

Assista à videoaula sobre processos e *threads* e compreenda como ocorre esse passo a passo. Disponível em: <<https://www.youtube.com/watch?v=BB0o8frWvrc>>.

Os processos são classificados em dois tipos:

- CPU-bound: ocupa mais recursos da unidade central de processamento (UCP), ou seja, passa mais tempo em execução e pronto. Facilmente encontrado em aplicações com maior quantidade de operações de cálculo.
- I/O-bound: este processo passa a maior parte do tempo em estado de espera. Encontrado em aplicações comerciais em que é necessário realizar muitas tarefas de leitura, gravação e processamento.



Pesquise mais

Assista ao vídeo de criação de processos no SOSim. Disponível em: <https://www.youtube.com/watch?v=_bMRr_oPBWg>.

Além disso, há dois canais para realizar a comunicação entre os processos. Eles são chamados de *foreground* e *background*. Processos *foreground* (primeiro plano)

são aqueles que permitem que o usuário interaja com ele, por exemplo, os de entrada e saída. Já os *background* (segundo plano) são os processos que não permitem a comunicação com o usuário durante o seu processamento (MACHADO; MAIA, 2013).



Pesquise mais

Leia o artigo e saiba mais sobre processos e *threads*. Disponível em: <<http://www.tecmundo.com.br/9669-o-que-sao-threads-em-um-processor-.htm>>.

Os processos podem ser independentes (não há vínculo com outros processos), subprocessos ou *threads*. Isso significa que há modos diferentes de implementar a concorrência, subdividindo o código em partes. Um processo (pai) pode gerar outros, que serão chamados de subprocessos (filhos), e esses compartilham quotas de recursos com o processo gerador.

Já o conceito de *thread* foi desenvolvido com o intuito de reduzir o tempo que se leva para criar um novo processo em aplicações concorrentes, bem como o uso de recursos. Isso é possível em função de um processo permitir que sejam criados ao menos um *thread*, o que o torna um processo *monothread* (processo suporta apenas um *thread*) ou *multithread* (um processo suporta a criação de vários *threads*).

Quando falamos em *thread*, quer dizer que um processo, ou os seus subprocessos, estão ocupando o mesmo endereço em memória, reduzindo o tempo de comunicação entre processos. Compartilham os contextos de *software* e de armazenamento, o que configura o contexto de hardware aplicado de maneira independente para cada processo que, por sua vez, é criado de forma que haja um *thread* correspondente a cada um. Com isso, há a otimização do processamento da informação quando controlado por *threads* (MACHADO; MAIA, 2013).



Assimile

A partir do conceito de múltiplos *threads* (*multithread*), é possível projetar e implementar aplicações concorrentes de forma eficiente, pois um processo pode ter partes diferentes do seu código, sendo executadas concorrentemente com um *overhead* menor do que utilizando múltiplos processos. Como os *threads* de um processo compartilham o mesmo espaço de endereçamento, a comunicação entre *threads* não envolve mecanismos lentos de intercomunicação entre processos, aumentando, consequentemente, o desempenho da aplicação (MACHADO; MAIA, 2013, p. 81).

Além disso, quando um processo é criado, é preciso alocar os recursos, o que consome muito tempo de processamento e, se o *thread* permite alocar no mesmo endereço de memória um processo e os seus subprocessos, isso faz com que haja a otimização desse tempo que seria gasto com a criação de outros processos e alocação de recursos para eles, que trabalharão de forma concorrente.

Há outros fatores que podem ser mencionados para se distinguir um processo de um *thread* e evidenciar a importância desses para os sistemas operacionais e o gerenciamento de processos. Dentre eles está o fato de que, para cada processo criado, além da alocação de recursos, a comunicação entre os processos é essencial para a otimização do tempo. Logo, há os seguintes mecanismos que servem para realizar o envio de sinais:

a) pipe: permite o tráfego unidirecional de informações entre processos e utiliza a estrutura de dados *array* com apenas duas posições, que indicam 0 para leitura e 1 para gravação;

b) semáforos: servem para testar e incrementar a sincronização de processos;

c) troca de mensagens: este também pode ser descrito como uma forma de sincronização e comunicação entre processos, uma vez que esses podem estar localizados em outro endereço na memória e, por esse motivo, será mais demorada a comunicação. Com isso, é possível dizer que um *thread* é uma sub-rotina de um programa que é executado de forma concorrente e assíncrona, portanto, aos processos em execução.

Essas definições de ações de *threads* devem ser estabelecidas no momento do planejamento da arquitetura do sistema operacional e são implementadas por desenvolvedores que associarão cada ação de um *thread* ao respectivo tipo de processo ou comportamento que o sistema deve ter.

Em função dessas características, *threads* também passam pelas mesmas mudanças de estados que os processos. Da mesma forma que há o bloco de controle de processos, há um bloco de controle de *threads* conhecido como TCB (*Thread Control Block*). O TCB é responsável por controlar a prioridade e o estado de execução, além de conter os bits de estado do *thread*.



Refleta

A independência entre os conceitos de processo e *thread* permite separar a unidade de alocação de recursos da unidade de escalonamento, que em ambientes *monothread* estão fortemente relacionadas. Em um ambiente *multithread*, a unidade de alocação de recursos é o processo onde todos os seus *threads* compartilham o espaço de endereçamento, descritores de arquivos e dispositivos de E/S (MACHADO; MAIA, 2013, p. 86).

Quando se implementa um *thread*, é preciso saber para qual arquitetura está sendo desenvolvida e como se dará a sua implementação. Nesse sentido, deve ser considerado que um *thread* influencia em desempenho de máquina, processos e concorrência. Desse modo, *threads* podem ser implementados em bibliotecas externas ao kernel, ou seja, no modo usuário, ou ainda, pelo próprio núcleo do sistema (modo kernel) e, também, por ambos os modos, chamado de modo híbrido.



Refleta

Talvez um dos maiores problemas na implementação de TMU (*threads* em modo usuário) seja o tratamento individual de sinais. Como o sistema reconhece apenas processos e não *threads*, os sinais enviados para um processo devem ser reconhecidos e encaminhados a cada *thread* para tratamento. No caso de recebimento de interrupções de *clock*, fundamental para a implementação do tempo compartilhado, esta limitação é crítica (MACHADO; MAIA, 2013, p. 90).

Lembre-se de que o sistema operacional, justamente por ser um *software*, precisa que todas as rotinas que executará sejam especificadas detalhadamente, pois interfere no funcionamento da máquina e até mesmo nos sistemas que interagem com ele. Nesse sentido, quando se fala de bibliotecas de rotinas externas ao núcleo, estamos falando de rotinas que deverão ser executadas quando um *thread* for criado ou mesmo um processo. Logo, tamanha a sua complexidade, a proporção de sistemas operacionais disponíveis no mercado, com relação à quantidade de aplicações e programas, sempre será muito distinta, pois um programa gerencia funções específicas de um processo de negócios. Um aplicativo também oferece funcionalidades para facilitar tarefas do cotidiano profissional. Já os sistemas operacionais precisam gerenciar os recursos e as chamadas de sistema necessárias ao bom funcionamento da máquina, *softwares* e dispositivos de entrada e saída.



Vocabulário

Overhead: excesso em tempo de processamento ou armazenamento.

Escalonador: é um serviço do sistema operacional que tem a função de determinar qual processo será liberado para execução de acordo com o seu nível de prioridade.

Sem medo de errar

Para que você possa levantar os dados de configurações que a clínica médica deve ter em seu servidor e estações de trabalho para instalar e implementar o ERP em sua

rotina de trabalho, a sugestão fica em:

- a) identificar o *software* de gestão integrada e quais são as suas configurações básicas ou padrão;
- b) verificar quais são as especificações e se são compatíveis com as estações de trabalho;
- c) levantar dados de cache, *clock* e núcleos/*threads*;
- d) identificar quais são as chamadas de sistema para o gerenciamento de um dos sistemas operacionais identificados.

Tabela 2.1 | Configurações servidor e estações de trabalho

Recursos	Servidor	Estações de trabalho
Processador	Intel® Xeon® Processor E7-8830	Intel® Core™ M vPro™
Sistema Operacional	Linux CentOS 6.5	Windows XP SP3, Seven, 8 e 8.1.
Cache	24 MB	4MB
Velocidade do Clock	2.13 GHz	900 MHz
Núcleos/ Threads	8/16	2/4

Fonte: Elaborada pelo autor (2015)

Tendo em vista que você, agora, precisa listar as chamadas de sistema, escolhemos o Linux para a sua apresentação. Observe no quadro a seguir como acontece o gerenciamento de processos neste sistema.

Quadro 2.2 | Tipos de chamadas de sistema em Linux

Chamada de sistema	Descrição
<code>pid = fork()</code>	Cria um processo filho idêntico ao pai
<code>pid = waitpid(pid, &statloc, opts)</code>	Espera o processo filho terminar
<code>s = execve(name, argv, envp)</code>	Substitui a imagem da memória de um processo
<code>exit(status)</code>	Termina a execução de um processo e retorna o status
<code>s = sigaction(sig, &act, &oldact)</code>	Define a ação a ser tomada nos sinais
<code>s = sigreturn(&context)</code>	Retorna de um sinal
<code>s = sigprocmask(how, &set, &old)</code>	Examina ou modifica a máscara do sinal
<code>s = sigpending(set)</code>	Obtém o conjunto de sinais bloqueados
<code>s = sigsuspend(sigmask)</code>	Substitui a máscara de sinal e suspende o processo
<code>s = kill(pid, sig)</code>	Envia um sinal para um processo
<code>residual = alarm(seconds)</code>	Ajusta o relógio do alarme
<code>s = pause()</code>	Suspende o chamador até o próximo sinal

Fonte: Tanenbaum (2009)



Atenção!

Acesse o site e veja mais especificações de processadores: <http://ark.intel.com/pt-br/products/53677/Intel-Xeon-Processor-E7-8830-24M-Cache-2_13-GHz-6_40-GTs-Intel-QPI>.



Lembre-se

Em um ambiente *multithread*, ou seja, com múltiplos *threads*, não existe a ideia de programas associados a processos, mas, sim, a *threads*. O processo neste modo tem pelo menos um *thread* de execução, mas pode compartilhar o seu espaço de endereçamento com inúmeros outros *threads* (MACHADO; MAIA, 2013, p. 84).

Avançando na prática

Pratique mais

Instrução

Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.

Introdução a processos: o modelo, criação, término, hierarquia, estados, implementação e threads

1. Competência de fundamentos de área	O aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional, bem como ter conhecimento sobre a sua gerência e como se dá o compartilhamento de recursos.
2. Objetivos de aprendizagem	Conhecer a evolução dos sistemas operacionais e suas respectivas especificidades; conhecer e saber identificar os principais processos e como ocorre o compartilhamento de recursos; conhecer como se dá a gerência de processos e de armazenamento de arquivos; conhecer e saber gerenciar os dispositivos de entrada e saída.
3. Conteúdos relacionados	Introdução a processos: o modelo, criação, término, hierarquia, estados, implementação e threads.
4. Descrição da SP	Considere que você precise implementar <i>threads</i> em um processo de carga automática no sistema de lançamento de notas, a fim de otimizá-lo. No entanto, além do procedimento implementado, você precisa desenvolver um programa que exibe uma mensagem de confirmação de criação do <i>threads</i> . Nesse caso, a recomendação é que você consulte o material de referência bibliográfica básica da disciplina e compreenda o processo relacionado, pois esse programa que cria <i>threads</i> será desenvolvido em uma linguagem de programação.

	<p>Veja qual foi a solução apresentada pelos autores Machado e Maia para resolver o problema. Com isso, entenda e explique o procedimento de implantação de um <i>thread</i>.</p>
5. Resolução da SP	<p>O programa a seguir cria três <i>threads</i> a partir de uma classe denominada Loop. Veja no exemplo em Java como seria essa implementação.</p> <pre>import java.util.*; public class CriaThreads{ public static void main (String [] args) { int i, n = 3; for (i =1; i <= n; i++){ Loop loop = new Loop (i); Thread t = new Thread (loop) ; t.start (); System.out.println("Thread " + i + "criado"); } } } Class Loop implemente Runnable { int j; public Loop (int i) { j = i; } public void run (){ Random random = new Random(); while (true) { System.out.println("Thread " + j + "executado"); try { Thread.sleep (random.nextInt (5000)); } catch (InterruptedException iex){} } } } (MACHADO; MAIA, 2013, p. 88)</pre>



Lembre-se

A utilização de processos independentes e subprocessos permite dividir uma aplicação em partes que podem trabalhar de forma concorrente. Um exemplo do uso de concorrência pode ser encontrado nas aplicações com interface gráfica, como em um *software* de gerenciamento de *e-mails*. Neste ambiente um usuário pode estar lendo suas mensagens antigas, ao mesmo tempo que pode enviar e receber novas mensagens (MACHADO; MAIA, 2013, p. 83).



Faça você mesmo

Elabore um relatório com o que compreendeu do mecanismo de um

thread. Diferencie *Job*, *Processo* e *Thread*. Em uma atividade em equipes, discutam sobre o objeto implantado, o programa CriaThreads e tente compreender esse processo que considera tantos aspectos fundamentais de eficiência de sua máquina.

Faça valer a pena!

1. Leias as afirmações e assinale a alternativa correspondente:

I – O *system call*, que pode ser explícito ou implícito.

II – No explícito, há uma instrução de qual chamada deverá ser executada no próprio programa.

III – No implícito, a instrução da chamada que deverá ser executada está apenas no kernel.

a) V-V-V.

b) F-F-F.

c) V-V-F.

d) V-F-V.

e) F-V-V.

2. Assinale a alternativa que contém a definição de arquitetura de Kernel Monolítica (MS- DOS e Unix):

a) São processos independentes e subprocessos.

b) Serve para ler mensagens antigas dos processos.

c) Realiza carga automática no sistema.

d) Refere-se a módulos executados separadamente, mas que compõem um único executável.

e) É um serviço de comunicação do sistema operacional com o usuário diretamente.

3. Complete as lacunas da frase com as palavras disponíveis na alternativa correspondente:

"No _____, as informações que o sistema operacional deve

controlar referem-se à quantidade de arquivos que poderão ser abertos concomitantemente, quais processos detêm prioridade de execução, tamanho do buffer de E/S, por exemplo”.

- a) Contexto de *hardware*.
- b) Contexto de armazenamento.
- c) Contexto de *software*.
- d) Modo kernel.
- e) Modo híbrido.

4. Cite e descreva os estados de processos.

5. Descreva ambiente *monothread* e *multithread*.

6. De acordo com as afirmações a seguir, assinale a alternativa que representa os respectivos conceitos de *threads*:

I – *Threads* podem ser implementados em bibliotecas externas ao kernel.

II – *Threads* podem ser implementados pelo próprio núcleo do sistema.

III – *Threads* podem ser implementados por ambos modos.

- a) Modo usuário, modo kernel, modo híbrido.
- b) Modo kernel, modo usuário, modo híbrido.
- c) Modo kernel, modo híbrido, modo usuário.
- d) Modo híbrido, modo kernel, modo usuário.
- e) Modo híbrido, modo usuário, modo kernel.

7. Dada a tabela a seguir de arquitetura de *threads*, assinale a alternativa que contém os seus respectivos modelos (modos de *threads*), de forma a completar a lacuna da tabela:

Ambientes	Arquitetura
Distributed Computing Environment (DCE)	
Microsoft Windows 2000	
Sun Solaris versão 2	

- a) Modo usuário, modo kernel, modo híbrido.
- b) Modo kernel, modo híbrido, modo usuário.
- c) Modo híbrido, modo usuário, modo kernel.
- d) Modo usuário, modo híbrido, modo kernel.
- e) Modo híbrido, modo kernel, modo usuário.

Seção 2.2

Comunicação entre processos e problemas clássicos de comunicação entre processos

Diálogo aberto

O sistema operacional se comunica com o usuário de três formas: através de procedimentos próprios do sistema, por meio da interação com os aplicativos ou, ainda, através das linguagens de comando. Cada um deles tem o seu respectivo acesso e armazenamento de dados reservado em memória e, se um arquivo for compartilhado, por exemplo, será preciso garantir a veracidade e precisão dessas informações.

Por esse motivo, o acesso às informações deve estabelecer qual é o tipo de comunicação que está acontecendo e se é em modo usuário ou em modo kernel. Para identificar qual deles deverá ser acionado, o SO recebe o *status* daquela situação, que é definido por uma sequência de bits de identificação (ID) no registrador responsável por essa operação. Quando falamos que um processo está acontecendo em modo usuário, isso quer dizer que apenas instruções chamadas não privilegiadas poderão ser executadas e, por isso, uma quantidade menor de instruções a executar. Já quando se trata de um processo que será executado no modo kernel, o sistema operacional tem acesso irrestrito às instruções do processador. Entenda que informações não privilegiadas são aquelas que não oferecem risco ao sistema e, privilegiadas, referem-se às instruções que podem interferir no funcionamento do kernel (MACHADO; MAIA, 2013).

Uma das funcionalidades do sistema de gestão integrada que será implementado para a clínica médica mencionada na seção de autoestudos 2.1 deverá trabalhar com um princípio muito comum em sincronização de processos, que é fundamentado no algoritmo de Dijkstra, que estudaremos a seguir.

Desse modo, a comunicação que será estabelecida com o sistema operacional será por meio da aplicação. Nesse caso, você precisa considerar que a sincronização dos processos é baseada no princípio proposto por Dijkstra, e explicá-lo passo a passo. Com isso, o seu cliente poderá identificar de que forma o sistema poderá alternar entre as mais variadas tarefas que deverá gerenciar. Para facilitar a compreensão, é usado o problema do filósofo, e faremos a respectiva associação aos processos e à sua sincronização.

Pensando nessa possibilidade, uma das funcionalidades do *software* de gestão integrada será a de identificar, a partir da localização do cliente final, um consultório mais próximo de acordo com a especialidade que ele deseja e o horário mais próximo para que realize uma ligação direta com o consultório, facilitando a busca e otimizando o tempo gasto pelo cliente, que, nesse caso, é o paciente. Agora, vamos compreender como esse processo é realizado. Então, bons estudos, pesquisas e práticas!

Não pode faltar

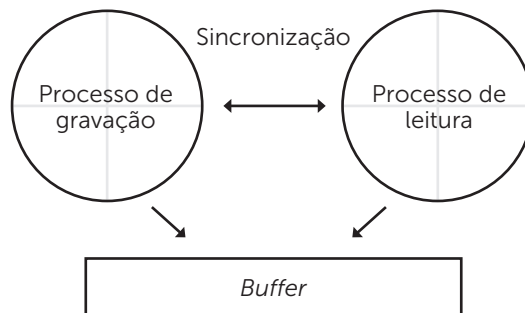
Em ambientes computacionais, os sistemas operacionais classificados como multiprogramáveis trouxeram a possibilidade de se estabelecer a concorrência durante o processamento de dados. Por esse motivo, você precisa compreender como acontece a comunicação e sincronização entre os processos que precisam ser executados. De que forma, afinal, pode ser feita? Nesse sentido, você precisa, primeiramente, compreender o que se entende por sincronização de processos e por que isso pode configurar a comunicação entre eles. O exemplo a seguir ilustra essas afirmações. Confira a seguir na Figura 2.2.



Assimile

A Figura 2.2 indica como acontece a troca de informações para operações de gravação e leitura entre processos concorrentes, em que há o compartilhamento do *buffer*, que armazenará temporariamente as informações para que sejam acessadas de forma mais rápida para processamento. A gravação ocorre apenas se o *buffer* estiver vazio, e, assim também, a leitura dos dados acontece apenas se houver dados para leitura. Nesse sentido, observe como ocorre a sincronização de leitura e gravação:

Figura 2.2 | Sincronização de leitura e gravação de processos



Fonte: Adaptado de Machado e Maia (2013, p. 94)

Para realizar a sincronização entre os processos, são acionados o que chamamos de mecanismos de sincronização. Esses visam garantir a integridade e confiabilidade das ações de sistema.

As primeiras especificações de concorrência, ou seja, as precursoras desse modelo de controle de comunicação entre processos foram desenvolvidas por Conway (1963), Dennis e Van Horn (1966). Trazem a notação dos comandos FORK e JOIN. O comando FORK tem por função realizar uma chamada do processo que está no *buffer* para ser executado e, a partir da sua identificação, o associa ao seu subprocesso, ou seja, ao processo filho. FORK também assume a função de acompanhamento de execução desse processo (MACHADO; MAIA, 2013).

Assim como FORK tem a finalidade de criar processos, o comando JOIN tem o objetivo de sincronizar os processos criados pelo FORK. Isso significa que, enquanto há um processo. "X" em execução, por exemplo, e o seu respectivo subprocesso, é possível que pelo comando FORK tenha sido criado ainda um novo processo "Z" e a alocação de recursos precisa ser gerenciada e os respectivos *status* dos processos também. Porém, quando isso acontece, o comando JOIN permitirá a execução de "X" apenas após o encerramento da execução do processo "Z" (MACHADO; MAIA, 2013).

Assim como FORK e JOIN, outro exemplo de notação de controle de concorrência e comunicação entre processos são os comandos PARBEGIN (antes chamado de COBEGIN, que tem por função criar um processo de forma aleatória) e PAREND (antes chamado de COEND, cria um novo processo apenas após o encerramento das execuções dos processos anteriores), que seguem o modelo introduzido pelo algoritmo de Dijkstra. Podem ser implementados por comandos simples de chamada a procedimentos ou ainda de atribuição.



Exemplificando

Machado e Maia (2013) trazem o exemplo de implementação dos comandos COBEGIN E COEND para calcular uma expressão aritmética. Então, observe que o exemplo traz a execução de todas as prioridades matemáticas da expressão e em seguida, faz o cálculo da expressão completa com os respectivos resultados obtidos na execução de cada uma das instruções. Suponha que a expressão aritmética seja $X := \text{SQRT}(1024) + (35.4 * 0.23) - (302/7)$. O exemplo deixa claro como ocorre a sincronização de processos com uso de especificação de concorrência com PARBEGIN e PAREND.

De acordo com as regras matemáticas e de execução, consideradas pelos sistemas operacionais, as prioridades matemáticas são preservadas,

então a lógica é a seguinte: acompanhe no algoritmo os comandos mencionados:

```
PROGRAM Expressao;

  VAR X, Temp1, Temp2, Temp3: REAL;

BEGIN

  PARBEGIN

    Temp1:= SQRT (1024);

    Temp2:= 35.4 * 0.23;

    Temp3:= 302 / 7;

  PAREND

  X:= SQRT (1024) + (35.4 * 0.23) – (302/7)

  Writeln ('x = ', X);

END.
```

Fonte: Machado e Maia (2013, p. 95).

Com os sistemas operacionais multiprogramáveis, trabalhar com processos concorrentes pode apresentar alguns problemas quando se trata do compartilhamento de recursos. Os principais mencionados por Machado e Maia (2013) estão correlacionados com o compartilhamento de um arquivo em disco e o compartilhamento de uma variável na memória principal entre dois processos. Para corrigir esses erros, são inseridos mecanismos de controle que possibilitam minimizar problemas de execução de processos concorrentes, estabelecendo condições de corrida, também conhecidas como *race conditions*.

Para tratar os erros, são propostos alguns algoritmos que reduzem a sua probabilidade de ocorrência. Dentre eles, podem ser citados:

a) Exclusão mútua (*mutual exclusion*): esse mecanismo impede que dois ou mais processos sejam executados compartilhando o mesmo recurso simultaneamente. Sendo assim, os processos precisam esperar o encerramento da execução para que possam utilizá-lo. Esse método evita que outro processo acesse a região crítica do programa; com isso, protocolos de entrada e saída são implementados para garantir que essa verificação seja realizada. A comunicação deve acontecer de forma sincronizada para que envie o *status* de encerramento de um processo e informe quando o outro se inicia, no entanto, isso ocorrerá apenas após a confirmação de encerramento do processo anterior.

Quando se trata de exclusão mútua, essa pode ser implementada com mecanismos de comunicação com o *hardware* ou com o *software*. Em soluções de *hardware*, podemos citar as de desabilitação de interrupções, em que todas são impedidas de entrar em execução ao solicitar a entrada na região crítica do programa e o *test-and-set*. No entanto, um problema que pode ser elencado quanto à desabilitação de interrupções é o risco de não acontecer a habilitação da instrução de interrupção após a mudança de *status* do processo (MACHADO; MAIA, 2013) e ele permanecer em espera ou não sair do modo de execução.



Refleta

Em sistemas com múltiplos processadores, essa solução torna-se ineficiente devido ao tempo de programação quando um processador sinaliza aos demais que as interrupções devem ser habilitadas ou desabilitadas. Outra consideração é que o mecanismo de clock do sistema é implementado através de interrupções, devendo esta solução ser utilizada com bastante critério (MACHADO; MAIA, 2013, p. 99).

Além dessa, a outra solução para o erro oriundo da exclusão mútua é a implantação da instrução *test-and-set*, que é basicamente uma instrução de máquina que trata uma exceção. Ela faz a leitura da variável, armazena o seu conteúdo em outra área e atribui um novo valor à variável então vazia. Não há interrupção durante a execução.

Sob o ponto de vista de comunicação com o *software*, são apresentados como solução quatro algoritmos de controle para os problemas relacionados aos processos de exclusão mútua (MACHADO; MAIA, 2013). São eles:

a) primeiro algoritmo: os processos alternam a execução na região crítica do programa através de uma repetição infinita, ou seja, acessa o recurso diversas vezes a fim de verificar e, se houver solicitação de execução, altera o *status* e termina o processo alternado de forma independente. Os dois processos utilizam a mesma variável global que indicam se este poderá acessar a região crítica ou não (MACHADO; MAIA, 2013);

b) segundo algoritmo: vem corrigir o problema do primeiro algoritmo que utiliza a mesma variável global e insere uma variável para cada processo criado. Cada variável é responsável por informar se o processo está ou não na região crítica. Um ponto de atenção nesse algoritmo é que ele não trabalha exclusivamente com a alternância dos processos, mas pode bloqueá-lo por tempo indeterminado, o que não garante a exclusão mútua e acarreta outro erro;

c) terceiro algoritmo: vem corrigir o problema apresentado no segundo algoritmo inserindo as instruções de atribuição de valores às variáveis antes do bloco de repetição

de verificação de disponibilidade de alocação de recurso, o que, então, permite garantir a exclusão mútua. No entanto, se dois processos em execução alteram o conteúdo das variáveis que foram criadas (segundo algoritmo), antes de iniciar execução, corre-se o risco de não acontecer o acesso à região crítica do programa, pois o kernel pode interpretar que o recurso já foi alocado e não há a execução dos processos. Os processos são executados de forma independente (MACHADO; MAIA, 2013);

d) quarto algoritmo: vem sanar o problema de comunicação de estados entre os processos, alterando o *status* da variável antes de acessar a região crítica, então existe também a possibilidade de o *status* das variáveis alterar para falso e os processos não entrarem em execução.

Para que você possa visualizar exemplos desses algoritmos, acesse o material disponível em: <<http://www.inf.ufrgs.br/~johann/sisop2/aula04.algorithms.2.pdf>>.

Observe que, mesmo com os algoritmos de correção que foram apresentados, ainda existe a possibilidade de ocorrer um erro de comunicação, ou seja, de identificação do *status*, seja em função do tipo de variável envolvida ou pela desabilitação das interrupções. Então, alguns outros cientistas e pesquisadores desenvolveram algoritmos para sanar essas lacunas e garantir que aconteça a exclusão mútua de processos sem gerar os erros mencionados. Dentre eles, podem ser citados o algoritmo de Dekker, em 1990, e Peterson G. L. (que definiram um algoritmo para tratar da exclusão mútua entre dois processos), que apresenta a possibilidade de exclusão mútua entre N processos. O algoritmo de Peterson resolve o problema da quantidade de vezes que o comando terá de repetir (indefinidamente ou *busy wait*, espera ocupada), pois insere a verificação antes de iniciar efetivamente a alternância dos processos de forma a garantir que não aconteça o bloqueio indefinido do processo (MACHADO; MAIA, 2013).

Mas, além da exclusão mútua, existem outros mecanismos. Confira a seguir os demais e suas respectivas descrições:

a) sincronização condicional: como o próprio nome diz, implementa a sincronização de execução dos processos associada a uma verificação condicional de acesso à região crítica. O exemplo mencionado de leitura e gravação de dados no *buffer* ilustra bem esse processo, pois um somente será executado quando o outro estiver com *status* de encerrado, para, então, mesmo que compartilhem recursos de forma concorrente, sincronizá-los, evitando perdas de dados e atribuição de *status* de alocação indevido, como já mencionado em exclusão mútua;

b) semáforos: este mecanismo foi implementado por Dijkstra em 1965, permite a prática da exclusão mútua com a inserção de condição para acesso à região crítica e execução dos processos. Utiliza as instruções DOWN, originalmente P de *proberen* ou teste; UP, originalmente V de *verhogen*, que significa incremento. DOWN e UP são instruções que não permitem interrupção. UP realiza o incremento ao valor do

semáforo e DOWN tem por função deixar o processo em estado de espera quando o semáforo estiver com o valor 0. Há dois tipos de semáforos: contadores que recebem valor positivo inclusive zero (0) e exclusão mútua com semáforos, que permitem apenas valores binários, ou seja, 0 ou 1 para indicar o *status* do processo (MACHADO; MAIA, 2013).

c) monitores: propostos por Brinch Hansen em 1972. São implementados pelo compilador e, por esse motivo, são considerados estruturados. Realiza um procedimento com variáveis encapsuladas e aplica a exclusão mútua, porém, apenas um processo pode executar as instruções de um monitor por vez. Os monitores trabalham com a estrutura de fila para gerenciar as solicitações de acesso ao recurso. Pode ser chamada a execução apenas a partir do programa em que foi declarado (comportamento próprio do encapsulamento). Há um algoritmo de exclusão mútua com o uso de monitores.

d) troca de mensagens: não necessita de variáveis compartilhadas, mas estabelece um canal de comunicação em que seja possível enviar (SEND) e receber (RECEIVE) mensagens para a sincronização de execução dos processos. Há a direta e a indireta. A primeira estabelece um endereço explícito para o processo receptor ou emissor. Na indireta, as mensagens podem ser alocadas pelo processo transmissor e retiradas pelo receptor; vários processos podem estar associados.

e) *deadlock*: ocorre essa situação quando um processo está aguardando por tempo indeterminado a alocação de um recurso ou um evento que não ocorrerá em função da alocação dinâmica de recursos que trabalham com concorrência. Mas como identificar essa situação e não prever? Para que não haja *deadlock*, é preciso que aconteça simultaneamente a exclusão mútua, a espera por recursos, a não liberação de um processo que aguarda um recurso de forma concorrente com outros processos, ou seja, não preempção e, ainda, precisa ocorrer o que se chama espera circular, em que um processo deve esperar a execução de um processo terminar para que utilize aquele determinado recurso (MACHADO; MAIA, 2013). Deve haver a prevenção, a detecção e a correção do *deadlock*.



Faça você mesmo

- a) investigue como são os algoritmos das soluções de *hardware*: desabilitação de interrupções e instrução *test-and-set*;
- b) teste os quatro algoritmos de exclusão mútua;
- c) em grupos de pesquisa, organizem-se para simular os algoritmos de sincronização condicional;
- d) dê preferência para as referências bibliográficas da disciplina, conforme indicado no material.



Pesquise mais

O vídeo é uma animação sobre o funcionamento do computador e de como ocorre a comunicação entre dispositivos e, também, ilustra um pouco a importância dessa, e, ainda, a necessidade de eficiência de processamento e da organização de arquivos nesse processo. Disponível em: <<https://www.youtube.com/watch?v=jH5gOJwCSQ>>.

Veja também material explicativo dos processos FORK e JOIN sobre concorrência de processos. Disponível em: <<http://slideplayer.com.br/slide/47099/>>.

Sem medo de errar

Agora, considerando que a aplicação (*software* de gestão integrada) trabalha de forma concorrente e esse é o tipo de comunicação que se estabelece, explique de que forma o algoritmo de Dijkstra pode auxiliar na alocação de recursos para a comunicação e sincronização de processos, oriundos da aplicação.

Dessa forma, lembre-se de que a tarefa agora é localizar o endereço de solicitação de informações *on-line* do sistema de gestão da clínica médica e informar, a partir disso, os consultórios e clínicas mais próximos do usuário, de acordo com o seu plano de saúde, além de oferecer o serviço de discagem direta para a realização do agendamento. Pensando nisso, explique o mecanismo de funcionamento do algoritmo proposto. Apresente o mecanismo e materiais de apoio com as suas possibilidades de aplicação.

Sendo assim, é possível associar a necessidade de processamento das solicitações dos usuários de forma concorrente. Com isso, vamos estudar o processo de Dijkstra que implanta o uso do conceito de semáforo para a sincronização dos processos. Nesse sentido, o exemplo que se enquadra nessas condições é o do problema dos filósofos. Nós consideraremos cada ação que eles precisam tomar como sendo os processos que o sistema deverá executar, implementando uma solução que previna a ocorrência de *deadlocks*. O problema: há uma mesa com cinco pratos e cinco garfos (os recursos), em que os filósofos podem sentar, comer e pensar (processos). Quando um filósofo para de pensar e deseja comer (mudança de estado do processo), ele precisa usar mais recursos: dois garfos à direita e à esquerda. Associando às ações do programa, vamos considerar que os recursos são aqueles que poderão ser compartilhados no servidor da aplicação e gerenciados pelo sistema operacional. As solicitações de clínicas e consultórios próximos ao cliente (paciente ou usuário) representam os processos. Para cada solicitação, será preciso alocar os recursos e com isso verificar, de acordo com o algoritmo proposto, a sua disponibilidade. Vamos ver como fica a solução proposta por Dijkstra em Machado e Maia (2013, p. 111):

Considere as seguintes regras: apenas quatro filósofos podem sentar-se à mesa simultaneamente (haverá uma determinação de quantidade de usuários e o sistema deve prever esse cenário e tratar de acordo com a demanda, ser desenvolvido o procedimento considerando este critério). Um filósofo só pode usar um garfo se ele estiver disponível. Ou seja, a solicitação do usuário será executada apenas se houver recurso para alocar que esteja de fato com o *status* de disponível. Um filósofo só pode usar primeiro o garfo da direita e depois o da esquerda (é estabelecida uma regra para priorizar o acesso à região crítica do programa para que aconteça a execução do processo). Veja como é o seu algoritmo:

```
PROGRAM Filosofo_2;
```

```
VAR
```

```
Garfos: ARRAY [0..4] of Semaforo:= 1;
```

```
Lugares : Semaforo := 4;
```

```
I : INTEGER;
```

```
PROCEDURE Filosofo (I: INTEGER);
```

```
BEGIN
```

```
REPEAT
```

```
    Pensando;
```

```
    DOWN (Lugares);
```

```
    DOWN (Garfos [I]);
```

```
    DOWN (Garfos [(I + 1) MOD 5]);
```

```
Comendo;
```

```
UP (Garfos [I]);
```

```
UP (Garfos [(I + 1) MOD 5]);
```

```
UP (Lugares);
```

```
UNTIL false;
```

```
END;
```

```
BEGIN
```

```
PARBEGIN
```

```
    FOR I:= 0 TO 4 DO
```

```
        Filosofo (I);
```

```
    PAREND;
```

```
END.
```



Atenção!

Assista ao vídeo que explica a lógica do algoritmo de Dijkstra e compreenda o mecanismo envolvido nesse tipo de sincronização de processos. Disponível em: <<https://www.youtube.com/watch?v=J4TZgD1As0Q>>.



Lembre-se

O conceito de semáforos foi proposto por E. W. Dijkstra em 1965, sendo apresentado como um mecanismo de sincronização que permitia implementar, de forma simples, a exclusão mútua e a sincronização condicional entre processos. De fato, o uso de semáforos tornou-se um dos principais mecanismos utilizados em projetos de sistemas operacionais e em aplicações concorrentes (MACHADO; MAIA, 2013, p. 107).

Avançando na prática

Pratique mais	
Instrução Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.	
Comunicação entre processos e problemas clássicos de comunicação entre processos	
1. Competência de fundamentos de área	O aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional e ter conhecimento sobre como se dá o compartilhamento de recursos e a sua gerência.
2. Objetivos de aprendizagem	Conhecer a evolução dos sistemas operacionais e suas respectivas especificidades; conhecer e saber identificar os principais processos e como ocorre o compartilhamento de recursos; conhecer como se dá a gerência de processos e de armazenamento de arquivos; conhecer e saber gerenciar os dispositivos de entrada e saída.
3. Conteúdos relacionados	Comunicação entre processos e problemas clássicos de comunicação entre processos.
4. Descrição da SP	De acordo com as necessidades de execução desse mecanismo de comunicação, explique o seu funcionamento e considere este procedimento como uma opção que reduz a possibilidade de ocorrência de erros para o bom funcionamento do <i>software</i> de gestão que pode ser implementado na clínica.
5. Resolução da SP	<p>Há três formas de aplicar a comunicação de processos por troca de mensagens:</p> <p>1ª) sincronização de envio, recepção e leitura de processos. O mesmo deve acontecer para o caso do recebimento de mensagens; o processo deve aguardar até que a mensagem esteja pronta para envio (rendevouz);</p> <p>2ª) utiliza o tempo de espera do processo de transmissão e permite enviar mensagens para outros destinatários, enquanto o processo de envio não finaliza o tratamento da mensagem;</p> <p>3ª) trabalha de forma assíncrona, ou seja, considera buffers para armazenar as informações e também outras formas de controle de envio e recebimento de mensagens.</p> <p>Considere o exemplo trazido por Machado e Maia (2013) para realizar a transmissão de mensagens:</p> <pre> PROGRAM Produtor_consumidor_4; PROCEDURE Produtor; VAR Msg: Tipo_Msg; BEGIN REPEAT Produz_Mensagem (Msg); SEND (Msg); UNTIL false; END; PROCEDURE Consumidor; VAR Msg: Tipo_Msg; BEGIN </pre>

	<pre> REPEAT RECEIVE (Msg); Consume_Mensagem (Msg); UNTIL false; END; BEGIN PARBEGIN Produtor; Consumidor; PAREND; END. </pre> <p>Fonte: Machado e Maia (2013, p. 119).</p>
--	---



Lembre-se

Troca de mensagens é um mecanismo de comunicação e sincronização entre processos. O sistema operacional possui um subsistema de mensagem que suporta esse mecanismo sem que haja necessidade do uso de variáveis compartilhadas. Para que haja a comunicação entre os processos deve existir um canal de comunicação, podendo esse meio ser um *buffer* ou um *link* de uma rede de computadores (MACHADO; MAIA, 2013, p. 117).



Faça você mesmo

É recomendada, além da leitura, um resumo do material indicado que trata da comunicação entre processos por troca de mensagens e exibe as formas de implementação com *sockets* com e sem conexão. Disponível em: <<http://www.oocities.org/walterchagas/process.html>>.

Faça valer a pena!

1. Descreva como é o processo de comunicação de processos a notação FORK e JOIN.
2. Assinale a alternativa que contém uma característica da comunicação e sincronização através do procedimento de exclusão mútua:
 - a) Implementa um procedimento com variáveis encapsuladas e aplica a exclusão mútua.
 - b) Deve haver a prevenção, a detecção e a correção do *deadlock*.

- c) DOWN e UP são instruções que não permitem interrupção.
- d) Impede que dois ou mais processos sejam executados compartilhando o mesmo recurso.
- e) Implementa a sincronização de execução dos processos associada a uma verificação condicional de acesso à região crítica.

3. Complete: "[...] No mecanismo de comunicação entre processos por _____, não se necessita de variáveis compartilhadas, mas se estabelece um canal de comunicação em que seja possível enviar (_____) e receber (_____) mensagens para a sincronização de execução dos processos. Há a comunicação direta e a indireta".

- a) parbegin/ PAREND/ RECEIVE.
- b) troca de mensagens/ SEND/ RECEIVE.
- c) RECEIVE/ troca de mensagens/ SEND.
- d) END/ BEGIN/ deadlock.
- e) deadlock/ PAREND/ BEGIN.

4. Assinale V ou F:

I – Monitores: são considerados estruturados. Trabalha com a estrutura de fila para gerenciar as solicitações de acesso ao recurso.

II – *Deadlock*: ocorre essa situação quando um processo está aguardando por tempo indeterminado a alocação de um recurso ou um evento que não ocorrerá em função da alocação dinâmica de recursos.

III – Semáforo: UP realiza o incremento ao valor do semáforo e DOWN tem por função deixar o processo em estado de espera quando o semáforo estiver com o valor 0.

- a) V-F-F.
- b) F-V-F.
- c) F-F-F.
- d) V-V-V.
- e) V-F-F.

5. Avalie o procedimento a seguir e assinale a alternativa correta:

```

REPEAT
    Pensando;
    DOWN (Lugares);
    DOWN (Garfos [I]);
    DOWN (Garfos [(I + 1) MOD 5]);
    Comendo;
    UP (Garfos [I]);
    UP (Garfos [(I + 1) MOD 5]);
    UP (Lugares);
UNTIL false;

```

6. Refere-se a um teste do procedimento de comunicação através de:

- a) Monitores.
- b) Semáforos.
- c) *Deadlock*.
- d) Exclusão mútua.
- e) *Send*.

7. Analise as afirmações e assinale a alternativa que apresenta a sequência correta no que tange ao mecanismo de exclusão mútua.

I – Os processos alternam a execução na região crítica do programa através de uma repetição infinita.

II – Visa sanar o problema de comunicação de estados entre os processos, alterando o *status* da variável antes de acessar a região crítica.

III – Insere uma variável global para cada processo associado.

- a) Segundo algoritmo/primeiro algoritmo/terceiro algoritmo.
- b) Quarto algoritmo/quinto algoritmo/terceiro algoritmo.
- c) Primeiro algoritmo/quarto algoritmo/segundo algoritmo.
- d) Primeiro algoritmo/terceiro algoritmo/segundo algoritmo.
- e) Primeiro algoritmo/terceiro algoritmo/quarto algoritmo.

Seção 2.3

Introdução ao escalonamento: conceitos, tipos e escalonamento de *threads*

Diálogo aberto

Olá, aluno! Vamos iniciar os estudos em escalonamento de processos. Afinal, estamos estudando o gerenciamento de processos e *threads* e aprendemos, na seção anterior, quais são os mecanismos de comunicação e sincronização de processos que os sistemas operacionais utilizam. Agora, precisamos compreender de que forma o sistema operacional seleciona esses processos e a partir de quais critérios.

Então, nesse sentido é que vamos aprender quais são as funções básicas quando o assunto é escalonamento, ou seja, a seleção do processo que será executado a partir do momento em que ele entra em estado de pronto e precisa efetivamente ser executado.

Com isso, algumas características podem ser elencadas, pois são consideradas para realizar a gerência do processador de forma a mantê-lo em exercício, ou seja, trabalhando, a maior parte do tempo possível, enquanto a máquina está em uso.

Sendo assim, sua tarefa é comprovar a eficiência do sistema operacional para realizar o gerenciamento dos processos do sistema de gestão integrado da aplicação da clínica médica de forma a otimizar, através das configurações de escalonamento, a execução dos processos solicitados pelo sistema. Fica a recomendação de que você utilize o simulador SOsim para que consiga realizar essa atividade.

Além dessas especificações, você também poderá verificar como acontece a definição da política de escalonamento em sistemas de tempo compartilhado, por exemplo, além de como acontecem os escalonamentos preemptivos e não preemptivos (MACHADO; MAIA, 2013).

Outras formas de escalonar também serão evidenciadas aqui em seu livro didático, tais como: escalonamento por filas, ou seja, em que o primeiro processo que entra na fila será também o primeiro alocado para processamento, o conhecido FIFO (*first in first out*).

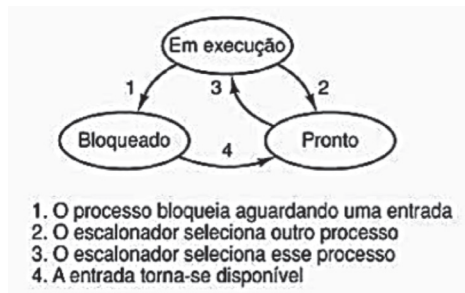
Para tal, fica a sugestão de leitura e estudos de seu material didático, que lhe proporcionará o contato necessário para que desenvolver as competências e habilidades necessárias e aplicar em sua rotina profissional quando necessário! Estude, investigue e pratique!

Desde já desejamos a você bons estudos e práticas!

Não pode faltar

Vamos iniciar retomando quais são os estados que um processo pode assumir. Você se recorda? Pois bem, temos basicamente três tipos de estados de processos: de execução, de pronto e de espera. Um processo fica em espera a partir de sua criação e permanece nesse estado até que esteja com todos os recursos dimensionados e devidamente alocados. Com isso, passa, então, ao estado de pronto. Com isso, ele poderá ser chamado para processamento e, até o término dessa operação, permanecerá no estado de execução. Observe a Figura 2.3, que traz os três estados que um processo pode assumir. Considere o estado “bloqueado” com o mesmo mecanismo e função do estado de “espera”, pois precisa que aconteça um evento que determine a mudança de estado para que possa entrar em execução. Essa definição de bloqueado foi mencionada por Andrew Tanenbaum (2010):

Figura 2.3 | Estados de um processo



Fonte: Tanenbaum (2010, p. 54)

O processo, quando entra em execução, permanece com esse estado até o término do processamento. Quando encerra, é liberado o acesso para outro processo, que deixará o estado de pronto e entrará em execução, e assim por diante. Mas como é possível controlar quais os processos que devem ter prioridade, ou, ainda, quais processos estão na fila para processamento?

Para que isso aconteça, os mecanismos de escalonamento de processos foram desenvolvidos. Alguns critérios de escalonamento são necessários e determinados de acordo com as características do sistema operacional. Dentre os critérios, podemos

elencar a análise de eficiência e utilização do processador. Nesse caso, o recomendado é que o nível de capacidade esteja ocupando, em média, 90% para ser considerado alto, ou seja, com bom potencial de aproveitamento do recurso. Outro elemento que estabelece critério para a definição do escalonamento é o *throughput*. Esse é um indicador que mostra quantos processos foram executados dentro de um intervalo de tempo. É uma medida diretamente proporcional, pois, quanto maior o *throughput*, maior será também a quantidade de tarefas realizadas naquele determinado período (MACHADO; MAIA, 2013).

Tempo de processador é outro critério relevante para a escolha ou determinação do tipo de escalonamento aplicado. Também descrito como Tempo de UCP, esse evidencia justamente o tempo que um processo leva para ser executado e finalizado. Além desse, há o tempo de espera que define o tempo em que um processo fica na fila dos processos em estado de pronto. Outro tempo importante a considerar é o de *turnaround*, que tem a função de apresentar o tempo total que um processo ocupa, desde a sua criação até o seu encerramento, e, por fim, o tempo de resposta (MACHADO; MAIA, 2013). Esse último apresenta o tempo que leva a partir da criação do processo para que este seja atendido pelo sistema e depende muito da velocidade atrelada aos dispositivos de entrada e saída, principalmente quando se trata de aplicações *web*. Imagine se o tempo de resposta for longo? O acesso com certeza ficará comprometido. O tempo de resposta também é importante e deve ser considerado ao estabelecer a política de escalonamento.

Quanto aos tipos de escalonamento, vamos estudar:

a) não preemptivos e preemptivos;	b) <i>first in first out</i> (FIFO);
c) <i>shortest job first</i> (SJF);	d) cooperativo;
e) circular;	f) por prioridades;
g) circular por prioridades;	h) por múltiplas filas;
i) por múltiplas filas com realimentação;	j) políticas de escalonamento em sistemas de tempo compartilhado e de tempo real.

Então, vamos iniciar as respectivas definições. Siga em frente!

A classificação adotada para definir a política de escalonamento em sistemas operacionais deve considerar se trabalhará com o modo não preemptivo ou preemptivo. O escalonamento não preemptivo foi implementado inicialmente para os sistemas tipo *batch* e foi um dos primeiros a ser utilizados em sistemas multiprogramáveis. Todos os recursos do processador ficam dedicados até que todo o processo seja finalizado ou por erro de execução, ou por tempo de processamento. Nesse caso, as instruções do processo já são desenvolvidas de forma a alterar para o estado de espera. Nesse modo não preemptivo, o sistema operacional não gera interrupções (MACHADO; MAIA, 2013).

Já o escalonamento preemptivo permite que interrupções do sistema operacional alterem o estado de execução de um processo para o de pronto, pois pode, em função de uma determinação por prioridade, alocar outro processo para execução. Isso também pode acontecer em sistemas de tempo real e em sistemas em que há o compartilhamento de processadores, balanceando os recursos da CPU entre os processos existentes.

O escalonamento do tipo FIFO (*first in first out*) ou fila considera o primeiro processo que entra em estado de pronto para ser o primeiro a entrar em estado de execução. Também conhecido como FIFO *scheduling* ou FCFS *scheduling*, agrupa os processos por ordem de chegada em estado de pronto e faz o escalonamento assim que chamados à execução, no caso, quando chegam a ser o primeiro processo da fila, e, com isso, o estado que estava em espera passa ao estado de pronto, e assim por diante. O tempo médio de execução não importa muito neste tipo de escalonamento em função da organização por fila (MACHADO; MAIA, 2013).

Mas, para melhorar as opções e utilizar um tipo de escalonamento que considera o tempo de execução de um processo e não a ordem de chegada na fila, temos a opção do SJF (escalonamento por *shortest job first*). Esse algoritmo seleciona o processo com o menor tempo de execução e este tem a prioridade, saindo do estado de pronto e passando ao de execução, e, então, o escalonamento realiza tal conferência novamente e ordena os processos de acordo com o seu tempo de processamento. Foi muito utilizado para os sistemas do tipo *batch*, pois o tempo de processamento era mensurado com base em análises estatísticas e daí, então, aplicado o algoritmo de escalonamento SJF. Mas ele também é aplicado a sistemas interativos, com a parada de processamento para a realização de algumas operações de entrada e saída e, com isso, repete essas operações até que o processo em execução se encerre efetivamente.

No entanto, quando isso acontece, o sistema operacional já não calcula mais o tempo que o processo levará para finalizar, ou seja, quanto tempo ainda de utilização da CPU precisará. Trabalha apenas com uma previsão de tempo de processamento com base no tempo da última operação de execução (MACHADO; MAIA, 2013). É considerado não preemptivo e uma das vantagens com relação a FIFO é que reduz o tempo total de processamento, o *turnaround*.



Pesquise mais

Assista ao vídeo que faz a demonstração do mecanismo de escalonamento SJF (*Shortwst-Job-Firts*). Disponível em: <<https://www.youtube.com/watch?v=OVWc4wDX1u4>>.

Baixe o simulador SOsim no *link*: <<http://www.training.com.br/sosim/>>.

Além desses escalonamentos, temos o conhecido como cooperativo. Esse escalonamento basicamente considera que um processo em execução, de forma voluntária, pode ceder o recurso para outro processo de forma cooperativa, através da verificação de uma fila de *status* dos processos em estado de pronto.

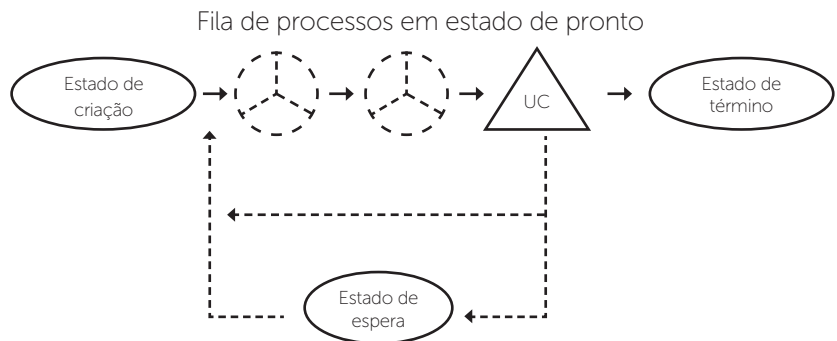
Outro importante que precisamos estudar é o escalonamento circular. Ele trabalha com fila (FIFO), e isso significa que o primeiro processo em estado de pronto será também o primeiro a ser enviado à execução, de forma a permanecer neste estado até que finde o seu processamento.



Assimile

Vamos, agora, compreender como é o mecanismo de escalonamento circular. Observe a Figura 2.4 e como acontece a alocação do processo para execução a partir do momento em que entra em estado de pronto.

Figura 2.4 | Escalonamento circular



Fonte: Adaptado de Machado e Maia (2013, p. 133)

Além dessa forma de escalonar e identificar a ocorrência, podem ser elencados alguns fatores que influenciam na mudança de estado. Por exemplo, um processo pode passar do estado de execução para o estado de espera quando excede o tempo de entrada em processamento e necessita de uma entrada preemptiva para retornar a fila. Outro fator que pode acarretar a mudança de estado no escalonamento circular é a entrada em estado de espera de forma voluntária, ou por razão não identificada (MACHADO; MAIA, 2013).

Confira, a seguir, a apresentação dos demais tipos de escalonamento existentes.

Até aqui você estudou alguns tipos de escalonamento; conheça agora o escalonamento por prioridades. Esse tipo de escalonamento precisa que seja determinada qual é a prioridade de execução de um processo e, então, o que for

operações de entrada e saída, com valores maiores do que os processos provenientes de operações de CPU (MACHADO; MAIA, 2013).

Nesse sentido, existem dois tipos de escalonamento circular: o dinâmico e o estático. As prioridades de processos no escalonamento estático são definidas no contexto de *software* e não são alteradas enquanto existir esse procedimento estabelecido no sistema. Já no escalonamento circular com prioridades dinâmicas, o administrador do sistema pode interferir manualmente nas configurações das prioridades e alterá-las de acordo com a política de escalonamento que foi estabelecida. No escalonamento dinâmico, o próprio sistema operacional pode intervir no nível de prioridade do processo para que este não demore muito tempo em execução, no caso, quando se tratam dos processos de E/S e, com isso, há um ganho em escalonamento que pode compensar a espera sem prejudicar os processos do tipo CPU-bound.

Mas existem outros escalonamentos que visam otimizar ainda mais a organização dos processos e o seu tempo de execução. Podemos citar o escalonamento por múltiplas filas, que, como o próprio nome diz, trabalha com a formação de várias filas que são tratadas de acordo com a importância da aplicação para o sistema operacional ou mesmo a quantidade e a área da memória que será alocada, ou seja, a prioridade não está associada ao processo e sim à fila de processos.



Refleta

Como processos possuem características de processamento distintas, é difícil que um único mecanismo de escalonamento seja adequado a todos. A principal vantagem de múltiplas filas é a possibilidade da convivência de mecanismos de escalonamento distintos em um mesmo sistema operacional. Cada fila possui um mecanismo próprio, permitindo que alguns processos sejam escalonados pelo mecanismo FIFO, enquanto outros pelo circular (MACHADO; MAIA, 2013, p. 137).

Além desse, ainda há um com o mesmo princípio, mas que implementa o escalonamento por múltiplas filas com realimentação, o que infere em um processo ser alternado de fila de acordo com a prioridade da fila. Esse é um controle que o próprio sistema operacional realiza para conseguir dinamizar o processamento, com base no comportamento do processo. Para que isso aconteça de forma ordenada, esse escalonamento utiliza o mecanismo de FIFO com a característica de controle por fatia de tempo, sendo assim, quanto maior a prioridade da fila, menor a fatia de tempo que o processo levará para ser encerrado, é inversamente proporcional.

Além dos tipos de escalonamento existentes nos sistemas operacionais, é preciso que se estabeleça uma política de escalonamento que pode ser para sistemas de tempo compartilhado ou de tempo real. Vamos falar agora da política de escalonamento

em sistemas de tempo compartilhado, que trabalham de modo mais interativo. Isso ocorre porque os usuários podem, através das aplicações, manipular informações do sistema, o que pode alterar o comportamento do sistema de modo geral, em função do nível de compartilhamento de recursos exigido no ambiente. Em políticas de escalonamento de sistemas de tempo real, o controle de tempo de execução do processo é mais rigoroso, pois não é permitido comprometer o processamento de modo geral, pois a atualização precisa ser constante. Um bom exemplo são os sistemas industriais voltados à produção e também de controle de tráfego aéreo (MACHADO; MAIA, 2013).



Faça você mesmo

Elabore uma tabela com uma sequência de processos; recomendo três. Defina o tempo de processamento para cada um e o seu respectivo nível de prioridade.

Desenvolva o diagrama de escalonamento dos processos por:

- prioridades;
- múltiplas filas com realimentação;
- circular;
- cooperativo.

Compreenda o mecanismo de escalonamento de todos eles e siga em frente!

Sem medo de errar

Considere que você precisa realizar um teste de eficiência do sistema operacional quando o sistema de gestão integrada está em execução, e de forma a identificar o comportamento da gerência de processos e como está acontecendo o escalonamento circular por prioridades. No SOsim você pode fazer as verificações a seguir, considerando que o tipo de escalonamento aplicado será o circular:

- a) configure no console do SOsim os parâmetros que o sistema deverá seguir;
- b) a princípio, crie quatro processos que tenham o mesmo nível de prioridade. Defina quais serão CPU-bound e quais serão I/O-bound;
- c) anote os tempos que os processos levam desde a criação até o encerramento, *turnaround*, bem como as mudanças de estados;

d) altere, também, a fatia de tempo que um processo pode levar, ou seja, configure esse parâmetro;

e) compare os tempos antes da alteração e depois, para poder dimensionar as respectivas mudanças de estados;

f) observe quais foram as variações em processamento, estados e tempo.

Tome nota dos parâmetros utilizados e também dos resultados obtidos nesse processo. Associe essa atividade às tarefas que o sistema operacional deverá controlar, com a implantação do sistema de gestão integrada de monitoramento e controle de agendamentos de consultas e exames para a rede da clínica médica.

O SOsim é um simulador de comportamento de um sistema operacional, no que tange ao gerenciamento e alocação de recursos necessários para a realização das tarefas necessárias. Desenvolvido pelo professor Luiz Paulo Maia, no Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro, com esse simulador é possível visualizar como ocorre e quais são os mecanismos de funcionamento de um sistema operacional multiprogramável. Aproveite e conheça mais sobre essa importante ferramenta em: <<http://www.training.com.br/sosim/>>.



Atenção!

Faça um relatório com as observações da simulação e compare os tempos de processamento quando você altera os parâmetros no SOsim. Veja como funcionam os mecanismos de escalonamento e aprenda mais. Pratique!

Assista, também, a um vídeo que mostra de uma forma bastante simples como você pode criar um minissistema operacional. É uma oportunidade de conhecer e ter contato com uma linguagem de programação (Visual Basic) e também de compreender como é o contexto de *software* de sistemas. Disponível em: <<https://www.youtube.com/watch?v=f6arCgdUDwc>>.



Lembre-se

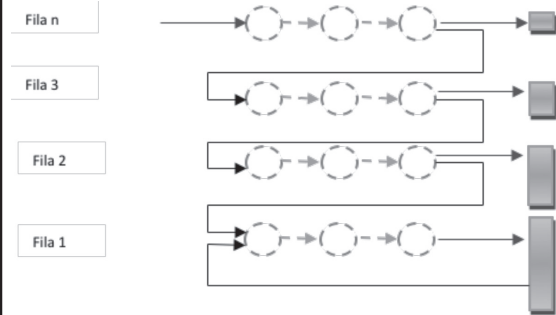
A principal vantagem deste escalonamento é permitir o melhor balanceamento no uso do processador em sistema de tempo compartilhado. Processos com o perfil I/O-bound devem receber do administrador do sistema prioridades com valores maiores que as dos processos CPU-bound. Isso permite ao sistema operacional praticar uma política compensatória entre processos de perfis distintos, compartilhando o processador de forma mais igualitária. Esse tipo de escalonamento é

amplamente utilizado em sistemas de tempo compartilhado, como o Windows e o Unix (MACHADO; MAIA, 2013, p. 136).

Avançando na prática

Pratique mais	
Instrução Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.	
Introdução ao escalonamento: conceitos, tipos, e escalonamento de threads	
1. Competência de fundamentos de área	O aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional, bem como ter conhecimento sobre como se dá o compartilhamento de recursos e a sua gerência.
2. Objetivos de aprendizagem	Conhecer a evolução dos sistemas operacionais e suas respectivas especificidades; conhecer e saber identificar os principais processos e como ocorre o compartilhamento de recursos; conhecer como se dá a gerência de processos e de armazenamento de arquivos; conhecer e saber gerenciar os dispositivos de entrada e saída.
3. Conteúdos relacionados	Introdução ao escalonamento: conceitos, tipos e escalonamento de <i>threads</i> .
4. Descrição da SP	Suponha que os parâmetros a utilizar são os do mecanismo de escalonamento por múltiplas filas com realimentação. Explique como é aplicado e a lógica deste procedimento realizado pelo sistema operacional.
5. Resolução da SP	Para compreender o escalonamento proposto, considere: a) no caso do escalonamento por múltiplas filas com realimentação, apenas a fila com menor prioridade utiliza o mecanismo de escalonamento circular. Para as demais, é o FIFO; b) o escalonamento só acontecerá em uma fila a partir do momento em que as de maior prioridade estiverem vazias; c) saiba também que, quanto maior a prioridade da fila, menor será a sua fatia de tempo; d) logo que o processo é criado, ele é direcionado para a fila de maior prioridade, em último lugar e, de acordo com a preempção por fatia de tempo, pode ser redirecionado a uma fila que tenha uma prioridade menor. Observe o esquema a seguir que representa o escalonamento por múltiplas filas com realimentação:

Figura 2.6 – Escalonamento por múltiplas filas com realimentação



Fonte: Adaptado de Machado e Maia (2013, p. 139).



Faça você mesmo

Simule esse tipo de escalonamento no SOsim. Crie ao menos três processos e defina prioridades e fatias de tempo diferentes. Aloque em filas com prioridades distintas e verifique o comportamento desse procedimento, as mudanças de estado e o tempo de processamento.



Lembre-se

O escalonamento por múltiplas filas com realimentação é um algoritmo de escalonamento generalista, podendo ser implementado em qualquer tipo de sistema operacional. Um dos problemas encontrados nesta política é que a mudança de comportamento de um processo CPU-bound para I/O-bound pode comprometer seu tempo de resposta. Outro aspecto a ser considerado é sua complexidade de implementação, ocasionando um grande *overhead* ao sistema (MACHADO; MAIA, 2013, p. 139).

Faça valer a pena!

1. Complete as lacunas da frase com as palavras da alternativa correta: "Um processo fica em _____ a partir de sua criação e permanece neste até que estejam os recursos dimensionados e devidamente alocados e ele passa, então, ao estado de _____. Com isso, ele poderá ser chamado para processamento e, até o término desta operação, permanecerá no estado de _____".

- a) espera/pronto/execução.
- b) estado/execução/pronto.
- c) espera/tempo/pronto.
- d) pronto/estado/execução.
- e) pronto/espera/criação.

2. Descreva o escalonamento por prioridade.

3. Explique qual é a diferença dos escalonamentos por múltiplas filas e por múltiplas filas com realimentação.

4. Associe na tabela e assinale a alternativa que contém a sequência obtida:

Conceito	Descrição
I – Escalonamento preemptivo	() Permite que o sistema operacional interrompa o processamento de acordo com a prioridade.
II – Escalonamento não preemptivo	() O processo, de forma voluntária, interrompe a execução, entra em espera e libera recursos para o processamento de outro processo que esteja no estado de pronto, e o escalonamento faz a respectiva verificação para a redistribuição dos processos.
III – Escalonamento colaborativo	() O sistema operacional não gera interrupções.

A sequência que representa os conceitos e suas respectivas definições é:

- a) II, I, e III.
- b) I, II e III.
- c) I, III e II.
- d) III, II e I.
- e) III, I e II.

5. Considere as afirmações:

I – Nesse tipo de escalonamento, é determinada qual é a prioridade de execução de um processo.

II – Se os valores das prioridades dos processos são iguais, então esses

serão ordenados em fila (FIFO).

III – Pode ocorrer a mudança voluntária do estado execução para o de espera, caso haja perda em eficiência do processador.

São características do escalonamento:

- a) Cooperativo.
- b) Preemptivo.
- c) Por prioridades.
- d) Por múltiplas filas.
- e) FIFO.

6. Assinale a alternativa que apresenta a sequência correta:

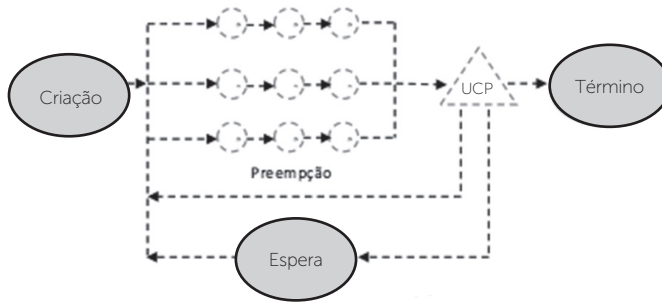
I – O escalonamento circular por prioridades só acontecerá em uma fila a partir do momento em que as de maior prioridade estiverem vazias.

II – Em um escalonamento por múltiplas filas com realimentação, quanto maior a prioridade da fila, menor será a fatia de tempo utilizada para executar os processos.

III – Em escalonamento por múltiplas filas com realimentação, logo que o processo é criado, ele é direcionado para a fila de maior prioridade, em último lugar e, de acordo com a preempção por fatia de tempo, ele pode ser redirecionado a uma fila que tenha uma prioridade menor.

- a) F-F-F.
- b) F-V-V.
- c) V-V-V.
- d) F-F-F.
- e) V-F-V.

7. Analise a figura e assinale a alternativa correta quanto ao tipo de escalonamento ilustrado:



- a) Circular.
- b) Cooperativo.
- c) Múltiplas filas.
- d) Por prioridade.
- e) FIFO.

Seção 2.4

Algoritmos de escalonamento: características, políticas, tipos e exemplos

Diálogo aberto

Olá, aluno! Bem-vindo a mais uma seção de autoestudos! Vamos conhecer outros algoritmos de escalonamento, identificar as suas principais características, bem como sintetizar a ideia de política de níveis de escalonamento, objetivos e critérios que precisam ser considerados quando se trata da otimização do processamento. Além disso, vamos conhecer também como é tratado pelo sistema operacional o escalonamento por *threads*.

Estudamos até aqui algumas especificidades quanto à definição e aos conceitos envolvidos em processos e *threads*. Nesse contexto, conseguimos conhecer como acontece a comunicação entre processos e de que forma o sistema operacional controla esse tipo de operação. Além desses, vimos também as definições de escalonamentos e quais deles são mais utilizados em sistemas multiprogramáveis.

No caso desta seção, como estudaremos outros algoritmos de escalonamento, temos de apresentar uma forma de selecionar ou avaliar um algoritmo de escalonamento que atenda às necessidades de desempenho de *software* e *hardware* que a empresa precisa, para que sejam otimizadas as rotinas do ERP e aconteça a efetiva implantação do sistema.

Mas você se recorda da realidade profissional que estamos trabalhando?

Então, vamos lembrar: devemos implantar um sistema de gestão integrada em uma clínica médica de forma a conseguir utilizar todo o potencial de seu parque tecnológico, incluindo servidores e os sistemas já instalados. Dessa forma, saber gerenciar as rotinas e entender as atividades controladas pelo sistema operacional e como ocorrem as ações que ele é responsável é de extrema importância.

Quanto mais associar os conceitos às variadas práticas e necessidades do mercado de trabalho, maiores serão as possibilidades de você aproveitar esse contato para aprender, analisar, construir e desenvolver novas formas de solucionar os possíveis problemas do cotidiano organizacional, quando se trata de ambientes computacionais.

Nesse sentido, você agora é convidado a participar de mais uma etapa desse processo de ensino-aprendizagem, associando sempre o conteúdo a uma possível situação de mercado.

Aproveite todo o potencial de estudos advindos das leituras, pesquisas, aulas e interações que o seu material didático e ambiente educacional oferecem! Desde já, bons estudos e práticas a você!

Não pode faltar

Já estudamos os algoritmos de escalonamento FIFO (*First in first out*), como sugerido por Machado e Maia (2013), ou ainda FCFS (*first come first served*) como mencionado por Silberschatz et al. (2004). O algoritmo de escalonamento FCFS considera que o primeiro processo a chegar na fila será o primeiro a entrar em execução, no entanto, este procedimento pode fazer com que processos pequenos e de custo baixo em termos de processamento esperem por processos mais longos. Ambos são de baixa complexidade.

Vimos também o escalonamento por prioridade em que o processo que tiver o maior nível será executado primeiro. Então, nesse sentido, visando reduzir o tempo de espera mesmo com níveis distintos de prioridade, foi implementado o algoritmo SJF (*shortest job first*). Além desses, vimos também o algoritmo de escalonamento circular ou RR (*round robin*), que considera a divisão do processo por fatias de tempo, também chamadas de quantum (SILBERSCHATZ et al., 2004).

Ao estudar os algoritmos utilizados para realizar o escalonamento de processos, temos de considerar fatores como níveis da política de escalonamento, bem como os objetivos e os critérios que são utilizados para essa ação. Sendo assim, as políticas de definição dos níveis de escalonamento podem ser (DEITEL, 2005):

a) de alto nível: que determina quais grupos de processos poderão trabalhar de forma concorrente, e, ainda, controla o número de processos em um espaço de tempo, ou seja, o seu grau de multiprogramação, procurando evitar o uso de todos os recursos do sistema para processamento. Com isso, alguns processos podem ter de esperar a conclusão de outros para que sejam executados;

b) de nível intermediário: em que são definidos os processos que competirão por recursos do processador. É possível interromper um processo e retomá-lo em seguida com o seu processamento para que possa garantir o bom desempenho da máquina;

c) de baixo nível: nessa política de escalonamento, normalmente há a atribuição de prioridade para o processo, com isso aumenta a probabilidade dele ser escolhido para processamento. Em uma política de escalonamento de baixo nível, há a determinação

de processos que estão ativos para que o sistema possa designar um processador quando estiver disponível.

Mas, afinal, qual é o objetivo de se estabelecer uma política de escalonamento?

Bem, de modo geral, o objetivo é potencializar o uso de recursos da máquina, porém vamos especificar essa informação. Com a realização de escalonamento, é possível maximizar o uso de recursos de forma a atender uma quantidade maior de processos e em menor tempo de execução. Minimizar os tempos de resposta também é um dos objetivos do escalonamento. Outro fator importante é trabalhar com memória e processador de forma a evitar que aconteça a interrupção por tempo indefinido de um determinado processo. Além disso, é importante que, com o escalonador, sejam impostas prioridades na escolha de processos e alocação de recursos. Reduzir a probabilidade de ocorrência de sobrecarga e, ainda, melhorar a previsão de duração do processo, ou seja, a sua previsibilidade. Este último é um conceito que determina o tempo mínimo de resposta, de espera e de execução de um processo.



Assimile

Sobrecarga frequentemente resulta em desperdício de recursos, mas uma certa porção dos recursos do sistema se investida efetivamente como sobrecarga pode melhorar muito o desempenho geral do sistema (DEITEL, 2005, p. 213).

Como mencionado, alguns critérios para se estabelecer a política de escalonamento adotada precisam ser considerados. Dentre eles, podemos elencar os critérios que consideram se o processo é orientado ao processador, orientado a operações de entrada e saída, em lote, ou se é um processo interativo. Veja a seguir as breves descrições de cada um deles segundo os critérios de comportamento de processos (DEITEL, 2005):

a) orientado a processador: critério que determina que um processo tende a utilizar todos os recursos do processador que lhe foi atribuído;

b) operações de E/S: utiliza o processador apenas para atender as requisições das operações de entrada e saída de dados e, em seguida, libera o recurso;

c) processo em lote: há a alocação de recursos de processamento sem a interação com o usuário para executar um determinado processo ou grupo de processos.

d) processo interativo: nesse critério, o usuário precisa participar com as entradas de dados. Os tempos de resposta do processo precisam ser mais rápidos.

Porém, não basta que se estabeleça a política de escalonamento de acordo com os respectivos níveis de priorização de processamento e alocação de recursos, nem que

se saiba os critérios que norteiam as ações do escalonamento se não forem de seu conhecimento os algoritmos que realizam essas tarefas, que são tão importantes para a análise de desempenho de um sistema computacional. A partir de agora, conheça também outros algoritmos de escalonamento. Siga em frente! Veja no Quadro 2.3 uma breve descrição de alguns deles:

Quadro 2.3 | Algoritmos de escalonamento

Algoritmo de escalonamento	Descrição
Escalonamento por próxima taxa de escalonamento mais próxima.	Conhecido como HRRN (highest response ratio next). Visa corrigir um erro relacionado ao escalonamento SJB/SPF mencionado acima, em que há essencialmente a priorização de processos maiores a menores, fazendo com que esses tenham de esperar mais tempo do que o previsto.
Escalonamento por menor tempo de execução- restante.	SRT (shortest remaining time), ao contrário do SJB, determina o processo que terá o menor tempo de execução, por estimativa e dá prioridade a esse.
Escalonamento por fração justa	FSS (fair share scheduling) esse algoritmo permite que, em sistemas multiusuários, os processos relacionados a um usuário sejam agrupados aos de outros usuários para processamento. Também restringe cada grupo ao seu respectivo subconjunto de recursos do sistema.
Escalonamento por prazo	Esse tipo de algoritmo necessita da disponibilização exata de recursos necessários de processamento para que o sistema operacional possa distribuir sem prejudicar as rotinas comuns e desempenho. Dessa forma, o objetivo é que se cumpra o prazo predeterminado para o processamento do conjunto descrito.
Escalonamento de tempo real	Visa utilizar os recursos de processamento de forma potencializada. Isso quer dizer que pode aplicar de acordo com a operação, um algoritmo diferente para cada um dos processos solicitados.
Escalonamento de threads Java	Esse tipo de escalonamento é realizado pelo sistema operacional e considera os níveis de suporte para threads, no entanto, não distingue se é um processo multithread ou se deve trabalhar individualmente com os processos se este for de usuário, e então, necessita de uma biblioteca padrão para escalonar. Se for de núcleo, o sistema escalona um por vez, de forma a considerar a alocação de recursos para cada um deles.

Fonte: Adaptado de Deitel (2005, p. 218- 228)

Vamos descrever melhor cada um deles?

O algoritmo HRRN trabalha de forma a considerar a relação existente entre o tempo que o processo leva para ser concluído e o seu tempo de espera. Com isso, quando

há a alocação do processo para execução, esta ação acontecerá até o encerramento das instruções, de forma ininterrupta e, por esse motivo, ele é não preemptivo. Há uma fórmula que é utilizada para o cálculo da prioridade inerente a cada processo. A fórmula é a seguinte (DEITEL, 2005):

$$\text{Prioridade} = (\text{tempo de espera} + \text{tempo de serviço}) / \text{tempo de serviço}$$

Nesse algoritmo de escalonamento, os processos menores têm preferência e há o favorecimento aos processos mais longos em função da identificação do seu tempo de espera.

O SRT, através de uma ação preemptiva, pode interromper um processamento em função da alocação temporária do recurso para que um processo menor seja eliminado da fila. Lembre-se de que essa determinação e ação do SRT são fundamentadas na estimativa de tempo de execução de um processo e a escolha se dá pelo que apresentar uma estimativa menor.



Refleta

O algoritmo SRT, teoricamente, oferece tempos de espera mínimos, mas, em certas ocasiões, devido à sobrecarga de preempção, o SPF pode se sair melhor. Por exemplo, considere um sistema no qual um processo em execução esteja quase no final e chegue um novo processo, cujo tempo estimado de serviço seja pequeno. O processo em execução pode se sair melhor? A disciplina do SRT faria a preempção, mas essa pode não ser a alternativa ótima. Uma solução é garantir que um processo em execução não possa mais sofrer preempção quando o tempo de execução restante atingir um determinado nível baixo (DEITEL, 2005, p. 219).

Muito interessante também é o algoritmo de escalonamento por fração justa, o FSS. Esse trabalha da seguinte forma: a partir da quantidade de processos e subprocessos, associados a um determinado indivíduo, que compartilha recursos da máquina com outros indivíduos, o escalonamento procura identificar o grupo de processos associados que mais necessita de recursos em função de suas operações (1), e também, o grupo de processos que pode até ser maior, porém, que o tipo de prioridade associada à atividade é menor em termos de consumo de recursos de processamento (2). Então, o sistema determinará uma quantidade de recursos para o processamento do grupo 1 que não comprometa o processamento do grupo 2, de forma literalmente mais justa, quanto à distribuição dos recursos. A razão dessa ação está entre a quantidade de recursos de processamento utilizado versus o tempo real decorrido para o processamento (DEITEL, 2005).

Outro algoritmo que elencamos é o escalonamento por prazo. Como enfatizado,

esse algoritmo necessita da requisição prévia de recursos do sistema, com a inserção exata que será necessária para que se cumpra o prazo previsto de processamento de um conjunto de processos.

Uma desvantagem desse tipo de escalonamento é que pode haver sobrecarga do uso de recursos, o que pode impactar no desempenho da máquina, principalmente no que tange à dedicação de serviços e recursos para outros processos. No entanto, esse é importante para o escalonamento de processos em tempo real. Uma das formas de trabalho deste algoritmo é alocar os processos que necessitam de menor quantidade de recursos para que tenham prioridade de execução.

Como mencionado, os algoritmos de escalonamento de processos em tempo real permitem a alocação de algoritmos distintos de acordo com a solicitação que foi realizada. Dessa forma, há algumas classificações para eles. Vejamos:

a) escalonamento de tempo real não crítico: visa garantir que processos em tempo real sejam executados, porém sem a garantia de que cumprirá alguma restrição de tempo;

b) escalonamento de tempo real crítico: ao contrário do anterior, garante o processamento antes do prazo estabelecido;

c) escalonamento de tempo real estático: nesse, as prioridades dos processos são determinadas uma única vez;

d) escalonamento de tempo real dinâmico: estes realizam o escalonamento por prioridades e isso acontece durante a execução dos processos. Pode haver a interrupção preemptiva, de acordo com as necessidades de alocação identificadas, de forma a preencher, inclusive, os espaços considerados de folga entre a seleção de processos ou de execução. Essa folga refere-se ao tempo restante ao processo para que seja finalizada a sua execução.

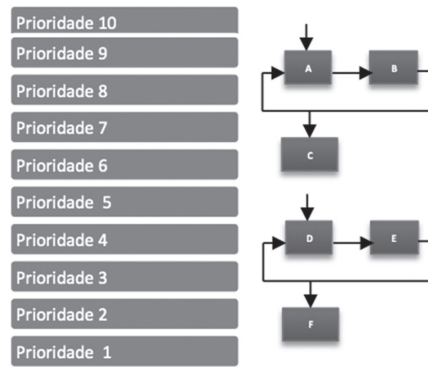
Por fim, vamos falar um pouco sobre o escalonamento de *thread* Java. No Quadro 2.3, evidenciamos que o modo de tratamento de um escalonamento de *thread* de usuário (escalonamento por *timeslicing* ou espaço de tempo) e de núcleo são distintos no sistema operacional. Outras considerações, como a quantidade de *threads*, sua ordem, bem como a prioridade estabelecida, devem ser definidas em nível de *software*, ou seja, no desenvolvimento, pelo responsável pela implementação. Nesse caso, pode ser aplicado, por exemplo, o algoritmo de escalonamento por fração justa, ou ainda, de acordo com a quantidade de *threads* alternância circular ou mesmo o algoritmo de escalonamento por intervalo de tempo. Tudo dependerá do tipo de *thread* e a quantidade de recursos necessários de forma a buscar sempre garantir um bom desempenho da máquina.



Exemplificando

Ao mencionar *threads*, precisamos considerar que na linguagem Java todo tipo de aplicação é *multithread*. Os *threads* Java recebem níveis de prioridade entre 0 e 1 (Thread.MIN_PRIORITY e Thread.MAX_PRIORITY, que são, respectivamente, comandos que definem valores constantes da prioridade, no caso, 1 no mínimo e 10 no máximo.)

Figura 2.7 | Escalonamento de threads Java



Fonte: Adaptado de Deitel (2005, p. 227)

Threads Prontos

1. Um *thread* pode ser executado até o encerramento das instruções.
2. Pode sair de execução e entrar em espera, para que outro processo de maior prioridade seja escalonado.
3. Pode sofrer preempção para que outro *thread* de maior prioridade seja executado.
4. O *thread* de maior prioridade pode executar todo o tempo até o seu encerramento ou, se for *multithread*, poderá realizar a alternância circular.



Faça você mesmo

Compreenda os mecanismos de escalonamento relacionados a *threads* em: <https://strongloop.com/strongblog/dica-da-semana-sobre-desempenho-node-js-monitoramento-de-ciclo-de-eventos/>.

Leia também matéria de suporte sobre o Windows 8. Disponível em: [https://technet.microsoft.com/pt-br/library/Cc771692\(v=WS.10\).aspx#BKMK_Scen2](https://technet.microsoft.com/pt-br/library/Cc771692(v=WS.10).aspx#BKMK_Scen2).



Pesquise mais

Leia e estude com atenção o material sobre escalonamento que está disponível em: <http://www.ece.ufrgs.br/~fetter/eng04008/sched.pdf>.

Leia mais sobre escalonamento no artigo Política de Escalonamento de Processos em Linux. Disponível em: <<http://revista.grupointegrado.br/revista/index.php/campodigital/article/view/344/159>>.

Sem medo de errar

Para essa atividade, é preciso fazer um levantamento de como avaliar um algoritmo que seja ideal para atender às necessidades tanto do sistema operacional quanto das aplicações que interagem no ambiente organizacional. Para tal, a seguir estão algumas contribuições que podem auxiliar nesse processo de seleção do algoritmo de escalonamento. Vamos lá!

A avaliação de um algoritmo de escalonamento está diretamente relacionada à sua seleção. Ele deve ser adequado às necessidades de processamento e funcionalidades que devem ser aliadas ao sistema computacional.

Desse modo, você precisa identificar quais são as necessidades de processamento do sistema. Para tal, podem ser elencados alguns pontos de atenção, como:

a) selecionar os critérios de escalonamento: orientado a processador, em lote, interativo, E/S;

b) saber qual é o tempo de resposta que a aplicação precisa (*throughput*);

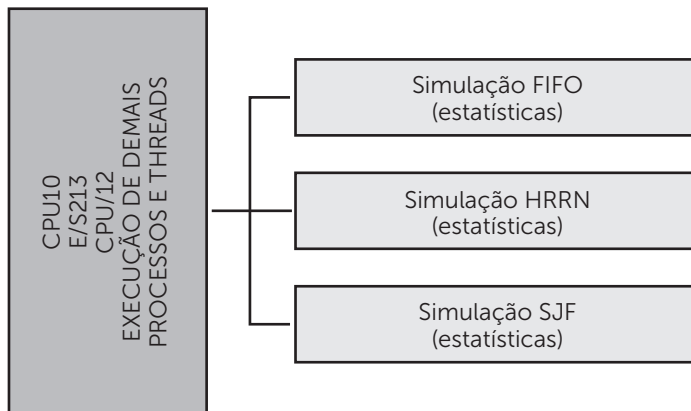
Ex.: será preciso potencializar e aumentar a utilização dos recursos de processamento da CPU com um tempo mínimo de resposta de "x";

c) ou, ainda, equalizar o processamento para que não exceda os limites de tempo de execução total e diminui tanto o *turnaround* quanto o *throughput*;

d) para tal realiza-se uma avaliação analítica que pode ser por: modelagem determinística, por enfileiramento, simulações e mesmo a implementação em nível de teste.

Veja um exemplo que dimensiona por simulação, as estatísticas de desempenho que podem auxiliar na determinação de um algoritmo de escalonamento:

Figura 2.8 | Simulações para avaliação de algoritmos de escalonamento



Fonte: Adaptado de Deitel (2005, p. 147)



Atenção!

Simulações podem ser caras e exigir muito tempo de uso de recursos do sistema computacional. Para verificar se os dados de uma simulação de fato são coerentes, é aconselhável que se codifique e avalie o comportamento do algoritmo diretamente no sistema operacional.



Lembre-se

O simulador possui uma variável representando um relógio; à medida que o valor dessa variável é aumentado, o simulador modifica o estado do sistema para refletir as atividades dos dispositivos, dos processos e do escalonador. Enquanto a simulação é executada, as estatísticas que indicam o desempenho do algoritmo são colhidas e impressas (DEITEL, 2005, p. 146).

Avançando na prática

Pratique mais

Instrução

Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.

Introdução ao escalonamento: conceitos, tipos e escalonamento de threads																				
1. Competência de fundamentos de área	O aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional, bem como ter conhecimento sobre como se dá o compartilhamento de recursos e a sua gerência.																			
2. Objetivos de aprendizagem	Conhecer a evolução dos sistemas operacionais e suas respectivas especificidades; conhecer e saber identificar os principais processos e como ocorre o compartilhamento de recursos; conhecer como se dá a gerência de processos e de armazenamento de arquivos; conhecer e saber gerenciar os dispositivos de entrada e saída.																			
3. Conteúdos relacionados	Introdução ao escalonamento: conceitos, tipos e escalonamento de threads																			
4. Descrição da SP	Para realizar a sua primeira avaliação de algoritmo através da modelagem determinística, estabeleça uma quantidade de processos que deseja testar, por exemplo, 5. Determine qual o tempo de espera de cada um. E teste de acordo com o algoritmo de escalonamento FIFO (FCFS). Represente esse procedimento e calcule o tempo médio de espera em milissegundos (quantum = 10 ms):																			
5. Resolução da SP	<table><tr><th>Processos</th><th>Tempo de Burst (tempo que a CPU leva para inserir outro processo da fila para execução e a espera por processos de E/S)</th><th>Tempo de espera (ms)</th></tr><tr><td>P1</td><td>10</td><td>0</td></tr><tr><td>P2</td><td>29</td><td>10</td></tr><tr><td>P3</td><td>3</td><td>39</td></tr><tr><td>P4</td><td>7</td><td>42</td></tr><tr><td>P5</td><td>12</td><td>49</td></tr></table>		Processos	Tempo de Burst (tempo que a CPU leva para inserir outro processo da fila para execução e a espera por processos de E/S)	Tempo de espera (ms)	P1	10	0	P2	29	10	P3	3	39	P4	7	42	P5	12	49
	Processos	Tempo de Burst (tempo que a CPU leva para inserir outro processo da fila para execução e a espera por processos de E/S)	Tempo de espera (ms)																	
	P1	10	0																	
	P2	29	10																	
	P3	3	39																	
	P4	7	42																	
	P5	12	49																	
	Considere que os tempos são:																			
	Representação:																			
	<table><tr><td>P1</td><td>P2</td><td>P3</td><td>P4</td><td>P5</td></tr><tr><td>0</td><td>10</td><td>39</td><td>42</td><td>49</td><td>61</td></tr></table>		P1	P2	P3	P4	P5	0	10	39	42	49	61							
P1	P2	P3	P4	P5																
0	10	39	42	49	61															
Tempo médio de espera = $(0 + 10 + 39+ 42+ 49)/5 = 28$ ms																				
Fonte: Deitel (2005, p. 145).																				



Faça você mesmo

Estabeleça outros tempos de espera para os cinco processos e calcule o seu tempo médio de espera a partir da modelagem determinística. Simule no soSim e tome nota dos resultados para comparar os algoritmos de escalonamento (SJF e RR) e o seu respectivo comportamento.



Lembre-se

Esse método apanha uma carga de trabalho específica predeterminada e define o desempenho de cada algoritmo para essa carga de trabalho. [...] A modelagem determinística é simples e rápida. Ela nos dá números exatos, permitindo que os algoritmos sejam comparados (DEITEL, 2005, p. 145).

Faça valer a pena!

1. Analise as afirmações. Com relação ao escalonamento de *threads*, é correto o que se afirma em:

I – Pode sair de execução e entrar em espera, para que outro processo de maior prioridade seja escalonado.

II – O *thread* de maior prioridade pode executar todo o tempo até o seu encerramento ou, se for *multithread*, poderá realizar a alternância circular.

III – Não pode sofrer preempção para que outro *thread* de maior prioridade seja executado.

- a) I e II.
- b) I, II e III.
- c) I e III.
- d) II e III.
- e) I apenas.

2. Complete as lacunas da frase com as palavras de uma das alternativas a seguir:

"O _____, através de uma ação _____, pode interromper um processamento em função da alocação temporária do recurso para que um processo menor seja eliminado da _____".

- a) SRT/não preemptiva/fila.
- b) SJF/preemptiva/fila.
- c) FIFO/preemptiva/pilha.
- d) SRT/não preemptiva/lista.
- e) SRT/preemptiva/fila.

3. Associe na tabela os conceitos às suas respectivas descrições:

Critérios	Descrição
a. Baixo nível	() Determina quais grupos de processos poderão trabalhar e controla o número de processos criados em um espaço de tempo.
b. Intermediário	() É possível interromper um processo e retomar em seguida com o seu processamento para que possa garantir o bom desempenho da máquina.
c. Alto nível	() Nessa política de escalonamento, normalmente há a atribuição de prioridade para o processo. Com isso, aumenta a probabilidade de ele ser escolhido para processamento.

Assinale a alternativa que contém a ordem de associação correta:

- a) a-b-c.
- b) b-c-a.
- c) a-c-b.
- d) c-b-a.
- e) b-a-c.

4. Descreva como é o mecanismo de funcionamento do escalonamento HRRN.

5. Explique como acontece a seleção de processos no algoritmo SRT.

6. As afirmações a seguir são verdadeiras ou falsas? Assinale a alternativa correta:

I – A modelagem determinística é simples e rápida.

II – O modo de tratamento de um escalonamento de *thread* de usuário (escalonamento por *timeslicing* ou espaço de tempo) e de núcleo são distintos no sistema operacional.

III – Escalonamento de tempo real não crítico: visa garantir que processos em tempo real sejam executados, porém sem a garantia de que cumprirá alguma restrição de tempo.

- a) F-F-F.
- b) V-V-V.

- c) V-F-V.
- d) F-F-V.
- e) F-V-V.

7. Está descrito o algoritmo de escalonamento por tempo real em:

I – Permitem a alocação de algoritmos distintos de acordo com a solicitação que foi realizada.

II – Escalonamento de tempo real dinâmico: esses realizam o escalonamento por prioridades e isso acontece durante a execução dos processos.

III – A partir da quantidade de processos e subprocessos que estão associados a um determinado indivíduo, sendo que este compartilha recursos da máquina com outros colegas, o escalonamento procura identificar o grupo de processos associados aos usuários de forma a identificar aquele que mais necessita de recursos em função de suas operações para que possa realizar o escalonamento.

- a) I e II.
- b) II e III.
- c) I, II e III.
- d) I e III.
- e) Apenas III.

Referências

DEITEL, H. M.; DEITEL P. J.; CHOFFNES, D. R. **Sistemas operacionais**. 3. ed. São Paulo: Prentice Hall, 2005.

MACHADO, Francis B.; MAIA, Luiz P. **Arquitetura de Sistemas Operacionais**. 5. ed. Rio de Janeiro: LTC, 2013.

SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGME, Greg. **Sistemas operacionais: sistemas e aplicações**. 6. ed. Rio de Janeiro: Elsevier, 2004.

STUART, Brian L. **Princípios de sistemas operacionais: projetos e aplicações**. São Paulo: Cengage Learning, 2010.

TANENBAUM, Andrew S. **Sistemas operacionais modernos**. 3. ed. São Paulo: Pearson Prentice Hall, 2009.

SISTEMA DE ARQUIVOS

Convite ao estudo

Prezado aluno, vamos iniciar os estudos dos sistemas de arquivos. Mas, antes, responda: “Você sabe o que é um arquivo?”. Muito bem, arquivos são estruturas de dados ou instruções, que estão relacionadas logicamente e se associam a um determinado contexto. Esse pode ser um arquivo executável, que contém instruções que serão interpretadas pelo processador, ou, ainda, um arquivo de dados que pode ser um texto, uma planilha, ou, até mesmo, dados de um banco de dados relacional, ou seja, os dados são organizados em estruturas conhecidas como tabelas, que se relacionam logicamente através de um elemento do registro que seja comum e permita a interligação dos dados, por exemplo, através de um número de CPF.

A fim de organizar esses arquivos, o sistema operacional conta com uma estrutura chamada sistema de arquivos. Esse considera que o arquivo é um conjunto de registros que será utilizado e manipulado posteriormente (MACHADO; MAIA, 2013). Você pode agora associar tais definições com a forma que trabalha com os seus arquivos e registros. Imagine utilizar o computador para trabalhar, gerando diversos tipos de arquivos. Se não existisse um mecanismo que permitisse a sua organização, em função do volume e tipo de informação que trabalhamos, haveria certa dificuldade em localizar mais rapidamente um arquivo, ou, ainda, recuperá-lo em caso de incidentes.

Assim, já deve ter se deparado com situações em que precisa de uma informação e não sabe em que pasta está. Por exemplo: se esqueceu onde salvou o arquivo, salvou em um formato errado que não é compatível com o software instalado, dentre outras situações que podem acontecer, inclusive, com a dificuldade de localização, como mencionado. Nesse contexto, você

sabe apenas que ele foi salvo. Claro que essa não é uma situação comum, porque somos todos organizados, correto? Sim, talvez seja a sua resposta. Caso contrário, temos aí um problema!

Mas, pensando em facilitar a organização de arquivos aos usuários, é que o sistema de arquivos se faz importante. Além disso, não apenas em nível de organização de arquivos em um computador, mas, principalmente, ao permitir que usuários acessem os arquivos e os tenha à disposição sempre que necessário. Nesse contexto, para entendermos a importância do sistema de arquivos em um sistema operacional, vamos trabalhar com um cenário de uma microempresa do setor de alimentos, que faz a distribuição de alimentos orgânicos para os supermercados da região Norte do país. Em função das quedas frequentes de energia, o computador do PMO (Gerente de Projetos), em um desses episódios, queimou. Todas as informações dos contratos estavam centralizadas nesse computador e não tinham uma política de *backup* bem estabelecida.

Diante dessa situação, precisam recuperar a maior quantidade possível de dados e informações do disco rígido. Porém, cientes de que essa não será uma tarefa fácil, precisarão, além de estabelecer alguns critérios de processo de segurança da informação, recuperar e verificar a melhor forma de organizar esses arquivos e conseguir manipular e gerenciar essas informações sem prejudicar o andamento dos projetos. Sua missão está dada. Bons estudos e práticas para você!

Seção 3.1

Arquivos: atribuição de nomes, estrutura, tipos, acesso, atributos e operações

Diálogo aberto

Os sistemas de arquivos servem para facilitar a organização de arquivos sob o ponto de vista do armazenamento e de sua identificação. Para que esse gerenciamento aconteça, é preciso que o sistema de arquivos realize operações para criar arquivos, bem como seja o responsável pela sua exclusão ou remoção. Além dessas funções, podemos mencionar outras básicas, como abrir, ler, gravar, fechar, identificar e reconhecer, obter e modificar arquivos com metadados, ou seja, que fazem referência ao dado, atribuindo características e especificando outras informações sobre o arquivo (STUART, 2011).

Nesse contexto, todos os componentes do sistema operacional se relacionam, pois não se trata apenas de armazenar e processar. Organizar esses dados é de extrema importância para que haja o respectivo direcionamento no sistema computacional. Com isso, em função do tipo de processo que será acionado, outro fator importante é a extensão dos arquivos, pois, para cada uma, será realizado um procedimento de armazenamento de alocação de recursos distinto. Conheça na Tabela 3.1, a seguir, algumas extensões que estudamos com certa frequência:

Tabela 3.1 | Extensão de arquivos

Extensão	Descrição
ARQUIVO.BAS	Arquivo fonte em BASIC.
ARQUIVO. COB	Arquivo fonte em COBOL.
ARQUIVO. EXE	Arquivo executável.
ARQUIVO. OBJ	Arquivo objeto.
ARQUIVO. PAS	Arquivo em Pascal.
ARQUIVO. TXT	Arquivo texto.

Fonte: Machado e Maia (2013, p. 195)

Acima, estão apresentadas algumas das extensões de arquivos existentes, de acordo com o aplicativo ou linguagem de programação em uso, ou seja, outras

extensões, como “.doc”, .xls”, por exemplo, respectivamente arquivos de texto e planilhas, precisam, da mesma forma, ser gerenciadas pelo sistema de arquivos. Além desses, há diversos outros que o sistema operacional deve reconhecer os formatos através do(s) seu(s) sistema(s) de arquivos.

A fim de aproximar a teoria e a prática profissional, vamos trabalhar com o cenário proposto de recuperação de dados e definição dos sistemas de arquivos que serão mais eficientes para as necessidades da microempresa do setor de alimentos. Para esse caso, o que você sugere para evitar que a empresa perca mais informações? Nesse contexto, para minimizar as perdas de dados, é necessário primeiramente realizar um procedimento que permite a recuperação dos dados. No entanto, não sendo essa uma tarefa muito fácil, você precisa especificar o passo a passo utilizado e os *softwares* que podem auxiliar nesse processo.

Não pode faltar

Vamos iniciar os estudos dos sistemas de arquivos e, para tal, precisamos compreender o que são, como se organizam, quais são os tipos, como são acessados sob o ponto de vista computacional, ou seja, de que forma o computador entende os comandos e o sistema operacional gerencia esses acessos. Além disso, é importante conhecer as características e os atributos dos sistemas de arquivos e quais são as suas respectivas funções.

Para o sistema operacional, quando um arquivo precisa ser aberto, criado, fechado, enfim, qualquer uma das funções que o sistema de arquivos precisa desempenhar, temos de saber que a operação em andamento é compreendida pelo computador como um processo que deverá ser executado e direcionar o driver correto para que essa ação aconteça (podemos chamar de driver o conjunto de rotinas que devem ser executadas de acordo com o formato do arquivo).

Nesse sentido, precisamos configurar as permissões necessárias à execução de cada uma dessas ações. Questões relacionadas à segurança e integridade dos dados, que precisam ser compartilhados não apenas no quesito recurso de armazenamento, como também de processamento, e, ainda, pertencer a um único arquivo, por exemplo, tornam-se ainda mais relevantes, em função do risco de falhas e inconsistência dos dados que estão sendo compartilhados. Confira, no Quadro 3.1, uma relação das principais funcionalidades dos sistemas de arquivos:

Quadro 3.1 | Operações dos sistemas de arquivos

Manipulação de arquivos	Manipulação de dados dos arquivos	Atributos de arquivos	Rotinas de entrada e saída de arquivos
Abrir	Ler	Tamanho	CREATE (criar)
Fechar	Escrever	Localização	OPEN (abrir)
Criar	Atualizar	Acessibilidade	READ (ler)
Destruir	Inserir	Tipo	WRITE (gravar)
Copiar	Apagar	Volatilidade	CLOSE (fechar)
Renomear	-	Atividade	DELETE (eliminar)
Listar	-	Backup	-

Fonte: Adaptado de Deitel et. al. (2005, p. 379); Machado e Maia (2013, p. 197-198)

Observe, no Quadro 3.1, as operações básicas que um sistema de arquivos precisa realizar. Nesse contexto, é possível trabalhar com os dados e registros dos arquivos. Essas ações consistem em permitir a leitura, atualização, escrita, inserção e remoção de dados. Além desses, o quadro aponta os atributos dos arquivos. O atributo “tamanho” refere-se à quantidade de dados que um arquivo armazena. Já localização fornece o diretório, dispositivo de armazenamento ou pastas em que se encontra o arquivo. Fica evidente a lógica da organização do sistema de arquivos, de forma a permitir sua localização. Acessibilidade atribui restrições quanto à permissão para manipular o arquivo. O tipo define se é um arquivo executável por exemplo, ou mesmo um arquivo de texto. Volatilidade refere-se à quantidade de vezes que um arquivo passa por alterações, atualizações e mesmo remoção de informações. Por fim, o Quadro 3.1 traz o atributo “atividade” como um indicador das alterações que foram realizadas no conteúdo do arquivo, ou seja, em seu registro.

Os registros podem ser definidos como do tipo lógico ou físico. São características dos registros lógicos: campos, que contêm atributos como nome, tipo e comprimento. Além disso, os registros lógicos podem representar um valor constante ou variável. Registros físicos referem-se às ações que serão desempenhadas pelo sistema de arquivos, como as exemplificadas no Quadro 3.1.



Assimile

Além da interface de usuário, o aspecto mais reconhecível de um sistema operacional é seu sistema de arquivos. Isso não é de se estranhar. Afinal, todo programa que executamos, toda imagem exibida e todos os nossos dados são gerenciados pelo sistema de arquivos. É o sistema de arquivos que determina como esses arquivos são armazenados e como podem ser identificados (STUART, 2011, p. 449).

Em nosso cotidiano, trabalhamos com diversos tipos de arquivos e também utilizamos vários meios de armazenamento desses dados. Podemos citar diversos dispositivos, desde o próprio disco rígido (*Hard Disk* – HD) da máquina, ou mesmo um HD externo, um CD, um pen drive, enfim, independente do tipo de mídia ou dispositivo de armazenamento, o sistema operacional tratará essas informações de arquivos isoladamente através do sistema de arquivos.



Reflita

[...] o disco é um recurso compartilhado, sua utilização deverá ser gerenciada unicamente pelo sistema operacional, evitando que a aplicação possa ter acesso a qualquer área do disco sem autorização, o que poderia comprometer a segurança e a integridade do sistema de arquivos (MACHADO; MAIA, 2013, p. 193).

Mas, afinal, como o sistema operacional realiza essa organização de arquivos? Em geral, os arquivos podem apresentar estruturas de armazenamento de dados distintas, variando de acordo com o tipo de informação do arquivo. Nesse contexto, você pode se perguntar: quais são essas estruturas de dados? Estamos nos referindo a arquivos e aos seus respectivos diretórios. Essa estrutura pode ser definida no momento da criação do arquivo, ou mesmo ser uma rotina automática, definida de acordo com a aplicação.

Além do aspecto da estrutura de dados do arquivo, outra característica do sistema de arquivos é a questão de sua organização no sistema computacional. Segundo Machado e Maia (2013), uma das formas mais simples de se organizar arquivos é através de bytes, o que não implica a existência de uma estrutura lógica, como as mencionadas, pois essas podem ser estabelecidas de acordo com os critérios da própria aplicação. Assim, é possível obter uma classificação para a organização de arquivos. Temos, então, a sequencial, também conhecida como não estruturada, a indexada e a relativa. Observe no exemplo:



Exemplificando

Vejamos o exemplo em que são apresentadas as organizações de arquivos do tipo sequencial e indexada:

a. Sequencial (não estruturada):

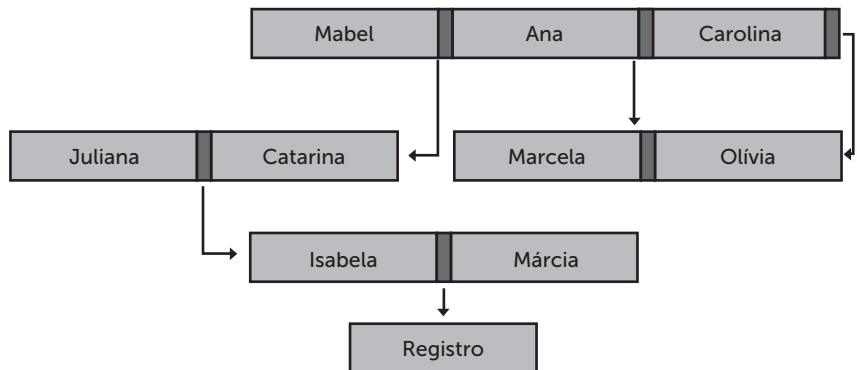
Byte

Nessa forma de organizar os arquivos, não há necessariamente uma estrutura lógica definida, preestabelecida. Por esse motivo, também, o acesso ao registro é precário, pois é necessária a leitura de todos os registros, até que seja encontrado o que está de fato sendo procurado. Então, o que se tem é a organização dos arquivos feita sequencialmente, de acordo com a limitação de bytes da aplicação, ou seja, o tamanho dos arquivos ou dos blocos, que é pré-determinado. A organização de arquivos obedece a uma ordem física.

b. Indexada

Observe, na Figura 3.1, a seguir, que os dados se interligam através de um índice, que, a partir de sua definição, organizará os dados de um registro de acordo com uma determinada regra. Nesse contexto, podemos citar como exemplo, no registro a seguir, em que os dados se interligam a partir de informações incomum com os demais. Essa ligação pode ser através de uma definição de idade, altura, sexo e classe social que sejam compatíveis. Nesse caso, é essencial que exista um dado em comum entre os registros do arquivo para que se obtenha tal indexação. Esse dado em comum é chamado de chave de acesso.

Figura 3.1 | Organização de arquivos indexada



Fonte: Machado e Maia (2013, p. 195)

c. Relativa ou direta

A organização de arquivos de forma relativa refere-se à alocação dinâmica e aleatória do arquivo. Por esse motivo, ele poderá ser localizado a partir de uma chave de acesso e do local em que está armazenado.

O sistema de arquivos terá a responsabilidade de gerenciar o caminho do dispositivo ou local de armazenamento do arquivo. Dessa forma, permite não apenas facilitar a localização dele, como também especificar quais são as suas características e atributos.

O nome do arquivo será limitado ao padrão e às regras estabelecidas pelo sistema de arquivos. A essa característica pertence, então, a disponibilização dos possíveis nomes que os arquivos podem assumir, conhecida como espaço de nomes. A função do sistema de arquivos, nesse caso, é converter os nomes em locais onde esse arquivo está armazenado. Os nomes dos arquivos podem solicitar a ação de drivers de dispositivos, por exemplo, de acordo com Stuart (2011), de comunicação ou ainda que realizam a interface com outros processos. Os nomes podem fazer referência a dados já existentes, ou que são gerados durante o processamento ou ainda, a nenhum dado.

A organização de arquivos antigamente compreendia apenas a especificação da unidade de fita em que se localizavam e em que eram armazenados, combinando em seu nome a identificação da unidade física e o respectivo nome do arquivo, sem necessariamente definir locais, atributos e informações de volatilidade, por exemplo. Do mesmo modo, quando surgiram os discos, eram especificadas as unidades de disco e o respectivo nome do arquivo, por exemplo, seguindo o padrão de espaço definido como $N = (A:)opt A [1,6]. A3$, que podemos interpretar da seguinte forma: $AS1:TESTE.TXT$, em que "AS1" é o da unidade de disco, em que se encontra o arquivo, e "TESTE.TXT", o nome do arquivo ali armazenado. Quando se trata da arquitetura dos sistemas de arquivos, essa prevê que é necessário identificar sempre a unidade de disco em que se encontra o arquivo, ou seja, está previsto que as unidades de disco podem ser particionadas, diferentemente da antiga ordenação em filas das fitas de armazenamento.

Outro fator importante a saber é a qual usuário ou conta está associado o arquivo, ou seja, a especificação da conta em que foi criado. A partir da definição do espaço de nomes equivalente a $D = \{0, 1, 2, 3 \dots 9\}$, temos, então, a definição de $N = ([D+, D+]) opt A [1,6]. A3$, o que permite números relativamente grandes nos nomes, porém com uma limitação de caracteres em cerca de três ou quatro. Como exemplo desse formato, temos, segundo Stuart (2013, p. 460): "[130,14]FORTH.ASM".

Outro exemplo que o autor supramencionado apresenta é para o formato $N = ([A [1,8].]) +]) opt A [1,6]. A3$, o que representa a permissão para que o nome do arquivo contenha até 8 caracteres, considerando letras e dígitos alfanuméricos. Nesse caso, o nome mencionado pode ser alterado para [TESTE01]FORTH.ASM. Ao correlacionar a identificação do diretório a que pertence o arquivo criado, obtemos o seguinte: $N = (A3:) opt ([D+ D+] +]) opt A [1,6]. A3$, o que representa a seguinte estrutura na definição do nome do arquivo: $DX0: [130,14]FORTH.ASM$.

Observe que, na estrutura apresentada, é possível que sejam estabelecidas as áreas

de usuário separadas para cada unidade física de armazenamento, sendo possível determinar quais são os componentes opcionais presentes no nome e com isso dividir os componentes de uma string inteira, ou seja, delimitar e identificar as partes da sequência de caracteres que compõem o nome do arquivo (STUART, 2011).

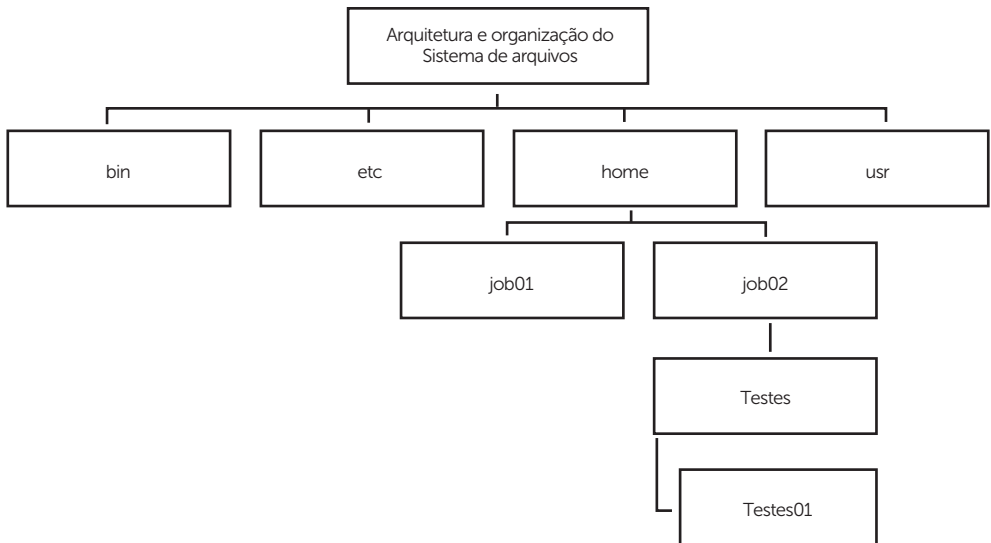


Faça você mesmo

Leia o artigo "OS Simulator: Um simulador de sistemas de arquivos para apoiar o ensino-aprendizagem de sistemas operacionais", disponível em: <<http://www.lbd.dcc.ufmg.br/colecoes/sbie/2010/0060.pdf>>. Acesso em: 18 ago. 2015. Descubra outros tipos de simuladores e baixe o OS Simulator para testar e praticar com os novos conhecimentos que foram agregados.

O sistema de arquivos identificará um arquivo a partir do seu nome, que é uma sequência de caracteres que podem conter letras maiúsculas, minúsculas, números, dependendo do tipo de aplicação, obedecendo a uma restrição de extensão e tipos de caracteres válidos (MACHADO; MAIA, 2013). Além dessas características, há também a possibilidade de um usuário determinar de forma hierárquica a organização dos seus arquivos, como se seguissem uma estrutura em árvore:

Figura 3.2 | Espaço de nomes hierárquico



Fonte: Adaptado de Stuart (2011, p. 461)

Vamos compreender melhor o que representa a Figura 3.2. Acompanhe a seguir a descrição dos seus elementos:

- bin: nessa pasta, encontram-se arquivos executáveis, que são necessários para a recuperação de dados e reparação de arquivos do sistema;
- etc: esse diretório armazena arquivos que servem para realizar a configuração de arquivos locais e *softwares*;
- home: esse diretório está associado ao usuário direta ou indiretamente. Nesse contexto, necessita da administração local;
- usr: esse diretório indica uma partição e os respectivos arquivos que podem ser compartilhados no modo somente leitura.

Agora que você já conheceu as características dos sistemas de arquivos, siga em frente e aprenda ainda mais!



Pesquise mais

O artigo disponível na *site* da Microsoft traz as principais características e diferenças sobre os sistemas de arquivos. Disponível em: <<https://support.microsoft.com/pt-br/kb/100108>>. Acesso em: 13 ago. 2015.

Sem medo de errar

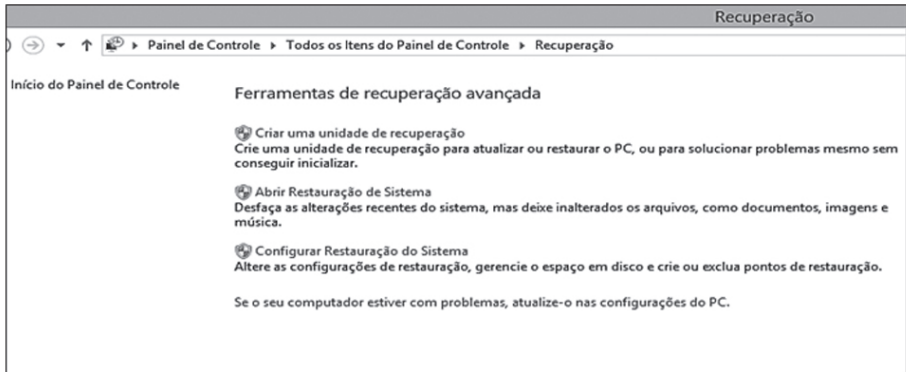
Considerando que estamos trabalhando com um sistema de arquivos do tipo NTFS e a fim de recuperar os dados perdidos com a queda de energia do computador do PMO da microempresa de alimentos, será apresentado o *software* Minitool Partition Recovery. Esse *software* pode restaurar os padrões deteriorados dos sistemas de arquivos do tipo IDE disk, SATA disk, SCSI disk and Removable disk, FAT12, FAT16, FAT32, VFAT, NTFS, NTFS file systems. Assista ao vídeo e siga o passo a passo sugerido! (Disponível em: <<https://www.youtube.com/watch?v=5gVmCo0fEbM>>. Acesso em: 19 ago. 2015). Vamos ter de seguir os seguintes passos para tentar recuperar os dados do computador que sofreu perdas de dados com queda de energia:

1. Instalar o *software* de recuperação, no caso o mencionado, Minitool Partition Recovery.
2. Selecione o diretório e a partição que precisa de reparos.
3. Será preciso verificar, através da função “Specified Range”, o local em que será realizada a verificação, e, em seguida, inicie a leitura do diretório.
4. Fique atento e selecione todas as partições utilizadas e não se esqueça de nenhuma. Caso contrário, aquelas que você, porventura, não tiver selecionado serão

apagadas.

5. Depois de realizado este procedimento, você precisa verificar se os arquivos de fato foram recuperados.

Figura 3.3 | Ferramentas de recuperação de arquivos disponíveis no sistema operacional



Fonte: O autor

6. Acesse o Painel de Controle e veja que também é possível realizar um procedimento de recuperação de dados. Observe a tela abaixo, do sistema operacional Windows 8, que apresenta essa funcionalidade.



Atenção!

Faça o *download* do software *Minitool Partition Recovery*. Disponível em: <<http://www.minitool-partitionrecovery.com/>>. Acesso em: 18 ago. 2015.



Lembre-se

Veja outras dicas de recuperação de dados em sistemas de arquivos NTFS, FAT32 e FAT. Disponível em: <<http://www.tecdicas.com/recuperar-particao-ntfs-fat32-e-fat-em-segundos/>>. Acesso em: 18 ago. 2015.

Avançando na prática

Pratique mais

Instrução

Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.

Arquivos: atribuição de nomes, estrutura, tipos, acesso, atributos e operações							
1. Competência de fundamentos de área	O aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional, bem como ter conhecimento sobre como se dá o compartilhamento de recursos e a sua gerência.						
2. Objetivos de aprendizagem	Conhecer o que é um arquivo, sua estrutura e atributos. Conhecer como é o mecanismo de organização e hierarquia para gerenciar arquivos e diretórios. Conhecer e saber como é a implantação do sistema de arquivos e, ainda, identificar outros mecanismos, como virtualização e o impacto dela para o sistema operacional. Conhecer as formas de fazer a proteção e garantir a segurança do sistema de arquivos.						
3. Conteúdos relacionados	Arquivos: atribuição de nomes, estrutura, tipos, acesso, atributos e operações.						
4. Descrição da SP	Dentre os serviços solicitados, uma microempresa contratou serviços de descrição dos sistemas de arquivos que utilizam, a fim de se precaver e saber como proceder no caso de ocorrer algum outro erro do sistema e determinar aquele que lhe oferecerá maior segurança e integridade dos arquivos. Descreva os principais tipos de sistemas de arquivos utilizados em Windows e em Linux.						
5. Resolução da SP	<table border="1"> <thead> <tr> <th>Sistemas de arquivos – Windows</th><th>Descrição</th></tr> </thead> <tbody> <tr> <td>FAT (FAT32- Windows 98/95 e Windows 2000)</td><td>Simples: Armazenado em local fixo para facilitar o acesso aos arquivos, cria uma cópia de segurança deles. Seu tamanho é pré-definido em função do volume de informações. A tabela FAT indica qual é o cluster em que o arquivo está armazenado. Não admite definição de permissões em arquivos FAT.</td></tr> <tr> <td>HPFS (Windows NT 3.1, 3.5 e 3.51.)</td><td>Organização de arquivos igual à do FAT, porém há a classificação automática no diretório, que é baseada nos nomes dos arquivos. Armazenamento em unidades físicas e não mais em clusters de partições. O arquivo pode ser composto por dados e atributos de nomeação de arquivos. Esse sistema de arquivos aponta para o que chamamos de FNODE (contém os dados dos arquivos). Provê maior eficiência no processamento sequencial.</td></tr> </tbody> </table>	Sistemas de arquivos – Windows	Descrição	FAT (FAT32- Windows 98/95 e Windows 2000)	Simples: Armazenado em local fixo para facilitar o acesso aos arquivos, cria uma cópia de segurança deles. Seu tamanho é pré-definido em função do volume de informações. A tabela FAT indica qual é o cluster em que o arquivo está armazenado. Não admite definição de permissões em arquivos FAT.	HPFS (Windows NT 3.1, 3.5 e 3.51.)	Organização de arquivos igual à do FAT, porém há a classificação automática no diretório, que é baseada nos nomes dos arquivos. Armazenamento em unidades físicas e não mais em clusters de partições. O arquivo pode ser composto por dados e atributos de nomeação de arquivos. Esse sistema de arquivos aponta para o que chamamos de FNODE (contém os dados dos arquivos). Provê maior eficiência no processamento sequencial.
Sistemas de arquivos – Windows	Descrição						
FAT (FAT32- Windows 98/95 e Windows 2000)	Simples: Armazenado em local fixo para facilitar o acesso aos arquivos, cria uma cópia de segurança deles. Seu tamanho é pré-definido em função do volume de informações. A tabela FAT indica qual é o cluster em que o arquivo está armazenado. Não admite definição de permissões em arquivos FAT.						
HPFS (Windows NT 3.1, 3.5 e 3.51.)	Organização de arquivos igual à do FAT, porém há a classificação automática no diretório, que é baseada nos nomes dos arquivos. Armazenamento em unidades físicas e não mais em clusters de partições. O arquivo pode ser composto por dados e atributos de nomeação de arquivos. Esse sistema de arquivos aponta para o que chamamos de FNODE (contém os dados dos arquivos). Provê maior eficiência no processamento sequencial.						

	NTFS	Organiza os arquivos em diretórios como o HPFS. Garante recuperação de dados, identificação e remoção de falhas fatais ao sistema e evita o envio de mensagens de erro ao sistema operacional, técnica conhecida como hot fixing. É possível a recuperação de dados em função de se trabalhar com registros que apontam o ponto em que ocorreu a interrupção ou falha, sendo possível recuperar o arquivo.								
	<table><tr><th>Sistemas de arquivos Linux</th><th>Descrição</th></tr><tr><td>Journaling</td><td>Área do sistema operacional responsável pelos registros de informações de arquivos e das respectivas atividades básicas, que foram realizadas em um determinado arquivo. Com isso, oferece alta tolerância a falhas. Há o Journal Físico que faz uma cópia dos arquivos depois de gravados no sistema de arquivos principal e o Journal Lógico, responsável por gravar os metadados que podem sofrer as alterações, como leitura/gravação e atualização de arquivos.</td></tr><tr><td>ReiserFS</td><td>Suporte a Journaling. Alocação dinâmica de arquivos. Utiliza a estrutura de dados para armazenamento conhecida como Árvore B+, sendo que os dados são armazenados em unidades chamadas de folhas. Mais eficiente para acesso e gravação do arquivo.</td></tr><tr><td>XFS (64 bits)</td><td>Suporte a Journaling, que mantém a gravação de metadados de arquivos de forma a preservar a sua consistência. Realiza alocação mínima, que varia de 512 bytes a 64 kilobytes. Desfragmentação e redimensionamento on-line para redistribuição dos espaços alocados aos registros.</td></tr></table>		Sistemas de arquivos Linux	Descrição	Journaling	Área do sistema operacional responsável pelos registros de informações de arquivos e das respectivas atividades básicas, que foram realizadas em um determinado arquivo. Com isso, oferece alta tolerância a falhas. Há o Journal Físico que faz uma cópia dos arquivos depois de gravados no sistema de arquivos principal e o Journal Lógico, responsável por gravar os metadados que podem sofrer as alterações, como leitura/gravação e atualização de arquivos.	ReiserFS	Suporte a Journaling. Alocação dinâmica de arquivos. Utiliza a estrutura de dados para armazenamento conhecida como Árvore B+, sendo que os dados são armazenados em unidades chamadas de folhas. Mais eficiente para acesso e gravação do arquivo.	XFS (64 bits)	Suporte a Journaling, que mantém a gravação de metadados de arquivos de forma a preservar a sua consistência. Realiza alocação mínima, que varia de 512 bytes a 64 kilobytes. Desfragmentação e redimensionamento on-line para redistribuição dos espaços alocados aos registros.
	Sistemas de arquivos Linux	Descrição								
	Journaling	Área do sistema operacional responsável pelos registros de informações de arquivos e das respectivas atividades básicas, que foram realizadas em um determinado arquivo. Com isso, oferece alta tolerância a falhas. Há o Journal Físico que faz uma cópia dos arquivos depois de gravados no sistema de arquivos principal e o Journal Lógico, responsável por gravar os metadados que podem sofrer as alterações, como leitura/gravação e atualização de arquivos.								
ReiserFS	Suporte a Journaling. Alocação dinâmica de arquivos. Utiliza a estrutura de dados para armazenamento conhecida como Árvore B+, sendo que os dados são armazenados em unidades chamadas de folhas. Mais eficiente para acesso e gravação do arquivo.									
XFS (64 bits)	Suporte a Journaling, que mantém a gravação de metadados de arquivos de forma a preservar a sua consistência. Realiza alocação mínima, que varia de 512 bytes a 64 kilobytes. Desfragmentação e redimensionamento on-line para redistribuição dos espaços alocados aos registros.									

	ext4 (última versão do extFS)	Suporte a Journaling. Indexação de diretórios ampliada. Muito bom para servidores, pois permite até 1024 PB (Petabytes) para partições e até 16 TB (Terabytes) por arquivos. Suporte à recuperação de arquivos em função da indexação realizada. Utiliza um mecanismo de pré-alocação que otimiza esse processo.
	JFS	Suporte a Journaling. Recuperação de dados mais eficaz em função da identificação exata do ponto em que se encontra o arquivo danificado ou o erro ocorrido. Baixo consumo do processador. Trabalha principalmente com arquivos grandes. Suporta até 2 TB.



Lembre-se

O HPFS organiza uma unidade em uma série de bandas de 8 MB e, sempre que possível, um arquivo é contido dentro de uma dessas bandas. Entre cada uma dessas bandas estão 2K bitmaps de alocação, que mantêm um registro de quais setores dentro de uma banda foram ou não foram alocados. A banda aumenta o desempenho porque o cabeçalho da unidade não precisa retornar para o topo lógico geralmente o cilindro do disco, mas para o bitmap de alocação de banda mais próximo para determinar onde um arquivo será armazenado. (Disponível em: <<https://support.microsoft.com/ptbr/kb/100108>>. Acesso em: 18 ago. 2015)



Faça você mesmo

Estude como funcionam outros sistemas de arquivos, em Linux por exemplo. Acesse: <http://guidalinux.altervista.org/suselinux-manual_pt_BR-10.1-10/cha.filesystems.html>. Acesso em: 13 ago. 2015.

Faça valer a pena

1. Analise o parágrafo a seguir e complete as lacunas com as palavras de uma das alternativas a seguir, referentes aos conceitos apresentados:

O _____ tamanho refere-se à quantidade de dados que um arquivo armazena; já localização fornece o diretório, dispositivo de armazenamento ou pastas em que se encontra o arquivo. Quanto a/à _____, fica evidente a lógica da organização do sistema de arquivos, de forma a permitir sua localização. _____ atribui restrições quanto à permissão para manipular o arquivo. O tipo define se é um arquivo executável, por exemplo, ou mesmo um arquivo de texto. _____ refere-se à quantidade de vezes que um arquivo passa por alterações, atualizações e mesmo remoção de informações. Por fim, o quadro traz atividade como um indicador de registros constantes em um arquivo.

- a) localização / acessibilidade / volatilidade / atributo
- b) acessibilidade, volatilidade / atributo / localização
- c) volatilidade / atributo / localização / acessibilidade
- d) atributo / localização / acessibilidade / volatilidade
- e) tipo / atributo / localização / acessibilidade

2. Associe na tabela o sistema de arquivo à sua respectiva descrição e assinale a alternativa que contém a ordem correta:

Sistema de arquivo	Descrição
I. NTFS	() Suporte a Journaling. Indexação de diretórios ampliada. Muito bom para servidores, pois permite até 1024 PB (Petabytes) para partições e até 16 TB (Terabytes) por arquivos.
II. ReiserFS	() Organiza os arquivos em diretórios, como o HPFS. Garante: recuperação de dados, identificação e remoção de falhas fatais ao sistema e evita o envio de mensagens de erro ao sistema operacional, técnica conhecida como hot fixing.
III. ext4	() Suporte a Journaling. Alocação dinâmica de arquivos. Utiliza a estrutura de dados para armazenamento conhecida como Árvore B+, sendo que os dados são armazenados em unidades chamadas de folhas.

- a) III, I, II.
- b) I, II, III.

- c) II, I, III.
- d) III, II, I.
- e) I, III, II.

3. Considere as afirmações e assinale a alternativa correta:

I – Os registros podem ser definidos como do tipo lógico ou físico.

II – Os registros lógicos: campos, que contêm atributos como nome, tipo e comprimento.

III – Registros físicos referem-se às ações que serão desempenhadas pelo sistema de arquivos.

Está correto o que se afirma em:

- a) Apenas II.
- b) Apenas I.
- c) Apenas I e II.
- d) Apenas I e III.
- e) I, II e III.

4. Descreva o mecanismo de identificação do espaço de nomes de um sistema de arquivos.

5. Quanto à organização de arquivos, podemos definir um dos tipos da seguinte forma:

Nessa forma de organizar os arquivos, não há necessariamente uma estrutura lógica definida, preestabelecida. Por esse motivo, o acesso ao registro é precário, pois é necessária a leitura de todos os registros até que seja encontrado o que está de fato sendo procurado. Então, o que se tem é a organização dos arquivos feita sequencialmente, de acordo com a limitação de bytes que a aplicação, que é responsável por esse controle. Obedecem a uma ordem física.

Essa descrição refere-se a qual tipo de organização de arquivos? Assinale a alternativa correta:

- a) Indexada.
- b) Analítica.

- c) Sequencial.
- d) Volátil.
- e) Relativa.

6. São verdadeiras:

I – bin: encontram-se arquivos executáveis que são necessários para a recuperação de dados e reparação de arquivos do sistema.

II – etc: armazena arquivos que servem para realizar a configuração de arquivos locais e *softwares*.

III – home: está associado ao usuário, direta ou indiretamente. Nesse contexto, necessita da administração local.

- a) bin, home e etc representam comandos do DOS.
- b) bin, home e etc representam diretórios que contém informações de configuração do sistema.
- c) bin, home e etc são extensões de arquivos.
- d) bin, home e etc representam nomes possíveis de arquivos de usuários.
- e) bin, home e etc são atributos do sistema operacional.

7. Descreva a função do *journaling* para os sistemas de arquivos do Linux.

Seção 3.2

Diretórios: diretórios simples, sistemas de diretório hierárquico, nomes de caminho e operações

Diálogo aberto

Na seção anterior, descrevemos uma forma de realizar a recuperação dos arquivos de uma máquina. Agora, a microempresa está em expansão, ganhando novos mercados e por esse motivo precisa descentralizar as operações de máquinas individuais e torná-las disponíveis em um servidor que contemple, inclusive, o gerenciamento de todos os aplicativos e operações que são realizadas.

No entanto, houve um problema na instalação do *software*, em função da falha na instalação de um arquivo de extensão “.vbs” (VBScript, que é um subconjunto do Visual Basic). Esse *software* será importante em função de permitir a gerência das operações através do mapeamento e da identificação automática das aplicações.

Nesse contexto, para o sistema operacional Windows, se o valor padrão do arquivo mencionado foi alterado, será necessário reconfigurar o valor dessa variável no seu respectivo diretório, para que não aconteça o erro de entrada – não há mecanismo de script para a extensão de arquivo “.vbs”. Então, apresente a solução para resolver esse problema e, dessa forma, o sistema operacional conseguirá executar as ações necessárias a cada vez que esse *software* for utilizado.

Você já estudou, na seção anterior, um pouco sobre os tipos de arquivos, seus atributos e as suas extensões. Quando se fala que um *software* será instalado e, em função de uma variável padrão que teve o seu valor alterado, ele não funciona corretamente, é preciso que esteja claro em sua mente que, em sistemas de arquivos, trabalhamos com conceitos que vão além do armazenamento, pura e simplesmente. Trata-se da importância desde a configuração de uma variável global situada em um determinado diretório até a alocação de arquivos propriamente dita, bem como a hierarquia e organização desses arquivos.

Então, agora que você já sabe qual é a sua tarefa nesta aula, siga em frente com a leitura dos conceitos envolvidos quando o assunto é diretórios e suas características. Teste os seus conhecimentos e bons estudos!

Não pode faltar

Já estudamos quais são os atributos dos arquivos e vamos lembrá-los, a fim de compreender a importância da sua configuração e instalação corretas, desde o próprio sistema de arquivos até as aplicações que precisarão de sua mais complexa organização. Veja, a seguir, as descrições dos atributos dos arquivos que podem influenciar no desempenho e eficiência na execução de processos interligados a eles:

- **Tamanho:** esse atributo especifica o tamanho a quantidade de caracteres ou bytes do arquivo.
- **Proteção:** esse atributo especifica padrões de segurança de acesso aos arquivos.
- **Dono/Proprietário/Usuário:** estabelece quem criou, ou seja, associa ao usuário ou conta.
- **Criação:** delimita data e hora de criação do arquivo, para que, a partir disso, facilite, inclusive, a busca pelo arquivo.
- **Backup:** disponibiliza data e hora da última atualização.
- **Organização:** indica qual é a lógica e a hierarquia utilizada para armazenar os arquivos nos respectivos diretórios.
- **Senha:** essa visa estabelecer um critério de acesso e implementar maior segurança para a realização de ações com os arquivos.

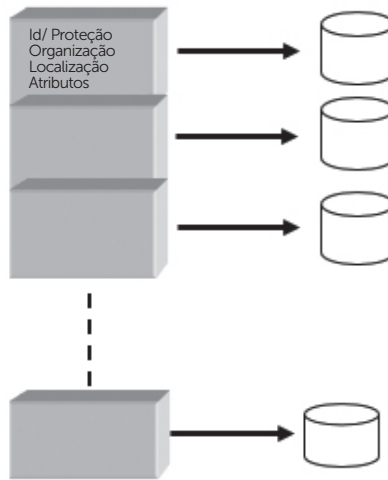
Além dessas características de atributos de arquivos, precisamos compreender como é a estrutura dos diretórios. A princípio, vamos conhecer a de nível único. Observe o exemplo a seguir e, em seguida, conheceremos também a estrutura de diretórios com dois níveis e em árvore.



Exemplificando

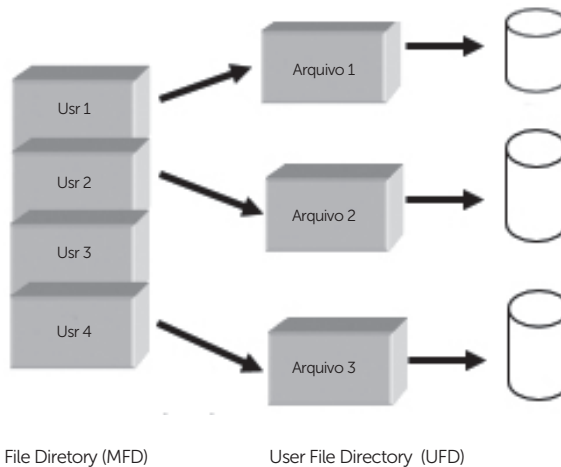
Confira, no exemplo a seguir, como acontece a localização de arquivos em estruturas de diretórios de nível único e com dois níveis:

Figura 3.4 | Estrutura de diretório de nível único



Fonte: Adaptada de Machado e Maia (2013, p. 198)

Figura 3.5 | Estrutura de diretório de dois níveis



Fonte: Adaptada de Machado e Maia (2013, p. 198)

Essa estrutura de diretórios de nível único, também chamada de *single-level directory*, tem muitas limitações. Por exemplo, os usuários não podem criar arquivos com o mesmo nome para evitar conflitos de acesso, pois, como observado nas Figuras 3.4 e 3.5, os arquivos recebem todas as características destacadas e são alocados para armazenamento. Mas, para evitar esse tipo de conflito, foi desenvolvida a estrutura de diretórios de dois níveis. O primeiro nível destina-se à divisão de contas de usuários e o outro à alocação dos arquivos criados.

A estrutura de diretório com dois níveis fez, conforme destacado na Figura 3.5, com que arquivos criados por contas de usuários distintos pudessem ter os mesmos nomes, pois não estão alocados no mesmo diretório e não causam danos de integridade ou conflitos na localização e acesso aos arquivos. Então, nessa estrutura, o diretório do usuário aponta para a sua respectiva área de alocação de arquivos, que foi destinada pelo próprio sistema de arquivos.

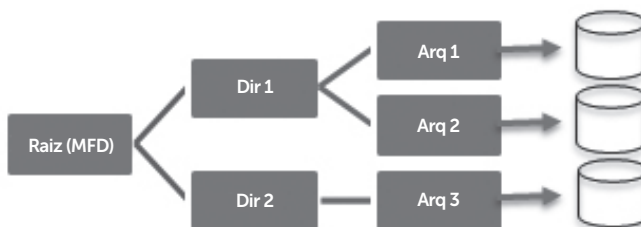


Refleta

A estrutura de diretórios é como o sistema organiza logicamente os diversos arquivos contidos em um disco. O diretório é uma estrutura de dados que contém entradas associadas aos arquivos em que cada entrada armazena informações com localização física, nome, organização e demais atributos (MACHADO; MAIA, 2013, p. 197).

Além dessas estruturas, temos ainda a de árvore, mapa de bits, lista encadeada e tabela de blocos livres. Observe as respectivas ilustrações:

Figura 3.6 | Estrutura de diretório em árvore



Fonte: Elaborada pelo autor (2015)

Na estrutura em árvore, a raiz é a própria área do usuário que está apontando para os diretórios e esses apontam para os seus respectivos arquivos. A análise se dá como em uma árvore em que a raiz é o próprio usuário, os galhos são os diretórios e as folhas são os arquivos. A essa sequência de caminhos dá-se o nome de path. Um exemplo comum é um usuário de nome JSD, que criou um arquivo chamado Teste.docx. No diretório, a sua localização se dará da seguinte forma: /JSD/Teste.docx. Lembre-se de que cada sistema de arquivo possui a sua própria regra de definição de nomes e espaços de nomes (MACHADO; MAIA, 2013).

Essa estrutura de organização de diretórios em árvores facilita também a organização dos arquivos para o usuário que pode separar os seus arquivos de acordo com as suas necessidades, pois permite que ele crie diversos níveis de diretórios, sendo que cada diretório pode ter outro diretório ou vários arquivos. A quantidade de níveis da estrutura em árvores pode variar de acordo com o sistema operacional (MACHADO; MAIA, 2013).



Assimile

Quando um arquivo é aberto, o sistema operacional procura a sua entrada na estrutura de diretórios, armazenando as informações sobre atributos e localização do arquivo em uma tabela mantida na memória principal. Essa tabela contém todos os arquivos abertos, sendo fundamental para aumentar o desempenho das operações com arquivos. É importante que ao término do uso de arquivos esses sejam fechados, ou seja, que se libere o espaço na tabela de arquivos abertos (MACHADO; MAIA, 2013, p. 198).

Para que o sistema operacional possa alocar os arquivos, é preciso que realize a gerência dos espaços disponíveis em disco, ou seja, quanto e quais partes do HD estão disponíveis. As estruturas de dados que podem ser usadas para essa alocação são basicamente lista e tabela, que também é conhecida como mapa de bits (bitmap).

Figura 3.7 | Mapa de bits

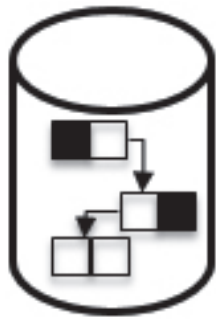
Mapa de bits
11001101
01110100
10000111
.
.
11100000

Fonte: Adaptada de Machado e Maia (2013)

Cada linha da tabela representa um bloco do HD. Os zeros (0) representam os espaços livres em memória, e um (1) os espaços ocupados. A desvantagem desse tipo de estrutura é que ocupa muito espaço em memória, uma vez que para cada bloco do HD deve existir uma entrada de dados na tabela.

Outra forma de realizar essa alocação de informações em blocos da memória é através de listas encadeadas, em que o controle ocorre através dos blocos do disco que estão sem arquivos, ou seja, livres. Isso é possível porque cada bloco contém uma área que determina o endereço que deverá ser utilizado pelo próximo bloco. A partir do primeiro bloco disponível que for encontrado, esse apontará para o endereço do próximo bloco disponível para realizar a alocação de dados, e, com isso, cria-se o conceito da lista encadeada. Observe a Figura 3.8:

Figura 3.8 | Alocação encadeada



Fonte: Adaptada de Machado e Maia (2013)

Outra forma comumente utilizada para alocação e gerência de espaço livre em memória é a tabela de blocos livres, que são alocados e também liberados de forma simultânea e, com isso, é possível visualizar apenas os espaços que realmente estão liberados no disco. Observe a ilustração:

Figura 3.9 | Tabela de blocos livres

Bloco	Contador
4	2
10	1
13	7
25	20

Fonte: Machado e Maia (2013, p. 201)

Quando se fala de gerenciamento de alocação de espaço em disco, podemos mencionar algumas formas de se realizar essa tarefa. Vamos conhecer a alocação contígua, alocação encadeada e a alocação indexada.



Faça você mesmo

Pesquise mais informações e *softwares* de simulação, testes, *games* e outros em: <<http://info.abril.com.br/downloads/windows/categoria/sistemas-operacionais/>>. Acesso em: 1º set. 2015.

A alocação contígua ou sequencial, como estudado na seção anterior, refere-se ao armazenamento do arquivo em blocos. Ele pode ser baseado no conceito de fila, nesse caso chamado de *first-fit*, que significa que o primeiro espaço livre no disco será também o primeiro a ser alocado se o seu tamanho for compatível com a do arquivo; outro modo de realizar a alocação contígua é chamado *best-fit*, em que há a alocação do menor espaço livre para um arquivo que seja compatível com esse pequeno

segmento de memória que esteja livre e, por fim, o mecanismo de alocação contígua, conhecido como *work-fit*. Nesse último, será alocado no espaço correspondente em disco o maior arquivo, no entanto, como não há uma ordenação por tamanho, é necessário buscar esse espaço por toda a lista antes da alocação (MACHADO; MAIA, 2013).

Na alocação encadeada, os blocos de disco são organizados logicamente de acordo com o ponteiro que indica o próximo bloco que está livre. Esse modo de alocação permite que o arquivo seja dividido em várias partes de acordo com o bloco de espaço livre no HD. A esse processo de divisão dá-se o nome de fragmentação. Com ela, é possível realizar a alocação de partes do arquivo, de acordo com o espaço disponível e ainda não necessariamente realizado de forma sequencial.

A desvantagem da alocação de forma fragmentada é que acaba resultando em um tempo maior para que o arquivo seja acessado ou mesmo em um aumento do tempo de gravação. Nesse caso, a fim de melhorar os tempos de leitura e gravação nas operações de entrada e saída de dados, é recomendada a realização do procedimento de desfragmentação do disco. A desfragmentação do disco é realizada de forma sequencial e contígua, seguindo o mesmo princípio. O acesso aos arquivos acontece de forma sequencial, o que torna essa a desvantagem desse mecanismo.

Além dessas e para finalizar os estudos sobre os diretórios, a alocação indexada visa solucionar algumas lacunas que a alocação de memória encadeada não contempla. Essa lacuna refere-se ao fato de, na alocação encadeada, não ser possível acessar o bloco de arquivos de forma direta. Com isso, na indexada, esse acesso é otimizado, pois mantém os ponteiros do registro para o bloco que deve ser lido ou acessado na sequência. Dessa forma, há um ganho em tempo de alocação, leitura e gravação dos dados.

Outra característica de que vale lembrar é que a alocação indexada não utiliza as informações de controle nos blocos de dados, pois há a indicação direta do objeto ou local de acesso subsequente.

O método de acesso utilizado para esse caso é conhecido como: acesso indexado ou acesso por chave. Por esse método, é possível estabelecer um índice que permita o acesso direto ao arquivo. Isso acontece em função de todo arquivo possuir uma área que determina esse índice, que será, então, acessado no momento em que for necessária a leitura, gravação, atualização ou qualquer uma das operações básicas que podemos realizar com arquivos. Nesse caso, os ponteiros indicam esses índices e procuram na estrutura de armazenamento, diretamente pelo índice estabelecido. Com isso, o ponteiro não acessará uma área que não seja correlata a ele e, por isso, acesso direto.

Confira, a seguir, uma demonstração em que há a necessidade de instalação de um *software* que auxiliará no controle de aplicações, porém algumas configurações

foram alteradas, e o *software* não está conseguindo executar as suas operações, pois não encontra um determinado arquivo que é chave para o seu bom funcionamento. Dessa forma, você poderá ter contato com uma situação que aproxima a teoria e os conceitos aqui apresentados, com uma situação que pode ocorrer no mercado de trabalho, no dia a dia de quem opera tais sistemas e administra o sistema operacional e as aplicações. Bons estudos!



Pesquise mais

Veja o que tem de novo para gerenciar aplicativos e arquivos do seu smartphone. Disponível em: <<http://codigofonte.uol.com.br/reviews/moborobo-gerenciador-de-smartphones-para-computador-review>>. Acesso em: 20 ago. 2015. Assista ao vídeo apresentado no artigo também!

Sem medo de errar

Para iniciar, vamos propor que a empresa utilize uma versão de sistema de gerenciamento de aplicações desenvolvida por um dos maiores fornecedores de soluções em produtos de *software*: a IBM. O *software* recomendado é o *Tivoli Application Dependency Discovery Manager*, ou TADDM. Esse *software* fornece um serviço capaz de identificar automaticamente e, ainda, fazer o mapeamento de aplicações que estão em uso. Com isso, os administradores do sistema podem acompanhar, inclusive, o estado da aplicação, suas configurações, quais recursos estão em estado mais crítico e fornecer uma visão mais clara sobre a interdependência tanto das aplicações quando dos sistemas de arquivos, sistemas operacionais e redes. Por esse motivo, está aqui a recomendação para facilitar o controle e gerenciamento das aplicações para a microempresa de alimentos. Como o problema atual visa corrigir o valor padrão de uma variável, que foi alterada diretamente no diretório, então vamos propor a seguinte solução, com base na proposta do próprio fornecedor da aplicação:

Primeiramente, há a identificação da falha no arquivo de extensão “.vbs”, em que foi alterado o valor padrão de uma variável de controle da aplicação.

1. Será exibida uma mensagem de erro através do arquivo denominado “taddm_7.2.1_install_msg”.
2. Recomenda-se como solução que o arquivo do Windows esteja associado à extensão.vbs e isto deve ser configurado.
3. A alteração de configuração será para que se tornem compatíveis. Dessa forma,

o valor padrão da variável deve indicar o *Microsoft Console Based Script Host*, para que possa ser associado. Outras informações de configuração podem ser obtidas no *site* do próprio sistema operacional.

Fonte: Disponível em: <http://www-01.ibm.com/support/knowledgecenter/SSPLFC_7.2.2/com.ibm.taddm.doc_7.2.2/InstallGuide/r_cmdb_install_troubleshooting.dita?lang=pt-br>. Acesso em: 21 ago. 2015.



Atenção!

Conheça o *Tivoli Application Dependency Discovery Manager*. Disponível em: <<http://www-03.ibm.com/software/products/pt/tivoliapplicationdependencydiscoverymanager>>. Acesso em: 21 ago. 2015.

Consulte informações sobre falhas em instalações de *softwares* de gerenciamento de aplicações como o TADDM. Disponível em: <http://www-01.ibm.com/support/knowledgecenter/SSPLFC_7.2.2/com.ibm.taddm.doc_7.2.2/InstallGuide/r_cmdb_install_troubleshooting.dita?lang=pt-br>. Acesso em: 21 ago. 2015.



Lembre-se

Saiba mais em: <http://www-01.ibm.com/support/knowledgecenter/SSANHD_7.5.3/com.ibm.sccd-sp.doc/rpm/t_swdist_taddm72x.html?lang=pt-br>. Acesso em: 21 ago. 2015.

Avançando na prática

Pratique mais	
Instrução Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.	
Diretórios: diretórios simples, sistemas de diretório hierárquico, nomes de caminho e operações	
1. Competência de fundamentos de área	O aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional, bem como ter conhecimento sobre como se dá o compartilhamento de recursos e a sua gerência.
2. Objetivos de aprendizagem	Conhecer o que é um arquivo, sua estrutura e atributos. Conhecer como é o mecanismo de organização e hierarquia para gerenciar arquivos e diretórios. Conhecer e saber como é a implantação do sistema de arquivos e ainda identificar outros mecanismos, como virtualização e o impacto dessa para o sistema operacional. Conhecer as formas de fazer a proteção e garantir a segurança do sistema de arquivos.

3. Conteúdos relacionados	Diretórios: diretórios simples, sistemas de diretório hierárquico, nomes de caminho e operações.
4. Descrição da SP	Para que consiga instalar e configurar todas as etapas do sistema de gerenciamento de aplicações que foi proposto para o cliente da microempresa de alimentos, você precisa seguir algumas recomendações. Isso porque o <i>software</i> , conhecido como TADDM (<i>Tivoli Application Dependency Discovery Manager</i>), precisa estar integrado ao servidor TADDM 7.2, conforme as recomendações do fornecedor. Então, descreva como é que são realizadas tais configurações, o passo a passo necessário para essa tarefa.
5. Resolução da SP	Siga o passo a passo sugerido para configurar a aplicação no seu respectivo servidor: 1. Localizar os JARs do Cliente de API do TADDM 7.2.x e copiá-los na estação de trabalho administrativa 2. Copiar os JARs do Cliente de API do TADDM 7.2.x na árvore de instalação do produto 3. Remover os JARs do Cliente de API do TADDM 7.1.x da árvore de instalação do produto 4. Modificar o arquivo XML de implementação do RPM 5. Reconstruir e reimplementar o arquivo EAR do Máximo.



Lembre-se

Os detalhes para cada uma dessas etapas estão descritos no artigo disponível em: http://www-01.ibm.com/support/knowledgecenter/SSANHD_7.5.3/com.ibm.sccd-sp.doc/rpm/t_swdist_taddm72x.html?lang=pt-br. Acesso em: 21 ago. 2015.



Faça você mesmo

Leia o artigo completo e observe quais são as características do aplicativo e o comportamento do sistema de arquivos, ou mesmo em que ele é impactado nesse tipo de operação. Boas práticas a você!

Faça valer a pena

1. A definição “_____”: esse atributo especifica o tamanho e a quantidade de caracteres ou bytes do arquivo” refere-se ao atributo:

- a) localização
- b) tamanho
- c) proteção
- d) criação
- e) acesso

2. Dadas as afirmações, assinale a alternativa que contém a informação correspondente:

I – Criação: delimita data e hora de criação do arquivo, para que, a partir disso, facilite, inclusive, a busca pelo arquivo.

II – *Backup*: disponibiliza data e hora da última atualização.

III – Organização: indica qual é a lógica e a hierarquia utilizada para armazenar os arquivos nos respectivos diretórios.

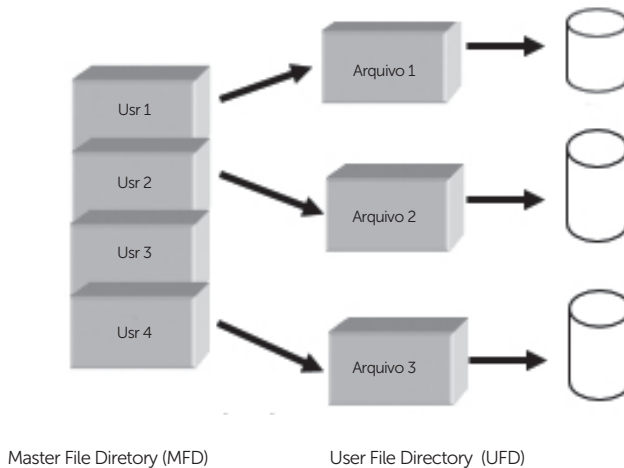
São corretas:

- a) I e II.
- b) II e III.
- c) Apenas I.
- d) I, II e III.
- e) Apenas III.

3. Dada a definição “visa estabelecer um critério de acesso e implementar maior segurança para a realização de ações com os arquivos”, assinale a alternativa correspondente ao conceito do atributo apresentado:

- a) Criação.
- b) Organização.
- c) Indexada.
- d) Encadeada.
- e) Senha.

4. Analise a figura e assinale a alternativa que contém o nome da estrutura de diretórios que essa representa:



Fonte: Adaptada de Machado e Maia (2013, p. 198)

- a) Diretório padrão.
- b) Diretório de nível único.
- c) Diretório com dois níveis.
- d) Diretório modelo.
- e) Diretório de lista encadeada.

5. Defina a estrutura de diretório em árvore.

6. Cite a principal desvantagem da alocação fragmentada.

7. Leia o trecho do texto abaixo e assinale a alternativa que contém os conceitos que preenchem as lacunas:

Ele pode ser baseado no conceito de fila, nesse caso chamado de _____, que significa que o primeiro espaço livre no disco será também o primeiro a ser alocado. Já se o seu tamanho for compatível com o do arquivo; outro modo de realizar a alocação contígua _____ aloca o menor espaço livre para um arquivo que seja compatível com esse pequeno segmento de memória que esteja livre e, por fim, o mecanismo de alocação contígua conhecido como _____. Nesse último,

será alocado no espaço correspondente em disco o maior arquivo, no entanto, como não há uma ordenação por tamanho, é necessário buscar esse espaço por toda a lista antes da alocação (MACHADO; MAIA, 2013).

- a) *first-fit* / *best-fit* / *work-fit*
- b) *fifo* / *work-fit* / *best-fit*
- c) *best-fit* / *work-fit* / *fifo*
- d) *fifo* / lista / pilha
- e) *best-fit* / pilha / *fifo*

Seção 3.3

Introdução à implementação do sistema de arquivos. Virtualização do sistema de arquivos e registro

Diálogo aberto

Olá, aluno. Vamos iniciar os estudos desta seção com os temas que abordam a implementação dos sistemas de arquivos, bem como a virtualização de arquivos e registros. Mas, afinal, como é que se aplicam esses conceitos na prática e dia a dia profissionais?

A fim de aproximar os conceitos com a sua aplicação no mercado de trabalho, trabalharemos, nesta aula, uma situação em que precisamos apresentar um modo de implementar um sistema de arquivos de rede, que seja compatível com os sistemas operacionais Windows, Unix e Linux.

Para realizar essa tarefa, você precisa apresentar um modo de implementação de um sistema de arquivos de redes que esteja de acordo com as especificações de um dos fornecedores do sistema operacional e explicar o procedimento.

Como estudado, você ainda pode fundamentar os seus estudos sobre o gerenciamento de sistemas de arquivos e os seus mecanismos de organização, assim como a sua alocação em memória. Nesse contexto, estudamos os modos de alocação indexada, mapa de bits, por lista encadeada, tabela de blocos livres. Ainda: outras estruturas também foram estudadas e podem auxiliar no desenvolvimento das competências necessárias para que você implemente o sistema de arquivos de rede que foi solicitado.

Analise se é uma estrutura de diretórios de nível único ou se se trata de uma estrutura de diretórios de dois níveis e elabore a proposta que atenda às necessidades de mudança do parque tecnológico da microempresa do setor alimentício que estamos estudando. Diante dessa pesquisa, retome os conceitos apresentados sobre a estrutura de diretórios em árvore, que é, inclusive, a mais utilizada atualmente. Considere também essa opção para a escolha do sistema de arquivos de rede que foi solicitado.

Então, agora que você já sabe a tarefa para realizar para esta aula, concentre as suas leituras no material didático e acesse os *links* sugeridos. Além disso, assista à webaula e resolva os exercícios propostos. Estudar faz bem e você pode aprender cada vez mais colaborando e contribuindo com novas situações que vivencia, além de poder associar os conceitos apresentados.

Desde já, bons estudos e práticas a você!

Não pode faltar

Implementação de sistemas de arquivos envolve basicamente criar para cada arquivo o seu respectivo descritor. Você sabe dizer o que é um descritor de arquivos? Então, confira a seguir a sua definição:



Assimile

Descritor de arquivos: “é um registro no qual são mantidas as informações a respeito do arquivo. Essas informações incluem os seus atributos, além de outros dados que não são visíveis aos usuários mas que são necessários para que o sistema operacional implemente as operações sobre arquivos” (OLIVEIRA; CARISSIMI; TOSCANI, 2010, p. 214).

O descritor de arquivos tem por função guardar as informações ou atributos dos arquivos. Ele possui as seguintes informações de acordo com os autores Oliveira, Carissimi e Toscani (2010, p. 214):

- nome do arquivo;
- sua extensão;
- tamanho, sempre definido em bytes;
- data e hora do último acesso;
- data e hora da última alteração;
- identificação do usuário que criou o arquivo;
- lista de usuários com permissões e tipos de acessos;
- localização dos arquivos.

O conteúdo do arquivo, em caso de interrupções, permanecerá intacto e,

consequentemente, o do descritor de arquivos também, sendo que o ideal é manter o descritor na mesma partição em que se encontra o arquivo. Isso facilita o acesso a ele. Lembre-se de que o descritor é acessado em todas as operações de leitura e escrita do arquivo.



Refleta

Para tornar mais rápido o acesso aos arquivos, o sistema de arquivos mantém na memória uma tabela contendo todos os descritores dos arquivos em uso. Quando um arquivo entra em uso, o seu descritor é copiado do disco para a memória. Ele pode ser acessado rapidamente sempre que necessário (OLIVEIRA; CARISSIMI; TOSCANI, 2010, p. 214).

Uma das funções do sistema operacional, para controlar os arquivos que estão em uso e aqueles que já não estão mais, é disponibilizar a informação correta. Por exemplo, se o descritor teve o seu valor alterado, o sistema operacional salvará uma cópia que sobrepõe a anterior e, portanto, essa cópia em disco estará sempre atualizada. As chamadas de sistema open/close também são incumbidas da tarefa de verificar se o arquivo foi alterado, se está em uso ou não mais. Para que isso aconteça, é preciso que um parâmetro de leitura seja executado ou mesmo de escrita, respectivamente, como apresentado a seguir: READONLY ou RO e READWRITE ou RW.

Uma tabela de descritores de arquivos abertos, também chamada de TDAA, é responsável por manter atualizadas as informações dos arquivos abertos. Isso ocorre para todos os processos do sistema, em função de um arquivo ser acessado por vários processos ao mesmo tempo (OLIVEIRA; CARISSIMI; TOSCANI, 2010).

Nesse contexto, quando há uma chamada de sistema do tipo open, o sistema de arquivos trata essas informações com as seguintes ações:

1. Localiza o respectivo descritor salvo em disco rígido, com a ação de *lookup*, em que há uma pesquisa entre os diretórios da partição em que se encontra o processo. Essa é uma operação interna do próprio sistema de arquivos. Se o arquivo não existir, o sistema de arquivos envia uma mensagem para o processo referente à chamada de sistema open. Se o procedimento de *lookup* encontrar o arquivo, deverá retornar uma mensagem que indique, respectivamente, o número da partição e o endereço do descritor.

2. Em seguida, há uma pesquisa que considera basicamente um índice do arquivo que se associa à tabela de descritores de arquivos abertos (TDAA) para verificar qual arquivo se encontra aberto. Essa busca utiliza os números de partição e de endereço do descritor liberados pelo procedimento de *lookup*.

3. Para o caso do arquivo ainda não estar aberto, é criado o seu respectivo descritor na TDAA, em um espaço que esteja livre, e ele já é salvo em memória no disco rígido do computador, portanto.

4. Depois de realizada a alocação em memória dos dados referentes ao arquivo, devidamente associados à tabela de descritores TDAA, o processo que solicitou a abertura do arquivo precisará de uma autorização do processo responsável por realizar essa tarefa. Nesse momento, acontece a verificação de permissões de acesso e a que tipo de processos isso se aplica. Por exemplo, caso o processo de solicitação de abertura de um arquivo seja o de um usuário sem a devida permissão de acesso, a chamada de sistema open se responsabiliza por executar e exibir o erro de acesso.

5. Apenas depois de todos os passos anteriormente descritos realizados será possível que o arquivo seja devidamente aberto, se assim for autorizado, para ser acessado. Com isso, a chamada de sistema close iniciará o procedimento de encerramento de processos que estão associados ao arquivo, liberando assim que identificar que já não há mais nada em andamento, o seu respectivo espaço na TDAA.

Lembre-se de que a principal função a trabalhar, quando se fala de implementação de arquivos, é criar um descritor para cada arquivo e associá-lo na tabela de descritores de arquivos abertos, liberando, com isso, o acesso ao arquivo solicitado, com as devidas verificações acerca da segurança dos dados ali contidos (OLIVEIRA; CARISSIMI; TOSCANI, 2010).

Um ponto de atenção é para a permanência de algumas informações do arquivo no descritor, pois elas são constantes e independem dos processos que os acessam. Mas, afinal, quais informações do arquivo dependem do processo? Por exemplo, a posição corrente no arquivo depende do processo associado à sua abertura, pois, para cada processo, há uma posição corrente do arquivo. Com isso, o sistema de arquivos permite ou não o acesso de acordo com a permissão do processo e do usuário a que se associa.

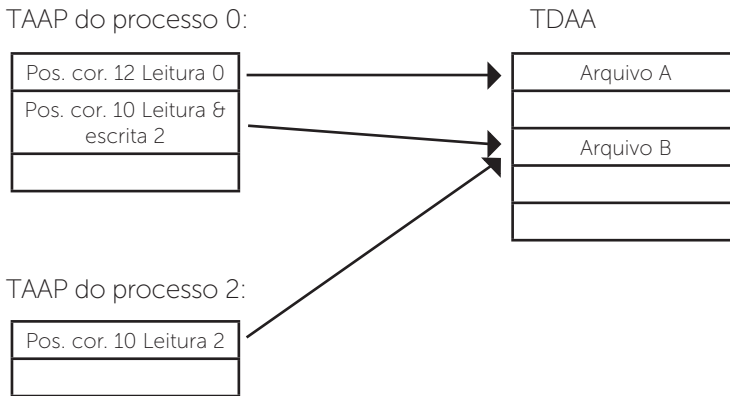
Com o intuito de reparar possíveis conflitos de acessos, o sistema de arquivos se encarrega de criar uma tabela descritiva de arquivos abertos por processo, chamada de TAAP. Então, a lógica é outra: nesse caso, para cada arquivo aberto é criada uma entrada na TAAP que contém a posição corrente do arquivo. Além dessa informação, há também o tipo de permissão associado e, ainda, um ponteiro que indica a sua correspondência na tabela dos descritivos de arquivos abertos.



Exemplificando

Como exemplo dessas operações, podemos incluir o uso das tabelas TAAP e TDAA. Observe na figura 3.10, que segue:

Figura 3.10 | Uso conjunto das tabelas TAAP e TDAA



Fonte: Oliveira, Carissimi, Toscani (2010, p. 217)

O que está evidenciado na Figura 3.10 é a existência de dois processos. O processo "0" tem o seu respectivo descritivo associado ao arquivo A e ainda esse mesmo processo acessa outro arquivo, o "B", e também se associam através do respectivo descritivo criado no momento de solicitação de abertura do arquivo. Então, temos um processo que trabalha com dois arquivos, ou seja, os acessa através de seus respectivos descritivos e posição corrente indicados. Além disso, o processo "0", para o arquivo B, tem permissões para realizar leituras e escrita.

A cada chamada de sistema de leitura ou escrita, o sistema de arquivos retornará o respectivo número de entrada na tabela de arquivos abertos por processo (TAAP). Com isso, a partir da TAAP, o sistema de arquivos poderá acessar o seu descritor na TDAA. Esse número de descritivo também é conhecido como handle do arquivo (OLIVEIRA; CARISSIMI; TOSCANI, 2010).

As duas tabelas são armazenadas em uma memória do próprio sistema operacional. Nesse caso, garantem que o usuário não tenha acesso a elas.

Como mencionado, as chamadas de sistema de leitura e escrita requerem que o sistema de arquivos realize duas funções que são extremamente importantes: a montagem e desmontagem dos blocos lógicos, além de sua localização, pois quem informa ao sistema operacional a quantidade de bytes do arquivo ou do bloco é o

próprio sistema de arquivos. Os parâmetros enviados ao sistema operacional pelo processo responsável pela chamada de sistema para ler o arquivo são (OLIVEIRA; CARISSIMI; TOSCANI, 2010):

READ (Handle, Var, NumBytes)

As instruções acima representam, respectivamente, o número do descritor do arquivo contido na TAAP, como resposta da chamada de sistema open, o Handle.

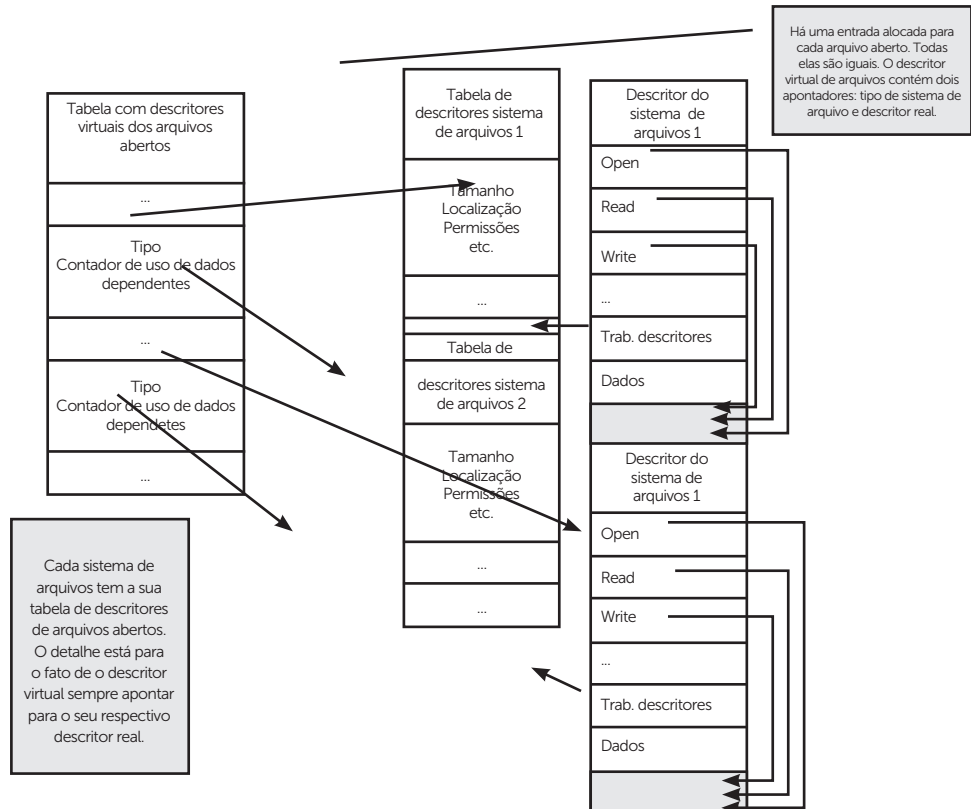
Var, que representa a palavra variável, indica a localização ou posição de endereçamento no processo em que serão alocados os dados de leitura o nome do arquivo, e NumBytes é a quantidade de bytes do bloco. Isso porque, para o sistema operacional, um arquivo nada mais é do que uma sequência de blocos organizados logicamente. Sendo assim, a responsabilidade de montar e desmontar os blocos é do sistema de arquivos.

E, quando estamos trabalhando com vários sistemas de arquivos, como é esse mecanismo de gerência e controle das operações básicas realizados por ele? Como é possível identificar, compartilhar e distribuir os blocos de bytes dos arquivos ordenadamente, facilitando ainda o seu acesso e localização, a considerar que os vários sistemas de arquivos estão em um único sistema operacional? Vejamos, a seguir, uma das soluções encontradas para tratar esses e outros aspectos do processamento e armazenamento de dados distribuídos:

Parte-se da premissa de que um sistema de arquivos pode administrar apenas uma partição e que há uma parte desse sistema que é independente de um dispositivo de armazenamento e outra que é dependente. Dessa forma, podemos inferir que, quando se trata de sistemas de arquivos, esses devem realizar ações específicas para cada tipo de meio físico de armazenamento. São considerados meios ou dispositivos de armazenamento CD, CD-ROM, pen drive, HD externo, disco rígido, entre outros.

Quando falamos de uma parte independente do sistema de arquivos, estamos trabalhando um conceito de virtualização, em que é implementada uma tabela que considera descritores virtuais, ou seja, denominada tabela com descritores virtuais dos arquivos abertos. Mas, afinal, o que isso significa? Veja na Figura 3.11 como é realizado esse mecanismo:

Figura 3.11 | Estrutura de dados para suportar múltiplos sistemas de arquivos



Fonte: Oliveira, Carissimi, Toscani (2010, p. 226)



Faça você mesmo

Assista ao vídeo disponível no *link*: <<https://www.youtube.com/watch?v=JxlAcsFQ-yl>>. (Acesso em: 26 ago. 2015). Você aprenderá algumas particularidades da virtualização de arquivos e como isso acontece e significa na prática. Bons estudos!



Pesquise mais

O artigo "OneDrive, Dropbox ou Google Drive: Qual serviço de armazenamento em nuvem é o ideal para você?" traz informações atualizadas sobre o gerenciamento de arquivos armazenados na nuvem. Disponível em: <<http://codigofonte.uol.com.br/artigos/onedrive-dropbox-ou-google-drive-qual-servico-de-armazenamento-em-nuvem-e-o-ideal-para-voce>>. Acesso em: 20 ago. 2015.

Sem medo de errar

Nessa atividade, você precisa apresentar para a microempresa uma solução de sistema de arquivo e como implementá-lo, desde que esse seja compatível com os sistemas operacionais Windows, Unix e Linux. Vamos, então, propor a implantação do sistema de arquivos de rede chamado NFS (*Network File System*). Suas funcionalidades incluem: melhoria de acessibilidade, oferecendo maior segurança, pois utiliza um protocolo chamado RPSEC_GSS, que implementa maior segurança, e essa é integrada à gestão de máquinas cliente e servidores. Além disso, ainda suporta aplicações de servidores clusterizados e redes geograficamente distribuídas, ou seja, de grande amplitude. Interage e integra serviços com o *Active Directory*. Esse sistema de arquivo é compatível com os sistemas operacionais mencionados e, para implementá-lo, será preciso:

1. Certificar-se de que estamos trabalhando com um ambiente predominantemente baseado em Unix, de forma que seja possível compartilhar arquivos NFS.
2. Verificar se temos as versões de Unix que executam sistemas NFS e Windows a partir da versão Windows Server 2012.
3. Instale os dois principais serviços do NFS, que são: Servidor NFS e Client NFS. O Servidor NFS pode ser instalado em uma máquina com Unix.
4. Seguir as instruções abaixo, de acordo com os dados do fornecedor.

Para instalar o sistema de arquivos de rede no servidor usando o Gerenciador do servidor:

1. Adicionar funções e para Assistente de recursos, no âmbito de funções de servidor, selecione o arquivo e serviços de armazenamento, se ele já não tiver sido instalado.
2. Sob arquivo e iSCSI serviços, selecione o Servidor de arquivos e servidor de NFS. Clique em Adicionar recursos para incluir recursos selecionados do NFS.
3. Clique em instalar para instalar os componentes NFS no servidor.

Para instalar o sistema de arquivos de rede no servidor usando o Windows PowerShell:

1. Inicie o Windows PowerShell. Clique no botão direito do mouse no ícone do PowerShell na barra de tarefas e selecione Executar como administrador.
2. Execute os seguintes comandos do Windows PowerShell:
3. PS C:\>Importação-módulo ServerManagerPS C:\>Add-WindowsFeature FS-NFS-ServicesPS C:\>Importação-módulo NFS



Atenção!

Siga as demais instruções do fornecedor para criar o compartilhamento de arquivos NFS. Verifique as especificações completas desse procedimento em: <<https://technet.microsoft.com/pt-br/library/JJ574143.aspx>>. Acesso em: 25 ago. 2015.



Lembre-se

Quando um controlador de domínio não é implantado, você pode usar um servidor de serviço de informação de rede (NIS) para fornecer informações de autenticação de usuário para o ambiente UNIX. Ou, se preferir, você pode usar a senha e grupo de arquivos que são armazenados no computador que está executando o serviço de mapeamento de nome de usuário. Disponível em: <<https://technet.microsoft.com/pt-br/library/JJ574143.aspx>>. Acesso em: 25 ago. 2015.

Avançando na prática

Pratique mais	
Instrução Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.	
Introdução à implementação do sistema de arquivos. Virtualização do sistema de arquivos e registro	
1. Competência de fundamentos de área	O aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional, bem como ter conhecimento sobre como se dá o compartilhamento de recursos e a sua gerência.
2. Objetivos de aprendizagem	Conhecer o que é um arquivo, sua estrutura e atributos. Conhecer como é o mecanismo de organização e hierarquia para gerenciar arquivos e diretórios. Conhecer e saber como é a implantação do sistema de arquivos e ainda identificar outros mecanismos, como virtualização e o impacto dessa para o sistema operacional. Conhecer as formas de fazer a proteção e garantir a segurança do sistema de arquivos.
3. Conteúdos relacionados	Introdução à implementação do sistema de arquivos. Virtualização do sistema de arquivos e registro.
4. Descrição da SP	Descreva os processos de implementação de arquivos de forma virtualizada em Linux, como outra opção para os clientes da microempresa do setor de alimentos. Pode utilizar procedimentos recomendados por fornecedores de sistemas operacionais como base para os seus trabalhos.

5. Resolução da SP

Vamos seguir o passo a passo recomendado por um fornecedor, em que há a descrição de como é o mecanismo de funcionamento de um sistema de arquivos virtual. Nesse caso, o S. O em questão é o Linux.

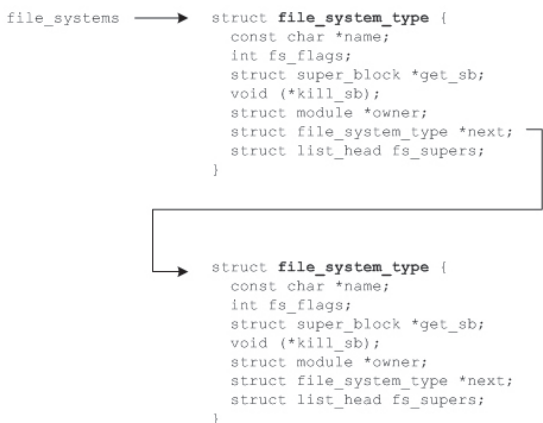
"Camada do Sistema de Arquivo Virtual

O VFS atua como o nível raiz da interface do sistema de arquivos. Ele mantém o rastreo dos sistemas de arquivos suportados atualmente, bem como dos sistemas de arquivos que estão sendo montados no momento. Os sistemas de arquivos podem ser incluídos ou removidos dinamicamente do Linux usando um conjunto de funções de registro. O kernel mantém uma lista dos sistemas de arquivos suportados atualmente, que pode ser visualizada a partir do espaço de usuário através do sistema de arquivos `/proc`. Esse arquivo virtual também mostra os dispositivos atualmente associados aos sistemas de arquivos. Para incluir um novo sistema de arquivos no Linux, `register_filesystem` é chamado. Ele obtém um único argumento definindo a referência para uma estrutura do sistema de arquivos (`file_system_type`), que define o nome do sistema de arquivos, um conjunto de atributos e duas funções `superblock`.

Um sistema de arquivos também pode ter seu registro removido.

Registrar um novo sistema de arquivos coloca o novo sistema de arquivos e suas informações pertinentes em uma lista `file_systems` (consulte a Figura 3.12 e `linux/include/linux/mount.h`). Essa lista define os sistemas de arquivos que podem ter suporte. É possível visualizar essa lista digitando `cat /proc/filesystems` na linha de comandos.

Figura 3.12 - Sistemas de Arquivos Registrados com o Kernel



```

file_systems → struct file_system_type {
    const char *name;
    int fs_flags;
    struct super_block *get_sb;
    void (*kill_sb);
    struct module *owner;
    struct file_system_type *next;
    struct list_head fs_supers;
}

→ struct file_system_type {
    const char *name;
    int fs_flags;
    struct super_block *get_sb;
    void (*kill_sb);
    struct module *owner;
    struct file_system_type *next;
    struct list_head fs_supers;
}

```

Outra estrutura mantida no VFS é o sistema de arquivos montado (consulte a Figura 3.12). Ela fornece os sistemas de arquivos que estão montados atualmente (consulte `linux/include/linux/fs.h`). Ela é vinculada à estrutura `superblock`, da qual eu falarei a seguir.

Figura 3.13 - A Lista de Sistemas de Arquivos Montados

```
current->namespace->list → struct vfsmount {
    struct list_head mnt_hash;
    struct vfsmount *mnt_parent;
    struct dentry *mnt_mountpoint;
    struct dentry *mnt_root;
    struct super_block *mnt_sb;
    struct list_head mnt_mounts;
    struct list_head mnt_child;
    atomic_t mnt_count;
    int mnt_flags;
    char *mnt_devname;
    struct list_head mnt_list;
}
```

mounted filesystem list

Fonte: Disponível em: <<https://www.ibm.com/developerworks/br/library/l-linux-filesystem/>>. Acesso em: 25 ago. 2015.



Lembre-se

Para incluir um novo sistema de arquivos, basta fornecer um descritor do sistema de arquivos no formato padrão definido pelo sistema operacional. Esse formato define a interface padrão a ser suportada, isso é, quais são as rotinas que devem ser implementadas e quais os seus parâmetros e valores de retorno. [...] Por exemplo, a interface padrão inclui uma rotina “inicializar”, que é sempre a primeira chamada quando um novo sistema de arquivos, pode, por exemplo, inicializar a sua tabela de descritores de arquivos específica (OLIVEIRA; CARISSIMI; TOSCANI, 2010, p. 227).



Faça você mesmo

Leia o artigo na íntegra sobre implementação de sistemas de arquivos em Linux. Disponível em: <<https://www.ibm.com/developerworks/br/library/l-linux-filesystem/>>. Acesso em: 25 ago. 2015.

Faça valer a pena

1. Assinale a alternativa que contém a especificação das afirmações verdadeiras:

- I – O conteúdo do arquivo em caso de interrupções permanecerá intacto.
- II – O conteúdo do descritor de arquivos se altera em caso de interrupção.

III – O ideal é manter o descritor na mesma partição em que se encontra o arquivo.

- a) I, II e III.
- b) I e II.
- c) I e III.
- d) II e III.
- e) I e III.

2. Complete a frase com uma das opções das alternativas a seguir:

Uma _____, também chamada de _____, é responsável por manter atualizadas as informações dos arquivos abertos e isto ocorre para todos os processos do sistema, em função de um arquivo ser acessado por vários processos ao mesmo tempo.

- a) tabela de descritores de arquivos abertos / TDAA
- b) tabela de registros / TDR
- c) tabela de dados de processos temporários
- d) tabela de processos abertos
- e) tabela de arquivos abertos por processo / TAAP

3. Explique o que é a tabela de arquivos abertos por processo.

4. Descreva como é o mecanismo de funcionamento do sistema de arquivos virtual do Linux.

5. As definições abaixo são verdadeiras ou falsas? Assinale a alternativa correspondente:

I – Há uma entrada alocada para cada arquivo aberto na tabela de descritores de arquivos virtuais.
II – Todas as tabelas de descritores são iguais. O descritor virtual de arquivos não precisa de apontadores de sistema de arquivo e de descritor real.
III – Cada sistema de arquivos tem a sua tabela de descritores de arquivos abertos. O detalhe está para o fato de o descritor virtual sempre apontar para o seu respectivo descritor real.

- a) V, V, V.
- b) F, F, F.
- c) V, V, F.

- d) V, F, V.
- e) F, V, V.

6. Assinale a alternativa que contém os parâmetros enviados ao sistema operacional. Lembre-se da estrutura: processo → chamada de sistema → leitura de arquivos:

- a) WRITE (Handle, Var, NumBytes)
- b) READ (NumBytes)
- c) READ (Handle)
- d) WRITE (Handle)
- e) READ (Handle, Var, NumBytes)

7. Complete a frase:

"[...] há uma pesquisa que considera basicamente um índice do arquivo que se associa à tabela de descritores de arquivos abertos (TDAA) para verificar qual arquivo se encontra aberto. Essa busca utiliza os números de partição e de endereço do descritor liberados pelo procedimento de _____".

- a) read
- b) lookup.
- c) write.
- d) TDAA.
- e) TAAP.

Seção 3.4

Introdução à segurança e mecanismos de proteção

Diálogo aberto

Nesta seção de estudos, continuamos com o cenário da microempresa de alimentos que está em expansão e precisa rever algumas questões de seu parque tecnológico. Compete a esse serviço apresentar as instruções de execução de um plano de segurança da informação.

Para realizar essa tarefa, uma das etapas inclui a instalação e configuração de um sistema de segurança de *firewall*, ou seja, que realize um filtro das informações e solicitações de tráfego de dados na rede da empresa. Você precisa, então, evidenciar qual o procedimento para a instalação e configuração dessa interface de *software*.

Será apresentado, no entanto, um *software* que é voltado para a gestão do *firewall* em sistemas operacionais Linux, tendo em mente que algumas das aplicações locais são controladas por *softwares* de gerenciamento de diretórios compatíveis com esse e também com o sistema operacional Windows.

Temos, então, diversos sistemas operacionais e serviços de *software* em questão, o que exige muitos recursos e configuração de regras de segurança de informação que de fato possam privilegiar os dois ambientes.

Para resolver essas questões, serão apresentados, nesta aula, os principais componentes do plano de segurança da informação, no que tange à/a:

- autenticação de usuários;
- criptografias;
- definição de senhas;
- mecanismos de autenticação biométricos;
- sistemas de segurança de recursos básicos, tais como: recursos de CPU e memória;

- tipos de ameaças.

Ao sistema operacional, no que tange à gestão e alocação dos recursos, o gerenciamento do sistema de arquivos, a escolha de aplicações que permitam otimizar os recursos e mapeá-los, requer a implementação de regras de segurança da informação e que essas sejam seguidas. Para tal, os conceitos fundamentais apresentados acima serão abordados com maior profundidade, a fim de proporcionar, além do conhecimento, o desenvolvimento de criticidade quanto a esses procedimentos. Desde já, bons estudos e práticas a você!

Não pode faltar

Em um mercado altamente competitivo, globalizado e tecnológico, algumas preocupações acerca da segurança da informação têm ganhado espaço tanto nas academias quanto no mundo corporativo. É preciso estabelecer os processos relacionados à segurança da informação e fazer com que se tornem uma prática nesses ambientes.

As configurações do sistema operacional precisam ser consideradas, pois nem sempre a negação de um determinado serviço pode significar que esteja em execução um procedimento relacionado à segurança da informação, mas, sim, que não há recurso disponível para realizar aquele determinado processo. Nesse contexto, há alguns pontos de atenção que precisamos abordar, que colaboram para com a elaboração de uma política ou plano de segurança de informação. São elementos básicos para esse procedimento as seguintes verificações, sob o ponto de vista do sistema operacional e suas configurações, conforme indica o Quadro 3.2:

Quadro 3.2 | Verificações de segurança realizadas pelo sistema operacional

Tipo de verificação	Descrição
Autenticação de usuário	Os processos estão associados aos usuários que os criaram no sistema. Por esse motivo, é necessário realizar esse procedimento de conferência que analisa o que ele pode e não pode fazer no sistema operacional e arquivos, além dos acessos que pode ter. Deve existir uma pré-definição das permissões desse usuário.
Proteção de recursos básicos	A segunda verificação de permissão deve identificar se o usuário pode acessar um determinado recurso ou suas configurações. São recursos: processador, memória, dispositivos de entrada e saída, listas de controle de acessos previamente registradas.
Tipos de ameaças	Trata-se da identificação de possíveis vírus, aplicativos e softwares maliciosos, bem como técnicas aplicadas que possam representar um risco a softwares, hardwares e, consequentemente, às informações.

Avaliação de segurança do sistema operacional	A Agência de Segurança Nacional dos Estados Unidos, em 1983, lançou um documento que contém os critérios confiáveis de avaliação de sistemas computacionais do Departamento de Defesa. Esse tem sido chamado de "Livro Laranja", subdividido em A, B, C e D.
Criptografia	Algoritmos aplicados à codificação e decodificação de informações em sistemas computacionais.

Fonte: Adaptado de Stuart (2011)

Quando se trata da autenticação dos usuários, é importante estabelecer suas respectivas senhas e *logins* de acesso, ou nomes que os identifiquem e os diferenciem. Como já é de conhecimento comum, uma senha de usuário não deve ser compartilhada em hipótese alguma com outro. Isso pode comprometer a segurança tanto das ações que ele tem permissão para executar junto ao sistema quanto a integridade dos arquivos e aplicações às quais tem acesso.

Outro fator importante a ser definido quanto às senhas de acesso dos usuários é o grau ou nível de segurança que oferecem. Então, critérios como não aceitar senhas repetidas, ou que estejam associadas a datas comuns aos usuários, devem ser configurados como regras a verificar no sistema também. Essas senhas devem estar protegidas do acesso indevido no sistema. Precisam ser criptografadas, ou seja, codificadas e decodificadas a partir de chaves que são correspondentes e testadas, que pertencem apenas ao próprio usuário ou ao administrador do sistema, que terão como verificar essa informação. Jamais deverá deixar em texto legível e disponível de visualização para demais usuários.

Mesmo utilizando as técnicas de criptografia, as senhas podem ser descobertas, caso o algoritmo seja descoberto. Para minimizar essa possibilidade, é possível utilizar o que se chama criptografia de sentido único, em que a senha informada pelo usuário no momento do login é comparada àquela que está armazenada e, no caso de não ser correspondente, o acesso será negado. Uma das técnicas mais utilizadas para criptografar dados é conhecida como função hash. Essa é de sentido único, como mencionado anteriormente. Leia mais sobre essa função de criptografia em: <<http://www.techtudo.com.br/artigos/noticia/2012/07/o-que-e-hash.html>>. (Acesso em: 2 set. 2015). Veremos algumas técnicas de criptografia na sequência deste material. Siga em frente!

No quesito autenticação, ainda são inseridas verificações de retorno de chamada, que têm como base o conhecimento prévio de onde ou local, estação de trabalho, ou número, que será realizada a solicitação de acesso.



Refleta

Ao acessarmos um sistema de modo remoto, especialmente por linhas de

telefone comuns, é frequente adicionarmos outra medida de segurança. O sistema no qual o usuário está fazendo *login* deve saber onde se espera que o usuário esteja. Suponha que se espere que o usuário ligue do número 555-1234. O usuário inicia a conexão, mas, antes do processo de *login* ser concluído, ambos interrompem a chamada e o sistema a retorna para o usuário no número 555- 1234. Se um impostor tentar se conectar de um número diferente, a tentativa de *login* não será bem-sucedida (STUART, 2011, p. 579).

Outro modo de se fazer autenticação é o chamado desafio/resposta. Já aconteceu com você de digitar a senha correta e o sistema informar que a senha está errada e você a digita novamente? Cuidado, pode ser que você tenha sofrido o que chamamos de ataque por reprodução ou repetição, em que, no momento em que você insere novamente a senha, essa é capturada indevidamente pelo *software*. Para impedir que isso aconteça, o ideal é que o sistema não esteja autorizado a realizar esse acesso ou receber essa segunda autenticação, exigindo que seja gerada uma nova senha, através do gerador sincronizado de números aleatórios. O intuito é aumentar a dificuldade de obtenção dessa senha no ato do *login*. Um exemplo, conhecido como desafio/resposta, consiste em o sistema solicitar, em um segundo acesso, que o usuário informe um número ou mensagem que foi gerado, de forma que, se essa informação for correta, ou seja, inserida por um usuário e não fruto de uma comparação de um *software* malicioso, com isso, o *login* possa se realizar (STUART, 2011).

Outro mecanismo muito utilizado, principalmente por bancos, ou mesmo níveis hierárquicos em empresas em que o acesso é restrito a determinadas informações, o usuário pode utilizar uma senha única para fazer um acesso, ou seja, cadastrada e inutilizada após esse acesso.

Além disso, a técnica biométrica também tem ganhado espaço no mercado, apesar de requerer mecanismos tecnológicos mais aprimorados do que os algoritmos e técnicas mencionados. A mais evasiva, como cita Stuart (2011), é a solicitação do DNA para autenticação de acesso, porém não muito utilizada e apenas em casos extremamente específicos.

A mais conhecida das técnicas de autenticação biométrica é através da digital, em que é capturada, digitalizada e a cada *login* ou tentativa de acesso é solicitado ao usuário para que seja comparada àquela que foi armazenada. Nesse mesmo contexto, a leitura da íris também é uma técnica de biometria bastante utilizada.

Mas além da autenticação, como vimos no Quadro 3.2, temos ainda as verificações que visam proteger do acesso indevido, os recursos básicos do computador. Nesse caso, o Quadro 3.3 traz o respectivo recurso e a característica relacionada à sua proteção:

Quadro 3.3 | Recurso e sua característica de proteção

Tipo de verificação	Descrição
CPU (Unidade Central de Processamento)	O acesso a CPU é controlado basicamente pelo código do escalonador e do contexto (hardware e software). Por esse motivo, o acesso deve ser restrito apenas a alguns tipos de recursos, como as instruções de software e a ação dos registradores.
Memória	Já o acesso de verificação do recurso de memória pode ocorrer de forma irrestrita através da CPU pelos registradores de base e de limite.
Listas de controle de acesso	Outra forma de se realizar a proteção dos recursos básicos é através de uma lista de controle de acesso, que também pode ser chamada de ACL (acrônimo de Access Control List), em que são combinadas as permissões, de acordo com cada um dos membros da lista, criando o conceito de identidade-permissão, que se associam a cada usuário e tipo de recurso.
Aptidões	Relacionamento entre pares de recursos- permissão ao invés de permissão por usuário.

Fonte: Adaptado de Stuart (2011)

Algumas instruções servem para interromper processos em execução na CPU, porém essa ação pode ser executada apenas por processos que contêm a respectiva prioridade ou privilégio para tal. Isso também se aplica aos processos que controlam as operações de entrada e saída. Então, por esse motivo, as verificações acontecem de acordo com a troca de contexto e o que foi previsto em termos de permissões e programado para aceitar, liberar ou não, que a ação do processo seja executada.

Normalmente, no contexto de *software*, há a criptografia da senha e o devido tratamento para alguns possíveis erros previamente mapeados, além das respectivas exceções. De forma geral, no contexto de *hardware*, a CPU não tem essa função de reconhecimento do usuário que criou o processo. Por esse motivo é que as verificações ocorrem no contexto do *software*.

O que acontece com relação à memória é a definição desse par de registradores (base e limite) no momento em que o processo é criado, e o sistema operacional já configura essas informações a ele. Recomenda-se que as áreas da memória que contêm arquivos executáveis recebam permissões apenas de leitura. No caso do compartilhamento de memória por múltiplos processos, é preciso que estejam configuradas as entradas dos endereços virtuais responsáveis pelo mapeamento das áreas de memória, sendo que os registradores é que controlarão o acesso às tabelas de segmentos e páginas.

No contexto da proteção de recursos básicos, ainda podemos estabelecer para arquivos a condição de esse ser somente leitura, como também acontece com os acessos aos dispositivos de entrada e saída. Os arquivos dividem-se em permissões

para o próprio usuário ou para um grupo. Nesse sentido, o usuário se classifica como: individual, os membros do grupo e demais usuários. Leve-se em consideração que, para cada tipo de usuário, é possível estabelecer permissões distintas entre gravação, leitura e escrita, ou mesmo atualização e exclusão de arquivos.

As permissões podem ser divididas em três campos de bits, conforme explica Stuart (2011), sendo um bit para indicar cada tipo de permissão contido na palavra de proteção. Dessa forma, há a comparação entre o tipo de solicitação do usuário e a permissão que foi estabelecida.



Assimile

Como é tratado o problema do controle de acessos? Veja o que Stuart (2011) aborda como exemplo de listas de controle de acesso. Observe:

Suponha cinco usuários chamados Louis, Sandra, Philip, Susan e Stephen, todos envolvidos em um curso identificado como CS342. Se Louis é o professor, Sandra é assistente e os demais são alunos, podemos utilizar uma ACL de modo que:

(Louis, LG) (Sandra, LG) (Philip, L) (Susan, L) (Stephen, L) (*, -)

para fornecer acesso de leitura e gravação ao professor e à assistente, acesso de leitura aos alunos e nenhum acesso a qualquer outra pessoa. Utilizamos o asterisco (*) como um coringa, que inclui qualquer usuário.

Outra ACL que realizaria o mesmo é:

(Louis, LG) (Sandra, LG) (CS342, L) (*,-)

Se todos os alunos estiverem no grupo Cs342 (STUART, 2011, p. 587).

Além desses mencionados, existe ainda o conceito de aptidão. Aptidão consiste na definição de acesso, de forma diferenciada através do relacionamento entre pares de recursos- permissão de um usuário ou grupo.



Exemplificando

Observe o exemplo que Stuart (2011) considera quando trabalha os conceitos de aptidões ou aprimoramento dos acessos e permissões:

Suponha que o arquivo que estamos compartilhando chama-se atribuições. A aptidão desse arquivo na lista do usuário Louis seria (atribuições, LG) e a

na lista do grupo CS342 seria (atribuições, L). Com essa entrada na lista de aptidões do grupo, não é necessária uma entrada similar nas listas dos alunos Philip, Susan e Stephen (STUART, 2011, p. 588).

Tendo em vista que, além desses mencionados, você ainda pode conhecer outros mecanismos de controle da segurança da informação, como os mencionados Livro Laranja e métodos de criptografia. Acesse os *links* a seguir e aprenda mais!



Pesquise mais

Você está pronto para as ameaças atuais de segurança? Esse é o título de um artigo publicado pela IBM, em que questões sobre a segurança dos dados e informações é tratada considerando as ameaças da atualidade. Disponível em: <http://www-01.ibm.com/software/br/security/?cmp&iio=bsec&cmp=security&ct=20c40401ew&cr=google&cm=k&csot=-&ccy=-&cpb=-&cd=-&ck=seguran%C3%A7a_em_sistemas_operacionais&cs=broadmatch&csr=alwayson&cn=ferramentassegurancaemti>. Acesso em: 1 set. 2015.



Faça você mesmo

Assista à palestra que aconteceu em uma das edições da Campus Party. Disponível em: <<https://www.youtube.com/watch?v=Q3QtALPY-Ms>>. Acesso em: 2 set. 2015.

Sem medo de errar

A microempresa utiliza em seu servidor o sistema operacional Linux. Por esse motivo, para atender à solicitação de uma das tarefas voltadas ao plano de segurança da informação da microempresa de alimentos, como sugerido, será preciso instalar um software conhecido como Gufw (*Graphical Firewall Uncomplicated*), que é uma interface gráfica voltada para a configuração do recurso de *firewall*. Esse tem por função delimitar o que é permitido trafegar pela porta de entrada de rede da empresa, quais limitações, restrições envolvidas e configurar essas regras. Abaixo, estão algumas configurações recomendadas pelo fornecedor do *software*:

- *Download* do Gufw: <<http://gufw.org/>>. Acesso em: 1 set. 20105.
- 2. Siga os passos do fornecedor e utilize o Synaptic:

1. Execute o Synaptic através do menu Sistema ► Administração ► Gerenciador de Pacotes Synaptic.

2. Navegue pela lista de pacotes disponíveis e dê clique duplo sobre o pacote gufw. Uma janela com a mensagem Marcar mudanças adicionais requeridas? deve aparecer. Clique no botão Marcar.

3. Clique no botão Aplicar, na barra de ferramentas do Synaptic. Uma janela de resumo deve aparecer. Clique no botão Aplicar dessa janela. Aguarde o pacote ser baixado e instalado. Terminal: `sudo apt-get install gufw`

4. Usando o Gufw: para acessar Gufw, vá para Sistema ► Administração ► Configuração do *firewall*.

5. Habilite o *Firewall*.

6. Adicionando regras: para adicionar regras, basta clicar no botão Adicionar botão e uma nova janela aparecerá. É recomendável verificar também a documentação internacional do UFW.

7. As regras são: Permitir, Negar, Rejeitar e Limite.

- Permitir: o sistema vai permitir o tráfego de entrada para uma porta.
- Deny: o sistema irá negar o tráfego de entrada para uma porta.
- Rejeitar: o sistema irá negar o tráfego de entrada para uma porta e informará o requerente para o sistema de ligação que tenha sido rejeitada.

8. Limite: o sistema irá recusar conexões se um endereço IP tentou iniciar 6 ou mais conexões nos últimos 30 segundos.

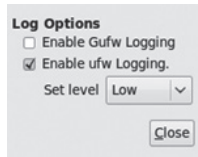
9. Pré-configurações: já deixe ativadas algumas opções para controlar as opções de *firewall* para programas e serviços comuns. Leia mais na documentação internacional do UFW.

10. Simples: nem todas as configurações do programa estão disponíveis no Gufw, mas ainda podemos adicionar regras para usarem o Simples guia. Leia mais na documentação internacional do UFW.

11. Advanced: às vezes, queremos configurar o acesso baseado em IPs específicos ou intervalos de IP, portanto, usamos o Advanced guia. Leia mais na documentação internacional do UFW.

12. Preferências: há apenas duas preferências disponíveis para o Gufw, e

pode ser controlado a partir de Editar ► Preferências.



13. Aqui se pode controlar o registo para ufw e para Gufw. O padrão é permitir o registo para ufw e desativar o registo para Gufw. Fonte: Disponível em: <<http://wiki.ubuntu-br.org/Gufw>>. Acesso em: 1 set. 2015.



Atenção!

Assista ao vídeo que pode ajudar a configurar os sistemas de *Firewell* do Ubuntu-Linux, utilizando o Gufw. Disponível em: <<https://www.youtube.com/watch?v=HWZG9FAJdOE>>. Acesso em: 1 set. 2015.



Lembre-se

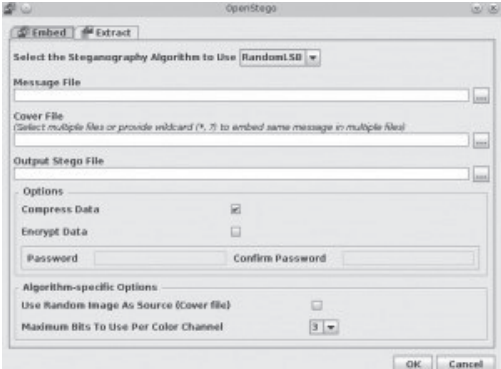
Os comprometimentos de segurança do sistema podem decorrer de várias fontes. Em algumas situações, erros de aplicações ou do SO podem permitir que usuários façam o que não gostaríamos. De modo similar, equívocos administrativos também podem permitir acessos não desejados aos recursos. Também, podemos nos deparar com *softwares* escritos para comprometer intencionalmente a segurança de um sistema (STUART, 2011, p. 588).

Avançando na prática

Pratique mais

Instrução

Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.

Introdução à segurança e mecanismos de proteção	
1. Competência de fundamentos de área	O aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional, bem como ter conhecimento sobre como se dá o compartilhamento de recursos e a sua gerência.
2. Objetivos de aprendizagem	Conhecer o que é um arquivo, sua estrutura e atributos. Conhecer como é o mecanismo de organização e hierarquia para gerenciar arquivos e diretórios. Conhecer e saber como é a implantação do sistema de arquivos e ainda identificar outros mecanismos, como virtualização e o impacto dessa para o sistema operacional. Conhecer as formas de fazer a proteção e garantir a segurança do sistema de arquivos.
3. Conteúdos relacionados	Introdução à segurança e mecanismos de proteção.
4. Descrição da SP	A fim de ampliar os mecanismos de segurança da informação, utilize os métodos de esteganografia para implantar na microempresa de alimentos. Dê um exemplo. Siga as orientações do fornecedor!
5. Resolução da SP	<p>Métodos de esteganografia: a esteganografia possui dois métodos principais:</p> <p>Método de inserção: o método de inserção envolve inserir conteúdo adicional no arquivo, como mensagens escondidas, marcadores de arquivo etc.</p> <p>Método de substituição: o método de substituição troca ou substitui os bits existentes em um dado arquivo, com isso fica imperceptível para o usuário visualizar ou identificar a alteração.</p> <p>O LSB: o método de substituição utiliza o LSB (<i>Least Signification Bit</i>) ou Bit menos significativo, que altera o bit mais significativo (que está mais à direita), alterando o bit 0 para 1 ou vice-versa.</p> <p>Podemos exemplificar:</p> <p>1 0 1 0 0 0 1 1</p> <p>Qual seria o LSB?</p> <p>1 0 1 0 0 0 1 1</p> <p>O LSB seria o bit em negrito mais a direita.</p> <p>OpenStego: O OpenStego é uma ferramenta de esteganografia capaz de esconder arquivos e utiliza o algoritmo LSB ou Random LSB para efetuar a esteganografia.</p> 

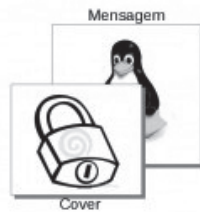
Para se efetuar a esteganografia é simples. No OpenStego, há 3 campos:

Message File: onde deverá ser informado o arquivo que ficará escondido.

CoverFile: o arquivo "mascarado", arquivo que será visualizado pelo usuário.

Output Stego File: será o arquivo de saída (arquivo que será exportado)

Na figura abaixo, é possível compreender melhor como funciona a ferramenta.

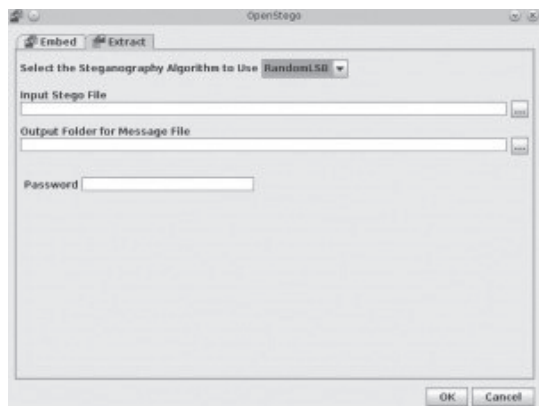


O usuário irá visualizar o arquivo Cover (cadeado). O conteúdo que está escondido (no caso a Mensagem) é a foto do nosso aclamado pinguim. Nos campos inferiores, é possível informar uma senha para o arquivo "esteganografado".



Exportar o arquivos

A imagem a seguir é a guia onde é realizada a extração do arquivo "esteganografado".



Para exportar o arquivo, selecione o arquivo no campo Input Stego File e no campo abaixo Output Folder for Message File informe a pasta que será extraído o arquivo. Caso o arquivo possua senha, insira no campo Password. Disponível em: <<http://sejalivre.org/esteganografia/>>. Acesso em: 1 set. 2015.



Lembre-se

A esteganografia esconde informações em texto legível. Os lugares em que essas informações podem ser ocultas incluem imagens, arquivos de áudio e vídeo, programas executáveis e até mesmo arquivos de texto.[...] Uma imagem em escala de cinza de 1024 x 768 pixels pode conter uma mensagem de até 96 KB (STUART, 2011, p. 592).



Faça você mesmo

Entenda e aprenda o que é Esteganografia. Artigo disponível em: <<http://sejalivre.org/privacidade-x-criptografia/>>. Acesso em: 1 set. 2015.

Faça valer a pena

1. Complete com uma das opções disponíveis na alternativa:

"_____, consiste em o sistema solicitar, em um segundo acesso, que o usuário informe um número ou mensagem que foi gerado, de forma que, se essa informação for correta, ou seja, inserida por um usuário e não fruto de uma comparação de um software malicioso, o login possa se realizar".

O procedimento acima refere-se à(ao):

- a) *Login*.
- b) Criptografia.
- c) Desafio/resposta.
- d) Aptidão.
- e) Lista de controle.

2. Analise as afirmações e assinale a alternativa correspondente:

I – Quando se trata da autenticação dos usuários, é importante estabelecer suas respectivas senhas e logins de acesso, ou nomes que os identifiquem e diferenciem.

II – O acesso a CPU é controlado basicamente pelo código do escalonador e do contexto (*hardware* e *software*).

III – Os arquivos dividem-se em permissões para o próprio usuário ou

para um grupo. Nesse sentido, o usuário se classifica como: individual, os membros do grupo e demais usuários.

- a) Autenticação / proteção de recursos básicos/ proteção de arquivos básicos.
- b) Autenticação / autenticação / ameaças.
- c) Ameaças / autenticação / ameaças.
- d) Proteção de recursos / criptografia / ameaças.
- e) Autenticação / criptografia / ameaças.

3. É correto o que se afirma em:

I – Gufw (*Graphical Firewall Uncomplicated*) é uma interface gráfica do Linux para configuração de *Firewall*.

II – Gufw (*Graphical Firewall Uncomplicated*) é uma interface gráfica própria do Windows.

III – Gufw (*Graphical Firewall Uncomplicated*) é uma interface gráfica compatível com todos os sistemas operacionais do mercado.

- a) II e III.
- b) I.
- c) I, II e III.
- d) II.
- e) III.

4. Descreva o que é ACL.

5. Quais são os métodos de esteganografia existentes? Explique.

6. Assinale a alternativa que completa as lacunas da frase a seguir:

"Já o acesso de verificação do recurso de _____ pode ocorrer de forma irrestrita através da CPU pelos registradores de base e de limite".

- a) processador
- b) registrador
- c) criptografia
- d) defesa
- e) memória

7. Analise a tabela e assinale a alternativa que corresponde aos conceitos apresentados:

Tipo de verificação	Tipo de verificação
I. Autenticação de usuário	() Verificação de permissão para acesso a: processador, memória, dispositivos de entrada e saída, listas de controle de acessos previamente registradas.
II. Proteção de recursos básicos	() Deve existir uma pré-definição das permissões desse usuário.
III. Tipos de ameaças	() Cavalo de troia, vírus, worm e malware

A sequência correspondente ao conceito apresentado está na alternativa:

- a) I, II e III.
- b) III, II e I.
- c) II, I e III.
- d) II, III e I.
- e) III, I e II.

Referências

DEITEL, H. M.; DEITEL P. J.; CHOFFNES, D. R. **Sistemas operacionais**. 3. ed. São Paulo: Prentice Hall, 2005.

MACHADO, Francis B.; MAIA, Luiz P. **Arquitetura de sistemas operacionais**. 5. ed. Rio de Janeiro: LTC, 2013.

OLIVEIRA, Romulo Silva de; CARISSIMI, Alexandre da S.; TOSCANI, Simão S. **Sistemas operacionais**. Porto Alegre: Bookman, 2010.

SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGME, Greg. **Sistemas operacionais: sistemas e aplicações**. 6. ed. Rio de Janeiro: Elsevier, 2004.

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

TANEMBAUM, Andrew S. **Sistemas operacionais modernos**. 3. ed. São Paulo: Pearson Prentice Hall, 2009.

GERENCIAMENTO DE DISPOSITIVOS

Convite ao estudo

Olá, aluno! Vamos continuar com os estudos e conhecer mais alguns aspectos fundamentais para o bom funcionamento dos sistemas operacionais, com uma abordagem voltada para a gerência de dispositivos. Com eles e os demais materiais estudados, você poderá desenvolver uma competência que é considerada fundamental para esta unidade curricular: o aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional, bem como ter conhecimento sobre como se dá o compartilhamento de recursos e a sua gerência.

Além disso, buscamos também como objetivos específicos desta unidade de ensino:

- conhecer os principais conceitos, as características da gerência de memória e o que é, além da virtualização de arquivos;
- conhecer o que é *swapping*, como o sistema operacional gerencia essa tarefa e aloca os devidos recursos, a fim de atender às demandas de processamento e alocação;
- conhecer os principais conceitos sobre memória virtual e os procedimentos necessários a essa implantação;
- conhecer como se dá a gerência de processos advindos das solicitações dos dispositivos de entrada e saída.

A fim de aproximar os conteúdos e as teorias com a sua aplicação prática, para esta unidade vamos considerar o cenário de uma empresa, cujo foco está em

fazer pesquisa e desenvolvimento na área de tecnologias da informação. Com o intuito de diminuir o gargalo existente na indústria de telecomunicações, no que tange aos recursos de memória e processamento, vamos estudar os principais conceitos envolvidos nas arquiteturas utilizadas atualmente.

A partir disso, podemos propor alguns melhoramentos, voltando sempre à evolução para a aplicação em dispositivos móveis, uma vez que a tendência de ampla utilização, para as mais diversas áreas, é real e crescente.

Dedicação e curiosidade são seus ingredientes para esta jornada de estudos! Desde já, bons estudos e práticas.

Seção 4.1

Gerenciamento de memória: conceitos, tipos, características e virtualização

Diálogo aberto

De acordo com as tendências de mercado, você já deve ter observado que cada vez mais os dispositivos móveis ganham funcionalidade que vão além de simplesmente realizar chamadas telefônicas e enviar mensagens instantâneas.

A quantidade de memória suficiente nos dispositivos móveis se tornou um desafio para as empresas que precisam armazenar o sistema operacional, aplicativos, registros e, ainda, deixar espaço para que o usuário realize o *download* de outros serviços e aplicativos disponíveis para os seus respectivos dispositivos celulares. A memória é imprescindível e sua quantidade adequada é um desafio para as fabricantes.

Para esta aula, vamos trabalhar com um cenário em que uma empresa de pesquisa e desenvolvimento precisa realizar um levantamento de tipos de serviços que podem ser utilizados pelos usuários, que venham a minimizar os problemas com armazenamento em seus aparelhos móveis, e explicar a causa desse gargalo nesse mercado, uma vez que os princípios e técnicas são os mesmos. Através de pesquisas de boas práticas, vamos identificar quais ferramentas podem auxiliar na otimização dos espaços de memória em dispositivos móveis.

Já estudamos o que são os sistemas operacionais, suas responsabilidades e, principalmente, quais são os seus recursos de gerenciamento de processos, threads, sincronização e comunicação, bem como a realização do escalonamento que favorecerá a execução das tarefas. Compreendemos, também, como ocorre a gerência dos dispositivos.

Agora, precisamos entender como funciona um outro recurso administrado pelo sistema operacional e que é de extrema importância para a realização de suas atividades. Estamos falando de gerência de memória.

E, então, como funcionam os mecanismos de alocação de memória? Quais são os tipos de memória considerados nas arquiteturas de *hardware* atualmente?

As respostas dessas e de outras perguntas serão respondidas no decorrer dos

estudos desta unidade de ensino.

Para essa, ficamos com a distinção entre memória lógica e física, partições fixas e variáveis e o que é memória virtual.

Desde já, bons estudos e práticas a você!

Não pode faltar

Vamos iniciar os estudos sobre os recursos e mecanismos adotados pelo sistema operacional para realizar o gerenciamento de memória, no entanto é preciso esclarecer que aqui serão abordados conceitos gerais e não, exclusivamente, com o foco em uma arquitetura em especial. Por esse motivo, quando a pretensão for compreender o gerenciamento dos recursos de acordo com uma determinada arquitetura de computador, será preciso investigar quais são as suas especificidades. Isso agregará e facilitará na parametrização do sistema operacional que trabalhará.

Sendo assim, vamos iniciar, então, a descrição do que é memória, como ela está dividida e de que forma realiza a alocação de seus espaços para as mais variadas aplicações de *software*, informações e dados.

Quando vamos comprar um computador, precisamos nos atentar a alguns recursos, não é mesmo? Dentre eles, quais são os que você mais tende a observar? Claro que depende do que cada indivíduo necessita para realizar suas tarefas cotidianas, porém, quando vamos escolher um novo computador para comprar, queremos garantir que tenhamos basicamente: processamento eficiente e alta capacidade de armazenamento.

Com isso, acabamos por observar no ato da compra de uma máquina: qual é o tipo de processador, quanto tem de memória RAM? Qual a capacidade de armazenamento do *hard disk* – disco rígido (HD)? O sistema operacional que já vem licenciado e instalado na máquina? Vejamos um exemplo de quando iniciamos essa busca por algumas configurações comuns para a atualidade. Vamos escolher uma marca qualquer e especificar, de acordo com as respectivas famílias de processadores e capacidade de armazenamento que atendam às mais variadas tarefas que um usuário pode precisar. Sem contar os que utilizam para o trabalho mais recursos de máquina, no entanto configurações especiais requerem pesquisas mais rigorosas. Não se esqueça!



Refleta

"[...] o disco é um recurso compartilhado, sua utilização deverá ser

gerenciada unicamente pelo sistema operacional, evitando que a aplicação possa ter acesso a qualquer área do disco sem autorização, o que poderia comprometer a segurança e a integridade do sistema de arquivos” (MACHADO; MAIA, 2013).

Então, se você fosse comprar hoje um computador, qual configuração escolheria? Vou sugerir a seguinte: processador de última geração (família “i3, i5 ou i7”), memória RAM 4 gigabytes e disco rígido 500 GB são suficientes para você trabalhar com vários recursos e aplicações. Essa é apenas uma dica! Mas como essa escolha pode influenciar no meu desempenho profissional? Pois é, compreender o mecanismo de funcionamento desses recursos é importante para todos os tipos de usuários.

Nesse contexto, vamos estabelecer, de acordo com Stuart (2011), uma hierarquia utilizada pela maioria das arquiteturas de computadores quando se trata de gerenciamento de memória, pelo sistema operacional. Vejamos na Tabela 4.1, a seguir, como é que são considerados os tipos de armazenamento. Esses estão divididos em: permanente, secundário, principal, cache e registradores, hierarquizados de acordo com a sua eficiência, porte e valor de mercado. Observe:

Tabela 4.1 | Classificação de hierarquia de memória

Hierarquia de memória
Registradores
Cache
Memória principal
Memória secundária
Memória permanente

Fonte: Adaptado de Stuart (2011, p. 218)

Vamos iniciar a descrição da hierarquia especificada com a descrição dos registradores. Eles fazem parte da unidade central de processamento (CPU) e são rápidos, pois armazenam a informação ou instrução de processo que deverá ser executada imediatamente.

Já a memória cache é considerada mais lenta que os registradores, porém mais rápida que a memória principal. Ela existe, pois são necessários alguns mecanismos de armazenamento, para permitir que a CPU acesse e realize o processamento da instrução de processo mais rapidamente do que se tivesse que acessar determinado dado ou instrução diretamente em uma das outras memórias: principal, secundária ou permanente. Isso porque o modo como se realiza a busca da informação é diferenciado e cada uma possui uma função diante do contexto de gerenciamento de memória do sistema operacional, sendo que a memória cache ganha cada vez mais espaço para

armazenamento de acordo com as novas arquiteturas de computadores e em função do crescimento das aplicações e necessidades de *software* (STUART, 2011).

Seguindo a sequência apresentada na Tabela 4.1, temos a memória principal, conhecida também como de armazenamento primário, pois é nela que se concentra o gerenciamento de memória realizado pelo sistema operacional, ou seja, nela se concentram as aplicações que serão diretamente solicitadas pelo processador através do mecanismo de endereçamento. Não é considerada muito rápida, porém pode ser de capacidade de armazenamento relevante na ordem de alguns gigabytes. Como exemplo, podemos citar a memória RAM (*Random Access Memory*) e a ROM (*Ready Only Memory*).

A memória RAM permite apenas as operações de leitura e escrita. É volátil, ou seja, assim que o computador é desligado, ela perde todas as instruções ali carregadas e disponibilizadas para acesso rápido pelos registradores e processos de programas em execução. Na ROM, ficam gravadas as especificações do sistema e que, por padrão, não devem ser alteradas. Por esse motivo, a memória ROM fica disponível apenas para leitura.

Na sequência, temos a memória secundária, que é responsável pelo armazenamento permanente de dados e não permite endereçamento, como, por exemplo, os pen drives, os CDs e DVDs. É considerada mais lenta em função de não permitir endereçamento e ter de ser localizada por um sistema de arquivos, que é tratado de forma diferenciada e separado do sistema operacional.

Além dessas, para finalizar o quesito hierarquia, temos a memória permanente. Essa permite que se estabeleça em uma política de segurança da informação, que seja realizado o *backup* dos dados, por exemplo, como acontece em um *datacenter* ou centro de dados, em que há grande quantidade de dados e, se não tivermos uma política de segurança da informação bem definida, colocamos em risco as operações que dependem dessas informações.

A memória permanente garante o acesso à informação que foi trabalhada, sendo uma função externa às de responsabilidade do sistema operacional. Portanto, quando analisamos a hierarquia, temos a definição da mais rápida para as mais lentas, no entanto, com alta capacidade de armazenamento.

Agora que já identificamos como o sistema operacional reconhece a informação de cada tipo de memória segundo a sua hierarquia, vamos conhecer também os conceitos relacionados à memória lógica e física e em que essas se diferem.

Nesse sentido, temos a definição da memória lógica atrelada ao endereçamento realizado, ou seja, a busca da informação pelo processo se dá através do endereço lógico que uma instrução ocupa em memória. Estamos nos referindo às instruções de máquina em que há a manipulação de dados contidos nas variáveis indicadas nas aplicações (OLIVEIRA et. al, 2011).

Quando falamos em memória física, é a referência que se faz ao tipo de memória em que será armazenada a informação. No caso, podemos definir de acordo com a hierarquia especificada anteriormente qual delas será melhor utilizar e de acordo com o tipo de aplicação.

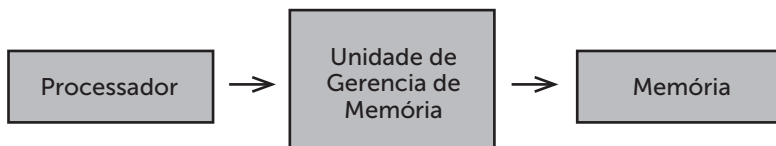
Temos, nesse contexto, a unidade de gerenciamento de memória, também conhecida como MMU (*Memory Management Unit*), cuja função é a de mapear os endereços lógicos em que estão as instruções nas memórias físicas.



Assimile

Observe na figura 4.1 qual é o fluxo que envolve o processador, a unidade de gerência de memória e o dado armazenado na memória:

Figura 4.1 | Fluxo de gerenciamento de recursos de memória e de processador

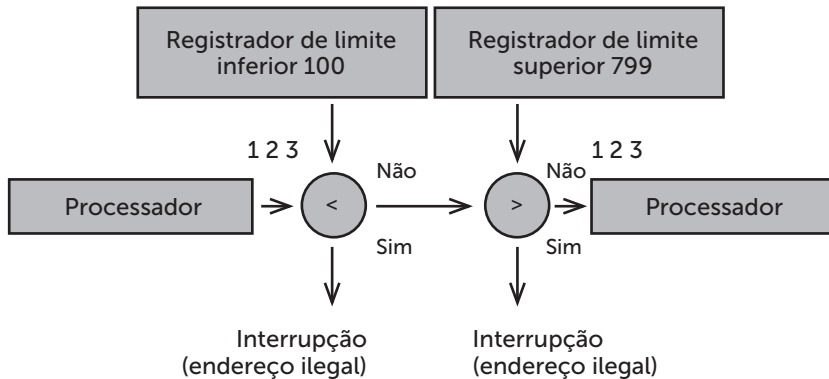


Fonte: Adaptado de Oliveira et al. (2010, p. 155)

A seguir, será descrito o processo de acesso ao endereço lógico que é gerado pelo processo e com isso a MMU direcionará o endereço lógico para o mesmo correspondente na memória física. Uma observação interessante é que, quando o próprio processo gera o endereço lógico e físico, eles são idênticos tanto na memória lógica, responsável pelos endereçamentos, quanto na memória física, que é correspondente à memória principal. Para realizar o gerenciamento de memória, a MMU de um processo de usuário, que é limitado, são utilizados dois registradores de limite.

De acordo com o processo, o conteúdo de um registrador de limite passa a ser o parâmetro para que o processo defina o espaço que esse deverá ocupar na memória lógica. Para o exemplo dado por Oliveira et al. (2011), foi utilizado o limite que varia entre 100 a 799 para a alocação em memória. Na Figura 4.2, temos exemplificado o mecanismo de proteção da memória com o uso de registradores de limite. Tudo o que estiver fora desse intervalo será desconsiderado. Observe a seguir:

Figura 4.2 | Proteção da memória com dois registradores de limite



Fonte: Adaptado de Oliveira et al.(2010, p. 155)



Pesquise mais

Vale a pena assistir à aula sobre tipos de memória e suas funções. Disponível em: <<https://www.youtube.com/watch?v=r-Ca80kDteA>>. Acesso em: 28 set. 2015.

É também de responsabilidade da gerência de memória manter em memória física ou principal a maior quantidade de processos residentes de forma que seja possível aproveitar ao máximo o compartilhamento de recursos, como, por exemplo, de processamento, ou seja, permitir que um número maior de processos leia as instruções e as execute (MACHADO; MAIA, 2013). As Figuras 4.1 e 4.2 visam demonstrar como ocorre a proteção das áreas de memória em que os processos trocam informações de endereçamento e alocação de forma protegida.

Além dos aspectos considerados, também precisamos aprender como acontece a divisão da memória em partes, processo esse também conhecido como particionamento, ou, ainda, alocação particionada.

Há três tipos de alocação de memória: alocação contígua simples, a técnica de *overlay* ou divisão da aplicação em módulos de forma a alocar de acordo com os espaços que cada um deles ocupará, e técnica de particionamento, como mencionado.

A alocação contígua simples é mais voltada à realidade dos primeiros sistemas operacionais que eram monoprogramáveis, sendo a memória principal dividida em duas grandes áreas: uma para alocar o sistema operacional e a outra para alocar as aplicações do usuário. Dessa forma, o desenvolvimento dessas aplicações de usuários tinha de respeitar as limitações da área de alocação das aplicações de usuário predeterminadas.

Já a técnica de *overlay* considera que, diante de uma aplicação, a divisão de módulos auxiliará na determinação do espaço de memória necessária a executar os módulos de forma independente. Isso quer dizer que é também no desenvolvimento da aplicação que deverá ser especificado o tamanho do módulo e como será a sua alocação de forma a permitir a execução de um módulo por vez.

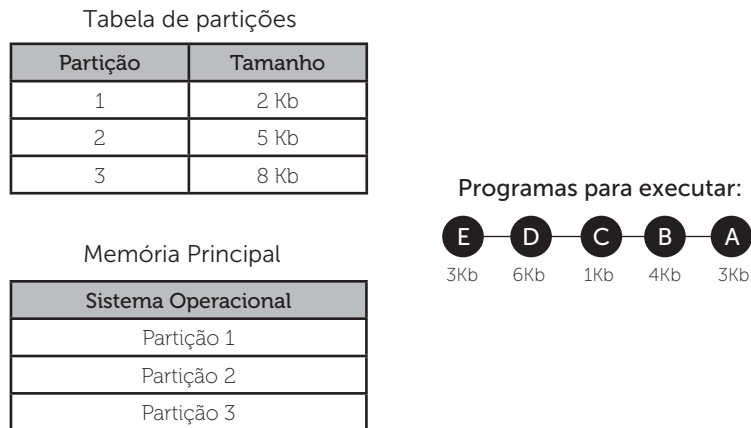
E a alocação particionada permite um maior aproveitamento dos espaços de memória principal. Nesse contexto, vamos conhecer os tipos de alocação de memória particionada existentes. São elas: estática ou dinâmica.



Exemplificando

Nos primeiros sistemas operacionais multiprogramáveis, as partições precisavam ter tamanho fixo, definido de acordo com as necessidades dos sistemas que seriam utilizados. Quando existia a necessidade de uma das partes ser desabilitada ou redimensionada, era preciso reconfigurar os sistemas e instalá-los de acordo com o tamanho das demais partições. A esse tipo de gerência de memória atribuiu-se o nome de alocação particionada estática ou fixa (MACHADO; MAIA, 2013). Observe na Figura 4.3 um exemplo desse mecanismo de gerência de memória:

Figura 4.3 | Alocação particionada estática



Fonte: Machado, Maia (2013, p. 149)

Quando trabalhamos sob a perspectiva da alocação particionada dinâmica, eliminamos a obrigatoriedade de definição do tamanho das partições, de acordo com o tamanho dos programas que serão instalados. Então, o que temos nesse tipo de particionamento? Como o sistema operacional disponibiliza os recursos?

Simples! Em alocação particionada dinâmica não temos mais a necessidade de definir o tamanho das partições, pois aos programas será disponibilizado apenas o espaço em memória que seja o suficiente para o seu armazenamento.

Com isso, há a otimização do recurso de alocação e, ainda, há a redução dos espaços vazios, ou seja, que por ventura não estejam sendo utilizados por uma aplicação e necessite desfragmentar o disco para realocar e organizar a memória novamente. Esse é um problema que se resolve com o que chamamos de fragmentação externa.

Quando acontece de, nas alocações, sobrar em memória uma determinada quantidade que é considerada relativamente baixa para armazenar outra aplicação, será necessário criar uma área com aquele tamanho que sobrou, mesmo que pequeno, para armazenar outro tipo de informação ou aplicação que requeira menos espaço. Esse procedimento refere-se à fragmentação eterna.

Outra forma é a realocação dinâmica dos espaços disponíveis, ou seja, todos os pequenos espaços são organizados de forma contígua, o que libera mais espaços para armazenamento de outras aplicações.



Faça você mesmo

Verifique em seu computador quantas partições foram criadas e qual é o espaço disponível em disco. Realize o procedimento de desfragmentação, disponível no painel de controle e analise, em termos de eficiência e rendimento da máquina, se está mais rápida.

Atenção:

Devemos ter o cuidado em desfragmentar o SSD, por ser um disco de estrutura sólida e não utilizar leitura mecânica e sim processos elétricos, podendo comprometer a vida útil do equipamento. Então, toda atenção ao realizar os testes!

Sem medo de errar

Com o objetivo de auxiliar no desenvolvimento de uma solução de *software* que venha colaborar com a diminuição do gargalo existente hoje no que tange ao espaço de memória nos dispositivos móveis, vamos apresentar uma proposta que vem sendo utilizada pelo mercado e que nem todo o público-alvo desses equipamentos sabe que é uma ferramenta de apoio bastante útil.

A empresa de pesquisa de desenvolvimento de novos produtos mencionada está

trazendo alternativas já existentes para auxiliar tanto nas frentes de desempenho do produto, fabricação, diminuição de resíduos e descartes desnecessários, além de redução de custos para fabricantes e parceiros, que podem usufruir das soluções existentes no mercado, de forma a incentivar também as pequenas operações e empresas de produtos de *software*. Então, conheça o procedimento recomendado para identificar qual é o aplicativo que será capaz de auxiliar nessa tarefa de maximizar a gerência de memória de um dispositivo móvel.

1. Identifique a marca, o modelo e o sistema operacional do dispositivo móvel para descobrir qual tipo de aplicativo temos disponível e qual o usuário principal ou root do aparelho caso esse não esteja liberado.

2. Vamos limpar a memória cache do aparelho com o *software* chamado *Cache Cleaner*. Serão removidos todos os arquivos temporários do seu celular e esse não precisa necessariamente estar rooteado.

3. Outro *software* que pode ser elencado que auxilia na liberação de espaços de memória é o Link2SD. Esta é uma ferramenta que remove os aplicativos que já vêm instalados na memória interna do celular para o cartão de memória. No lugar do arquivo, fica um *link*, como um atalho que aponta para o endereço do aplicativo no cartão de memória externo. Quando instalado no dispositivo, o próprio aplicativo auxilia na identificação dos aplicativos padrão e indica o passo a passo para realizar a transferência.

4. Além dessas duas ferramentas, o *backup* de dados do aparelho móvel é sempre recomendado.

5. Quanto à desinstalação de arquivos possivelmente ocultos no dispositivo móvel, é possível utilizar para essa tarefa um aplicativo conhecido como *Easy Uninstaller*, que permite desinstalar vários arquivos a partir de uma seleção, em lote.

6. Outra opção é sempre salvar os seus arquivos como fotos e vídeos no cartão de memória externa, pois facilita o gerenciamento e a otimização da memória interna do dispositivo.



Atenção!

Leia o artigo que traz na íntegra o passo a passo e as melhores práticas para melhorar o desempenho de seu dispositivo móvel no que tange aos recursos de memória. Disponível em: <<http://imasters.com.br/mobile/android/espaco-precioso-economize-espaco-em-dispositivos-android/?t=1519021197&source=single>>. Acesso em: 25 set. 2015.



Lembre-se

A melhor estratégia a ser adotada por um sistema depende de uma série de fatores, sendo o mais importante o tamanho dos programas processados no ambiente. Independentemente do algoritmo utilizado, o sistema possui uma lista de áreas livres, com o endereço e tamanho de cada área (MACHADO; MAIA, 2013, p. 154).

Avançando na prática

Pratique mais

Instrução

Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.

Gerenciamento de memória: conceitos, tipos, características e virtualização

1. Competência de fundamento de área	O aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional, bem como ter conhecimento sobre como se dá o compartilhamento de recursos e a sua gerência.
2. Objetivos de aprendizagem	Conhecer os principais conceitos, características da gerência de memória e, o que é e a virtualização de arquivos.
3. Conteúdos relacionados	Gerenciamento de memória: conceitos, tipos, características e virtualização.
4. Descrição da SP	A empresa de pesquisa e desenvolvimento precisa entregar um relatório sobre ações do sistema operacional para realizar a alocação dinâmica em servidores virtualizados. Então, siga as instruções de um dos fornecedores de sistemas operacionais para iniciar as comparações:
5. Resolução da SP	Quando se trata de virtualização de servidores, temos um gerenciador de tarefas de distribuição de recursos entre as máquinas virtuais criadas. Então, seguindo essa premissa, precisamos verificar de que forma acontece essa gerência no que tange aos recursos de memória e de processamento. Para tanto, veja as instruções do fornecedor quando o assunto é alocação dinâmica de recursos em servidores, a considerar o sistema operacional Windows Server: O passo a passo será o seguinte: 1. Iniciar os recursos e adicioná-los ao pool de recursos disponíveis. 2. Notificar os dispositivos e aplicativos registrados sobre os recursos iniciados, de forma que estes dispositivos e aplicativos possam ajustar suas alocações de recursos. 3. No caso dos processadores: iniciar um balanceamento de recursos do sistema de device drivers participantes, para que possam desconectar e reconectar seus manipuladores de interrupção de hardware, e assim fazer uso dos novos recursos.

	<p>Os device drivers não participantes não estão incluídos, de forma que a realocação não interrompe seus serviços.</p> <p>4. Nos casos das pontes de hospedeiros de E/S: verificar os dispositivos conectados ao novo barramento de E/S e, se necessário, iniciar um novo balanceamento de recursos.</p> <p>5. Na conclusão dessas etapas, notificar o processador de serviços que a inclusão automática foi concluída.</p> <p>(Fonte: STANEK, Willian. Virtualização de servidores: particione e conquiste. Technet Microsoft. Disponível em: <https://technet.microsoft.com/pt-br/magazine/gg598495.aspx>. Acesso em: 30 set. 2015.</p> <p>Em seguida, teremos de saber também, como acontece com a adição de recursos. Temos também a possibilidade de realizar a substituição. Observe as orientações do mecanismo de funcionamento desse procedimento segundo o fornecedor do software:</p> <p>"A substituição automática está disponível apenas para memória e processadores (e apenas quando o recurso de substituição é idêntico ao recurso original). O processador de serviços controla uma operação de substituição por meio de:</p> <ol style="list-style-type: none"> 1. Seleção dos recursos sobressalentes disponíveis e necessários. 2. Ativação e inicialização dos recursos. Com memória, o estado dos módulos de memória antigos é copiado para os novos módulos de memória. 3. Notificação ao Windows Server 2008 sobre a iminente operação de substituição. O sistema operacional entra em um estado de suspensão pseudo-S4. Com processadores, o sistema operacional e o firmware do sistema copiam o estado dos processadores antigos para os novos. Com memória, todas as alterações de estado são copiadas para os novos módulos de memória. 4. Mapeamento dos recursos de substituição para a partição de hardware e substituição dos recursos antigos. 5. Notificação ao Windows Server 2008 sobre a conclusão da substituição. O sistema operacional sai do estado de suspensão e retoma as operações regulares. 6. Desativação dos recursos antigos e notificação ao gerenciador de serviços e ao aplicativo de gerenciamento do sistema que a substituição está concluída." <p>(Fonte: STANEK, Willian. Virtualização de servidores: particione e conquiste. Technet Microsoft. Disponível em: <https://technet.microsoft.com/pt-br/magazine/gg598495.aspx>. Acesso em: 30 set. 2015.</p> <p>Com isso, temos também de avaliar a possibilidade de implantar essa solução de virtualização ou se mantêm os melhoramentos previstos para os servidores locais.</p>
--	---



Lembre-se

A primeira implementação de memória virtual foi realizada no início da década de 1960, no sistema Atlas, desenvolvido na Universidade de Manchester. Posteriormente, a IBM introduziria este conceito comercialmente na família System/370, em 1972. Atualmente, a maioria

dos sistemas implementa memória virtual, com exceção de alguns sistemas operacionais de supercomputadores (MACHADO; MAIA, 2013, p. 159).



Faça você mesmo

Leia e aprenda mais sobre virtualização: Disponível em: <<https://technet.microsoft.com/pt-br/magazine/gg598495.aspx>>. Acesso em: 29 set. 2015.

Faça valer a pena

1. Complete a frase de acordo com a alternativa que contém os respectivos conceitos apresentados:

"[...] a _____, também conhecida como _____, que tem por função _____ lógicos em que estão as instruções nas memórias físicas."

- a) unidade central de processamento/CPU (Central Processing Unit)/ controlar processos.
- b) unidade de gerenciamento de memória/MMU (Memory Management Unit)/mapear os endereços
- c) central de distribuição de memória/receptores de sinais/conectar os operadores.
- d) inserção de memória/recursos de alocação de memória virtual/configurar os registradores.
- e) segurança dos dados/unidade central de processamento/registra os dados.

2. O trecho do texto abaixo indica qual tipo de memória? Assinale a alternativa correspondente:

"É considerada mais lenta que os registradores, porém, mais rápida que a memória principal. Ela existe, pois são necessários alguns mecanismos de armazenamento, para permitir que a CPU acesse e realize o processamento da instrução de processo mais rapidamente do que se tivesse que acessar determinado dado ou instrução diretamente em uma das outras memórias: principal, secundária ou permanente. Isso porque o modo como se realiza a busca da informação é diferenciado e cada uma possui uma função diante do contexto de gerenciamento de memória do sistema operacional, sendo que a memória cache ganha cada vez mais espaço para armazenamento de acordo com as novas arquiteturas de

computadores e em função do crescimento das aplicações e necessidades de software” (STUART, 2011).

- a) Memória secundária.
- b) Memória principal.
- c) Memória cache.
- d) Memória externa.
- e) Registradores.

3. Assinale a alternativa que contém uma das principais responsabilidades da gerência de memória:

- a) Manter em memória física ou principal, a maior quantidade de processos residentes de forma que seja possível aproveitar ao máximo o compartilhamento de recursos.
- b) Permitir que um pequeno número de processos leia as instruções e as execute de acordo com a sua ordem na fila.
- c) Buscar a informação do processo nos registradores.
- d) Compartilhar a alocação dinâmica com os processadores.
- e) Executar instruções contidas nas memórias cache e registradores.

4. Assinale a alternativa que contém os tipos de particionamento que podem ser realizados:

- a) Estático e rotativo.
- b) Dinâmico e rotativo.
- c) Estático e predominante.
- d) Estático e dinâmico.
- e) Dinâmico e alternativo.

5. Assinale a alternativa que contém a descrição da técnica de *overlay*:

- a) Determina o tamanho fixo de uma partição.
- b) Especifica o tamanho do módulo e como será a sua alocação de forma a permitir a execução de todos os módulos de uma vez.
- c) Delimita e compartilha os recursos de uma partição com outros sistemas operacionais.

- d) Estabelece um limite de memória e libera após a execução da instrução.
- e) Considera que, diante de uma aplicação, a divisão de módulos auxiliará na determinação do espaço de memória necessária a executar os módulos de forma independente.

6. Explique o conceito de alocação particionada estática.

7. Explique o conceito de alocação particionada dinâmica.

Seção 4.2

Swapping: conceitos, tipos e suas características

Diálogo aberto

Nesta aula, vamos estudar um conceito que está diretamente relacionado aos espaços reservados em memória principal para alocar processos residentes e em um determinado momento precisam ceder esse endereço ou posição de memória a outros processos não residentes, e como ocorre esse mecanismo de gerência de memória.

Nesse contexto, quando falamos em técnicas e mecanismos de gerência de memória, é importante conhecer o que é *swapping*.

Além disso, conhecer as formas de se ampliar a eficiência da multiprogramação em sistemas operacionais é como caminhar para identificar novas formas de compartilhamento de recursos e, ainda, visualizar outras possibilidades de desenvolvimento de soluções.

Mas você pode se perguntar: afinal, esse procedimento ainda ocorre nos sistemas operacionais atuais? Sim, e é cada vez mais comum inclusive para a gerência de memórias virtuais.

Com esta aula, vamos aprofundar os conhecimentos sobre esse procedimento que o sistema operacional realiza para gerenciar a memória, quando ocorre e porque motivos ele foi desenvolvido.

Além disso, precisamos estreitar nossos estudos junto ao mercado de trabalho e, dessa forma, diante do cenário de uma empresa de pesquisa e desenvolvimento, vamos propor um levantamento de técnicas para melhorar o nível de multiprogramação utilizando *swapping* em memórias virtuais.

Não se esqueça de visitar o seu material didático para esclarecer dúvidas de assuntos já abordados, tais como: gerência de processos, sincronização, diretórios e sistemas de arquivos.

Agora que você já sabe o que precisa pesquisar e o que vai aprender, mãos à obra!

Bons estudos e práticas a você!

Não pode faltar

Logo no início desta aula, vimos que estudaremos os conceitos relacionados à gerência de memória e que o foco está em compreender o que é *swapping*. Então, siga em frente e conheça mais uma técnica, utilizada em multiprogramação para otimizar recursos e, com isso, ampliar a eficiência do seu computador.

A técnica de *swapping* foi desenvolvida com o intuito de solucionar um problema comum em multiprogramação, que é a falta de espaço na memória principal. Então, ela propõe que, ao invés de um processo residente em memória principal, esse seja enviado por tempo determinado para a memória secundária, para dar espaço suficiente para que um processo não residente seja alocado e, com isso, após a sua execução, o espaço volta a ser liberado para que aquele processo residente retorne ao endereço de origem.



Assimile

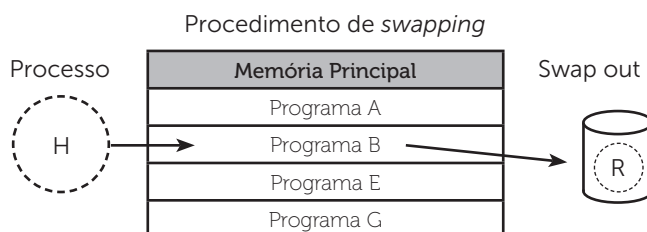
Nesta situação, o sistema escolhe um processo residente, que é transferido da memória principal para a memória secundária (*swap out*), geralmente o disco. Posteriormente, o processo é carregado de volta da memória secundária para a memória principal (*swap in*) e pode continuar sua execução como se nada tivesse ocorrido (MACHADO; MAIA, 2013, p. 156).

Observe no exemplo a seguir qual é a lógica dessa técnica e como o sistema operacional realiza essa gerência:



Exemplificando

Figura 4.4 | Técnica de *swapping*



Fonte: Adaptado de Machado, Maia (2013, p. 156)

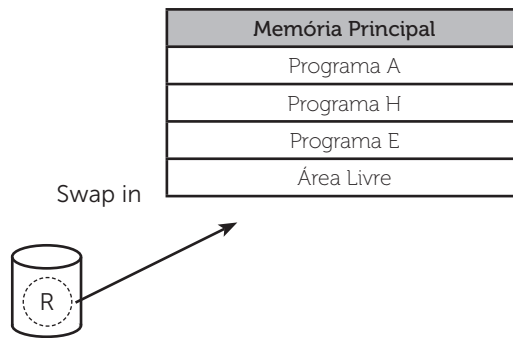
O que a Figura 4.4 apresenta é a identificação de um processo não residente em memória principal, que necessita de mais espaço para ser alocado e entrar em execução. Para que essa concessão aconteça, é

necessário que seja aplicado um algoritmo que verifique qual processo residente está em estado de espera e não será, portanto, alocado em seguida para execução. Com isso, é possível ceder o seu espaço ou endereço de memória, para que um outro processo utilize aquele recurso e entre efetivamente em execução.

Quando o processo escolhido para ser retirado da memória principal está em estado de espera ou de pronto, podendo ser considerado não residente, recebem o nome de *outswapped*.

Em seguida, quando o processo não residente alocado encerra a sua execução, ele é retirado da memória principal retorna ao local de origem.

Figura 4.5 | Processo de entrada de arquivo em *swap in*



Fonte: Machado, Maia (2013, p. 156)

A Figura 4.5 traz o procedimento de devolução do processo residente que estava alocado em disco, ou seja, memória secundária, para a área da memória principal que se encontra liberada.

A técnica de *swapping* somente poderá ser implementada a partir do uso de um registrador de alocação. Esse registrador de alocação é acionado no momento em que o programa ou aplicativo em uso é carregado e recebe o endereço inicial de alocação que foi disponibilizado para ele em memória.

A cada nova solicitação de memória, será somado ao conteúdo desse registrador e com isso, é gerado o seu endereço físico. Isso permite que um determinado programa possa ser carregado em qualquer posição de memória (MACHADO; MAIA, 2013).



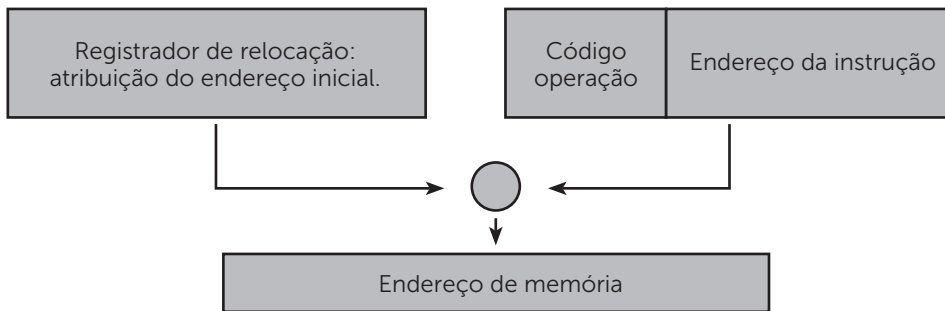
Pesquise mais

Aqui fica uma indicação de palestra sobre implantação de técnicas de gerência de memória virtual em *softwares* livres. Disponível em: <<https://>

www.youtube.com/watch?v=clU0oVnWvcU>. Acesso em: 2 out. 2015.

Na Figura 4.6, temos um exemplo do mecanismo de carregamento de um programa e a sua respectiva referência no registrador de relocação.

Figura 4.6 | Definição de endereço inicial do processo e endereço de memória



Fonte: Machado, Maia (2013, p. 157)

Mas, afinal, qual é a vantagem de se aplicar a técnica de *swapping*?

Vamos listar a seguir algumas delas, confira:

- Ao se realizar *swapping*, é possível compartilhar mais recursos ou endereços na memória principal.
- Com melhor aproveitamento da memória principal, conseguimos também, otimizar o processamento e maximizar a utilização dos recursos da máquina de modo geral.

No entanto, alguns pontos de atenção também podem ser listados como o custo de processador com as operações de entrada e saída (*swap out* e *swap in*) junto à memória principal. Ou seja, é possível perder um pouco de eficiência quando se realizam essas operações, pois os recursos podem estar dedicados justamente ao *swapping* e, com isso, não ser alocados recursos suficientes para que os processos residentes na memória principal entrem em execução.

Quando temos esse tipo de situação acontecendo em gerência de memória, temos o que se chama de *thrashing*, que é um dos problemas mais graves que podem ocorrer e comprometer a eficiência da máquina.



Refleta

Mesmo com o aumento da eficiência da multiprogramação e, particularmente, da gerência de memória, muitas vezes um programa não podia ser executado por falta de uma partição livre disponível. A técnica de

swapping foi introduzida para contornar o problema da insuficiência de memória principal (MACHADO; MAIA, 2013, p. 156).

Outros fatores como não realizar *swapping* de código de processos devem ser levados em consideração, pois imagine se dois processos estejam dividindo o mesmo espaço de memória por *swap*, como deve ser tratado esse problema? A preferência fica em realizar *swap* apenas de dados e de elementos da pilha entre os processos que estão compartilhando do mesmo espaço em memória principal e deixar os segmentos em que estão alocados os códigos intactos até que seja finalizada a execução do processo.

Então, é importante reforçar a ideia de que não há vantagens em se aplicar a técnica de *swapping* quando o espaço liberado em memória não for o suficiente para alocar todos os dados necessários de um determinado processo.

Relembrando que o *hardware* responsável por realizar essa tarefa de seleção do processo que poderá realizar *swapping* é o de gerência de memória. Se a MMU conseguir decodificar uma quantidade relativamente grande de endereços, isso impacta em um ganho de flexibilidade tanto no que tange à escolha do processo que entrará em *swapping* quanto na velocidade. Pois se há a necessidade de alocar apenas parte do processo, como descrito no parágrafo anterior, temos menores tempos de leitura e gravação de dados (STUART, 2011).

E quais são os tipos de *swapping* utilizados? A seguir, vamos listar os respectivos tipos de se organizar e estabelecer uma política de realização de *swapping*:

- *Swapping* por paginação ou paginação de *swapping*: a técnica de paginação é realizada pela unidade de gerência de memória. Com isso, torna-se mais viável utilizá-la, pois as páginas são de tamanhos relativamente pequenos e independente dessa característica, e os segmentos ou quadros de página têm o mesmo tamanho. Isso permite que sejam copiadas páginas de forma individual para a memória principal e memória secundária ou disco. É possível, através de *swapping* de paginação, fazer *swap* de processos inteiros (STUART, 2011). Para que a página seja devidamente identificada, é utilizado o bit de presença "P". Esse bit de presença verifica a existência da página em memória principal ou secundária.
- *Strings* de referência: lista de referência à sequência ordenada de acessos a página para leitura ou gravação.
- Políticas de substituição global (em que os processos podem solicitar mais páginas do que aquelas que a MMU alocou para *swap*) e local (em que o *swap* pode acontecer apenas entre as páginas que o processo possui).
- Mínimo de Belady: política de *swap* criada por Laszlo Belady, em que afirma que uma página que ainda não tenha sido utilizada será a mais viável escolha para se

realizar *swap* para memória secundária.

- FIFO: primeira página a entrar primeira a sair.
- Segunda chance: extensão de FIFO com uma verificação. Se o bit de acesso for "0", a página será enviada por *swap* para a memória secundária. Se o bit de acesso for "1", então o bit de acesso é deletado e a página reinserida ao final da fila.
- Algoritmo do relógio: infere em mais uma análise sob a política da segunda chance, pois organiza, a partir da ideia de ponteiros do relógio, as páginas ao seu redor e, então, se for preciso realizar *swap* em memória secundária, teremos aquela página que está sendo apontada para verificação do bit de acesso.
- Conjunto de trabalho: define-se como o conjunto de páginas que o processo utiliza em um determinado intervalo de tempo.
- Frequência de falha: a partir das falhas de paginação entre os processos, a tendência nessa política é verificar se há páginas sem uso ou com pouco uso. Se houver, será feito *swap* para memória secundária das páginas sem tanta utilidade, pertencentes ao processo que apresentou mais páginas residentes em memória principal do que em seu conjunto de trabalho.
- Segmentos paginados: representam os segmentos dos espaços de endereçamento e, se esses forem relativamente grandes, há a sua paginação, a fim de torná-lo menor.
- Arquivos mapeados na memória: uma vez um arquivo mapeado, este será armazenado. A alocação de páginas para um determinado processo ocorre toda vez em que há uma solicitação de mapeamento de um arquivo em memória.
- Cópia-na-escrita: por exemplo a *fork* (UNIX), em que cria um processo-filho que é a cópia do processo-pai, porém, o que há em termos de *swap* é a cópia no mesmo segmento de memória, tanto do processo-filho quanto do processo-pai de seu mapeamento de endereço em memória virtual, sendo apenas liberados para leitura. Caso aconteça a tentativa de gravação, há uma interrupção e será registrada pela MMU.
- Desempenho: faz referência direta à eficiência e aos cuidados na escolha da política de substituição local e global. São considerados os fatores:
 - Tempos de acesso.
 - *Thrashing*: como mencionado anteriormente.



Faça você mesmo

Analise o exemplo:

Tempo médio de acesso:

Para colocar isso em perspectiva, considere o código executável em linha reta, ignorando quaisquer acessos a dados. Se cada página tiver 4096 bytes e cada instrução, 4 bytes, cruzaremos um limite de página a cada 1024 instruções. Isso implica que $f = 0,000976$ se nenhuma página estiver carregada. Se o tempo de acesso a memória for igual a 50nS (nanossegundos) e o tempo de acesso ao disco for igual a 10mS (milionésimos de segundos), então o tempo médio de acesso (t) será de 9760 nS. O efeito final é que a execução dessas instruções será 200 vezes mais lenta do que seria se todas as instruções estivessem residentes na memória (STUART, 2011, p. 252).

Nesta mesma referência, encontram-se outros exemplos no que tange à aplicação dos tipos de *swapping* listados acima e de que forma acontecem. Elabore um relatório com as diferenças e investigue quais delas os sistemas operacionais mais utilizam.

Sem medo de errar

Os procedimentos a seguir são sugeridos para adicionar uma memória virtual *swap* em um servidor Debian 7.0 Wheezy. Confira o tutorial sugerido no *site* Fator Binário, que traz informações sobre a configuração dessa memória. Vamos lá!

1. Verifique se a memória de troca está instalada:
 - `swapon -s`: mostrará uma tabela em branco
 - `free -m`: mostrará todos os valores zerados na linha "Swap:"
2. Verifique também o espaço livre no HD do servidor:
 - `df -h`
3. Vamos criar uma partição Swap em disco de 2G:
 - `fallocate -l 2G /swapfile`
4. Verifique se a partição Swap foi criada corretamente:
 - `ls -lh /swapfile`

Esse comando resultará a seguinte instrução:

- `-rw-r--r-- 1 root root 2.0G Apr 3 14:19 /swapfile`

5. Definir as permissões do arquivo e habilitações de sistema com a respectiva partição para a memória Swap:

- `chmod 600 /swapfile`
- `mkswap /swapfile`
- `swapon /swapfile`

6. Se quiser deixar a partição como permanente, será preciso editar o arquivo de nome:

• `/etc/fstab`: com a seguinte instrução para inserção de uma nova linha e salve em seguida:

- `/swapfile none swap sw 0 0`

7. Utilize o comando abaixo para evitar alguns erros de inicialização do servidor:

- `mount -a`

8. Você pode usar o seguinte comando para verificar a quantidade de memória em uso:

- `watch -n 5 free -m` : (o parâmetro `-n 5` define uma nova leitura a cada 5 segundos):



Atenção!

Leia o artigo na íntegra e aprenda mais! Disponível em: <<http://fatorbinario.com/linux-adicionando-memoria-virtual-swap-ao-vps/>>. Acesso em: 2 out. 2015.



Lembre-se

Para que a técnica de *swapping* seja implementada é essencial que o sistema *loader* que implementa a relocação dinâmica de programas. Um loader relocável que não ofereça esta facilidade permite que um programa seja colocado em qualquer posição de memória, porém, a relocação é realizada apenas no momento do carregamento (MACHADO; MAIA, 2013, p. 157).

Avançando na prática

Pratique mais	
Instrução Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.	
Swapping: conceitos, tipos e suas características	
1. Competência de fundamentos de área	O aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional, bem como ter conhecimento sobre como se dá o compartilhamento de recursos e a sua gerência.
2. Objetivos de aprendizagem	Conhecer o que é swapping, como o sistema operacional gerencia esta tarefa e aloca os devidos recursos, a fim de atender às demandas de processamento e alocação.
3. Conteúdos relacionados	Swapping: conceitos, tipos e suas características.
4. Descrição da SP	É uma situação comum que os usuários de computadores queiram destinar uma das partições de sua máquina a outro sistema operacional. Com isso, é preciso definir manualmente, muitas vezes, o espaço que será reservado às operações de <i>swapping</i> . Sua tarefa é investigar, de acordo com um fornecedor de sistemas operacionais, quais os procedimentos que podem ser aplicados. Siga em frente!
5. Resolução da SP	<p>Vamos instalar o Ubuntu em um notebook que já tem instalado o Windows e será preciso, portanto, fazer o que se chama de Dual Boot.</p> <p>Para isso, será preciso criar manualmente as partições necessárias para cada operação, como, por exemplo, a memória em que serão trocados arquivos (Swap) e a memória em que serão armazenados.</p> <p>Você pode seguir o passo a passo recomendado no artigo indicado no site "Mundo Ubuntu", que contém as informações detalhadas para a realização desse procedimento.</p> <p>Essa é uma prática interessante e te ajuda a fixar os conceitos estudados nesta aula.</p> <p>Siga as recomendações para:</p> <p>"Veja como criar um pendrive de boot com o Ubuntu 12.04. Leia esta matéria caso seu computador não esteja reconhecendo o boot pelo pendrive (boot pela USB)."</p> <p>E teste os procedimentos indicados!</p> <p>Fonte: Instalação do Ubuntu12.04 mostrando manualmente a criação de partições swap em dual boot com windows. Disponível em: <http://www.mundoubuntu.com.br/tutoriais/instalacao/77-instalacao-do-ubuntu-12-04-mostrando-manualmente-a-criacao-das-particoes-swap-e-home-em-dual-boot-com-o-windows-7>. Acesso em: 2 out. 2015.</p>



Lembre-se

Os primeiros sistemas operacionais que implementaram esta técnica

surgiram na década de 1960, como o CTSS do MIT e OS/360 da IBM. Com a evolução dos sistemas operacionais, novos esquemas de gerência de memória passaram a incorporar a técnica de *swapping*, como a gerência da memória virtual (MACHADO; MAIA, 2013, p. 157).



Faça você mesmo

Agora que você já conheceu a técnica, porque foi desenvolvida e como está sendo aplicada atualmente, elabore um relatório que contemple outros sistemas operacionais e o passo a passo para definir o espaço para troca de arquivos em memória, *swapping*.

Faça valer a pena

1. O que é *swapping*? Assinale a alternativa correspondente:

- a) Processo de gerência de memória secundária pelo sistema de arquivos.
- b) Técnica que estabelece políticas de trocas de espaços de memória principal para secundária para alocação de processos.
- c) Técnica que realiza a sobreposição de processos na memória principal.
- d) Técnica que auxiliar no processamento de dados de pilha apenas.
- e) Procedimento de recuperação de arquivos em RAM.

2. Analise as afirmações e assinale a alternativa correspondente:

I. *Swapping* faz a verificação de disponibilidade de troca de um processo da memória principal para a memória secundária.

II. Através de um algoritmo que é responsável por verificar qual processo residente está estado de espera, um processo pode ceder o seu espaço para que um outro processo utilize aquele recurso.

III. Quando o processo escolhido para ser retirado da memória principal está em estado de espera ou de pronto, podem ser considerados não residentes e, como foram eles os selecionados, recebem o nome de *outswapped*.

- a) V, V, V.
- b) F, F, F.
- c) V, F, F.

- d) F, V, V.
- e) V, V, F.

3. *Swap out* refere-se à:

- a) operação de saída de um processo da memória secundária para os registradores;
- b) operação de entrada de um processo da memória principal para os registradores;
- c) operação de saída de um processo da memória principal para ser armazenado em disco ou memória secundária;
- d) operação de entrada de um processo em memória principal;
- e) operação de entrada de um processo em memória cache e saída para RAM.

4. *Swap in* refere-se à:

- a) operação de saída de um processo da memória secundária para os registradores;
- b) operação de entrada de um processo da memória principal para os registradores;
- c) operação de saída de um processo da memória principal para ser armazenado em disco ou memória secundária;
- d) operação de entrada de um processo que estava em memória secundária e é enviado para memória principal;
- e) operação de entrada de um processo em memória cache e saída para RAM.

5. O que é *trashing* no que concerne o gerenciamento de memória? Descreva.

6. Descreva o *swapping* por paginação.

7. Assinale a alternativa que contém a descrição da política de *swapping* de segmento de paginação.

- a) Alocação de páginas para um determinado processo ocorre toda vez

em que há uma solicitação de mapeamento de um arquivo em memória.

b) Cópia no mesmo segmento de memória, tanto do processo-filho quanto do processo-pai de seu mapeamento de endereço em memória virtual, sendo apenas liberados para leitura.

c) Representam os segmentos dos espaços de endereçamento e, se esses forem relativamente grandes, há a sua paginação, a fim de torná-lo menor.

d) É o conjunto de páginas que o processo utiliza em um determinado intervalo de tempo.

e) Lista de referência à sequência ordenada de acessos à página para leitura ou gravação.

Seção 4.3

Memória virtual: conceitos, paginação, segmentação e virtualização

Diálogo aberto

Olá, aluno! Vamos à penúltima aula desta unidade curricular e com ela aprenderemos o que é memória virtual. Apesar de já termos tido contato com esse conceito em algumas de nossas atividades, vamos conhecer todos os relacionados.

Até aqui, estudamos nesta unidade o que é a gerência de memória, quais são os seus tipos, como é realizada a alocação de recursos e os tipos de particionamento. No caso, nós fizemos uma breve descrição dos particionamentos do tipo fixo e dinâmico.

Além disso, estudamos também a técnica de *swapping*. Essa técnica requer que se estabeleça uma política para a realização da troca de processos da memória principal para a memória secundária. Dessa forma, elencamos algumas, como *swapping* por paginação, que é um conceito que será abordado também nesta seção, quando descrevermos o que significa para o sistema operacional separar em páginas a sua disponibilidade em memória para alocação de processos.

Nesse contexto, vamos estudar o que é memória virtual e para que serve. Diante dessa técnica, é possível estabelecer critérios de seleção e organização de processos para execução e aumento da velocidade de processamento.

É muito comum encontrarmos, por exemplo, um computador que esteja mais lento, com tempos de resposta mais longos. Nesse contexto, precisamos avaliar o que acontece com o *hardware* e o *software* para uma possível modificação.

No caso, supondo que a empresa de pesquisa e desenvolvimento está comprometida em ensinar mecanismos de otimização de memória virtual, vamos trabalhar para aumentar o espaço disponível a ela. A sua tarefa agora é auxiliar na investigação do processo necessário para inserir esse acréscimo de memória.

Ficam as seguintes recomendações: leia atentamente seu material didático, realize as atividades de aprendizagem propostas e acesse os *links* em destaque para ampliar o seu conhecimento acerca desse assunto.

Desde já, desejamos bons estudos e práticas a você!

Não pode faltar

Você sabe qual é a diferença entre a memória RAM, a técnica de *swapping* e a memória virtual? Então, este será nosso ponto de partida nessa aula. Siga em frente!

Vamos recapitular as respectivas definições conceituais. A primeira delas que devemos observar é a memória virtual. O conceito de memória virtual foi criado para solucionar alguns problemas como o da fragmentação do disco. A memória virtual também é útil quando se utiliza a técnica de *overlay*, em que apenas parte de um processo é dimensionada e alocada a uma área de memória principal, sendo este processo parcial dividido em módulos. Porém podem faltar recursos para alocar os processos, ainda que divididos em módulos, em um espaço que seja suficiente. Então, a gerência desse serviço é mais complexa.

A memória virtual se mostra como uma solução em função de trabalhar com uma quantidade maior de processos que compartilham a memória principal, uma vez que, destes processos já estão alocadas apenas as partes residentes. Isso faz com que menos recursos de processamento sejam utilizados.



Refleta

Memória virtual é uma técnica sofisticada e poderosa de gerência de memória, em que as memórias principal e secundária são combinadas dando ao usuário a ilusão de existir uma memória muito maior que a capacidade real da memória principal. O conceito de memória virtual fundamenta-se em não vincular o endereçamento feito pelo programa dos endereços físicos da memória principal. Desta forma, programas e suas estruturas de dados deixam de estar limitados ao tamanho da memória física disponível, pois podem possuir endereços associados à memória secundária (MACHADO; MAIA, 2013, p. 159).

Existem três técnicas para implementar memória virtual: paginação, segmentação e segmentação com paginação. É comum associar o conceito de memória virtual com vetores, em função de, apesar de os dados serem alocados em posições distintas, não haver a necessidade de saber a exata posição de um determinado dado. Basta que instruções de código sejam desenvolvidas para que no contexto de *software* possam trazer a instrução de acordo com a sua respectiva identificação. A diferença é que um aplicativo ou *software* não fará a referência direta aos endereços de memória física do processo, e sim ao seu respectivo endereço de memória virtual. Em termos

de processamento, por esse necessitar apenas do endereço físico, ou seja, temos a conversão do endereço virtual ao seu em memória principal. A esse procedimento de conversão de endereços atribui-se o nome de mapeamento, a considerar que um processo é constituído pelo seu espaço de endereçamento e pelos contextos de *hardware* e de *software*.

Já a memória RAM (*Randon Access Memory*) é um *hardware* que armazena, de acordo com uma limitação que pode ser da ordem de gigabytes, uma determinada quantidade de informações, porém, temporariamente. O acesso aos dados é realizado de forma não sequencial e, por esse motivo, ela tem em seu nome o termo “randômica” (aleatória), em que o acesso é feito a um processo sempre que solicitado, de maneira aleatória, não obedecendo a uma ordem de acesso.

A velocidade de um computador para processar informações está diretamente relacionada ao tamanho de sua memória RAM, em função de ser essa a responsável por armazenar o processo de forma que não tenha de ser mapeado o seu endereço em memória secundária, e seja acessado diretamente e em sua respectiva posição e enviado para ser executado. É considerada volátil, por precisar de uma fonte de energia que a alimente.

Então, os processos saem da fila de execução ou porque foram concluídos e, com isso, já liberaram o espaço para que outro processo seja alocado, ou porque o computador foi desligado e não há mais a necessidade de armazenar dados de processo para execução imediata.



Assimile

E quando o problema parece ser o tamanho da memória virtual? Nesse caso, você precisa analisar bem o problema e diferenciar se há a necessidade de alteração de MV ou se é mais viável adicionar memória RAM em seu computador. Veja a seguir um artigo da Microsoft acerca dessa situação que requer uma análise mais criteriosa de viabilidade:

“Se faltar ao seu computador a quantidade de memória RAM necessária para executar um programa ou uma operação, o Windows usa a memória virtual para compensar. Para descobrir a quantidade de RAM existente no computador, consulte descobrir a quantidade de RAM existente no computador. A memória virtual combina a RAM do computador com espaço temporário no disco rígido. Quando a RAM fica insuficiente, a memória virtual move os dados da RAM para um espaço chamado arquivo de paginação. Isso libera a RAM para que o computador possa concluir seu trabalho. Quanto mais RAM um computador tem, mais rápido ele irá executar os programas. Se a falta de RAM estiver diminuindo o desempenho do computador, é possível que você fique tentado

a aumentar a memória virtual para compensar. Entretanto, como o computador pode ler dados da RAM com muito mais rapidez do que de um disco rígido, a melhor solução é adicionar RAM.”

Fonte: Disponível em: <<http://windows.microsoft.com/pt-br/windows/what-is-virtual-memory#1TC=windows-7>>. Acesso em: 5 out. 2015.

Mas, afinal, como a técnica de *swapping* interage com a memória principal?

Esse identifica a oportunidade de realizar uma troca da memória principal para a memória secundária (*swap out*) de um processo que tenha o mesmo tamanho em termos de segmentos alocados e, assim que o processo alocado encerra a execução, libera este espaço para o processo original residente retornar da memória secundária para a memória principal (*swap in*).

Foi na universidade de Manchester que houve a primeira implementação de memória virtual. O sistema operacional era o Atlas, desenvolvido em 1962 nesta universidade.



Pesquise mais

No *site* de um desenvolvedor conhecido de sistemas operacionais e outros *softwares*, a Microsoft disponibiliza artigos que podem auxiliar na compreensão e aplicação dos conceitos sobre memória virtual aqui estudados. Disponível em: <<http://windows.microsoft.com/pt-br/windows/what-is-virtual-memory#1TC=windows-7>>. Acesso em: 5 out. 2015.

A principal vantagem é que um *software* pode fazer referência a um processo que esteja fora da memória principal, pois deixaram de ser limitados ao tamanho da memória física para que possam ser selecionados e entrar em execução.

Sendo assim, a memória secundária é utilizada pelo sistema operacional como uma forma de expandir a memória de trabalho. Apenas uma parte do processo que é residente permanece na memória principal.

E, quanto ao mapeamento dos endereços que mencionamos, como é esse mecanismo?

Com um algoritmo de mapeamento inserido na decodificação do endereço virtual para o endereço físico do processo, temos condições de não alocar contiguamente os processos em memória principal, considerando-se que em sistemas operacionais modernos a unidade de gerência de memória fica responsável por essa conversão ou tradução de endereços (MACHADO; MAIA, 2013).

Nesse contexto de mapeamento de endereços, podemos elencar uma estrutura de dados em tabela que contém as informações de cada processo. Assim, a cada vez que ele for solicitado, será acessada a sua respectiva tabela com as informações de seu endereço virtual. Observe o exemplo de tabela de endereços virtuais utilizada no mapeamento de blocos de dados:



Exemplificando

Tabela 4.2 | Espaço virtual x tamanho do bloco

End. Virtual	Tamanho do bloco	Número de blocos	Nº de entradas na tabela de mapeamento
2^{32} endereços	512 endereços	2^{23}	2^{23}
2^{32} endereços	4 K endereços	2^{30}	2^{30}
2^{64} endereços	4 K endereços	2^{52}	2^{52}
2^{64} endereços	64 K endereços	2^{48}	2^{48}

Fonte: Machado, Maia (2013, p. 163)

A tabela representa uma relação inversamente proporcional em que, quanto maior o bloco, teremos menos entradas na tabela de endereços virtuais.

Esses blocos a que nos referimos na tabela podem ser de tamanho fixo (paginação) ou de tamanho variável (segmentação).

Vamos agora falar dos conceitos de memória virtual por paginação. Já falamos anteriormente que, em função do tamanho dos segmentos de endereços alocados para os processos, muitas vezes o sistema operacional tratará essas informações em blocos menores, chamados de páginas, e criará uma de mesmo tamanho para fazer referência a esta na memória secundária.

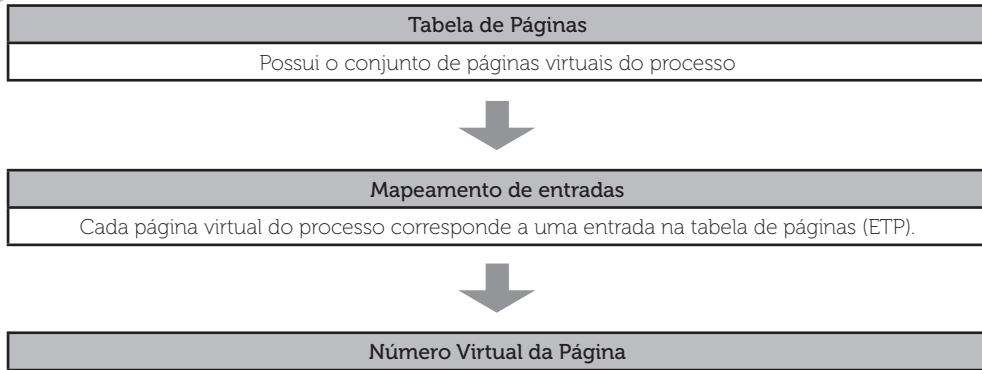
Para as páginas da memória principal, damos o nome de páginas reais, também chamadas de frames. E páginas virtuais quando se trata da respectiva paginação em memória virtual.

O fluxo de mapeamento de memória virtual pode ser definido da seguinte forma:

Tabela 4.3 | Fluxo de mapeamento de memória virtual

Processo
Possui uma tabela de páginas própria.





Fonte: Adaptado de Machado, Maia (2013, p. 163)

Estamos dizendo, na Figura 4.3, que todo processo possui a sua respectiva tabela de páginas e que cada uma das páginas representa uma entrada na tabela de páginas. Além disso, que a sua localização pode ser realizada a partir o número virtual da página, também chamado de (NVP), que é uma espécie de índice para localizar a informação do processo na memória virtual.

A ETP, ou entrada na tabela de páginas, é composta por informações como o bit de validade do processo, que indica através da representação "0" que está na página principal e "1", que o processo se encontra na memória virtual.

Além disso, se o processo estiver na memória virtual e a unidade de gerência de memória identificar a partir do bit de validade que a página não está também na memória principal, acontece um procedimento chamado de *page fault*, pois identificou que está faltando a correspondência na memória principal. Com isso, uma cópia do bloco de dados da memória virtual é enviado para a memória principal e é corrigido o problema, porém, em função de aumentar as operações de entrada e saída, podemos ter perdas de eficiência da máquina. Por esse motivo, controla-se esse tipo de comportamento do sistema por meio de um indicador, conhecido como taxa de paginação do processo. Ao fato de a página virtual ser enviada para a sua respectiva tabela de processo na memória principal, atribui-se o nome de *page in*, ou entrada de página.

Quando esse procedimento de *page in* se torna frequente, pode acarretar em perda de eficiência da máquina.

Da mesma forma, quando uma página está identificada como *page fault*, o seu processo passa do estado de execução para o estado de espera enquanto acontece a transferência da memória secundária para a memória principal. Com isso, há a troca de contexto e a respectiva atualização da informação na tabela de mapeamento e a restauração das informações do processo para que possa novamente entrar em procedimento de escalonamento, retornando à fila de processos que se encontram em estado de pronto (MACHADO; MAIA, 2013).

Você agora deve estar se perguntando: de que forma o sistema operacional gerencia a demanda de processos no contexto da alocação de recursos em memória virtual? Para contornar essa situação e permitir o controle de demanda, foi criada a política de busca de páginas. Essa pode ser classificada em: paginação por demanda, ou seja, há a transferência da memória secundária para a principal apenas quando há a referência, e, ainda, há a paginação antecipada, voltada também para o controle de demanda, no entanto com o acréscimo de páginas, caso o processo venha precisar já tem algumas alocadas e a disposição.

Além da política de buscas, temos também as seguintes políticas de:

1. Alocação de páginas.
2. Substituição de páginas.
3. *Working set*.
4. Algoritmos de substituição de páginas:
 - a. Ótimo.
 - b. Aleatória.
 - c. FIFO (*First in First out*).
 - d. LFU (*Least- Frequently- Used*).
 - e. LRU (*Least- Recently- Used*).
 - f. NRU (*Not- Recently- Used*).
 - g. FIFO com *buffer* de páginas.
 - h. FIFO circular.
5. Política para estabelecer o tamanho da página.
6. Paginação com múltiplos níveis.
7. Tradução de endereços virtuais em endereços reais.
8. Proteção de memória.
9. Compartilhamento de memória.

Além dos conceitos aqui apresentados, não podemos esquecer que, no que diz respeito ao conceito de memória virtual, também temos a sua implementação realizada por segmentação. Nesse sentido, essa pode ser realizada de acordo com uma segmentação por paginação, ou, ainda, realizando *swapping* de memória virtual. Além

disso, um ponto de atenção que precisamos reforçar aqui é a questão do *trashing*, que pode acontecer nas operações diretamente relacionadas à memória virtual.




Faça você mesmo

De acordo com as políticas mencionadas, elabore um relatório que contenha as respectivas definições e um exemplo de aplicação de cada uma delas! Bons estudos.

Sem medo de errar

Vamos, então, aprender a configurar o tamanho da memória virtual de sua máquina a partir das recomendações da Microsoft, fornecedora do sistema operacional Windows. Confira abaixo o procedimento adotado:

“Se você receber avisos de que a memória virtual é insuficiente, aumente o tamanho mínimo do arquivo de paginação. O Windows define o tamanho mínimo inicial do arquivo de paginação como a quantidade de memória RAM instalada no computador, e o tamanho máximo, como três vezes a quantidade de RAM instalada no computador. Se você vir avisos nesses níveis recomendados, aumente os tamanhos mínimo e máximo.

1. Para abrir Sistema, clique no botão Iniciar , clique com o botão direito em Computador e clique em Propriedades.
2. No painel esquerdo, clique em Configurações avançadas do sistema. Se você for solicitado a informar uma senha de administrador ou sua confirmação, digite a senha ou forneça a confirmação.
3. Na guia Avançado, em Desempenho, clique em Configurações.
4. Clique na guia Avançado e, em Memória virtual, clique em Alterar.
5. Desmarque a caixa de seleção Gerenciar automaticamente o tamanho do arquivo de paginação de todas as unidades.
6. Em Unidade [Rótulo do Volume], clique na unidade que contém o arquivo de paginação que você deseja alterar.
7. Clique em Personalizar Tamanho e digite um novo tamanho em megabytes na caixa Tamanho inicial (MB) ou Tamanho máximo (MB). Em seguida, clique em Definir e em OK.”

Fonte: Disponível em: <<http://windows.microsoft.com/pt-br/windows/change-virtual-memory-size#1TC=windows-7>>. Acesso em: 05 out. 2015.



Atenção!

Veja a referência completa do artigo e mais informações sobre o procedimento no *site*: <<http://windows.microsoft.com/pt-br/windows/change-virtual-memory-size#1TC=windows-7>>. Acesso em: 5 out. 2015.



Lembre-se

Existe um forte relacionamento entre a gerência de memória virtual e a arquitetura de *hardware* do sistema computacional. Por motivos de desempenho, é comum que algumas funções da gerência da memória virtual sejam implementadas diretamente no *hardware*. Além disso, o código do sistema operacional deve levar em consideração várias características específicas da arquitetura, especialmente o esquema de endereçamento do processador (MACHADO; MAIA, 2013, p. 159).

Avançando na prática

Pratique mais	
Instrução Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.	
Memória virtual: conceitos, paginação, segmentação e virtualização	
1. Competência de fundamentos de área	O aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional, bem como ter conhecimento sobre como se dá o compartilhamento de recursos e a sua gerência.
2. Objetivos de aprendizagem	Conhecer os principais conceitos sobre memória virtual e os procedimentos necessários a essa implantação.
3. Conteúdos relacionados	Memória virtual: conceitos, paginação, segmentação e virtualização.
4. Descrição da SP	Uma vez que o foco dessa pesquisa repousa sobre memória virtual e suas formas de implementar, realize um levantamento em uma base de referência, que compare o procedimento de implementação de memória virtual por paginação com o de segmentação.
5. Resolução da SP	Quando falamos em implementação de memória virtual por meio de segmentação, podemos elencar as seguintes características que também a diferem do processo de paginação, a começar pelos atributos de sua tabela de entrada de segmento (ETS). Observe a seguir:

Tabela 4.4: Campos da ETS

Campo	Descrição
Tamanho	Especifica o tamanho do segmento.
Bit de validade	Indica se o segmento está na memória principal.
Bit de modificação	Indica se o segmento foi alterado.
Bit de referência	Indica se o segmento foi recentemente referenciado, sendo utilizado pelo algoritmo de substituição.
Proteção	Indica a proteção do segmento.

Fonte: Machado e Maia (2013, p. 183).
Além dessas características, observe os apontamentos de Machado e Maia (2013, p. 184), que comparam os procedimentos de paginação com segmentação:

Tabela 4.5: Paginação x segmentação

Característica	Paginação	Segmentação
Tamanho dos blocos de memória	Iguais	Diferentes
Proteção	Complexa	Mais simples
Compartilhamento	Complexo	Mais simples
Estruturas de dados dinâmicas	Complexo	Mais simples
Fragmentação interna	Pode existir	Não existe
Fragmentação externa	Não existe	Pode existir
Programação modular	Dispensável	Indispensável
Alteração do programa	Mais trabalhosa	Mais simples

Fonte: Machado e Maia (2013, p. 184).

Vantagens da segmentação com relação à paginação:

- a. É compatível com estruturas de dados dinâmicas, como pilhas e filas. Na paginação, a alteração do vetor requer a criação de novas páginas e a alteração da ETP.
- b. As transferências de memória secundária para a memória principal acontecem apenas com a referência dos segmentos. Os softwares precisam ser desenvolvidos em módulos para aumentar o grau de multiprogramação.
- c. Quanto à fragmentação em paginação, temos esse problema internamente, enquanto que, na segmentação, a fragmentação externa pode ocorrer.
- d. O mapeamento em segmentação é mais simples, pois mapeia estruturas lógicas e não as páginas (MACHADO; MAIA, 2013).



Lembre-se

Outra técnica é a segmentação com paginação, em que o espaço de endereços de memória é dividido em segmentos e cada segmento divide-se em páginas.



Faça você mesmo

Em sua concepção, tratando-se de gerência de memória seja por paginação, segmentação ou segmentação por paginação, qual é a importância de se seguir uma política que determine como serão realizados os compartilhamentos, buscas, substituições e acessos? Você, diante da arquitetura de computadores utilizada atualmente, já se encontra em condições de avaliar outras possibilidades de se fazer gerência de memória visando otimizar recursos e aumentar eficiência de máquina?

Faça valer a pena

1. Analise as afirmações e assinale a alternativa correspondente:

I. Foi criada para solucionar alguns problemas, como o da fragmentação do disco, ou mesmo quando se utiliza a técnica de *overlay*.

II. Trabalha com uma quantidade maior de processos que compartilham a memória principal.

III. O acesso aos dados é realizado de forma não sequencial e, por esse motivo, ela tem em seu nome o termo “randômica” (aleatória), em que o acesso é feito a um processo sempre que solicitado, de maneira aleatória, não obedece a uma ordem de acesso.

Estamos, respectivamente, falando de:

- a) RAM, *swap* e memória virtual;
- b) *swap*, RAM e memória virtual;
- c) memória virtual, *swap* e RAM;
- d) memória virtual, memória virtual e RAM;
- e) RAM, RAM, *swapping*.

2. Complete as lacunas da frase com as opções de uma das alternativas respectivamente:

"O conceito de _____ fundamenta-se em não vincular o endereçamento feito pelo programa dos endereços físicos da _____. Desta forma, programas e suas estruturas de dados deixam de estar limitados ao tamanho da memória física disponível, pois podem possuir endereços associados à _____" (MACHADO; MAIA, 2013, p. 159).

- a. memória secundária/memória virtual/memória principal;
- b. memória virtual/memória principal/memória secundária;
- c. memória virtual/memória secundária/memória híbrida;
- d. memória virtual/memória virtual/memória secundária;
- e. memória virtual/memória secundária/memória principal.

3. São técnicas de implementação de memória virtual:

- a. Paginação, segmentação e segmentação com paginação.
- b. *Swapping*, *trashing*, FIFO.
- c. Randômico, integral, distorção.
- d. Cliente Servidor, roteirização.
- e. Escalonamento, sincronização e *threads*.

4. É verdadeiro o que se afirma em:

I. A principal vantagem em memórias virtuais é que um *software* pode fazer referência a processos que estejam fora da memória principal, pois deixaram de ser limitados ao tamanho da memória física para que possam ser selecionados e entrar em execução.

II. A principal vantagem é que em memórias virtuais o acesso ao dado é idêntico ao acesso randômico e não utiliza identificadores de processos ou bit de validade.

III. O bit de validade indica o tempo de execução dos processos que estão na memória virtual.

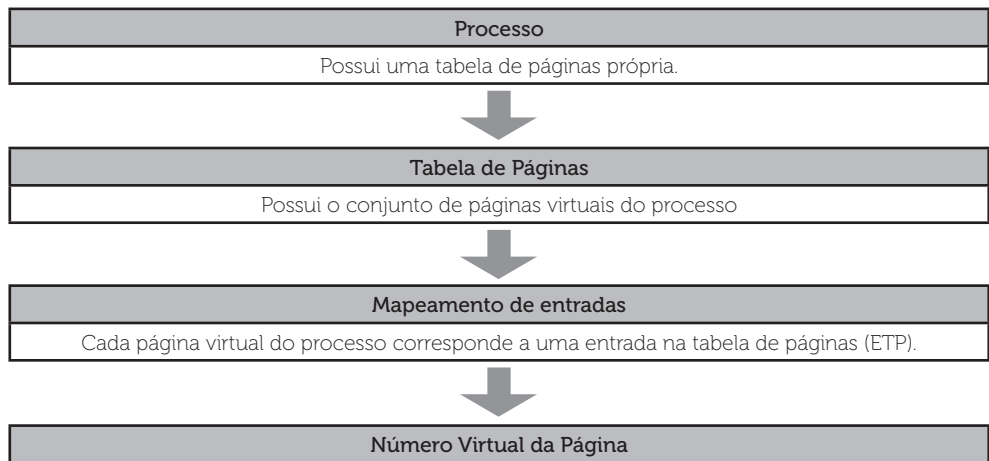
- a. I e II.
- b. Apenas I.
- c. Apenas II.
- d. II e III.
- e. I, II e III.

5. Explique o mecanismo de mapeamento de processos em memória virtual:

6. Qual é a relação com a memória virtual dos conceitos de *page fault* e *page in*?

7. Analise a figura e assinale a alternativa correspondente:

Figura 4.8 | Fluxo de mapeamento de memória virtual



Fonte: Adaptado de Machado e Maia (2013, p. 163).

A figura representa:

- que todo processo possui a sua respectiva tabela de páginas e que cada uma das páginas representa uma entrada na tabela de páginas. Além disso, que a sua localização pode ser realizada a partir o número virtual da página;
- a ETP, ou entrada na tabela de páginas, que é composta por informações como o bit de validade do processo que indica através da representação "0" que está na página principal e "1", que o processo se encontra na memória virtual;
- o processo na memória virtual e a unidade de gerência de memória identificar a partir do bit de validade que a página não está também na memória principal, acontece um procedimento chamado de *page fault*;
- a estrutura de página virtual enviada para a tabela de processo na memória principal e se chama relatório de paginação;
- o fluxo de operações do sistema operacional.

Seção 4.4

Gerenciamento de dispositivos de entrada e saída: conceitos, rotinas, tipos e suas características

Diálogo aberto

Estamos encerrando os estudos dos fundamentos de sistemas operacionais. Chegou o momento de conhecer como acontece a gerência de dispositivos de E/S e a sua importância para o sistema operacional. Com isso, vamos aproximar a teoria e a prática a partir da implementação de técnicas e, também, da compreensão das funções dos *drivers* para as operações de leitura e escrita.

Você precisará identificar uma forma de a empresa de P&D em questão desenvolver as melhorias necessárias, competentes ao driver responsável pela recepção, codificação e transmissão dos dados que são provenientes das ações do usuário no sistema, com as operações de E/S.

Para resolver essa situação, você precisa apresentar um tipo de driver, os seus modos de operação, bem como descrever brevemente de que modo acontecem as operações de interpretação das operações de leitura e escrita.

Além disso, aqui conseguimos atingir o objetivo de conhecer como acontece a gerência de dispositivos de E/S pelo sistema operacional, o que nos permite identificar, nos computadores que trabalhamos, a necessidade de se atualizar algum driver, por exemplo, ou mesmo que tipo de problema e a sua respectiva solução, o que certamente pode nos ajudar a desempenhar as tarefas auxiliadas pelo uso do computador, em nosso cotidiano.

Não se esqueça de recorrer ao seu material, mesmo às unidades de ensino anteriores, para relembrar conceitos já estudados, como gerência de processos, de memória, sistemas de arquivos e os assuntos a esses interligados.

Além disso, resolver os exercícios e tirar suas dúvidas é muito importante. Então, leia com atenção o seu livro didático e acompanhe regularmente as atualizações em conteúdo pertinentes à disciplina.

Não pare por aqui, continue avançando e conheça ainda mais sobre o universo da

tecnologia da informação, pois essa fará parte de toda a sua trajetória profissional.

Ficamos por aqui. Obrigado pela sua atenção e dedicação aos estudos!

Não pode faltar

Considerada como uma das mais complexas tarefas que o sistema operacional realiza, a gerência dos dispositivos de entrada e saída é responsável por facilitar a comunicação entre as solicitações dos usuários e as aplicações. Para tanto, é preciso que a arquitetura de comunicação entre os dispositivos de E/S e os respectivos *hardwares* e *softwares* estejam interligados e possam atender a essas demandas.

Vamos conhecer melhor a arquitetura em camadas e seus elementos, ilustrados na Figura 4.9, e também seremos apresentados aos principais conceitos relacionados a:

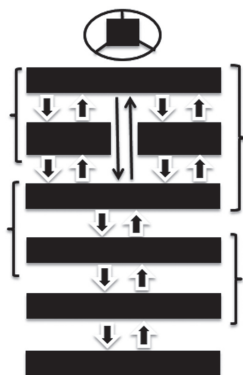
- subsistema de E/S;
- *device Driver*;
- controlador;
- dispositivos de E/S;
- discos magnéticos ou SGBD.

Além disso, conheceremos também como se aplicam questões de segurança da informação na gerência de dispositivos de entrada e saída.



Assimile

Figura 4.9 | Arquitetura de camadas da gerência de dispositivos



Fonte: Machado e Maia (2013, p. 209)

A arquitetura de gerência de dispositivos é dividida em camadas que se distribuem da seguinte forma:

1. Dispositivos ou periféricos de entrada e saída (E/S): mecanismos que permitem a interação entre usuário e máquina de forma amigável e segura junto às aplicações.

2. Controlador: esse *hardware* faz a interface entre a solicitação do usuário e o driver. É composto por memória e registradores programados para enviar as instruções ao respectivo driver. Aqui é que são armazenadas as sequências de bits que o dispositivo de E/S envia. Quando não houver erros, o controlador enviará a informação do bloco para a memória principal.

3. *Device driver*: nessa camada, há instruções que realizarão a comunicação das solicitações enviadas pelo controlador, ao subsistema de E/S. Sua função principal é interpretar ou traduzir as instruções recebidas, para comandos que são compreensíveis tanto para os controladores quanto para o subsistema de E/S.

4. Subsistema de E/S: tem a função de distinguir, de acordo com cada dispositivo, as solicitações e executar as rotinas de comunicação que realiza entre as aplicações, dos sistemas de arquivos e dos sistemas de gerenciamento de bancos de dados, além das solicitações advindas do *device driver*.

5. Sistema de arquivos, sistemas de gerenciamento de bancos de dados e as aplicações se relacionam de forma a gerar as demandas de processos.

Com isso, observamos o isolamento no tratamento de informações de dispositivos para com as informações do processo.



Pesquise mais

O artigo "Como funcionam os dispositivos de entrada e saída", disponível em: <http://www.devmedia.com.br/como-funcionam-os-dispositivos-de-entrada-e-saida/28275>, traz informações sobre como o sistema operacional consegue interpretar as solicitações que vêm dos dispositivos de entrada e saída. Acesso em: 5 out. 2015.

Em função da quantidade de tipos de dispositivos de E/S ser bastante diversificada, temos de trabalhar com a camada de subsistema de E/S. Essa tem por função, como mencionado, isolar as atividades e solicitações dos periféricos, aplicações, sistemas de arquivos e SGBDs, de forma a permitir que os periféricos interajam com qualquer um deles e suas demandas sejam tratadas de forma diferente.

Com isso, é preciso ter um conjunto de rotinas, também chamado de rotinas de entrada e saída, para fazer essa tarefa de comunicação otimizada. Devemos fixar a forma

de como se estabelece essa comunicação, sendo que as rotinas de E/S é que geram as demandas de execução das operações de E/S.

As operações de entrada e saída podem estar organizadas logicamente, ou, de forma estruturada, quando armazenadas em dispositivos como pen drives ou disco rígido, por exemplo, ou, ainda, de forma não estruturada, quando o dispositivo não permite esse tipo de configuração.

Temos, com isso, algumas formas de se estabelecer comunicação. São elas:

- Chamada explícita: caracteriza-se pela chamada realizada diretamente pelo sistema operacional, a partir de comandos de alto nível.
- Chamada implícita: caracteriza-se pela realização das operações de E/S para leitura e gravação, através de programação de alto nível (linguagem C, por exemplo). Além desta, ainda pode realizar as chamadas de rotinas contidas nas bibliotecas de funções dessas linguagens de programação.

A camada de subsistema de E/S também considera a comunicação a partir do sincronismo, permitindo que aconteça, portanto, de forma síncrona ou assíncrona. Leia e analise a reflexão a seguir sobre essa classificação:



Refleta

As operações de E/S podem ser classificadas conforme o seu sincronismo. Uma operação é dita síncrona quando o processo que realizou a operação fica aguardando no estado de espera pelo seu término. A maioria dos comandos das linguagens de alto nível funciona desta forma. Uma operação é dita assíncrona quando o processo que realizou a operação não aguarda pelo seu término e continua pronto para ser executado. Neste caso, o sistema deve oferecer algum mecanismo de sinalização que avise ao processo que a operação foi terminada (MACHADO; MAIA, 2013, p. 210).

Esse subsistema de E/S também trabalha com uma técnica chamada *buffering*, que deriva do espaço de memória intermediária conhecida como *buffer*, em que são carregados os blocos de dados de forma a facilitar o acesso ao dado e se mais algum processo solicitar informações, e essas estiverem no intervalo do bloco de dados, não haverá a necessidade de se realizar mais uma operação de E/S (MACHADO; MAIA, 2013).

O subsistema de E/S verifica deve receber a programação correspondente a cada dispositivo, ou seja, o driver para cada um e, assim, sempre que um periférico for reconhecido pela máquina o driver deverá ser imediatamente chamado pelo subsistema e instalado.

Quando falamos de driver, estamos nos referindo diretamente ao processo de comunicação com o subsistema de E/S e também, com os controladores. Em uma visão mais ampla, temos o reconhecimento do dispositivo pelo subsistema de E/S. Já em uma visão mais próxima, micro, temos o drive, que contém as especificações de cada um dos dispositivos de E/S. Os drivers recebem as informações e as codifica para que o controlador ou o subsistema possam reconhecer aquelas instruções e chamar os processos inerentes a cada uma das tarefas ou operações de E/S.

Um driver pode conter as instruções e comandos de um dispositivo, bem como de um grupo de dispositivos, de forma a facilitar a programação do driver e o espaço ocupado para armazenar instruções de mesmo tipo. Algumas das características dos drivers devem ser consideradas, como:

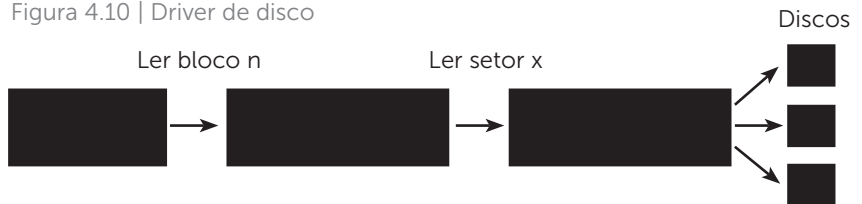
- número de registradores do controlador;
- comandos do dispositivo;
- o fluxo e processos de funcionamento do dispositivo a que se refere.



Exemplificando

Quando o sistema recebe uma solicitação de E/S para a leitura de dados, podemos citar como exemplo o funcionamento do modo síncrono de se realizar esta operação. A informação de localização deve chegar ao cilindro, trilha e setor. Observe na Figura 4.10 como ocorre esse procedimento de comunicação síncrona:

Figura 4.10 | Driver de disco



Fonte: Machado e Maia (2013, p. 212)

Assim, quando o driver recebe essa demanda de leitura de um bloco, deverá informar ao disco todos os dados de localização daquele determinado bloco de dados.

O controlador de entrada e saída tem a função de intermediar a comunicação com os dispositivos de E/S. É um *hardware* e assemelha-se a uma placa ou vem implementado no próprio chip do processador.

Ele é composto por:

- memória;
- registradores específicos que realizam a comunicação com o driver.

No controlador, forma-se o bloco de dados das operações de leitura e ele é transferido, se não for identificado nenhum tipo de erro, para a memória principal. Então, podemos dizer que há, respectivamente, uma transferência de blocos de dados que estão armazenados internamente no *buffer* e são transferidos para o *buffer* de E/S. Essa tarefa é realizada pelo processador ou por um controlador que utilize a técnica chamada de DMA (*Direct Memory Access* ou Acesso Direto à Memória Principal).



Pesquise mais

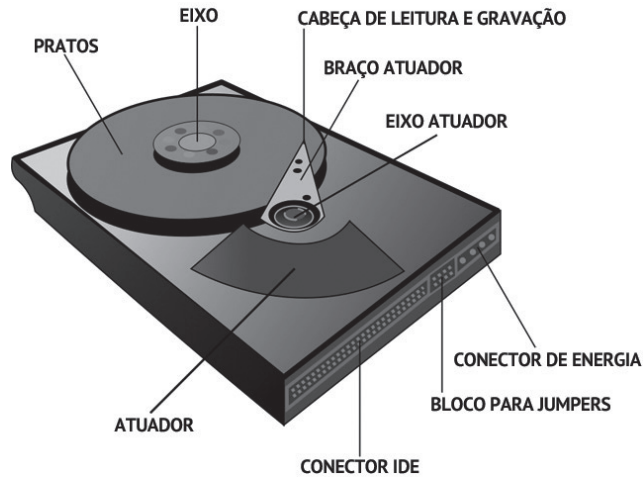
Leia sobre DMA. Disponível em: <<http://www.hardware.com.br/termos/dma>>. Acesso em: 8 out. 2015.

Mas você sabe ou se recorda quais são os dispositivos de entrada e saída? Vamos rever esta informação. Acompanhe a seguir quais são os dispositivos que fazem a comunicação entre o sistema operacional e o ambiente externo:

Podemos classificar como dispositivos de entrada de dados: teclado, mouse, microfone, pen drive, CDs, DVDs. Já como os de saída de dados, naturalmente, podemos citar as impressoras, o monitor, bem como aqueles que podem servir tanto para a entrada como para a saída de dados: modems, pen drives, CDs, DVDs, HDs externos, entre outros. Atualmente, temos até os aparelhos celulares como uma forma de se estabelecer essa comunicação também e solicitações de entrada e saída. Então, quando conectamos o nosso celular via cabo ou *bluetooth*, também estamos realizando operações de E/S. Observe a necessidade de quando se conecta um deles a uma outra máquina ou dispositivo, de um mecanismo que reconheça e permita que seja dada sequência nas operações tanto de leitura quanto de gravação e interface com as aplicações e sistemas de arquivos requeridos.

Além desses, ainda temos os discos magnéticos, que merecem atenção especial em função da quantidade de dados que podemos armazenar e, justamente por isso, fatores como segurança da informação e proteção devem ser levados em consideração. Esses discos magnéticos são compostos por discos sobrepostos que são unidos por um eixo vertical, que permite a sua rotação e em uma velocidade constante. Os discos são compostos por trilhas, compostas por setores. As trilhas dos discos sobrepostos formam um cilindro, sendo que para cada disco há um procedimento que permite a leitura e gravação de dados.

Figura 4.11 | *Hard disk* ou disco rígido



Fonte: <<https://www.oficinadanet.com.br/post/8632-como-funciona-um-disco-rigido-hd>>. Acesso em: 8 out. 2015.



Faça você mesmo

Elabore um relatório que contenha os tempos de leitura e gravação em disco: *seek*, latência rotacional e transferência e também traga quais são as técnicas de RAID e as diferenças. Investigue formas de implementação advindas dos fornecedores de *software*.

Sem medo de errar

Agora, vamos conhecer como é que se estabelece a comunicação entre a solicitação de ação do sistema pelo usuário através dos dispositivos de entrada e saída. O intuito é fazer com que seja compreendido esse procedimento, que servirá como fundamento para os melhoramentos que a empresa de pesquisa e desenvolvimento precisa investigar. Siga em frente e aproveite para testar algumas das soluções!

Como a empresa de P&D pretende encontrar as melhores práticas e disseminá-las, vamos apresentar, aqui, a interpretação do processo realizada por Vila, que é um pesquisador nessa área. Confira a seguir os seus apontamentos e explicações de conceitos que podem auxiliar nas ações de melhoramentos desse mecanismo:

- *Line Discipline*: pode ser definida como uma interface entre os processos de usuários e os drivers de terminal, ou seja, responsáveis por realizar a codificação e decodificação das solicitações de usuários em informações ou instruções de máquina. Esse driver pode trabalhar de dois modos de operação.

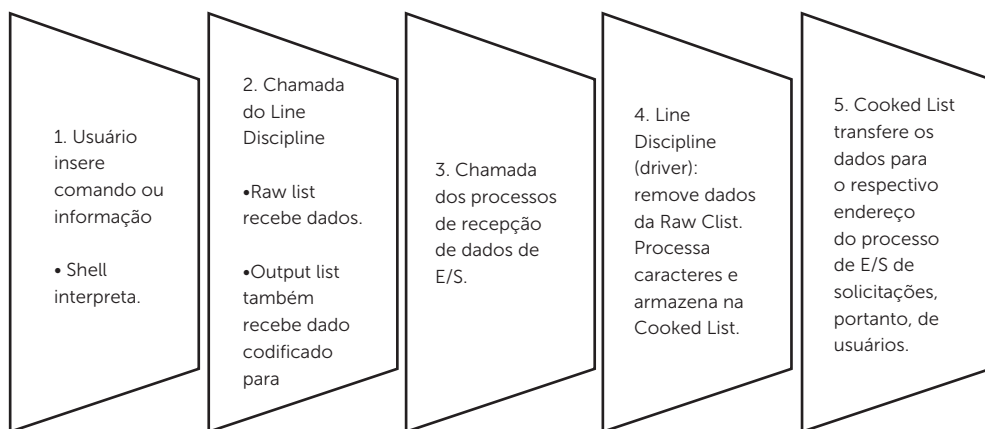
- *Raw* e *Cooked*: responsáveis pela transmissão e codificação, respectivamente, dos caracteres de entrada para os processos oriundos de solicitações de usuários. Essa informação deve passar pelo interpretador de comandos, também conhecido como *shell*.

Além desses conceitos, para entender melhor sobre como é o funcionamento das ações de leitura e escrita pelo sistema operacional, também podemos citar a "Clist" ou também conhecida por lista de caracteres. A Clist trabalha com uma lista encadeada de blocos de caracteres, chamada de Cblocks. Essa armazena um vetor do tipo caractere com as informações enviadas pelos dispositivos de E/S e aponta para o próximo bloco de endereços, considerando o tamanho do vetor, ou seja, considera os endereços de início e fim, como se fosse um tipo de buffer e faz, efetivamente, três operações de leitura e escrita:

- *Raw Clist*: responsável por armazenar os dados de entrada.
- *Cooked Clist*: responsável por armazenar os dados de entrada processados pelos modos de operação mencionados no *line discipline*.
- *Output Clist*: essa função do Cblock armazena os dados de saída que deverão ser exibidos para o usuário.

Agora, podemos entender como funciona uma operação de leitura:

Figura 4.12 | Processo de leitura e escrita de dados de E/S



Fonte: Adaptado de VILA, Fabrício. Universidade Federal de Campina Grande. Disponível em: <http://fubica.lsd.ufcg.edu.br/hp/cursos/so/LabSO/ent_saida.html#e1-d>. Acesso em: 7 out. 2015.



Atenção!

Um sistema operacional possui diversos drivers. Cada um será responsável por uma tarefa diferente e é constituído, portanto, de funções distintas, por exemplo, existem drivers de disco, drivers de rede, drivers de vídeo.



Lembre-se

O device driver, ou somente driver, tem como função implementar a comunicação do subsistema de E/S com os dispositivos, por intermédio dos controladores de E/S. Enquanto o subsistema de E/S trata de funções ligadas a todos os dispositivos, os drivers tratam apenas dos seus aspectos particulares (MACHADO; MAIA, 2013, p. 211).

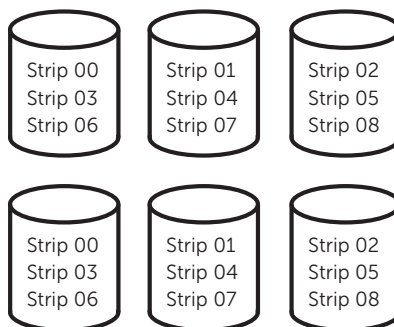
Avançando na prática

Pratique mais	
Instrução Desafiamos você a praticar o que aprendeu, transferindo seus conhecimentos para novas situações que pode encontrar no ambiente de trabalho. Realize as atividades e depois compare-as com as de seus colegas.	
Gerenciamento de dispositivos de entrada e saída: conceitos, rotinas, tipos e suas características	
1. Competência de fundamentos de área	O aluno deverá ser capaz de identificar quais são as principais funções de um sistema operacional, bem como ter conhecimento sobre como se dá o compartilhamento de recursos e a sua gerência.
2. Objetivos de aprendizagem	Conhecer como se dá a gerência de processos advindos das solicitações dos dispositivos de entrada e saída.
3. Conteúdos relacionados	Gerenciamento de dispositivos de entrada e saída: conceitos, rotinas, tipos e suas características.
4. Descrição da SP	Todos sabemos que as instituições financeiras movimentam muitos dados e precisam garantir a segurança da informação em seu aspecto mais amplo e, também, criar mecanismos que permitam que os serviços oferecidos on-line e mesmo entre agências, estejam sempre à disposição de seus clientes. Então, uma das técnicas que utilizam é a conhecida como "Espelhamento" ou RAID1. O representante do órgão fiscalizador dos bancos entrou em contato com a empresa de P&D para solicitar melhorias e possibilidades de se realizar com segurança esse procedimento de replicação dos dados em outros discos secundários, que permitam a garantia da oferta dos serviços bancários aos seus clientes. Então, a fim de atender a essa solicitação, a empresa deve elaborar um relatório que evidencie esse mecanismo de espelhamento e explique como funciona e a melhor forma de se implementar. Sua tarefa é explicar esse mecanismo. Siga em frente!
5. Resolução da SP	A técnica conhecida como espelhamento permite a replicação de dados do disco principal para um disco secundário. A essa replicação, podemos atribuir um conceito chamado de redundância, que é basicamente o fato de repetir ou permitir o acesso à informação, quantas vezes for necessário.

Dessa forma, quando o disco principal vir a ter algum tipo de falha, teremos a mesma informação disponível em um disco secundário. Essa gerência de discos está diretamente interligada às responsabilidades do sistema de arquivos.

Então, para todas as operações de E/S que forem solicitadas, há a sincronização entre o disco principal e seus espelhos, para que aconteça a replicação dos dados. Observe a Figura 4.13, que ilustra esta atividade do sistema:

Figura 4.13: Espelhamento



Fonte: Adaptado de Machado e Maia (2013, p. 218).



Lembre-se

O overhead adicional exigido nesta operação é pequeno, e o benefício da proteção justifica a implementação. Apesar da vantagem proporcionada pela redundância oferecida por esta técnica, a capacidade útil do subsistema de discos com a implementação do RAID1 é de apenas 50% (MACHADO; MAIA, 2013, p. 218).



Faça você mesmo

Siga as orientações do fornecedor para fazer a implantação. Abaixo um tutorial disponibilizado pela Microsoft para configurar RAID1 no Windows Server 2008 R2 Server Core:

“Para criar um disco espelhado (Mirror), também conhecido como RAID 1, são necessários no mínimo dois discos. Em um volume espelhado todos os dados são gravados em dois discos diferentes. Se um disco falhar, os dados serão preservados no outro disco. Neste exemplo, um volume espelhado de 1000MB será criado usando 1000MB em cada disco. Antes de criar um volume espelhado você deve:

Carregar o comando DISKPART:

DISKPART

Listar os discos existentes:

LIST DISK

Dentro do DISKPART, selecione os discos que irá usar no seu volume RAID 1 (lembrando que são somente 2 discos):

Exemplo:

SELECT DISK 1

Converter o disco para dinâmico:

CONVERT DYNAMIC

Se o disco estiver como Read Only, você não poderá convertê-lo para dinâmico. Então desative o atributo:

ATTRIBUTES DISK CLEAR READONLY

Agora com os discos convertidos para dinâmico siga os próximos passos para criar o volume espelhado:

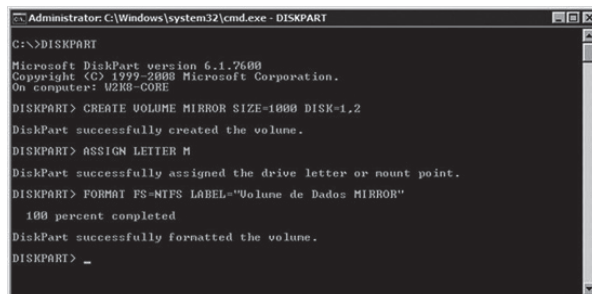
CREATE VOLUME MIRROR SIZE=1000 DISK=1,2

Defina a letra da unidade para o volume criado:

ASSIGN LETTER M

Formate a partição recém-criada:

FORMAT FS=NTFS LABEL="Volume de Dados MIRROR"



```
Administrator: C:\Windows\system32\cmd.exe - DISKPART
C:\>DISKPART
Microsoft DiskPart version 6.1.7600
Copyright (C) 1999-2006 Microsoft Corporation.
On computer: UZERB-CORE

DISKPART> CREATE VOLUME MIRROR SIZE=1000 DISK=1,2
DiskPart successfully created the volume.
DISKPART> ASSIGN LETTER M
DiskPart successfully assigned the drive letter or mount point.
DISKPART> FORMAT FS=NTFS LABEL="Volume de Dados MIRROR"
100 percent completed
DiskPart successfully formatted the volume.
DISKPART> _
```

Fonte: DONDA, Daniel. <<http://social.technet.microsoft.com/wiki/contents/articles/3993.aspx>>. Acesso em: 07 out. 2015.

Faça valer a pena

1. Associe os conceitos na tabela e assinale a alternativa correspondente:

I. Dispositivos de E/S	() Esse hardware faz a interface entre a solicitação do usuário e o driver.
II. Controlador	() Nessa camada, há instruções que realizarão a comunicação das solicitações enviadas pelo controlador ao subsistema de E/S.
III. Device driver	() Tem a função de distinguir de acordo com cada dispositivo, as solicitações e executar as rotinas de comunicação que realiza entre as aplicações, dos sistemas de arquivos e dos sistemas de gerenciamento de bancos de dados, além das solicitações advindas do device driver.
IV. Subsistema de E/S	Mecanismos que permitem a interação entre usuário e máquina de forma amigável e segura junto às aplicações.

- a) I, II, III e IV.
- b) IV, III, II e I.
- c) II, III, IV e I.
- d) III, II, I e IV.
- e) IV, I, II, e III.

2. É verdadeiro o que se afirma em:

I. Em função da quantidade de tipos de dispositivos de E/S ser bastante diversificada, temos de trabalhar com a camada de Subsistema de E/S, que faz a interface com o processo diretamente.

II. A camada de subsistema de E/S tem por função isolar as atividades e solicitações dos periféricos, aplicações, sistemas de arquivos e SGBDs, de forma a permitir que os periféricos interajam com qualquer um deles e suas demandas sejam tratadas de forma diferente.

III. Temos de fixar a forma de como se estabelece essa comunicação, sendo que as rotinas de E/S é que geram as demandas de execução das operações de E/S.

- a) Apenas III.
- b) I e III.
- c) Apenas II.
- d) II e III.
- e) Apenas I.

3. Explique brevemente o modo de funcionamento dos *device drivers* ou simplesmente driver:

4. Assinale a alternativa que contém os componentes de um driver:

- a) Número de registradores do controlador; comandos do dispositivo e o fluxo de processos de funcionamento do dispositivo a que se refere.
- b) Quantidade de bits de um bloco de dados e o seu endereço de memória.
- c) Endereço de memória e registradores utilizados.
- d) Mapa de bits da memória e alocação da RAM.
- e) Quantidade de registradores e informações da RAM.

5. As afirmações a seguir referem-se respectivamente a que tipo de interação ou classificação das operações de E/S? Assinale a alternativa correta:

I. Caracteriza-se pela chamada realizada diretamente pelo sistema operacional, a partir de comandos de alto nível.

II. Caracteriza-se pela realização das operações de E/S para leitura e gravação, através de programação de alto nível (linguagem C, por exemplo). Além dessa, ainda podem realizar as chamadas de rotinas contidas nas bibliotecas de funções dessas linguagens de programação.

- a) Roteamento/ mapa de bits.
- b) Chamada explícita/ chamada implícita.
- c) Chamada de processo/ device driver.
- d) Rotina de E/S / chamada implícita.
- e) Chamada implícita/ chamada explícita.

6. O controlador de operações de E/S é composto por:

- a) registradores e mouse;
- b) teclado e memória;
- c) registro e banco de dados;
- d) processos e aplicações;
- e) memória e registradores.

7. Como o controlador realiza a transferência de blocos de dados que estão armazenados internamente no *buffer*, transferindo-os para o *buffer* de E/S da memória principal?

Referências

VILAR, Francisco. Universidade Federal de Campina Grande. Disponível em: <http://fubica.lsd.ufcg.edu.br/hp/cursos/so/LabSO/ent_saida.html#e1>. Acesso em: 6 out. 2015.

MACHADO, Francis B.; MAIA, Luiz P. **Arquitetura de Sistemas Operacionais**. 5. ed. Rio de Janeiro: LTC, 2013.

STUART, Brian L. **Princípios de sistemas operacionais**: projetos e aplicações. São Paulo: Cengage Learning, 2010.

Anotações

[illegible]

Anotações

[illegible]

Anotações

[illegible]

Anotações

[illegible]

Anotações

[illegible]

Anotações

[illegible]

Anotações

[illegible]

ISBN 978-85-8482-233-1



9 788584 822331 >