INSTITUTO FEDERAL SUL-RIO-GRANDENSE
UNIVERSIDADE ABERTA DO BRASIL
Programa de Fomento ao Uso das
TECNOLOGIAS DE COMUNICAÇÃO E INFORMAÇÃO NOS CURSOS DE GRADUAÇÃO - TICS

SISTEMAS OPERACIONAIS

José Antônio Oliveira de Figueiredo

Ministério da Educação
GOVERNO FEDERAL BRASIL PAÍS RICO É PAÍS SEM POBREZA

Apostila de Sistemas Operacionais

FIGUEIREDO, José Antonio Oliveira de

2012/1

Produzido pela Equipe de Produção de Material Didático da
Universidade Aberta do Brasil do Instituto Federal Sul-rio-grandense

PRESIDÊNCIA DA REPÚBLICA

Dilma Rousseff
PRESIDENTE DA REPÚBLICA FEDERATIVA DO BRASIL

MINISTÉRIO DA EDUCAÇÃO

Fernando Haddad
MINISTRO DO ESTADO DA EDUCAÇÃO

Luiz Cláudio Costa
SECRETÁRIO DE EDUCAÇÃO SUPERIOR - SESU

Eliezer Moreira Pacheco
SECRETÁRIO DA EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

Luís Fernando Massonetto
SECRETÁRIO DA EDUCAÇÃO A DISTÂNCIA – SEED

Jorge Almeida Guimarães
PRESIDENTE DA COORDENAÇÃO DE APERFEIÇOAMENTO DE PESSOAL DE NÍVEL SUPERIOR - CAPES

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE [IFSUL]

Antônio Carlos Barum Brod
REITOR

Daniel Espírito Santo Garcia
PRÓ-REITOR DE ADMINISTRAÇÃO E DE PLANEJAMENTO

Janete Otte
PRÓ-REITORA DE DESENVOLVIMENTO INSTITUCIONAL

Odeli Zanchet
PRÓ-REITOR DE ENSINO

Lúcio Almeida Hecktheuer
PRÓ-REITOR DE PESQUISA, INOVAÇÃO E PÓS-GRADUAÇÃO

Renato Louzada Meireles
PRÓ-REITOR DE EXTENSÃO

IF SUL-RIO-GRANDENSE
CAMPUS PELOTAS

José Carlos Pereira Nogueira
DIRETOR-GERAL DO CAMPUS PELOTAS

Clóris Maria Freire Dorow
DIRETORA DE ENSINO

João Róger de Souza Sastre
DIRETOR DE ADMINISTRAÇÃO E PLANEJAMENTO

Rafael Blank Leitzke
DIRETOR DE PESQUISA E EXTENSÃO

Roger Luiz Albernaz de Araújo
CHEFE DO DEPARTAMENTO DE ENSINO SUPERIOR

IF SUL-RIO-GRANDENSE
DEPARTAMENTO DE EDUCAÇÃO A DISTÂNCIA

Luis Otoni Meireles Ribeiro
CHEFE DO DEPARTAMENTO DE EDUCAÇÃO A DISTÂNCIA

Beatriz Helena Zanotta Nunes
COORDENADORA DA UNIVERSIDADE ABERTA DO BRASIL – UAB/IFSUL

Marla Cristina da Silva Sopeña
COORDENADORA ADJUNTA DA UNIVERSIDADE ABERTA DO BRASIL – UAB/IFSUL

Cinara Ourique do Nascimento
COORDENADORA DA ESCOLA TÉCNICA ABERTA DO BRASIL – E-TEC/IFSUL

Ricardo Lemos Sainz
COORDENADOR ADJUNTO DA ESCOLA TÉCNICA ABERTA DO BRASIL – E-TEC/IFSUL

IF SUL-RIO-GRANDENSE
UNIVERSIDADE ABERTA DO BRASIL

Beatriz Helena Zanotta Nunes
COORDENADORA DA UNIVERSIDADE ABERTA DO BRASIL – UAB/IFSUL

Marla Cristina da Silva Sopeña
COORDENADORA ADJUNTA DA UNIVERSIDADE ABERTA DO BRASIL – UAB/IFSUL

Mauro Hallal dos Anjos
GESTOR DE PRODUÇÃO DE MATERIAL DIDÁTICO

PROGRAMA DE FOMENTO AO USO DAS TECNOLOGIAS DE COMUNICAÇÃO E INFORMAÇÃO NOS CURSOS DE GRADUAÇÃO –TICs

Raquel Paiva Godinho
GESTORA DO EDITAL DE TECNOLOGIAS DE INFORMAÇÃO E COMUNICAÇÃO – TICS/IFSUL

Ana M. Lucena Cardoso
DESIGNER INSTRUCIONAL DO EDITAL TICS

Lúcia Helena Gadret Rizzolo
REVISORA DO EDITAL TICS

EQUIPE DE PRODUÇÃO DE MATERIAL DIDÁTICO – UAB/IFSUL

Lisiane Corrêa Gomes Silveira
GESTORA DA EQUIPE DE DESIGN

Denise Zarnottz Knabach
Felipe Rommel
Helena Guimarães de Faria
Lucas Quaresma Lopes
Tabata Afonso da Costa
EQUIPE DE DESIGN

Catiúcia Klug Schneider
GESTORA DE PRODUÇÃO DE VÍDEO

Gladimir Pinto da Silva
PRODUTOR DE ÁUDIO E VÍDEO

Marcus Freitas Neves
EDITOR DE VÍDEO

João Eliézer Ribeiro Schaun
GESTOR DO AMBIENTE VIRTUAL DE APRENDIZAGEM

Giovani Portelinha Maia
GESTOR DE MANUTENÇÃO E SISTEMA DA INFORMAÇÃO

Anderson Hubner da Costa Fonseca
Carlo Camani Schneider
Efrain Becker Bartz
Jeferson de Oliveira Oliveira
Mishell Ferreira Weber
EQUIPE DE PROGRAMAÇÃO PARA WEB

SUMÁRIO

SUMÁRIO

GUIA DIDÁTICO GD

APRESENTAÇÃO

Prezado(a) aluno (a),

Olá pessoal, sejam bem-vindos à disciplina de Sistemas Operacionais na modalidade a distância.

A disciplina está dividida em duas partes, subdivididas em diversas unidades. Na primeira parte, estudaremos a parte operacional do sistema e, na segunda, o funcionamento e a arquitetura conceitual de funcionamento do sistema operacional.

As unidades foram planejadas para que o você possa conhecer a fundamentação teórica necessária e testar ou verificar o funcionamento do objeto estudado na prática.

Nossos laboratórios foram planejados e testados em máquina virtual - evitando a necessidade de se ter mais que um computador à disposição.

Bom trabalho!

Objetivo Geral

Ao final desta disciplina o aluno será capaz de instalar e configurar um sistema operacional, além de compreender os mecanismos de funcionamento dos sistemas operacionais modernos.

Habilidades

- Conhecer e instalar Sistemas Operacionais;
- Gerenciar um sistema operacional, no que tange ao gerenciamento de programas; gerenciamento de arquivos; gerenciamento de usuários de grupos;
- Compreender os serviços que permitem a interoperabilidade;
- Entender o conceito de processo e multiprogramação;
- Compreender a função do escalonador;
- Compreender os mecanismos de gerencia de memória;
- Compreender os sistemas de arquivos e suas diferenças;
- Compreender os sistemas de gerenciamento de I/O;

Metodologia

A disciplina será desenvolvida em 100 horas através do Ambiente Virtual de Aprendizado Moodle, onde serão disponibilizados materiais a serem estudados para subsidiar a aprendizagem. O Moodle será o canal de comunicação direto entre discentes e tutores, com as seguintes possibilidades de interação:

- Disponibilizar aos discentes as tarefas a serem realizadas.
- Publicar os materiais de apoio e de leitura complementar.
- Acompanhar o desempenho dos discentes em relação às atividades propostas.
- Interagir com a turma através de fórum de discussão, salas de chat e correio eletrônico.
- Acessar e avaliar as tarefas realizadas pelos discentes.
- Estimular o trabalho cooperativo entre os discentes.
- Promover o estudo e o exercício prático autônomo.
- Acompanhar a frequência de acesso ao ambiente pelos discentes.
- Acessar links interessantes e relacionados ao curso.

Primeira Semana

As atividades a serem desenvolvidas na primeira semana são:

1. Apresentação do professor, da disciplina e questões gerais.

Segunda Semana

As atividades a serem desenvolvidas na segunda semana são:

1. Leitura e estudo do conteúdo: Introdução, histórico e tipos de Sistemas Operacionais.
2. Participação no fórum de discussão proposto.

Terceira Semana

As atividades a serem desenvolvidas na terceira semana são:

1. Leitura e estudo do conteúdo: Instalação de sistema operacional e gerenciamento de usuários
2. Assistir a vídeo aula: Instalação de um sistema operacional.
3. Atividade prática: instalação e configuração de usuários em um sistema operacional.

Quarta Semana

As atividades a serem desenvolvidas na quarta semana são:

1. Assistir animação: Porque gerenciar arquivos e suas permissões.
2. Leitura e estudo do conteúdo: Gerenciamento de arquivos.
3. Atividade prática: Manipulação de arquivos e diretórios.
4. Participação no fórum para solução de problemas da atividade prática.

Quinta Semana

As atividades a serem desenvolvidas na quinta semana são:

1. Assistir animação: Porque gerenciar pacotes.
2. Leitura e estudo do conteúdo: Gerenciamento de pacotes
3. Atividade prática: Instalação/testes/remoção de programas
4. Participação no fórum para solução de problemas da atividade prática.

Sexta Semana

As atividades a serem desenvolvidas na sexta semana são:

1. Leitura e estudo do conteúdo: Interoperabilidade e Antivírus
2. Atividade prática: Instalação e configuração de programas para interoperabilidade
3. Participação no fórum para solução de problemas da atividade prática.

Sétima Semana

As atividades a serem desenvolvidas na sétima semana são:

1. Atividade presencial

Oitava e Nona Semanas

As atividades a serem desenvolvidas na oitava e nona semana são:

1. Assistir animação: Animação mostrando o abrir e fechar programas.
2. Leitura e estudo do conteúdo: Multiprogramação.
3. Atividade prática: Criação/Destruição e gerenciamento de processos
4. Participação no fórum para solução de problemas da atividade prática.

Décima Semana

As atividades a serem desenvolvidas na décima semana são:

1. Assistir animação: Animação mostrando diversos programas abertos em um sistema, buscando chamar a atenção para o comportamento do processador
2. Leitura e estudo do conteúdo: Gerenciamento do processador
3. Atividade prática: Manipulação de prioridade de processos
4. Participação no fórum para solução de problemas da atividade prática.

Décima Primeira Semana

As atividades a serem desenvolvidas na décima primeira semana são:

1. Assistir animação: Animação mostrando diversos programas abertos em um sistema, buscando chamar a atenção para o comportamento da memória
2. Leitura e estudo do conteúdo: Gerenciamento de memória
3. Atividade: Exercício sobre gerenciamento de memória.

Décima Segunda e Décima Terceira Semanas

As atividades a serem desenvolvidas na décima segunda e decima terceira semanas são:

1. Assistir animação: Animação sobre manipulação de arquivos.
2. Leitura e estudo do conteúdo: Gerenciamento de arquivos.
3. Atividade prática: Programação com manipulação de arquivos.
4. Participação no fórum para solução de problemas da atividade prática.

Décima Quarta Semana

As atividades a serem desenvolvidas na décima quarta semana são:

1. Assistir animação: Animação sobre utilização de dispositivos de I/O
2. Leitura e estudo do conteúdo: Gerenciamento de I/O
3. Atividade prática: Instalação de driver de dispositivo
4. Participação no fórum para solução de problemas da atividade prática.

Decima Quinta Semana

As atividades a serem desenvolvidas na décima quinta semana são:

1. Atividade presencial

Referências:

FERREIRA, Rubem E. Linux: **Guia do Administrador do Sistema**. São Paulo: Novatec, 2003.

OLIVEIRA, Rômulo Silva de; CARISSIMI, Alexandre da S.; TOSCANI, Simão Sirineo. **Sistemas operacionais**. 3. ed.Porto Alegre : Bookman ; UFRGS, 2008.

TANENBAUM, Andrew S. **Sistemas operacionais modernos**. 2.ed. São Paulo: Pearson Prentice Hall, 2003.

BATTISTI, Júlio. **Windows Vista: curso completo**. Rio de Janeiro: Axel Books, 2007.

SIEVER, Ellen; et al. **LINUX : guia essencial.** Tradução João Tortello. 5.ed. Porto Alegre: Bookman, 2006.

Currículo Professor-Autor

José Antônio Oliveira de Figueiredo

Bacharel em Ciência da Computação pela Universidade de Passo Fundo (2004) e Especialista em Educação a Distância (2010). Atualmente é professor titular do Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense. Possui larga experiência profissional em TI com ênfase em sistemas operacionais, infraestrutura de TI, tendo atuado principalmente nos seguintes temas: softwares para engenharia clínica, software médicos, imagens médicas, infraestrutura de TI e redes de computadores. Trabalhando atualmente com computação em dispositivos móveis na plataforma Android.

<http://lattes.cnpq.br/8031243494666833>

TICs

A

Introdução, histórico e tipo de sistemas operacionais

Unidade A
Sistemas Operacionais

UNIDADE A

INTRODUÇÃO, HISTÓRICO E TIPO DE SISTEMAS OPERACIONAIS

Introdução

Um sistema informatizado qualquer pressupõe a existência de um ou mais hardwares e de um ou mais usuários. Até certo ponto, a existência da informática justifica-se na existência destes dois personagens. No entanto, para que possa haver interação entre o hardware e o usuário, precisamos de um intermediário – o sistema operacional.

Figura A.1 - Visão em camadas
Fonte: do Autor

O sistema operacional oferece ao usuário uma abstração simplificada do hardware, ou seja, facilita e viabiliza seu uso. Para que isto seja possível, o sistema operacional atua gerenciando todo o hardware disponível dentro deste computador – seja ele um simples PC ou um supercomputador da lista TOP500.

Se prestarmos atenção ao usarmos um computador, perceberemos que normalmente fazemos uso de aplicações – também conhecidos como programas. Estas aplicações NÃO SÃO o sistema operacional e sim programas que rodam sobre ele. Este conceito de camadas pode ser visualizado na figura A.1.

Existem diversos sistemas operacionais disponíveis para se trabalhar e o funcionamento de todos é semelhante, do ponto de vista conceitual, mas diferente do ponto de vista de implementação. Em outras palavras o desenvolvedor A desenvolve um sistema operacional de um jeito a, enquanto o desenvolvedor B desenvolve um sistema operacional do jeito b. Ambos fazem a mesma coisa – mas de formas diferentes.

Visão de estudo

Quando pensamos em desenvolvimento ou estudo de um sistema operacional, precisamos considerar qual a abstração que estamos fazendo, ou seja, por qual "ponto de vista" estamos analisando.

O gerente de recursos

Um computador é um conjunto, normalmente complexo de hardwares e periféricos, interconectados. Para fazer com que esta monte de hardware funcione, precisamos de um software que faça um gerenciamento eficiente destes recursos. Veja na citação abaixo a função do sistema operacional como um gerente recursos.

> *" o trabalho do sistema operacional é oferecer uma alocação ordenada e controlada dos processadores, das memórias e dos dispositivos de E/S entre os vários programas que competem por eles" [Tanenbaum, 2010, p19].*

A máquina virtual

Um computador é um conjunto, normalmente complexo de hardwares e periféricos, que trabalha em uma linguagem bastante complexa. Para que pudéssemos fazer uso eficiente e produtivo deste computador precisaríamos conhecer as linguagens de máquina de cada dispositivo – (processador, hd, disquetes, etc). Desta forma, precisamos do sistema operacional para fazer a tradução de toda esta linguagem de máquina para uma linguagem que seja humanamente possível de se trabalhar. Veja a citação abaixo:

> *"o sistema operacional esconde do programador o hardware ... e também oculta muitas coisas desagradáveis relacionadas com interrupções, temporizadores, gerenciamento de memória e outros recursos de baixo nível" [Tanenbaum, 2000. p 19].*

Conceituando

De forma mais conceitual um sistema operacional pode ser definido como "uma camada de software colocada entre o hardware e os programas que executam tarefas para os usuários" [Oliveira, 2008, p 1].

Kernel

Kernel ou núcleo do sistema operacional é o local onde ocorre todo processamento pesado de um SO e via de regra, nós enquanto usuários (programadores ou não) não temos acesso a esta área. Para que se possa trabalhar, faz-se necessário uma interface de comandos que poderá ser em modo gráfica ou modo texto.

a) modo texto: ambiente em que temos apenas uma shell – também conhecido como prompt de comandos. Neste modo, toda manipulação é feita por comandos digitados e executados pelo teclado. Não há botões ou mouse. A shell é largamente conhecida em sistemas operacionais Unix-like.

b) modo gráfico: ambiente em que interagimos com o computador em um ambiente de trabalho gráfico, com botões, ícones, normalmente através de um mouse. Este ambiente se popularizou com os sistemas operacionais da família Windows. Outros sistemas, inclusive Unix-like, também dispunham deste modo de trabalho.

Salientamos que tanto a shell quanto o ambiente gráfico não fazem parte do kernel. Funcionam como uma aplicação que roda sobre o núcleo do sistema. Este conceito pode ser visto na figura A.2 que mostra um ambiente muito comum para sistemas operacionais da família GNU/Linux, em que podemos escolher qual ambiente gráfico ou shell nos adequamos mais.

Figura A.2 - Shell e Ambiente Gráfico
Fonte: do Autor

História

Desde Charles Babbage (1792-1871) até os dias atuais, os computadores e os sistemas operacionais tem passado por evoluções e revoluções significativas. Nesta parte deste artigo, iremos apresentar um pouco desta evolução.

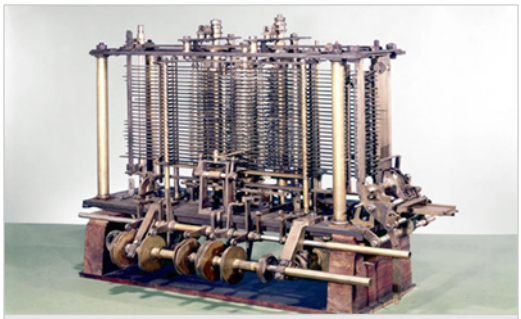

Figura A.3 - Motor Analítico, 1871
Fonte: Site da Science Museum

Saiba mais:

Charles Babbage: e Ada Lovelace

<http://pt.wikipedia.org/wiki/Charles_Babbage>; <http://pt.wikipedia.org/wiki/Ada_Lovelace>

Primeira Geração (1945-1955) – Válvulas e painéis de conectores

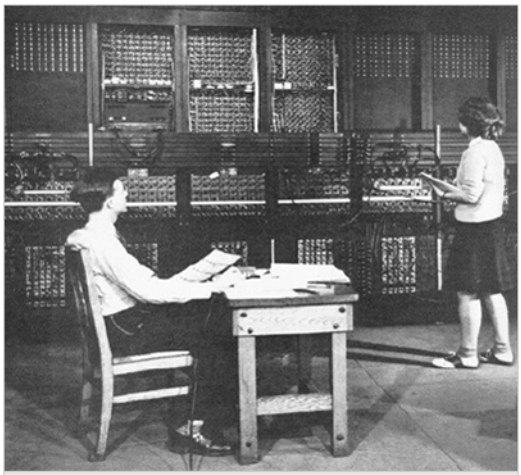

Figura A.4 - Eniac, 1946
Fonte: Site da Columbia University

Os computadores de primeira geração não tinham um sistema operacional propriamente dito. A sua programação (e utilização) era feita diretamente por cabos e conexões – ou seja, o programador precisava conhecer a linguagem de máquina do computador em questão. O principal representante desta geração de computadores é o Eniac. No final da década de 50, estes programas passaram a ser gravados em cartões perfurados, permitindo ao programador executar seus programas com mais facilidade.

Segunda Geração (1955-1965) – Transistores e sistemas de lote

Com o surgimento do transistor, os computadores reduziram de tamanho e passaram a ser fabricados e vendidos – somente grandes empresas poderiam adquirir sistemas deste tipo. Os programas eram escritos em papel e registrados em cartões perfurados – que poderiam ser lidos e executados diversas vezes pelos computadores. Mas para executar os programas criados, alguém deveria inserir os cartões (na sequência correta) o computador então fazia seus cálculos e o resultado era impresso e entregue ao programador novamente.

Ainda nesta geração, algumas modificações foram feitas para "otimizar" o tempo de máquina desocupada, dando origem ao sistema de processamento em lote (batch system) – termo que é usado até hoje em determinados tipos de programas.

O processamento em lote:

- programador escreve o programa e grava em cartões perfurados;
- cartões são lidos por um computador "barato" IBM-1401 e gravados em fita magnética;
- operador rebobina a fita magnética e leva até o computador "caro" IBM-7094 onde o programa é então executado;
- Os resultados obtidos com esta computação são gravados em outra fita magnética e levados a outro IBM-1410 onde os dados são lidos e impressos;
- resultado é entregue ao programador.

Nestes sistemas de computação as funções do sistema operacional acabam sendo executadas pelo operador de máquina, que faz o controle dos cartões, fitas e resultados.

Figura A.5 - IBM 7094, 1962
Fonte: Site da Columbia University

Terceira Geração (1965-1980) – Circuitos Integrados e multiprogramação

Nesta época, a eletrônica havia melhorado bastante e os transistores sofreram uma drástica miniaturização, sendo desenvolvidos em circuitos integrados, causando uma melhoria significativa no desenvolvimento de hardware de comutadores. Desta foram as grandes empresas desenvolveram maquinas mais elaboradas como o 360 da IBM, que tinha pretendia ser uma máquina que atendesse desde grandes cálculos científicos até cálculos comerciais. Estes equipamentos não chegaram a dominar o mercado, mas importantes conceitos da computação atual foram incusos no hardware/sistema operacional como por exemplo a multiprogramação e o compartilhamento de tempo.

Saiba mais:

Circuito Integrado

<http://pt.wikipedia.org/wiki/Circuito_integrado>

Nesta época um consórcio de gigantes como o MIT, Bell Labs e General Eletrics procuraram desenvolver um audacioso sistema de computação centralizada – designado por MULTICS. O conceito era ter um sistema computadorizado central, que atendesse simultaneamente (por compartilhamento de tempo) centenas de usuários espalhados. Este modelo era inspirado no sistema de distribuição elétrica, em que você conecta um aparelho na tomada e usa. O objetivo era prover serviço de computação a toda uma cidade. O projeto não deu certo, mas introduziu centenas de conceitos importantíssimos à computação atual.

Figura A.6 - IBM 360, 1971
Fonte: Site da Columbia University

Paralelamente ao gigante MULTICS, micro-computadores foram surgindo no mercado de circuito integrados de menor custo como o PDP-1 da empresa DEC. Ainda no fim da década de 60, Ken Tompson, cientista da computação (aqui já existiam profissionais de TI) e participante do projeto MULTICS, desenvolveu uma versão monousuário e simplificada deste, que mais tarde daria origem ao UNIX.

Quarta Geração (1980-atual) – Computadores pessoais

Como a tecnologia eletrônica avançava a uma velocidade fantástica, surgiu a tecnologia LSI (Large Scale Integration), que possibilitava construir chips com milhares de transistores. Com estes chips, agora em placas de circuito bem menores que nas gerações anteriores, alguns desbravadores iniciaram a saga do computador pessoal e dos sistemas operacionais.

Destacam-se a empresa Microsoft, com o sistema operacional MS-DOS em processadores da família x86 da Intel; e a empresa Apple com o sistema operacional MacOS maquinas Macintosh com processadores PowerPC da Motorola.

Figura A.7 - IBM-PC e Apple Macintosh, 1984
Fonte: Site da Columbia University

Saiba mais:

Filme Piratas do Vale do Silício, direção de Martyn Burke

Sistemas operacionais desta geração de maquinas, denominados computadores pessoais buscam ter uma interface amigável e serem de fácil utilização – até porque são direcionadas à um público que normalmente não é da área de TI. No entanto os sistemas operacionais de grande porte, para máquinas com maior poder de processamento, continuaram a evolução e também tem sua fatia de mercado.

Tipos de sistemas operacionais

Dado a diversidade de sistemas operacionais disponíveis no ramo de TI, as vezes torna-se necessário fazer uma classificação destes sistemas, que facilite a uma especificação para algum equipamento.

A classificação não é determinante, embora facilite o entendimento de alguns aspectos relacionados à sistemas operacionais.

Supercomputadores

São sistemas operacionais dedicados a supercomputadores utilizados normalmente para aplicações científicas, como simulações e cálculos avançados.

Saiba mais:

Lista TOP 500

<http://www.top500.org/lists/2010/11 >

Os principais exemplares destes tipos de sistema operacional são:

- GNU/Linux
- CNK/SLES9
- CentOS
- AIX
- Windows HPC 2008

Grande porte

São os sistemas operacionais para computadores de grande porte – tipo mainframe. Máquinas deste tipo são encontradas em grandes corporações para processamento de grandes quantidades de dados. Ex: BANRISUL

Os principais exemplares deste tipo de sistemas são:

- OS/390
- OS/360
- GNU/Linux

Servidores

São os sistemas operacionais para máquinas com alto desempenho orientadas a aplicações de médias e pequenas empresas. São computadores facilmente encontrados no mercado, normalmente baseadas na plataforma x86.

Os principais exemplares deste tipo de sistemas são:

- GNU/Linux
- Windows 2008 Server
- Unix HP-UX

Computadores pessoais

São os sistemas operacionais para máquinas com desempenho normal e orientadas a aplicações residenciais. Normalmente baseadas na plataforma x86, podendo ser multiprocessados ou não.

Os principais exemplares deste tipo de sistemas são:

- Windows (diversas versões)
- GNU/Linux
- Mac OS Snow Leopard;

Sistemas Embarcados

São os sistemas operacionais para máquinas de baixo desempenho, normalmente com uma função bem definida como por exemplo celulares ou microcontrolados em geral.

Os principais exemplares deste tipo de sistemas são:

- Windows Mobile
- GNU/Linux – na versão android por exemplo
- Simbiam

Sistemas de tempo real

São os sistemas operacionais para máquinas de tempo real, normalmente usadas em simuladores

Os principais exemplares deste tipo de sistemas são:

- VxWorks
- QNX

Síntese

- o sistema operacional é um programa de certa complexidade, responsável pelo controle (que precisa ser eficiente) de todo o hardware disponível no computador além de servir de interface amigável para o usuário;
- visão top-down: ver o sistema operacional como máquina virtual – que esconde a verdade sobre hardware do usuário/programador.
- Visão bottom-up: ver o sistema operacional como um gerente de recursos – que aloca de forma eficiente e ordenada todos os recursos que um computador possui;
- Primeira geração:
  - computadores gigantes, baseados em válvulas.
  - Não há sistemas operacionais
  - operação/programação diretamente nos cabos e conexões;
- Segunda geração:
  - computadores médios – com baixo poder de processamento, baseados em transistores;
  - Não há sistemas operacionais - figura do operador que faz o “transporte” das fitas;
  - operação/programação por cartões perfurados/fita magnética;
  - princípios ainda vigentes como execução em lote; fila de entrada, etc.
- Terceira geração
  - computadores de vários tamanhos – baseados em circuitos integrados;
  - boom da informática;
  - surgimento de diversos sistemas operacionais e conceitos fundamentais
  - projeto MULTICS
  - primeira versão UNIX;
  - IBM OS/360
  - multiprogramação, timesharing,
- Quarta geração
  - geração do PC – baseado em LSI
  - melhoramento da usabilidade do sistema operacional
  - surgimento da Apple, Microsoft e outras grandes empresas e produtos

Atividades

1. Como primeira atividade relacionada a nossa disciplina, faça um relato no fórum da unidade sobre a sua experiência com um sistema operacional antigo – preferencialmente o mais antigo que você tenha usado. Além disso, acompanhe neste mesmo fórum os relatos dos colegas e comente-os.

Procure relatar sobre dificuldades, problemas, facilidades, as novidades trazidas na época...

TICs

B

Instalação de Sistemas Operacionais e Gerenciamento de usuários

Unidade B
Sistemas Operacionais

UNIDADE B

INSTALAÇÃO DE SISTEMAS OPERACIONAIS E GERENCIAMENTO DE USUÁRIOS

Introdução

Para que um um computador seja útil a um usuário qualquer, seja programador ou usuário comum, precisamos fazer com que este computador "inicialize com sistema operacional". Uma das formas de fazer um sistema operacional inicializar em um computador é instalar este na máquina – uma vez instalado, o SO ficará "residente" na máquina e funcionará por tempo indeterminado (até que haja alguma falha).

Nesta unidade faremos um estudo sobre os processos e procedimentos envolvidos em uma instalação de um um sistema operacional. Além da instalação, serão vistos os procedimentos de gerenciamento de usuários.

Para este fim, será utilizado o Sistema Operacional Debian GNU/Linux, licenciado sobre GPL e considerado um sistema operacional universal.

Processo de BOOT

Para iniciar, o computador precisa "bootar"[1] (lê se butar) e "BOOT" é um termo usado para designar a carga de qualquer sistema operacional em um *hardware*.

Após os testes iniciais executados pelo POST – Power On Self Test do hardware, o computador procura por uma unidade com *flag* de inicialização (boot), esta unidade poderá ser um HDD; um CD/DVD; um bom e velho disquete ou até mesmo as recentes *pendrives*. Nestas unidades de disco, deverão conter pelo menos um programa instalador, que funcionará como um *wizard*, orientando a instalação do sistema operacional.

Vale salientar que cada fabricante/distribuidor de sistema operacional cria e distribui seu próprio instalador, desta forma, sempre encontraremos diferenças e particularidades entre estes programas. No entanto, os passos elementares serão sempre muito semelhantes.

Para que o boot seja possível, o computador precisa ser informado quais são as prováveis unidades de inicialização – esta configuração é feita no setup[2].

Saiba mais:

Hardware Manual Completo, Carlos Morimoto, disponível em:

<http://www.hardware.com.br/livros/hardware-manual/configuracao-bios-setup.html>

1. Outros termos usados : "dar boot", "inicializar"
2. Também conhecida por BIOS;

Ao encontrar o disco inicializável, o computador carrega este boot e iniciará os passos da instalação

Atenção:

Todos os procedimentos deste curso serão executados em máquina virtual. Caso sejam feitos em um computador real, lembre-se de fazer uma cópia de segurança de seus dados particulares.

Dicas:

Para fazer esta instalação será necessário o uso de um disco de instalação do Debian 6 – codinome Squezze que pode ser baixado do site do projeto. Diversos meios de obtenção do Debian estão disponíveis mas sugerimos que seja baixado e gravado pelo menos o DVD-1.

Maiores informações em: *<http://www.debian.org/distrib/>* - sugerimos que seja utilizado o DVD.

Instalação

Requisitos para instalação

- Disco de instalação do Debian;
- Máquina virtual criada

Primeiros passos

Selecione o DVD como unidade de boot preferencial, para isso, durante a tela de inicialização (figura B.1) pressione a tecla "ESQ" para menu de boot - aqui vamos precisar de bastante reflexo porque esta tela passa rápido.

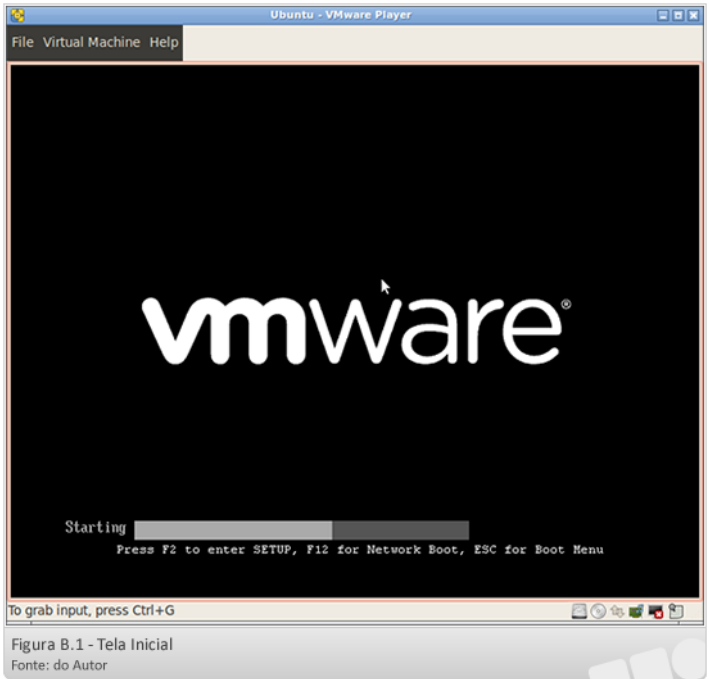

Figura B.1 - Tela Inicial
Fonte: do Autor

Após isto, conforme pode ser visto na figura B.2, escolhemos o CDROM como origem de boot e tecle enter.

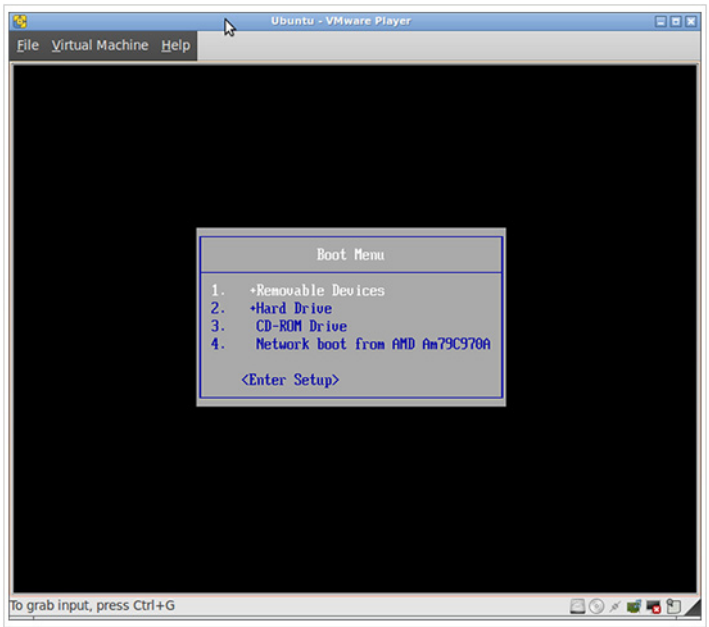

Figura B.2 - Menu de Boot
Fonte: do Autor

Após o boot

Após o boot, a máquina é inicializada pelo disco de instalação do Debian apresentando a tela para escolha modo de instalação – figura B.3. Na tela são apresentados dois modos de instalação e opções avançadas. Neste material, usaremos o método normal (*Install*), até porque o método de instalação gráfica (*Graphical install*), apesar de ser mais intuitivo mas é bem mais pesado.

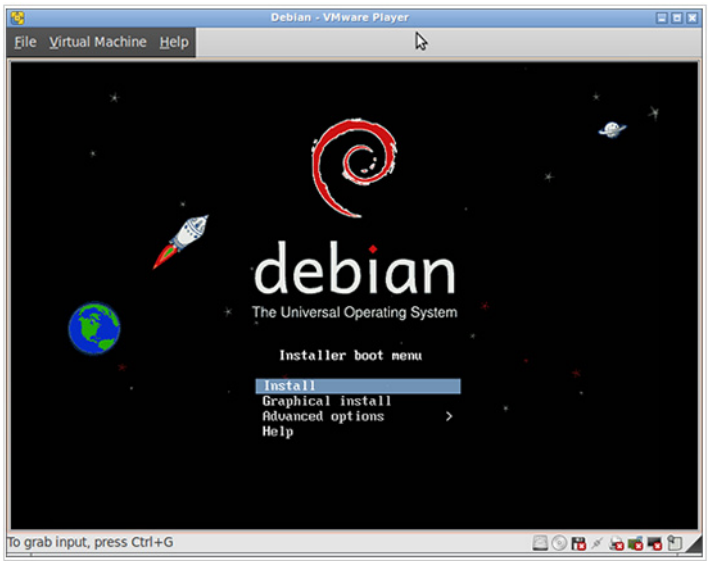

Figura B.3 - Menu de Instalação
Fonte: do Autor

As próximas telas irão coletar algumas informações que definirão diversas características do sistema:

- Escolha de língua: onde escolhemos Português do Brasil; a partir deste momento todas as mensagens e textos estarão em português – não deixe de ler;
- Escolha da localidade: onde escolhemos Brasil;
- Escolha de teclado: onde normalmente escolhemos Português Brasileiro (ANBT) para teclados com tecla 'Ç';

Configurando a rede

Após a coleta de informações sobre sua localidade, língua e correlatos, será necessário fazer a configuração de rede – que poderá será usada durante a instalação do sistema operacional. O instalador tentará fazer a configuração automática da rede.

Caso a configuração automática de rede falhe, sera apresentada uma mensagem de falha e a seguir um questionamento, conforme figura B.4, se a configuração deverá ser feita manualmente.

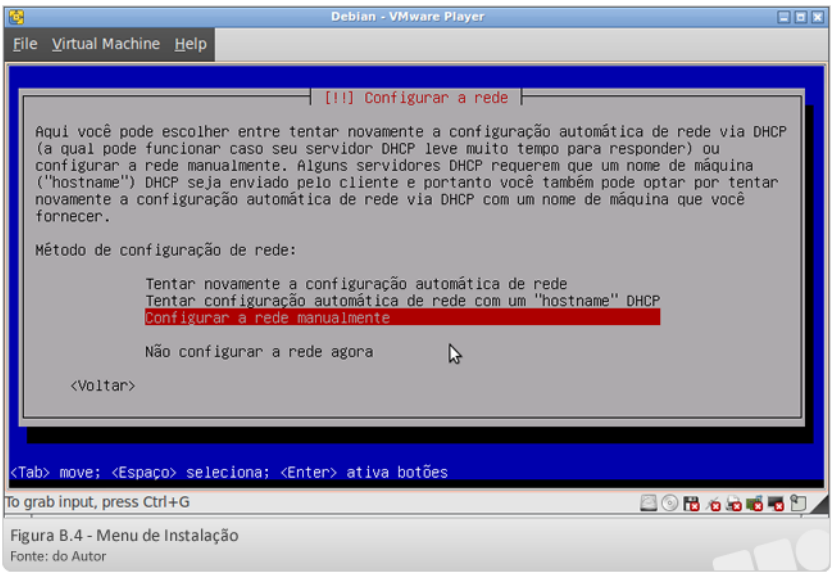

Figura B.4 - Menu de Instalação
Fonte: do Autor

Como estamos fazendo uso de um DVD de instalação, o qual contém todos os pacotes que iremos necessitar, não precisamos configurar a rede neste momento. Podemos escolher a opção 'Não configurar a rede agora'.

Se a configuração automática funcionou, o instalador entrará diretamente na tela de escolha de nome de computador (figura B.5) e domínio, onde iremos definir qual será o nome do nosso computador na rede e domínio de rede – aqui, poderemos aceitar a sugestão do instalador 'debian' e 'localdomain', já que estas informações não irão interferir em nenhuma característica do sistema.

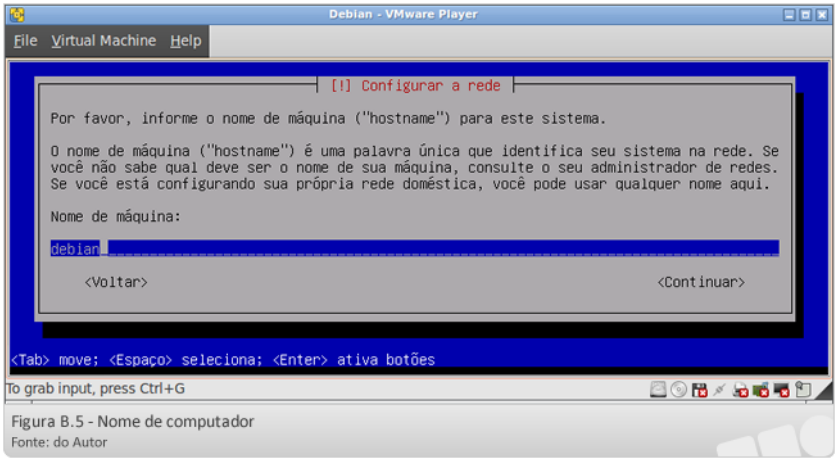

Figura B.5 - Nome de computador
Fonte: do Autor

A utilização de rede permitirá ao instalador baixar e instalar a versão atualizada de todos os pacotes, o que é muito bom do ponto de vista de segurança e estabilidade do sistema instalado, contudo esta método é bastante demorado.

Configurando usuários

Nesta etapa serão configurados os usuários mínimos do sistema que estamos instalando. Vale salientar que a distribuição Debian exige pelo menos dois usuários para funcionar. Um usuário que tem a função de administrar o sistema – conhecido como root e um usuário normal com qualquer nome.

A próxima tela, figura B.6, pede que seja escolhida uma senha para o usuário root. Esta senha deverá ser confirmada na tela subsequente. (não mostrada aqui. O texto apresentado dá algumas dicas de criação desta senha para que o sistema esteja mais protegido.

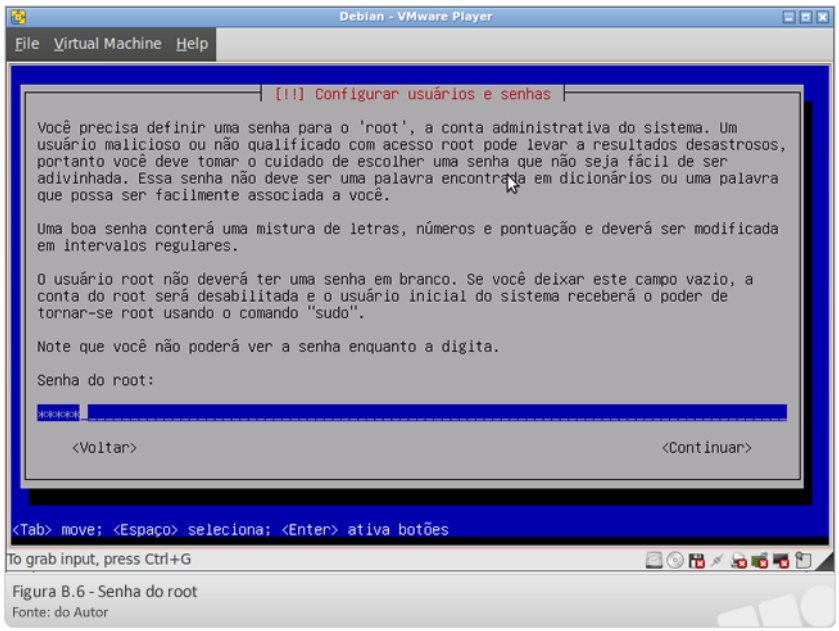

Figura B.6 - Senha do root
Fonte: do Autor

Um invasor mal intencionado poderá tentar descobrir esta senha, por isso faça uso de uma “senha forte” além de não esquecê-la.

Após a criação da senha do root, o instalador exige a configuração de pelo menos um usuário comum, que deverá ser usado para operação normal do computador. São solicitados: nome completo do usuário, nome da conta e senha além de sua confirmação.

Particionamento

O sistema operacional precisará preparar uma área do disco rígido para ficar instalado, além de uma área para dados de usuários. Chamamos este procedimento de particionamento e formatação do disco.

O instalador nos dá uma gama bastante grande de opções de particionamento do disco, no entanto, faremos uso do método mais simples.

Atenção:

Vale lembrar que todos os dados contidos nos discos serão perdidos

a) Método de particionamento:

Aqui deverá ser escolhido o método de particionamento a ser executado no ato da instalação. Os métodos disponíveis são:

- Assistido – usar o disco inteiro: significa que usaremos o wizard de particionamento para o disco inteiro. Faremos uso deste método;
- Assistido – usar o disco inteiro e configurar LVM: significa usar todos os discos presentes no computador e configurar um Volume Lógico – não abordado neste material;
- Assistido – usar o disco inteiro e LVM criptografado: significa usar todos os discos presentes no computador e configurar um Volume Lógico com criptografai – não abordado neste material;
- Manual: significa particionar o disco de forma manual – não abordado neste material

b) Escolha de disco:

Aqui é selecionado o disco que iremos particionar. Como na nossa máquina virtual temos apenas um HD criado, verificamos que apenas um disco é mostrado. Devemos escolher este disco.

c) Escolha de esquema de particionamento:

Como usamos o método assistido, temos 3 opções para o particionamento

- todos os arquivos em uma única partição (para iniciantes) – neste esquema apenas uma partição de uso geral é criada.
- partição /home separada - separa em um partição específica os dados de usuários – não abordado neste material.
- partições /home, /usr , /var e /tmp separadas – separa em várias partições os dados de pastas específicas – não abordado neste material.

Sempre será criado uma partição específica chamada *swap* [3].

d) Finalização:

Uma tela, figura B.7, mostrando as partições a serem criadas é apresentada pedindo sua confirmação.

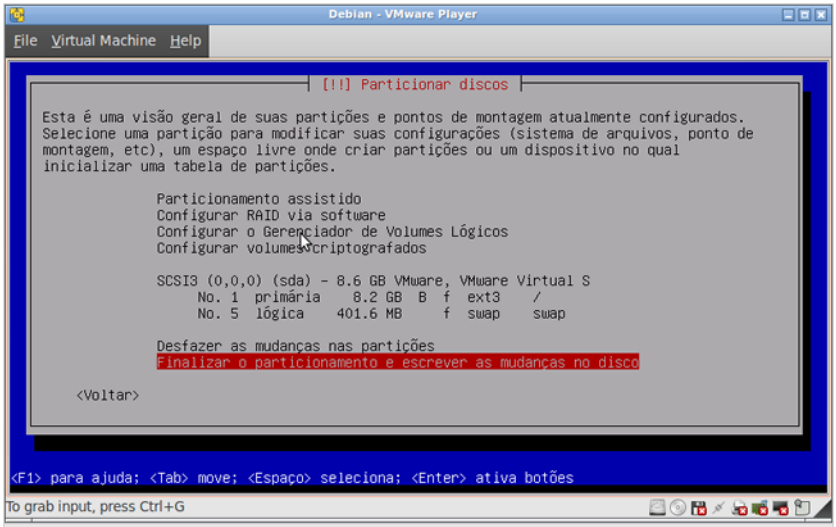

Figura B.7 - Confirmar tabela de particionamento
Fonte: do Autor

e) Confirmação:

Ainda uma nova tela é apresentada perguntando se você tem certeza que deseja particionar e formatar os discos. É a última chance antes de desistir.

3. Área de troca ou memória virtual.

Copiando pacotes

Após o particionamento e formatação o instalador faz uma cópia para o disco de alguns pacotes básicos, fundamentais para o novo sistema. Na sequência, o instalador precisará saber se você quer adicionar outro DVD à lista de repositórios do sistema instalado (o Debian 6 é composto por 8 DVDs, mas precisamos apenas do DVD 1).

Além disso, o instalador quer saber se você irá fazer uso de 'espelhos de rede' para atualização de pacotes em tempo de instalação. Para isto, a rede deverá estar funcionando adequadamente. Sugerimos que não seja usado nenhum espelho de rede para que a instalação não demore muito – dependendo da velocidade da internet isto pode demorar bastante, já que são 1146 pacotes para verificar e atualizar.

Feito as escolhas, o sistema apresenta uma tela para seleção de software – figura B.8.

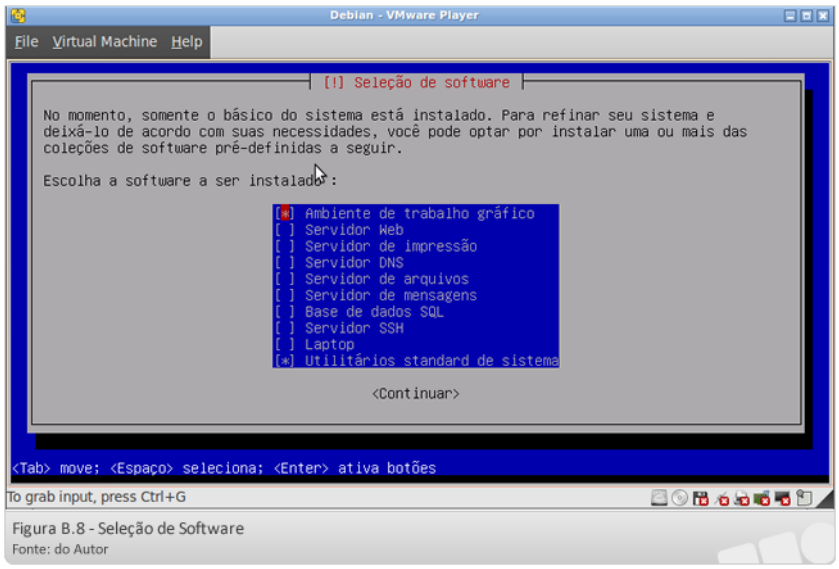

Figura B.8 - Seleção de Software
Fonte: do Autor

Aqui escolhemos que tipo de máquina teremos. A seleção é feita com a barra de espaços e para o nosso sistema, deverão ser selecionados pelo menos o primeiro 'Ambiente de trabalho gráfico' e a última 'Utilitários standards do sistema'.

Tendo feito as escolhas o sistema começará a instalação, lembrando que se estivermos usando espelhos de rede esta instalação poderá demorar bastante. Uma barra de progresso é apresentada conforme a figura B.9.

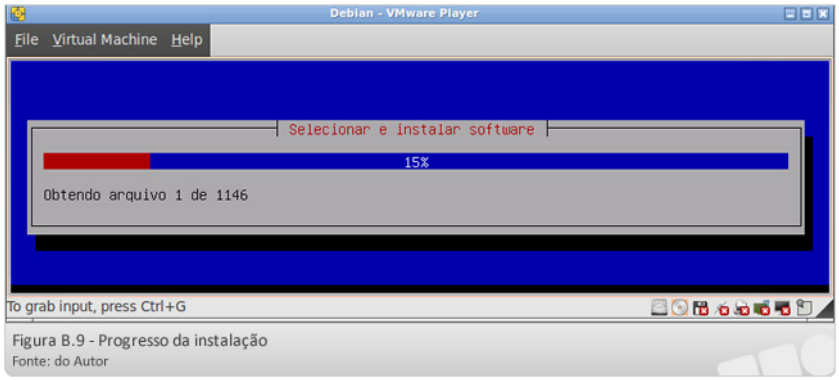

Figura B.9 - Progresso da instalação
Fonte: do Autor

Finalizando a instalação

Ao terminar a instalação dos 1146 pacotes, o instalador irá solicitar, conforme figura B.10, a instalação do carregador de inicialização GRUB. Este utilitário é o responsável pelo boot (inicialização) do sistema

recentemente instalado – logo, se o GRUB não for instalado, o sistema não irá *bootar*. Evidentemente, como qualquer outro utilitário ou configuração do Debian, poderá ser instalado/configurado posteriormente.

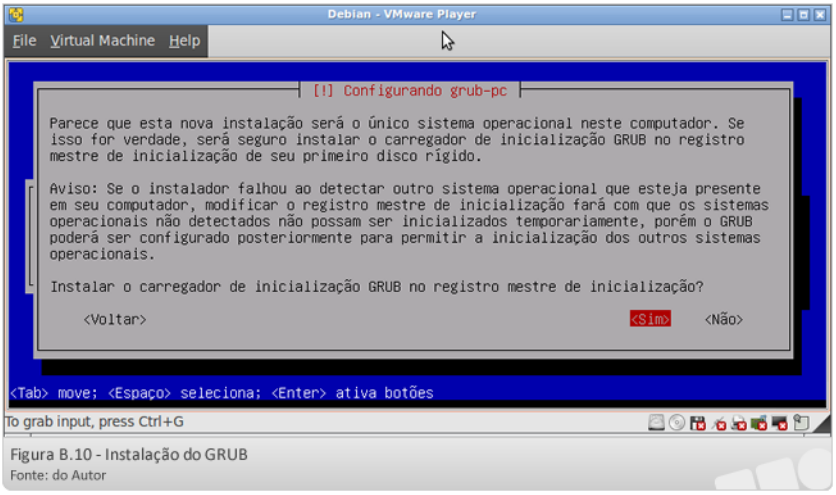

Figura B.10 - Instalação do GRUB
Fonte: do Autor

Gerenciamento de usuários

Gerenciar usuários é uma prática muito importante em qualquer sistema operacional, pois isto garante que todos os usuários terão suas personalizações e proteção dados. Além disso, o uso correto dos usuários – usuário comum para trabalho comum e root EXCLUSIVAMENTE para operações de administração incrementam a segurança do sistema.

Dicas:

De acordo com o manual do kernel, núcleo do sistema operacional nunca devemos usar o nome root em vão: "*Don't take the name of root in vain*"

As operações de gerenciamento de usuários são operações de administração de sistema, logo será necessário que você esteja logado como root. No entanto, como premissa de segurança, o root não loga no ambiente gráfico. Você deverá logar como usuário comum criado no momento da instalação, abrir um terminal e trocar para usuário root com o comando su.

Saiba mais:

Guia Foca GNU/Linux, Capítulo 12 - Comandos para manipulação de contas, disponível em:

< http://www.guiafoca.org/cgs/guia/inic_interm/ch-cmdc.html>

Gerenciamento de usuários

Criando usuários

O comando de criação de usuários é o adduser, ele adiciona um usuário, cria sua pasta de dados e uma senha – além de outras informações de usuário. Nos exemplos abaixo, são criados duas novas contas uma chamada maria e outra joao (observe que joao está sem acento porque é uma conta de usuário e não o nome de uma pessoa).

```
# adduser joao

# adduser maria
```

Removendo usuários

O comando de remoção de usuários userdel removo o usuário escolhido do sistema, podendo apagar também seus dados pessoais.

```
#userdel joao
```

apaga o usuário joao PRESERVANDO sua pasta de arquivos pessoal /home/joao

```
#userdel – maria
```

apaga o usuário maria REMOVENDO sua pasta de arquivos pessoal /home/maria

Gerenciamento de grupos

Grupos são um mecanismo de gerenciamento em que agrupamos usuários por "afinidade" ou característica. Por exemplo, podemos ter um conjunto de usuários que pertençam a um grupo de professores e outro grupo de usuários que pertençam a um grupo de alunos. Alguns destes usuários poderão pertencer a outros grupos. Este agrupamento de usuários será necessário para compreender o funcionamento do gerenciamento de permissões de arquivos da próxima unidade. Esta situação é representada no desenho 11 em que temos um grupo de professores, um grupo de alunos e um grupo de pesquisa formado por um aluno e um professor.

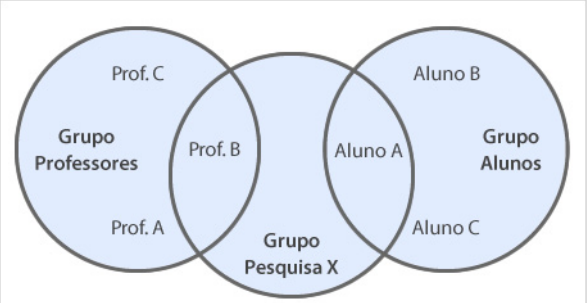

Figura B.11 - Representação de grupos
Fonte: do Autor

a) Criando Grupos

O comando de criação de grupos addgroup cria um novo grupo no sistema. Todos os grupos criados estão vazios.

```
# addgroup professores

# addgroup alunos

# addgroup pesquisax
```

b) Associando Usuários a grupos

O comando que associa usuários a grupos.

```
# adduser aluno1 alunos

# adduser aluno2 alunos
```

c) Removendo Grupos

O comando de remoção de grupos groupdel apaga grupos existentes

```
# groupdel alunos
```

os grupos apagados serão removidos do sistema bem como todas as suas ligações com os usuários.

Síntese

- A instalação de qualquer sistema operacional moderno normalmente é “auto-guiada”;
- As telas contém um texto explicativo que explica o que precisa ser feito;
- O gerenciamento de usuários é muito importante para organizar e proteger um sistema;
- O root (também chamado de administrador) é usuário especial, com poderes sobre todo o sistema– nunca use o administrador para trabalho cotidiano, sob pena de poder danificar o sistema;
- adduser – cria usuário;
- addgroup - cria grupo.

Atividades

1. Instalar o sistema operacional GNU/Linux - Debian 6.0 Squezze;
2. Criar os usuários aluno1, aluno2, professor1, professor2;
3. Criar os grupos alunos,professores, pesquisa;
4. Associar os usuários aos grupos correspondentes;
5. Associar os usuários aluno1 e professor1 ao grupo pesquisa;
6. Entregar no AVA: uma cópia do arquivo /etc/group.

TICs

C

Gerenciamento de Arquivos

Unidade C
Sistemas Operacionais

UNIDADE C

GERENCIAMENTO DE ARQUIVOS

Introdução

O objetivo fim de um sistema informatizado é a produção, é auxiliar o trabalho de pessoas, é viabilizar e agilizar processos, pesquisas, rotinas, etc. Em outras palavras, o computador precisa ser útil e esta utilidade está fortemente ligada a manipulação de arquivos. O objetivo desta unidade é compreender a estrutura e os mecanismos de manipulação de arquivos e diretórios de um sistema operacional Unix-like. Na unidade H serão vistos como o sistema operacional implementa um sistema de arquivos. Todos as explicações e exercícios serão feitos focando o modo terminal de operação. Evidentemente, as mesmas atividades podem ser executadas no modo gráfico, no entanto, não raramente nos deparamos com situações em que não temos ambiente gráfico a disposição para trabalho ou recuperação de algum sistema.

A árvore invertida

A estrutura de dados que organiza os arquivos em um sistema Debian é de árvore invertida, onde tudo vem de uma raiz e se expande entre NÓS (plural de nó), que correspondem aos diretórios, e terminam em folhas, que correspondem aos arquivos. A figura C.1 procura representar esta árvore.

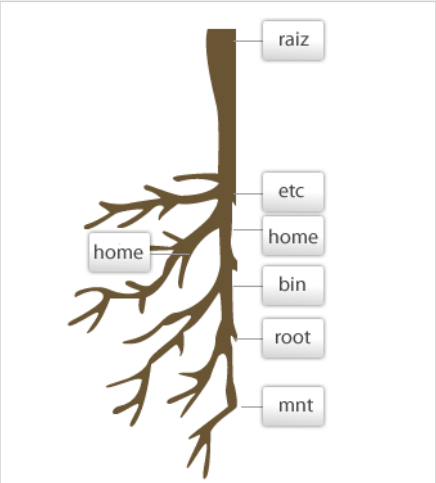

Figura C.1 - Árvore invertida
Fonte: do Autor

Vale salientar que a estrutura de diretórios de um sistema Unix-like segue uma lógica pré-definida (e bem definida) facilitando a organização dos arquivos de sistema e arquivos de usuários, nesta organização, os arquivos serão armazenados por afinidade ou semelhança de função. Por outro lado esta estrutura não é fechada – se você achar conveniente pode "quebrar" o padrão e adaptar uma nova estrutura, mas faça isto com cuidado e principalmente não esqueça o que foi feito.

Na tabela 1 vemos uma estrutura convencional de diretórios do Debian.

Diretório	Conteúdo
bin	Binários de comandos essenciais
boot	Arquivos estáticos do gerenciador de partida
dev	Arquivos de dispositivos
etc	Configurações de sistema específicas da máquina
home	Diretórios home de usuários
lib	Bibliotecas compartilhadas essenciais e módulos do kernel
media	Contém pontos de montagem para mídias removíveis
mnt	Ponto de montagem para montagem temporária de um sistema de arquivos
proc	Diretório virtual contendo informações do sistema (kernels 2.4 e 2.6)
root	Diretório Home do usuário root
sbin	Binários essenciais do sistema
sys	Diretório Virtual contendo informações do sistema (kernels 2.6)
tmp	Arquivos temporários
usr	Hierarquia secundária
var	Dados variáveis
srv	Dados para serviços fornecidos pelo sistema
opt	Pacotes de aplicativos e programas adicionais

Tabela 1: Árvore padrão Debian
fonte http://www.debian.org

Ao fazermos uso do Debian nossos arquivos de usuários ficarão armazenados sempre na pasta de arquivos pessoais do usuário. Por exemplo, na instalação que fizemos na unidade anterior, foram criados diversos usuários entre eles aluno1 e professor1.

- Usuário professor1 – seus dados ficam armazenados em /home/professor1. Esta informação é lida (e verbalizada) da seguinte forma: “barra home barra professsor1”.
- Usuário aluno1 – seus dados ficam armazenados em /home/aluno1. Esta informação é lida (e verbalizada) da seguinte forma: “barra home barra aluno1”.

Detalhes:

- o “caminho absoluto” parte sempre da raiz, identificado pelo primeiro sinal / (o barra);
- sempre que temos um subdiretório ou algum arquivo dentro de um diretório superior, mostramos esta ligação novo /;

Comandos para diretórios

Para conseguir trabalhar ou manipular arquivos e diretórios em um computador, é preciso saber “andar” por entre os diretórios de forma eficiente.

Trocar de diretório

O comando de troca de diretório é o cd – acrônimo para "call directory" e seu uso é :

- $ cd /var/log ⟶ abre o diretório /var/log, onde estão os logs do sistema;
- $ cd /home/aluno1 ⟶abre o diretório /home/aluno1, onde estão os dados do usuário aluno1. Pode-se dizer entra na casa do usuário aluno1;
- $ cd /home/aluno1/projetos ⟶ tenta abrir o diretório /home/aluno1/projetos (este diretório não existe, logo o shell irá informar o erro);
- $ cd --help ⟶ mostra os detalhes do comando.

Listar arquivos

o comando para ver o conteúdo de um diretório é o ls – acrônimo de "list" e seu uso é:

- $ ls ⟶ mostra a lista de arquivos e diretórios;
- $ ls -l ⟶ mostra a lista de arquivos e diretórios em detalhes;
- $ ls --help ⟶ mostra os detalhes do comando.

Criar diretórios

O comando de criação de diretórios é mkdir – acrônimo de "make directory'" e seu uso é:

- $ mkdir teste ⟶ cria diretório teste dentro do diretórios em que estou.
- $ mkdir /tmp/lixo ⟶ cria diretório lixo dentro da pasta /tmp;
- $ mkdir --help ⟶ mostra os detalhes do comando.

Remover diretórios

O comando para remoção de diretórios é o rmdir – acrônimo de "remove directory". Esta comando exige que o diretório esteja vazio, mas exitem outros comandos de remoção forçada com mesmo com conteúdo.

- $ rmdir teste ⟶ remove o diretório criado anteriormente (deve estar vazio);
- $ rmdir /tmp/lixo ⟶ remove o diretório criado no teste anterior – vale observar que /tmp é preservado, sendo apagado somente o lixo.
- $ rmdir --help ⟶ mostra os detalhes do comando.

Renomear diretórios

O comando para renomear diretórios (é o mesmo para renomear tudo) é o mv – acrônimo para "move". Este comando pode ser usado para modificar o nome de algo como a localização de algo. Seu uso é:

- $ mv teste novonome ⟶ move o diretório teste para novonome;
- $ mv /tmp/lixo /tmp/lixao ⟶ move os diretório lixo para lixao;
- $ mv --help ⟶ mostra os detalhes do comando.

Os comandos acima são fundamentais para a operação de um computador em modo texto – lembre-se que nem sempre teremos um sistema operacional com ambiente gráfico disponível mas o o modo texto é padrão. Além disso, outras o formas de "fazer" podem ser encontradas.

Estes comandos são idênticos para todas as distribuições sofrendo pequenas variações nos sistemas Microsoft®.

Saiba mais:

Guia FOCA Linux; Capítulo 2.3 Diretório, disponível em:

<http://www.guiafoca.org/cgs/guia/inic_interm/ch-bas.html#s-basico-diretorio >

Arquivos

Arquivos são objetos digitais, armazenados em uma mídia qualquer (disco rígido, CDROM, rede, etc,). Arquivos possuem uma razão bem definida para existirem, podem possuir algum conteúdo que faz sentido para algum usuário, por exemplo um relatório de conclusão de curso ou uma fotografia; ou são arquivos de sistema, relacionados ao funcionamento do sistema operacional ou de algum programa.

Nesta parte deste material vamos estudar um pouco mais sobre a manipulação de arquivos.

Manipulação de arquivos

A manipulação de arquivos segue a mesma lógica dos comandos para diretórios, com algumas variações. Os comandos de renomear, mover, remover são idênticos. Comandos específicos para arquivos podem ser:

a) Criar arquivo

Arquivos podem ser criados de diversas formas, sendo as mais comuns:

- $ touch arquivo-teste ⟶ cria um arquivo texto de nome arquivo-teste vazio;
- $ nano arquivo-teste ⟶ cria um arquivo texto de nome arquivo-teste direto no editor de textos;

b) Ver conteúdo

Podemos ver o conteúdo de arquivos de diversas formas, sendo 3 as mais usuais:

- editor de texto: quando fazemos uso de um editor de texto – este método nos permite editar e modificar um arquivo (desde que tenhamos permissão para isso);

```
$ nano /home/aluno1/texto
```

- com o cat: o cat pode ser usado para mostrar o conteúdo de um arquivo de forma estática, ou seja, todas as informações do arquivo são jogadas na tela de forma estática – sem possibilidade de edição.

```
$ cat /home/aluno/texto
```

- com o tail: o comando tail mostra as últimas linhas do arquivo texto. No entanto, usando um modificador chave, podemos ver o arquivo crescendo. Este comando é muito útil na visualização de logs de erro

```
$ tail -f /var/log/messges
```

Permissões de acesso

Todos os sistemas operacionais baseados em Unix, herdaram seu rigoroso (e muito eficiente) método de controle de permissões de arquivos e diretórios. Desta forma, precisamos compreender o funcionamento deste mecanismo.

Premissas:

- todo arquivo ou diretório estará associado a um usuário e a um grupo de usuários;

- todo arquivo ou diretório tem permissões explicitamente declaradas para seu usuário, seu grupo e para outros usuários que não são nem o dono nem parte do grupo;
- as permissões são:
  - r – para Read (leitura)
  - w – para Write (escrita)
  - x – para eXecute (execução)

Vendo as informações

Para ver as permissões de arquivos e diretórios em aluma pasta, precisamos executar o comando "ls -l". este comando irá retornar a lista de arquivos e diretórios mostrando as permissões e os usuários do arquivo. Como no exemplo abaixo:

```
drwxr-xr-x     3 aluno1 aluno1  4096 2010-03-10 17:27  anjuta

-rwxr-xr-x     1 aluno1 aluno1  6392 2010-06-02 16:43  a.out

-rw-r--r--     1 aluno1 aluno1    73      2010-06-02 16:43    teste.c

drwxr-xr-x     3 aluno1 alunos  4096 2011-04-25 09:34  enquete

-rwxr--r--     1 aluno1 alunos   892      2010-03-20 00:15    fila.py
```

No trecho de código acima vemos 5 objetos e há muitas informações a serem identificadas:

- primeiro campo – o primeiro campo identifica se o objeto é um arquivo (-) ou um diretório (d);
- segundo campo – permissões para usuário, grupo e outros – sempre em conjunto de 9 sinais;
- terceiro campo - contagem de *hard links*[1] ;
- quarto e quinto campos – usuário e grupo do arquivo ou diretório;
- sexto campo – tamanho do arquivo; se for um diretório informa quantos bytes usamos para armazenar o nome do diretório em disco;
- sétimo campo – data e hora da criação do objeto;
- oitavo campo – nome do objeto.

Dono e grupo

Todo arquivo ou diretório pertence a algum dono e a algum grupo. Desta forma podemos definir quem é o proprietário do arquivo e qual grupo ele irá pertencer, para depois definir qual a permissão que cada um destes participantes terão sobre o arquivo.

Como visto no item anterior, o dono e o grupo são mostrados pelo comando ls -l – noquarto e quinto campo respectivamente. É possível (e muitas vezes necessário) modificar o dono de algum arquivo ou diretório.

- O comando de troca de dono é o *chown* – acrônimo de "*change owner*" ou troca de dono.
- O comando de troca de grupo é o *chgrp* – acrônimo de "*change group*" ou troca de grupo.

1. Esta informação não é importante no momento

Para executar estes testes, basta criar (como root) uma pasta chamada teste na raiz;

- Logar como root

```
$ su – root
```

- Criar a pasta teste

```
# mkdir /teste
```

- Verificar quem é o dono da pasta

```
# ls -l /teste
```

- Trocar o dono para aluno

```
# chown aluno1 /teste
```

- Trocar o dono da pasta para pesquisa

```
# chgrp pesquisa  /teste
```

Entendendo as permissões

As permissões determinarão o tipo de acesso que cada usuário terá em determinado arquivo ou diretório. Após o identificador de arquivo/diretório, teremos 3 campos para cada tipo de usuário.

U	G	O
rwx	rwx	rwx

Onde:

- o primeiro conjunto de 3 identificadores rwx de permissão determina a forma de acesso para o usuário dono do arquivo;
- o segundo conjunto de 3 identificadores rwx de permissão determina a forma de acesso para o grupo do arquivo;
- o terceiro conjunto de 3 identificadores rwx de permissão determina a forma de acesso para os usuários que não estão no grupo e não é o dono do arquivo.

Cada permissão pode ser incluída ou removida (se preferir ativada/desativada), modificando a forma de acesso. Esta modificação de permissão se dá com o comando “chmod” - acrônimo de “change mode” ou mudar modo de acesso.

De forma mais prática, vamos criar um arquivo teste qualquer – vazio mesmo.

```
$ touch teste
```

Ao darmos o comando ls -l verificamos as permissões deste arquivo que são as seguintes:

```
-rwxr-xr-x
```

Dividindo os códigos vemos que:

- “-” indica que é um arquivo;
- “rwx” - que o usuário tem permissão de leitura, escrita e execução;
- “r-x” - indica que o grupo tem permissão somente de leitura e execução;
- “r-x” - indica que outros tem permissão somente de leitura e execução;

Vamos supor que este arquivo é um arquivo muito confidencial... precisamos fazer algumas modificações:

a) Bloqueando acesso aos outros: ⟶ $ chmod o-rwx teste

desta forma dizemos outros perdem leitura escrita e execução – somente dono e grupo poderão ler/escrever/executar o arquivo;

b) Bloqueando acesso ao grupo e outros: ⟶ $ chmod go-rwx teste

desta forma dizemos que grupo e outros perdem leitura escrita e execução – somente o dono pode ler/escrever/executar o arquivo;

c) Liberando somente leitura e escrita para o grupo: ⟶ $ chmod o+rw teste

desta forma dizemos que o grupo ganha leitura escrita;

d) Liberando somente execução para grupo e outros:

⟶ aqui precisamos retirar as permissões dadas anteriormente;

⟶ $ chmod go-rw teste

⟶ $ chmod go+x teste

⟶ assim dizemos que grupo e outros perdem leitura/escrita e ganham execução;

Saiba mais:

Guia FOCA Linux; 10.10 Modo de permissão octal, disponível em:
<http://www.guiafoca.org/cgs/guia/inic_interm/ch-perm.html#s-perm-octal>

Permissões em diretórios

O mecanismos de manipulação de permissões para diretórios é o mesmo de arquivos. Contudo a função de cada permissão é diferente.

- R – permite listar conteúdo do diretório. Caso esteja negado, o diretório não mostrara se u conteúdo.
- W – permite gravar conteúdo dentro do diretório. Caso esteja negado, o sistema operacional não vai permitir a criação de arquivos ou diretórios dentro do diretório.
- X – permite entrar no diretório. Caso esteja negado, o sistema operacional bloqueia a abertura ou a entrada dentro do diretório. Esta propriedade é facilmente confundida com a permissão R.

Vale salientar que o usuário root está acima destas permissões, tanto que em sistema Unix, é conhecido como "superusuário". Esta característica confere ao root poder total sobre o sistema.

Atenção:

Não use o usuário administrador para funções corriqueiras. A conta de root é dedicada à administração do sistema

Divisões lógicas do disco

Ao instalarmos um sistema operacional ou ao adicionarmos um disco ao nosso computador, precisamos particionar e formatar este disco. Na verdade, não conseguimos usar um disco, seja ele um HD, um CRROM, uma pendrive ou qualquer outro tipo de unidade de armazenamento sem que estejam devidamente preparados com o particionamento.

Mas o que significa particionar e formatar?

Partições

Particionar um disco significa criar áreas dentro de um disco físico. Cada área será utilizada terá alguma função no nosso sistema. Em sistemas Unix-like precisamos de no mínimo duas partições, sendo:

- principal: é uma partição que pode ocupar quase todo o disco, nela estarão os arquivos de sistema, usuários e tudo o que o computador precisa para funcionar. É referenciada pelo sistema Debian como / (chamamos isto de raiz);
- swap: área especialmente reservada para memória virtual do sistema operacional.

No entanto, podemos criar estruturas de particionamento diferentes desta mínima procurando organizar nosso computador. Por exemplo:

- principal: com arquivos de sistema, referenciada como /
- swap: com memória virtual;
- home: com arquivos de usuários, referenciada como /home.

Ou ainda

- principal: com arquivos de sistema, referenciada como /
- swap: com memória virtual;
- home: com arquivos de usuários, referenciada como /home.
- Outra: com outras coisas que queremos separar – tipo

Formatação

A formatação é o procedimento de criar um sistema de arquivo em uma partição. O sistema de arquivo é o jeito pelo qual os arquivos são registrados no disco – cada sistema tem o seu formato. Em outras palavras, um sistema de arquivos do Linux, é diferente do sistema de arquivos do DOS e assim por diante. Veremos mais sobre sistemas de arquivos na unidade H.

Atenção:

Uma partição contém somente uma um sistema de arquivos por vez

Particionando e formatando

Para este exercício, será necessário um disco vazio – que pode ser acrescentado na máquina virtual.

- Desligue o computador virtual e selecione a máquina que estamos usando;
- Clique em "*Edit virtual machine settings*";
- clique no botão "*ADD*";
- escolha "*Hard Disk*" e clique *Next*;
- escolha "*Create new virtual disk*" e clique em *Next*;
- escolha "*SCSI (Recommended)*" e clique em *Next*;
- escolha "*Split virtual disk into multiple files*", matenha o tamanho em 8,0 GB e clique em *Next*;
- não modifique o *File Name* e clique em *Save*.

Agora ao iniciar a máquina virtual, você terá um novo disco para experimentarmos.

Requisitos:

- uma unidade e disco sem dados;
- acesso de root – parar isso, faça *logon* normalmente com usuário, abra um terminal e execute o comando su - root; este comando irá abrir uma sub-sessão com privilégios de administração (observe que o sinal de shell mudará para #);
- paciência e cuidado;
- uma lida rápida no *help* do fdisk (# fdisk -h).

a) Vendo os HD's e partições disponíveis:

- comando # fdisk -l

```
root@debian:~# fdisk -l

Disk /dev/sda: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cilindros of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0005e4df

Dispositivo Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1         996     7993344   83  Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2             996        1045      392193    5  Estendida
/dev/sda5             996        1045      392192   82  Linux swap / Solaris

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cilindros of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

O disco /dev/sdb não contém uma tabela de partições válida
root@debian:~#
```

Analisando a captura de tela acima, verificamos nos destaques sublinhados que foram detectados dois discos, sendo:

- /dev/sda que corresponde ao primeiro disco usado na instalação;
- /dev/sdb que corresponde ao segundo disco recentemente adicionado;

Observa-se também que o disco /dev/sda possui 3 divisões sendo:

- /dev/sda1 – partição primária com formato ext3 de 7993344 Bytes;
- /dev/sda2 – partição estendida de 392193 Bytes;
- /dev/sda3 – partição swat de 392192 Bytes

Observa-se também que o disco /dev/sdb não possui partições. Nosso objetivo agora é criar partições. A título de exemplo, vamos criar 3 partições, sendo:

- 1 ext3 com 3GBytes – padrão para sistemas linux;
- 1 ext4 com 3GBytes – evolução do padrão ext3 tenderá a substituir o ext4;
- 1 fat32 com o resto de bytes – padrão para discos que precisam ser compartilhados como pendrives e carões de memória. Este padrão não tem controle de permissões de acesso aos arquivos.

b) Particionando

entrar no particionador:

fdisk /dev/sdb ⟶ Comando (m para ajuda):

Atenção:

Não particionar o disco /dev/sda – caso contrario nosso sistema se perderá

Ao entrar no particionador, recebemos o prompt Comando (m para ajuda): onde podemos digitar m para obter ajuda. Neste pequeno guia, podemos ver, por exemplo, usamos 'q' para sair.

1. criar a primeira partição: comando n
   - A letra 'n' irá criar nova partição e o particionador pede o tipo que queremos primária ou estendida. - escolhemos partição primária;
   - O prompt nos pergunta qual o número da partição - como é a primeira, escolhemos 1;
   - o prompt nos pergunta o primeiro cilindro (onde a partição começa) – como é nossa primeira partição, escolhemos cilindro 1;
   - o prompt nos pergunta qual o último cilindro (até onde nossa partição vai) – como queremos uma partição de 3GBytes, informamos o TAMANHO da partição com o comando +3G;
   - neste ponto nossa primeira partição está criada.
2. ver a partição criada: comando p
   - como podemos ver na captura de tela abaixo a partição /dev/sdb1 foi criada;

```
Comando (m para ajuda): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cilindros of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xc4cc3d31

Dispositivo Boot      Start         End      Blocks   Id  System
/dev/sdb1               1         393     3156741   83  Linux
```

3. criar a segunda partição: comando n
   - A letra 'n' irá criar nova partição e o particionador pede o tipo que queremos primária ou estendida. - escolhemos partição primária;
   - O prompt nos pergunta qual o número da partição - como é a segunda, escolhemos 2;

- o prompt nos pergunta o primeiro cilindro (onde a partição começa) – como é nossa segunda partição, devemos começar de onde a primeira terminou, para isto basta teclarmos enter e o particionador define o primeiro cilindro vago como incio da nossa partição.
- o prompt nos pergunta qual o último cilindro (até onde nossa partição vai) – como queremos uma partição de 3GBytes, informamos o TAMANHO da partição com o comando +3G;
- neste ponto nossa segunda partição está criada.

4. Ver as partições criadas: comando p

- como podemos ver na captura de tela abaixo as partições /dev/sdb1 e /dev/sdb2 foram criadas;

```
Dispositivo Boot     Start       End      Blocks   Id  System
/dev/sdb1               1       393     3156741   83  Linux
/dev/sdb2             394       786     3156772+  83  Linux
```

5. criar a terceira partição: comando n

- A letra 'n' irá criar nova partição e o particionador pede o tipo que queremos primária ou estendida - escolhemos partição primária;
- O prompt nos pergunta qual o número da partição - como é a terceira, escolhemos 3;
- O prompt nos pergunta o primeiro cilindro (onde a partição começa) – como é nossa terceira partição, devemos começar de onde a segunda terminou, para isto basta teclarmos enter e o particionador define o primeiro cilindro vago como incio da nossa partição.
- O prompt nos pergunta qual o último cilindro (até onde nossa partição vai) – como esta é a última partição, queremos que ela ocupe todo o resto do disco, basta teclarmos enter. TODO o espaço disponível é alocado a nossa terceira partição.
- neste ponto nossa terceira partição está criada.

6. Ver as partições criadas: comando p

- como podemos ver na captura de tela abaixo as partições /dev/sdb1, /dev/sdb2 e /dev/sdb3 foram criadas;

```
Dispositivo Boot     Start       End      Blocks   Id  System
/dev/sdb1               1       393     3156741   83  Linux
/dev/sdb2             394       786     3156772+  83  Linux
/dev/sdb3             787      1044     2072385   83  Linux
```

7. Mudando o tipo das partições: comando t

- Todas as partições foram criadas para o mesmo tipo de sistema de arquivos. Isto pode ser verificado pelo código 83 em ID – que corresponde ao padrão ext. Precisamos modificar o ID /dev/sdb3 que deve ser preparada para fat32.
- Após teclar t, o prompt nos pergunta qual o número da partição - como é a terceira, escolhemos 3;
- O prompt nos pergunta o código hexadecimal que irá identificar o tipo que iremos definir. Aqui pressionamos L para ver os códigos possíveis – o particionador tem uma grande variedade de opções para escolha.
- Ao verificar a lista, temos (na primeira coluna) encontramos o código b, que corresponde ao padrão W95 FAT32.
- Ao verificarmos com o comando p temos:

```
/dev/sdb3             787      1044     2072385    b  W95 FAT32
```

c) Formatando

TODAS as partições precisam ser formatadas cada uma com no seu formato correspondente. O Debian possui o utilitário mkfs – acrônimo de "make file system" - que facilita bastante a formatação das partições.

Atenção:

Não formatar partições do disco /dev/sda – caso contrario nosso sistema se perderá

Ainda como root digite mkfs e pressione a tecla TAB – o sistema mostrará todas as formas que o mkfs disponibiliza.

Formatando – comando mkfs

```
# mkfs.ext3 /dev/sdb1
```

- formata /dev/sdb1 com o sistema de arquivos ext3;

```
# mkfs.ext4 /dev/sdb2
```

- formata /dev/sdb2 com o sistema de arquivos ext4

```
# mkfs.vfat /dev/sdb3
```

- formata /dev/sdb3 com o sitema de arquivos fat32

Neste momento as 3 novas partições do nosso segundo disco estão prontas para uso.

d) Usando as partições

Para podermos usar uma partição em um sistema Unix-Like, precisamos fazer um procedimento de montagem, onde um diretório especialmente criado servira de "link" para nossa partição – sempre que quisermos usar a partição X usamos na verdade o diretório em que ela está "montada". A ideia se assemelha a criar um atalho de um diretório para uma partição – sempre que quisermos usar esta partição precisamos fazer a montagem.

Saiba mais:

Guia FOCA Linux; 5.11 Pontos de montagem, disponível em:

<http://www.guiafoca.org/cgs/guia/inic_interm/ch-disc.html#s-disc-pontomontagem >

1. criando os pontos de montagens

- você pode criar um ponto de montagem em qualquer lugar do seu sistema de arquivos, no entanto o Debian sugere sempre o diretório /mnt – acrônimo de "mount";
- dentro de /mnt precisamos criar um diretório para cada partição a ser montada – para fins de exercício, sugiro os seguintes nomes ponto1, ponto2 e ponto3.

```
# mkdir ponto1

# mkdir ponto2

# mkdir ponto3
```

2. montando – comando mount -t <tipo> <origem> <ponto>

```
# mount -t ext3 /dev/sdb1 /mnt/ponto1
```

- significa montar a partição /dev/sdb1 do tipo ext3 em /mnt/ponto1;

```
# mount -t ext4 /dev/sdb2 /mnt/ponto2
```

- significa montar a partição /dev/sdb2 do tipo ext4 em /mnt/ponto2;

```
# mount -t vfat /dev/sdb3 /mnt/ponto3
```

- significa montar a partição /dev/sdb3 do tipo FAT32 em /mnt/ponto3;

- neste momento as 3 partições estão montadas.

3. verificando – comando df

- Como pode ser visto na captura de tela abaixo, as partições foram montadas – este comando é inofensivo;
- Obviamente, podem ser vistas também as partições do primeiro disco /dev/sda
- # df -h ⟶ irá apresentar resultados mais interessantes.

```
root@debian:~# df
Sist. Arq.      1K-blocos     Usad Dispon.  Uso% Montado em
/dev/sda1         7867856  3421416  4046776 46% /
tmpfs              127372        0   127372  0% /lib/init/rw
udev               123032      176   122856  1% /dev
tmpfs              127372        0   127372  0% /dev/shm
/dev/sdb1         3107092    70236  2879020  3% /mnt/ponto1
/dev/sdb2         3107124    70168  2879120  3% /mnt/ponto2
/dev/sdb3         2068328        4  2068324  1% /mnt/ponto3
```

4. Usando

- Por padrão, o ponto de montagem é de uso exclusivo do root – para usuários comuns poderem trabalhar nestas áreas, sugiro que seja criado uma pasta e passar esta para o usuário e/ou grupo que está precisando – trocando o nome do dono/grupo.
- A titulo de exercício, vamos criar uma pasta para o usuário aluno1 grupo alunos dentro de /mnt/ponto1
  - # cd /mnt/ponto1 ⟶ vamos para ponto de montagem específico;
  - # mkdir teste-aluno1 ⟶ cria-se a pasta; se dermos o comando ls -l verificamos que o dono da pasta é o root; precisamos então mudar o dono com o comando chown – acrônimo para “change owner” e chgrp – acrônimo para “change group”
  - # chown -R aluno1 teste-aluno1 ⟶ muda o dono do diretório e de todos os seus subdiretórios;
  - # chgrp -R alunos teste-aluno1 ⟶ muda o grupo do diretório e todos os seus subdiretórios;
  - # ls -la ⟶ para verificar;
    - drwxr-xr-x 2 aluno1 alunos 4096 Jun 2 09:51 teste-aluno1
  - Agora o usuário aluno1 e o grupo alunos poderão usar a pasta teste-aluno1

Síntese

- arquivo é uma forma de armazenar algum conteúdo;
- diretório é um arquivo especial que guarda outros arquivos ou diretórios;
- o sistema de arquivos em Linux é representado em forma de árvore invertida, onde tudo começa na raiz;
- arquivos e diretórios pertencem a um usuário e a um grupo;
  - chown e chgrp
- arquivos e diretórios possuem permissão de acesso atribuídas ao dono, ao grupo e aos outros;
  - chmod
  - UGO ⟶ Usuario (dono), Grupo e Outros;
  - RWX ⟶ Read (leitura), Write (escrita) e X (eXecução)
- um sistema de arquivos deve estar contido dentro de uma partição
  - uma partição só pode ter um sistema de arquivos por vez;
  - podemos criar/modificar as partições de um disco conforme for melhor;

- esta modificação causa a perda dos dados contidos no disco modificado;
    - fdisk
    - mkfs.ext3, mkfs.vfat etc.

Atividades

1. No diretório /var, criar dois novos subdiretórios chamados:

a) pasta-aluno1

- para este diretório, mudar o dono, o grupo para aluno1;
- mudar as permissões para que somente o dono da pasta tenha permissão de uso – bloqueando acesso dos usuários do grupo e outros usuários;

b) pasta-professor1

- para este diretório, mudar o dono, o grupo para professor1;
- mudar as permissões para que somente o dono da pasta tenha permissão de uso – bloqueando acesso dos usuários do grupo e outros usuários;

c) pasta-pesquisa

- para este diretório, mudar o grupo para pesquisa;
- mudar as permissões para que o grupo da pasta possa modificar o conteúdo - bloqueando acesso ao dono da pasta e outros usuários;

d) postar o resultado do comando ls -l na pasta /var;

2. Reparticionar o disco /dev/sdb

a) criar 1 partições primárias 2 GB;

- /dev/sdb1 – como ext3

b) criar 2 partições primária de 3 GB cada;

- /dev/sdb2 como fat32
- /dev/sdb3 como ntfs

c) montar as partições em /mnt

d) postar o resultado do comando df -h

D

Gerenciamento de pacotes e interoperabilidade

Unidade D
Sistemas Operacionais

UNIDADE D

1. GERENCIAMENTO DE PACOTES

Introdução

Um sistema informatizado é composto por diversos aplicativos; são estes aplicativos que tornam o computador útil – e como visto na sessão anterior, o objetivo de um sistema informatizado é a produção (no sentido de viabilizar e agilizar o trabalho do homem). Em outras palavras, para que possamos trabalhar precisamos de aplicativos úteis e à nossa atividade.

Se trabalhamos com textos, precisamos de um editor de textos; se trabalhamos com cálculo estatístico, precisaremos de algum programa específico e/ou uma planilha eletrônica; para design gráfico, editores gráficos; ou seja, para cada atividade haverá uma ou mais ferramentas disponíveis.

O objetivo desta unidade é compreender o mecanismo de instalação/remoção (gerenciamento) de aplicativos (ou programas) em um sistema operacional Unix-like.

Todos as explicações e exercícios serão feitos focando o modo terminal de operação. Evidentemente, as mesmas atividades podem ser executadas no modo gráfico, no entanto, não raramente nos deparamos com situações em que não temos ambiente gráfico a disposição para trabalho ou recuperação de algum sistema.

O que é um pacote

Em sistemas GNU/Linux nos referenciamos aos programas disponíveis para instalação pelo termo pacote. Este termo é utilizado porque um programa, qualquer que seja, é composto por mais de um arquivo – um programa acaba sendo um pacote de arquivos executáveis, de configuração, de manual, de documentação, de fontes e outros mais.

Cada distribuição GNU/Linux tem um conjunto pré aprovado de pacotes para instalação, que irão acompanhar a versão da distribuição. Por exemplo, a distro Debian 6 (de codinome Squeeze) que instalamos em máquina virtual ainda unidade B, tem um montão de pacotes disponíveis. Veja a citação abaixo retirada do site da distribuição:

> *"O Debian GNU/Linux é mais que um simples SO: ele vem com mais de 29000 pacotes contendo softwares pré-compilados e distribuídos em um bom formato, que torna fácil a instalação deles na sua máquina." [Debian, 2011, www.debian.org]*

Como podemos ver, temos vários pacotes disponíveis para uso – obviamente, estes pacotes são distribuídos entre as mais diversas funções. A distro Debian é conhecida por ter um grande conjunto de pacotes prontos para instalação, porém outras distros também fornecem seus pacotes pré compilados.

Podemos ainda instalar programas "não empacotados", direto de algum fornecedor de programas ou programas feitos por nós mesmos, bastando para isto termos os binários ou o código fonte (aqui cada caso é um caso).

DPKG

O dpkg é um programa para gerenciamento de pacotes em linha de comando (modo texto), sempre executado pelo administrador do sistema. Sua sintaxe é bastante simples:

- Instalar ⟶ # dpk -i <nome-do-pacote.deb>
- Remover ⟶ # dpkg -e <nome-do-pacote.deb>
- outras opções ⟶ # dpkg -h

A principal desvantagem do dpkg é a não resolução de dependências, ou seja, se para instalar um pacote X, temos uma dependência do pacote Y a instalação irá falhar. Então precisamos instalar baixar e instalar primeiro Y (resolver a dependência) depois instalar X. Além disso, outra desvantagem é que o pacote instalado precisa ser previamente baixado.

Ainda nesta unidade veremos meios mais eficientes de gerenciamento de pacotes.

Saiba mais:

Guia Foca GNU/Linux, Capítulo 20 - Sistema de gerenciamento de pacotes, disponível em:

<http://www.guiafoca.org/cgs/guia/inic_interm/ch-dpkg.html#s-dpkg-pacotes>

Instalando um pacote com o DPKG

Vamos fazer a instalação do programa JOE, que é um pequeno editor de textos em modo texto e que pode ser obtido em em um repositório Debian e não tem nenhuma dependência a ser resolvida. O programa joe pode ser obtido no link ftp://ftp.br.debian.org/debian/pool/main/j/joe/joe_3.7-2_i386.deb.

Após baixado o pacote ficará guardado no diretório Downloads[1] do usuário logado.

- Abrir um terminal e logar como root[2] ;
- Executar o comando de instalação

```
# dpkg -i  joe_3.7-2_i386.deb
```

- Testar
  - abrir o programa digitando joe no terminal
  - ajuda com CTRL+K + H
  - sair sem salvar com CTRL+C
  - sair salvando CTRL+K+X

Removendo um pacote

Vamos remover o programa joe para fins de exercício – caso você tenha gostado do editor e queria usá-lo, basta fazer a reinstalação.

- A partir de qualquer diretório;
- Acesso de administrador;
- Executar o comando de remoção

```
# dpkg -r  joe
```

- Testar
  - tentar abrir o programa joe – vermos que o mesmo não está mais disponível.

1. Esta é uma característica do Debian 6.0 – versões anteriores não usam este diretório padrão.
2. Obter acesso de root com o comando $ su - root

apt-get

Obter pacotes e resolver dependências manualmente não é uma prática muito simples e muito pouco praticada pelos administradores de sistemas Unix-Like. Já faz algum tempo, sistemas baseados em GNU/Linux oferecem um mecanismo mais avançado e bastante eficiente de instalação de pacotes baseado em repositórios.

Repositórios, também chamados de espelhos de rede online, são servidores que ficam disponíveis na internet em tempo integral, distribuindo pacotes. Para fazermos uso destes repositórios basta configurar nosso computador (que se comporta como um cliente) para "buscar" pacotes no endereço definido.

Configurando o repositório

A configuração de repositórios em nossa máquina instalada com sistema operacional Linux é feita no arquivo /etc/apt/sources.list – como o próprio nome sugere, é onde informamos quais são as nossas fontes de pacotes para o programa apt-get. Para editar o arquivo precisamos de acesso root e um editor de textos qualquer – vamos 'zerar' o arquivo e incluir nossas próprias configurações.

```
# gedit /etc/apt/sources.list
```

```
deb http://ftp.br.debian.org/debian/ lenny main contrib non-free

deb http://security.debian.org/ lenny/updates main contrib non-free
deb-src http://security.debian.org/ lenny/updates main contrib non-free

deb http://volatile.debian.org/debian-volatile lenny/volatile main contrib non-free
deb-src http://volatile.debian.org/debian-volatile lenny/volatile main contrib non-free
```

Em cada linha, temos diversas informações, a saber:

- deb – no início de cada linha, informa que o repositório é do formato debian;
- http://ftp.br.debian.org/debian – indica o servidor que estamos usando. Existem vários no mundo para cada distribuição. A distro Debian mantem uma lista dos principais espelhos de rede em: http://mirror.debian.org/status.html
- lenny (codinome da versão Debian – temos diversas versões online)
  - sarge – debian 4.0
  - lenny – debian 5.0
  - squeeze – debian 6.0
  - stable (irá identificar sempre a ultima distribuição estável – no caso a squeeze)
- main – identifica a sessão de pacotes principais da distribuição
- contrib – identifica a sessão de pacotes que são contribuição da comunidade
  - os principais pacotes (mais frequentemente utilizados) não estão nesta sessão, desta forma, poderíamos omiti-la.
- non-free – identifica a sessão de pacotes não livres disponíveis
  - idem a sessão contrib.

Dois outros servidores estão sendo configurados neste arquivo.

- Servidor security
  - distribui atualizações de segurança
- servidor volatile
  - distribui arquivos temporários como por exemplo 'antivirus database'.

Atualizando o o apt-get

O gerenciador de pacotes apt-get trabalha com um banco de informações locais sobre os pacotes disponíveis e instalados. Sempre que uma modificação é feita no sources.list precisamos, OBRIGATÓRIAMENTE, atualizar este banco de dados local. O comando de atualização é:

```
# apt-get update
```

Este comando faz com que nosso computador conecte ao servidor configurado (no caso ftp.br.debian.org) e baixe a lista de pacotes mais atualizada – assim, o nosso computador fica atualizado, sabendo quais pacotes:

- estão disponíveis para instalação;
- já estão instalados e estão aptos para atualização;

Como dispomos de informações atualizadas sobre nosso pacotes, podemos fazer atualizações regulares em nosso sistema operacional – esta prática é altamente recomendada e vai garantir a estabilidade e a confiabilidade do sistema instalado. Para fazer a atualização, utiliza-se o comando:

```
# apt-get upgrade[3]
```

Consultado e instalando com o apt-get

Para fazer instalações de pacotes com o apt-get, precisamos saber apenas o nome do pacote e a sintaxe da instalação. A sintaxe será sempre:

```
# apt-get install <nome do pacote>
```

Para o caso do programa joe testado anteriormente pelo método dpkg:

```
# apt-get install joe
```

Normalmente a maior dificuldade dos administradores de sistemas é saber qual o nome correto do pacte a ser instalado. Para resolver este problema, o sistema apt-get oferece um mecanismo de busca por palavra aproximada – poderíamos então procurar por programas editores de texto similares ao joe. A diretiva de consulta é o search

```
# apt-cache search  <palavra ou parte da palavra a ser pesquisada>
```

Em nosso exemplo

```
# apt-cache search joe
```

Removendo pacotes com apt-get

Invariavelmente, podemos precisar remover pacotes – o apt-get faz a remoção também.

```
# apt-get remove <nome do pacote>
```

- que remove apenas os binários – ou os executáveis.

3. ATENÇÃO: O procedimento de atualização é demorado porque muitos pacotes serão atualizados. Certifique-se de ter disponível uma conexão de banda larga.

ou ainda

```
# apt-get purge <nome do pacote>
```

- que remove os binários e os arquivos de configuração.

Aptitude

O aptitude também é um programa de gerenciamento de pacotes, serve como alternativa ao apt-get. Ambos funcionam de forma bastante semelhante mas o aptitude é mais eficiente ???

Para o aptitude, o conceito de repositório e de pacotes é o mesmo que para o apt-get ou dpkg.

Sintaxe de uso

- pesquisa:

```
# aptitude search <nome-do-pacote>
```

- instalação:

```
# aptitude install<nome-do-pacote>
```

- remoção:

```
# aptitude remove <nome-do-pacote>

# aptitude remove --purge <nome-do-pacote>
```

Modo GUI

O aptitude apresenta um formato de utilização diferenciado, que permite uma certa navegação na árvore de pacotes disponíveis/instalados/atualizáveis. Uma captura de tela do programa pode ser visto na figura D.1. Nesta instalação, temos 1295 pacotes instalados; 2834 não instalados; no caso não há pacotes para serem atualizados.

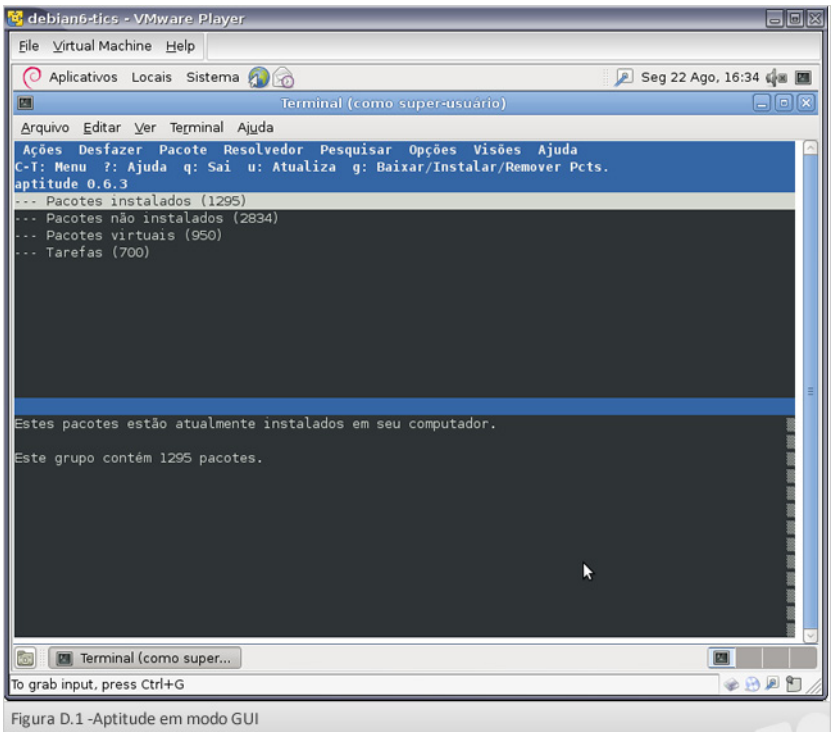

Figura D.1 -Aptitude em modo GUI
Fonte: do Autor

Gerenciadores em modo gráfico

Os gerenciadores de pacotes também podem ser utilizados em ambiente gráfico, o que torna o uso bastante intuitivo. O programa utilizado irá depender do ambiente gráfico que você estará utilizando. No caso do gnome, o gerenciador é o synaptic, que pode ser visto na figura D.2.

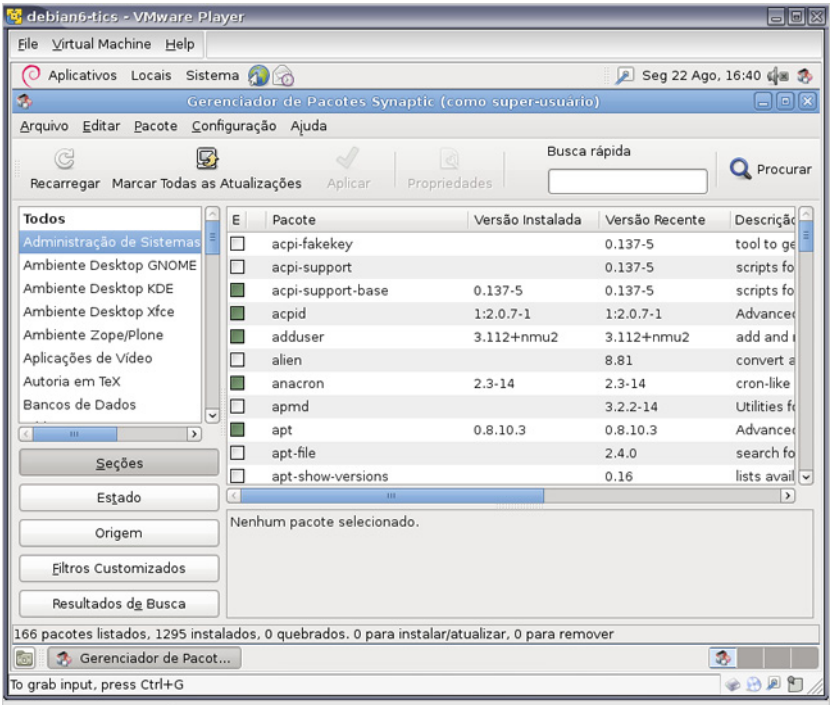

Figura D.2 - Gerenciador Synaptic
Fonte: do Autor

Neste caso, temos a facilidade do uso do mouse – que, via de regra, está presente somente em computadores Desktop ou Workstations.

Síntese

- Os sistemas operacionais Unix-Like possuem variados mecanismos de gerenciamento de pacotes, que buscam facilitar a atividade de administração do sistema;
- pacote é o nome usado para referencias os programas disponíveis para instalação;
    - um pacote contem todas os arquivos necessários ao funcionamento de determinado programa, como binários, arquivos de configuração, bibliotecas, páginas de manual e outros.
    - a distribuição Debian, na versão 6.0, oferece 29000 pacotes que são distribuídos em um formato de fácil utilização.
- São vários os mecanismos de gerenciamento de pacotes disponíveis na distribuição Debian:
    - dpkg
        - requer que o pacote esteja baixado no diretório corrente;
        - requer que as dependências estejam previamente resolvidas;
    - apt-get
        - requer que repositórios estejam previamente configurados ;
        - requer conexão com internet (para repositórios online) ;
        - instala os pacotes solicitados e resolve suas dependências automaticamente;
    - aptitude
        - o sucessor do apt-get;
        - é mais fácil de utilizar;
        - possui uma interface GUI;
    - gerenciadores gráficos
        - para computadores desktop, temos a possibilidade de uso de um gerenciador gráfico - por exemplo o synaptic para o ambiente gnome.

Atividades

1. Pesquisar, instalar, testar e remover os seguintes pacotes:

a) iptraf
b) iperf
c) joe
d) abiword
e) dia
f) vim-gnome
g) bluefish (baixar e instalar)

- Observação – o objetivo aqui é praticar o uso dos gerenciadores de pacotes, logo é recomendado exercitar com todos os gerenciadores estudados (apt-get, aptitude, synaptic e dpkg);

UNIDADE D

2. INTEROPERABILIDADE E ANTIVÍRUS

Introdução

Conforme verificado na unidade anterior, o objetivo fim de um sistema informatizado é a produção; é auxiliar no trabalho executado pelas pessoas; é viabilizar e agilizar processos, pesquisas, rotinas, etc. Em outras palavras, o computador precisa ser útil e esta utilidade está fortemente ligada ao quesito interoperabilidade de sistemas. Entende-se por interoperabilidade a funcionalidade que um sistema tem de trabalhar em conjunto com outros sistemas, buscando melhorar a produtividade e a eficiência do conjunto.

Nesta unidade trabalhamos com a instalação, a configuração e a utilização de programas específicos para prover um pouco mais de produtividade.

Interoperabilidade

Significa instalar, configurar e fazer uso de programas (pacotes) que permitam a utilização desta máquina por outras máquias ou pessoas – ou seja, que tornem a máquina produtiva de alguma forma. Isto normalmente está associado a configuração de servidores de rede – tipo intranet, por exemplo: instalar e configurar um servidor de arquivos local.

Existem uma infinidade de serviços que fazem/provêm interoperabilidade. Não pretendemos (e nem conseguiríamos) esgotar todos os serviços com seus detalhes em uma unidade (precisaríamos de muitas unidades para isto). Desta forma estudaremos a instalação, configuração e utilização de dois serviços fundamentais que são: servidor de arquivos com autenticação e servidor de páginas intranet.

Servidor de arquivos

Em uma rede local, é comum precisarmos de um servidor de arquivos, que faz o armazenamento/distribuição de arquivos de forma centralizada. Melhorando, desta forma a produtividade dos usuários; facilitando a mobilidade e até mesmo o gerenciamento de segurança como backup e antivírus. Na figura D.3 podemos visualizar um diagrama de LAN com um servidor de arquivos central.

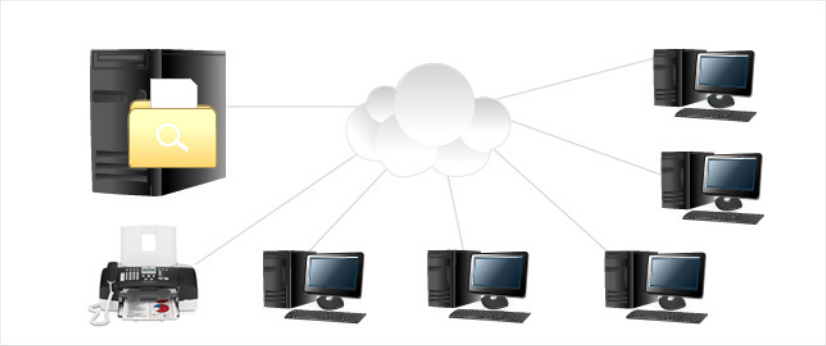

Figura D.3 - LAN com um servidor de arquivos central
Fonte: do Autor

Servidor SAMBA

SAMBA – acrônimo para SMB[1] , é um programa de rede que interliga sistemas GNU/Linux com computadores Windows[2]. O serviço conta com diversas funcionalidades, das quais destacamos: compartilhamento de pastas/arquivos e impressoras; autenticação de usuários e máquinas Windows em uma LAN. O serviço é instalado em um computador (servidor) GNU/Linux, que irá servir toda nossa rede interna.

Instalação:

A instalação é bastante simples, basta termos acesso a internet e os repositórios devidamente configurados – conforme visto na unidade anterior. O comando de instalação é:

```
# aptitude install samba
```

- Verifique que o aptitude irá instalar automaticamente todas as dependências do samba que são: samba-common e samba-common-bin.
- O instalador solicita que seja informado o nome do grupo de trabalho – neste caso, a configuração default poderá ser mantida porque o samba será configurado manualmente posteriormente.

Configuração:

A configuração do samba é feita no arquivo smb.conf que fica SEMPRE na pasta /etc/samba. O smb.conf é um arquivo texto e pode ser modificado com qualquer editor de texto simples (poderíamos, por exemplo, usar o JOE).

```
# gedit /etc/samba/smb.conf
```

Ao abrir o arquivo verificamos que ele não está vazio, mas tem diversas diretivas[3] de exemplo. Poderíamos simplesmente fazer algumas modificações neste arquivo fazer uso do sistema, mas nosso objetivo aqui é fazer uma configuração totalmente nova.

```
# Sample configuration file for the Samba suite for Debian GNU/Linux.

#

#
```

1. Protocolo utilizado na comunicação de rede entre sistemas operacionais Windows.
2. Windows é marca registrada da Microsoft Corporation.
3. Diretiva – nome dado a cada item de configuração dentro do smb.conf

```
# This is the main Samba configuration file. You should read the

# smb.conf(5) manual page in order to understand the options listed

# here. Samba has a huge number of configurable options most of which

# are not shown in this example
```

Configuração mínima:

```
# configuração mínima de teste
[globals]
        Netbios name = servidor
        Workgroup = intranet
        security = share
[publico]
        path = /var/samba/publico
        writable = yes
        comment = Pasta pública
        browseable = yes
        guest ok = yes
```

Como já registrado na unidade anterior, qualquer diretiva com o simbolo # no início é um comentário (ou explicação) a linha será desconsiderada pelo sistema. O arquivo de configuração mínimo tem duas sessões principais, que são:

[globals]

é a sessão que define as configurações globais de todo o sistema, ou seja, que servirão para todas as outras sessões. Diversas diretivas configuram a sessão globals, das quais destacamos as seguintes:

- Netbios Name ⟶ configura o nome do servidor na rede, ou seja, é o nome pelo qual nosso servidor será identificado por TODOS os clientes da rede em questão;
- Workgroup ⟶ configura o grupo de trabalho em que servidor e clientes devem estar. Aqui salientamos que todos devem estar no mesmo grupo de trabalho, caso contrário, os serviços de rede poderão não funcionar adequadamente.

Saiba mais:

Configuração da rede no Windows, disponível em:

<http://www.hardware.com.br/tutoriais/configurando-rede-Windows/>

[publico]

é a sessão que define um compartilhamento (que será identificado na rede como publico. Observa-se que este compartilhamento não possui nenhuma restrição de acesso.

- Path ⟶ esta diretiva indica qual diretório (pasta) do sistema está sendo compartilhada. Obviamente, este diretório precisa existir e respeitar as regras e permissões de acesso que serão determinadas pelo sistema. Neste caso, a pasta é publica, então as permissões serão RWX para todos os usuários.
- Writeable ⟶ esta diretiva indica que este compartilhamento permite escrita. Poderíamos ter compartilhamentos de distribuição de arquivos sem necessidade de escrita.
- Comment ⟶ indica um comentário para o compartilhamento;
- Browseable ⟶ indica que o compartilhamento é 'navegável', ou seja, seus arquivos podem ser vistos na rede. Novamente há compartilhamentos em que não queremos que os arquivos sejam vistos.
- Guest ok ⟶ indica que a pasta aceita visitantes, ou seja, usuários sem autenticação no sistema.

Testando

Para testar, é necessário ter pelo menos um cliente Windows configurado no mesmo grupo de trabalho do servidor (no caso intranet). Para fazer uso do compartilhamento, basta abrir o ícone "Meus Locais de Rede" o compartilhamento deverá estar disponível com na figura D.4. neste caso, bastará copiar arquivos e pastas para o compartilhamento selecionado. Vale salientar que o mesmo é publico e não oferece nenhuma proteção aos dados ali armazenados.

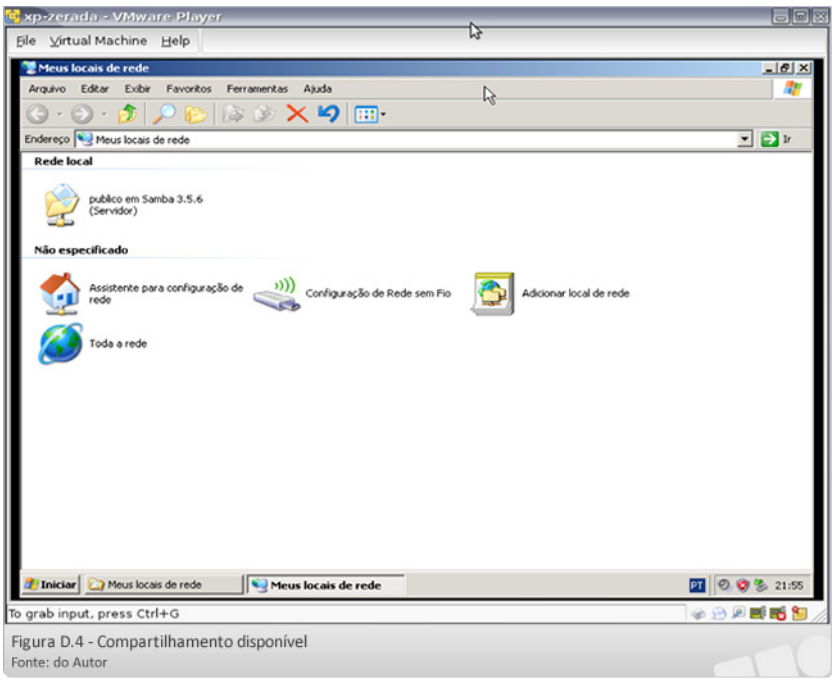

Figura D.4 - Compartilhamento disponível
Fonte: do Autor

Aprimorando os compartilhamentos

A configuração acima oferece uma funcionamento minimalista. Nas próximas configurações, vamos explorar alguns aspectos do servidor samba. Neste exemplo, precisaremos ter um cenário configurado, com usuários e grupos previamente definidos no sistema operacional. O diagrama de blocos abaixo representa este cenário.

Basicamente, temos precisaremos criar dois grupos, sendo um chamado chefia, que contém os usuários chefe1 e chefe2; e outro grupo chamado trabalho, que irá conter os usuários trab1, trab2 e trab3. Estes usuários podem (e é recomendado) ter seu acesso ao Linux bloqueado, ou seja, não precisarão logar no servidor – para bloquear o usuário utiliza-se o comando:

```
# passwd -l <usuario>
```

```
# configuração mínima de teste

[globals]
        Netbios name = servidor
        Workgroup = intranet
        security = user

[publico]
        path = /var/samba/publico
        writable = yes
        comment = Pasta pública
        browseable = yes
        guest ok = yes
```

```
[chefia]
        path = /var/samba/chefia
        comment = Pasta da chefia
        valid users = chefe1, chefe2
        writeable = yes
        browseable = yes

[trabalho]
        path = /var/samba/trabalho
        comment = Pasta dos funcionários
        valid users = +trabalho, chefe1, chefe2
        writeable = yes
        browseable = yes
```

Nesta configuração, temos 3 compartilhamentos e acesso restrito por usuários, descritos abaixo.

[globals]

- a diretiva de configuração global para todo o sistema;
- security user habilita a restrição de acesso por usuário;

[publico]

- compartilhamento público a todos os usuários não tem nenhuma mudança em relação ao público da configuração minima.

[chefia]

- compartilhamento da pasta /var/samba/chefia
- para simplificar o funcionamento, utilizamos compartilhamento total na pasta, ou seja a+rwx para chefia.
- valid users = chefe1, chefe2
- permite que somente os usuários chefe1 e chefe2 tenham acesso ao compartilhamento;
- não há a diretiva guest ok = yes

[trabalho]

- compartilhamento da pasta /var/samba/trabalho
- para simplificar o funcionamento, utilizamos compartilhamento total na pasta, ou seja a+rwx para trabalho.
- Valid users = +trab, chefe1, chefe2
- permite acesso de todos os usuários do grupo trab (+trab) e dos usuários chefe1 e chefe2.
- não há a diretiva guest ok = yes

Habilitando os usuários no samba

Para que os usuários tenham acesso aos compartilhamentos, é necessário incluir os mesmos no samba. Isto é feito com o comando: smbpasswd -a <usuário>; o comando irá solicitar uma nova senha para o usuário – esta será a senha pela qual o usuário irá se autenticar no compartilhamento desejado. Para excluir um usuário *#passwd -x <usuário>*

```
# smbpasswd -a chefe1

# smbpasswd -a trab2
```

Testando

O procedimento de teste será o mesmo feito na etapa anterior. Só que neste momento, ao acessar o compartilhamento, será necessário informar a senha de autenticação; o samba por sua vez irá dar as permissões conforme definido na configuração do sistema. Para que o Windows “esqueça” a senha, é necessário fazer *logoff*.

Antivírus

O uso de antivírus não é obrigatório em sistemas GNU/Linux, mas como este sistema que estamos configurando é um servidor de arquivos que atenderá toda uma rede interna o uso da ferramenta passa a ser fortemente recomendado. Outra situação em que o antivírus é fortemente recomendado é no uso de servidores de e-mail (que não é o nosso foco).

Hoje existem diversas soluções para antivírus em servidores SAMBA com Linux, das quais sugerimos o Clamav.

Clamav

Para instalar o antivírus:

```
# aptitude install clamav clamav-daemon clamav-freshclam
```

Atualizando

Após a instalação, é necessário atualizar o banco de dados da ferramenta:

```
# freshclam
```

Procurando por arquivos infectados

Feito a instalação, basta rodar o antivírus na pasta do servidor de arquivos.

```
# clamscan -r  /var/samba
```

Automatizando

Fazendo uso do crontab, podemos automatizar a execução da atualização e do escaneamento. No exemplo abaixo está sendo agendado uma atualização (freshclam) para as 4h todos os dias do mês, todos os meses e todos os dias da semana e também um escaneamento para as 4h30min todos os dias do mês, todos os meses e todos os dias da semana.

```
# crontab -e

0 4 * * * freshclam

30 4 * * * clamscan -r  /var/samba
```

Saiba mais:

Agendador de tarefas CRON, disponível em:

<http://www.guiafoca.org/cgs/guia/inic_interm/ch-manut.html#s-manut-cron>

Servidor de páginas intranet

O servidor de páginas intranet é um serviço de hospedagem WEB, dentro de um ambiente LAN. Este serviço que viabiliza a interoperabilidade, oferecendo diversos recursos WEB baseados, por exemplo, em PHP e banco de dados.

Servidor APACHE

O programa utilizado para este tipo de serviço é o servidor Apache que é um dos software livres mais antigos e conhecidos, já que representa , além é claro, de ser um dos mais robustos e funcionais. Neste material, estudaremos apenas algumas funcionalidades essenciais para disponibilização de uma intranet. Vale salientar que não será abordado a programação e apenas o serviço de hospedagem, integrando com o PHP e um banco de dados como o MysSql.

Instalação do servidor:

O pacote responsável pelo serviço é o apache2; ao fazer a instalação, diversos dependências serão automaticamente resolvidas. O comando de instalação é:

```
# aptitude install apache2
```

Testando

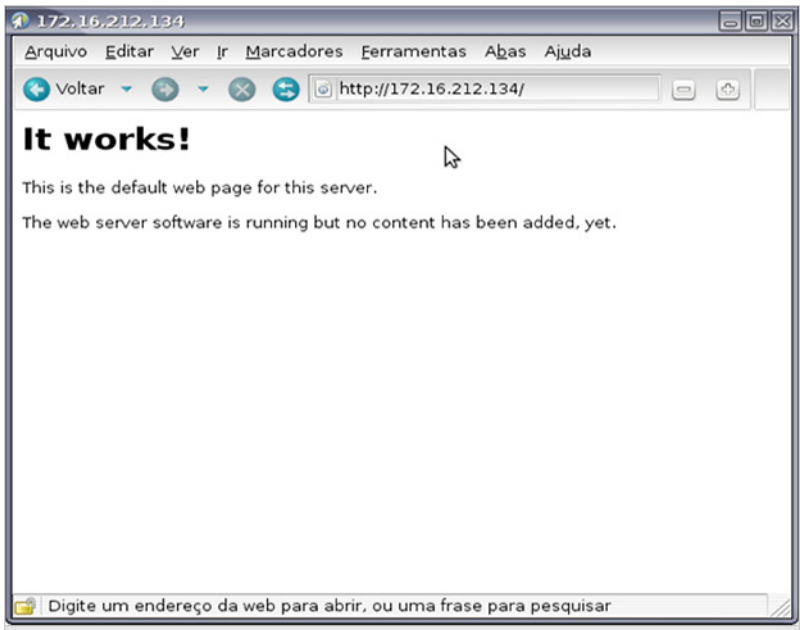

Figura D.5 - It works do apache
Fonte: do Autor

Por *default*, o apache traz um site de testes, que responde no endereço IP do servidor. Para testar basta digitar o endereço IP no browerde sua preferência (desde que esteja instalado em seu computador). A resposta sera "It Works", conforme pode ser verificado na figura D.5.

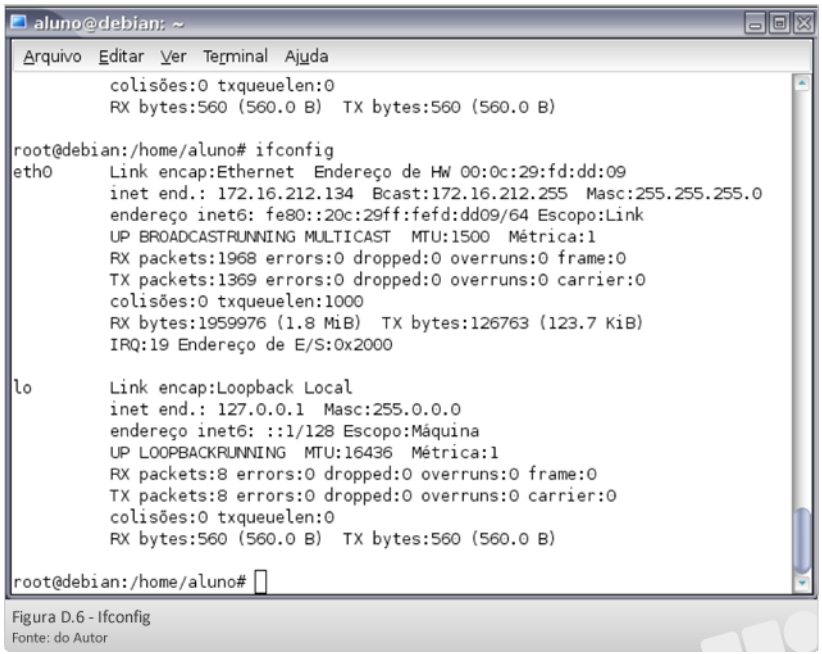

Figura D.6 - Ifconfig
Fonte: do Autor

Todos os próximos testes serão feitos da mesma forma. Vale salientar que o endereço IP será particular de seu computador. O comando de verificação do endereço IP é ifconfig, conforme figura D.6, na captura abaixo, podemos ver a interface eth0 com inet end. 172.16.212.134.

Estrutura de configuração

O apache tem uma estrutura particular de funcionamento, sendo dividido em área de configuração, onde são armazenados todos os arquivos de configuração e área de conteúdo, onde são armazenados os conteúdos dos sites.

Especificamente no caso do apache, apenas o usuário root tem permissão de modificação em ambas as áreas.

Criando um novo site chamado intranet

```
# nano /etc/apache2/sites-available/intranet
```

```
<Virtualhost 172.26.212.134:80>
    ServerName intranet
    Documentroot /var/www/intranet
</Virtualhost>
```

Na configuração apresentada, um site simples está sendo configurado. Todas as configurações são feitas entre as diretivas <VirtualHost> e </VirtualHost> que delimitam a configuração do site.

```
<Virtualhost 172.26.212.134:80>
    ServerName intranet
    Documentroot /var/www/intranet
</Virtualhost>
```

- Cada site, criado é um VirtualHost;
- DocumentRoot /var/www/intranet ⟶ esta diretiva informa o local onde está o CONTEÚDO do site configurado;
- ServerName intranet ⟶ informa o nome do site que está sendo configurado

Ativando o site criado

O site precisa ser alimentado com um conteúdo. No caso, iremos simplesmente criar um arquivo index. html com o conteúdo abaixo descrito, que apenas mostra a menagem “Intranet me construção”. Isto pode ser feito com qualquer editor de textos, com por exemplo o JOE

```
<html>
<body>
<h1> Intranet em construção </h1>
</body>
</html>
```

O site criado não entra em atividade imediatamente, o administrador do sistema precisa declarar explicitamente que um determinado site entrará (ou sairá) de atividade. Isto é feito com o par de comandos a2ensite, para ativar e a2dissite para desativar. No nosso caso, iremos ativar o site da seguinte forma:

```
# a2ensiste intranet
```

Caso alguma modificação seja feita no arquivo de configuração do site, então será necessário desabilitar e reabilitar o site. O comando para desabilitar é: #a2dissite intranet

Após a ativação o apache precisará ser recarregado, isto é feito com o comando:

```
# /etc/init.d/apache2 reload
```

A partir deste momento a página já estará funcionado mesmo que com um conteúdo experimental, que poderá ser visualizado pelo browser. Figura D.7 mostra a captura de tela da página web de nossa Intranet.

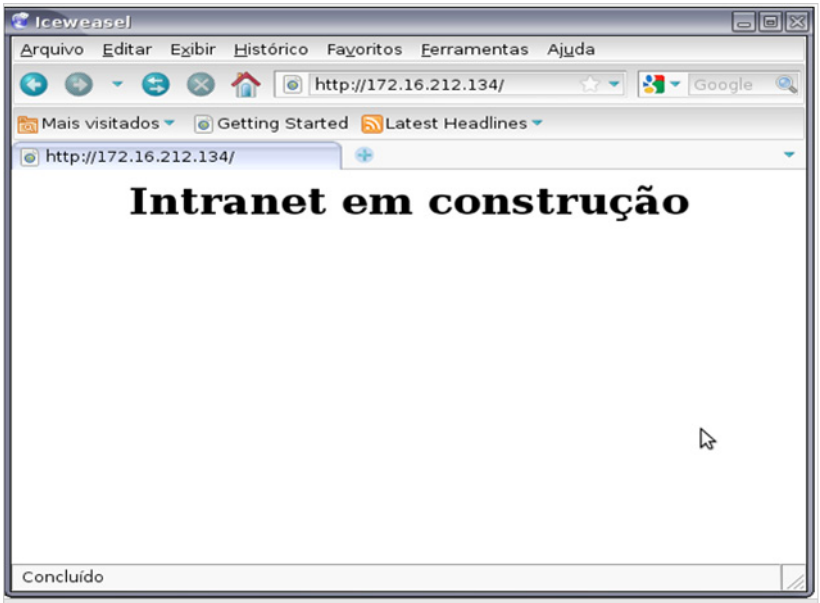

Figura D.7 - Teste site intranet
Fonte: do Autor

Aprimorando

O apache oferece uma gama de opções que podem ser utilizadas aprimorando o site da intranet – suporte a PHP e banco de dados por exemplo, suporte a HTTPS ou autenticação.

Ativando o suporte a PHP e Mysql

```
# aptitude install libapache2-mod-php mysql-server php-mysql
```

Na instalação acima todos os pacotes estão sendo informados em uma única linha (ou comando), mas poderiam ser informados separadamente.

Para testar, criamos um teste.php com a função phpinfo(); apenas para obter informações do servidor. O resultado pode ser verificado na figura D.8, onde está sendo mostrado o suporte mysql habilitado.

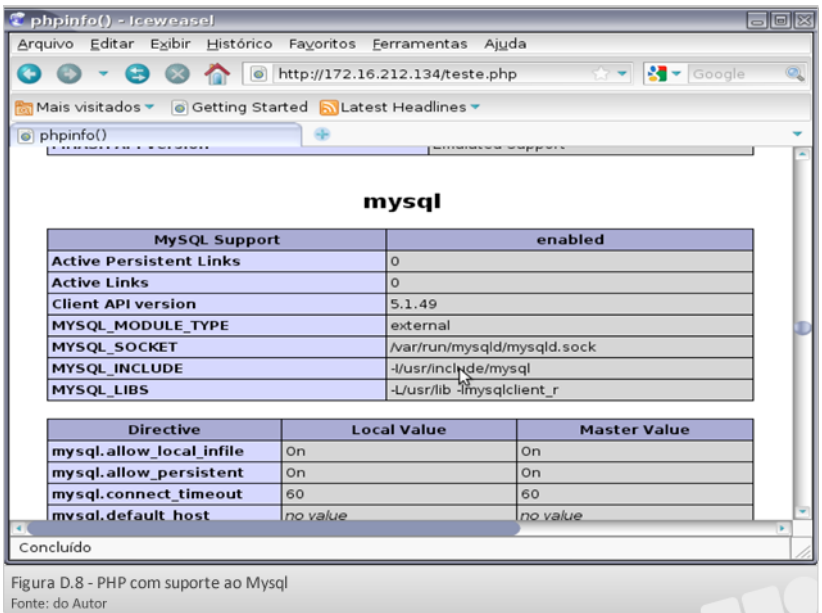

Figura D.8 - PHP com suporte ao Mysql
Fonte: do Autor

Ativando o SSL

SSL é uma camada de proteção que é adicionada ao HTTP, formando o HTTPS, largamente utilizado por bancos e sistema de e-comerce. Para ativarmos o SSL, de forma bastante simples, faremos uso de um certificado auto assinado, gerado conforme o comando abaixo:

```
# openssl req $@ -new -x509 -out /etc/apache2/certificado.pem -nodes -keyout /
etc/apache2/certificado.pem
```

Este comando pede diversas informações demográficas, para identificação do certificado.

Saiba mais:

Entendendo o mercado de certificados SSL, disponível em:

<http://www.hardware.com.br/dicas/mercado-ssl.html/>

```
<Virtualhost 172.16.212.134:80>
    ServerName intranet.lan
    Documentroot /var/www/intranet
</Virtualhost>

<Virtualhost 172.16.212.134:443>
    DocumentRoot /var/www/intranet

    SSLEngine ON
    SSLCertificateFile /etc/apache2/certificado.pem
</Virtualhost>
```

- A configuração:
  - a configuração da parte HTTP não sofre modificação;
  - incluímos um novo virtualhost, que abre a porta 443;
  - informamos novamente a pasta de conteúdo do site – desta vez da parte que queremos proteger por ssl;
  - liga-se o 'motor' de ssl em SSLEngine ON;
  - informa-se a localização e o certificado que foi criado na etapa do certificado auto assinado.

Como o site sofreu modificações, é preciso reativá-lo e reiniciar o apache;

```
# a2dissite intranet

# a2ensiste intranet

# /etc/init.d apache2 reload
```

Testando

Para testar, abrimos um brower e apontamos para o endereço de nosso site: HTTP://172.16.212.134 – no primeiro acesso, por motivos de segurança, o browser irá avisar que o certificado não é confiável (lembre-se que nosso certificado é do tipo auto assinado). Forçamos a aceitação de uma exceção e prosseguimos no site.

Cada browser tem sua forma de aceitar exceções; no iceweasel precisamos abrir a a opção "entendendo os riscos" e clicar no botão "Adicionar Exceção" e após confirmar a exceção de segurança, conforme pode ser visto na Figura D.9.

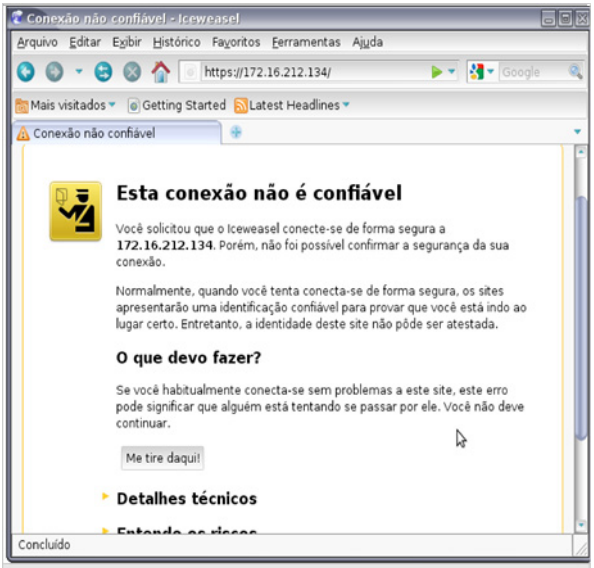

Figura D.9 - Conexão não confiável
Fonte: do Autor

A figura D.10 mostra o site seguro sendo acessado. Sabemos que está sob HTTPS porque o browser mostra um pequeno cadeado na barra de status, além de informar isto ao lado do endereço do site na barra de endereços.

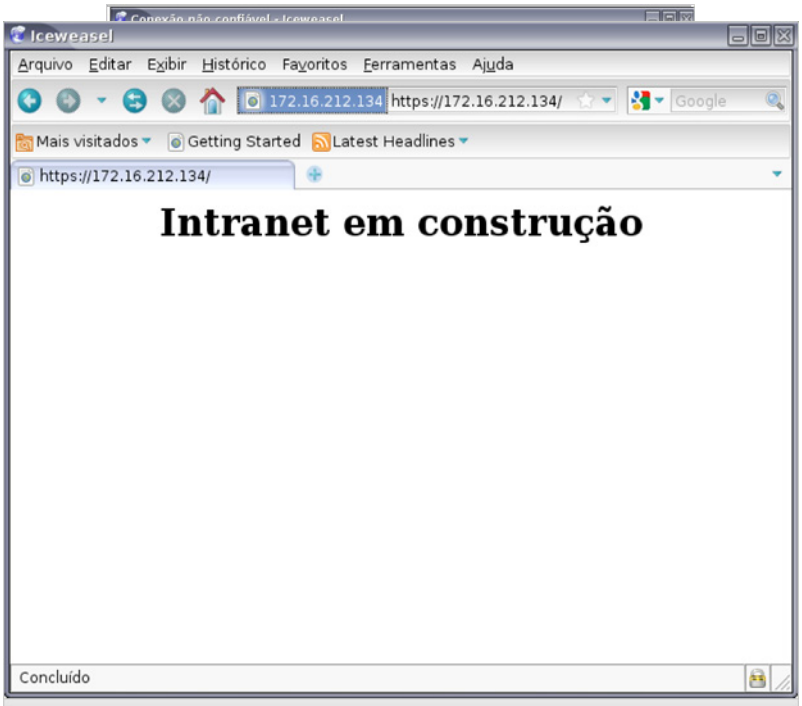

Figura D.10 - Site seguro
Fonte: do Autor

Aplicações

Neste material, mostramos duas ferramentas que viabilizam a interoperabilidade, sendo que ambas oferecem opções de funcionamento/configuração bem mais avançadas. No entanto, cada configuração irá depender diretamente da aplicação, do ambiente e da equipe envolvida com o servidor. Além disso, no caso do Apache + PHP + Mysql, é necessário que um site (portal ou aplicação) tenha sido desenvolvido para "ser hospedado" no nosso servidor. A funcionalidade estará no site desenvolvido e não no programa servidor.

Síntese

- existem diversas ferramentas que viabilizam a interoperabilidade entre sistemas operacionais;
- o principal objetivo da interoperabilidade é melhorar a produtividade de um ambiente de trabalho;
- SAMBA oferece um serviço de servidor de arquivos em uma rede de cliente Windows:
  - Instalar o samba com aptitude install samba
- APACHE oferece um serviço de servidor HTTP e HTTPS

Atividades

1. Criar um compartilhamento SAMBA para o usuário aluno

2. Criar uma área do site protegida por usuário/senha, utilizando um dos métodos de autenticação do apache.

Multiprogramação

Unidade E
Sistemas Operacionais

UNIDADE E

MULTIPROGRAMAÇÃO

Introdução

Qualquer computador pessoal trabalha com diversos programas em execução "simultânea", ou seja, independentemente de sistema operacional que usamos, podemos abrir um editor de texto, ouvir música, ler e-mails, jogar e outras coisas. Temos a nítida sensação que o computador está executando tudo ao mesmo tempo.

Esta capacidade multitarefa do computador é o objeto de estudo desta unidade. Vamos estudar o que é um processo, seu ciclo de vida e o tratamento dado pelo computador (hardware + sistema operacional) a este.

Monoprogramação vs Multiprogramação

Sistemas monoprogramado

Também são conhecidos como monotarefa, são os sistemas capazes de executar apenas um processo por vez, ou seja, nunca poderemos ouvir música enquanto digitamos um texto; basicamente, teremos que fazer uma atividade por vez. Isto é uma característica que certamente não desejaríamos em nossos computadores. Os sistemas monoprogramados são encontrados nos primórdios da história dos sistemas operacionais e o exemplo clássico é o DOS (PC-DOS de 1981).

Na figura E.1 verificamos a memória ocupada por apenas um programa. Verifica-se que o a memória é desperdiçada porque não é toda utilizada.

Figura E.1 - Memória monoprogramada
Fonte: do Autor

Sistemas multiprogramado

Também são conhecidos como multitarefa, são os sistemas capazes de suportar diversos processos em execução (ouvir música enquanto digitamos um texto). Sistemas são caracterizados por:

timesharing

Um sistema multiprogramado "divide" o tempo da CPU entre os diversos processos que estão sendo executados;

pesudoparalelismo

Esta característica de "divisão de tempo" causa no usuário uma sensação de que tudo está acontecendo em paralelo (simultaneamente), ou seja, temos a nítida impressão que o processador está executando ao mesmo tempo os processos do editor de texto e do player, enquanto que na verdade está havendo uma divisão do seu tempo – daí o termo falso paralelo. Cada um dos processos ganha uma fatia de tempo do processador.

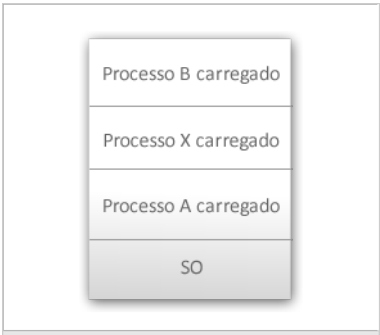

Figura E.2 - Memória multiprogramada
Fonte: do Autor

Na figura E.2, verificamos que a memória fica dividia entre vários processos. Observa-se um melhor aproveitamento do espaço de memória, mas há um aumento na complexidade do gerenciamento da memória.

O Windows NT é o primeiro sistema operacional da Microsoft com capacidade de multiprogramação – as versões Windows 95 e Windows 98 fazem uma emulação de multiprogramação.

Cabe ao sistema regular esta divisão de tempo e nos capítulos subsequentes vamos verificar como é possível (ou como é feita) esta multiprogramação.

Processo vs Programa

Antes de mais nada é preciso compreender o significado de processo sua relação com o programa. Sempre que criamos algum programa, escrevemos seu código em um editor de programas (também conhecido como IDE), este código fonte passa então por um processo de compilação que gera um arquivo executável.

Este executável é um PROGRAMA, ou seja, um arquivo com um conjunto de instruções passíveis de execução pelo meu sistema informatizado[1]. Este arquivo fica armazenado no disco rígido pronto para ser executado – aguardando a ordem do usuário para entrar em execução.

Um PROGRAMA é um arquivo executável, estático, passivo, contem um conjunto de instruções (em código de máquina) pronto para ser executado pelo sistema.

1. Neste contexto, sistema representa o par Hardware + Sistema Operacional.

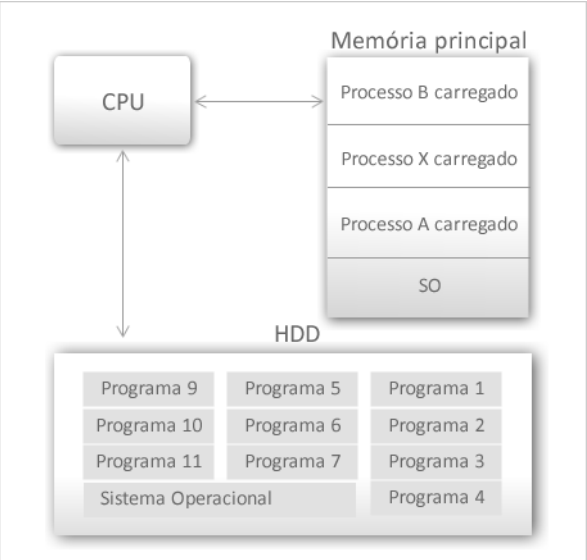

Figura E.3 - Processo vs programa
Fonte: do Autor

Ao entrar em execução, o programa é carregado para a memória e neste momento passa a ser considerado um processo. Enquanto o processo existir (estiver carregado na memória) irá passar por diversos estados e transformações, sendo em algum momento encerrado (deixa de existir). O processo normalmente é composto por uma cópia do código de máquina, chamado área de programa; e uma parte para variáveis, chamado área de dados.

Um PROCESSO é um programa em execução, caracterizado por:

- Ser dinâmico e ativo;
- Poder criar outros processos;
- Poder executar chamadas de sistema.
- É criado e é encerrado ou morto;

Ciclo de vida do processo

Criação

Todo processo precisa ser criado; a criação é o momento em que o sistema operacional "carrega" ou instancia o programa na memória principal. A partir deste momento o processo irá trabalhar por algum tempo e irá encerrar. Os processos são criados em diversas situações, por exemplo:

- ao ligar o computador – neste momento o sistema operacional cria uma grande quantidade de processos (do próprio sistema operacional) e de programas de usuários como por exemplo a shell ou o ambiente gráfico, ou ainda alguma backdoor que esteja infectando o sistema operacional.
- Quando o usuário solicita – momento em que o usuário abre um programa qualquer. Isto pode criar um ou mais processos na memória;
- Quando um processo cria outro processo – processos podem criar outros processos, situação muito comum para "delegar" tarefas.

Encerramento

Todo processo precisa ser encerrado em algum momento; isto pode acontecer naturalmente, quando o processo é encerrado por si próprio (fechamento do programa) ou de forma forçada, quando o sistema operacional (ou o administrador do sistema) matar algum processo. Os processos são encerrados em diversas situações, por exemplo:

- Encerramento do programa – usuário parou de utilizar o editor de textos então encerra o programa;
- Processo foi morto – o sistema operacional pode estar precisando de memória e para isto faz uma limpeza, matando os processos fora de uso; ou o administrador do sistema entendeu que precisava matar algum processo;
- Desligamento/reinicialização do computador - neste momento todos os processos devem ser encerrados; se algum processo não quiser se encerrar naturalmente, o sistema operacional "faz o serviço".

Relacionamento inter processos

Os processos em execução podem ou não ter algum tipo de relacionamento entre si – isto varia conforme a origem (fabricante) do sistema operacional. Sistemas operacionais Unix-Like[2] utilizam o conceito de grupo de processos onde há uma hierarquia e passagem de herança explicita entre processos, ou seja, um processo (pai) irá criar outro (filho) e assim sucessivamente.

Todo processo filho herda algumas características de seu pais (permissões de acesso por exemplo). Desta forma, temos uma estrutura em árvore, conforme figura E.4.

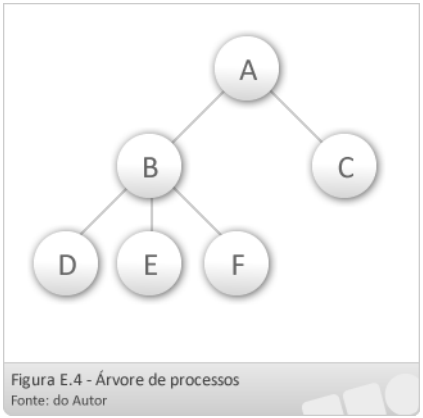

Figura E.4 - Árvore de processos
Fonte: do Autor

- Os o processos B e C são filhos de A;
- os processos D, E e F são filhos de B;
- C não tem filhos;
- Se processo B encerrar, processos D, E e F também serão encerrados;
- Se processo A encerrar, todos os outros processos serão encerrados também;
- As permissões de A são herdadas por seus filhos.

Em nossa máquina virtual instalada, podemos ver a árvore de processos usando o programa pstree[3] , que deve ser utilizado em conjunto como less para facilitar a visualização (para sair da visualização, usar q). Como visto na figura E.5, o resultado do comando será uma árvore, mostrando os processos e seus filhos, é claro na forma de uma árvore.

```
$ pstree | less.
```

2. Sistemas operacionais originários ou inspirados do Unix.
3. Este programa é fornecido pelo pacote psmisc.

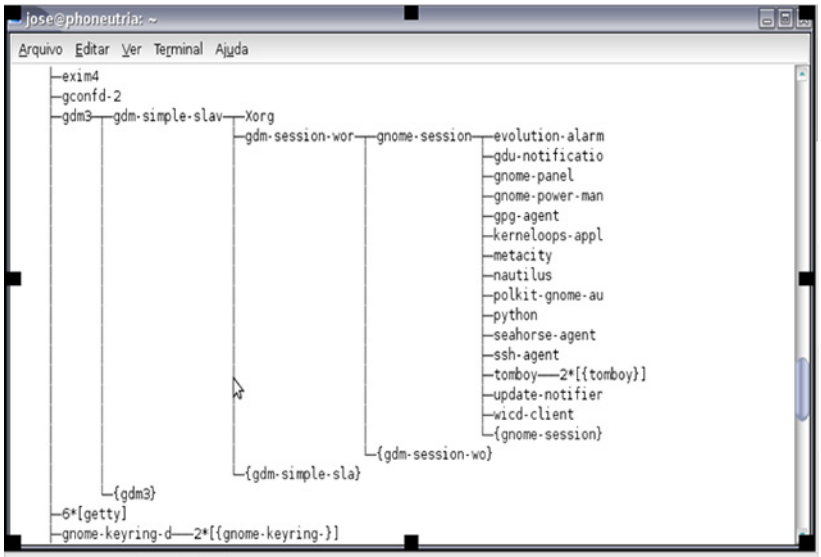

Figura E.5 - Trecho resultado do pstree
Fonte: do Autor

Estados do processo

De forma geral, durante sua existência, o processo podem assumir 3 estados básicos que são:

Em execução:

Neste estado o processo está no processador, sendo processado. Cada processador trabalha com apenas um processo por vez[4] . O tempo que o processo ocupa o processador irá depender do tipo do escalonador do sistema operacional – visto na próxima unidade.

Pronto:

Neste estado, o processo está pronto, aguardando sua vez de ser executado. Há uma fila de espera que está sujeita a regras específicas, também do escalonador, estudadas na próxima unidade. Um sistema operacional pode inclusive ter mais que uma fila de prontos na espera pelo processador.

Bloqueado:

Este é um estado em que o processo fez uma requisição de I/O, através de uma chamada de sistema, e está aguardando o resultado (a resposta) deste. Procedimentos de I/O são muito lentos quando comparados com a velocidade do processamento puro; então o processo pode entrar em um estado especial enquanto aguarda a resposta do dispositivo de I/O. Neste estado, não deve haver consumo de processador, "sobrando recurso" para outros processos.

Transição entre os estados do processo

Durante a existência dos processos, eles "transitam" entre os estados Pronto, Execução e Bloqueado. As transições possíveis são (acompanhar pela figura E.6) :

- Pronto para execução: quando um processo da fila de prontos é destacado e "ganha" o processador.
- Execução para pronto: quando o processo que está executando perde o processador e volta para a fila de prontos. Esta "perda" se dá por motivos gerenciados pelo escalonador.

4. CPU's multicore terão um processo por núcleo.

- Execução para bloqueado: quando o processo que está executando faz uma solicitação de I/O, perde o processador (compulsoriamente) e vai para a lista de processos aguardando resultado de sua solicitação;
- Bloqueado para pronto: quando a solicitação de I/O finaliza (a resposta do dispositivo é entregue ao processo) o processo volta para a fila de prontos. Neste retorno o processo se submete as regras impostas pelo gerenciador desta fila.

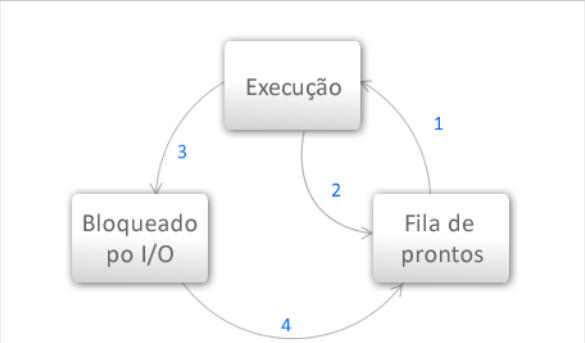

Figura E.6 - Estados e transições dos processos
Fonte: do Autor

Modos do processador

O processador tem dois modos de operação, que são:

Modo Usuário

Também chamado de modo protegido. É o modo em que o processador está habilitado exclusivamente na execução de processos de usuários, ou seja, os programas dos usuários como editor de textos, compilador, browser e todos os outros. Tudo que for executado neste modo é protegido contra acesso de outro processo e impedido de acessar áreas protegidas de outros processos e do próprio sistema operacional. Diversas restrições são implementadas para proteger o conjunto. Basicamente operações que não interessam ao usuário.

Modo Supervisor

No modo supervisor, o processador está atendendo interrupções de hardware. Neste modo não há restrições impostas e o processador executa as funções protegidas. As operações que podem ser exemplo neste caso são:

- reset de hardware;
- formatação de disco;
- execução de daemons de rede;

Interrupções

Interrupções, também conhecidas IRQ (*Interrupt Request*) por são sinais de hardware enviados ao processador por algum dispositivo, chamando a atenção deste para alguma ação específica. Sempre que algum device precisar de algo do processador irá disparar um pedido de interrupção.

A interrupção lembra uma chamada de telefone em uma residência, onde uma pessoa está executando uma tarefa comum, lendo um livro por exemplo. Neste momento o telefone toca, gerando uma interrupção, neste caso a pessoa terá de parar sua atividade e atender o telefone, ou seja, tratar a requisição de interrupção.

Exemplificando em um computador real, se uma placa de rede recebe algum pacote na sua entrada, este fluxo de dados que entra, precisará de algum tratamento então a placa de rede gera um pedido de

interrupção ao processador. Quando o processador atender a requisição, irá dar o tratamento necessário aos dados.

A transição entre os estados dos processos, PRONTO ↔ EXECUÇÃO, EXECUÇÃO ↔ BLOQUEADO e EXECUÇÃO ↔ PRONTO, é controlada por um dispositivo chamado RTC (*Real Time Clock*); um relógio que gera um sinal de interrupção em intervalos regulares. Ou seja, o processador sabe que é hora de trocar de processo em execução quando o RTC avisa.

Threads

Threads, também conhecidos como processos leves, porque dividem o fluxo de um processo em vários fluxos menores (e mais rápidos em termos de processamento). O processo transformado em threads é chamado multi-thread. Atualmente, a utilização de programação baseada em threads para processos 'pesados' é considerado uma boa prática.

A figura E.7 mostra uma comparação de um processo em relação a threads.

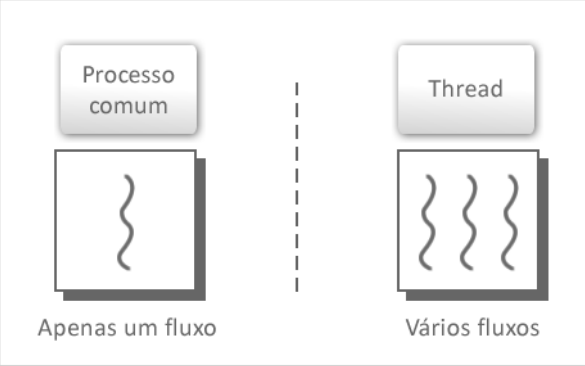

Figura E.7 - Processo vs thread
Fonte: do Autor

Criando processos

Uma programação bem interessante é a criação de processos, que pode ser facilmente desenvolvida na linguagem C em sistemas unix-like.

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>


int main(int argc,char *argv[]){

 pid_t fork1;

 int pid_filho,pid_pai,status,i;


 printf("PAI: Meu pid é: %d não tenho filhos - execute $ps -ef  para
ver meu PID\n",(int)getpid());           printf("PAI: - tecle ENTER para
continuar\n");
```

```
      getchar();


      fork1 = fork();
      if(fork1 == -1){
                  perror("Falha na criação do filho");
                  _exit(-1);}
      else if(fork1 == 0){       //area do filho
                  pid_filho = getpid();
                  pid_pai = getppid();
                  printf("FILHO: Eu sou o filho e meu pid é: %d \n",pid_filho);
                  printf("FILHO: Meu processo pai é: %d \n",pid_pai);
                  printf("FILHO: Minha tarefa é repetir bom dia 5 vezes\n");
                  for(i=1;i<=5;i++){
                        sleep(1);
                        printf("FILHO: Bom dia nº %d!\n",i);}
                  printf("FILHO: encerrei minha atividade - tecle ENTER para
    continuar\n");
                  getchar();
                  _exit(0);}
       else { //area do pai
                  wait(&status);
                  printf("PAI: Meu pid é: %d \n",(int)getpid());
                  printf("PAI: Criei um filho com PID: %d \n", (int)pid_filho);
                  printf("PAI: Minha tarefa dizer boa noite 2 vezes\n");
                  for(i=1;i<=2;i++){
                        sleep(1);
                        printf("PAI: Boa noite nº %d!\n",i);}
                  printf("PAI: Finalizei - tecle ENTER para continuar\n");
                  getchar();
                  _exit(0);}
    }
```

No código acima, o processo PAI cria um processo FILHO que executa uma ação qualquer e retorna para o PAI. Neste programa podemos observar a técnica de programação envolvida na criação de processos.

Uma vez em execução, podemos acompanhar a criação e o encerramento dos processos pelo comando pstree, visto anteriormente.

Síntese

- Processos são programas em execução na memória principal, enquanto programas são conjunto de instruções prontas para serem executadas e armazenados em memórias secundárias;
- Os sistemas atuais trabalham com o conceito de multiprogramação, em que temos diversos processos carregados na memória e que são executados “como que em paralelo” pelo conceito de pseudo-paralelismo.
- Sistemas mais antigos trabalham com o conceito de monoprograma, em que o sistema operacional carrega apenas um processo na memória;
- Interrupções são um mecanismo de sinalização por hardware enviadas por dispositivos de hardware, usadas normalmente para solicitar alguma ação do processador.
- Threads são um tipo de processo que tem vários fluxos de dados e que tem um processamento mais leve se comparado com um processo convencional.
- Os principais comandos de manipulação de processos em linux são:
    - $ ps ⟶ para ver a lista de processos em execução;
    - $ pstree ⟶ para ver a lista de processos em forma de árvore;
    - funções fork/join são funções em C para criação de processos;

Atividades

1. Verificar a árvore de processos em um sistema Linux;

2. Responder o questionário:

    a) Explique, com suas palavras a diferença entre processos e programa;
    b) Diferencie sistemas monoprogramado de multiprogramado.
    c) Qual o significado de pseudoparalelismo
    d) Qual o objetivo da interrupção?
    e) Detalhe os métodos de operação do processador;

F

Gerenciamento do processador

Unidade F
Sistemas Operacionais

UNIDADE F

GERENCIAMENTO DO PROCESSADOR

Introdução

Como visto na unidade anterior, o processador trabalha com diversos processos, que estão carregados na memória e que disputam entre si pelo processador. Cabe ao sistema operacional atender a todos estes processos de forma eficiente e conveniente. Nesta unidade vamos estudar como é feito o gerenciamento do processador e como é possível acontecer a multiprogramação.

Bloco descritor de processo

Quando criamos um programa qualquer, precisamos ter diversas variáveis que serão utilizadas durante a execução do programa e que terão alguma função de controle, mensagem, acumulador ou o que for necessário. Por exemplo, no código em C abaixo, temos uma variável x, do tipo inteiro, que é usada para controlar um loop.

```
#include<stdio.h>

main(){


int x;

for(x=0;x<10;x++){

  printf(" %d ",x);

  }

}
```

O sistema operacional é um programa como outro qualquer, que executa um conjunto de tarefas específico; foi escrito em alguma linguagem de programação; foi compilado e é executado; e como qualquer outro programa, possui variáveis e estruturas de controle internas. Uma destas estruturas de controle é o descritor de processos – que é uma estrutura de dados (*struct*) para armazenar dados dos processos enquanto o SO estiver executando.

Os dados do processo que o descritor de processo armazena, variam conforme o sistema operacional – a grosso modo poderíamos citar:

- lista de arquivos abertos pelo processo;
- permissões do processo;
- prioridade do processo;
- hora de criação do processo;
- tempo de CPU já gasto pelo processo;
- informações sobre o estado de execução do processo (contexto de execução).

No trecho de código a seguir, podemos verificar a criação de uma estrutura de controle de processos que poderia ser utilizada em um sistema operacional fictício. No caso, podemos ver a criação de uma struct chamada desc_proc, que possui diversos campos, cada um com uma função específica no controle dos processos.

Dentro da função main temos a inicialização de uma tabela de descritores – identificada como tab_desc, que definirá a quantidade máxima de processos que o sistema irá manipular simultaneamente. Também são inicializados as "filas" de descritores livres – identificada por desc_livre e fila de espera da CPU – identificada por espera_cpu. A lista de processos criados vais sendo armazenada na variável tab_desc, que vai "consumindo" espaços de desc_livre.

```
struc desc_proc{
    char estado_atual;
    int prioridade;
    unsigned int inicio_memoria;
    unsigned int tamanho_memoria;
    struct arquivos_abertos[20];

    unsigned int tempo_cpu;
    unsigned int proc_pc;
    unsigned int proc_sp;
    unsigned int proc_acc;
    unsigned int proc_rx
    struct desc_proc *proximo;
};

int main(int argc, char *argv){
    struct desc_proc tab_desc[MAX_DESC_PROC];
    struct desc_proc *desc_livre;
    struct desc_proc *espera_cpu;

  for(i=0; i<MAX_DESC_PROC-1; ++i){
                tab_desc[i].proximo = &tab_desc[i+1];}

    tab_desc[i].proximo = NULL;
    desc_livre = &tab_desc[0];
    exit(0);
}
```

Esta estrutura de dados tem como função gerenciar e controlar toda a existência de um processo criado dentro de um determinado sistema operacional. Em outras palavras, é a variável que "guarda" todo contexto do processo enquanto este existir. A estrutura é muito importante porque é através desta que o sistema operacional pode ter o controle sobre a existência dos processos e principalmente viabilizar a multiprogramação.

Como a multiprogramação acontece?

Na unidade anterior verificamos que o sistema operacional trabalha com um mecanismo de multiprogramação, atendendo diversos processos "quase que simultaneamente" dando para o usuário a sensação que tudo está acontecendo em paralelo – isto é denominado pseudoparalelismo. Na figura F.1 vemos um diagrama (mesmo mostrado na unidade anterior) que apresenta os estados e as transições entre estes estados. Neste ponto podemos nos perguntar como o sistema operacional consegue "fazer" a multiprogramação.

Contexto de execução

Um processador possui diversos registradores internos que são utilizados no processamento; estes registradores são contadores, acumuladores, registradores de deslocamento e outros mais. Quando um processo está sendo processado, está fazendo uso destes registradores que vão sofrendo modificações "em tempo de execução", ou seja, seus valores são atualizados e modificados.

O estado em que se encontram estes registradores, em determinado momento do processamento de um processo é chamado de contexto de execução, ou seja, contexto de execução é o nome dado para o conjunto de informações de um processo em um determinado momento de sua execução entre o início e o fim de sua atividade.

Vale destacar que o contexto de execução NÃO é o resultado final do processamento e sim estados intermediários do processamento.

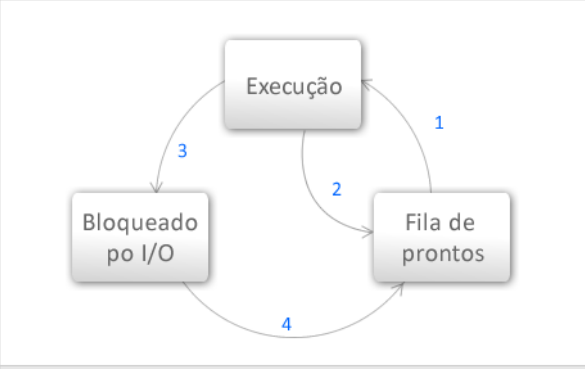

Figura F.1 - Estados de processo e suas transições
Fonte: do Autor

O contexto da execução pode ser comparado a uma fotografia do processador.

Chaveamento de contexto

A multiprogramação consiste em trocar o processo que está executando a cada período, de forma que todos os processo que estão carregados na fila de prontos "ganhem" o processador por um pequeno intervalo de tempo, tenham uma fatia de processamento garantida e retornem para a fila de prontos. Este procedimento, que alterna processos na execução, acontece constantemente enquanto houver

processos na fila. O chaveamento de contexto é o nome dado a esta troca entre os processos que estão na fila de processos prontos e o que está em execução.

O tempo[1] de troca entre os processos é definido pelo sistema operacional em conjunto com o hardware e a título de exemplo, vamos utilizar uma media de 20ms[2] . Ou seja, a cada 20ms, o hardware envia uma requisição de interrupção para o processador, avisando que é hora de trocar de processo em execução.

Dinâmica do chaveamento do contexto entre pronto e execução

- processo A está em execução....
- o quantum do processo A acaba e o hardware envia uma interrupção para o processador avisando que é hora de troca de processo;
- sistema operacional faz uma cópia do contexto da execução do processo A (que parou), guardando estas informações no bloco descritor de processo;
- o próximo processo da fila de prontos (processo C) é então carregado no processador, neste momento o sistema operacional carrega o contexto de execução do processo C (guardado até então no descritor de processo) para os registradores do processador;
- O processador começa então a "nova" execução "do ponto onde parou".
- Quando os processos ENCERRAM sua execução, saem da fila de prontos.

Dinâmica do chaveamento do contexto entre execução e bloqueado

- processo A está em execução....
- Ao executar um pedido de I/O, uma interrupção é gerada e o processo é bloqueado;
- O contexto de execução do processo A é então copiado para a fila de processos bloqueados, que irão aguardar o resultado da requisição de I/O.
- O próximo processo da fila de prontos (processo C) é então carregado no processador, neste momento o sistema operacional carrega o contexto de execução do processo C (guardado até então no descritor de processo) para os registradores do processador;
- O processador começa então a "nova" execução "do ponto onde parou".
- Quando o resultado da I/O executada pelo processo A entregar seu resultado, o processo sai da fila de bloqueados e volta para a fila de prontos.

Este ciclo repete-se enquanto houver processos na fila de execução. Sempre ocorrendo a cópia do contexto de execução que está armazenado no bloco descritor de processo e o processador ou do processador para o bloco descritor de processos.

A organização da fila de processos prontos e a escolha do próximo processo a "ganhar" o processador, é definido pelo algoritmo escalonador.

Escalonador

Escalonador de curto prazo,também chamado scheduler, é a parte do sistema operacional que organiza e determina o funcionamento da fila de prontos e o tempo que cada processo terá de execução no processador.

O escalonador é avaliado em dois aspectos principais, que são;

1. Quantum é o nome dado à quantidade tempo que um processo tem no processador.
2. Leia-se 20 milisegundos, onde 1 milisegundo = a milésima parte de 1 segundo.

Quanto a produção (*throughput*[1])

- o escalonador precisa manter o processador ocupado por mais tempo e produzir mais em menos tempo;
- basicamente, queremos que o processador seja bem utilizado e não fique ocioso.

Quanto ao tempo de resposta (*turnaround time*)

- baixo tempo médio de espera na fila do processador;
- basicamente, queremos que um processo não fique muito tempo esperando na fila.

Conforme o tipo de escalonador aplicado no sistema operacional, teremos um impacto diferente nos diferentes tipos de processos.

Tipos de processos

I/O Bound

São processos caracterizados por fazer mais uso de requisições de I/O. Estes processos não "gastam" processador porque trabalham mais com entrada e saída. Exemplos:

- editores de texto: basicamente controlam teclado, mouse, discos, etc;
- browsers: basicamente utilizam rede;

CPU Bound

São processos caracterizados por fazer mais uso de CPU. Estes processos precisam de CPU para trabalhar, fazendo pouca I/O. Exemplos:

- programas de criptografia: basicamente fazem muitos cálculos matemáticos para aplicar nos dados a serem criptografados/descriptografados.
- programas de cálculos científicos: cálculos matemáticos.

Algoritmos escalonadores

Estes algoritmos escalonadores são clássicos, ou seja, nem sempre são ou foram utilizados comercialmente.

Algoritmo FIFO *(first-in/first-out)*

Este algoritmo implementa uma fila simples, onde o primeiro processo que chega na fila é executado até seu encerramento (processo não sai do processador até terminar seu processamento).

Vantagens:

- Favorece processos do tipo CPU Bound porque estes "pegam" a CPU e não soltam mais;

Desvantagens:

- Causa um aumento no tempo de espera da fila;

Analogia com o banco:

- Este escalonador equivale a uma fila de banco, com apenas UM caixa, onde a ordem de atendimento é definida pela chegada na fila. A insatisfação dos clientes está em:
  - Não há atendimento prioritário;
  - Se algum cliente tiver muitos papéis para serem processados, o tempo de todos na fila irá aumentar.

1. É a quantidade de dados processados em um determinado espaço de tempo.

Algoritmo SJF *(Shortest Job First)*

Trabalhos mais curtos primeiro - este algoritmo implementa uma fila ordenada pelo menor tempo de execução, ou seja, quanto menor o tempo de execução mais na frente da fila o processo está.

Vantagens:

- Favorece processos I/O Bound, porque são rápidos na ocupação do processador;
- Processos rápidos ficam pouco tempo na fila de espera – diminui o tempo médio de espera.

Problemas:

- Postergação indefinida – processos considerados lentos ficarão esperando na fila por tempo indeterminado;
- Impossibilidade de implementação – é impossível prever o tempo de execução de um processo.

Analogia com o banco:

- Este escalonador equivale a uma fila de banco, com apenas UM caixa, onde a ordem de atendimento é definida pela quantidade de papéis a serem processados – se for rápido passa na frente. A insatisfação dos clientes está em:
    - Não há atendimento prioritário;
    - Se algum cliente tiver muitos papéis para serem processados, irá ficar na fila aguardando até que todos os "rapidinhos" sejam atendidos;
    - Muitos clientes afirmam que seu atendimento é "rapidinho", mas na verdade tem muitos papeis para processamento – causando indignação na fila.

Algoritmo por prioridade

Trabalhos marcados com prioridade primeiro – este algoritmo implementa uma fila ordenada pela prioridade, ou seja, todo processo deve ter uma marca de prioridade[4] . Esta prioridade irá definir a posição a ser ocupada na fila.

Vantagens:

- Favorece processos com prioridade – independentemente de tipo;

Desvantagens:

- Postergação indefinida de processos com baixa prioridade;

Analogia com o banco:

- Este escalonador equivale a uma fila de banco, com apenas UM caixa, onde a ordem de atendimento é definida pela prioridade do cliente, que se tiver prioridade alta passa na frente. A insatisfação dos clientes está em:
    - Clientes com alta prioridade são sempre atendidos primeiro – deixando os de baixa prioridade por longos períodos na fila;

Algoritmo por prioridade com *aging*

Para eliminar a desvantagem de postergação indefinida do algoritmo escalonador por prioridade, podemos implementar um mecanismo de aumento de prioridade conforme o tempo de espera na fila. Isto é conhecido por aging[5] - e no caso, a cada "rodada" de execuções, a prioridade do processo que está na fila aumenta, até que o mesmo será atendido.

1. A definição da prioridade é também função do sistema operacional, mas não do escalonador.
2. Envelhecimento.

Algoritmo por prioridade preemptivo e não-preemptivo

- Prioridade preemptivo – neste algoritmo escalonador, sempre que um processo de maior prioridade (do que o que está sendo executado) chegar na fila de prontos, o algoritmo retira o processo em execução e entrega o processador ao recém-chegado de maior prioridade.
- Prioridade não-premptivo – neste algoritmo escalonador, quando um processo de maior prioridade (do que esta sendo executado) chegar na fila de prontos, o algoritmo não interfere na execução. Espera que a execução acabe para entregar o processador ao recém-chegado de maior prioridade.

Algoritmo *Round Robin* (fatia de tempo)

Este algoritmo escalonador é o mais comum entre os sistemas operacionais convencionais e que viabiliza a multiprogramação efetivamente – o algoritmo trabalha com fatia de tempo. Cada processo na fila tem um tempo de execução (já definido como quantum); a cada quantum, o processador troca de processo em execução (pelo mecanismo de chaveamento de contexto).

Este algoritmo ainda pode ser implementado, de forma associada, ao algoritmo por prioridade (que poderá ser preemptivo ou não-preemptivo). Garantindo um mecanismo de escalonamento de fatia de tempo com prioridade.

Múltiplas filas

Sistemas operacionais atuais trabalham com o conceito de múltiplas filas, em que o processador de duas ou mais filas de espera para execução, cada qual com um algoritmo escalonador específico. Podemos, por exemplo, termos um sistema com duas filas de espera, sendo:

Fila de processos do tipo *foreground*

Processos com interação direta do usuário, normalmente executados na presença do usuário e com necessidade de baixo tempo de resposta. Editores de texto e planilhas são típicos exemplos de processos deste tipo.

Esta fila poderia trabalhar com um algoritmo escalonador *round-robin* com prioridade preemptiva. Certamente os usuários de um sistema deste tipo teriam uma boa relação entre produção e de tempo de resposta do sistema computacional.

Fila de processos do tipo *background*

Processos sem interação de usuário, normalmente executados sem a presença do usuário ou disparados em *batch*. Podem, por exemplo ser executados durante a madrugada (quando normalmente ninguém mais está usando o sistema). Processos para cálculo de folha de pagamento ou processos de computação científica podem ser exemplos deste tipo de fila. Tipicamente, processos que consomem muita CPU.

Esta fila poderia trabalhar com um algoritmo escalonador do tipo fila (*FIFO*) e os processos seriam executados do início ao fim sem problemas (até porque não tem usuário requisitando a máquina).

Síntese

- bloco descritor de processo funciona como uma estrutura do sistema operacional que faz o controle dos processos que podem ser criados e dos processos existentes e prontos para execução; além disso esta estrutura armazena dados vitais à execução dos processos como o contexto de execução, prioridade e outras informações;
- contexto de execução é o conjunto de informações que o processo tem, e que mostram o estado da execução do processo. Estas informações são usadas para parar/continuar a execução do processo "do ponto em que parou" no momento do chaveamento de contexto;
- chaveamento de contexto é a troca feita entre um processo que está sendo executado (e parou) por outro que está na fila de prontos (e ganhou o processador). Nesta ação, o sistema operacional salva o contexto de execução do processo em execução diretamente no bloco descritor de processo e carrega o contexto de execução do novo processo nos registradores do processador (para este continuar de onde parou). Esta ação acontece por tempo indeterminado enquanto houverem processos a serem executados.
- Escalonador é o mecanismo responsável pela organização da fila de espera de processos prontos. Há diversos tipos de algoritmos escalonadores.
    - **FIFO** – first-in, first-out é o algoritmo de fila – a organização é por ordem de chegada; favorece processos CPU-bound e causa o aumento do tempo de espera da fila;
    - **SJF** – Shortest job first é o algoritmo de trabalhos mais rápidos primeiro; favorece processos rápidos e causa postergação indefinida de processos considerados lentos. Além do mais, não há como antever o tempo de execução de um processo.
    - **Prioridade** – é o algoritmo que atende primeiro processos com prioridade mais alta; pode causar postergação indefinida de processos com baixa prioridade;
    - **Prioridade com aging** – é um algoritmo que, a cada "rodada" de execução, promove o aumento de prioridade dos processos que estão na fila – evitando assim a postergação indefinida.
    - **Preemptivo vs não-preemptivo** – algoritmos preemptivos retiram o processo que está em execução do processador para atender a um novo processo de maior prioridade na fila; enquanto não-preemptivos não retiram o processo atualmente em execução.
    - **Round-robin** – algoritmos de fatia de tempo, em que cada processo ganha um tempo (quantum) para execução. Além disso, este algoritmo pode ser implementado com controle de prioridade preemptivo ou não-preemptivo. Este é um algoritmo largamente utilizado em sistemas operacionais comerciais.
    - **Múltiplas filas** – conceito em que o sistema operacional possui múltiplas filas de atendimento, separadas por tipo de processos.

Atividades

1. O exercício a seguir tem como objetivo demonstrar a aplicação e o efeito do controle de prioridade na execução de processos em sistemas Linux. O Linux trabalha com prioridades que vão de -19 (MAIOR prioridade) a 19 (MENOR prioridade), sendo que Para isto será utilizada a máquina virtual preparada nas unidades anteriores.

a) Executar o comando ALT+F2 – e no campo executar aplicativo digitar glxgears, que irá abrir uma tela como a apresentada na figura F.2. Executar o glxgears DUAS VEZES, ficando com duas telas idênticas.

Figura F.2- glxgears
Fonte: do Autor

b) Abrir um terminal e digitar o comando **$ top** para monitorar os processos em execução;

- Os processos glxgears irão competir pelo processador entre si e com outros processos.
- O comando **top** mostra diversas colunas entre elas a coluna **PID**, que identifica o número do processo; a coluna **NI** que identifica a prioridade de um processo e a coluna **COMMAND** que identifica o nome do processo;
- devemos conseguir "ver" dois processos chamados glxgears com PID's diferentes e prioridade 0;

c) Abrir um novo terminal e logue como root[6] (ajustar na tela para que se consiga ver as duas telas glxgears, a tela do comando top e este novo terminal). O comando utilizado será o renice, que muda a prioridade de um processo em tempo de execução. Neste novo terminal, vamos modificar a prioridade de um dos glxgears em execução.

```
# renice -n 19 -p <PID ESCOLHIDO>
```

- -n 19 é a menor prioridade possível que um processo pode ter;
- -p <PID ESCOLHIDO> é o número PID do glxgears que você elegeu para o teste.

d) Execute o comando com diversas prioridades e observe anotando o que acontece.

- -n 19
- -n 10
- -n 0
- -n -10
- -n 19

2. Pesquisar qual(quais) é(são) o(s) algoritmo(s) escalonador(es) dos sistemas operacionais abaixo:

a) AIX
b) Linux
c) BSD

6. Usar comando $su – root.

TICs

G

Gerenciamento da memória

Unidade G
Sistemas Operacionais

UNIDADE G

GERENCIAMENTO DA MEMÓRIA

Introdução

Na unidade anterior, verificamos que o sistema operacional trabalha com processos, que são programas em execução. Esta processo precisa ser "segurado" em algum lugar enquanto está sendo executado, o que inclui seu código e seus dados. O hardware e o sistema operacional mantém os processos em uma área denominada memória principal – também conhecida como RAM.

Cabe ao sistema operacional, em parceria com o hardware, fazer o gerenciamento eficiente deste espaço onde ficam os processos durante sua existência. Nesta unidade, vamos estudar o gerenciamento de memória clássico utilizado em computadores. Vale salientar que sistemas computacionais modernos (comerciais) aumentam a complexidade desta gerência.

Tipos de memória dentro de um sistema computacional

Os sistemas, trabalham separando a memória física e a memória lógica detalhadas a seguir.

A memoria física

Memória física é aquela implementada pelos circuitos eletrônicos, que é palpável, que queima e que frequentemente precisamos aumentar. Ocorre que esta memória não é endereçada diretamente pelo processo, ou seja, o sistema operacional cria uma interface para que os processos façam uso do recurso memória. A memória física possui um espaço de endereçamento físico que são os endereços dos circuitos.

A memoria lógica

Memória lógica é aquela que o processo vê e usa; esta memória é mapeada na memória física, ou seja, o processo tem um área de memória que somente ele acessa e conhece, no entanto ele (o processo) não tem ideia de qual lugar da memória física está sendo usado. Esta memória lógica possui um espaço de endereçamento lógico que somente o processo tem acesso, ou seja, cada processo tem o seu próprio espaço de endereçamento.

MMU – Memory Management Unit

Unidade de Gerenciamento de memória é um circuito, implementado no hardware (normalmente interno ao processador) que provê os mecanismos básicos de gerenciamento de memória. Disto, decorre que o gerenciamento de memória que o sistema operacional implementa está intimamente ligado ao hardware utilizado. A figura G.1 representa a relação MMU/memória.

É na MMU que o ocorre o mapeamento dos endereços lógicos do processo nos respectivos endereços físicos dos circuitos, além de prover algum mecanismo de proteção.

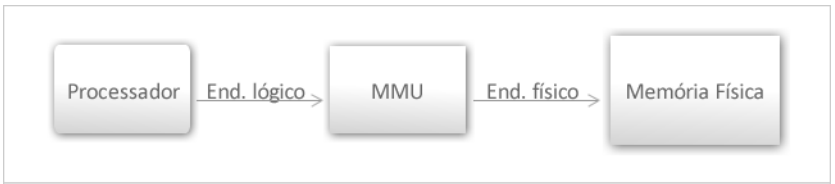

Figura G.1 - Diagrama MMU
Fonte: Adaptado de: Sistemas Operacionais, CARISSIMI, Rômulo et al, pg 100, 3ª ed, Bookman.

MMU – Mecanismos de proteção

Sempre que temos processos carregados na memória, precisamos garantir que os processos não invadam a área de memória de outro processo. A seguir, são apresentados dois possíveis métodos de proteção de memória, sendo que diversos outros mecanismos podem ser implementados. Nos dois métodos a seguir, os registradores de limite são informações do contexto de execução do processo e devem ser armazenados no descritor de processo também.

Proteção por limite inferior/superior

A ideia é que a MMU saiba quais são os limites de endereçamento que o processo tem. O processo não acessar nenhum endereço abaixo do limite inferir e acima do limite superior. Se o processo tentar acessar algum endereço fora deste escopo será gerado uma interrupção informado ao sistema operacional que houve tentativa de acesso de endereço ilegal. O esquema é mostrado na figura G.2.

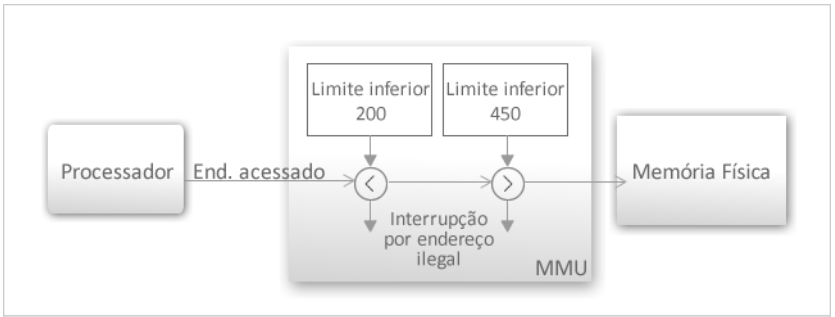

Figura G.2 - MMU com proteção por limite superior/inferior
Fonte: Adaptado de: Sistemas Operacionais, CARISSIMI, Rômulo et al, pg 100, 3ª ed, Bookman.

No esquema da figura G.2, o endereço acessado pelo processo, deverá estar entre 200 e 450. Neste caso, o endereço lógico (antes da MMU) será o mesmo endereço físico (depois da MMU). Caso esteja fora do range estabelecido pelos registradores, a MMU gera uma interrupção e encerra o processo.

Proteção por limite/base

Outro mecanismo de proteção da memória é uso do esquema registrador limite/base (figura G.3) em que a MMU sabe o limite superior e calcula o deslocamento do registrador base para acessar o endereço. Aqui já temos um mecanismo de mapeamento entre o endereço lógico e o endereço físico (real).

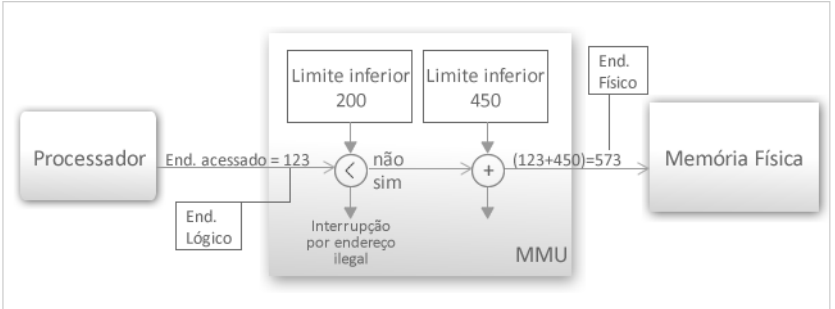

Figura G.3 - MMU com proteção por limite/base
Fonte: Adaptado de: Sistemas Operacionais, CARISSIMI, Rômulo et al, pg 101, 3ª ed, Bookman.

Formas de gerência de memória

A seguir, são apresentados alguns dos métodos clássicos de gerenciamento de memória. Vale lembrar que aplicações comerciais modernas utilizam-se de variações e adaptações destes mecanismos.

Partições Fixas

Considerando sistemas multiprogramados, está é a forma mais simples de gerenciamento de memória. Quando o sistema operacional é carregado, divide a memória disponível em diversos partições fixas e de tamanho variado. Conforme os processos vão sendo alocados, as partições vão sendo ocupadas de forma aleatória. A figura G.4 mostra este esquema de divisão da memória.

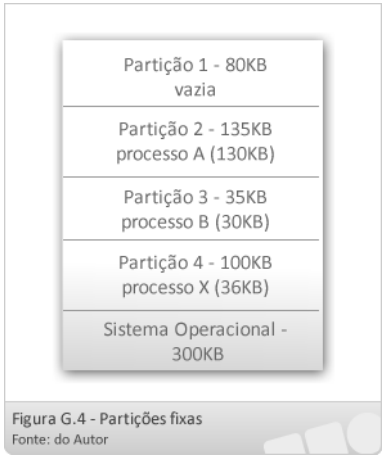

Figura G.4 - Partições fixas
Fonte: do Autor

Vantagens

- é um sistema bastante simples de ser implementado.

Desvantagens

- fragmentação interna: os processos carregados nas partições normalmente não ocupam todo o espaço disponível. Exemplos:
  - partição 2 é de 135KB, mas o processo ocupa apenas 130KB; foram desperdiçados 5KB que não poderão ser aproveitados em nenhum outro caso.
  - Partição 4 é de 100KB com um processo de apenas 36KB; foram desperdiçados 64KB que não poderão ser aproveitados em nenhum outro caso.
- Fragmentação externa
  - temos uma partição livre de 80KB e um processo de 81KB para ser carregado; ocorre que temos memória livre, mas não poderá ser utilizada porque é menor que o processo carregado. Não há um reordenamento do espaço disponível para uma ocupação “mais inteligente.

Partições variáveis

Neste mecanismo de gerenciamento todo o espaço disponível é controlado pelo sistema operacional e as partições vão sendo criadas e ocupadas por processos no tamanho exato da partição.

Vantagens

- não possui fragmentação interna, já que a partição usada é alocada com o tamanho exato do processo – isto significa um melhor aproveitamento do espaço de memória.

Desvantagens

- Apresenta fragmentação externa, porque conforme o sistema vai alocando espaços vai gerando lacunas;

- exige um mecanismo "eficiente" para gerenciamento de lacunas;

A figura G.5 procura representar a dinâmica das partições variáveis. As situações são explicadas a seguir:

a) temos o processo A com 85KB, B com 150KB, C com 50KB e um espaço disponível de 100KB;
b) O processo B encerrou, gerando uma lacuna de 150KB, enquanto os outros processos continuam em execução;
c) Um novo processo foi carregado (processo D) ocupando 100KB da lacuna gerada em b). com esta ocupação, uma lacuna de 50KB é gerada.

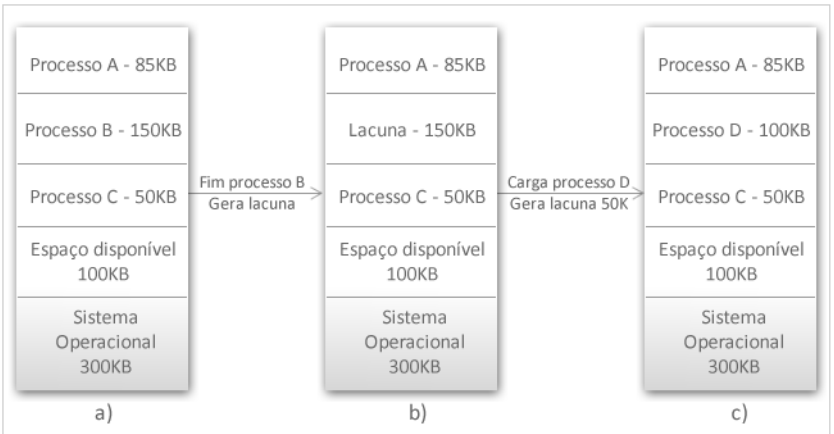

Figura G.5 - Partições variáveis
Fonte: do Autor

É necessário que o sistema tenha o controle dos espaços disponíveis e que tenha um mecanismo de escolha de "qual espaço" utilizar na locação de um novo processo. Os mecanismos possíveis são:

- *First-fit* – utiliza a primeira lacuna com tamanho mínimo necessário disponível;
- *Best-fit* – utiliza a lacuna que resultar na menor sobra (que gere uma nova lacuna de menor tamanho possível);
- *Worst-fit* - utiliza a lacuna que resultar na maior sobra (que gere uma nova lacuna com maior tamanho possível);
- *Circular-fit* - utiliza a lacuna disponível após a última sobra.

Este controle costuma ter um alto custo computacional, ou seja, o processador e o sistema operacional perdem muito tempo útil no controle destas estruturas.

Parágrafos

Pode-se reduzir o problema da fragmentação adotando unidades de alocação de tamanho fixo (32 bytes por exemplo). Estes blocos fixos são conhecidos como parágrafos. Quando um processo precisar de memória, irá ocupar um numero exato de parágrafos (que são blocos de 32 bytes). Isto permite ao sistema ter uma fragmentação interna máxima de 31bytes por processo carregado, o que é bem pouco significativo.

A figura G.6 procura representar a divisão por parágrafos. A memória está dividia em n blocos de 32 bytes (parágrafos) o processo A ocupa 3 blocos; o processo B ocupa 2 blocos; o processo C ocupa 3 blocos. Supondo que o processo B seja de 33 bytes, o maior espaço que desperdiçado será de 31 byte, que foram alocados mas não utilizados.

Este mecanismo mantem a necessidade de controle das lacunas disponíveis e por consequência ainda apresenta a desvantagem da fragmentação externa – tal qual o sistema de partições variáveis.

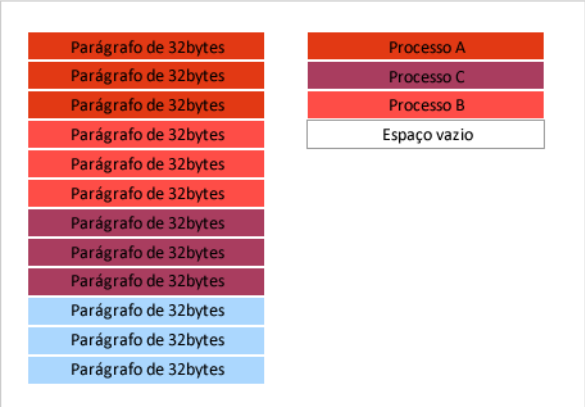

Figura G.6 - Parágrafos
Fonte: do Autor

Paginação

A paginação é um mecanismo de gerenciamento de memória que apresenta pequena fragmentação interna e nenhuma fragmentação externa. Isto garante ao sistema que o implementa um melhor aproveitamento do hardware de memória. Páginas são blocos de alocação de tamanho fixo, que são ocupados por processos.

Basicamente, a memória física é dividia em páginas e estas páginas são mapeadas na memória lógica. Este mapeamento é feito pelo uso de uma tabela de páginas, que faz referência entre as páginas lógicas e as páginas físicas; além disso, as páginas possuem um registro de deslocamento interno, facilitando a localização da informação pelo sistema operacional.

Vale lembrar que memória lógica é a memória que o processo usa/vê e a memória física é a placa de memória efetivamente. A o interfaceamento entre as duas memórias é feito pelo hardware MMU, que no caso, irá implementar a tabela de paginas para o mapeamento. A figura G.7 procura representar este esquema.

Na figura, vemos a memoria lógica dividida em 3 páginas de 4 bytes cada e cada página com um processo (os processos estão divididos em 4 partes); vemos a memória física dividida também em n páginas. Cada página é identifiada com um endereço de 5 bits, sendo que os 3 primeiros bits (mais significativos) identificam o número da página enquanto os 2 últimos bites (menos significativos) indicam a localização (deslocamento) do byte dentro da página.

Está sendo mostrado também uma tabela de páginas, que é montada dentro da MMU e faz a relação entre as páginas. No caso:

- a página lógica 000 está associada à página física 010;
- a página lógica 001 está associada à página física 101;
- e a página lógica 010 está associada a página 000;

Para localizar a informação Y2, vemos que está na página lógica 001, com deslocamento 01. este endereço é convertido na tabela de páginas para página física 101, com deslocamento 01.

- Y2 está no endereço físico 101 01.

Para localizar Z3 por exemplo, verificamos que o mesmo está na página lógica 010, com deslocamento 10. Este endereço é convertido pela tabela de páginas e encontramos a página lógica 000, com deslocamento 10.

- Z3 está no endereço físico 000 10.

Observa-se que não há necessidade de haver continuidade entre as páginas, ou seja, a página pode ser montada em qualquer localização da memória física.

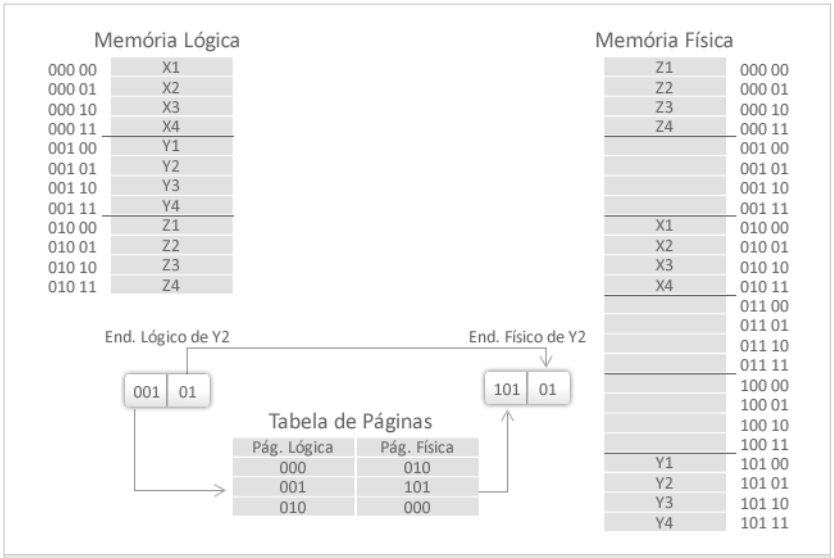

Figura G.7 - Página
Fonte: Sistemas operacionais. Oliveira, Rômulo Silva et al. 3ª ed. Porto Alegre: Bookman. Pg 106.

O problema deste mecanismo de paginação está em definir o melhor tamanho de página, uma memória dividida em páginas grandes terá uma quantidade menor de páginas, dando menos "trabalho" para o processador cuidar do mapeamento. Por outro lado, isto aumenta a fragmentação interna (em cada página). Conforme Oliveira "na prática, os tamanhos de página variam entre 1Kbyte e 8Kbytes".

Alguns outros controles também precisam ser implementados, como um mecanismo controle de páginas livres e controle de páginas ocupadas.

Segmentação

Outra forma de gerenciamento de memória, mais intuitiva para o programador e o compilador é a segmentação, que divide a memória em segmentos tipicamente correspondentes a: código, dados estáticos, dados dinâmicos e pilha de execução.

Neste mecanismo a memória lógica é dividida em segmentos; uma endereços desta memória passa a ser identificado por um segmento + um deslocamento em relação ao inicio do segmento. Na carga do processo, estes segmentos são copiados para a memória física e a tabela de segmentos (também "hospedada" na MMU) é construída.

Para endereçar sua área de memória, os processos geram endereços lógicos compondo o número do segmento associado a um deslocamento dentro do segmento. A MMU recebe esta informação e procura o segmento correspondente na tabela de segmentos e o deslocamento é comparado com o limite – se exceder, há uma violação de endereço e uma interrupção de acesso ilegal a memória é gerada, encerrando o processo. Se o deslocamento é valido, a MMU soma o deslocamento ao limite base para localizar o endereço na memória física.

A figura G.8 procura demonstrar o mecanismo de segmentação.

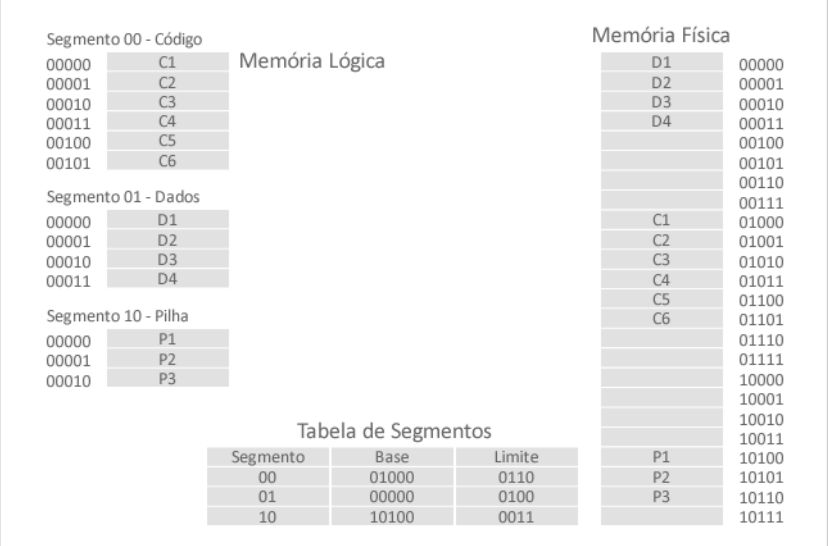

Figura G.8 - Segmentação
Fonte: Sistemas operacionais. Oliveira, Rômulo Silva et al. 3ª ed. Porto Alegre: Bookman. Pg 112.

Para localizar informações em um sistema de memória segmentada:

Localizando D1

- o endereço lógico gerado será: segmento 01 com deslocamento 00000;
- verificando na MMU, o segmento 01 tem base 00000 e limite 0100;
    - a MMU diz que o endereço procurado é válido, porque deslocamento 00000 está entre a base e o limite;
    - endereço físico é a soma entre base 00000 com deslocamento 00000, no caso 00000;

Localizando C3

- o endereço lógico gerado será: segmento 00 com deslocamento 00010;
- verificando na MMU, o segmento 00 tem base 01000 e limite 0110;
    - a MMU diz que o endereço procurado é válido, porque deslocamento 00010 está entre a base e o limite estabelecidos;
    - endereço físico é a soma entre base 01000 com deslocamento 00010, no caso 01010;

Localizando P2

- o endereço lógico gerado será: segmento 10 com deslocamento 00001;
- verificando na MMU, o segmento 10 tem base 10100 e limite 0011;
    - a MMU diz que o endereço procurado é válido, porque deslocamento 00001 está entre a base e o limite estabelecidos;
    - endereço físico é a soma entre base 10100 com deslocamento 00001, no caso 10101;

Localizando P4

- este é um endereço “fantasma” - não existe na memória lógica;
- o endereço lógico INFORMADO É: segmento 10 com deslocamento 00011;
- verificando na MMU, o segmento 10 tem base 10100 e limite 0011;
    - a MMU diz que o endereço procurado NÃO É válido, porque deslocamento 00011 mais base 10100 (10111) excedem o limite estabelecido na MMU 0011);

Segmentação paginada

Apesar de ser mais intuitiva para os programadores, a segmentação traz novamente o problema da fragmentação externa, causado pelas repetidas alocações/liberações de blocos de memória (segmentos).

A segmentação paginada resolve o problema desta fragmentação, fazendo uma associação entre segmentação e paginação.

Neste caso temos uma memória lógica que lista uma tabela de segmentos e cada segmento contém uma tabela de páginas (exclusiva do segmento). O endereço de uma informação é composto pelo conjunto número do segmento, número da página e deslocamento dentro da página.

Swapping

O swap é uma técnica que permite ao sistema operacional trabalhar com mais memória física do que realmente se tem disponível no banco da memória principal. O gerenciador de memória trabalha com uma área de disco rígido (HDD) reservada especialmente para "expandir" a memória física do computador em caso de falta deste recurso em tempo de execução de processos.

Esta área de armazenamento especial é de baixíssima velocidade, ou seja, tem um tempo de acesso muito alto; desta forma, o sistema operacional "transfere" para esta área preferencialmente os processos que sofreram algum tipo de bloqueio e estão fora da lista de processos prontos. O processo ainda existe mas por algum motivo (I/O por exemplo) está fora da lista de prontos.

O swaping é um recurso necessário como segurança para o funcionamento do sistema operacional, no entanto, não queremos que seja utilizado – dado que sua velocidade de operação é muito lenta.

Síntese

- O sistema operacional faz o gerenciamento da memória do sistema de forma integrada com o hardware, ou seja, há uma interdependência entre hardware e sistema operacional;
- O sistema operacional trabalha com dois tipos de memória, que são:
    - memória física – são os componentes eletrônicos da memória propriamente dita. Os processos não tem acesso direto ao hardware de memória;
    - memória lógica – é a memória que o processo vê e utiliza;
    - MMU – Memory Management Unit é a unidade de gerenciamento de memória, faz a interface entre memória lógica e memória física, além de implementar diversos mecanismos de proteção da memória física.
- Gerenciamento – exitem diversas formas/mecanismos de gerenciamento da memória, sendo que alguns são mais/menos eficientes:
    - Partições fixas – a memória é dividida em espaços fixos de tamanhos variados; problemas são fragmentação externa e interna;
    - Partições variáveis - memória é dividida em partições com o tamanho exato solicitado pelo processo; causa muita fragmentação externa, devido a grande quantidade de lacunas entre partições criadas;
    - Partições com parágrafos – memória é dividida em blocos de tamanho fixo e os blocos são alocados aos processos conforme a demanda. Possui baixa fragmentação interna e alguma fragmentação externa.
    - Paginação – memória lógica e física são dividias em páginas e na MMU há uma tabela que relaciona a página existente na memória lógica com a página existente na memória física. Não possui fragmentação externa e pouca fragmentação interna.
    - Segmentação – memória lógica e física são dividias em segmentos e na MMU há uma tabela que relaciona o segmento na memória lógica com o segmento na memória física. Causa fragmentação externa.
    - Segmentação paginada – cria um mecanismo de segmentos com páginas; cada segmento tem um conjunto de páginas; a MMU faz o mapeamento entre os segmentos e as páginas da memória lógica com a memória física;
    - Swaping – é um mecanismo de expansão da memória física no disco rígido; no caso de falta de memória física disponível, processos podem ser "transferidos" para a área de swap onde normalmente são bloqueados, devido ao baixo tempo de resposta deste tipo de área.

Atividades

1. Represente a memória física para a seguinte tabela de páginas e a seguinte representação da memória lógica.

00000	x1
00001	x2
00010	x3
00011	x4
00100	a1
00101	a2
00101	a3
00110	a4
00111	v1
01000	v2
01001	v3
01010	v4

pág. Lógica	Pág. Física
000	100
001	000
010	001

TICs

H

Sistema de arquivos

Unidade H
Sistemas Operacionais

UNIDADE H

1. SISTEMA DE ARQUIVOS

Introdução

Apesar de todos os procedimentos internos de um sistema operacional serem importantes para o funcionamento da máquina, é com o sistema de arquivos que o usuário mais entra em contato. Pode-se dizer que o usuário "vê" o sistema de arquivos como sendo a parte mais importante do sistema operacional – até porque muitas informações pessoais estão confiadas a eficiência e confiabilidade do sistema de arquivos de um determinado sistema operacional. Nesta unidade, vamos estudar as os principais conceitos que compõem um sistema de arquivos com seus principais mecanismos de gerenciamento.

Conceitos básicos

Arquivos

Arquivos são estruturas, que armazenam dados sensíveis para o usuário, ou seja, que tem alguma importância para este. O arquivo pode ser de vários tipos, mas sempre com alguma finalidade dentro do sistema. O usuário, que neste caso poderá ser um usuário comum ou um programador avançado, confia que seus dados estejam devidamente armazenados em arquivos.

Diretórios

Diretórios (atualmente designados por pastas) são arquivos especiais, que armazenam listas de arquivos e outros diretórios. São utilizados para facilitar a organização dos sistemas de arquivos modernos. Basicamente, não concebemos um sistema operacional que não permita uma separação de arquivos conforme nossa intenção.

Disco

A mídia é o meio onde os dados são armazenados, podendo ser um disco rígido, um cdrom, uma pendrive. Cada mídia tem um mecanismo particular de armazenar os dados e cabe ao sistema operacional apresentar uma abstração compreensível deste sistema. Em outras palavras, o sistema operacional deve oferecer uma interface amigável e eficiente para que o usuário possa manipular seus arquivos.

Partições

Uma partição é uma divisão lógica que o disco deve ter e que é gerenciado pelo sistema operacional. Todo sistema operacional trabalha com o conceito de partições para fins de possibilitar a abstração que se propõe. Um disco deve possuir pelo menos uma partição onde será gravado o sistema de arquivos daquele sistema operacional em específico.

Sistema de arquivos

Uma partição deve ser preparada com um e somente um sistema de arquivos (uma partição não comporta mais que um sistema de arquivos). Este sistema de arquivos irá definir como será o arquivo propriamente dito além de definir com o será o gerenciamento de suas operações e características.

O arquivo

O arquivo, que seria a menor parte do sistema de arquivos; que conforme visto anteriormente possui algum significado para o usuário possui também alguns atributos – que variam de sistema operacional parra sistema operacional. De forma geral os atributos podem ser:

- tipo – texto, binário, registro, imagem.
- Tamanho – informa o espaço do disco utilizado pelo arquivo em bytes;
- Último acesso – alguns sistemas informam quando o arquivo foi aberto pela última vez;
- Última modificação – sistema informa quando foi a última alteração que o arquivo sofreu;
- dono do arquivo – informa qual usuário é o dono do arquivo;
- permissões de acesso – informa quais são as permissões de acesso ao arquivo;

Operações básicas com arquivos

Naturalmente queremos que o sistema operacional nos permita fazer o "manuseio" dos arquivos que gerencia. Logo, é necessário que o sistema ofereça um conjunto de operações básicas para este serviço.

Criação

- operação para criação de arquivos sempre que for necessário;

Remoção

- operação para apagar arquivos sempre que for necessário;

Leitura

- operação de abrir um arquivo e visualizar seu conteúdo; obedecendo o conjunto de permissões especificado.

Alteração

- operação para abrir um arquivo e fazer alterações dentro deste; obedecendo o conjunto de permissões especificado.

Escrita no fim do arquivo

- modificação de um arquivo, adicionando conteúdo ao seu fim;

Execução do código contido

- especificamente para arquivos binários ou executáveis. Capacidade de execução do código.

Alteração de permissões

- deve ser possível, para o proprietário do arquivo, modificar as permissões de um arquivo.

Outras operações podem ser compostas a partir de operações básicas, como por exemplo:

Cópia de arquivo

Esta operação é composta basicamente pelas seguintes operações básicas:

- criação de novo arquivo: cria novo arquivo de destino
- leitura do arquivo: arquivo de origem é lido;
- escrita no fim do arquivo: conteúdo lido é gravado no fim do arquivo recém-criado.

•

Movendo arquivo

Esta operação é composta basicamente pelas seguintes operações básicas:

- criação de novo arquivo: cria novo arquivo de destino
- leitura do arquivo: arquivo de origem é lido;
- escrita no fim do arquivo: conteúdo lido é gravado no fim do arquivo recém-criado.
- remoção do arquivo de origem: arquivo antigo é apagado.

Estrutura interna do arquivo

Basicamente, o sistema operacional não se preocupa com o conteúdo do arquivo (não faz diferença o que tem dentro do arquivo) e, via de regra, o sistema operacional vê o arquivo como "uma sequencia de bytes, cujo significado é conhecido pelo usuário" [Oliveira, pg, 146, 2008]. Por outro lado, a estrutura interna, ou a forma com que os dados são armazenados interfere na em alguns aspectos do acesso deste arquivo.

O arquivo pode ter diversos formatos (estruturas) internas de armazenamento dos dados. Este formato está diretamente ligado à aplicação que faz uso do arquivo.

- Texto – organizados em linhas/parágrafos;
- Binário – organizado em segmentos de dados;
- Registros – organizados em campos;
- Imagem – organizados em conjuntos de pixels.

Extensão de nome

Arquivos podem ter extensões de nome, que são um adendo ao nome, frequentemente utilizados pelo usuário para auxílio na organização do usuário proprietário do arquivo, não interferindo na forma com que o sistema operacional trata estes arquivos, mas informando

As vezes arquivos com mesma função, mas de programas diferentes possuem extensões diferentes. Por exemplo arquivos texto feitos no editor Word possuem extensão .doc, enquanto arquivos texto feito no editor Broffice possuem extensão .odt.

Na tabela a seguir, são apresentados algumas extensões clássicas.

Tipo	Extensão
Arquivo texto	.txt
Arquivo fonte C	.c
Arquivo word	.doc / .docx
Arquivo Writer Broffice	.odt

Extensões clássicas de arquivos

Controle de acesso

O controle de acesso é um recurso para sistemas operacionais multiusuários para garantir a proteção dos arquivos, determinando regras de acesso e bloqueio. Todo arquivo possui um conjunto de informações informando a qual usuário ele pertence e as regras de permissões de acesso.

O controle de permissões, via de regra, começa no ato de login de um usuário quando o usuário informa seu login e senha; caso seja autenticado no sistema o usuário ganha permissão de acesso e inicia uma

sessão de trabalho, que poderá ser em um ambiente de texto ou gráfico. A partir deste momento o sistema operacional sabe quem está logado e todos os processos disparados pelo usuário "herdarão" seus direitos de acesso.

Ao fazer um acesso a um arquivo o sistema operacional confere se o usuário que está executando a ação consta na lista de permissões do arquivo acessado; estando autorizado o acesso acontece normalmente; não estando autorizado o sistema operacional emite um aviso de acesso não autorizado. Exemplo:

- arquivo notas.txt
    - pertence ao usuário professor, que tem permissão total de acesso;
    - professor pode liberar acesso do tipo somente leitura para usuário aluno;

O sistema operacional pode ainda trabalhar com o conceito de grupos de usuários, em que diversos usuários são associados a um grupo (normalmente esta associação acontece por alguma ponto em comum ou semelhança de comportamento). A partir deste ponto, poderemos liberar o acesso a um grupo de usuários. Exemplo

- arquivo notas.txt
    - pertence ao usuário professor, que tem permissão total de acesso;
    - professor pode liberar acesso do tipo somente leitura UM GRUPO de alunos;

Forma de acesso

O arquivo pode ser acessado de diversas formas, o que poderá interferir no desempenho deste ou daquele tipo de arquivo. Abaixo vemos os principais tipos de acesso.

Acesso sequencial:

O arquivo começa a ser lido do início para o fim. Este é o método mais simples de se acessar um arquivo em que as informações são buscadas parte por parte (sequencialmente).

É um mecanismo utilizado, por exemplo, por:

- compiladores que precisam acessar o arquivo-fonte de forma sequencial;
- impressora que precisa imprimir o arquivo do início ao fim.

Este método de acesso é inadequado para pesquisa de dados em arquivos do tipo registro – porque para buscar um registro em qualquer posição do arquivo, teríamos que percorrer o mesmo desde o início.

Acesso relativo:

O acesso relativo é utilizado para pesquisa em uma coleção de registros. Basicamente ao buscar uma informação, informamos exatamente o registro que queremos ler. Desta forma, temos um acesso muito mas rápido, isto porque não precisamos percorrer todo o arquivo para recuperar uma informação. Por outro lado este mecanismo é ineficiente para busca em arquivos texto (código fonte por exemplo), já que não temos uma estrutura de registros interna.

Este sistema de acesso é equivalente ao acesso a banco de dados gerenciado por um SGBD.

Posição corrente:

Neste mecanismo a leitura será feita a partir do local em que estamos dentro do arquivo (posição corrente) – não é necessário informar a posição a ser lida. Além disso, é possível reposicionar o "cursor" dentro do arquivo para novas leituras.

Implementação de arquivos

A implementação de arquivos define como um sistema operacional cria e mantém um arquivo no seu sistema. Diversos mecanismos e formas poderão ser implementados, havendo diferenças entre sistemas operacionais. Neste ponto vamos estudar a forma elementar ou clássica para implementação de arquivos.

Descritor de arquivos

O sistema operacional tem uma estrutura de controle interna, chamada descritor de arquivo[1] , que guarda todas as informações de um determinado arquivo. Em outras palavras, cada arquivo possui um descritor que "descreve" o arquivo. As informações mínimas que compõem um descritor de arquivos são:

- nome do arquivo;
- extensão do arquivo (nos casos em que for necessário);
- tamanho do arquivo;
- último acesso;
- ultima modificação no arquivo;
- identificação do proprietário do arquivo;
- lista permissões com usuários e a respectiva permissão;
- localização do arquivo no disco – informa o endereço do arquivo dentro do disco.

O descritor de arquivos é uma informação persistente, ou seja, não se perde quando o sistema é desligado. Para que isto seja possível os descritores de um sistema de arquivos são guardados no próprio disco, dentro da própria partição onde estão os arquivos a que eles (os descritores) se referem. No ato da criação da partição, o sistema operacional reserva algumas áreas do disco para armazenar guardar estes descritores – evidentemente, estas são áreas protegidas e que um usuário comum não tem acesso.

Para acessar um arquivo, o descritor de processo precisa antes ser carregado na memória e ficará carregado pelo tempo que o arquivo estiver sendo manipulado. Ao fazer o encerramento de uso do arquivo, o descritor que está na memória é "salvo" no disco, atualizando as informações para o próximo uso.

TDAA – Tabela de Descritores de Arquivos Abertos

Para fazer acesso a algum arquivo o processo solicitante precisa disparar uma chamada de sistema informando sua intenção e com qual arquivo. Esta relação é mantida, na memória do sistema operacional, em uma estrutura chamada tabela de descritores de arquivos abertos. Desta forma, o sistema operacional sabe quais arquivos estão abertos e por quais processos, além de controlar qual o tipo de acesso o processo está executando no arquivo.

Tipicamente, uma TDDA carrega todas as informações do descritor de arquivos, além de controlar quantos e quais processos estão acessando/manipulando determinado arquivo.

TAAP – Tabela de Arquivos Abertos por Processo

O sistema operacional também precisa ter controle sobre as variáveis de cada processo em relação a um determinado arquivo. Isto ocorre porque cada processo pode abrir um mesmo arquivo de formas diferentes; aqui entra uma outra estrutura de controle chamada Tabela de Arquivos Abertos por Processo que mantém informações que relacionam o processo com o descritor de arquivos aberto. Por exemplo:

1. O descritor de arquivos é uma estrutura equivalente ao descritor de processos

- modo de acesso do processo no arquivo-fonte
  - leitura/escrita (RW)
  - somente leitura (RO)
- localização do cursor de leitura dentro do arquivo;
- alterações do processo (antes do salvamento).

Procedimento de abertura de um arquivo

O procedimento de abertura de um arquivo poderá variar entre sistemas operacionais. Quando um processo emite uma chamada de sistema para abertura de um arquivo, o sistema operacional segue (basicamente) os seguintes passos:

1. procura no disco o descritor de arquivo que se busca
   a) esta pesquisa é feita no diretório corrente ou no diretório informado na chamada de sistema;
2. Se o descritor não é encontrado então o arquivo não existe e um aviso é dado ao processo solicitante;
3. Se o descritor é encontrado, um código (tipo um endereço), composto pela partição + endereço do descritor é retornado para a chamada de sistema;
4. O sistema de arquivos consulta a TDAA para verificar se o descritor de arquivo localizado já está aberto.
   a) Se NÃO ESTÁ aberto, carrega o descritor de arquivo na TDDA;
   b) Se JÁ ESTÁ aberto, incrementa o número o registro de quantos processos estão usando este descritor da arquivo;
5. Uma vez que o descritor de arquivo está carregado na TDDA o sistema operacional confere se o processo tem permissão de manipulação do arquivo;
   a) Se permissão está certa prossegue liberando o arquivo para uso do processo;
   b) Senão gera um erro que é comunicado ao processo e fecha o arquivo;

Procedimento de fechamento de um arquivo

O procedimento de fechamento de um arquivo funciona de forma semelhante nos sistemas operacionais.

Basicamente, para fechar um arquivo:

1. o sistema de arquivos localiza o descritor na TDAA e decrementa o contador de processos usando o descritor de arquivo;
2. quando este contador chegar a zero, o sistema de arquivos sabe que não há mais processos fazendo uso do arquivo; então o registro é fechado, ou seja, o espaço na TDAA é liberado para uso por outro descritor de arquivo.

Manipulando arquivos

Um sistema operacional tem um conjunto bastante grande de chamadas de sistema, disponíveis para uso dos programadores, através das bibliotecas de funções das linguagens de programação. Este conjunto de funções é conhecido como API[2] da linguagem de programação especifica. Um programador precisará ter o mínimo de domínio destas funções para programar para em determinado sistema operacional com determinada linguagem.

2. Application Programming Interface

Abrindo um arquivo com a linguagem C

Basicamente precisaremos de:

- uma variável especial (descritor de arquivos) que será um ponteiro para o endereço do arquivo - FILE *descritor;
- uma função de abertura de um arquivo – fopen(“nome do arquivo”,rw);
- uma ação para manipulação de o arquivo aberto – fputs(“frase para gravar”, arquivo);
- uma função para fechamento do arquivo – fclose(arquivo);

Codigo para criação de arquivo - baseado no curso de linguagem c - ufmg

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv){
int x;
FILE *fd;
char str[80];
char aviso[160];

if(argc != 2){
 fprintf(stderr, “Informe ./criar arquivo”);
 exit(1);}

fd=fopen(argv[1],”w”);
if(!fd) {
 fprintf(stderr, “Não foi possivel criar o arqivo”);
 exit(1);}
for(x=0;x<10;x++){
 strcpy(aviso,”Voce digitou: “);
 printf(“Digita uma string pra gravar no arquivo\n”);
 fgets(str,50,stdin);
 strcat(aviso,str);
 fputs(aviso,fd);
 if(ferror(fd)){
        fprintf(stderr, “Não foi possivel gravar no arqivo”);
        exit(1);
        }
 }
fclose(fd);
return 0;
}
```

O pequeno programa acima mostra a utilização de manipulação de arquivos. Para testar, basta compilar na maquina virtual configurada:

```
$ gcc -o arq arq.c
```

Para testar:

```
$ ./arq teste.txt
```

Síntese

- O sistema operacional é responsável por apresentar de uma interface amigável para gerenciamento de arquivos; o sistema de arquivos do sistema operacional irá viabilizar o gerenciamento dos arquivos do sistema;
- Arquivo é uma estrutura, que fica armazenada em um meio e que contem dados que fazem sentido para algum usuário, mas não tem importância para o sistema operacional.
- Diretórios são arquivos especiais – seu conteúdo são listas de arquivos e outros diretórios. Os diretórios são úteis para facilitar a organização dos arquivos do sistema/usuários.
- Um disco (ou outro meio de armazenamento) deverá ter pelo menus uma partição (que é ma divisão lógica) com um sistema de arquivos (que determina como os arquivos são armazenados).
- Todo arquivo possui uma estrutura interna, seus dados poderão ser estruturados na forma de linhas/parágrafos, registros, código ou pixels; outras formas podem ser encontradas e/ou desenvolvidas.
- O sistema operacional poderá acessar os arquivos de diversas formas (sequencial, relativo, ou posição corrente); a escolha do método irá depender do tipo de arquivo/aplicação que está sendo usada.
- O arquivo possui uma estrutura de dados para seu controle – chamada descritor de arquivo, que controla diversas informações de todos os arquivos do disco; cada arquivo possui um descritor de arquivo;
- O sistema de arquivos mantém uma tabela de descritores de arquivos abertos – para controlar quais arquivos estão abertos e por quais processos em determinado momento;
- O sistema de arquivos mantém uma outra tabela de arquivos abertos por processo – para controlar variáveis que são pertinentes para um processo especificamente;

Atividades

1. Montar um programa que faça uma cópia do arquivo criado no exercício feito pelo código anterior.

UNIDADE H

2. SISTEMA DE ARQUIVOS

Introdução

Na parte anterior, verificamos alguns conceitos fundamentais sobre o sistema de arquivos. Nesta unidade, vamos aprofundar um pouco mais na forma com que o sistema de arquivos gerencia o arquivo no disco, focando principalmente na forma com que os arquivos são "armazenados" no espaço do disco.

O HD

O HD, também chamado de memória secundária, é um dispositivo eletromecânico que armazena dados em um disco magnético. Dentro do hd, temos diversos componentes que trabalham de forma integrada. As principais partes internas do disco são (acompanhar na figura 1):

Discos (platter)

São os discos magnéticos onde as informações são gravadas. Observando a figura atentamente, pode-se verificar que o HD apresentado contém dois discos magnéticos. HD's comuns são fabricados com 2, 3 ou 4 discos, sendo que cada disco é gravável nas duas faces.

Cabeça leitora

É o componente que grava/lê o disco. Esta peça nunca encosta na superfície do disco e sim fica a uma distância milimétrica. Para cada disco, há duas cabeças leitoras (uma para cada face).

Braço de leitura

Este componente faz a movimentação da cabeça leitora na área do disco, para que o conjunto localize o local do disco onde está a informação buscada.

Figura H.1 - Partes do HD
Fonte: http://www.hardware.com.br

Saiba mais:

Hardware, O guia definitivo, disponível em:

<http://www.hardware.com.br/livros/hardware/capitulo-hds-armazenamento.html>

Geometria do disco

A organização interna do disco determina como será o trabalho do hardware e do sistema operacional para gravação/recuperação de dados. Todos os discos possuem a mesma geometria, sendo que é neste "esquema" que será feito a partição e criado o sistema de arquivos. Na figura H.2 podemos observar o esquema da geometria – no caso, representado com 4 discos.

Cada superfície do disco é dividida em trilhas (*track* na figura) e setores (*sector* na figura); as trilhas são linhas, na forma de círculo concêntrico, que são endereçadas por um número (0,1,2 ...) - a trilha mas externa (a maior delas) recebe o endereço 0. Cada trilha é subdividida em setores, de 512 bytes, onde são armazenados as informações.

As trilhas , que estão nas duas superfícies de cada disco são perfeitamente alinhas com os outros discos e suas superfícies. O conjunto de trilhas (de todos os discos) com mesmo endereço formam um cilindro (*cylinder* na figura). Na figura, vemos que a trilha 0 de todas as superfícies estão alinhadas – podemos abstrair este desenho e visualizar o cilindro. Cada cilindro recebe o endereço da trilha.

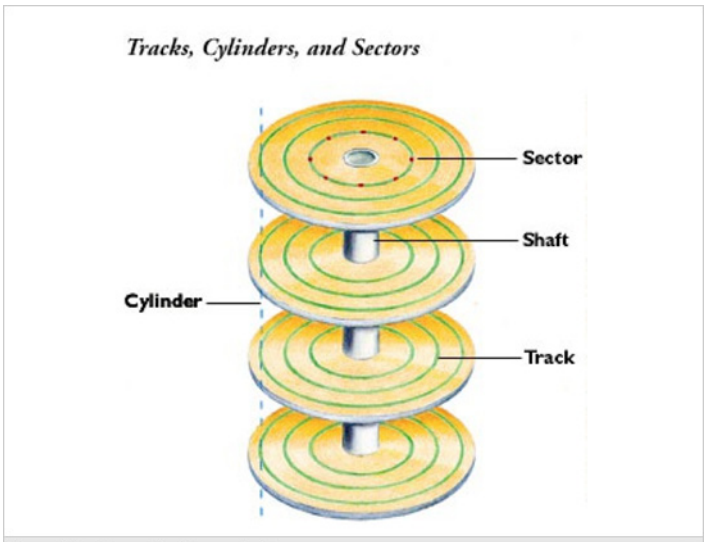

Figura H.2 - Geometria interna do disco
Fonte: http://www.hardware.com.br

O hardware do disco faz o controle desta geometria – que vem especificado pelo fabricante, por um procedimento chamado formatação física. Não podemos fazer modificações nesta formatação, porque implicaria em modificar a geometria do disco.

Acessado um dado

O acesso a um dado em um disco é um processo bastante complexo, que envolve diversas interfaces, que fazem "traduções" até que se possa chegar ao local físico em que está o dado. Neste subitem, fazemos uma síntese deste procedimento de localização de informações.

Para que nós usuários possamos acessar uma arquivo, basta informarmos em qual partição, em qual diretório e qual o nome do arquivo que queremos.

O processo envia esta informação para o sistema de arquivos (através de uma chamada de sistema), que faz uma conversão deste "endereço" para outra informação chamada bloco físico. O bloco físico é então enviado à controladora do disco, que faz nova conversão deste endereço e localiza em qual cilindro, em qual superfície e em qual setor está a informação buscada.

Uma vez localizado o dado, este é lido e enviado pela controladora ao sistema operacional, que repassa ao processo.

Alocação de arquivos

O sistema de arquivos pode fazer a organização dos setores lógicos de várias formas, cada qual com suas vantagens/desvantagens, que irão impactar no desempenho e forma de funcionamento do sistema de arquivo. Evidentemente, alguns métodos de alocação não são utilizados comercialmente, enquanto outros se aproximam bastante dos métodos de alocação nestes sistemas.

Descritor de ARQ1	Descritor de ARQ2	Descritor de ARQ3
• inicio = 3; • fim = 6;	• inicio = 8; • fim = 10;	• inicio = 11; • fim = 12;

Figura H.3 - Alocação contígua
Fonte: do autor

Observa-se que os arquivos ocupam blocos físicos do disco em sequência, por outro lado, se ARQ2 precisar aumentar de tamanho, não há blocos contíguos disponíveis. Em outras palavras, temos espaço em disco disponível, mas não podemos alocar para ARQ2. Uma solução seria reorganizar todos (ou parte) dos arquivos para que o espaço disponível passe a ficar contíguo.

Vantagens

- simplicidade para implementação;
- velocidade na busca de todos os blocos do arquivo;

Desvantagens

- inviabiliza/dificulta o crescimento do arquivo.

Alocação encadeada

Para resolver o problema do crescimento de arquivos, precisa-se de um mecanismo que facilite este crescimento. Para isto, utiliza-se uma estrutura de dados em que o cada bloco sabe o endereço do próximo bloco do arquivo. A figura H.4 procura demonstrar este recurso. No disco hipotético apresentado, há 3 arquivos e seus descritores de arquivo simplificados são:

Descritor de ARQ1	Descritor de ARQ2	Descritor de ARQ3
• inicio = 3; • tamanho = 4;	• inicio = 8; • tamanho 4;	• inicio = 11; • tamanho = 3;

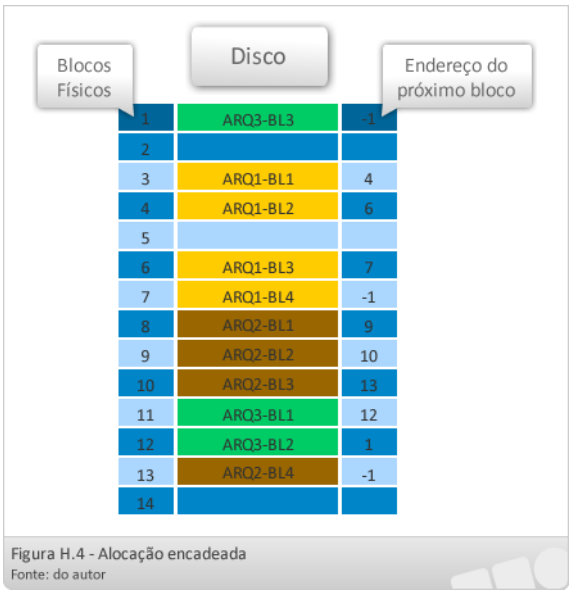

Figura H.4 - Alocação encadeada
Fonte: do autor

Cada bloco físico reserva uma parte de seu espaço para registrar o endereço do próximo bloco, por exemplo 4 bytes. Desta forma, o arquivo não cresce de forma contígua, o que pode ser observado em ARQ2, que está "espalhado" pelo disco. Além disso, o bloco descritor não aponta mais o fim do arquivo e sim seu tamanho.

O último bloco de cada arquivo aponta para um endereço chave -1 (que no caso indica fim do arquivo).

Vantagens

- permite o crescimento do arquivo;
- velocidade na busca de todos os blocos do arquivo;

Desvantagens

- "perde" espaço em disco para fazer o encadeamento - para blocos de 4 Kbytes, utilizando 4 bytes de cada bloco, perde-se 0,1% do espaço em disco para este controle.
- NÃO permite acesso relativo – para acessar o bloco 4 de ARQ2, precisamos percorrer todo o arquivo – o que é um problema para arquivos grandes;

Alocação indexada

A alocação indexada permite o crescimento do arquivo além de permitir o acesso relativo. Neste sistema, a lista de blocos ocupados pelo arquivo é transferida para dentro do descritor de arquivo, onde uma lista indexada é criada e armazena os endereços do próximo bloco. A figura H.5 procura demonstrar este esquema. No disco hipotético apresentado, há 2 arquivos, que são ARQ1 e ARQ2.

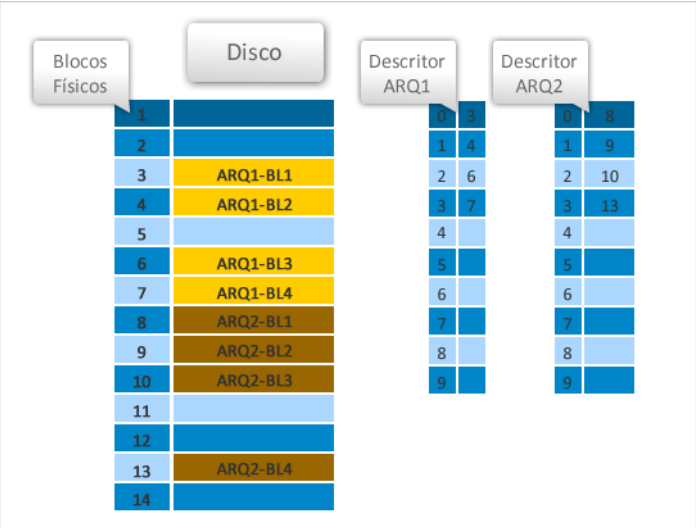

Figura H.5 - Alocação indexada
Fonte: do autor

Cada descritor de arquivo possui uma tabela indexada que registra o endereço do próximo bloco do arquivo, permitindo por exemplo buscarmos diretamente ; desta forma, podemos permitir o crescimento do arquivo e também acessar qualquer registro do arquivo diretamente.

No entanto, o crescimento do arquivo é limitado ao tamanho da tabela de índices, que no caso da figura H.5, poderemos endereçar 10 blocos de 4KB cada, o que nos permite um arquivo de tamanho máximo de 40KBytes (que é um tamanho de arquivo bem limitado).

Para permitir arquivos maiores, teríamos que ter uma tabela de índices maior - por exemplo para arquivos de 2GBytes precisamos de uma tabela com 500.000[1] índices, o que irá consumir muita memória (já que os descritores de arquivos ficam carregados na memória).

Vantagens

- permite o crescimento do arquivo;
- permite acesso relativo;

Desvantagens

- crescimento de arquivo é limitado ao tamanho da tabela de índices;

Alocação com níveis de indireção na indexação.

Para resolver o problema do crescimento de arquivos, adota-se níveis de indireção na tabela de indexação. O método de localização fica muito parecido com o método de indexação convencional. A figura H.6 procura representar o mecanismo[2].

1. 2GB / 4KB = 500.000
2. Deve-se considerar que não é possível representar muitos blocos físicos.

Figura H.6 - Alocação com níveis de indireção
Fonte: do autor

No descritor de arquivos, que agora contem 11 índices, os 10 primeiros índices são apontadores diretos, ou seja, apontam blocos os blocos físicos do arquivo diretamente – o que permite um arquivo de até 40KB[3] . Para permitir o crescimento do arquivo, utiliza-se o índice de número 10 aponta para um bloco físico no disco (no caso bloco 18) que por sua vez funciona como um indexador.

Como cada bloco físico contém 4KB e um cada indexador ocupa 4bytes, podemos ter em um bloco físico 1024 índices. Verifica-se que o bloco 18 está apontando para 1024 blocos físicos (não é possível desenhar isto); o que nos permite aumentar nosso arquivo para 4,2MBytes.

Como Fazer:

Considerando que 1KB = 1024B então 4KB = 4096bytes o valor exato disponibilizado é:

(4096 x 10 índices diretos) + (1024 índices indiretos x 4096) = 4235264 bytes que arredondando dá 4MBytes.

E se precisarmos de um arquivo maior, usamos dois níveis de indireção (apontador duplamente indireto, que no caso, irá apontar para 1024 APONTADORES de 1024 blocos físicos de 4KB. Isto permite ao arquivo atingir 4,2GBytes.

3. Sempre considerando blocos físicos de 4KB.

Como Fazer:

Considerando que 1KB = 1024B então 4KB = 4096bytes o valor exato disponibilizado com apontador duplamente indexado é: (4096 x 10 índices diretos) + (1024 índices indiretos x 4096) + (1024 x (1024 índices indiretos x 4096))= 4299202560 bytes, que arredondando dá 4GBytes.

E se ainda assim, nosso arquivo precisar aumentar de tamanho, lançamos uso do apontador triplamente indexado, que aponta para 1024 APONTADORES, onde cada um apontam para 1024 outros APONTADORES, que por sua vez apontam para 1024 blocos físicos de 4K. Isto permite ao arquivo atingir 4,4TBytes – o que, para os padrões atuais, é considerado um arquivo bem grande.

Como Fazer:

Considerando que 1KB = 1024B então 4KB = 4096bytes o valor exato disponibilizado com apontador duplamente indexado é: (4096 x 10 índices diretos) + (1024 índices indiretos x 4096) + (1024 x (1024 índices indiretos x 4096)) + (1024 x (1024 x (1024 índices indiretos x 4096))) = 4402345713664 bytes, que arredondando dá 4,4 TBytes.

Múltiplos sistemas de arquivos

Normalmente, um computador pessoal precisa ter acesso a diversos sistemas de arquivos porque acessamos diversas mídias – normalmente cada tipo de mídia tem seus próprio sistema de arquivos, por exemplo:

- CDROM – sistema de arquivos ISO9660;
- DVDROM – sistema de arquivos UDF ou RAW;
- Cartão de memória – sistema de arquivos FAT16, FAT32 e suas variações;
- HD com múltiplas partições, cada qual com seu próprio sistema de arquivos;

Desta forma, o sistema operacional precisa "reconhecer" e manipular diversos sistemas de arquivos. Logo há necessidade de uma interface universal para todos estes mecanismos para gerenciamento de arquivos. Os dois principais sistemas de arquivos para acesso múltiplo são:

- Virtual File System – VFS, desenvolvido pela SUN;
- File System Switch – FSS, desenvolvido pela AT&T;

Síntese

- O hd é um dispositivo eletromecânico complexo, para acessar um dado em sua estrutura interna, é preciso que o sistema operacional informe em qual bloco físico o dado está. Internamente, a controladora do hd transforma esta informação para um endereço contendo Cilindro, Disco e Setor.
- O sistema de arquivos é um mecanismo que organiza e gerencia a forma com que os dados são armazenados. As principais formas organização são:
  - alocação contígua: simples mas não permite o crescimento do arquivo;
  - alocação encadeada: permite o crescimento do arquivo mas não permite acesso relativo;
  - alocação indexada: permite acesso relativo e um crescimento limitado do arquivo;
  - alocação indexada com níveis de indireção na indexação: permite acesso relativo e um crescimento considerável do arquivo;

Atividades

1. Identificar as quais características estudados nos mecanismos de alocação clássicos estão presentes nos dois sistemas de arquivos FAT32 (Microsoft) e ext3 (Sistemas Unix-like).

Gerenciamento de I/O

Unidade I
Sistemas Operacionais

UNIDADE

GERENCIAMENTO DE I/O

Introdução

Sempre que fazemos uso de um sistema informatizado, seja em um PC; um servidor; um supercomputador; um celular ou *smartphone* ou até mesmo um forno micro-ondas microprocessado; estamos fazendo uso de dispositivos de entrada (IN) e saída (OUT). Estes dispositivos servem para permitir que nós usuários possamos interagir com o sistema. Para que um PC tenha usabilidade (sirva para alguma coisa), precisamos de no mínimo um teclado; um mouse e um monitor; sem os quais não temos interação com a máquina. Nesta unidade vamos estudar como o sistema operacional faz o gerenciamento de I/O.

Dispositivos de I/O

Verifica-se que existe uma grande diversidade de dispositivos de entrada e saída para computadores. Diante desta variedade, podemos fazer algumas classificações:

Quanto a função

Armazenamento

Os dispositivos de armazenamento servem para armazenar arquivos – que representam dados de usuários ou programas de usuários. Nesta categoria, há uma variedade bastante grande de dispositivos, que por sua vês são encontrados em grande variedades de tipos, tamanhos, formatos. Estas características irão variar conforme a aplicação a que dispositivos se destina, no entanto seu funcionamento básico será sempre o mesmo. São exemplos de dispositivos de armazenamento:

- HD (disco rígido);
- Leitores de CDROM/DVDROM;
- Leitores de discos flexíveis;
- unidades de fita magnética;
- Cartões de memória.

Interface humana

Esta categoria de dispositivo tem por objetivo permitir que o usuário possa interagir com o sistema. Esta interação pode acontecer de diversas formas e novamente irá depender do objetivo da aplicação. São exemplos de dispositivos de interface humana:

- teclado, *mouse*, monitor;
- tela para *touch* (*touch-pad, touch-screen*);
- equipamentos de realidade virtual (luvas, óculos, etc);
- sistema de som (microfone, altofalantes);
- câmeras de vídeo.

Comunicação inter-máquinas

Esta categoria de dispositivo tem por objetivo viabilizar a comunicação ou a interação entre sistemas. Uma comunicação cliente/servidor; um sistema distribuído são exemplos de aplicações destes dispositivos. São exemplos de dispositivos de comunicação inter-máquinas:

- placa de rede (cabeada, sem fio);
- *bluetooth*;
- USB;
- WIMAX;
- Infra-vermelho;
- I2C.

Quanto a direção da comunicação

Outra classificação interessante é feita quanto ao sentido da comunicação.

Dispositivos de entrada

São dispositivos de entrada de informações em um sistema. Exemplo:

- teclado;
- *mouse*;
- microfone;
- scanner.

Dispositivos de saída

São dispositivos de saída de informações em um sistema. Exemplo:

- vídeo;
- impressora;
- alto-falantes.

Dispositivos de entrada e saída

São dispositivos que permitem tanto a entrada quanto a saída de informações em um sistema. Exemplo:

- tela *touch-screen*
- placa de rede.

Quanto ao tipo de comunicação

Dentro de um computador, todas as informações (comandos ou dados) são binárias – e são expressas em bytes (1 byte é um conjunto de 8 bits). Tipicamente, computadores modernos trabalham com larguras de barramento de 64bits ou 32bits.

Nossos dispositivos também podem ser classificados (separados) quanto a forma com que estes bits são transportados entre o dispositivo e o computador. Os dois principais tipos de comunicação de dispositivos com o computador são:

Paralelo

Neste tipo de comunicação os bits são enviados em paralelo (simultaneamente) por um caminho chamado barramento. Todas os bits saem e chegam ao mesmo tempo no no device.

- Maior interferência e ruídos;
- Curtas distâncias
- Baixas frequências;

São exemplos de dispositivos paralelos:

- Impressora com interface paralela;
- HD's PATA (antigos)
- HD's SCSI (antigos)
- Controladora de vídeo PCI, AGP;

Serial

Neste tipo de comunicação os dados da comunicação são postos em uma "fila" e enviados um após o outro em um único fio (meio de comunicação);

- Menor ruído e interferências (varia conforme a distância);
- Maiores distâncias;
- Maiores frequências;

São exemplos de dispositivos seriais:

- Impressora com interface serial ou USB;
- HD's SATA (atuais)
- HD's SAS (atuais)
- Controladoras de vídeo PCI-X

Atenção:

Atualmente, muitos dispositivos estão sendo migrados para comunicação serial porque este padrão permite um aumento da frequência da comunicação com menor taxa de interferência. Como consequência temos um melhor desempenho.

Interfaceamento

Um dispositivo, por qualquer que seja, nunca se conecta diretamente ao barramento do computador – sempre é necessário um hardware intermediário que faz o controle interno do dispositivo e cria uma interface de acesso com o hardware. A figura I.1 procura representar isto.

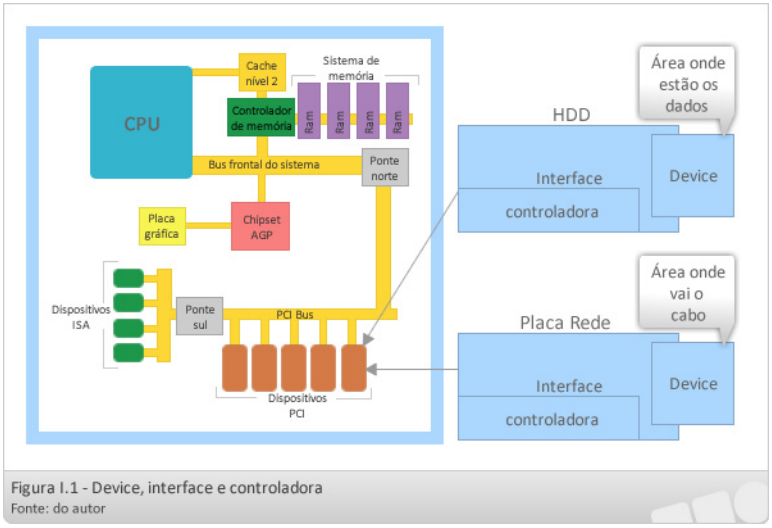

Figura I.1 - Device, interface e controladora
Fonte: do autor

Verifica-se na figura I.1 a representação hipotética de um computador com 2 *devices*, sendo um HD e uma Placa de rede. O *device* não está ligado diretamente ao computador; a ligação é feita através de uma interface, que faz a mediação entre o dispositivo e o computador.

Controladora

Toda interface precisa e uma controladora dedicada. Esta controladora é um processador que o fabricante do *device* projetou para que seja feito o controle da interface e do próprio *device*. A controladora conhece a arquitetura interna e os mecanismos de funcionamento do *device*.

A controladora apresenta ao sistema um conjunto de operações genéricas para manipulação dos dados do *device*; estas operações genéricas são solicitadas à controladora do *device* pelo sistema operacional. A controladora "traduz" o pedido do sistema operacional para ações de controle do *device*. São exemplos de operações genéricas:

- "ler dados" - informa ao device que se quer ler algum dado dele;
- "escrever dados" - informa ao device que se quer gravar algum dado nele;
- "ler status" - para verificar como está a situação do device. Pode estar ocupado, aguardando, hibernando ou algum outra situação específica do tipo de device;
- "reinicializar" - para que o device reinicialize seu mecanismo e controles internos.

Endereçamento do dispositivo

Um problema importante é saber como o processador endereça ou "encontra" o dispositivo entre diversos outros. Este mecanismo depende basicamente da arquitetura do hardware em uso.

O mapeada em espaço de memória

Neste mecanismo, típico da arquitetura CISC, os dispositivos de I/O são endereçados como "se fossem" um endereço de memória. Basicamente o espaço de endereços de memória é dividido e até um endereço é memória e dali pra diante é I/O – o sistema operacional sabe o que está acessando pelo endereço usado. A figura I.2 procura mostrar este esquema de endereçamento.

Device	Address range (hexadecimal)	Size
RAM	0000 - 7FFF	32 KiB
General purpose I/O	8000 - 80FF	256 bytes
Sound controller	9000 - 90FF	256 bytes
Video controller RAM	A000 - A7FF	2 KiB

Figura I.2 - I/O mapeada em memória
Fonte: do autor

No exemplo hipotético:

- O range de 0000 até 7FFF é memória, permite uma quantidade de 32 KB;
- O range 9000 até 90FF é endereçamento da controladora de som;

I/O mapeada em espaço de I/O

Neste mecanismo, típico da arquitetura RISC, os dispositivos de I/O são endereçados diretamente, ou seja, o espaço de endereçamento de I/O é completamente dissociado do espaço de endereçamento da memória. A figura I.3 procura mostrar este esquema.

Device	Address range (hexadecimal)	Size
RAM	0000 - 7FFF	32 KiB
General purpose I/O	0000 - 00FF	256 bytes
Sound controller	0100 - 01FF	256 bytes
Video controller RAM	0200 - 09FF	2 KiB

Figura I.3 - I/O com mapeamento próprio
Fonte: do autor

No exemplo hipotético

- O range de 0000 até 7FFF é memória, permite uma quantidade de 32 KB;
- O range de 0000 até 00FF é endereçamento da controladora de som;

Comunicação com o dispositivo

Como se dá a comunicação dos dados entre o dispositivo e o processador. As 3 formas são:

I/O programada

- Neste mecanismo, não há uso de chamada de sistema;
- O controle de entrada e saída é responsabilidade do programador, o que aumenta a complexidade do programa escrito;
- Processo acessa a I/O mas não sai do processador (muita perda de tempo);
    - Atualmente os sistemas operacionais não permitem mais este tipo de comunicação.

Interrupções

- Neste mecanismo, já há uso de chamada de sistema;
- O processos pede I/O para o sistema operacional, que envia para o device;
- Quando device tem a resposta, envia uma IRQ (requisição de interrupção) para o processador, que para o que está fazendo e atende o device, levando seus resultados para a área de memória do processo solicitante;
    - requer mecanismo de controle de IRQ (presente em todos os computadores atuais).

DMA

- Direct Memory Accesss
- Neste mecanismo, já há uso de chamada de sistema;
- O processos pede I/O para o sistema operacional, que envia para o device;
- Quando device tem a resposta, entrega direto na área de memória do processo solicitante;
- A grande vantagem é a economia de tempo do processador (que não se preocupa em fazer o transito de informações entre I/O e processo;
    - requer circuito controlador de DMA.

Subsistema de I/O

Os sistemas de I/O costumam ter algum nível de complexidade, isto ainda é associado a grande diversidade de dispositivos de I/O no mercado; diversos fabricantes; diversos modelos; diversos tipos. Como o sistema operacional controla tudo isto ?

O subsistema de I/O é um módulo do sistema operacional responsável por este controle. Os principais objetivos deste subsistema são:

- reduzir o número de rotinas de acesso (para simplificar);
- permitir a inclusão de um novo device no sistema operacional sem ter que modificar a interface para o usuário final;
- padronizar o acesso aos devices.

O subsistema de I/O é organizado (separado) em 4 camadas fundamentais, cada qual com uma funcionalidade. Cada camada "entrega" à camada superior uma abstração mais simples para acessar a camada inferior; quanto mais alta a camada mais simples ou padronizado é mecanismo de acesso (a abstração). Aqui por abstração, entende-se o nível de complexidade necessário para acessar o dispositivo.

A figura I.4 procura representar este subsistema.

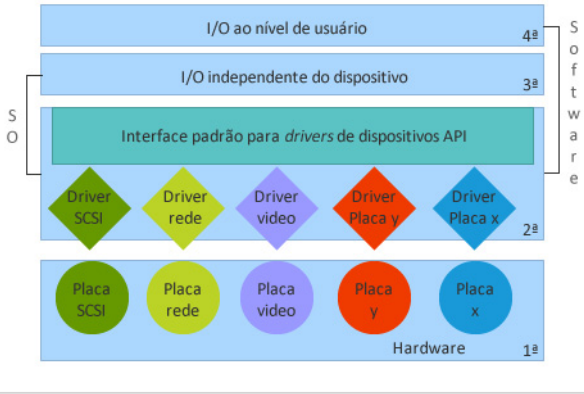

Figura I.4 - Camadas do subsistema de I/O
Fonte: do autor

Primeira camada

Esta camada representa o hardware propriamente dito – todos os dispositivos com suas respectivas interfaces e controladoras. Neste nível, toda atividade acontece em nível de comandos eletromecânicos (no caso do HD e leitores de CD/DVD) e eletrônicos.

Segunda camada

Cada device presente na camada de hardware possui seu conjunto próprio de operações e modo de funcionamento. Nesta camada o sistema operacional faz sua interface com o hardware, para cada hardware presente deve haver um driver fazendo a "tradução" dos comandos do sistema operacional para o hardware.

A camada 2 fornece a camada 3 uma visão uniformizada dos dispositivos presentes no computador. Aqui, fica clara a afirmação de Tanenbaum, feita na unidade A, "o sistema operacional esconde do programador o hardware ... e também oculta muitas coisas desagradáveis relacionadas com interrupções, temporizadores, gerenciamento de memória e outros recursos de baixo nível".

Device Driver

O *device driver* é um módulo de software fornecido pelo fabricante do dispositivo que é "anexado" ao sistema operacional e permite que estes se "conversem". O device driver é um componente obrigatório para todos os dispositivos presentes. Nos sistemas operacionais atuais, o device driver costuma ser automaticamente carregado na instalação do dispositivo – caso isto não aconteça, o dispositivo não funcionará e será necessário fazer a instalação manualmente.

O driver é sempre dependente do sistema operacional, ou seja, um driver compilado para o sistema operacional GNU/Linux não será compatível com o sistema operacional Microsoft Windows[1]

1. Marcar registrada da Microsoft Corporation

Terceira camada

A terceira camada implementa um conjunto de funções comuns a todos os dispositivos – aqui não temos mais a diversidade das camadas 1 e 2. Os principais procedimentos implementados são:

Escalonamento de I/O:

- controla o acesso ao dispositivo;
    - busca organizar o acesso do dispositivo – principalmente quando há concorrência pelo recurso;
- algoritmos escalonadores análogos ao escalonador do processador;

Denominação:

- controla a identificação dos dispositivos, para que o sistema operacional possa fazer referência a este ou aquele dispositivo.
    - Em Linux, os nomes dos dispositivos podem ser vistos na pasta /dev[2]
    - O HD SATA por exemplo:
        - /dev/sda ⟶ identifica o primeiro HD Sata do sistema
        - /dev/sdb ⟶ identifica o segundo HD Sata do sistema
        - /dev/sda1 ⟶ identifica a primeira partição do primeiro HD Sata
        - /dev/sda2 ⟶ identifica a segunda partição do primeiro HD Sata

Bufferização:

- implementa um mecanismo de memória intermediária para fazer o ajuste entre a quantidade de informação que se quer acessar/gravar em um determinado dispositivo;

Cache de dados:

- mecanismo para armazenar os dados mais acessados de um dispositivo, para otimizar o desempenho;
- alguns sistemas operacionais implementam cache de disco com prefetching, onde o objetivo é antecipar a ação do usuário – carregar neste cache os próximos arquivos que serão abertos pelo usuário.

Alocação/liberação:

- mecanismo que gerencia a reserva e a liberação de um recurso;
- impressora por exemplo pode ser usada por usuário por vez – para controlar isto, implementa-se um mecanismo de spool para "enfileirar" as requisições dos usuários.

Direitos de Acesso:

- controle de permissões de acesso para dispositivos;
    - podemos permitir/negar a utilização da impressora para este ou aquele usuário.

Tratamento de erros:

- alguns erros podem ser tratados diretamente nesta camada, para diminuir a quantidade de erros ocorridos na próxima camada.

Quarta camada

A quarta camada é a API que o programador faz uso para manipulação dos dispositivos. Aqui tem-se acesso às funções que irão controlar os dispositivos do computador.

Síntese

- Dispositivos de I/O são responsáveis interação entre usuário ↔ maquina e máquina ↔máquina.
- Os dispositivos de i/o podem ser classificados de diversas formas:

  função
  - armazenamento, interação homem-máquina, interação máquina-máquina;

  direção na comunicação
  - entrada, saída e entrada/saída;

  tipo de comunicação
  - serial / paralelo
- todo dispositivo precisa de uma interface para trabalhar
  - esta interface é um circuito que fica entre o barramento do computador e o dispositivo propriamente dito.
  - Toda interface é controlada por um processador dedicado ao dispositivo;
- os dispositivos de I/O podem ser endereçados diretamente ou com o se fossem uma área de memória;
- os dispositivos de I/O podem transmitir seus dados de 3 formas
  - I/O programada em que o programador faz todo o controle da transmissão dos dados (método não mais utilizado porque desperdiça tempo de processamento);
  - Interrupções - o dispositivo envia uma interrupção ao processador avisando que os dados estão disponíveis; o processador para o que está fazendo e envia os dados do device para o processo;
  - DMA – o dispositivo de I/O entrega seus dados diretamente na memória do processo solicitante.
- Para gerenciar os dispositivos o sistema operacional tem um “modulo” chamado subsistema de I/O, que é dividido em 4 camadas distintas; cada camada tem por função “simplificar” o acesso ao dispositivos . Este sistema oculta do usuário a verdade sobre o hardware do computador.
  - Primeira camada – representa o hardware propriamente dito;
  - Segunda camada – representa os drivers de dispositivo que são módulos de software que fazem a interface entre o device e o sistema operacional.
  - Terceira camada – representa uma interface de uniformização e regulamentação no acesso dos dispositivos, o oferecendo à camada superior uma abstração mas acessível;
  - Quarta camada – representa a API de programação do sistema operacional para o acesso aos dispositivos; possui a abstração mais alta

2. Dev – acrônimo para devices

Atividades

1. Na máquina virtual que configuramos no início da disciplina, instalamos o sistema operacional GNU/Linux 2.6.32 (distribuição Debian Lenny). Este kernel reconhece automaticamente muitos dispositivos de hardware.

a) listar os drivers carregados com o comando #lsmod;

b) manipulação de drivers em linux

- carregar o driver tun (um driver para tunelamento - criar tuneis de rede) - para carregar o driver # modprobe tun
- liste novamente os drivers carregado e localize o driver tun
- remova o driver #rmmod tun
- liste novamente os drivers carregado e localize o driver tun