Módulo 8

Sistema Operacional

Apêndice A

Exercícios

Versão 1.0 - Mar/2008

Autor
-

Equipe
Rommel Feria
John Paul Petines

Necessidades para os Exercícios

Sistemas Operacionais Suportados

NetBeans IDE 5.5 para os seguintes sistemas operacionais:

- Microsoft Windows XP Profissional SP2 ou superior
- Mac OS X 10.4.5 ou superior
- Red Hat Fedora Core 3
- Solaris™ 10 Operating System (SPARC® e x86/x64 Platform Edition)

NetBeans Enterprise Pack, poderá ser executado nas seguintes plataformas:

- Microsoft Windows 2000 Profissional SP4
- Solaris™ 8 OS (SPARC e x86/x64 Platform Edition) e Solaris 9 OS (SPARC e x86/x64 Platform Edition)
- Várias outras distribuições Linux

Configuração Mínima de Hardware

Nota: IDE NetBeans com resolução de tela em 1024x768 pixel

Sistema Operacional	Processador	Memória	HD Livre
Microsoft Windows	500 MHz Intel Pentium III workstation ou equivalente	512 MB	850 MB
Linux	500 MHz Intel Pentium III workstation ou equivalente	512 MB	450 MB
Solaris OS (SPARC)	UltraSPARC II 450 MHz	512 MB	450 MB
Solaris OS (x86/x64 Platform Edition)	AMD Opteron 100 Série 1.8 GHz	512 MB	450 MB
Mac OS X	PowerPC G4	512 MB	450 MB

Configuração Recomendada de Hardware

Sistema Operacional	Processador	Memória	HD Livre
Microsoft Windows	1.4 GHz Intel Pentium III workstation ou equivalente	1 GB	1 GB
Linux	1.4 GHz Intel Pentium III workstation ou equivalente	1 GB	850 MB
Solaris OS (SPARC)	UltraSPARC IIIi 1 GHz	1 GB	850 MB
Solaris OS (x86/x64 Platform Edition)	AMD Opteron 100 Series 1.8 GHz	1 GB	850 MB
Mac OS X	PowerPC G5	1 GB	850 MB

Requerimentos de Software

NetBeans Enterprise Pack 5.5 executando sobre Java 2 Platform Standard Edition Development Kit 5.0 ou superior (JDK 5.0, versão 1.5.0_01 ou superior), contemplando a Java Runtime Environment, ferramentas de desenvolvimento para compilar, depurar, e executar aplicações escritas em linguagem Java. Sun Java System Application Server Platform Edition 9.

- Para **Solaris**, **Windows**, e **Linux**, os arquivos da JDK podem ser obtidos para sua plataforma em http://java.sun.com/j2se/1.5.0/download.html
- Para **Mac OS X**, Java 2 Plataform Standard Edition (J2SE) 5.0 Release 4, pode ser obtida diretamente da Apple's Developer Connection, no endereço: http://developer.apple.com/java (é necessário registrar o download da JDK).

Para mais informações: http://www.netbeans.org/community/releases/55/relnotes.html

Colaboradores que auxiliaram no processo de tradução e revisão

Aécio Júnior
Alberto Ivo da Costa Vieira
Alexandre Mori
Alexis da Rocha Silva
Allan Wojcik da Silva
Antonio José Rodrigues Alves Ramos
Angelo de Oliveira
Bruno da Silva Bonfim
Carlos Fernandes Gonçalves
Denis Mitsuo Nakasaki
Felipe Gaúcho
Jacqueline Susann Barbosa
João Vianney Barrozo Costa
Luiz Fernandes de Oliveira Junior
Marco Aurélio Martins Bessa
Maria Carolina Ferreira da Silva
Massimiliano Giroldi
Paulo Oliveira Sampaio Reis
Ronie Dotzlaw
Seire Pareja
Thiago Magela Rodrigues Dias
Vinícius Gadis Ribeiro

Auxiliadores especiais

Revisão Geral do texto para os seguintes Países:

- **Brasil** – Tiago Flach
- **Guiné Bissau** – Alfredo Cá, Bunene Sisse e Buon Olossato Quebi – ONG Asas de Socorro

Coordenação do DFJUG

- **Daniel deOliveira** – JUGLeader responsável pelos acordos de parcerias
- **Luci Campos -** Idealizadora do DFJUG responsável pelo apoio social
- **Fernando Anselmo -** Coordenador responsável pelo processo de tradução e revisão, disponibilização dos materiais e inserção de novos módulos
- **Rodrigo Nunes -** Coordenador responsável pela parte multimídia
- **Sérgio Gomes Veloso -** Coordenador responsável pelo ambiente JEDI™ (Moodle)

Agradecimento Especial

John Paul Petines – Criador da Iniciativa JEDI™
Rommel Feria – Criador da Iniciativa JEDI™

Original desta por – McDougall e Mauro – Solaris Internals. Sun Microsystems. 2007.

1. Objetivos

Neste apêndice veremos exercícios que podem ser realizados para reforçar o conhecimento sobre cada lição deste módulo.

Não será apresentada a resolução dos mesmos, cabe ao estudante tirar suas próprias conclusões.

2. Exercícios

2.1. Lição 1 – Introdução ao Solaris

1. Abrir a lista de processos no seu sistema operacional atual (*Ctrl-Alt-Del* no **WindowsXP**, **ps aux** no terminal em sistemas **Unix**) e observe os diferentes processos que estão sendo executados no sistema. Pesquise cada um desses processos. Quais processos são parte do sistema operacional, e quais não são? Por que?
2. Além dos objetivos aqui discutidos para um sistema operacional, que outros objetivos deveriam ser incluídos?
3. Sistemas Operacionais, tais como, Linux, MacOS, Windows e Solaris são conhecidos. Entretanto, existem muitos outros sistemas operacionais. Pesquise e faça uma lista de pelo menos outros cinco sistemas e uma breve descrição de cada.
4. Uma técnica chamada de **virtualização** permite um sistema operacional rodar como uma aplicação sobre um outro sistema operacional. Leia mais sobre este assunto e o descreva com suas palavras.

2.2. Lição 2 - Instalação

1. Considerar o efeito da Internet no seu cotidiano. Como deveriam evoluir os sistemas operacionais considerando este fato?
2. Que outras aplicações um sistema robusto demandaria? E um sistema distribuído?

2.3. Lição 3 – Comandos Básicos e Scripting

1. Criar um *script* para copiar um arquivo. Deverá mostrar uma mensagem de erro caso o arquivo não exista. Por exemplo:

```
$ ./politecp /etc/passwd /export/home/alice
Deseja que eu copie o arquivo /etc/passwd para /export/home/alice
Cópia finalizada! Tenha um bom dia.
```

2. Criar um *script* para aceitar um número informado pelo usuário e verificar se este é maior ou menor do que um determinado número.

```
Entre o número [0-100]: 53
Muito grande
Entre o número [0-100]: 23
Muito pequeno
Entre o número [0-100]: 42
Parabéns, você acertou em 3 tentativas!
```

3. Criar um *script* para listar todo o conteúdo do diretório **/usr/sbin** em ordem decrescente de tamanho e direcionar a saída para um determinado arquivo.
4. Criar um *script* para modificar a lista de controle de acesso de um arquivo por meio de um conjunto de perguntas. Exemplo:

```
$ ./mychangemod test.txt
Você quer que este arquivo possa ser lido pelo usuário [s/n]? s
Você quer que este arquivo possa ser gravado pelo usuário [s/n]? s
Você quer que este arquivo possa ser executado pelo usuário [s/n]? n
Você quer que este arquivo possa ser lido pelo grupo [s/n]? s
Você quer que este arquivo possa ser gravado pelo grupo [s/n]? s
Você quer que este arquivo possa ser executado pelo grupo [s/n]? n
Você quer que este arquivo possa ser lido por outros [s/n]? s
Você quer que este arquivo possa ser gravado por outros [s/n]? n
Você quer que este arquivo possa ser executado por outros [s/n]? n
Rodando chmod 664 test.txt
Ok
```

5. Criar um programa Java que use **System.out** e **System.err**. Use os comandos de redirecionamento para mostrar qual saída é a regular e qual é a de erro.

2.4. Lição 4 – Processo no Solaris

1. Dados os seguintes processos e seus tempos de de CPU, todos chegando ao tempo 0.
   - P1 – 5 segundos
   - P2 – 15 segundos
   - P3 – 9 segundos
   - P4 – 6 segundos
   - P5 – 2 segundos

   Apresente a tabela de execução de processos para esses processos, segundo o esquema de escalamento abaixo:

   a) Primeiro a chegar e primeiro servido

   b) Trabalho mais curto

   c) Cíclico com Quantum de Tempo de 2

2. Bob, que é um barbeiro que corta cabelos de acordo com um esquema de prioridades, com preempção, mas sem envelhecimento, recebeu os seguintes clientes ao abrir a barbearia às 8 da manhã:

Nome	Tempo de Corte	Prioridade	Hora de Chegada
Alfred	50 min.	Média	8:00
Eugene	5 min.	Muito alta	8:30
Vincent	10 min.	Média	8:45
Dennis	5 min.	Baixa	9:00
Jerico	25 min.	Alta	9:10

   Apresente o Gráfico de Execução de Cortes de Cabelo de Bob, tempo de espera e tempo de troca por cliente, e Utilização do Barbeiro (utilização de CPU) dados os seguintes fatos adicionais:
   - Divida a execução de Cortes de Cabelo em fatias de 5 minutos cada
   - Se tiver que escolher entre clientes de mesma prioridade, escolher o que chegou primeiro
   - Tempo de mudança de contexto de **Bob**: 5 min. A mudança de contexto acontece sempre que **Bob** muda de cliente, independente de ter ou não terminado o corte do cliente
   - No início de cada fatia de 5 minutos, **Bob** escolhe a próxima pessoa para cortar o cabelo, ou seja, a decisão de cortar o cabelo de alguém é tomada antes que a mudança de contexto ocorra
   - Se o seu cabelo não está sendo cortado então você está esperando, incluir este tempo de Mudança de Contexto

3. **Charlie**, parceiro de **Bob**, que faz alocação de tempo por processo cíclico com intervalo de tempo de 20 minutos e 5 minutos de tempo de mudança de contexto, recebeu os seguintes clientes um dia, na seguinte ordem, às 8 da manhã:

Nome	Tempo de Corte

Steve	50 min.
Bert	5 min.
John	10 min.
Mark	5 min.
Henry	25 min.

4. Apresente o Gráfico de Execução de Cortes de Cabelo de Charlie, tempo de espera e tempo de troca por cliente, e Utilização do Barbeiro.

2.5. Lição 5 – Java Thread

1. Há três crianças querendo usar um balanço. Contudo, somente uma delas pode usar o balanço por vez. Cada criança alterna entre ter vontade de se balançar ou fazer uma pausa para recuperar o fôlego, enquanto outra criança usa o balanço. Implemente isto com o uso de *threads*. Faça com que cada thread de criança tenha um nome associado via construtor (i.e., Kid k1 = new Kid("alice")). Use um retardo aleatório de no máximo 10 segundos para usar o balanço e para tomar fôlego. A saída deverá ser como a mostrada abaixo:

```
Alice: Eu quero usar o balanço
Alice: Eu uso o balanço por 5940 mili-segundos
Bob: Eu quero usar o balanço
Charlie: Eu quero usar o balanço
Alice: Eu me cansei. Pausa de 4430 mili-segundos
Bob: Eu uso o balanço por 4361 mili-segundos
Alice: Eu quero usar o balanço
Bob: Eu me cansei. Pausa de 6940 mili-segundos
Charlie: Eu uso o balanço por 4301 mili-segundos
...
```

2. Crie duas threads, MyInput e MyOutput. MyInput é uma thread que aceita nomes de usuários. As entradas deste usuário são impressas por MyOutput. (Note que MyOutput tem que esperar que você digite seus comandos antes de você poder imprimi-los). MyInput e MyOutput fazem isto sucessivamente até você digitar fim.

```
MyOutput: Aguardando entrada do usuário
MyInput: Por favor entre um nome: Alice
MyOutput: Olá Alice!
MyOutput: Aguardando entrada do usuário
MyInput: Por favor entre um nome: Bob
MyOutput: Olá Bob!
MyOuput: Aguardando entrada do usuário
MyInput: Por favor entre um nome: fim
MyInput terminando...
MyOutput terminando...
```

3. Alice e Bob estão no escritório. Há uma porta pela qual eles têm que passar de vez em quando. Como Bob é bem educado, se ele chega até à porta, ele se assegura de sempre abrir a porta e esperar que Alice passe pela porta antes dele. Alice, por si só, simplemente passa pela porta. Sua tarefa é escrever uma thread Alice e Bob que imite este comportamento. Para simular ter que passar por aquela porta, inclua um retardo aleatório em cada *thread*, de no máximo 5 segundos. A saída do seu código deve ser como se segue:

```
Alice: Eu passo pela porta
Alice: Eu passo pela porta
Bob: Eu chego até a porta e espero por Alice...
Alice: Eu passo pela porta
Bob: Eu sigo a Alice
Alice: Eu passo pela porta
Bob: Eu chego até a porta e espero por Alice...
...
```

4. Modifique o problema Produtor-Consumidor para considerar o cenário de *buffer* cheio pela limitação do tamanho do vetor em 10 e fazendo o Consumidor mais lento do que o Produtor.
5. Implemente uma solução para o problema Leitores-Escritores através da criação de 2 *threads* de Leitor e 2 *threads* de Escritor e um objeto compartilhado MyFile com os métodos read() e write(). Novamente, muitas *threads* poderiam estar lendo o arquivo, mas somente uma das *threads* poderá estar escrevendo no arquivo. Para simular os retardos de "leitura" e "escrita", use Thread.sleep() fazendo com que cada thread pause por no máximo 10 segundos (retardo aleatório) durante a leitura e a escrita. Sua saída deverá se parecer com a seguinte:

```
Reader1: Eu quero ler o arquivo
Reader1: Lendo arquivo por 1420 mili-segundos
Reader2: Eu quero ler o arquivo
Reader2: Lendo arquivo por 4504 mili-segundos
Writer1: Eu quero escrever no arquivo
Writer2: Eu quero escrever no arquivo
Reader1: Leitura do arquivo terminada
Reader2: Leitura do arquivo terminada
Writer1: Escrevendo no arquivo por 5930 mili-segundos
Reader1: Eu quero ler o arquivo
Reader2: Eu quero ler o arquivo
Writer1: Escrita no arquivo terminada
Writer2: Escrevendo no arquivo por 6041 mili-segundos
Writer2: Escrita no arquivo terminada
Reader1: Lendo arquivo por 9543 mili-segundos
Reader2: Lendo arquivo por 3461 mili-segundos
...
```

2.6. Lição 6 – Observabilidade

1. Há uma ponte de faixa única perto da sua escola. Um dia você encontrou ao acaso o seguinte cenário, envolvendo 6 carros.

a) Qual é o Gráfico de Alocação de Recursos para esta ponte? (Dica: A ponte tem o comprimento exato de seis recursos)

b) A situação foi resolvida empurrando dois carros para fora da ponte. Que esquema de recuperação de *Deadlock* é este?

c) Dê dois esquemas para prevenir esta situação de ocorrer novamente no futuro. Indique que condição necessária é eliminada.

2. Você ganhou uma quantidade expressiva de dinheiro numa loteria e decidiu aplicar em um banco. Seu critério primário na escolha do banco é se há dinheiro suficiente para satisfazer todas as suas necessidades.

Nome	**Empréstimo Atual**	**Teto do Empréstimo**
Alice	$ 5000	$ 7000
Verde-2	$ 0	$ 9000
Azul-3	$ 3000	$ 15000
Amarelo-4	$ 1000	$ 20000
Rosa-5	$ 1000	$ 2000

Quantia disponível: $ 3000

2.7. Lição 7 – ZFS

1. Crie um nome de usuário **douglas** com o diretório *home* em /export/home/douglas. Agora, faça deste diretório um zpool espelhado usando dois arquivos, pool1 e pool2, localizados no diretório raiz (*root*). Além disso, faça com que este *pool* tenha uma cota de disco de 1GB.

2.8. Lição 8 – Zonas no Solaris

1. Prepare uma zona cujo diretório raiz esteja armazenado no diretório /mysecondzone na zona global. Faça com que esta zona monte um diretório especial chamado /shared tanto no diretório raiz da zona global quanto na zona local. Demonstre (depois da instalação) que quaisquer mudanças feitas na zona global são também vistas na zona local.

Parceiros que tornaram JEDI™ possível

Instituto CTS
Patrocinador do DFJUG.

Sun Microsystems
Fornecimento de servidor de dados para o armazenamento dos vídeo-aulas.

Java Research and Development Center da Universidade das Filipinas
Criador da Iniciativa JEDI™.

DFJUG
Detentor dos direitos do JEDI™ nos países de língua portuguesa.

Politec
Suporte e apoio financeiro e logístico a todo o processo.

Instituto Gaudium
Fornecimento da sua infra-estrutura de hardware de seus servidores para que os milhares de alunos possam acessar o material do curso simultaneamente.