**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**1 Prova Segunda Semestre 2013 manha**

Questões:

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**1 Prova Segunda Semestre 2013 noite**

Questões:

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Prova primeiro semestre 2013 noite**

**1) o que é thread e processo e principal diferença.**

**2) Explicar transparência e dar exemplo.**

**3) Explica espera ocupada**

**4) O que é instruções de alta atomicidade**

**5) Swap (nao sabia o q era)**

**6) mostrar ausência de inanição no algoritmo de dekker**

**7) Explicar aquele algoritmo do choosing**

**8) explicar condição de corrida**

**Prova Final Manhã 2 semestre 2012**

- Uma questão de RMI

- Uma questão de Messageria

- Duas questões de codificação

- Uma questão socket

- Uma questão deadlock

- Uma questão Semáforo e Monitor

**Prova Final Noite 2 semestre 2012**

**1. Na programacao concorrente, precisamos resolver os problemas de exclusão mútua e sincronização condicional. Explique estes conceitos e indique como podem ser obtidos com semáforos e monitores.**

**2. Explique o conceito de deadlock e mostre duas maneiras de evitá-los (lembre-se das estratégias utilizadas no problema do jantar dos filósofos)**

**3. Indique situações onde seria preferivel utilizar protocolo UDP no lugar de TCP, e vice-versa. Justifique sua resposta.**

**4. Porque a interface remota é importante na implementação de uma invocação remota de metodos? Em JavaRMI, como essa interface é construída?**

**5. Qual a diferença entre utilizacao de filas(Queue) e tópicos (Topic) na comunicação via JMS?**

**6. Um processo P1 executa as seguintes instruções (em sequência):**

- instução-1a

- instrução-1b

Um processo P2 executa as seguintes instruções (em sequencia)

- instução-2a

- instrução-2b

Utilizando semáforos mostre como é possível sincronizar estes processos de modo a garantir que instrucao-1a sempre seja executada antes de instrucao-2b, e que instrucao-2b sempre seja executada antes de instrucao-1b

**2º chamada da 1º Prova Manhã/ Noite**

- Questão Decker

- Questão Petterson

- Questão Swap

- Questão para falar sobre Tolerância a falhas no Sist.Distribuido

- Questão para falar sobre Modelo de Hardware e Modelo de Software no Sist. Distribuido

- Questão Transparência Requisição e Persistência

- Questão falando sobre desabilitar interrupção para exclusão mútua.

**Primeira Prova DSD2 - 2º semestre 2012 - manhã**

**1. Explicar o principal meio de troca de informação em sistemas distribuídos**

Uma aplicação consiste em diversos processos que se comunicam através de trocas de mensagens (ou chamadas remotas de métodos) e cada processo consiste de múltiplas threads que se comunicam através de mensagens enviadas pela rede.

**2. Explicar transparência de localização e de replicação**

Transparência de localização: oculta o lugar de instalação de um recurso.

Transparência de replicação: oculta o fato de um recurso estar replicado (por motivos de desempenho

ou de tolerância a falhas).

**3. Explicar a consequencia da ausencia de memoria compartilhada em sistemas distribuidos**

A ausência desse tipo de recurso nos sistemas distribuídos trás as seguintes consequências:

- Não existe um relogio global. A inexistencia desse relogio leva a uma “assincronia” entre os diversos

processadores.

- É dificil observar as propriedades globais do sistema (tais como terminação ou deadlock).

- É impossivel conhecer o estado global do sistema (definido como um conjunto de estados locais que

ocorrem simultaneamente).

**4. Explicar como sistemas distribuidos garantem tolerância a falhas**

As falhas dos diversos componentes do sistema devem ser mascaradas. Isto em geral é obtido por redundância, permitindo que um outro componente assuma as funções do faltoso.

**5. Explicar pq se inverter 2 comandos no algoritmo de peterson ele fica errado**

Ele deixa de garantir exclusão mútua.

Teríamos mudança no valor de wantCS[0] de false para true, deixando o algoritmo incorreto.

Duas threads conseguirá entrar na região crítica ao mesmo tempo.

**6. Mostrar que o algoritmo de dekker garante Exclusão Mútua**

O Algoritmo de Dekker resolve o problema da exclusão mútua, pois o algoritmo utiliza uma abordagem baseada em “espera ativa”, na qual um processo, que não pode acessar a região crítica, fica em ciclo de testes com variáveis compartilhadas.

**7. Explicar o algoritmo de baker e alterar para q mais de uma thread entre na CS**

**Explicação resumida:**

Inicializa as estruturas de trabalho. requestCS(), estagio 1: Ti indica sua chegada a porta.

requestCS(), estagio 1: Ti procura o maior numero de senha. requestCS(), estagio 1: Ti faz number[i] = 1

+ maior numero encontrado. requestCS(), estagio 1: Ti indica saida da porta. requestCS(), estagio 2: Ti

compara senha com (todo) Tj . requestCS(), estagio 2: espera se Tj esta na porta. requestCS(), estagio 2:

espera se number[i] > number[j] (e Tj interessado na CS). releaseCS(), Ti indica liberação da CS.

**Explicação no detalhe:**

- O construtor inicializa as estruturas de trabalho (o numero de threads e os vetores choosing[] e number

[]).

- Metodo requestCS(): uma thread Ti executando este metodo deve passar pelos dois estagios descritos:

1. O primeiro estagio representa a porta:

- Ti indica sua chegada a porta (choosing[ i ] = true).

- Ti percorre o vetor number[] procurando pelo maior elemento (primeiro laco for()).

- Ti ajusta sua senha (number[i]) para ser uma unidade maior que a maior encontrada.

- Ti indica sua sada da porta.

Observação: se duas (ou mais) threads estiverem simultaneamente na porta, ambas podem terminar com números de senha idênticos.

2. O segundo estagio representa a comparac~ao de senhas (segundo laço for()). Para cada thread Tj , Ti age da seguinte forma:

- Aguarda ate que Tj n~ao esteja na porta (primeiro laco while()).

- Aguarda (segundo laco while()) ate que Tj n~ao esteja interessada na região critica (number [j] == 0) ou que sua senha seja menor que a de Tj (number[i] < number[j]). Empates são resolvidos pelos identicadores dos processos.

Observação: apos vericar estas condic~oes para todos as threads, Ti pode entrar na regi~ao crtica.

- Metodo releaseCS(): Ti simplemente manifesta seu desinteresse na regi~ao crtica (number[i] = 0).

**Primeira Prova Dsd2 - 2 semestre 2012 - noite**

1 explicar o principal meio de troca de informação em sistemas paralelos

2 explicar transparencia de acesso e de replicação

3 explicar a consequencia da ausencia de memoria compartilhada em sistemas distribuidos

4 explicar como sistemas distribuidos garantem escalabilidade

5- explicar pq se inverter 2 comandos no algoritmo de peterson ele fica errado

6 - mostrar que o algoritmo de dekker garante EM

7- explicar o algoritmo de baker e alterar para q k threads entre na CS

**Prova Especial Dsd2 Noite 1 Semestre 2012**

**1) Algo referente à exclusão mútua - atomicidade alta - swap**

**2) Fazer semáforo ou monitor - Jarro de biscoitos. Irmã + nova só pode comer um biscoito qdo a irmã + velha comeu 2 biscoitos. A mãe tem que ser avisada quando jarro esta vazio.**

**3) Fazer semáforo ou monitor -Saldo conta. Deposito e Saque. A pessoa não pode ter conta negativa. O valor do saque tem que ser menor que o valor do saldo da conta. Dá um problema sobre saque de dinheiro na conta.**

**4) definir sistema paralelo e distribuido**

**5) Explicar semaforo em sistemas concorrentes**

**6) Explicar monitor em sistema concorrente**

**7) Explicar “invocação remota de métodos”.**

**Prova Especial Dsd2 Manhã 1 Semestre 2012**

**1 - definir sistema paralelo e distribuido**

**2- pq o semaforo fornece exclusao mutua usando test and set**

**3- explicar semaforo em sistemas concorrentes**

**4- explicar monitor em sistema concorrente**

**5- ???**

**6- fazer programa semaforo-monitor. qual?**

**7) fazer programa semaforo-monitor. qual?**

**Prova DSD2 Noite Final 1 Semestre 2012**

1)**explique troca de mensagem em JMS usando diferentes tipos de comunicacao ponto a ponto e publisher**

Mensageria por tópico um publicador publica uma mensagen e todos os assinantes recebem a mensagem publicada

Mensageria por fila o publicador publica uma mensagem e o primeiro da fila recebe essa mensagem e retira da fila

**Porque os metodos p e v do semaforo devem ser atomicos?**

As operações de incrementar e decrementar devem ser operações atômicas, ou indivisíveis, ou seja, enquanto um processo estiver executando uma dessas duas operações, nenhum outro processo pode executar outra operação sob o mesmo semáforo, devendo esperar que o primeiro processo encerre a sua operação sob o semáforo. Essa obrigação evita condições de disputa entre vários processos.

**RMI sobre sockets TCP para desenvolvimento de aplicacoes distribuidas**

Para isso deve se usar um socket orientado a conexão pois ele fará a conexão com o outro lado e sera acessado usando uma interface de fluxo: open-read-write-close.

**Numa empresa com um servidor critico novo que precisa estar funcioando sempre,e um velho que para de vez em quando. Melhor jeito de implementar um meio de comunicação entre eles (UDP, TCP, RMI, MESSAGERIA)?**

A melhor opção seria o RMI pois é um sistema destribuido e caso o sevidor antigo caia, o novo daria conta do recado, já que o RIM é tolerante á falha.

**O q e' serializacao utilizada na RMI**

Serialização é o fato de transformar um objeto em bytes e e transmiti-lo de forma serial por um stream que fara a transmissão atravéz de uma origem e um destino.

quando você serializa um Objeto, seus dados na memória ficam armazenados em sequência, assim podendo transportá-lo pela rede.

**Qual o papel do servidor de nomes (rmiregistry) em java RMI?**

Antes dos clientes poderem invocar um método de um objeto remoto, deve-se primeiro obter uma referência para este objeto. Tipicamente, o cliente obtém uma referência a um objeto remoto como o valor de retorno de uma chamada de método.

**- fazer algoritmo contador semaforo binario**

**Prova Dsd2 Manhã Final 1 semestre 2012**

**1)papel stub e skeleton**

Responsaveis pela transparencia de localização em RMI

O stub representa o servidor no lado cliente e o skeleton representa o lado cliente no servidor,

o stub empacota a mensagem faz o marshaling e envia para o skeleton que por sua vez faz o demarshaling chama o metodo necessário e recebe o valor retornado, logo após o skeleton fa o marshaling e transmite para o stub que faz o demarshaling e repassa para o cliente.

**2)papel do jndi na jms**

A JNDI é um serviço de nomes de diretorios, e tem o papel de obter recursos atrvéz dos nomes, para isso ela utiliza o metodo lookup.

**3)EXPLIQUE A principal CARACTERISTICA DA TROCA DE MENSAGENS EM JMS, USANDO DIFERENTES TIPOS DE COMUNICACAO(PONTO A PONTO E PUBLISHER/SUBSCRIBer**

Mensageria por tópico um publicador publica uma mensagen e todos os assinantes recebem a mensagem publicada

Mensageria por fila o publicador publica uma mensagem e o primeiro da fila recebe essa mensagem e retira da fila

**-p() e v() semáforo. algo com exclusao mutua**

**Fazer um semaforo ou um monitor de uma creche. criancas 3 para 1 monitora. algo assim.**

**Fazer "comparativo" de rmi com o socket tcp**

**7)e a outra é uma questao q vc tem que responder com qual protocolo melhor resolveria o problema dado e justificar.**

Protocolo UDP é bom para stream de video ou audio coisas desse tipo pois nem todos os pacotes chegam, nesse caso só algumas partes do video ou audio se perderiam não causando danos maiores.

Já o protocolo TCP é mais indicado quando for transmitir arquivos de texto pois se você perder uma certa quantidade de pacotes pode interferir no resultado final do arquivo.

se eu lembrar melhor as questões volto aqui e comento!!!

bons estudos e boa sorte galera!!! =)

Se alguem souber as questões da turma da noite e as provas segunda chamada....

Parace que as questões da primeira prova(2012) turma manhã são essas:

* Diferenca entre paralelismo x sistemas distribuídos
* Principais vantagens
* Espera ocupada
* Diferenca entre thread stop e thread interrupt
* 3 códigos:

1 - Se o programa atendia o requisito de exclusao mútua (se o código estivesse errado era para corrigir)

2 - Mesma coisa do primeiro, porém de outra forma.

3 - Um código para explicar o que ele fazia

alguem atualiza com as ultimas provas da turma da manha e noite??

acredito que na especial ele deve inverter as provas, como fez em dsd1

Primeira Prova manha 22/09/2011

**1) Explique os conceitos de condicao de corrida, regiao critica e exclusao mutua**

**Condição de corrida [goe06, p. 20]**

Condição que ocorre quando o resultado de uma computação depende do sincronismo das instruçoes de

multiplas threads que estejam executando a computação.

**Região critica**

Seção de codigo que acessa estado compartilhado. Tambem chamada de seção critica.

**Exclusão mutua**

Ato de assegurar que duas ou mais threads não se encontrem executando, simultaneamente, as respectivas

regioes criticas.

**2) Defina espera ocupada, indique para que é utilizada nos algoritmos de sincronizacao e qual seu principal problema.**

**3) Uma variavel inteira x tem valor inicial 0. Dois diferentes processos concorrentes tentam acessar esta variavel, um deles para incrementa-la  e outro para decrementa-la.Quais os possiveis valores finais de x? justifique sua resposta.**

**4) Explique como funciona o mecanismo de interrupcao das threads(metodo Thread.interrupt()).**

5) **A listagem abaixo mostra um algoritmo (incorreto) para obtencao de exclusao mutua. Mostre que é capaz de fornecer exclusao mutua, indique qual seu erro e demostre que ele esta incorreto**

*public class Exclusao implements Lock{*

*private volatile boolean wantCS[]={false,false}*

*public void requestCS(int i){*

*wantCS[i]=true;*

*while(wantCS[1-i]);*

*}*

*public void releaseCS(int i){*

*wantCS[i]=false;*

*}*

*}*

Resposta(laper):

t0: wantcs[i]==false && wantcs[0]==true

t1: wantcs[0]==false && wantcs[1]==true

t0: faz wantcs [0]=true

t1: faz wantcs [1]=true

t0: le wantcs[1]==true e bloqueia

t1: le wantcs[0]==true e bloqueia

**6)Duas threads rodando de forma concorrente executam as seguintes instrucoes**

**Thread A       Thread B**

**instrucao a1   instrucao b1**

**instrucao a2   instrucao b2**

**Utilizando semafaros, modifique o codigo das threads para que a1 seja executada antes de b2 e b1 antes de a2.**

Resposta(Laper):

semafaro sa(0)

semafaro sb(0)

thread A      thread B

A1             B1

sb.release()  sa.release()

sa.acquire()  sb.acquire()

A2             B2

7)Uma jarra de biscoitos é compartilhada por duas irmas, Luciana e Gabriela, usando a seguinte regra: Gabriela pode pegar um biscoito apenas depois que Luciana (a mais velha) tiver pego pelo menos dois (em duas ocasioes separadas), enquanto Luciana pega um biscoito sempre que tem vontade. A mae e avisada por qualquer uma delas e sempre que elas procurarem um biscoito e a jarra estiver vazia, quando entrao ela torna a encher a jarra

Exame especial de manhã 30/06/2011

**1º o que é thread e processo e qual a principal diferença entre eles.**

Processo é a abstração de um programa em execução e cuida do gerenciamento de recursos.

Thread é uma tarefa de um mesmo processo executada concorrentemente e compartilha recursos.

**2º como é possivel obter exclusão mutua e sincronização utilizando monitores?**

A semântica dos monitores garante que um dado instante haverá apenas uma thread executando o método de entrada, garantindo exclusão mútua.

As variáveis de condição wait() e notify() garantem sincronização condicional.

Thread.interrupt

**4º Vc tem um hospital e deseja enviar as informaçoes atualizadas para o sistema sobre o monitoramento dos pacientes do hospital.qual o tipo de protocolo vc utilizaria? foi algo desse tipo.**

TCP. Pois é um protocolo orientado a conexão e confiável, que garante a entrega e ordenação das mensagens.

**Duas threads com dois processos em cada. A1, A2, B1 e B2**

**6º caiu tmb o codigo da 3º implementação incorreta de exclusão mutua pra achar o erro e demonstrar os passos do processamento das threads.**

O erro do attempt3 é ausência de inanição. Uma thread tentando entrar na região crítica, eventualmente pode conseguir.

**7º fazer um monitor contador**

int contador;

Public contador {

this.contador = contador;

}

syncronized acquire() {

while (this.contador == 0) this.wait();

}

syncronized release() {

this.contador++;

this.notify();

}

Alguem tem o que caiu na ultima prova noite 16/06/2011?

16/06/2011 - DSD2 ultima Prova Manhã

O que caiu na prova de DSD II - manhà?

**1) O que são e para que servem qual o papel(algo do tipo não lembro direito) o stub e o skeleton?**

O stub e o skeleton são responsáveis pela transparência de localização do RMI. O stub representa o servidor no lado do cliente, o skeleton representa o cliente no lado servidor

Passos da invocação remota

1. A invocação do método remoto pelo cliente, e uma invocação direta para o método

correspondente ao stub

2. O stub empacota o nome do método, junto com os argumentos, esta operação se

chama marshaling

3. O stub transmite para o skeleton

4. O skeleton recebe a mensagem e reconstrói os argumentos, isto se chama

unmarshaling

5. O skeleton faz uma invocação direta ao método do servidor e receb o valor retornado

6. O skeleton faz o empacotamento do valor retornado e o envia para o stub

7. O stub reconstrói o valor de retorno e repassa o resultado para o cliente, que ficou bloqueado durante todo o processo

**2)Descreva os protocolos UDP e TCP e cite suas diferenças.**

UDP

• Simples e eficiente

• Não utiliza conexão

• Não garante ordem e entrega de pacotes

• Pacotes de tamanho fixo

• Utiliza datagramas

• Funciona melhor numa rede confiável

TCP

• Mais confiável e menos eficiente

• Orientado a conexão

• Garante entrega e ordem de pacotes

• Pacotes trabalham com fluxo de bytes, sendo seu tamanho variável

• Utiliza Socket e Server Socket

**3) EXPLIQUE A principal CARACTERISTICA DA TROCA DE MENSAGENS EM JMS, USANDO DIFERENTES TIPOS DE COMUNICACAO(PONTO A PONTO E PUBLISHER/SUBSCRIBER**

Messageria é um midleware de mensagens, trabalha tanto com mensagens síncronas (necssario aguardar pelo recebimento da mensagem) quanto assíncronas (pode-se fazer outras atividades, enquanto aguarda o recebimento da mensagem). As mensagens possuem 3 propriedades: propriedades, atributos e corpo. Em messageria temos 2 tipos de distribuição de mensagens:

Topico: envia mensagem para todo mundo, pode –se utilizar publicadores e assinantes, em que assinantes específicos, recebem as mensagens publicadas pelos publicadores

Fila: envia mensagem para o primeiro que acessar a fila, retirando a mensagem. Pode-se utilizar seletores, para especificar para quem é a mensagem

**4) Como pode ser obtido Exclusão mutua e sincronização condicional em semaforos.**

A exclusão mutua é obtido, quando o semáforo é binário, e suas variáveis a ser modificada estao entre os métodos acquire() e release(). Já a sincronização condicional é obtida quando um outro semáforo lança release(), então ele esta liberando a outra thread para processsamento, o que faz com que esta outra thread verifique um condição verdadeira e retorne o processamento

**5) Defina Deadlock e de um exemplo de onde ocorre.**

Deadlock é a condição de bloqueio de 2 ou mais entidades que esperam pela liberação de recursos que estão a posse de demais, em uma cadeia circular. O deadlock ocorre principalmente no jantar dos filósofos, em que um filosofo fica pensando, então ele fica faminto e quer comer, para comer ele necessita de 2 garfos, que podem estar sendo utilizado por outros filósofos, ele pode pegar um garfo, e ficar aguardando pelo outro garfo, mas se isso acontecer com todos, todos ficarão esperando, o que fará com que ninguém consiga obter o garfo pra comer. Uma das formas de evitar o deadlock é fazer com que um dos filósofos inverta a ordem de pegar os garfos (da direita para a esquerda), outra forma é fazer com que os garfos sejam obtidos simultaneamente

**Ficou faltando uma questão teorica q num to conseguindo lembrar. a ultima questão era algo do tipo implementar um monitor para semaforos partindo destes conceitos aqui: P() e V() é só o que eu lembro desta questão pratica.**

**Só tinha uma questão pratica o resto tudo era teorica.**

                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       31/03/2011

Primeira Avaliação DSD2 - Noite

Não foi nessa ordem mas segue o que lembro das questões

**1) Diferença entre processo e thread:**

Processo: Abstração de um programa em execução e cuida do gerenciamento de recursos (memória, arquivos, conexões).

Thread: Tarefa de um mesmo processo que executa concorrentemente e compartilha recursos do mesmo processo, é uma entidade própria. Além disso, um único processo pode ter diversas threads, ou seja, pode executar diversas atividades simultaneamente (ou concorrentemente).

**2)O que é transparência? Dê um exemplo prático?**

**3) Diferença entre Thread.interrupt() e Thread.stop():**

   A primeira paralisa a thread e a segunda para a execução da thread.

**4) implementar algoritimo de exclusao mutua com a função swap**

5) Algoritmo Attempt 1, explicar porque nao tem exclusao mutua

Faltam duas questoes que não me lembro se alguem lembrar por favor postem no doc.

29/03/2011

Conceitos para a primeira prova.

**- Diferença entre processo e thread:**

Processo: Abstração de um programa em execução e cuida do gerenciamento de recursos (memória, arquivos, conexões).

Thread: Tarefa de um mesmo processo que executa concorrentemente e compartilha recursos do mesmo processo, é uma entidade própria. Além disso, um único processo pode ter diversas threads, ou seja, pode executar diversas atividades simultaneamente (ou concorrentemente).

**- Exclusão Mútua:**

R1: Assegurar que duas ou mais threads não se encontrem executando, simultaneamente, as respectivas regiões críticas. Exclusão mutua pode ser obtida se as regiões críticas forem executadas de forma atômica.

R2: Ex0clusão mútua é o ato de assegurar que duas ou mais threads não se encontrem executando, simultaneamente, as respectivas regiões críticas.

**- O que é transparência? Sempre é melhor buscar níveis altos de transparência?**

            Transparência é uma característica do sistema distribuído que permite ocultar dos usuários os detalhes da distribuição.

            Acesso: Ocultar as diferenças na forma de acesso ao sitema.

            Localização: Ocultar o lugar de instalação de um recurso.

            Replicação: Oculta o fato de um recurso estar replicado.

            Persistência: Oculta o fato de um recurso estar armazenado em meio volátil ou permanente.

**- O que é condição de corrida? Quando ela pode dar problema?**

            Condição que ocorre quando o resultado de uma computação depende do sincronismo das instruções de múltiplas threads que estejam executando a computação.

O erro depende do sincronismo das duas threads e pode se manifestar muito raramente.

**- Problemas attempt 1, 2 e 3 (exclusão mútua):**

            Attempt 1: Não funciona porque o testando o openDoor e setando para falso não funciona atomicamente. A variável compartilhada não grava quem setou false no openDoor. Violação exclusão mútua: Duas threads enxergam openDoor.

            Attempt 2: Ambos processos podem setar wantCS = true e entrar em loop infinito. Cavalheirismo: Duas threas cedem a vez.

            Attempt 3: Falha na condição de progresso. Se o P1 não está interessado na seção crítica, P0 então fica preso. Violação da ausência de ate: Thread nunca utiliza CS para mudar o turno.

**- Deadlock:**

            Condição de bloqueio em que duas ou mais entidades esperam pela liberação de recursos que estão de posse das demais, em uma cadeia circular.

            Ex: Jantar dos filósofos.

            Acontece o deadlock se todos os filósofos adquirem simultaneamente o garfo da direita por exemplo e assim todos ficam bloqueados esperando a liberação do garfo da esquerda que esta de posse do filosofo da esquerda, ou seja, ninguém ira mais sair dessa situação de bloqueio. Isso pode ser evitado fazendo com que um dos filósofos adquira um garfo na ordem inversa ou fazendo com que os filósofos peguem os 2 garfos simultaneamente (ATOMICAMENTE).

**- O que é atomicidade:**

            R1: Duas operações A e B são atômicas uma em relação à outra se, do ponto de vista de uma thread executando A, quando outra thread executa B, ou toda a operação de B foi executada e nenhuma parte de A foi executada.

            R2: Uma operação é atômica se é atômica em relação a todas as outras que acessam o mesmo estado (inclusive ela mesma). Uma thread nunca pode enxergar outra thread no meio da execução de sua região critica.

**- Diferença entre Thread.interrupt() e Thread.stop():**

            A primeira paralisa a thread e a segunda para a execução da thread.

**- Quais valores podem ser obtidos, quando há uma disputa de 3 processos?**

            1, 2 ou 3 pois pode ocorrer que alguma thread pode buscar o valor de x desatualizado sendo que a atribuição de um valor depende 3 instruções no processador. Para não ocorrer esse problema teríamos que certificar que a operação é atômica.

**- Defina o que é espera ocupada, como ela é usada e qual é o principal problema dela.**

            Fato de desperdiçar ciclos do processador. Pode fazer com que uma outra thread que tenha que ter acesso não entre na sessão crítica.

**- Vantagens e desvantagens de sistemas paralelos e distribuídos:**

***Distribuídos***

            Vantagens: Escalabilidade, modularidade e baixo custo.

            As diversas máquinas podem ser de tipos completamente diferentes, cada um com sua própria estrutura e arquitetura, permitindo mais independência sem ficar restrito a uma máquina apenas, arquitetura ou SO.

            Desvantagens: Utilização de Troca de mensagem pela rede o que demanda controle e esforço para os processos estarem sincronizados, ausência de Memória Compartilhada e ausência de Relógio Global.

***Paralelos***

            Vantagens: Eficiência na atualização, comunicação e na distribuição de alterações de dados compartilhados. Utilização da memória compartilhada.

            Desvantagens: Caso a máquina hospedeira do processo para de funcionar, o processamento é interrompido.

**- Problemas algoritmo da padaria:**

            Requer o trabalho de cada processo para obter o bloqueio, mesmo se não houver contenção e exige que cada processo use timestamps que são ilimitados em tamanho.

29/03/2011

Segue essas questoes que nao estoa respondidas que tem muita chance de cair!!!

**4 - Algoritmo de Dekker implementa a característica de progresso? Justifique.**

*public class Dekker implements Lock {*

*boolean querSC[]={false, false};*

*int vez = 1;*

*public void requestSC(int i) {*

*int j = 1 - i;*

*querSC[i]=true;*

*while (querSC[j]){*

*if (vez == j) {*

*querSC[i] = false;*

*while (vez == j); // busy wait*

*querSC[i] = true;*

*}*

*}*

*}*

*public void releaseSC(int i) {*

*vez = 1 - i;*

*querSC[i] = false;*

*}*

*}*

Resposta:

**5 - Explicar o Algoritmo de BackeryLock e modificar o algoritmo para**

**ele aceitar que invés de 1 thread acessar o região crítica k threads**

**acessam essa região.**

public class BakeryLock implements Lock {

    private final int N;

    private volatile boolean[] choosing; // inside doorway

    private volatile int[] number;

    public BakeryLock(int numProc) {

        N = numProc;

        choosing = new boolean[N];

        number = new int[N];

        for (int j = 0; j < N; j++) {

            choosing[j] = false;

            number[j] = 0;

        }

    }

    public void requestCS(int i) {

        // step 1: doorway: choose a number

        choosing[i] = true;

        for (int j = 0; j < N; j++)

            if (number[j] > number[i])

                number[i] = number[j];

        number[i]++;

        choosing[i] = false;

        // step 2: check if my number is the smallest

        for (int j = 0; j < N; j++) {

            while (choosing[j]); // process j in doorway

            while ((number[j] != 0) &&

                    ((number[j] < number[i]) ||

                    ((number[j] == number[i]) && j < i)))

                ; // busy wait

        }

    }

Exame Especial 20/12/10 DSD2 MANHA

**01)PROCESSO X THREADe a principal diferença**

Processo: abstração de um programa em execução e cuida do gerenciamento de recursos.

Thread: tarefas de um mesmo processo que executam concorrentemente e compartilham recursos do mesmo processo, é uma entidade própria. Alem disso, um unico processo pode ter diversas threads, ou seja, pode executar diversas atividades simultaneamente (ou concorrentemente).

**02) pq o processos d primitivas de sincronização sao melhores do que os algoritmos de exclusao mutua**

**03) Achar o erro do programa e encontrar um exemplo onde ocorre**

public class Attempt1 implements Lock {

private volatile boolean openDoor = true;

public void requestCS(int i) {

while (!openDoor); // busy wait

openDoor = false;

}

public void releaseCS(int i) {

openDoor = true;

}

}

Attempt 1 – Porta – Violação da Exclusão Mutua

- duas threds enxergam openDoor.

**04) DUAS THREADS OP1 E OP2 X=0;Y=0;(X=1 , RETUNR Y);(Y=2 , RETUNR 3\*X)**

Se elas forem executadas de forma atomica, o resultado será y=2 e x=3, se não pode se variar o resultado ocorrendo o problema de condição de corrida(resultado depende do sincronismo).

**05) SERVIDOR NOVO E VELHO (UDP, TCP, RMI, MESSAGERIA)?**

A comunicação a ser implementação nesta situação, seria por messageria, porque o cliente e o servidor necessariamente não precisam estar disponíveis ao mesmo tempo (comunicação assíncrona), ou seja, com isso o cliente pode executando outras operações.

**06) P() E V() – ATOMICIDADE**

P() corresponde ao comando acquire() do semáforo e o v() ao comando release() como eles acessam uma mesma regiao critica se eles não forem atômicos, será permitindo  outra thread acessar a cessão critica. : uma operação é atômica se é atômica em relação a todas as utras que acessam o mesmo estado (inclusive ela mesma).

**07) Elaborar em portugol ou java o seguinte problema:**

**Em uma stub cada monitor deve tomar conta de no maximo 3 criancas.**

**(Existe restricoes para entrar criancas e para monitor sair)**

Exame Especial 17/12/10 DSD2 Noite

**01)PROCESSO X THREAD**

Processo: abstração de um programa em execução e cuida do gerenciamento de recursos.

Thread: tarefas de um mesmo processo que executam concorrentemente e compartilham recursos do mesmo processo, é uma entidade própria. Alem disso, um unico processo pode ter diversas threads, ou seja, pode executar diversas atividades simultaneamente (ou concorrentemente).

**02) Qual o problema do algoritmo a seguir.**

public class PetersonLock implements Lock {

private volatile boolean wantCS[] = {false, false};

private volatile int turn = 1;

public void requestCS(int i) {

int  j = 1 - i;

wantCS[i] = true;    // coloco condicao true pra entrar na CS

turn = i;            // seto o turn p/ propria thread ao inves de outra

while (wantCS[j] && (turn == j));

}

public void releaseCS(int i) {

wantCS[i] = false;

}

}

Resposta: A thread que entrar primeiro na CS, vai sempre colocar o turn para ela mesma. assim a verificação será falso,

não haverá a condição de exclusão mútua, pois duas threads poderão entrar na CS.

**03) Exclusão Mútua e Sincronização Condicional com semáforos.**

Exclusao mutua:assegurar que duas ou mais threads não se encontrem executando, simultaneamente, as respectivas regiões críticas. Exclusão mutua pode ser obtida se as regiões críticas forem executadas de forma atômica.

Sinc. Cond.:a thread deve aguardar até que uma condição se torne verdadeira para que ela possa prosseguir na região crítica.synchronized (obj).

**04) DUAS THREADS OP1 E OP2 X=0;Y=0;(X=1 , RETUNR Y);(Y=2 , RETUNR 3\*X)**

Se elas forem executadas de forma atomica, o resultado será y=2 e x=3, senão pode se variar o resultado ocorrendo o problema de condição de corrida(resultado depende do sincronismo).

**05) SERVIDOR NOVO E VEIO (UDP, TCP, RMI, MESSAGERIA)?**

A comunicação a ser implementação nesta situação, seria por messageria, porque o cliente e o servidor necessariamente não precisam estar disponíveis ao mesmo tempo (comunicação assíncrona), ou seja, com isso o cliente pode executando outras operações.

**06) P() E V() – ATOMICIDADE**

P() corresponde ao comando acquire() do semáforo e o v() ao comando release() como eles acessam uma mesma regiao critica se eles não forem atômicos, será permitindo  outra thread acessar a cessão critica. : uma operação é atômica se é atômica em relação a todas as utras que acessam o mesmo estado (inclusive ela mesma).

**07) Elaborar em portugol ou java o seguinte problema:**

**Em uma creche cada monitor deve tomar conta de no maximo 3 criancas.**

**(Existe restricoes para entrar criancas e para monitor sair).**

DSD2 semestre 2010 - noite 07/12/2010

**1- O que são e para que servem qual o papel(algo do tipo não lembro direito) o stub e o skeleton?**

**2 - O que é RMI registry e Objetivos pra que serve?**

**Resposta:**

Conceito

  - Base de dados que converte o nome lógico para o endereço do objeto remoto.

  - Servidor – Registro o serviço ao RMI Registry.

  - Cliente – Consulta o RMI Registry para encontrar um serviço.

 Objetivos

  Fornecer transparência de localização, de modo que o cliente não precise conhecer a localização do servidor.

**3 - Explique o que é e como implementar uma interface remota?**

**Resposta:**

· Descreve os métodos para a chamada remota;

· Descreve o comportamento do serviço;

· Deve ser pública;

· Obrigatoriamente deve estender Java.rmi.Remote.

· Obrigatoriamente todos os métodos devem lançar Java.rmi.RemoteException

O papel da interface ser um contrato, é esconder a implementação de um determinado serviço. Desta forma a implementação do serviço pode ser alterada inúmeras vezes sem preocupar com as aplicações clientes, contato que mantenha a compatibilidade do serviço com a interface. Ela pode ser implementada por qualquer classe. INTERFACE tem que extender a interface REMOTE e todos os METODOS da interface do serviço devem lançar obrigatoriamente a exceção REMOTEEXCEPTION.

**4 - Qual a principal característica das mensagens em middleware e qual a sua principal vantagem.**

**5 - Descreva os protocolos UDP e TCP e cite suas diferenças.**

**6- PORQUE É NECESSARIO UTILIZAR UMA INSTRUCAO WHILE NO LUGAR DE IF, EM VARIAVEIS**

**DA CONDICAO EM JAVA?**

**resposta:**O if nao garantida a condição de saída, ele precisa verificar mais de 1 vez.Se verificar só uma, não é garantida a condição de saída do release.

**7- CONSTRUA UM ALGORITMO..... SEMAFORO CONTADO**

**R, usando apenas semáforos binários. EM JAVA OU PORTUGOL.** (sugestão do prof: usar uma variável inteira simples e dois semáforos binários)

Este codigo está no arquivo dsd2-primitivassincronizacao.pdf na página 14

import java.util.concurrent.Semaphore;

public class SemaphoreLock implements Lock {

private final Semaphore sem;

public SemaphoreLock() {

sem = new Semaphore(1);

}

public void requestCS(int pid) {

try {

2

sem.acquire();

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

public void releaseCS(int pid) {

sem.release();

}

}

TA AI GALERA...

ULTIMA PROVA de DSD2  SEMESTRE 2010 - MANHA: 07/12/10

 O

É NÓIS Q TÁ..

**1-COMO OS SEGUINTES TIPOS DE DADOS SAO PASSADOS COMO PARAMETROS EM UMA**

**INVOCACAO REMOTA EM JAVA RMI?**

**1.TIPOS PRIMITIVOS**

Passados por valor

**2.OBJETOS NAO REMOTOS**

Serializados e passados por valor

**3.OBJETOS REMOTOS**

Passados por referência. (o stub correnspondente é serializado e enviado)

**2- QUAL O PAPEL DO SERVIDOR DE NOMES (JNDI) EM JMS?**

O papel do servidor de nomes em JMS é fornecer transparência de localização.

**3- EXPLIQUE O QUE É SERIALIZAÇÃO DE OBJETOS E COMO É EMPREGADA EM JAVA RMI?**

A serialização de objetos é o processo de se transformar um objeto em um fluxo de bytes

de modo que possam ser enviados pela rede, e ser reconstituídos do outro lado.

O RMI permite que objetos sejam enviados como parâmetro em uma chamada remota de

método. Neste caso o objeto é passado por valor, e é serializado neste processo.

**4- EXPLIQUE A principal CARACTERISTICA DA TROCA DE MENSAGENS EM JMS, USANDO DIFERENTES TIPOS DE COMUNICACAO(PONTO A PONTO E PUBLISHER/SUBSCRIBER)**

**5- SUPONHA UMA APLICACAO DISTRIBUIDA EM QUE A NECESSIDADE DE COMUNICACAO EFICIENTE.**

**MENSAGENS DE TAMANHO FIXO EM REDE LOCAL muito CONFIAVEL E OS PACOTES ENVIADOS TEM TAMANHO DETERMINADO. QUAL TIPO UTILIZARIA,**

**E JUSTIFIQUE A SUA RESPOSTA.**

UDP. Porquê não há necessidade de ter certeza que há conexão estabelecida entre o servidor e o cliente (TCP), pois a rede local é MUITO CONFIÁVEL.

Com isso, pode-se enviar pacotes diretamente sem efetuar esse tipo de verificação.

-----

UDP. O protocolo UDP utiliza datagramas que são pacotes de tamanho fixo. Além disto, a comunicação através deste protocolo é ágil, pois ele não estabelece uma conexão entre o cliente e o servidor. Estabelecer uma conexão entre os dois pontos causaria um overhead na comunicação, diminuindo a velocidade de comunicação. O fato de não estabelecer uma conexão não é crítico na integridade dos dados enviados, pois a estrutura de rede onde esta comunicação acontece é confiável.

**6- PORQUE É NECESSARIO UTILIZAR UMA INSTRUCAO WHILE NO LUGAR DE IF, EM VARIAVEIS**

**DA CONDICAO EM JAVA?**

O if nao garantida a condição de saída, ele precisa verificar mais de 1 vez.Se verificar só uma, não é garantida a condição de saída do release.

-----

Monitores Java: Uma thread T0 testa a variável de condição e decide aguardar ser notificada para retomar sua execução (obj.wait()). Quando uma thread T1 notifica (obj.notify()) a thread T0, o monitor não é cedido a ela (T0). A thread T1 continua executando até o final do método. Apos o obj.notify() de T1 pode haver alguma instrução que modifica novamente o valor da variável de condição que T0 testa. Isto obriga que a T0 faça o teste novamente, ou seja, use um while no lugar de if para garantir que a condição seja satisfeita até T1 finalizar TODA sua execução.

**7- CONTRUA UM ALGORITMO..... SEMAFORO CONTADOR, usando apenas semáforos binários. EM JAVA OU PORTUGOL**. (sugestão do prof: usar uma variável inteira simples e dois semáforos binários)

// um semáforo é binário desde que tenha sempre 0 ou 1 permissão

Semaphore sem = new Semaphore(0);

Semaphore mutex = new Semaphore(1);

int count = 0;

acquire() {

mutex.acquire();

if(count == 0) {

mutex.release();

sem.acquire();

mutex.acquire

,();

}

count--;

mutex.release();

}

release() {

mutex.acquire();

if(count == 0)

sem.release();

count++;

mutex.release();

}

2ª chamada Primeira prova manha (05/10/2010)

- peguntou sobre sistemas parelelos e distribuidos e as vantagens

- sobre Deadlock e dar 2 exemplos quando ocorrem

- sobre sincronização condicional de espera ocupada. se o resultado da o mesmo? q iferença tem?

- sobre o qual problema no attempt 2

- O q acontece se tirar o choosing o algoritimo da padaria

- implementar algoritimo de exclusao mutua com a função swap

- implementar semaforo ou monitor do refeitor

**Primeira prova noite (28/09/2010)**

**- peguntou sobre sistemas parelelos e distribuidos e as vantagens**

**- sobre Deadlock e dar 2 exemplos quando ocorrem**

**- sobre o qual problema no attempt 2**

**- Aquele do algoritmo de Peterson**

**- A de implementeção acho que foi igual da prova da manhã**

**Primeira prova manhã (28/09/2010)**

1. O que é transparencia? sempre é melhor buscar niveis altos de transparencia?

  Transparencia é uma caracteristica do sistema distribuido que permite ocultar dos usuários os detalhes da distribuição.

1. O que é condição de corrida?  quando ela pode dar problema?

Condição que ocorre quando o resultado de uma computação depende do sincronismo das instruções de múltiplas threads que estejam executando a computação.

O erro depende do sincronismo das duas threads e pode se manifestar muito raramente.

1. O que é exclusão mútua e como conseguir ela com semáforo e monitor

Exclusão mútua é o ato de assegurar que duas ou mais threads não se encontrem executando, simultaneamente, as respectivas regiões críticas.

Semáforo

Controla o número de threads que podem acessar um recurso ou executar uma ação simultaneamente.

Monitores

São primitivas de sincronização de nível mais alto que os semáforos. São basicamente constituídos por uma estrutura que encapsula os dados compartilhados pelas diversas threads.

1. Colocou o código do Attemp 3 e perguntou a falha dele ( a falha era condicao de progresso)
2. Colocou o código do algorítimo que tem o turn e perguntou o que acontece se trocar a linha do WantCs com a do Turn
3. Duas threads com dois processos em cada. A1, A2, B1 e B2. Quais as possibilidades de ordem de execução? Mostrar como fazer pra A1 ser executada antes de B2 e A2 ser executada antes de B1 usando semáforo.
4. Implementar um monitor contador, com dois métodos increment() e decrement(). O contador não pode ficar com menos que o valor mínimo e nem mais que o valor máximo. (essa é a única que realmente tem que implementar código)

**Segunda Prova - manhã 21/05/2010**

**1)**      **Definir como que um cliente usa o RMI e como é implementada.**

Rafael: Forma de invocar metodos de objetos remotos como se estes estivessem na mesma maquina.

O Objetivo do RMI é tornar os principios de programação orientada a objetos disponiveis para o desenvolvimento de sistemas distribuidos. RMI pode ser encarada tambem como uma forma

de alto nivel para enviar e receber mensagens.

Os passos para implementar uma RMI são:

1-Definir a interface remota

2-Implementar o serviço

3-Desenvolver o servidor

4-Desenvolver o cliente

5-Iniciar o RMI registry e o servidor

6-Rodar o cliente

\_\_\_\_

**2)**      **Defina este conceitos: Exclusão mutua, sincronização condicional e explique como é utilizado em monitores.**

Exclusão mutua: Ato de assegurar que duas ou mais threads não se encontrem execuntando, simultaneamente, nas respectivas regioes crticas.

Pode ser obtida se as regiões criticas forem executadas de forma atômica.

**sincronização condicional=** um processo atrasa  sua execução até  que  uma condição torne-se verdadeira

\_\_\_\_\_

**3)**      **Definir e justificar qual dos métodos estudados você usaria para implementar um sistema de comunicação de fichas de pacientes em um hospital.**

Alexandre:

Acho que ta parecida com uma resposta que um cara deu na outra prova:

Partindo do principio que a rede em que irão trafegar estas informações não é confiável, utilizaria o protocolo TCP, pois este é um protocolo confiável e orientado a conexão.

Isto significa que, para trocar mensagens, os dois extremos da comunicação devem estabelecer uma conexão entre eles; isto garante que todas as mensagens serão entregues, e na mesma ordem em que foram enviadas.

Outra diferença em relação ao protocolo UDP é que os dados das mensagens aparecem como um fluxo de bytes contínuo, sem pacotes de tamanho fixo.

\_\_\_\_\_\_

**4)**      **Defina Deadlock e de um exemplo de onde ocorre.**

Rafael:

Deadlock: Condição de bloqueio em que duas ou mais entidades esperam pela liberação de recursos que estão de posse das demais, em uma cadeia circular.

Exemplo:  Jantar dos Filosofos.

Acontece o deadlock se todos os filosofos adquirem simultaneamente o

garfo da direita por exemplo e assim todos ficam bloqueados esperando a liberação do garfo

da esquerda que esta de posse do filosofo da esquerda, ou seja, ninguem ira mais sair dessa

situação de bloqueio.

Isso pode ser evitado fazendo com que um dos filosofos adquira um garfo na ordem inversa

ou fazendo com que os filosofos peguem os 2 garfos simultaneamente(ATOMICAMENTE).

\_\_\_\_

**5)**      **Ele deu um caso de alguns canibais muito parecido com o barbeiro dorminhoco, pediu para mostrar como era feito a implementação  de um jeito onde tenha sincronização.**

int porçoes=0

sem panelacheia=0;

sem panelavazia=0;

sem mutex=1

cosinheiro:

       while(true){

               panelavazia.alquire();

               colocarporçãonapanela(M);

               panelacheia.release()

}

selvagem:

while(true){

       mutex.alquire();

       if (porçoes==0) {

       panelavazia.release()

       penelacheia.alquire()

       porçoes=m

}

 porçoes--

 pegarporçãodapanela()

 mutex.release()

 comer()

}

**Segunda Prova - Antiga**

**1) pq as operacoes P() e V() do semaforo precisam ser atomicas**

Duas operações atômicas P() (obtém acesso a um recurso) e V() (libera acesso). Se não for atômico P() permitira a thread entrar na seção critica e decrementará o contador e v() ira incrementá-lo novamente permitindo  outra thread acessar a cessão critica.

**2) como fazer exclusao mutua e sincronizacao condicional com monitores**

Apenas uma thread pode estar executando qualquer método de

entrada em determinado momento.

 Método wait() faz uma thread bloquear na fila da variável de

condição.

Método notify() retira uma das threads da fila.

wait() e notify() devem ser executados por threads ativas no

monitor.

**3) com o algoritmo dos jantares dos filosofos, indicar o pq ocorre deadlock**

**Condição de bloqueio em que duas ou mais entidades esperam pela**

**liberação de recursos que estão de posse das demais, em uma cadeia circular.**

A solução mostrada apresenta um sério problema: suponha que todos os filósofos adquirem simultaneamente a posse do garfo à sua esquerda e bloqueiam esperando pelo garfo à direita (que está de posse do filósofo à direita!). Esta condição de espera circular é conhecida como deadlock e é um dos principais perigos da programação concorrente quando diversas threads competem pelo acesso a múltiplos recursos.

Neste exemplo, o deadlock é causado pela simetria da solução apresentada: todos os filósofos adquirem os recursos na mesma ordem (garfo direito após o garfo esquerdo). Isto pode ser evitado fazendo com que um deles adquira os garfos em ordem inversa (direita/esquerda); a implementação édeixada como exercício.

Outra estratégia possível é fazer com que um filósofo sempre adquira ambos os garfos simultaneamente.

Uma estratégia freqüentemente adotada para evitar deadlocks é exigir que locks para todos os recursos compartilhados sejam obtidos obedecendo sempre uma determinada ordem. Deadlocks serão tratados com profundidade em aulas futuras.

**4) uma situação parecida com o problema de leitor-escritor é só entender esse algoritmo**

**O problema leitor-escritor**

- Múltiplos leitores e escritores acessando uma base de dados:

- Regras:

Vários leitores podem acessar a base simultaneamente.

Ausência de conflitos de escrita-escrita: um escritor não pode

acessar a base juntamente com outro escritor.

Ausência de conflitos de leitura-escrita: um escritor não pode

acessar a base juntamente com um leitor

**5) qual a diferenca entre UDP e TCP**

UDP User Datagram Protocol | características:

- Simples e eficiente.

- Não fornece garantias de entrega e ordenação de mensagens.

- Não utiliza conexões.

- Trabalha com datagramas (pacotes de tam

anho conhecido).

- Soquete Java: DatagramSocket.

TCP Transmission Control Protocol | características:

\_ Orientado a conexão.

\_ Oferece garantia de entrega e ordenação.

\_ Trabalha com fluxos de bytes (sem início e fim definidos).

\_ Mais confiável e menos eficiente.

\_ Soquetes Java: Socket e ServerSocket.

**1a Prova Noite - 26/03/2010!!! (de manha foi assim tbm)**

**1. o que é atomicidade**

Resposta1 Vinicius Horta: Duas operações A e B são atômicas uma em relação à outra se, do ponto de vista de uma thread executando A, quando outra thread executa B, ou toda a operação de B foi executada e nenhuma parte de A foi executada.

Resposta 2 Rafael : Definição Operação Atomica: Uma operação é atômica se é atômica em relação a todas as outras que acessam o mesmo estado (inclusive ela mesma).

(Uma thread nunca pode enxergar outra thread no meio da execucao de sua regiao critica.) Pag. 17 do PDF Exlusão Mutua

**2. diferença entre Thread.interrupt() e Thread.stop()**

public class Lock implements Lock

**Resposata 2 Rafael :**

Thread.interrupt()

Thread.stop() interrompe bruscamente a execução da Thread.

**3. quais valores podem ser obtidos, quando há uma disputa de 3 processos-> isto é aquela parada de load, increment e store da variavel**

**Se três threads executarem a instrução x = x+1; quais os possiveis resultados? Justifique**

tem que fazer um chines dessa parada ai pq tres processos numa mesma variavel da um monte de combinacao de valor possivel

Griffo: ACHO QUE: O valor final da variável X será sempre 3, porém ela pode ser assim...(1,2,3 ou 2,1,3 ou 3,2,1, ou 3,1,2, ou 2,3,1 ou 1,3,2)

Rafael Definição: O problema iustrado denomina-se Condição de corrida – Condição que ocorre quando um resultado de uma computação depende do sincronismo das instruções de múltiplas threads que estejam executando a computação.

Conforme disse o griffo, no final do processamento das 3 threads o valor de x será 3.

Vinicius: Possiveis resultados: 1, 2 ou 3 pois pode ocorrer que alguma thread pode buscar o valor de x desatualizado sendo que a atribuição de um valor depende 3 instruções no processador. Para não ocorrer esse problema teriamos que certificar que a operação é atomica.

Antonio: concordo com o vinicius (acima);

**4. , o que aconteceria se tirasse a variavel chossing algoritmo BAKERYLOCKERdo programa**

**5. escrever um programa com exclusao mutua e função swap**

Osso!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Attempt 1 – Porta – Violação da Exclusão Mutua

- duas threds enxergam openDoor.

Attempt2 – Cavalheirismo Violação do requisito de progresso

– Duas threds cedem a vez.

Attempt3 - Turnos alternados Violação de ausência de inanição

- Thread nunca utiliza CS para mudar turn.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Primeira prova semestre passado **(2 semestre de 2009)**

**1 - Explique o que é transparência no contexto de sistemas**

**distribuídos e dê um exemplo de transparência.**

**Resposta:** TRANSPARÊNCIA É OCULTAR DO USUARIO OS DETALHES DE DISTRIBUIÇÃO DO SISTEMA.

EXISTE 4 TIPOS:

ACESSO: OCULTAR AS DIFERENCAS NA FORMA DE ACESSO AO SISTEMA.

LOCALIZACAO: OCULTAR O LUGAR DE INSTALACAO DE UM RECURSO.

REPLICACAO: OCULTA O FATO DE UM RECURSO ESTAR REPLICADO.

PERSISTENCIA: OCULTA O FATO DE UM RECURSO DE ESTAR ARMAZENADO EM MEIO VOLATIL E OU PERMANETE

**2 - Defina o que é Se 2 threads executarem a instrução x = 0, como ela é usada e qual é o**

**principal problema dela.**

Resposta: Fato de desperdiçar ciclos do processador.

Principal problema: Pode fazer com que uma outra thread que tenha que ter acesso nao entre na sessão critica.

**3 - O que é processo, o que é thread e defina a diferença entre as duas.**

**Resposta:**

PROCESSO: ABSTRAÇAO DE UM PROGRAMA EM EXECUÇAO

- GERENCIA RECURSOS (MEMORIA, ARQUIVOS, CONEXOES)

THRED: TAREFA DE UM MESMO PROCESSO QUE EXECUTA CONCORRENTEMENTE, COMPARTILHANDO TODOS OS RECURSOS DO PROCESSO.

- EXECUTAM CONCORRENTEMENTE EM SEU PROPRIO CONTEXTO LOCAL

- COMPARTILHAM RECURSOS DO PROCESSO

**4 - Algoritmo de Dekker implementa a característica de progresso? Justifique.**

public class Dekker implements Lock {

    boolean querSC[]={false, false};

    int vez = 1;

    public void requestSC(int i) {

        int j = 1 - i;

        querSC[i]=true;

        while (querSC[j]){

            if (vez == j) {

                querSC[i] = false;

                while (vez == j); // busy wait

                querSC[i] = true;

            }

        }

    }

    public void releaseSC(int i) {

        vez = 1 - i;

        querSC[i] = false;

    }

}

**Resposta:**

**5 - Explicar o Algoritmo de BackeryLock e modificar o algoritmo para**

**ele aceitar que invés de 1 thread acessar o região crítica k threads**

**acessam essa região.**

public class BakeryLock implements Lock {

    private final int N;

    private volatile boolean[] choosing; // inside doorway

    private volatile int[] number;

    public BakeryLock(int numProc) {

        N = numProc;

        choosing = new boolean[N];

        number = new int[N];

        for (int j = 0; j < N; j++) {

            choosing[j] = false;

            number[j] = 0;

        }

    }

    public void requestCS(int i) {

        // step 1: doorway: choose a number

        choosing[i] = true;

        for (int j = 0; j < N; j++)

            if (number[j] > number[i])

                number[i] = number[j];

        number[i]++;

        choosing[i] = false;

        // step 2: check if my number is the smallest

        for (int j = 0; j < N; j++) {

            while (choosing[j]); // process j in doorway

            while ((number[j] != 0) &&

                    ((number[j] < number[i]) ||

                    ((number[j] == number[i]) && j < i)))

                ; // busy wait

        }

    }

    public void releaseCS(int i) { // exit protocol

        number[i] = 0;

    }

}

**Resposta:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Rafael: Questoes Prova Final Manha!!!!!

**DSD 2 - Especial -Questões do livro texto - GARG**

**Questão)** Vantagens e desvantagens do sistemas paralelos e distribuidos.

Resposta:

Vantagens Distribuidos

    Escalabilidade, modularidade, confiabilidade e baixo custo

    Os diversos nodos ( máquinas hospedeiras de processos) podem ser de tipos completamentes diferentes, cada um com sua própria estrutura

    e arquitetura e com seu próprio sistema operacional, permitindo dessa forma mais inndependência e não ficar restrito a uma máquina, arquitetura

    ou sistema operacional

Desvantagem do Distribuído

    Utilização de Troca de mensagem pela rede o que demanda controle e esforço para os processos estarem sincronizados

    Ausência de Memória Compartilhada

    Ausência de Relógio Global

Vantagens Paralelos

    Eficiência na atualização de dados compartilhados

    Eficiência na comunicação e distribuição de alterações nos dados compartilhados.

    Utilização de Memória compartilhada

Desvantagem Paralelos

    Caso a máquina hospedeira do processo para de funcionar, o processamento é interrompido

**Questão) Vantagens e desvantagens de mensagens síncronas e assíncronas em aplicações distribuidas.**

Resposta: Sincronismo (necessita de relógio global, não pode-se obter sincronismo em sistemas dispersos)

Assincronismo (não existe tempo de transmissão de mensagens, impossibilita diferenciar atrasos de falhas) TIME-STAMP. todas as mensagens sao marcadas.

**Questão) Lista de todas as diferenças entre uma chamada de método local e uma invocação de método remoto.**

Resposta: A latência de uma chamada local é bastante menor que de uma chamada remota

Numa chamada local, o método é executado apenas uma vez.

As chamadas remotas têm um tempo de resposta maior do que as chamadas locais

passagem de parametro em chamadas locais: tipo primitivo=VALOR; outro tipo(objeto)=referencia

passagem de parametro em chamadas remotas: tipo primitivo=VALOR; tipo não remoto(serializado)= referencia; tipo remoto(stub é serializado)=referencia

confere se ta certo.

As chamadas remotas são mais vulneráveis e possuem falhas mais complexas

Invocação remota transparência de localização

As invocações de métodos remotos são iguais as invocações aos métodos locais, porém as diferenças entre os objetos remotos e locais devem estar expressos nas suas interfaces

**Questão)** Como é que vai fornecer semáforos em um ambiente distribuído?

Resposta:

**Questão) Dê vantagens e desvantagens do uso de RMI sobre sockets TCP para o desenvolvimento de aplicações distribuídas.**

Resposta: Mudanças feitas no objeto do lado do servidor não são refletidas do lado do cliente

Faz com que os objetos continuem em execução contínua a espera de chamadas de clientes remotos.

Confiabilidade

Ordenação de entrega das mensagens

TCP é necessário abrir e fechar socketes no RMI apenas chamada ao método remoto

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Syncronized: Esse modificador faz com que um objeto que está sendo acessado só possa ser acessado novamente quando a tarefa que está sendo realizada sobre ele seja concluida.

Processos assíncronos se comunicam pela da troca de mensagens através da comunicação entre os processos

Exclusão Mútua

Evitam que um recurso em comum seja utilizado simultaneamente, como variáveis globais, por pedaços de código chamados seções críticas.

Exemplos destes recursos são flags, contadores e queues, usados para comunicação entre códigos que são executados concorrentes, como uma aplicação e seus tratadores de interrupção

tcps clássicos permitem deadlocks, no qual um processo recebe um semáforo, outro processo recebe o outro e ambos permanecem no estado de bloqueado esperando que o outro

semáforo seja liberado.

Efeitos colaterais: Starvation no qual um processo nunca consegue obter recursos necessários para sua execução e inversão de prioridade no qual uma thread de maior prioridade espera por uma de menor prioridade.

Um semáforo é uma estrutura de dados que consiste em um número inteiro e em uma fila que armazena descritores de tarefas. O conceito de semáforos consiste na colocação de

proteções em torno do código que acessa esta estrutura para oferecer acesso limitado aos dados. Em geral, a estrutura de dados é uma fila, funcionando em regime de primeiro a

entrar, primeiro a sair. O semáforo tem uma variável especial protegida (ou tipo abstrato de dados) que tem como função o controle de acesso a recursos compartilhados (por exemplo,

um espaço de armazenamento) num ambiente multitarefa.

Há um problema em semáforos, o deadlock. Este ocorre quando um processo altera o valor do semáforo e entra na região crítica, mas ocorre uma falha na saída da região porque ele

foi danificado ou morto.

Sincronização: Objetivo é manter a consistência dos dados.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Questões semestre passado**

**Primeira Prova**

**1) Diferença entre sistema distribuido e paralelo. Cite vantagens e desvantagens**

Distribuidos: Composto por múltiplos processadores conectados por uma rede de comunicação, e que se comunicam uns com os outros atraves de mensagens envidas na rede

     Vantagens: Escalabilidade, Modularidade e Heterogeneidade, compartilhamento de Dados e Recursos, Estrutura Geografica, Baixo Custo.

Paralelo: Composto por múltiplos processadores concorrentes, que se comunicam uns com os outros utilizando memória compartilhada

     Vantagens: Eficiência na atualização de dados compartilhados, Eficiência na comunicação e distribuição de alterações de dados compartilhados, Eficiência na obtenção de paralelismo de grau fino.

**2) O que é thread e processo**

Processo: Abstração de um programa em execução.

Thread:  Tarefas de um mesmo processo que executam concorrentemente, compartilhando todos os recursos do processo.

3) estude a Attempt 3 das notas de aula

4) estude algoritmo de Peterson

5) estude algoritmo da padaria de Lamport

--------------------------------------------------------------------------------------------------------------------------

**1--criar um monitor contador (principalmente essa)**

acho que poderia ser isso aqui!!!!!

class However {

int contador;

Public However(contador) {

this.contador = contador;

}

syncronized P() {

while (this.contador <= 0) this.wait();

this.contador--;

}

syncronized V() {

this.contador++;

this.notify();

}}

**ULTIMA PROVA!!!**

**Outra Prova - 3**

**Q1) Papel da INTERFACE e como exatamente é implementada.**

O papel da interface é ser um contrato, contém apenas a declaração dos métodos.

Possui apenas o cabeçalho (assinatura) dos métodos sem nenhuma implementação.

Deve ser pública, estender Java.rmi.remote. e os métodos lançarem remote.exception.

**Q2) Principal característica de MESSAGERIA e vantagem.**

Comunicação realizada de forma assíncrona (o remetente da mensagem não fica bloqueado e não terá necessidade de ficar aguardando a resposta), as mensagens podem ser mantidas no servidor sendo entregues assim que o servidor estiver disponível. Suporte à comunicação síncrona e assíncrona, suporte a transações no envio e recebimento de mensagens, mensagens armazenadas em fila, grau de tolerância é muito alto.

**Q3) Era sobre monitores, não lembro exato como era, mas se não me engano, ele queria saber como os monitores garantem exclusão mutua e sincronização.**

Sincronização condicional: cada variável x possui uma fila de threads bloqueadas e duas operações, podendo ser executadas por threads dentro do monitor

x.wait(): thread executora é bloqueada e colocada na fila de x.

x.notify(): se a fila não estiver vazia, uma das threads é escolhida e colocada em estado de pronta para processar.

**Q4) O que é RMI REGISTRY e objetivos**

Base de dados que converte nomes lógicos para o endereço do objeto remoto, servidor registra e cliente consulta.

Objetivos RMI: Fornecer transparência de localização, de modo que o cliente não precise conhecer a localização do servidor.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ULTIMA PROVA!!!**

**1--vantagem e desvantagem do RMI - OK**

Vantagens RMI.

1.) Transparência de localização

2.) Modelagem fácil e natural de sistemas distribuídos

3.) sintaxe da chamada de método local = sintaxe da chamada de método remoto

4.) Acesso a objetos remotos

5.) Protocolo cliente servidor

6.) API de alto nível

7.) Transparente

Desvantagens de RMI.

1.) Não há garantia da entrega dos pacotes

2.) Não há ordenação no envio dos pacotes

3.) Objeto remoto e servidor de nomes (Registry) devem estar na mesma máquina

4.) Comunicação síncrona, o cliente não pode fazer outra tarefa enquanto espera a resposta do servidor

5.) limitação para usuários de browser proxy server

**2--o q é serialização e como pode ser feita em RMI e messageria**

Serialização: conversão do objeto em bytes, sendo seus parâmetros passados como valor.

-- Alguem ajuda a completar esta questão!

Messageria: suponho que a serialização em messageria seria o conceito de tópicos, quando as mensagens são entregues numa fila e os assinantes recebem a cópia das mensagens, mesmo não estando disponíveis no momento.

Para que seja possível serializar um objeto é necessário que a classe do objeto a serializar implemente a interface java.io.Serializable.

**3--Situação hipotética: c ta numa empresa com um servidor critico novo e**

**um não tão crítico, véio, que para de vez em quando quase sempre. Melhor**

**jeito de implementar um meio de comunicação entre eles (UDP, TCP, RMI, MESSAGERIA)? - OK**

(eu e mais uns lá colocamos messageria, pois pelo fato de haver um servidor

controlando a comunicação, o fato de o véio cair, não vai interromper o novo)

Resposta semestre passado:

A comunicação a ser implementação nesta situação, seria por MESSAGERIA, porque o cliente e o servidor necessariamente não precisam estar disponíveis ao mesmo tempo (comunicação assíncrona), ou seja, com isso o cliente pode executando outras operações.

**4--Ponto a ponto publisher/subscribe(o que é como funciona)**

**Comunicação ponto a ponto:** permite que um programa envie mensagens para uma fila específica que será lida por outro.

A resposta de **publisher/subscribe** é o conceito de tópico, onde refere-se ao conceito de assinatura. Cada assinatura recebe uma cópia da mensagem, o que faz com que a mesma possa ser distribuída por todos os clientes interessados, distribuída através de broadcast. Implementa um tipo de comunicação publisher/subscriber.

Wikipedia (publisher/subscriber): Este modelo suporta a publicação de mensagens para um determinado [tópico](http://pt.wikipedia.org/wiki/T%C3%B3pico_%28mensagens%29) de mensagens (*message topic*). O(s) "assinante(s)" (*subscriber*) podem registrar interesse em receber ("em assinar") mensagens de um tópico. Neste modelo, nem o "publicador" (*publisher*) ou o "assinante" sabem um do outro.

Wikipedia (ponto-a-ponto): No modelo ponto a ponto, ou por filas, um "produtor" (*producer*) envia mensagens para uma fila e um "consumidor" (*consumer*) as lê. Neste caso, o produtor conhece o destino da mensagem e a envia diretamente para a fila do consumidor

**5--passagem de parametros(quais sao os tipos) - OK**

R:

- Tipos primitivos são passados por valor.

- Objetos não remotos são passados por valor utilizando serialização.

- Objetos remotos são passados por referencia (o stub correspondente é serializado e enviado).

**6--criar um monitor contador**

Ninguem tem a manha de fazer essa questão não???

acho que poderia ser isso aqui!!!!!

class However {

int contador;

Public However(contador) {

this.contador = contador;

}

syncronized void acquire(){

while (this.contador <= 0) this.wait();

this.contador--;

}

syncronized void release() {

this.contador++;

this.notify();

}}

(seria isso ai?)

-- semáforo contador ---- ( É MONITOR QUE ELE QUER)

import java.util.concurrent.Semaphore;

public class SemaphoreLock implements Lock {

private final Semaphore sem;

public SemaphoreLock() {

sem = new Semaphore(1);

}

public void requestCS(int pid) {

try {

sem.acquire();

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

public void releaseCS(int pid) {

sem.release();

}

}

**Fontes:**

http://paginas.fe.up.pt/~eol/AIAD/aulas/JINIdocs/rmi2.html

http://pt.wikipedia.org

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Questões semestre passado

**ULTIMA PROVA!!!**

**Q1) Papel da INTERFACE e como exatamente é implementada.**

O papel da interface é ser um contrato, é esconder a implementação de um determinado  serviço. Desta forma a implementação do serviço pode ser alterada inúmeras vezes sem preocupar com as aplicações clientes, contanto que mantenha a compatibilidade do serviço com a interface.

O primeiro passo na criação de um serviço RMI é criar uma descrição desse Serviço de modo que os clientes possam saber como invocá-lo. Esta descrição baseia-se em um importante princípio: a definição do comportamento (em Java, a entidade para esta definição chama-se **interface**) e sua implementação devem ser feitas separadamente.

Na interface temos apenas o cabeçalho dos métodos sem nenhum tipo de implementação, estabelecendo um contrato; já na classe que define (implementa) o serviço temos os métodos com suas implementações, obedecendo ao contrato estabelecido pela interface. No caso do RMI, deve-se criar uma interface descrevendo os métodos que podem ser chamados remotamente e esta interface deverá obedecer as seguintes restrições:

**INTERFACE REMOTA**

·         Descreve os métodos para a chamada remota;

·         Descreve o comportamento do serviço;

·         Deve ser pública;

·         Obrigatoriamente deve estender Java.rmi.Remote.http://docs.goolge.com/Doc?docid=0ATOuMQ0B6nm1ZDQ3NnNzZ18wZ3huZ3A4ZnE&hl=pt\_BR&pli=1

·         Obrigatoriamente todos os métodos devem lançar Java.rmi.RemoteException

INTERFACE tem que extender  a interface REMOTE e todos os METODOS da interface do serviço devem lançar obrigatoriamente a exceção REMOTEEXCEPTION.

**Q2) Principal característica de MESSAGERIA e vantagem.**

**Conceito**

JMS (Java Message Service) é a API padrão do Java que permite a construção de componentes que criem, enviem, recebam e leiam mensagens, fornecendo uma forma de comunicação confiável, desacoplada e assíncrona.

**Características**

Comunicação realizada de forma assíncrona (ou seja o remetente da mensagem não terá necessidade de ficar aguardando a resposta daquela mensagem).

Comunicação assíncrona (o remetente da mensagem não fica bloqueado esperando a resposta).

Caso o destinatário esteja off-line, as mensagens podem ser mantidas no servidor de aplicação, garantindo a entrega assim que o destinatário estiver on-line.

-Suporte à comunicação síncrona e assíncrona;

-Suporte a transações no envio e recebimento de mensagens;

- Mensagens armazenadas em fila;

- Grau de tolerância é muito alto;

·         Fila (queue): é uma fila no sentido usual, que contém uma cópia de cada mensagem, que será consumida pelo primeiro cliente que a retirar da fila.

·         Implementa um tipo de comunicação ponto a ponto.

**Tópico (topic):** trabalha com o conceito de assinatura. Cada assinatura recebe uma cópia da mensagem, o que faz com que a mesma possa ser distribuída por todos os clientes interessados. Implementa um tipo de comunicação **publisher/subscriber**.

Desacoplamento - Comunicação Assincrona, permite que o servidor não necessita realizar a conexão com o cliente.

Corpo - Transmite os dados, corpo da mensagem, cabeçalho da msg (String)

Propriedade e Atributos - toda msg têm uma propriedade específica, como também um atributo.

Fila - Comunicação ponto a ponto, onde há vários consumidores na fila

Tópico - assinatura a msg são distribuídas via broadcast.

**Q3) Era sobre monitores, não lembro exato como era, mas se não me engano, ele queria saber como os monitores garantem exclusão mutua e sincronização.**

Os métodos sincronizados implementam os métodos de entrada e fornecem exclusão mútua para acesso aos dados controlados pelo monitor.

Apenas uma thread pode estar executando qualquer método de entrada em determinado momento.

**Exclusão Mútua** - Ato de assegurar que duas ou mais threads não se encontrem executando, simultaneamente, as respectivas regiões críticas.

**Sincronização condicional** - A thread deve aguardar até que uma condição se torne verdadeira para que ela possa  prosseguir na região crítica.

**Sincronização** - Métodos de entrada são implementados com a palavra reservada “synchronized” que permite definir um bloco sincronizado da seguinte forma: synchronized (obj) f...g. É comum utilizar a referência this como objeto de sincronização. **Método de entrada synchronized (obj).**

**Variáveis de condição para sincronização condicional.**

**Q4) O que é RMI REGISTRY e objetivos**

**Conceito**

 - Base de dados que converte o nome lógico para o endereço do objeto remoto.

 - Servidor – Registro o serviço ao RMI Registry.

 - Cliente – Consulta o RMI Registry para encontrar um serviço.

**Objetivos**

Fornecer transparência de localização, de modo que o cliente não precise conhecer a

localização do servidor.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PRIMEIRA PROVA!!!**

**1) Pediu para explicar o que era e como implementar uma interface remota.**

**(pessoal essa questão caiu no semestre passado a noite)**

**R: Definição -**Forma de invocar metodos de objetos remotos como se estes estivessem na mesma maquina.

O Objetivo do RMI é tornar os principios de programação orientada a objetos disponiveis

para o desenvolvimento de sistemas distribuidos. RMI pode ser encarada tambem como uma forma

de alto nivel para enviar e receber mensagens.

Os passos para implementar uma RMI são:

1-Definir a interface remota

2-Implementar o serviço

3-Desenvolver o serviço

4-Desenvolver o cliente

5-Iniciar o RMI registry e o servidor

6-Rodar o cliente

**2) Você iria implementar uma aplicação que envia informações para seu**

**cliente sobre atividades esportivas, pedia que você explicasse os métodos**

**e protocolos que você usaria.**

RESPOSTA 1(MARCINHO): Partindo do principio que a rede em que irão trafegar estas informações não é confiável,

utilizaria o protocolo TCP, pois este é um protocolo confiável e orientado a conexão. Isto significa que, para trocar mensagens,

os dois extremos da comunicação devem estabelecer uma conexão entre eles; isto garante que todas as

mensagens serão entregues, e na mesma ordem em que foram enviadas. Outra diferença em relação

ao protocolo UDP é que os dados das mensagens aparecem como um fluxo de bytes contínuo, sem pacotes

de tamanho fixo.

RESPOSTA 2:

**3) Deu um caso de uma ponte onde só pode passar pessoas em uma direção,**

**não pode passar nos dois sentidos simultâneamente, pediu para explicar como**

**implementaria essa aplicaçã**o.

É o mesmo caso do Leitor-Escritor. Em que:

* Várias pessoas podem ler simultaneamente
* Não pode ter conflitos do tipo escrita-escrita
* e não pode ter leitura-escrita

Então a partir destes conceitos temos:

* Várias pessoas podem passar por um mesmo lado simultâneamente.
* Duas pessoas não podem passar por lados opostos ao mesmo tempo.

A implementação seria utilizando um semáforo, para controle do acesso à ponte, que é um meio compartilhado.

Cada lado da ponte seria demarcada por um nome. No primeiro ponto A e no segundo ponto B.

Quando um carro que encontra-se no ponto A desejasse passar, ele iria chamar o método "iniciaPassagemPontoA", e quando o mesmo quisesse sair, iria chamar o método "finalizarPassagemPontoA".

A mesma coisa ocorre quando um carro encontra-se no ponto B, que iria chamar os métodos de entrada e saída "iniciaPassagemPontoB" e "finalizarPassagemPontoB" subsequentemente.

Seria adicionado dois semáforos binários, o semaforoControlePonte e o semaforoControleVariaveis, que fará o controle de acesso a ponte e o controle das variáveis.

Também será adicionado as seguintes variáveis: numAcessoA, numAcessoB. Para que seja possível fazer a contagem da quantidade de pessoas que desejam acessar de cada lado, e em seguinda dar a possibilidade de fazer o balanceamento.

O Método iniciaPassagemPontoA ficaria assim:

public void iniciaPassagemPontoA() throws InterruptedException {

    semaforoControleVariaveis.acquire();

    numAcessoA++;

if (numAcessoA== 1) semaforoControlePonte.acquire();

    semaforoControleVariaveis.release();

}

O Método iniciaPassagemPontoB ficaria assim:

public void iniciaPassagemPontoB() throws InterruptedException {

    semaforoControleVariaveis.acquire();

    numAcessoB++;

if (numAcessoB== 1) semaforoControlePonte.acquire();

    semaforoControleVariaveis.release();

}

O Método finalizarPassagemPontoA ficaria assim:

public void finalizarPassagemPontoA () throws InterruptedException {

semaforoControleVariaveis.acquire();

numAcessoA--;

if (numAcessoA== 0) semaforoControlePonte.release();

semaforoControleVariaveis.release();

}

O Método finalizarPassagemPontoB ficaria assim:

public void finalizarPassagemPontoB() throws InterruptedException {

semaforoControleVariaveis.acquire();

numAcessoB--;

if (numAcessoB== 0) semaforoControlePonte.release();

semaforoControleVariaveis.release();

}

**4) Pedia para você explicar o deadlock no algoritmo do jantar dos filósofos com semáforo.**

**(pessoal essa questão caiu no semestre passado a noite)**

**R:**Deadlock é a condição em que duas ou mais entidades esperam pela liberação de recursos que estão

de posse das demais, em uma cadeia circular.

No jantar dos filosofos acontece o deadlock se todos os filosofos adquirem simultaneamente o

garfo da direita por exemplo e assim todos ficam bloqueados esperando a liberação do garfo

da esquerda que esta de posse do filosofo da esquerda, ou seja, ninguem ira mais sair dessa

situação de bloqueio.

Isso pode ser evitado fazendo com que um dos filosofos adquira um garfo na ordem inversa

ou fazendo com que os filosofos peguem os 2 garfos simultaneamente(ATOMICAMENTE).

**5) Os métodos P() e V() do semáforo tem que ser executado em forma atômica,**

**demonstrar se não for executado desse jeito não terá exclusão mútua.**

**(pessoal essa questão caiu no semestre passado a noite)**

**R:**P() corresponde ao comando acquire() do semáforo e o V() ao comando release() que

respectivamente incrementam e decrementam a variável interna do semáforo que controla

o número de permissões. A variável interna é por si só uma região crítica, e o seu acesso

de forma não atômica pode resultar em uma situação onde duas Threads consigam

executar um comando acquire() onde só existe uma permissão.

A operação P() é implementada pelo método acquire() (há sobrecargas com timeout e para adquirir diversas permissões simultaneamente).

A operação V() é implementada pelo método release() (há sobrecargas para liberar diversas permissões simultaneamente)

\*\*\*\*\_\*\*\*\*\_\*\*\*\*\_\*\*\*\*\_\*\*\*\_\*\*\*\_

Prova final Manhã 2º semestre 2012

**1)Cite as vantagens e desvantagens de fazer sinconização condicional com monitores(era algo assim).**

**2)Explique deadlock e cite duas condições para evitar o deadlock no jantar dos filosofos.**

**3)Porque o protocolo TCP utiliza um socket no cliente e um socket no servidor, enquanto protocolo UDP não utiliza?**

**4) Explique a função da interface remota e como implementa-la e porque ela é importante?**

**5)Programa Semáforo Contador.**

**6) Messageria - Relacionado ao desacoplamento. Definição desacoplamento porque o desacoplamento é importante e qual é a função.**

**7)Programa da montaha russa, existe uma thread para o carrinho que executa as funções carrega(); anda(); descarrega(); e existe n threads para os usuários da montanha russa. O carrinho so anda quando está completamente cheio. Implemente um pseudocodigo para a situação.**

**(essa da montanha russa está a pergunta no pdf que o Laper é alucinado - vide correio fumec).**

**Boa sorte ai galera.**

**Prova final 1º semestre 2013**

**1) O que era e pra que servia sincronização condicional e exclusão mutua.**

**2) O que era deadlock e como evitar .**

**3) Em uma rede muito confiavel, que os pacotes têm tamanho fixo e a comunicação deve ser rápida, qual protocolo usar e porque.**

**4) Porquê usar a instrução while ao inves de if em monitores**

**5) Descrever, as diferenças da invocação remota**