UNIP
UNIVERSIDADE PAULISTA

SISTEMAS DA INFORMAÇÃO

MATERIAL INSTRUCIONAL ESPECÍFICO

TOMO 7

CQA/UNIP – Comissão de Qualificação e Avaliação da UNIP

SISTEMAS DA INFORMAÇÃO

MATERIAL INSTRUCIONAL ESPECÍFICO

TOMO 7

Christiane Mazur Doi

Doutora em Engenharia Metalúrgica e de Materiais, Mestra em Ciências - Tecnologia Nuclear, Especialista em Língua Portuguesa e Literatura, Engenheira Química e Licenciada em Matemática, com Aperfeiçoamento em Estatística. Professora titular da Universidade Paulista.

José Carlos Morilla

Doutor em Engenharia de Materiais, Mestre em Engenharia de Materiais e em Engenharia de Produção, Especialista em Engenharia Metalúrgica e Física e Engenheiro Mecânico, com MBA em Gestão de Empresas. Professor adjunto da Universidade Paulista.

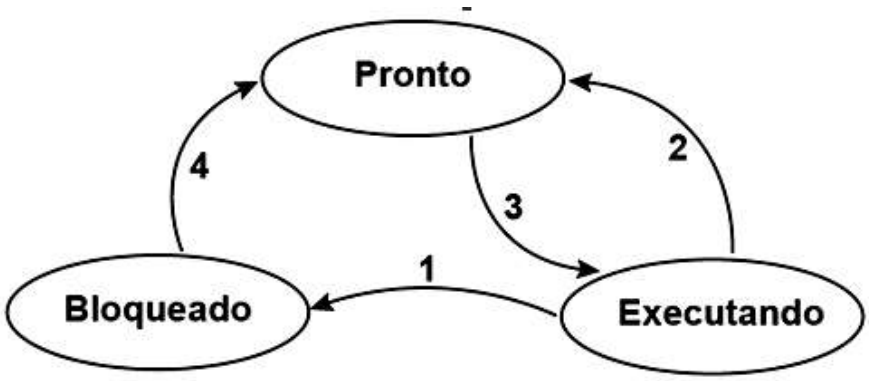
Tiago Guglielmeti Correale

Doutor em Engenharia Elétrica, Mestre em Engenharia Elétrica e Engenheiro Elétrico (ênfase em Telecomunicações). Professor titular da Universidade Paulista.

Questão 1

Questão 1.[1]

Os estados que um processo alcança podem ser modelados por meio do diagrama exibido a seguir.



TANENBAUM, A. S. *Sistemas operacionais modernos*. 3. ed. São Paulo: Pearson, 2010 (com adaptações).

No diagrama de estados apresentado, as transições causadas pelo escalonador de processos são:

A. 1 e 2.

B. 1 e 3.

C. 1 e 4.

D. 2 e 3.

E. 2 e 4.

1. Introdução teórica

1.1. Processos

Quando um programador escreve um programa em alguma linguagem de programação compilada, como a linguagem C, a tarefa do compilador é criar um binário executável para um usuário. Esse arquivo binário contém um conjunto de instruções na linguagem de máquina de determinada arquitetura, sendo que essa máquina pode ser real ou virtual.

Enquanto um programa não é executado, ou seja, enquanto não é submetido à execução na sua arquitetura, ele é simplesmente um arquivo armazenado no disco ou em algum local de armazenamento permanente, como a memória ROM. Para ser executado, o

[1]Questão 17 – Enade 2014.

programa deve ser carregado na memória principal (RAM) e em sistemas multitarefas e ser gerenciado pelo escalonador de processos.

É importante observar que cada processo é composto de, ao menos, 4 elementos: um contexto de software, um contexto de hardware, o espaço de endereçamento e o programa, conforme figura 1.

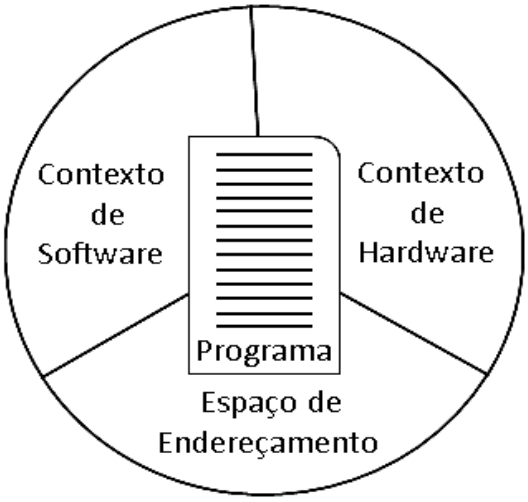


Figura 1. Estrutura de um processo.
Fonte. MACHADO e MAIA, 2013 (com adaptações).

1.2. Escalonador de processos

Um sistema operacional multitarefa pode ser executado até mesmo em um computador de um microprocessador, com um núcleo de processamento. Nesses casos, o sistema operacional deve ser capaz de dividir o tempo de execução da CPU entre os diversos processos que rodam simultaneamente. Em um sistema operacional multitarefa, o escalonador de processos é responsável por executar os diversos processos simultâneos.

O microprocessador de uma máquina moderna executa em velocidade maior do que a velocidade da maioria dos dispositivos ligados direta ou indiretamente aos seus barramentos. Além disso, a banda disponível para o acesso ao barramento é limitada e o barramento pode tornar-se congestionado, dependendo da sua utilização.

Frequentemente, a CPU pode requerer um dado que não está diretamente disponível em um de seus registradores ou em sua memória cache interna. Nesse caso, é necessário acessar a memória principal ou algum dispositivo, como o disco rígido ou a rede. O acesso a esses dispositivos costuma ser mais lento do que o acesso à CPU. Assim, a CPU pode passar grande quantidade de ciclos enquanto espera o recebimento de dados requisitados. Nesse

intervalo de tempo de espera, o processo pode ficar parado e aguardar os dados adicionais. Se a CPU também ficasse parada, sua capacidade de processamento seria desperdiçada, uma vez que o tempo de processamento útil seria gasto apenas no modo espera.

Alternar a execução para um outro processo que esteja pronto para ser executado e que precise de tempo de processamento da CPU é uma estratégia para que a CPU possa, então, executar esse processo por um período limitado ou até que esse processo seja bloqueado por algum motivo. Se o processo bloqueado anteriormente estivesse disponível, nesse momento, sua execução poderia ser retomada. As interrupções são ativadas assim que algum evento relevante seja detectado, como a chegada de dados.

Dessa forma, podemos identificar pelo menos três estados de um processo em um sistema operacional multitarefa:

- estado de execução, no qual a CPU executa corretamente o processo;
- estado bloqueado, no qual a execução de um processo ou de um grupo de processos está suspensa porque aguarda algum evento, como o recebimento de um dado;
- estado pronto, no qual os processos que poderiam ser executados estão "aguardando a sua vez".

Um dos papéis do escalonador de processos é decidir se o processo continua em execução ou se há alternância para outro processo no estado "pronto". Se o escalonador decidir executar outro processo, devido ao esgotamento do período de tempo alocado para a sua execução, o processo atual passa para o estado pronto e o outro processo passa a ser executado, saindo do estado pronto para o estado de execução.

2. Resolução da questão

No modelo simplificado do enunciado, apenas as transições 2 e 3 são geradas pelo escalonador, que decide qual processo terá tempo de CPU em dado instante. A transição 1 ("bloqueada") é gerada quando um processo está aguardando alguma informação e não pode mais continuar sua execução. A transição 4 (de "bloqueada" para "pronta") é gerada quando a informação aguardada é recebida.

Alternativa correta: D.

3. Indicações bibliográficas

- MACHADO, F. B.; MAIA, L. P. *Arquitetura de sistemas operacionais.* 5. ed. Rio de Janeiro: LTC, 2013.
- SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Fundamentos de sistemas operacionais – princípios básicos.* Rio de Janeiro: LTC, 2013.
- TANENBAUM, A. D. *Sistemas operacionais modernos*. 3. ed. São Paulo: Pearson Prentice Hall, 2010.

Questão 2

Questão 2.[2]

Na figura 1, abaixo, está representado, esquematicamente, um processo de tradução de um programa (arquivo fonte) em um código binário. Esse processo de compilação clássica é utilizado em compiladores como os das linguagens C e Pascal.

Figura 1 - Processo de compilação clássica

Na figura 2, abaixo, está representado, esquematicamente, um processo de tradução de um programa (arquivo fonte) em um código intermediário. Esse processo híbrido é utilizado em compiladores como os das linguagens Java e C#.

Figura 2 - Processo híbrido

Considerando que, em ambos os processos, o código binário é o que será executado pelo computador e que a execução de dois programas gerados, cada qual por um dos processos apresentados, ocorre em situações equivalentes, avalie as afirmativas a seguir.

I. Há portabilidade para a execução de ambos os programas gerados em cada processo.

II. Na execução do programa gerado por meio do processo híbrido, o consumo de memória é maior do que na execução pelo processo de compilação clássica.

III. O desempenho na execução do programa gerado pelo processo de compilação clássica é melhor do que na execução pelo processo híbrido.

É correto o que se afirma em

A. I, apenas.

B. III, apenas.

C. I e II, apenas.

D. II e III, apenas.

E. I, II e III.

[2]Questão 13 – Enade 2014.

1. Introdução teórica

1.1. Tipos de linguagens de programação

Existem três abordagens para a construção de linguagens computacionais, conforme segue.

- Linguagens compiladas.
- Linguagens interpretadas.
- Linguagens que utilizam abordagem híbrida.

Essas três abordagens podem ser mais bem compreendidas se considerarmos a evolução histórica da computação, uma vez que seu desenvolvimento está ligado ao contexto tecnológico no qual elas surgiram.

1.2. Primórdios da computação

Computadores são dimensionados para trabalhar com instruções diretamente executadas pelo microprocessador. Essas instruções caracterizam determinada arquitetura e costumam ser de caráter lógico, caráter aritmético ou destinadas à manipulação de dados vindos da memória. Elas costumam ser bem simples se comparadas às linguagens de alto nível e lidam diretamente com o hardware de uma plataforma específica.

Nos primórdios da informática, os programadores tinham de escrever seus programas diretamente na linguagem de máquina, algumas vezes sem o auxílio de qualquer tipo de ferramenta. Nos primeiros computadores eletrônicos, um programa devia ser feito por meio de ligações físicas no computador, como era o caso do ENIAC.

Um dos primeiros avanços para facilitar a programação de computadores foi o desenvolvimento das linguagens dos montadores (*assemblers*). Em vez de trabalhar diretamente em linguagem de máquina, com números em binários, octais ou hexadecimais, os montadores utilizavam mnemônicos, instruções textuais substituídas pela linguagem de máquina. Outro avanço foi a inclusão das macros, análogas às funções nas linguagens estruturadas e compostas de instruções mais complexas, que possibilitaram o surgimento de instruções com comportamento mais rico (*take it or leave it?*).

1.3. Desenvolvimento das linguagens compiladas e das linguagens interpretadas

Com o objetivo de simplificar o processo de desenvolvimento de software, foram desenvolvidas linguagens de programação de mais alto nível, como o FORTRAN, o COBOL e o LISP. Essas linguagens possibilitaram o desenvolvimento de programas não ligados totalmente a dada arquitetura computacional.

Isso também criou um problema prático: o programador escreve o programa em uma linguagem de alto nível, mas o computador só é capaz de "compreender" a sua linguagem de máquina.

Uma das formas de resolver esse problema é utilizar um programa que faz o papel de um "tradutor": o código é originalmente escrito em uma linguagem de alto nível, é "traduzido" para uma linguagem de baixo nível e, depois, é "traduzido" para a linguagem de máquina.

E em um problema de tradução da linguagem, podemos escrever um texto em, por exemplo, português, e, depois de terminado, contratamos alguém para traduzir o texto para outra língua, por exemplo, o inglês. Nesse caso, o texto original encontra-se pronto antes de ser feita a tradução para outra língua. Ou, ainda, poderíamos ditar um texto na língua original para o tradutor, que iria, então, fazer uma tradução simultânea.

Essas duas analogias mostram a diferença entre linguagens compiladas e linguagens interpretadas. As linguagens compiladas funcionam de forma similar ao primeiro cenário: inicialmente escrevemos o programa, que corresponde ao texto na língua original, e, depois, submetemos esse programa ao compilador, que faz o papel do tradutor.

Já as linguagens interpretadas funcionam de forma similar à segunda abordagem: as instruções são passadas aos poucos ao interpretador da linguagem, que faz tradução simultânea para a linguagem de máquina.

Um programa em uma linguagem compilada pode ser analisado pelo compilador, que, por sua vez, pode identificar uma série de otimizações. Dessa forma, o compilador não vai "traduzir" o programa de uma linguagem para outra; vai fazer com que o programa execute de forma mais rápida ou que ele seja mais eficiente no consumo de memória. Contudo, essa abordagem apresenta uma desvantagem: é necessário que o programa tenha condições que possibilitem a compilação completa.

Em uma linguagem interpretada, temos a vantagem de não precisarmos do programa completo para a execução. Porém, como temos apenas "fragmentos" de um programa

completo, muitas otimizações não são possíveis de serem feitas. Há a possibilidade de utilizarmos um JIT (*just in time compiler*), que pode otimizar a execução de diversas formas, inclusive as que não estão disponíveis em um programa compilado (AYCOCK, 2003).

Um programa é portável quando pode ser compilado em diversas plataformas diferentes sem nenhuma modificação no seu código. No entanto, o código binário gerado, executável ou de biblioteca, é específico de uma plataforma.

Em linguagens interpretadas, o mesmo código "intermediário" é diretamente consumido pelo interpretador em diferentes plataformas. Do ponto de vista do usuário, é como se um mesmo programa fosse executado em diversas plataformas sem a necessidade de uma nova compilação.

1.4. Desenvolvimento da abordagem híbrida

Em linguagens como Java e C#, o compilador não gera um código para uma arquitetura virtual. Essa arquitetura corresponde a uma abstração de um computador real e roda de forma simulada em uma máquina real. Por exemplo, no caso da linguagem Java, o compilador gera um binário que roda na máquina virtual Java (JVM), a arquitetura virtual, que apresenta as mesmas características em todas as plataformas onde ela existe.

Esse binário gerado para a máquina virtual (ou código intermediário, dependendo da linguagem) deve ser convertido em uma linguagem de máquina que possa ser executada em dada arquitetura, em um processo similar ao da interpretação. Como a etapa final é feita pela máquina virtual na plataforma específica, essa abordagem torna possível escrevermos programas que rodem em diversas plataformas diferentes, sem a necessidade de compilações específicas. Diferentemente das linguagens puramente interpretadas, como, por exemplo, Python ou Javascript, esse código é gerado a partir do código original, mas do código intermediário gerado para a máquina virtual.

A abordagem híbrida pode oferecer alguns benefícios de ambas as abordagens, a puramente compilada e a puramente interpretada, mas também oferece desafios. Primeiramente, o programa é tão portável quanto a disponibilidade de máquinas virtuais existentes em diferentes plataformas. Isso significa que deve haver grande esforço para o desenvolvimento e a manutenção das máquinas virtuais em diferentes plataformas. Máquinas virtuais podem ser bastante sofisticadas para serem desenvolvidas e mantidas e devem comportar-se da mesma forma nas diferentes plataformas.

2. Análise das afirmativas

I – Afirmativa incorreta.
JUSTIFICATIVA. O código binário gerado no processo de compilação clássico é específico para uma plataforma.

II – Afirmativa correta.
JUSTIFICATIVA. Frequentemente, o consumo de memória de um programa gerado pelo processo híbrido tende a ser maior. Isso ocorre porque, normalmente, não é possível fazer algumas otimizações como as encontradas em um programa puramente compilado, o que ocasiona maior consumo de memória. Além disso, o próprio ambiente utilizado para a interpretação ocupa espaço na memória e precisa de tempo de processamento. Por um lado, o JIT, pode oferecer algumas técnicas de otimização não existentes no processo de compilação tradicional, uma vez que pode analisar o programa em tempo de execução, algo que não pode ser feito na abordagem puramente compilada. Por outro, o JIT e o ambiente de execução também ocupam espaço na memória.

III – Afirmativa correta.
JUSTIFICATIVA. Do ponto de vista da execução, o desempenho é o foco e não o consumo de memória. Ainda que um JIT também ofereça várias otimizações para acelerar o processo de execução, na maior parte dos casos, os programas gerados pelo processo convencional de compilação para uma arquitetura específica costumam ser mais rápidos, especialmente por utilizarem diversas técnicas de otimização.

Alternativa correta: D.

3. Indicações bibliográficas

- AYCOCK, J. A brief history of just-in-time. *ACM Computing Surveys (CSUR)*, v.35, n.2, p.97-113, 2003.
- SCOTT, M. L. *Programming language pragmatics*. 4. ed. Waltham: Morgan Kaufmann, 2015.

Questão 3

Questão 3.[3]

Considere uma situação em que um professor que queira saber se existem alunos cursando, ao mesmo tempo, as disciplinas A e B, tenha implementado um programa que

1) inicializa um array a de 30 posições que contém as matrículas dos alunos da disciplina A;

2) inicializa outro array b de 40 posições, que contém as matrículas dos alunos da disciplina B;

3) imprime a matrícula dos alunos que estão cursando as disciplinas A e B ao mesmo tempo.

Considere, ainda, que os arrays foram declarados e inicializados, não estão necessariamente ordenados, e seus índices variam entre 0 e n -1, sendo n o tamanho do array.

```
1. for ( i = 0 to 29 ) {
2.     for( j = 0 to 39 ) {
3.
4.
5.
6.     }
7. }
```

Com base nessas informações, conclui-se que o trecho a ser incluído nas linhas 3, 4 e 5 do código acima, para que o programa funciona corretamente, é

A.

```
3. if (a[i] == b[j]) {
4.       print(a[i]);
5. }
```

B.

```
3. if (a[j] == b[i]) {
4.       print(a[j]);
5. }
```

C.

```
3. if (a[i] == b[j]) {
4.       print(a[j]);
5. }
```

[3]Questão 9 – Enade 2014.

D.

```
3. if (a[i] == b[i]) {
4.      print(a[i]);
5. }
```

E.

```
3. if (a[j] == b[j]) {
4.      print(a[j]);
5. }
```

1. Introdução teórica

1.1. Estruturas de dados estáticas (arrays)

As linguagens de programação costumam prover recursos para a organização eficiente dos dados. Esses recursos podem ser divididos em dois tipos básicos de estruturas de dados: estáticas e dinâmicas.

As estruturas de dados estáticas apresentam um espaço fixo em memória, que é fixado pelo programador no tempo de compilação. A vantagem dessas estruturas é permitir que o programador preveja exatamente o tamanho de memória que será consumido pelo programa. Além disso, o programador tem como saber, de forma bastante simples, os limites de todas as estruturas de memória de antemão.

Como o tamanho dessas estruturas é fixo, não é possível alocar mais espaço em memória caso ele se torne necessário. Isso significa que o programador deve saber alocar espaço na memória para o programa durante sua escrita ou antes da compilação. Se o programador não souber a quantidade de memória necessária com antecedência, sua única alternativa será alocar um espaço maior do que o necessário, o que pode desperdiçar memória, mas garante o funcionamento do programa.

Em muitas linguagens de programação armazenadas sequencialmente, como a linguagem C, um array é implementado como um conjunto de posições de memória de um mesmo tipo (RICHARDSON, 2007). Os arrays formam uma espécie de "reticulado", com posições de memórias vizinhas e com o mesmo tamanho. Dessa forma, a declaração de um array envolve a definição de um tipo e de um tamanho máximo e o conhecimento da posição de memória inicial. Os elementos são acessados por meio de um índice, que normalmente inicia em zero. Como todos os elementos são do mesmo tipo, o array é uma

estrutura de dados homogênea. Como os dados são acessados por um índice e de forma sequencial, o array é uma estrutura linear. Como o tamanho de um array é definido em tempo de compilação, seu tamanho é fixo e trata-se de uma estrutura estática (PAHUJA, 2007).

Por meio de um conjunto de células vizinhas (ou de posições de memória vizinhas), podemos visualizar um array na linguagem C, como ilustrado na figura 1. Nessa imagem, há o array com N elementos, sendo que cada elemento é um número inteiro que ocupa um tamanho fixo na memória, dependendo da plataforma, e é acessado por meio de índices (0 até N-1). Isso totaliza N elementos.

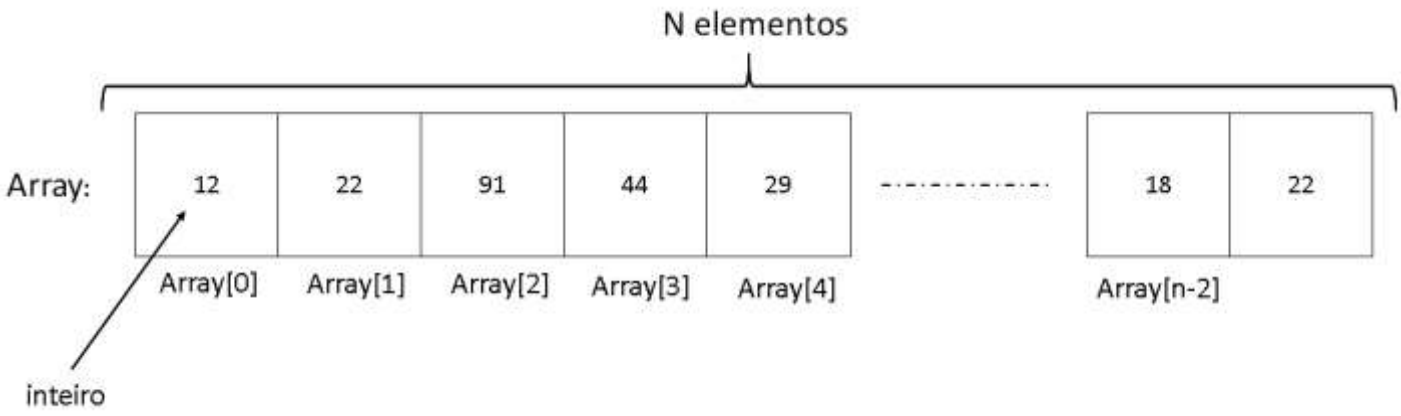


Figura 1. Array de números inteiros.

Normalmente, as linguagens de programação também disponibilizam outras estruturas de dados mais poderosas, como os vetores. No caso da linguagem C++, os vetores têm uma notação similar à notação dos arrays, mas apresentam um mecanismo de alocação dinâmica de memória. Eles podem "crescer" à medida que adicionamos novos elementos, não sendo necessário saber o seu tamanho no tempo de compilação.

2. Resolução da questão

Cada array contém as matrículas correspondentes a cada uma das turmas. É necessário identificar as matrículas que pertencem aos dois arrays ao mesmo tempo. A variável i é utilizada para percorrer o array a, de 30 elementos, enquanto a variável j é utilizada para percorrer o array b, de 40 elementos. O loop for da linha 1 percorre o array a, enquanto o loop for da linha 2 percorre o array b. Assim, a condição de igualdade ocorre quando a[i]==b[j]. Na condição de igualdade, podemos imprimir tanto a[i] quanto b[j],

porque são iguais, embora o problema ofereça apenas alternativas relacionadas ao array a. Essa análise leva à seleção da alternativa A.

Nas alternativas B, C e E, erroneamente, utiliza-se o array a com o índice j. Esse índice varia de valores de 30 até 39, levando a uma série de acessos incorretos de memória para valores de j superiores a 29. Na alternativa D, o array b é indexado pela variável i, o que também é incorreto.

Alternativa correta: A.

3. Indicações bibliográficas

- CELES, W.; CERQUEIRA, R.; RANGEL, J. R. *Introdução à estrutura de dados.* Rio de Janeiro: Campus, 2004.
- PAHUJA, S. *A Practical approach to data structures and algorithms*. New Delhi: New Age International, 2007.
- RICHARDSON, D. R. *The book on data structures*. Lincoln: iUniverse, 2002.

Questões 4

Questão 4.[4]

A tabela abaixo apresenta o ranking das 20 linguagens de programação mais populares.

Jul 2014	Jul 2013	*Change*	*Programming Language*	*Ratings*	*Change*
1	1		C	17.145%	-0,48%
2	2		Java	15.688%	-0,22%
3	3		Objective-C	10.294%	+0,05%
4	4		C++	5.520%	-3,23%
5	7	^	(Visual) Basic	4.341%	+0,01%
6	6		C#	4.051%	-2.16%
7	5	v	PHP	2.916%	-4.27%
8	8		Python	2.656%	-1.38%
9	10	^	JavaScript	1.806%	-0,04%
10	12	^	Transact-SQL	1.759%	+0,19%
11	9	v	Perl	1.627%	-0,52%
12	13	^	Visual Basic .NET	1.495%	+0,24%
13	37	^^	F#	1.093%	+0,86%
14	11	v	Ruby	1.072%	-0,51%
15	45	^^	ActionScript	1.067%	+0,86%
16	-	^^	Swift	1.054%	+1,05%
17	17		Delphi/Object Pascal	1.031%	+0,34%
18	15	v	Lisp	0,829%	-0,04%
19	18	v	MATLAB	0,781%	+0,10%
20	20		Assembly	0,777%	+0,20%

Disponível em: <http://www.tiobe.com>. Acesso em: 23 jul. 2014 (adaptado).

[4]Questão 14 – Enade 2014.

Considerando a classificação das linguagens de programação apresentada na tabela, avalie as afirmativas a seguir.

I. O ranking é liderado por uma linguagem de terceira geração e de paradigma funcional.

II. Linguagens com licença livre têm participação inferior a 20%, se somados os percentuais de popularidade.

III. É maior, no ranking, a participação de linguagens que utilizam o paradigma orientado a objetos que a das linguagens que utilizam os demais paradigmas.

É correto o que se afirma em:

A. II, apenas.

B. III, apenas.

C. I e II, apenas.

D. I e III, apenas.

E. I, II e III.

1. Introdução teórica

Paradigmas de linguagens de programação

Existem diversas linguagens de programação. Muitas dessas linguagens apresentam características semelhantes, mas podem ser agrupadas em diferentes paradigmas. Segundo Tucker e Noonan (2008), "um paradigma de programação é um padrão de resolução de problemas que se relaciona a um determinado gênero de programas e linguagens".

Podemos dividir as linguagens de programação em dois grandes grupos:

- linguagens imperativas;
- linguagens declarativas.

A principal diferença entre esses dois tipos está na forma com que o programador expõe a resolução do problema. Nas linguagens declarativas, o programador deve descrever "como" o programa deve se comportar (SCOTT, 2016). Assim, o programa é composto de "ordens" que o computador deve seguir para resolver dado problema.

A outra abordagem possível é a declarativa, em que o programador expõe "o quê" o computador deve fazer, mas não necessariamente "como" fazer (SCOTT, 2016). A maioria dos programadores está acostumada a trabalhar com linguagens imperativas. Um problema computacional pode ser descrito não apenas em função dos passos para a sua solução, mas também listando e descrevendo as restrições à sua solução.

Alternativamente à divisão em linguagens imperativas e declarativas, é possível identificar 6 categorias de linguagens de programação, conforme segue (SCOTT, 2015).

- Linguagens funcionais: baseadas em um modelo de definição recursiva de funções, inspirada pelo cálculo lambda proposto por Alonzo Church. Exemplos: Lisp, ML, Haskell (SCOTT, 2015).
- Linguagens dataflow (fluxo de dados): modelo computacional que corresponde ao fluxo de informação entre nós, de forma paralela. Exemplos: Id e Val (SCOTT, 2015).
- Linguagens da família von Neumann: nome dado em homenagem ao físico e matemático John von Neumann (1903-1957). O modelo computacional de von Neumann é o mais comum. O computador é visto como uma CPU que atua sobre uma memória e contém tanto o programa quanto os dados e as variáveis para a sua solução. Nesse modelo, o programa é considerado uma série de declarações que atuam de forma direta ou inireta no estado da CPU e na memória. As linguagens mais populares estão nessa categoria, como Fortran, Ada e C (SCOTT, 2015).
- Linguagens orientadas a objetos: essa família de linguagens foi especificamente desenvolvida para facilitar o desenvolvimento de modelos orientado a objetos. Nesse paradigma, o programa é composto de classes que vão dar origem a objetos (processo de instanciação de uma classe). Cada classe apresenta um conjunto de métodos e atributos e o relacionamento entre as classes é feito de forma a incorporar informações sobre o domínio do problema ou sobre o design da solução. Exemplos: C++ e Java (SCOTT, 2015).
- Linguagens de scripts: esse tipo de linguagem apresenta elementos mistos de todas as categorias anteriores. Os scripts normalmente são feitos para integrar diversas soluções. Exemplos: Perl, Python e Ruby (SCOTT, 2015).
- Linguagem de montagem (assembly): linguagem que representa a linguagem de máquina por meio de códigos facilmente memorizáveis (mnemônicos). Essas linguagens também são chamadas de segunda geração, sendo que a programação diretamente em binários compreensíveis para a máquina é de primeira geração.

2. Análise das afirmativas

I – Afirmativa incorreta.

JUSTIFICATIVA. A linguagem C é uma linguagem de paradigma imperativo e da família von Neumann, apesar de ser de terceira geração.

II – Afirmativa incorreta.

JUSTIFICATIVA. Se somarmos os percentuais de participação de apenas duas linguagens de licença livre (como C e C++), obtemos mais de 20%.

III – Afirmativa correta.

JUSTIFICATIVA. No quadro 1, podemos identificar as linguagens que suportam o paradigma orientado a objetos de diferentes formas.

Quadro 1. Linguagens que suportam o paradigma orientado a objetos.

Linguagem	**Percentual**
Java	15,688%
Objective-C	10,294%
C++	5,520%
C#	4,051%
Python	2,656%
Javascript	1,806%
Perl	1,627%
Visual Basic .NET	1,495%
F#	1,093%
Ruby	1,072%
ActionScript	1,067%
Swift	1,054%
Delphi/Object Pascal	1,031%
Lisp	0,829%
MATLAB	0,781%
Total:	**50,06%**

Como temos mais de 50% das linguagens que suportam o paradigma orientado a objetos, podemos dizer que a afirmativa III está correta.

Alternativa correta: B.

3. Indicações bibliográficas

- SCOTT, M. L. *Programming language pragmatics*. 4. ed. Waltham: Morgan Kaufmann, 2015.
- TUCKER, A. B.; NOONAN, R. E. *Linguagens de programação: princípios e paradigmas*. 2. ed. São Paulo: McGrawhill, 2008.

Questão 5

Questão 5.[5]

Leia o texto a seguir.

A virtualização de servidores, acompanhada de solução para Gerenciamento Eletrônico de Documentos (GED), trouxe uma série de benefícios para as escolas de uma instituição: acumulados ao longo de 50 anos, os prontuários dos estudantes passaram a ser acessados em tempo real, as atividades pedagógicas realizadas em ambiente mais sólido e o gerenciamento centralizado do ambiente passou a permitir que a equipe da instituição se concentre em tarefas estratégicas. No médio prazo, a meta da instituição é virtualizar toda a infraestrutura de servidores e eliminar o uso de papel nos processos, trocando documentos em papel por documentos eletrônicos armazenados nos servidores virtuais e acessíveis em qualquer unidade da rede de escolas.

Computerworld*. Virtualização: quatro casos de sucesso no Brasil.* Disponível em <http://cio.com.br>. Acesso em 29 jul. 2014 (com adaptações).

Em relação ao estudo de caso mencionado acima, avalie as asserções a seguir e a relação proposta entre elas.

I. A tecnologia de virtualização de servidores está proporcionando diversos benefícios para as organizações, como redução de custos e de *downtime* e melhoria no desempenho, na segurança e no gerenciamento da infraestrutura de tecnologia da informação (TI). No entanto, sozinha, a virtualização não é suficiente para permitir que as organizações se concentrem em tarefas estratégicas.

PORQUE

II. A virtualização, se associada a outras tecnologias atuais como GED, *Cloud Computing* e *Grid Computing*, possibilita que as organizações foquem no negócio, eliminando os ricos associados à área de TI.

A respeito dessas asserções, assinale a opção correta.

A. As asserções I e II são proposições verdadeiras, e a asserção II justifica a I.

B. As asserções I e II são proposições verdadeiras, e a asserção II não justifica a I.

C. A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.

D. A asserção I é uma proposição falsa, e a II é uma proposição verdadeira.

E. As asserções I e II são proposições falsas.

[5]Questão 16 – Enade 2014.

1. Introdução teórica

Virtualização de servidores

A capacidade de processamento de computadores é superior ao consumo típico de uma única aplicação: as máquinas atuais dispõem dessa extraordinária capacidade de entrada/saída. Por isso, é possível que uma máquina fique ociosa por um tempo, o que significa que uma mesma máquina poderia ser utilizada para mais de uma aplicação simultaneamente e, dessa forma, ser mais eficiente.

Um dos objetivos no desenvolvimento de um sistema operacional é o isolamento dos processos em execução e dos usuários entre si. Cada usuário deve utilizar o sistema, influenciando os demais usuários, como ocorre no caso de comunicação. Cada processo deve executar de forma independente, excetuando os casos em que essa comunicação é deliberada.

Em uma situação extrema, poderíamos querer rodar ao mesmo tempo mais de um sistema operacional em uma máquina. Além de abstrair o acesso ao hardware, podemos abstrair completamente o funcionamento desse hardware, oferecendo a possibilidade de simular um hardware virtual. Dessa forma, permite-se efetivamente criar "máquinas virtuais" definidas por software. Essa é a virtualização.

A virtualização oferece uma série de vantagens: as máquinas virtuais rodam em uma máquina real e podem rodar diferentes sistemas operacionais, com configurações específicas para cada aplicação, o que leva a um processo de atualização e migração mais rápido e mais eficiente.

No entanto, existe um custo computacional associado a esse tipo de solução, porque uma série de operações feitas por um hardware dedicado devem ser feitas, em um segundo momento, por um software.

Conforme Marshall, Reynolds e McCroy (2006), para ilustrar as aplicações executando diretamente em um servidor, temos a figura 1. Os diversos aplicativos rodam sobre o sistema operacional diretamente instalado na máquina real.

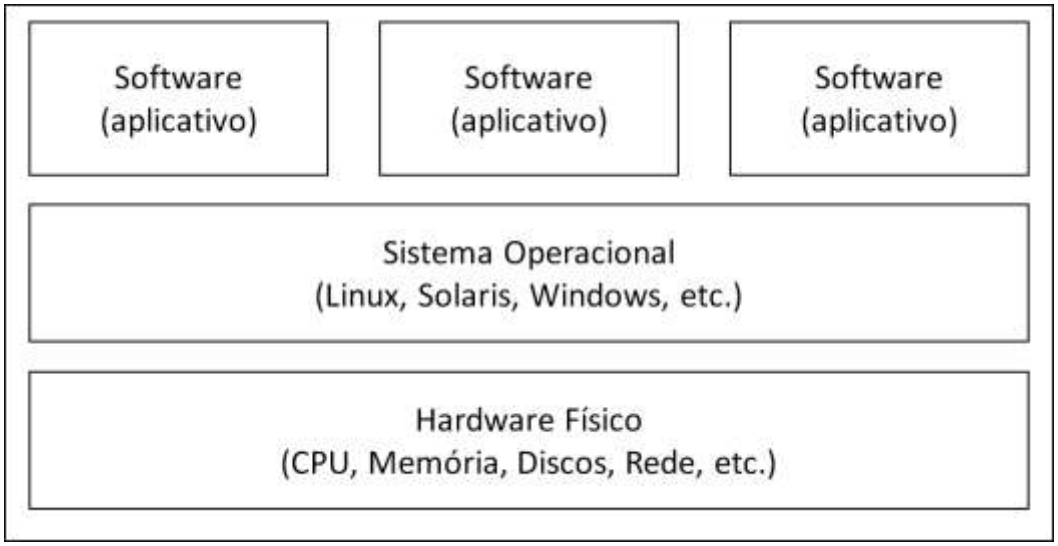


Figura 1. Arquitetura tradicional.
Fonte. MARSHALL, REYNOLDS e MCCROY, 2006 (com adaptações).

Outra possibilidade é ilustrada na figura 2. Nesse caso, um conjunto de máquinas virtuais é executado em um hardware real. Cada máquina virtual pode executar um sistema operacional diferente e ter um conjunto de configurações específicas. Além disso, cada aplicativo em cada máquina virtual está completamente isolado dos aplicativos que rodam nas demais máquinas virtuais. A "máquina real" executa um hypervisor, uma espécie de sistema operacional mínimo, projetado para executar máquinas virtuais (MARSHALL, REYNOLDS e MCCROY; 2006). Essa é a principal estrutura utilizada em servidores de empresas que vendem serviços de virtualização.

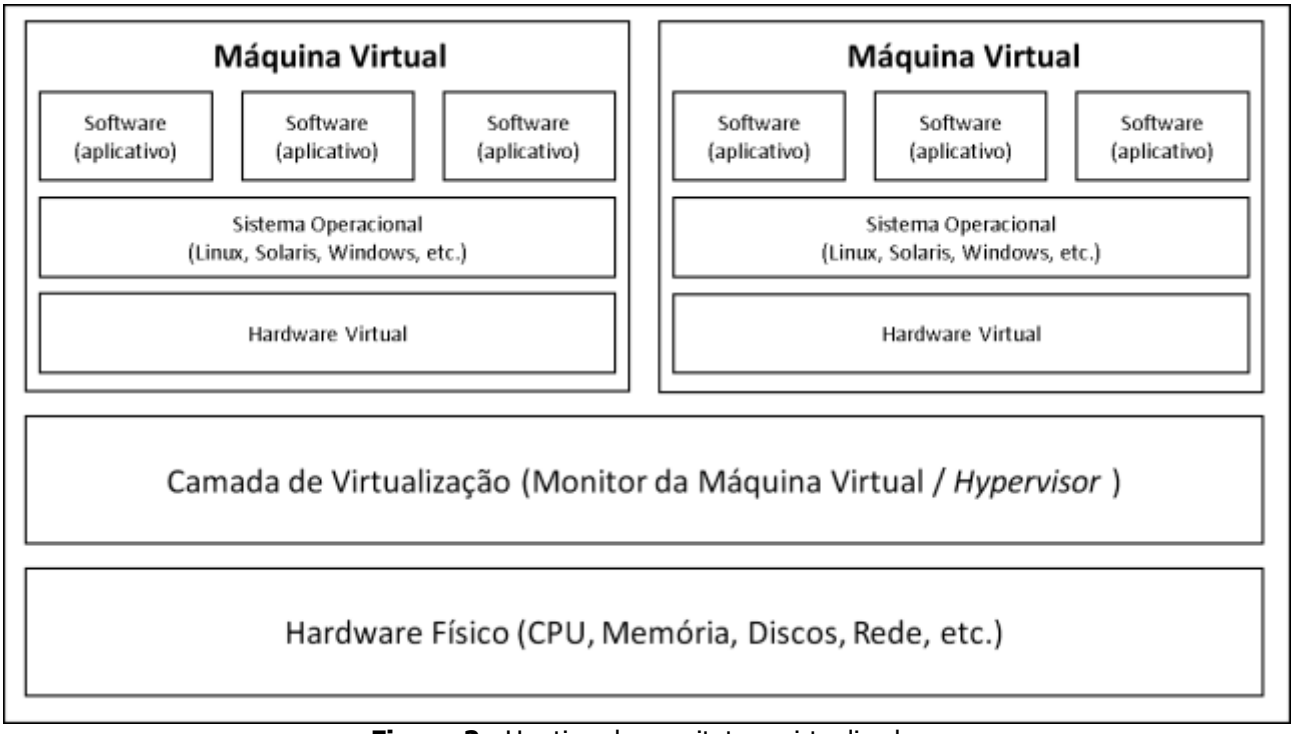


Figura 2. Um tipo de arquitetura virtualizada.
Fonte. MARSHALL, REYNOLDS e MCCROY, 2006 (com adaptações).

Na figura 3, apresenta-se outra possibilidade: um sistema operacional executa em um hardware real e esse sistema tem um software de virtualização instalado, como alguns produtos da VMWare ou o VirtualBox da Oracle, entre outros. Esse software "virtualiza" a máquina real e torna-a capaz de executar diversas máquinas virtuais, cada uma com um sistema operacional potencialmente diferente. Essa abordagem é bastante comum em desktops e workstations, devido à sua praticidade (MARSHALL, REYNOLDS e MCCROY; 2006). No entanto, essa abordagem impõe um custo computacional mais elevado do que a proposta apresentada na figura 2, que tende a ser mais eficiente e, por isso, é bastante utilizada em servidores.

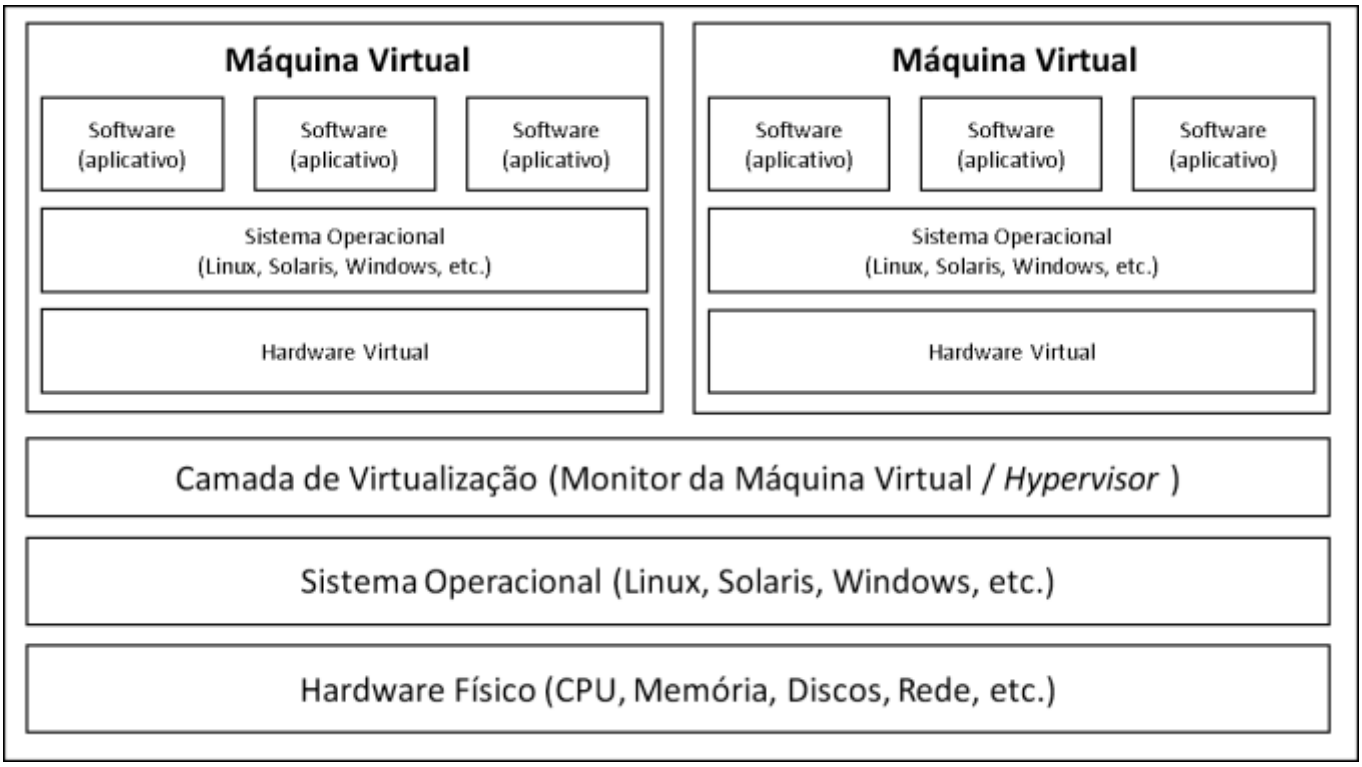


Figura 3. Arquitetura virtualizada rodando sobre um sistema operacional.
Fonte. MARSHALL, REYNOLDS e MCCROY, 2006 (com adaptações).

2. Análise das asserções

I – Asserção falsa.

JUSTIFICATIVA. Ainda que a virtualização de servidores possa ajudar a reduzir custos de infraestrutura da área de tecnologia da informação (TI), não se pode afirmar que a simples utilização da virtualização aumente a segurança da área, uma vez que a segurança depende de muitos outros fatores, que afetam tanto sistemas convencionais quanto sistemas virtualizados.

II – Asserção falsa.

JUSTIFICATIVA. A virtualização pode ajudar a reduzir custos, especialmente se aliada a outras tecnologias, como a computação em nuvem. Contudo, a simples adoção dessas tecnologias não garante que as organizações deem a devida atenção a isso, porque depende de uma série de medidas administrativas e estratégicas. A mera adoção de determinada tecnologia não garante o sucesso de uma corporação; pode contribuir para que haja sucesso, mas também pode prejudicar o desempenho da empresa.

Alternativa correta: E.

3. Indicação bibliográfica

- MARSHALL, D.; REYNOLDS, W. A.; MCCROY, D. *Advanced Server Virtualization: VMware and Microsoft Platforms in the Virtual Data Center*. Boca Raton: Auerbach Publications, 2006.

Questão 6

Questão 6.[6]

Um analista de sistemas usa um computador para executar apenas 2 tipos de operações: processamento lógico/aritmético e acesso a disco.

O computador é um servidor com apenas um processador de um núcleo (core). Além do sistema operacional multitarefa e dos processos do sistema, tal servidor serializa os acessos ao disco e é usado para executar apenas os programas que o analista solicita. O analista trabalha com dois tipos de programas: aqueles que realizam em torno de 100% de operações lógicas/aritméticas (quantidade desprezível de acesso a disco), denominados *cpu-bound*, e aqueles que executam em torno de 70% de acesso a disco e 30% de operações lógicas/aritméticas, denominados *io-bound*. Certa vez, o analista executou alguns programas isoladamente e mediu os tempos de execução de cada um. Os dados obtidos foram registrados na tabela I. Em seguida, ele combinou programas e os submeteu simultaneamente em grupos. Os resultados foram registrados na tabela II.

Tabela I

Programa	Tipo	Tempo de Execução (ut)
P01	CPU-BOUND	20
P02	CPU-BOUND	40
P03	CPU-BOUND	60
P04	IO-BOUND	50
P05	IO-BOUND	100
P06	IO-BOUND	150

Tabela II

Grupo	Tempo de Execução (ut)
1: P01+P02+P03	123
2: P04+P05+P06	210
3: P03+P05+P06	160
4: Todos	306

Dias depois, revendo as tabelas, ele teve dúvida quanto à correção dos tempos anotados. Considere ut=unidade de tempo.

[6]Questão 19 – Enade 2014.

Com relação a essa situação, avalie as afirmativas a seguir.

I. É possível que o tempo total do grupo 4 seja 306ut.

II. O tempo total do grupo 1 deveria ser, no mínimo, de 100ut.

III. O tempo total do grupo 2 deveria ser, no mínimo, de 300ut.

IV. O tempo total do grupo 3 deveria estar acima de 175ut e abaixo de 310ut.

É correto o que se afirma em

A. III, apenas. B. I e II, apenas. C. I e IV, apenas.

D. II, III e IV, apenas. E. I, II, III e IV.

1. Introdução teórica

Processos e consumo de recursos

Em um computador, os processos podem requerer dois tipos de recursos:

- recursos de processamento;
- recursos de entrada/saída.

O recurso de processamento corresponde às ações típicas de uma CPU, como a execução de operações lógico-aritméticas. O recurso de entrada/saída corresponde às diversas operações em que dados são recebidos ou enviados do barramento do computador para a memória principal do computador ou para o microprocessador.

Para compreender os tipos de processos, suponha uma solução numérica de uma equação diferencial. Nesse caso, devem ser feitas operações matemáticas, que requerem o uso de elevada capacidade de processamento (memória RAM), especialmente em operações de ponto flutuante. Dependendo do tipo de equação diferencial e do método numérico utilizado, pode ser preciso o uso de muita memória RAM. É bastante provável que não sejam necessárias muitas operações de entrada e saída, uma vez que o problema vai ser resolvido por meio de um método numérico pré-estabelecido e as informações requeridas para sua resolução podem ser estabelecidas pelas próprias equações, pelas condições iniciais e pelo método de solução escolhido.

Vamos supor um problema diferente: em uma empresa com muitos empregados, é necessário fazer um relatório que envolva informações sobre cada um dos funcionários e sobre os diversos setores da empresa. As operações matemáticas do relatório podem ser muito simples, resumindo-se apenas às quatro operações aritméticas básicas. Contudo, um grande número de dados precisa ser resgatado, provavelmente de um banco de dados. É

bastante provável que essas informações estejam armazenadas em um disco ou em um sistema de storage, o que significa que a máquina que estiver rodando o relatório potencialmente tem de fazer elevado número de acessos a algum dispositivo externo. Esse processo pode envolver, também, a rede, dependendo da arquitetura do sistema, o que significa grande volume de entrada e de saída de dados. Nesse caso, as operações que envolvem o microprocessador são simples, mas muitas ações dependem da entrada e da saída de dados.

Esses dois exemplos ilustram duas situações possíveis do uso de um computador:

- processos que são limitados pela capacidade de processamento da máquina (CPU-bound);
- processos que são limitados pela capacidade de entrada e saída da máquina (IO-bound).

Alguns processos são limitados por entrada e saída de dados e outros são limitados pela capacidade de processamento. Programas que estão parados esperando pela entrada ou pelas saídas de dados não precisam bloquear a CPU. Ela pode estar livre para executar instruções de processos limitados por processamento e com pouca entrada e saída. Dessa forma, um sistema operacional multitarefa roda em uma máquina com um único microprocessador, que, por sua vez, tem um único núcleo de processamento e pode executar processos em paralelo, visto que esses processos requerem recursos de natureza diferente.

É possível que, se tivéssemos os processos CPU-bound e IO-bound executando em paralelo, seus tempos de execução poderiam não se somar completamente. A CPU pode trabalhar em um processo enquanto aguarda dados do outro processo, mas existe uma penalidade ao mudar de um processo para outro, devido ao tempo de mudança de contexto, o que significa que o tempo de execução não é totalmente igual ao tempo de paralelismo perfeito, o tempo do processo mais longo. Isso é percebido por um pequeno acréscimo no tempo de execução simultânea dos processos.

2. Análise das afirmativas

I – Afirmativa correta.

JUSTIFICATIVA. É possível que o tempo total seja 306 uts (unidades de tempo) se pensarmos no seguinte caso: suponha que todas as operações de entrada e saída de dados (IO-bound) sejam executadas de forma serial, com tempo de execução igual a 50ut+100ut+150ut=300ut. Durante o tempo de espera pelas operações de entrada e saída,

o processador pode executar as operações CPU-bound, mesmo que essas operações fossem executadas de forma serial, o que levaria no máximo 20ut+40ut+60ut=120ut, tempo muito menor do que o das operações IO-bound. O valor total de 306ut, maior do que 300ut, é devido ao tempo adicional das mudanças de contexto.

II – Afirmativa incorreta.
JUSTIFICATIVA. O tempo total do grupo 1, correspondente aos processos CPU-bound, deve ser de, no mínimo, 20ut+40ut+60ut=120ut, uma vez que a CPU tem apenas um único núcleo e provavelmente não seja capaz de executar mais de uma operação simultaneamente. Contudo, é importante observar que algumas CPUs especiais, mesmo sendo compostas de apenas um núcleo, podem ser capazes de executar instruções de forma paralela.

III – Afirmativa incorreta.
JUSTIFICATIVA. O tempo do grupo 2 pode ser menor do que 300ut, dependendo da forma como é feita a entrada e a saída dos dados. Por exemplo, é comum que discos rígidos tenham memórias internas e que a leitura de dados normalmente seja feita em blocos. Dessa forma, cada busca retorna uma quantidade maior de dados do que aquilo que foi estritamente pedido. Os dados adicionais são altamente prováveis de serem requisitados posteriormente e ficam armazenados na memória do disco. A leitura dessa memória é muito mais rápida do que a leitura mecânica. Assim, o tempo de 50ut+100ut+150ut=300ut corresponderia ao pior caso possível e é provável que o tempo real seja bem inferior.

IV – Afirmativa correta.
JUSTIFICATIVA. O grupo 3 corresponde a uma mistura de processos IO-bound e um processo CPU-bound. Podemos imaginar que o processo P03, que leva a 60ut, executa de forma paralela aos processos P05 e P06. O tempo de execução do processo P06 é de 150ut, mas devemos ter em mente que existe sobrecarga na execução de três processos simultâneos (especialmente se pensarmos que dois deles dependem de recursos de entrada e saída). Assim, é razoável pensarmos que o tempo de execução do conjunto seja superior a 150ut (175ut, como indicado no enunciado). Porém o tempo de execução desses processos deve ser inferior a 310ut, que seria maior do que o pior tempo possível de execução de todos os processos simultaneamente.

Alternativa correta: C.

3. Indicações bibliográficas

- MACHADO, F. B.; MAIA, L. P. *Arquitetura de sistemas operacionais.* 5. ed. Rio de Janeiro: LTC, 2013.
- SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Fundamentos de sistemas operacionais – Princípios básicos.* Rio de Janeiro: LTC, 2013.
- TANENBAUM, A. D. *Sistemas operacionais modernos*. 3. ed. São Paulo: Pearson Prentice Hall, 2010.

Questões 7, 8 e 9

Questão 7.[7]

Considere que uma empresa que presta serviços de transporte de pacientes em ambulâncias para clientes conveniados disponha de um sistema de controle e gerenciamento de atendimentos e viagens realizados. Considere, ainda, que, em atendimento, é utilizada uma ambulância e são registrados a data e o convênio a que o atendimento está vinculado. Em um atendimento, uma ambulância realiza uma ou mais viagens e, a cada viagem, é incrementado um número sequencial que começa em 1. Nessa situação, o esquema relacional simplificado, mostrado a seguir, foi projetado para suportar um banco de dados que controle a operação. No esquema, as chaves primárias têm seus atributos componentes sublinhados.

Paciente (CodPaciente, Nome, Endereço)

Convênio (CodConvênio, Empresa, Plano)

Atendimento (CodAtendimento, CodPaciente, CodConvênio, Data, Finalidade)

Viagem (CodAtendimento, Sequência, Origem, Destino)

Com base nas informações e no esquema apresentado, avalie as afirmações a seguir.

I. CodConvênio é uma chave estrangeira na tabela Atendimento.
II. CodAtendimento não pode ser chave estrangeira na tabela Viagem porque faz parte da chave primária.
III. CodPaciente nunca pode assumir valores nulos na tabela Atendimento porque é uma chave estrangeira.
IV. CodPaciente->Nome, Endereço; CodConvênio->Empresa, Plano; CodAtendimento->CodPaciente,CodConvênio, Data, Finalidade; CodAtendimento, Sequência->Origem, Destino; são dependências funcionais corretamente deduzidas.

É correto apenas o que se afirma em

A. I.
B. II.
C. I e IV.
D. II e III.
E. III e IV.

[7]Questão 30 – Enade 2014.

Questão 8.[8]

Um sistema de informação usa um banco de dados relacional que possui tabelas cujos esquemas em SQL estão representados a seguir.

```
CREATE TABLE Artista
(
id INTEGER PRIMARY KEY,
nome VARCHAR(40) NOT NULL,
CPF CHAR(11) NOT NULL,
dataNascimento DATE,
UNIQUE (CPF)
);
CREATE TABLE Evento
(
id INTEGER PRIMARY KEY,
descricao VARCHAR(60) NOT NULL,
numMaxConvidados INTEGER DEFAULT 0,
CHECK (numMaxConvidados >= 0)
);
CREATE TABLE Atuacao
(
idArtista INTEGER,
idEvento INTEGER,
PRIMARY KEY (idArtista, idEvento),
FOREIGN KEY (idArtista) REFERENCES Artista,
FOREIGN KEY (idEvento) REFERENCES Evento(id)
);
```

O sistema também possui uma consulta que integra um de seus relatórios, conforme indicado a seguir.

[8]Questão 31 – Enade 2014.

```
SELECT A.nome, E.descricao
FROM Evento E FULL JOIN Atuacao T ON E.id = T.idEvento
FULL OUTER JOIN Artista A ON T.idArtista = A.id
```

Considerando que todas as tabelas possuem dados, o resultado da consulta utilizada no relatório é

A. o nome de todos os artistas combinados com a descrição de todos os eventos.
B. a descrição de todos os eventos e, caso haja artistas alocados, os seus nomes.
C. o nome de todos os artistas e a descrição de todos os eventos em que eles atuam.
D. o nome de todos os artistas e, caso eles participem de eventos, a descrição do evento.
E. o nome de todos os artistas, a descrição de todos os eventos e, caso eles se relacionem, os dois combinados.

Questão 9.[9]

Os sistemas de informação podem ser classificados em sistemas OLTP (Online Transaction Processing) e sistemas OLAP (Online Analytical Processing).

A respeito desses tipos de sistemas, avalie as asserções a seguir e a relação proposta entre elas.

I. Em sistemas OLTP, os dados são normalmente estruturados em um modelo relacional normalizado, e em sistemas OLAP, os dados são normalmente estruturados em um modelo multidimensional; contudo, esses sistemas estão inter-relacionados.

PORQUE

II. Os sistemas OLAP, na maioria das vezes, obtêm dados a partir de processos ETL (*Extract, Transform, Load*), que recuperam e transformam os dados obtidos a partir das bases de dados dos sistemas OLTP.

A respeito dessas asserções, assinale a opção correta.

A. As asserções I e II são proposições verdadeiras, e a asserção II justifica a I.
B. As asserções I e II são proposições verdadeiras, e a asserção II não justifica a I.
C. A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.
D. A asserção I é uma proposição falsa, e a II é uma proposição verdadeira.
E. As asserções I e II são proposições falsas.

[9]Questão 15 – Enade 2014.

1. Introdução teórica

1.1. Banco de dados relacionais

Frequentemente, problemas na área de computação lidam com grande quantidade de dados, gerados de forma artificial por um programa, coletados do mundo externo (por exemplo, por meio de cadastros ou de instrumentos conectados ao computador) ou frutos de interação do usuário com o computador. Muitas vezes, queremos que esses dados sejam armazenados de forma permanente ou por um período de tempo potencialmente longo.

Podemos armazenar dados de várias formas. Por exemplo, um simples arquivo de texto gravado em um pen drive é uma forma válida de armazenamento de dados. Porém, conforme a quantidade de dados armazenados aumenta, pode surgir uma série de problemas, como encontrar dados específicos em meio a uma grande quantidade de dados disponíveis, relacionar os dados entre si e atualizar e remover dados.

Com o aumento da capacidade de armazenamento dos computadores e com a sua popularização, várias técnicas de organização e de gerenciamento de dados foram desenvolvidas. Uma forma bastante importante no desenvolvimento de grandes sistemas é a dos bancos de dados relacionais.

Seguindo a abordagem de Ritchie (2002), podemos dizer, de forma simplificada, que os bancos de dados relacionais são uma coleção de tabelas interligadas.

As tabelas são formalmente chamadas de relações e cada uma de suas linhas (que são fisicamente registros da tabela) é chamada de tupla. Cada tabela tem várias colunas, chamadas de atributos. Como exemplo, considere a tabela 1 a seguir.

Tabela 1. Exemplo de uma tabela em um banco de dados.

Compositor	
Nome	**Gênero**
Ludwig van Beethoven	Clássico
Johann Sebastian Bach	Barroco
John Coltrane	Jazz
Miles Davis	Jazz
Antônio Carlos Jobim	MPB

Na tabela 1, "Compositor", os atributos (colunas) são "Nome" e "Gênero". Uma tupla dessa tabela seria: "Miles Davis; Jazz".

O diagrama de entidades e relacionamento apresenta uma notação especial para o projeto de banco de dados. Nesses diagramas, existem três elementos básicos: tipos de entidade, relacionamentos e atributos.

Em particular, uma entidade é um membro ou uma instância de um tipo de entidade. Os atributos são as propriedades dos tipos de entidade e os relacionamentos são as associações entre os tipos de entidades (MANNINO, 2008).

Cada linha de uma tabela deve ter uma combinação única de elementos, sem linhas repetidas. Isso significa que a linha toda deve ser única, mas não quer dizer que não exista a possibilidade de que alguns valores individuais não se repitam. Por exemplo, em uma tabela contendo o nome, o CPF e o salário de cada um dos funcionários de uma empresa, pode haver mais de um funcionário com o mesmo nome e salário (mas nunca com o mesmo CPF, que é único para cada indivíduo). Logo, cada linha dessa tabela é única.

Um conjunto de colunas com uma combinação única é chamado de superchave. Se pudermos remover elementos da superchave até o ponto em que temos uma combinação mínima de colunas, essa superchave mínima é uma chave candidata. Quando uma chave candidata é especialmente designada para uma tabela, dizemos que ela é uma chave primária.

Para ser possível referenciar tabelas diferentes, pode-se adicionar a chave primária de uma tabela em outra tabela, o que permite a seleção de um registro específico de uma tabela a partir de outra; a isso dá-se o nome de chave estrangeira (CORONEL e MORRIS, 2016; MANNINO, 2008).

Em um diagrama de entidade-relacionamento, existe a possibilidade de que uma entidade dependa de outra. Essa entidade pode precisar que parte da sua chave primária venha de outros tipos de entidades (ou até mesmo toda a chave). Nesse caso, dizemos que ela é uma entidade fraca (MANNINO, 2008).

Com o objetivo de garantirmos a integridade das entidades e dos dados armazenados em um banco de dados, introduzimos diversos tipos de restrições, especialmente ligados a cada tipo de integridade.

Assim, se quisermos garantir a integridade referencial, a chave estrangeira de uma tabela deve conter os valores vindos da chave primária da tabela pai (ou da chave candidata) ou o valor nulo (CORONEL e MORRIS, 2016).

A integridade semântica é a apresentação de um valor consistente com o tipo de dado de cada atributo/coluna. A integridade da entidade é a chave primária única não nula de cada tupla (CONRAD, MISENAR e FELDMAN, 2012).

1.2. Consultas e linguagem SQL

Para simplificar o desenvolvimento de programas que utilizam bancos de dados, a linguagem SQL (Structured Query Language) foi desenvolvida pela IBM, com base em um paradigma declarativo, em vez de um paradigma imperativo. Por exemplo, a linguagem C utiliza um paradigma imperativo, enquanto a linguagem Lisp utiliza um paradigma funcional e declarativo.

Dessa forma, o programador deve concentrar-se na busca pelos registros desejados do banco, e não na forma como esses registros são encontrados no banco de dados. Os registros lidos e armazenados são um problema do sistema de gerenciamento de banco de dados, e não da consulta. Além disso, a linguagem SQL foi desenvolvida especificamente para a utilização em banco de dados relacionais.

Após o seu desenvolvimento inicial, a linguagem SQL tornou-se um padrão, tendo sido revisada diversas vezes desde a década de 1980 até o início dos anos 2000. Ela é muito utilizada em aplicações que utilizam banco de dados e é amplamente suportada pela maioria dos sistemas de gerenciamento de banco de dados.

Ainda que a maioria dos fornecedores empregue uma versão similar da linguagem SQL, frequentemente, há algumas diferenças de comandos e de funcionalidades que variam de fornecedor para fornecedor. O desenvolvedor deve estar atento para a implementação específica que está sendo utilizada, o que significa sempre consultar a documentação do sistema de gerenciamento de banco de dados em uso.

Segundo Silberschatz, Korth e Sudarshan (2011), a linguagem SQL pode ser dividida nas partes a seguir.

- Data definition language (DDL).
- Data manipulation language (DML).
- Data query language (DQL).
- Comandos de integridade de dados.
- Definições de visões (views).
- Controle de transações.
- SQL embutido (embedded SQL).
- SQL dinâmico (dynamic SQL).

- Autorização.

Cada uma dessas partes tem um conjunto de expressões e comandos característicos, voltados para uma finalidade específica. Por exemplo, a DQL permite a consulta ao banco de dados e dispõe do comando SELECT, que seleciona um grupo de colunas em uma ou mais tabelas, baseada em determinado critério de seleção.

1.2.1. Comando SELECT

O comando SELECT permite a consulta ao conteúdo de uma tabela ou de um conjunto de tabelas de um banco de dados.

Segundo Silberschatz, Korth e Sudarshan (2011), a estrutura básica de uma consulta a um banco de dados, quando utilizada a linguagem SQL, é composta por três cláusulas: SELECT, FROM e WHERE.

De forma simplificada, podemos dizer que o comando seleciona um conjunto de "colunas" especificadas pela cláusula SELECT nas tabelas (ou relações) especificadas pela cláusula FROM, seguindo as "condições" impostas pela cláusula WHERE. Por exemplo, suponha uma tabela (relação) chamada "Compositor", que contenha os atributos nome e gênero, conforme mostrado na tabela 1. Se quisermos selecionar todos os nomes dos compositores dessa tabela, basta fazermos a seguinte consulta:

```
SELECT Nome FROM Compositor;
```

Obtemos, como resultado, a relação da tabela 2.

Tabela 2. Resultado da consulta SELECT Nome FROM Compositor.

Ludwig van Beethoven
Johann Sebastian Bach
John Coltrane
Miles Davis
Antônio Carlos Jobim

Com a cláusula WHERE, podemos adicionar condições à consulta. Por exemplo, se o gênero for apenas Jazz, temos o que segue.

```
SELECT Nome FROM Compositor WHERE Gênero = 'Jazz';
```

Nesse caso, obtemos a tabela 3 a seguir.

Tabela 3. Resultado da consulta SELECT Nome FROM Compositor WHERE Gênero = 'Jazz'.

John Coltrane
Miles Davis

1.2.2. FULL (OUTER) JOINs

Em algumas ocasiões, pode ser necessário selecionar todos os registros de várias tabelas simultaneamente. Como exemplo dessa situação, suponha que se queira mesclar duas tabelas, combinando os elementos vindos de ambas as tabelas envolvidas, mesmo quando não houver uma correspondência direta entre os campos da consulta. Nesse caso, utiliza-se o FULL OUTER JOIN (que pode ser abreviado para FULL JOIN).

Quando existe a necessidade de uma correspondência entre campos, utiliza-se o INNER JOIN (VIEIRA, 2009).

1.3. OLTP vs OLAP

Segundo Powell (2006), dizemos que um banco de dados OLTP (*Online Transaction Processing*) é uma "arquitetura de banco de dados altamente concorrente e otimizada para o acesso de pequenas quantidades de dados".

Dessa forma, um modelo de banco de dados transacional é comumente utilizado para processar grande quantidade de transações, muitas delas simultâneas e com acesso a um conjunto compartilhado de dados. Cada transação utiliza apenas um pequeno fragmento dos dados armazenados (POWELL, 2006). As empresas e corporações interagem com esses sistemas típicos no dia a dia (operações de cadastro, compra, venda, entre outras).

Além dos sistemas para operação do dia a dia das empresas, também são necessários sistemas que auxiliem na avaliação e no gerenciamento em um nível mais estratégico. Esses sistemas coletam dados de diversas operações da empresa nos seus diversos setores. Isso faz com que esse sistema lide com elevada quantidade de dados, frequentemente sumarizados em diversos tipos de relatórios.

Antes de falarmos sobre sistemas OLAP, é preciso definirmos um conceito relacionado, o de data warehouse.

Segundo Inmon (2005), um data warehouse é uma "coleção de dados orientada a assuntos, integrados, não voláteis e variáveis no tempo, para apoiar a tomada de decisões gerenciais". No data warehouse, rodam aplicações que trabalham com grande quantidade de dados e que, normalmente, geram grande variedade de relatórios (POWELL, 2006).

Em função dessas características e especialmente para ter um desempenho satisfatório, soluções de data warehouse devem ser implementadas adequadamente por banco de dados que utilize um modelo não normalizado, voltado para a análise de grande volume de informações. Os sistemas que trabalham e analisam os dados do data warehouse são chamados de sistemas OLAP (Online Analytical Processing) e costumam utilizar um modelo multidimensional; deve ser notado que existem outras definições de OLAP e de data warehouse. Na tabela 4, retirada de Jürgens (2003), temos um resumo das principais diferenças entre sistemas OLTP e OLAP.

Tabela 4. Diferenças entre OLTP e OLAP.

Característica	**OLTP**	**OLAP**
Nível de detalhamento	Detalhado	Agregado
Quantidade de dados por transação	Pequeno	Grande
Views	Pré-definidas	Definidas pelo usuário
Operações típicas de escrita	*Update, insert, delete*	Inserção de grandes volumes (*bulk insert* ou carga)
Idade dos dados armazenados	Atuais (60-90 dias)	Históricos, atuais, previsões, 5-10 anos
Modelos das Tabelas	Relacional (tipicamente)	Multidimensional
Número de usuários	Elevado	Poucos/Moderado
Tamanho do banco de dados	Médio(tipicamente)	Grande

Fonte. JÜRGENS, 2003 (com adaptações).

1.3.1. Operações de ETL

A maioria dos dados em um sistema OLAP veio, direta ou indiretamente, de um sistema OLTP. Dados de diversos sistemas devem ser manipulados, analisados e integrados de forma a permitir a geração das tabelas em um modelo multidimensional. Além disso, os sistemas OLAP são tipicamente utilizados pela alta gerência de uma empresa para analisar seu desempenho, prever o comportamento do mercado e auxiliar na tomada de decisões.

Os dados utilizados vêm de vários outros sistemas e, nesse processo, precisam ser manipulados. O processo de extração, manipulação e carga de dados para um sistema OLAP é chamado de ETL (*extract, transform, load*).

2. Análise das afirmativas e das asserções

Questão 7.

I – Afirmativa correta.
JUSTIFICATIVA. CodConvênio é a chave primária da tabela Convênio e foi adicionada à tabela Atendimento para ser possível relacionar as duas tabelas, que é o objetivo de uma chave estrangeira.

II – Afirmativa incorreta.
JUSTIFICATIVA. CodAtendimento é tanto uma chave primária na tabela Viagem quanto uma chave estrangeira para a tabela Atendimento.

III – Afirmativa incorreta.
JUSTIFICATIVA. É possível que o campo CodPaciente assuma o valor nulo na tabela Atendimento. Apenas a chave primária CodAtendimento não pode assumir o valor nulo.

IV – Afirmativa correta.
JUSTIFICATIVA. Todas as dependências ilustradas são de natureza funcional. Por exemplo, um paciente deve ter um nome e um endereço (ainda que o endereço seja um termo muito genérico). Um convênio envolve uma dependência entre uma empresa e um plano. Um atendimento envolve um paciente, um convênio, uma data e uma finalidade. Um atendimento, com um número de sequência, tem uma origem e um destino.
Alternativa correta: C.

Questão 8.

O SELECT seleciona o campo NOME da tabela Artista e o campo descrição da tabela Evento. Como ambos os JOINs são do tipo OUTER, todos os registros de ambas as tabelas

são retornados, mas, caso exista uma relação, vinda da tabela evento, os nomes e os artistas são selecionados de forma combinada com o evento.

Alternativa correta: E.

Questão 9.

I – Asserção verdadeira.
JUSTIFICATIVA. Um sistema OLTP deve ter excelente desempenho e alta disponibilidade. Por lidar com grande volume de transações e de usuários simultâneos, esse sistema deve ser altamente otimizado. Para isso, é importante que transações lidem com a menor quantidade possível de dados, que o armazenamento não seja redundante e que haja integridade referencial. Dessa forma, esses sistemas utilizam um modelo relacional normalizado. No entanto, sistemas OLAP, utilizados para a análise de grande volume de dados, costumam utilizar o modelo multidimensional.

II – Asserção verdadeira.
JUSTIFICATIVA. Boa parte dos dados de um sistema OLAP vem dos sistemas OLTP, pois eles contêm as informações do dia a dia das empresas. Contudo, esses dados precisam ser integrados e analisados de forma a permitir a geração do modelo multidimensional. Esses dados devem ser extraídos e carregados dos diversos sistemas da empresa. Muitas vezes, eles podem conter diversos tipos de problemas e várias imprecisões. Depois de uma etapa intermediária de manipulação, os dados devem ser carregados no sistema OLAP. Os processos de ETL (Extract, Transform and Load) integram e carregam os dados das diversas fontes no *data warehouse*.

Relação entre as asserções: ambas as asserções são verdadeira, e a II justificativa a I.

Alternativa correta: A.

3. Indicações bibliográficas

- CONRAD, E.; MISENAR, S.; FELDMAN, J. *CISSP Study Guide*. Waltham: Newnes, 2012.

- CORONEL, C.; MORRIS, S. *Database systems: design, implementation and management.* 12. ed. Boston: Cengage Learning, 2016.
- INMON, W.H. *Building the data warehouse*. Indianapolis: Wiley, 2005.
- JÜRGENS, M. Index Structures for data warehouses. Berlin: Springer, 2003.
- MANNINO, M. V. *Projeto, desenvolvimento de aplicações e administração de Banco de Dados.* 3. ed. Porto Alegre: AMGH, 2008.
- POWELL, G. *Beginning database design*. Indianapolis: John Wiley & Sons, 2006
- RITCHIE, C. *Relational database principles.* London: Thomson Learning, 2002.
- SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. *Database system concepts.* 6. ed. New York: McGraw-Hill, 2011.
- VIEIRA, R. *Beginning Microsoft SQL server 2008 programming*. Indianapolis: John Wiley & Sons, 2009.

Questão 10

Questão 10.[10]

As transações eletrônicas na Internet precisam de mecanismos de segurança capazes de garantir autenticidade, confidencialidade e integridade das informações.

Com relação a esse contexto, avalie as afirmativas a seguir.

I. Criptografia assimétrica é um método em que é utilizado um par de chaves: uma pública e uma privada.

II. Certificado digital é um documento eletrônico assinado digitalmente que permite associar uma pessoa ou entidade a uma chave pública.

III. Assinatura digital é um método de autenticação de informação digital tipicamente tratado como análogo à assinatura física em papel.

IV. VPN (Virtual Private Network) é um dispositivo de uma rede de computadores por meio do qual se aplica uma política de segurança a determinado ponto da rede.

É correto apenas o que se afirma em

A. I e II.

B. I e IV.

C. III e IV.

D. I, II e III.

E. II, III e IV.

1. Introdução teórica

1.1. Conceitos de segurança da informação

1.1.1. Criptografia

Frequentemente, é necessária a troca de informações por meios inseguros, o que significa que o tráfego de informação pode ser interceptado. Nesses casos, não queremos que um terceiro seja capaz de compreender o conteúdo da mensagem, mesmo tendo acesso aos bits transmitidos. Para isso, é feita a criptografia, que pode ser encarada, de forma extremamente simplificada, como o embaralhamento da mensagem a ser transmitida de forma a impedir que um terceiro entenda o seu conteúdo, mesmo tendo sido capaz de interceptar a sua transmissão.

[10]Questão 22 – Enade 2014.

Formatos primitivos de criptografia utilizavam apenas técnicas que dependiam de um algoritmo fixo para a construção da mensagem criptografada. Dessa forma, qualquer pessoa que soubesse o funcionamento do algoritmo seria capaz de decifrar o conteúdo da mensagem. Nessas circunstâncias, é fundamental o sigilo do algoritmo: a única forma de prevenir que uma pessoa não autorizada seja capaz de ler a mensagem é o desconhecimento do processo de criptografia.

Essa abordagem sofre de vários problemas. Como o sigilo do algoritmo é fundamental, cada grupo de pessoas que precisa se comunicar de forma privativa deve ter um algoritmo em particular. Como esse algoritmo deve ser secreto, ele não pode ser analisado pela comunidade acadêmica (a sua divulgação tornaria o algoritmo inútil).

Além disso, se uma das pessoas do grupo de comunicação tiver de ser excluída por algum motivo, como demissão, o processo perde o seu valor e um novo algoritmo deve ser desenvolvido.

1.1.2. Criptografia com chave (simétrica e assimétrica)

Para resolver o problema lustrado, foi criado o sistema de criptografia utilizando chaves. Podemos observar, na figura 1, o diagrama simplificado de um sistema de comunicação que utiliza criptografia sem chaves.

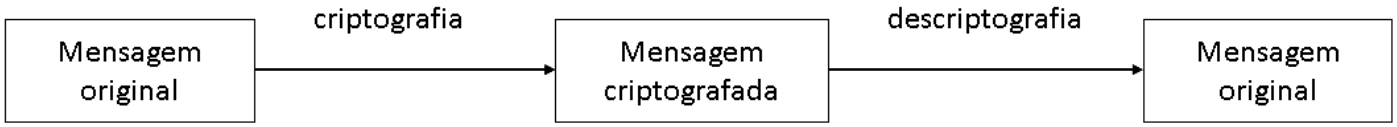


Figura 1. Processo de criptografia e descriptografia sem chaves.
Fonte. KOŚCIELNY, KURKOWSKI e SREBRNY, 2013 (com adaptações).

Na figura 1, a mensagem original deve ser criptografada por um algoritmo, transmitida e, finalmente, descriptografada por outro algoritmo no receptor. Podemos melhorar consideravelmente a segurança do processo se incorporarmos a ele uma chave: uma cadeia de bits que é utilizada para a encriptação e descriptação da mensagem. Dessa forma, tanto quem transmite a informação quanto quem a recebe necessita conhecer não apenas o algoritmo de criptografia, mas também a chave. Quando utilizamos a mesma chave no processo de encriptação e no de descriptação, dizemos que trabalhamos com criptografia simétrica, conforme figura 2.

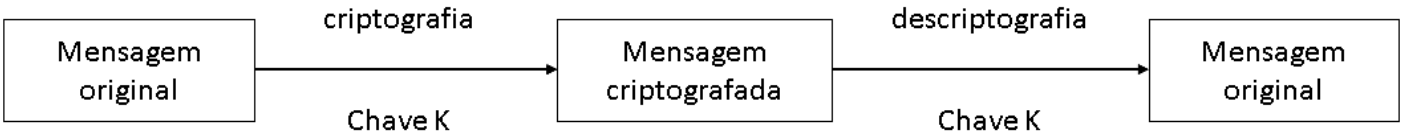


Figura 2. Processo de criptografia e descriptografia simétrica.
Fonte. KOŚCIELNY, KURKOWSKI e SREBRNY, 2013 (com adaptações).

É possível modificar o esquema da criptografia simétrica com o uso de chaves diferentes para a encriptação e para a descriptação, como ilustrado na figura 3. Em muitos casos, a chave K_1 é divulgada e, por isso, é chamada de chave pública. A chave K_2 deve ser conhecida apenas pela pessoa que vai ler a mensagem, sendo chamada de chave privada.

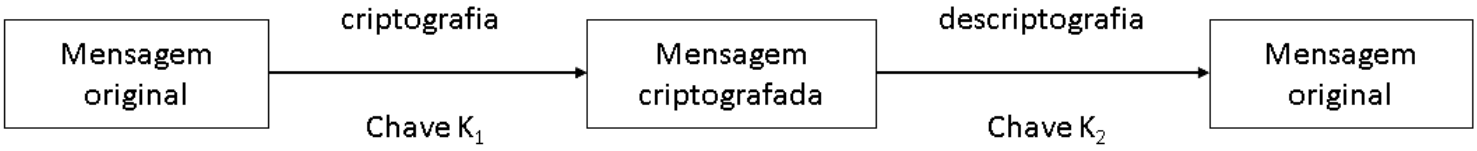


Figura 3. Processo de criptografia e descriptografia assimétrica.
Fonte. KOŚCIELNY, KURKOWSKI e SREBRNY, 2013 (com adaptações).

Tanto no processo de criptografia simétrica quanto no processo de criptografia assimétrica, é fundamental que a chave (privada, no segundo caso) seja guardada de forma segura. No entanto, o algoritmo pode ser totalmente conhecido, ser avaliado pela comunidade acadêmica e ser implementado por empresas terceiras. Uma vez que as chaves não sejam conhecidas, apenas o conhecimento do algoritmo não afeta a qualidade e a segurança do processo de criptografia.

Para provar que determinada pessoa é a "dona" de uma chave pública, utiliza-se um certificado digital. Nesse caso, uma terceira entidade tem responsabilidade legal em associar uma chave com uma pessoa (ou uma empresa) e emitir o certificado digital.

1.1.3. Assinaturas digitais

O mesmo esquema apresentado anteriormente pode ser utilizado para resolver um outro tipo de problema: a autenticidade de documentos. Em vez de "embaralharmos" a mensagem, para que ninguém (a não ser o destinatário) seja capaz de receber a informação, pode-se querer verificar a autenticidade da comunicação.

Uma vez que a mensagem original seja assinada na sua origem (com o uso de uma chave privada), a autenticidade e a integridade podem ser checadas com o uso de uma

chave pública (KATZ, 2010). Dessa forma, o destinatário pode checar se a mensagem recebida é autêntica e se não foi adulterada de alguma forma por um terceiro.

1.2. VPNs (Virtual Private Networks)

Originalmente, redes de computadores funcionavam de forma isolada, não havendo comunicação entre várias redes diferentes. Elas eram construídas em organizações para interligar um conjunto de máquinas, todas pertencentes à mesma empresa ou ao mesmo departamento.

Conforme o uso de computadores e redes ampliava-se na empresa, muitas filiais passavam a ter a sua própria rede privada. Essas redes eram verdadeiras "ilhas de comunicação", interligando máquinas de um conjunto restrito e local (SCOTT, WOLFE e ERWIN, 1999). Com o tempo, surgiu a necessidade de interligar essas redes, formando-se uma rede que cobrisse toda a empresa, mesmo que em lugares diferentes.

Outro tipo de rede bastante comum são as redes públicas. O exemplo mais utilizado é a internet. Esse tipo de rede interconecta livremente máquinas de diferentes usuários e empresas, tem a grande vantagem de abranger áreas grandes geográficas e permite a livre troca de informação entre diferentes indivíduos.

No entanto, isso esconde uma desvantagem: a informação que circula na rede pode ser facilmente interceptada por terceiros e há o risco de vazamento de informações confidenciais de pessoas jurídicas e de pessoas físicas.

A troca de mensagens criptografadas oferece a possibilidade de comunicação segura sob um meio compartilhado inseguro. Com isso, pode-se construir uma rede virtual que roda na rede real: os pacotes de informação são criptografados e distribuídos pela rede pública. Apenas as máquinas que possuírem a chave serão capazes de ler a informação original. Assim, torna-se possível a utilização das redes públicas de forma segura, mantendo-se a vantagem do seu baixo custo.

2. Análise das afirmativas

I – Afirmativa correta.

JUSTIFICATIVA. A assimetria da criptografia assimétrica é decorrente da diferença entre chaves públicas e chaves privadas. Obter a chave privada da chave pública é, ainda, muito custoso.

II – Afirmativa correta.
JUSTIFICATIVA. É importante saber a procedência da chave pública e, para isso, utiliza-se o certificado digital.

III – Afirmativa correta.
JUSTIFICATIVA. O propósito da assinatura digital é garantir a autenticidade de uma mensagem.

IV – Afirmativa correta.
JUSTIFICATIVA. Uma VPN é uma rede de dados criptografados que são publicamente distribuídos por redes públicas.

Alternativa correta: D.

3. Indicações bibliográficas

- KATZ, J. *Digital signatures.* Heidelberg: Springer, 2010.
- KOŚCIELNY, C.; KURKOWSKI, M.; SREBRNY, M. *Modern Cryptography Primer: Theoretical Foundations and Practical Applications*. Heidelberg: Springer, 2013.
- SCOTT, C.; WOLFE, P.; ERWIN, M. *Virtual private networks*. Sebastopol: O'Reilly, 1999.

ÍNDICE REMISSIVO

Questão 1	Sistemas operacionais. Processos. Escalonador de processos.
Questão 2	Linguagens de programação. Linguagens compiladas e interpretadas. JIT.
Questão 3	Programação. Estrutura de dados estáticas. Arrays.
Questão 4	Linguagens de programação. Paradigmas de linguagens de programação. Linguagens orientadas a objetos.
Questão 5	Máquinas virtuais. Virtualização de servidores. Arquitetura de sistemas.
Questão 6	Sistemas operacionais. Processos. Processos IO-Bound e CPU-Bound.
Questão 7	Banco de dados. Modelo relacional. Chave primária. Chave estrangeira.
Questão 8	Banco de dados. Modelo relacional. Consultas SQL.
Questão 9	Banco de dados. OLAP e OLTP. Data warehouse.
Questão 10	Segurança da informação. Criptografia. Redes de computadores.