Sistemas Operacionais: Conceitos e Mecanismos

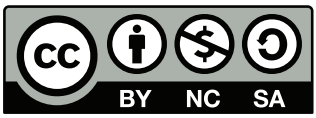
– Caderno de Exercícios –

Prof. Carlos A. Maziero, Dr.
DAINF – UTFPR

2 de maio de 2014

Sistemas Operacionais: Conceitos e Mecanismos

Sobre o autor: Carlos Alberto Maziero é professor adjunto do Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná (UTFPR) desde julho de 2011. Anteriormente, foi professor titular do Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná (PUCPR), entre 1998 e 2011, e professor adjunto do Departamento de Automação e Sistemas da Universidade Federal de Santa Catarina (UFSC), de 1996 a 1998. Formado em Engenharia Elétrica (UFSC, 1988), tem Mestrado em Engenharia Elétrica (UFSC, 1990), Doutorado em Informática (Université de Rennes I - França, 1994) e Pós-Doutorado em Segurança da Informação (Università degli Studi di Milano – Italia, 2009). Tem atuação em pesquisa nas áreas de sistemas operacionais, segurança de sistemas e sistemas distribuídos.



Este texto foi produzido usando exclusivamente software livre: Sistema Operacional *GNU/Linux* (distribuições *Fedora* e *Ubuntu*), compilador de texto LaTeX 2_ε, gerenciador de referências *BibTeX*, editor gráfico *Inkscape*, criadores de gráficos *GNUPlot* e *GraphViz* e processador PS/PDF *GhostScript*, entre outros.

Sumário

Capítulo 1

Conceitos básicos

1. Quais os dois principais objetivos dos sistemas operacionais?

2. Por que a abstração de recursos é importante para os desenvolvedores de aplicações? Ela tem utilidade para os desenvolvedores do próprio sistema operacional?

3. A gerência de atividades permite compartilhar o processador, executando mais de uma aplicação ao mesmo tempo. Identifique as principais vantagens trazidas por essa funcionalidade e os desafios a resolver para implementá-la.

4. O que caracteriza um sistema operacional de tempo real? Quais as duas classificações de sistemas operacionais de tempo real e suas diferenças?

5. O que diferencia o *núcleo* do restante do sistema operacional?

6. Seria possível construir um sistema operacional seguro usando um processador que não tenha níveis de privilégio? Por quê?

7. O processador Pentium possui dois bits para definir o nível de privilégio, resultando em 4 níveis distintos. A maioria dos sistemas operacionais para esse processador usa somente os níveis extremos (0 e 3, ou 00_2 e 11_2). Haveria alguma utilidade para os níveis intermediários?

8. Quais as diferenças entre *interrupções*, *exceções* e *traps*?

9. Quais as implicações de mascarar interrupções? O que pode ocorrer se o processador ignorar interrupções por muito tempo? O que poderia ser feito para evitar o mascaramento de interrupções?

10. O comando em linguagem C `fopen` é uma chamada de sistema ou uma função de biblioteca? Por quê?

11. Monte uma tabela com os benefícios e deficiências mais significativos das principais arquiteturas de sistemas operacionais.

12. Relacione as afirmações aos respectivos tipos de sistemas operacionais: distribuído (D), multi-usuário (M), desktop (K), servidor (S), embarcado (E) ou de tempo-real (T):

    [ ] Deve ter um comportamento temporal previsível, com prazos de resposta claramente definidos.

    [ ] Sistema operacional usado por uma empresa para executar seu banco de dados corporativo.

    [ ] São tipicamente usados em telefones celulares e sistemas eletrônicos dedicados.

    [ ] Neste tipo de sistema, a localização física dos recursos do sistema computacional é transparente para os usuários.

    [ ] Todos os recursos do sistema têm proprietários e existem regras controlando o acesso aos mesmos pelos usuários.

    [ ] A gerência de energia é muito importante neste tipo de sistema.

    [ ] Sistema que prioriza a gerência da interface gráfica e a interação com o usuário.

    [ ] Construído para gerenciar de forma eficiente grandes volumes de recursos.

    [ ] O MacOS X é um exemplo típico deste tipo de sistema.

    [ ] São sistemas operacionais compactos, construídos para executar aplicações específicas sobre plataformas com poucos recursos.

13. A operação em modo usuário permite ao processador executar somente parte das instruções disponíveis em seu conjunto de instruções. Quais das seguintes operações não deveriam ser permitidas em nível usuário? Por quê?

(a) Ler uma porta de entrada/saída
(b) Efetuar uma divisão inteira
(c) Escrever um valor em uma posição de memória
(d) Ajustar o valor do relógio do hardware
(e) Ler o valor dos registradores do processador
(f) Mascarar uma ou mais interrupções

14. Considerando um processo em um sistema operacional com proteção de memória entre o núcleo e as aplicações, indique quais das seguintes ações do processo teriam de ser realizadas através de chamadas de sistema, justificando suas respostas:

(a) Ler o relógio de tempo real do hardware.
(b) Enviar um pacote através da rede.
(c) Calcular uma exponenciação.
(d) Preencher uma área de memória do processo com zeros.
(e) Remover um arquivo do disco.

15. Coloque na ordem correta as ações abaixo, que ocorrem durante a execução da função `printf("Hello world")` por um processo (observe que nem todas as ações indicadas fazem parte da seqüência).

[ ] A rotina de tratamento da interrupção de software é ativada dentro do núcleo.
[ ] A função `printf` finaliza sua execução e devolve o controle ao código do processo.
[ ] A função de biblioteca `printf` recebe e processa os parâmetros de entrada (a string "`Hello world`").
[ ] A função de biblioteca `printf` prepara os registradores para solicitar a chamada de sistema `write()`
[ ] O disco rígido gera uma interrupção indicando a conclusão da operação.
[ ] O escalonador escolhe o processo mais prioritário para execução.
[ ] Uma interrupção de software é acionada.

[ ] O processo chama a função `printf` da biblioteca C.

[ ] A operação de escrita no terminal é efetuada ou agendada pela rotina de tratamento da interrupção.

[ ] O controle volta para a função `printf` em modo usuário.

16. Considere as afirmações a seguir, relativas aos diversos tipos de sistemas operacionais:

    I. Em um sistema operacional de **tempo real**, a rapidez de resposta é menos importante que a previsibilidade do tempo de resposta.

    II. Um sistema operacional **multi-usuários** associa um proprietário a cada recurso do sistema e gerencia as permissões de acesso a esses recursos.

    III. Nos sistemas operacionais **de rede** a localização dos recursos é transparente para os usuários.

    IV. Um sistema operacional **de tempo real** deve priorizar as tarefas que interagem com o usuário.

    V. Um sistema operacional **embarcado** é projetado para operar em hardware com poucos recursos.

    Indique a alternativa correta:

    (a) As afirmações II e III estão corretas.

    (b) Apenas a afirmação V está correta.

    (c) As afirmações III e IV estão erradas.

    (d) As afirmações III, IV e V estão erradas.

    (e) Todas as afirmações estão erradas.

    Justifique as afirmações julgadas erradas (Assim: *VII está errada porque ...*):

17. Considere as afirmações a seguir, relativas às diversas arquiteturas de sistemas operacionais:

    I. Uma máquina virtual de sistema é contruída para suportar uma aplicação escrita em uma linguagem de programação específica, como Java.

II. Um hipervisor convidado executa sobre um sistema operacional hospedeiro.

III. Em um sistema operacional micro-núcleo, os diversos componentes do sistema são construídos como módulos interconectados executando dentro do núcleo.

IV. Núcleos monolíticos são muito utilizados devido à sua robustez e facilidade de manutenção.

V. Em um sistema operacional micro-núcleo, as chamadas de sistema são implementadas através de trocas de mensagens.

Indique a alternativa correta:

(a) Todas as afirmações estão erradas.

(b) As afirmações II e III estão corretas.

(c) As afirmações II e IV estão erradas.

(d) Apenas a afirmação V está correta.

(e) As afirmações II e V estão corretas.

Justifique as afirmações julgadas erradas (Assim: *VII está errada porque ...*):

18. O utilitário `strace` do UNIX permite observar a sequência de chamadas de sistema efetuadas por uma aplicação. Em um terminal UNIX, execute `strace date` para descobrir quais os arquivos abertos pela execução do utilitário `date` (que indica a data e hora correntes). Por que o utilitário `date` precisa fazer chamadas de sistema?

19. O utilitário `ltrace` do UNIX permite observar a sequência de chamadas de biblioteca efetuadas por uma aplicação. Em um terminal UNIX, execute `ltrace date` para descobrir as funções de biblioteca chamadas pela execução do utilitário `date` (que indica a data e hora correntes). Pode ser observada alguma relação entre as chamadas de biblioteca e as chamadas de sistema observadas no ítem anterior?

Capítulo 2

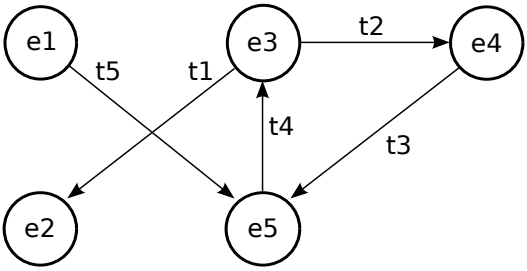
Gerência de atividades

1. Explique o que é, para que serve e o que contém um PCB - *Process Control Block*.

2. O que significa *time sharing* e qual a sua importância em um sistema operacional?

3. Como e com base em que critérios é escolhida a duração de um *quantum* de processamento?

4. Considerando o diagrama de estados dos processos apresentado na figura a seguir, **complete o diagrama** com a transição de estado que está faltando (t_6) e **apresente o significado** de cada um dos estados e transições.



5. Indique se cada uma das transições de estado de tarefas a seguir definidas é possível ou não. Se a transição for possível, dê um exemplo de situação na qual ela ocorre (*N*: Nova, *P*: pronta, *E*: executando, *S*: suspensa, *T*: terminada).

- $E \rightarrow P$
- $E \rightarrow S$
- $S \rightarrow E$
- $P \rightarrow N$
- $S \rightarrow T$
- $E \rightarrow T$
- $N \rightarrow S$
- $P \rightarrow S$

6. Relacione as afirmações abaixo aos respectivos estados no ciclo de vida das tarefas (N: Nova, P: Pronta, E: Executando, S: Suspensa, T: Terminada):

[ ] O código da tarefa está sendo carregado.
[ ] A tarefas são ordenadas por prioridades.
[ ] A tarefa sai deste estado ao solicitar uma operação de entrada/-saída.
[ ] Os recursos usados pela tarefa são devolvidos ao sistema.
[ ] A tarefa vai a este estado ao terminar seu *quantum*.
[ ] A tarefa só precisa do processador para poder executar.
[ ] O acesso a um semáforo em uso pode levar a tarefa a este estado.
[ ] A tarefa pode criar novas tarefas.
[ ] Há uma tarefa neste estado para cada processador do sistema.
[ ] A tarefa aguarda a ocorrência de um evento externo.

7. Desenhe o diagrama de tempo da execução do código a seguir, informe qual a saída do programa na tela (com os valores de x) e calcule a duração aproximada de sua execução.

```
1  int main()
2  {
3     int x = 0 ;
4
5     fork () ;
6     x++ ;
```

```
7     sleep (5) ;
8     wait (0) ;
9     fork () ;
10    wait (0) ;
11    sleep (5) ;
12    x++ ;
13    printf ("Valor de x: %d\n", x) ;
14 }
```

8. Indique quantas letras "X" serão impressas na tela pelo programa abaixo quando for executado com a seguinte linha de comando:

   ```
   a.out 4 3 2 1
   ```

   Observações:

   - `a.out` é o arquivo executável resultante da compilação do programa.
   - A chamada de sistema `fork` cria um processo filho, clone do processo que a executou, retornando o valor zero no processo filho e um valor diferente de zero no processo pai.

```
1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <unistd.h>
4  #include <stdlib.h>
5
6  int main(int argc, char *argv[])
7  {
8     pid_t pid[10];
9     int i;
10
11    int N = atoi(argv[argc-2]);
12
13    for (i=0; i<N; i++)
14       pid[i] = fork();
15    if (pid[0] != 0 && pid[N-1] != 0)
16       pid[N] = fork();
17    printf("X");
18    return 0;
19 }
```

9. O que são *threads* e para que servem?

10. Quais as principais vantagens e desvantagens de *threads* em relação a processos?

11. Forneça dois exemplos de problemas cuja implementação *multi-thread* não tem desempenho melhor que a respectiva implementação sequencial.

12. Associe as afirmações a seguir aos seguintes modelos de *threads*: a) *many-to-one* (N:1); b) *one-to-one* (1:1); c) *many-to-many* (N:M):

    [ ] Tem a implementação mais simples, leve e eficiente.
    [ ] Multiplexa os *threads* de usuário em um pool de *threads* de núcleo.
    [ ] Pode impor uma carga muito pesada ao núcleo.
    [ ] Não permite explorar a presença de várias CPUs pelo mesmo processo.
    [ ] Permite uma maior concorrência sem impor muita carga ao núcleo.
    [ ] Geralmente implementado por bibliotecas.
    [ ] É o modelo implementado no Windows NT e seus sucessores.
    [ ] Se um *thread* bloquear, todos os demais têm de esperar por ele.
    [ ] Cada *thread* no nível do usuário tem sua correspondente dentro do núcleo.
    [ ] É o modelo com implementação mais complexa.

13. Considerando as implementações de *threads* N:1 e 1:1 para o trecho de código a seguir, a) desenhe os diagramas de execução, b) informe as durações aproximadas de execução e c) indique a saída do programa na tela. Considere a operação `sleep()` como uma chamada de sistema (*syscall*).

    Significado das operações:

    - `thread_create`: cria uma nova *thread*, pronta para executar.
    - `thread_join`: espera o encerramento da *thread* informada como parâmetro.

- `thread_exit`: encerra a *thread* corrente.

```
1  int y = 0 ;
2
3  void threadBody
4  {
5     int x = 0 ;
6     sleep (10) ;
7     printf ("x: %d, y:%d\n", ++x, ++y) ;
8     thread_exit();
9  }
10
11 main ()
12 {
13    thread_create (&tA, threadBody, ...) ;
14    thread_create (&tB, threadBody, ...) ;
15    sleep (1) ;
16    thread_join (&tA) ;
17    thread_join (&tB) ;
18    sleep (1) ;
19    thread_create (&tC, threadBody, ...) ;
20    thread_join (&tC) ;
21 }
```

14. Explique o que é escalonamento *round-robin*, dando um exemplo.

15. Considere um sistema de tempo compartilhado com valor de quantum t_q e duração da troca de contexto t_{tc}. Considere tarefas de entrada/saída que usam em média p% de seu quantum de tempo cada vez que recebem o processador. Defina a eficiência $\mathcal{E}$ do sistema como uma função dos parâmetros t_q, t_{tc} e p.

16. Explique o que é, para que serve e como funciona a técnica de *aging*.

17. A tabela a seguir representa um conjunto de tarefas prontas para utilizar um processador:

Tarefa	t_1	t_2	t_3	t_4	t_5
ingresso	0	0	3	5	7
duração	5	4	5	6	4
prioridade	2	3	5	9	6

Indique a seqüência de execução das tarefas, o tempo médio de vida (*tournaround time*) e o tempo médio de espera (*waiting time*), para as políticas de escalonamento a seguir:

(a) FCFS cooperativa

(b) SJF cooperativa

(c) SJF preemptiva (SRTF)

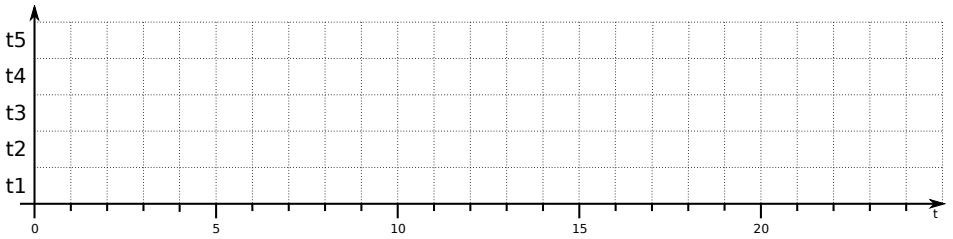
(d) PRIO cooperativa

(e) PRIO preemptiva

(f) RR com $t_q = 2$, sem envelhecimento

Considerações: todas as tarefas são orientadas a processamento; as trocas de contexto têm duração nula; em eventuais empates (idade, prioridade, duração, etc), a tarefa t_i com menor i prevalece; valores maiores de prioridade indicam maior prioridade.

Para representar a sequência de execução das tarefas use o diagrama a seguir. Use $\boxed{\times}$ para indicar uma tarefa usando o processador, $\boxed{-}$ para uma tarefa em espera na fila de prontos e $\boxed{\ }$ para uma tarefa que ainda não iniciou ou já concluiu sua execução.



18. Idem, para as tarefas da tabela a seguir:

Tarefa	t_1	t_2	t_3	t_4	t_5
ingresso	0	0	1	7	11
duração	5	6	2	6	4
prioridade	2	3	4	7	9

19. Explique os conceitos de *inversão* e *herança de prioridade*.

20. Você deve analisar o software da sonda *Mars Pathfinder* discutido no livro-texto. O sistema usa escalonamento por prioridades preemptivo, sem envelhecimento e sem compartilhamento de tempo. Suponha que as tarefas t_g e t_m detêm a área de transferência de dados durante todo o período em que executam. Os dados de um trecho de execução das tarefas são indicados na tabela a seguir (observe que t_g executa mais de uma vez).

Tarefa	t_g	t_m	t_c
ingresso	0, 5, 10	2	3
duração	1	2	10
prioridade	alta	baixa	média

Desenhe o diagrama de tempo da execução sem e com o protocolo de herança de prioridades e discuta sobre as diferenças observadas entre as duas execuções.

Capítulo 3

Comunicação entre tarefas

1. Quais são as vantagens e desvantagens das abordagens a seguir, sob as óticas do sistema operacional e do programador de aplicativos?

   (a) comunicação bloqueante ou não-bloqueante
   (b) canais com *buffering* ou sem *buffering*
   (c) comunicação por mensagens ou por fluxo
   (d) mensagens de tamanho fixo ou variável
   (e) comunicação 1:1 ou M:N

2. Explique como processos que comunicam por troca de mensagens se comportam em relação à capacidade do canal de comunicação, considerando as semânticas de chamada síncrona e assíncrona.

3. Em relação à sincronização na comunicação entre processos, podemos afirmar que:

   I. Na comunicação semi-bloqueante, o emissor espera indefinidamente pela possibilidade de enviar os dados.
   II. Na comunicação síncrona ou bloqueante, o receptor espera até receber a mensagem.
   III. Um mecanismo de comunicação semi-bloqueante com prazo $t = \infty$ equivale a um mecanismo bloqueante.
   IV. Na comunicação síncrona ou bloqueante, o emissor retorna uma mensagem de erro caso o receptor não esteja pronto para receber a mensagem.

V. A comunicação com semântica bloqueante usando canais sem *buffer* é chamada *Rendez-Vous*.

As asserções corretas são:

(a) I, III
(b) II, III, V
(c) I, II, IV
(d) II, III
(e) III, IV, V

Justifique as afirmações julgadas erradas (Assim: *VII está errada porque ...*):

4. Em relação à sincronização na comunicação entre processos, podemos afirmar que:

I. Na comunicação semi-bloqueante, o emissor espera pelo envio dos dados, mas o receptor não.
II. Se o canal de comunicação tiver capacidade nula, emissor e receptor devem usar mecanismos não-bloqueantes.
III. A comunicação não-bloqueante em ambos os participantes só é viável usando canais de comunicação com *buffer* não-nulo.
IV. Os *pipes* do UNIX são um bom exemplo de comunicação bloqueante.
V. Um mecanismo de comunicação semi-bloqueante com prazo $t = 0$ equivale a um mecanismo bloqueante.

As asserções corretas são:

(a) I, II, IV
(b) II, III
(c) III, IV, V
(d) I, IV
(e) III, IV

Justifique as afirmações julgadas erradas (Assim: *VII está errada porque ...*):

5. Dadas as seguintes características dos mecanismos de comunicação:

   I. A comunicação indireta (por canais) é mais adequada para sistemas distribuídos.

   II. Canais com capacidade finita somente são usados na definição de algoritmos, não sendo implementáveis na prática.

   III. Na comunicação direta, o emissor envia os dados diretamente a um canal de comunicação.

   IV. Na comunicação por fluxo, a ordem dos dados enviados pelo emissor é mantida do lado receptor.

   V. Na comunicação por troca de mensagens, o núcleo transfere pacotes de dados do processo emissor para o processo receptor.

   As asserções erradas são:

   (a) II, III
   (b) I, III
   (c) II, IV
   (d) III, V
   (e) I, IV

   Justifique as afirmações julgadas erradas (Assim: *VII está errada porque ...*):

6. Dadas as seguintes características dos mecanismos de comunicação:

   I. Na comunicação por troca de mensagens, o processo emissor copia o conteúdo da mensagem no *buffer* do processo receptor.

   II. O *buffer* do canal de comunicação entre dois processos distintos é geralmente mantido pelo núcleo do sistema operacional.

   III. Se a capacidade do *buffer* do canal de comunicação for considerada infinita, somente o receptor pode se bloquear.

IV. As filas de mensagens POSIX são um exemplo de canal de comunicação com capacidade nula.

V. O protocolo de rede TCP é um exemplo de comunicação por fluxo de dados.

As asserções erradas são:

(a) I, III

(b) II, III

(c) I, IV

(d) II, IV

(e) II, V

Justifique as afirmações julgadas erradas (Assim: *VII está errada porque ...*):

7. Dadas as seguintes características dos mecanismos de comunicação:

I. A memória compartilhada provê mecanismos de sincronização para facilitar a comunicação entre os processos.

II. A troca de dados através de memória compartilhada é mais adequada para a comunicação em rede.

III. Processos que se comunicam por memória compartilhada podem acessar a mesma área da RAM.

IV. Os pipes Unix são um bom exemplo de comunicação M:N.

V. A comunicação através de memória compartilhada é particularmente indicada para compartilhar grandes volumes de dados entre dois ou mais processos.

As asserções corretas são:

(a) I, III, V

(b) I, II

(c) III, IV

(d) II, IV

(e) III, V

Justifique as afirmações julgadas erradas (Assim: *VII está errada porque ...*):

8. Dadas as seguintes características dos mecanismos de comunicação:

   I. Em um mecanismo de *mailbox*, cada mensagem enviada é replicada a todos os receptores.
   II. Em um canal de eventos, as mensagens enviadas são distribuídas alternadamente entre os receptores.
   III. As filas de mensagens POSIX são um bom exemplo de canal de eventos.
   IV. Nas filas de mensagens POSIX, as mensagens transitam através de arquivos em disco criados especialmente para essa finalidade.
   V. Em UNIX, um *pipe* é um canal de comunicação unidirecional que liga a saída padrão de um processo à entrada padrão de outro.

   As asserções corretas são:

   (a) I, III
   (b) II
   (c) III, IV
   (d) V
   (e) nenhuma delas

   Justifique as afirmações julgadas erradas (Assim: *VII está errada porque ...*):

Capítulo 4

Coordenação entre tarefas

1. Explique o que são *condições de disputa*, mostrando um exemplo real.
2. Em relação aos mecanismos de coordenação:

   I. A estratégia de inibir interrupções para evitar condições de disputa funciona em sistemas multi-processados.

   II. Os mecanismos de controle de entrada nas regiões críticas provêem exclusão mútua no acesso às mesmas.

   III. Os algoritmos de *busy-wait* se baseiam no teste contínuo de uma condição.

   IV. Condições de disputa ocorrem devido às diferenças de velocidade na execução dos processos.

   V. Condições de disputa ocorrem quando dois processos tentam executar o mesmo código ao mesmo tempo.

   As asserções corretas são:

   (a) I, III
   (b) II, V
   (c) II, III
   (d) I, IV
   (e) IV, V

Justifique as afirmações julgadas erradas (Assim: *VII está errada porque ...*):

3. Explique o que é *espera ocupada* e por que os mecanismos que empregam essa técnica são considerados ineficientes.

4. Em que circunstâncias o uso de espera ocupada é inevitável?

5. Em relação aos mecanismos de coordenação:

   I. Instruções do tipo *Test&Set Lock* devem ser implementadas pelo núcleo do SO.
   II. O algoritmo de Peterson garante justiça no acesso à região crítica.
   III. Os algoritmos com estratégia *busy-wait* otimizam o uso da CPU do sistema.
   IV. Uma forma eficiente de resolver os problemas de condição de disputa é introduzir pequenos atrasos nos processos envolvidos.
   V. Um semáforo é composto por um contador inteiro e uma fila de processos suspensos.

   As asserções corretas são:

   (a) I, III
   (b) I, V
   (c) II, V
   (d) I, IV
   (e) III, IV

   Justifique as afirmações julgadas erradas (Assim: *VII está errada porque ...*):

6. Considere `ocupado` uma variável inteira compartilhada entre dois processos A e B (inicialmente, `ocupado = 0`). Sendo que ambos os processos executam o trecho de programa abaixo, explique em que situação A e B poderiam entrar simultaneamente nas suas respectivas regiões críticas.

```
while (true) {
       regiao_nao_critica();
       while (ocupado) {};
       ocupado = 1;
       regiao_critica();
       ocupado = 0;
}
```

7. Em que situações um semáforo deve ser inicializado em 0, 1 ou $n > 1$?

8. Por que não existem operações *read*(*s*) e *write*(*s*) para ler ou ajustar o valor corrente de um semáforo?

9. Mostre como pode ocorrer violação da condição de exclusão mútua se as operações *down*(*s*) e *up*(*s*) sobre semáforos não forem implementadas de forma atômica.

10. A implementação das operações *down*(*s*) e *up*(*s*) sobre semáforos deve ser atômica, para evitar condições de disputa sobre as variáveis internas do semáforo. Escreva, em pseudo-código, a implementação dessas duas operações, usando instruções TSL para evitar as condições de disputa. A estrutura interna do semáforo é indicada a seguir. Não é necessário detalhar as operações de ponteiros envolvendo a fila *task_queue*.

```
struct semaphore
{
   int lock = false ;
   int count ;
   task_t *queue ;
}
```

11. Usando semáforos, escreva o pseudo-código de um sistema produtor/consumidor com dois buffers limitados organizado na forma $X \rightarrow B_1 \rightarrow Y \rightarrow B_2 \rightarrow Z$, onde X, Y e Z são tipos de processos e B_1 e B_2 são buffers independentes com capacidades N_1 e N_2, respectivamente, inicialmente vazios. Os buffers são acessados unicamente através das operações *insere*(B_i, *item*) e *retira*(B_i, *item*) (que não precisam ser detalhadas). O número de processos X, Y e Z é desconhecido.

    Devem ser definidos os códigos dos processos X, Y e Z e os semáforos necessários, com seus significados e valores iniciais.

12. O trecho de código a seguir apresenta uma solução para o problema do jantar dos filósofos, mas ele contém um erro. Explique o código e explique onde está o erro e porque ele ocorre. A seguir, modifique o código para que ele funcione corretamente.

```
1  #define N 5
2
3  sem_t garfo[5] ; // 5 semáforos iniciados em 1
4
5  void filosofo (int i)
6  {
7     while (1)
8     {
9        medita ();
10       sem_down (garfo [i]) ;
11       sem_down (garfo [(i+1) % N]) ;
12       come ();
13       sem_up (garfo [i]) ;
14       sem_up (garfo [(i+1) % N]) ;
15    }
16 }
```

13. Suponha três robôs (*Bart*, *Lisa*, *Maggie*), cada um controlado por sua própria *thread*. Você deve escrever o código das *threads* de controle, usando semáforos para garantir que os robôs se movam sempre na sequência *Bart* → *Lisa* → *Maggie* → *Lisa* → *Bart* → *Lisa* → *Maggie* → ···, um robô de cada vez. Use a chamada `move()` para indicar um movimento do robô. Não esqueça de definir os valores iniciais das variáveis e/ou dos semáforos utilizados. Soluções envolvendo espera ocupada (*busy wait*) não devem ser usadas.

14. O *Rendez-Vous* é um operador de sincronização forte entre **dois** processos ou *threads*, no qual um deles espera até que ambos cheguem ao ponto de encontro (*rendez-vous*, em francês). O exemplo a seguir ilustra seu uso:

```
Processo A                    Processo B
   A1 () ;                       B1 () ;
   rv_wait (rv) ;                rv_wait (rv) ;
   A2 () ;                       B2 () ;
   rv_wait (rv) ;                rv_wait (rv) ;
   A3 () ;                       B3 () ;
```

Considerando a relação $a \rightarrow b$ como "a ocorre antes de b" e a relação $a \parallel b$ como "a e b ocorrem sem uma ordem definida", temos as seguintes restrições de sincronização:

- $\forall(i,j), A_i \rightarrow B_{j>i}$ e $B_i \rightarrow A_{j>i}$ (imposto pelo *Rendez-Vous*)
- $\forall(i,j), A_i \rightarrow A_{j>i}$ e $B_i \rightarrow B_{j>i}$ (imposto pela execução sequencial)
- $\forall(i,j), A_i \parallel B_{j=i}$ (possibilidade de execução concorrente)

Escreva o pseudo-código necessário para implementar *Rendez-Vous*, usando semáforos ou mutexes. Não esqueça de inicializar as variáveis e semáforos utilizados. Soluções que incorram em espera ocupada (*busy wait*) não serão aceitas.

```
1  // estrutura que representa um RV
2  typedef struct rv_t
3  {
4     ... // completar
5  } rv_t ;
6
7  // operador de espera no RV
8  void rv_wait (rv_t *rv)
9  {
10    ... // completar
11 }
12
13 // inicialização do RV
14 void rv_init (rv_t *rv)
15 {
16    ... // completar
17 }
```

15. Uma *Barreira* é um operador de sincronização forte entre **N** processos ou *threads*, no qual eles esperam até que todos cheguem à barreira. O exemplo a seguir ilustra seu uso:

```
Processo A
   A1 () ;
   barrier_wait (b) ;
   A2 () ;
   barrier_wait (b) ;
   A3 () ;

Processo B
   B1 () ;
   barrier_wait (b) ;
   B2 () ;
   barrier_wait (b) ;
   B3 () ;
```

```
Processo C
   C1 () ;
   barrier_wait (b) ;
   C2 () ;
   barrier_wait (b) ;
   C3 () ;

Processo D
   D1 () ;
   barrier_wait (b) ;
   D2 () ;
   barrier_wait (b) ;
   D3 () ;
```

Considerando a relação $a \rightarrow b$ como "a ocorre antes de b" e a relação $a \parallel b$ como "a e b ocorrem sem uma ordem definida", temos as seguintes restrições de sincronização:

- $\forall(i,j), X \neq Y, X_i \rightarrow Y_{j>i}$ (imposto pela barreira)
- $\forall(i,j), X_i \rightarrow X_{j>i}$ (imposto pela execução sequencial)
- $\forall(i,j), X \neq Y, X_i \parallel Y_{j=i}$ (possibilidade de execução concorrente)

Escreva o pseudo-código necessário para implementar barreiras para N processos, usando semáforos ou mutexes. Não esqueça de inicializar as variáveis e semáforos utilizados. Soluções que incorram em espera ocupada (*busy wait*) não serão aceitas.

```
1  // estrutura que representa uma barreira
2  typedef struct barrier_t
3  {
4     ... // completar
5  } barrier_t ;
6
7  // operador de espera no RV
8  void barrier_wait (barrier_t *barrier)
9  {
10    ... // completar
11 }
12
13 // inicialização de barreira para N processos
14 void barrier_init (barrier_t *barrier, int N)
15 {
```

```
16    ... // completar
17 }
```

16. Implemente uma solução em C para o problema do produtor/consumidor, usando threads e semáforos no padrão POSIX.

17. Implemente uma solução em C para o problema do produtor/consumidor, usando threads e variáveis de condição no padrão POSIX.

18. Implemente uma solução em C para o problema dos leitores/escritores com priorização para escritores, usando threads e semáforos POSIX.

19. Implemente uma solução em C para o problema dos leitores/escritores com priorização para escritores, usando threads e *rwlocks* POSIX.

20. Explique cada uma das quatro condições necessárias para a ocorrência de impasses.

21. Na prevenção de impasses:

    (a) Como pode ser feita a quebra da condição de posse e espera?

    (b) Como pode ser feita a quebra da condição de exclusão mútua?

    (c) Como pode ser feita a quebra da condição de espera circular?

    (d) Como pode ser feita a quebra da condição de não-preempção?

22. Como pode ser detectada a ocorrência de impasses, considerando disponível apenas um recurso de cada tipo?

23. Uma vez detectado um impasse, quais as abordagens possíveis para resolvê-lo? Explique-as e comente sua viabilidade.

24. Em relação aos impasses:

    I. As condições necessárias para a ocorrência de impasses são: exclusão mútua, posse e espera, não-preempção e espera circular.

    II. A condição de não-preempção indica que os processos envolvidos no impasse devem ser escalonados de forma não-preemptiva.

    III. A condição de não-preempção pode ser detectada graficamente, no grafo de alocação de recursos.

IV. A detecção e recuperação de impasses é bastante usada, pois as técnicas de recuperação são facilmente aplicáveis.

V. A condição de exclusão mútua pode ser quebrada através do uso de processos gerenciadores de recursos ou de áreas de *spool*.

As asserções corretas são:

(a) II
(b) I, V
(c) I, III
(d) III, IV
(e) II, V

Justifique as afirmações julgadas erradas (Assim: *VII está errada porque ...*):

25. Em relação aos impasses:

I. A quebra da condição de não-preempção só pode ser aplicada a recursos simples como arquivos e semáforos.

II. A quebra da condição de posse e espera consiste em forçar todos os processos a solicitar seus recursos em uma ordem global única e pré-fixada.

III. As condições necessárias para a ocorrência de impasses são também suficientes se houver somente um recurso de cada tipo no conjunto de processos considerado.

IV. A resolução de impasses através de *rollback* só pode ser implementada em processos que executem I/O ou interação com o usuário.

V. Uma vez detectado um impasse, ele pode ser facilmente resolvido através da preempção dos recursos envolvidos.

As asserções corretas são:

(a) III, V
(b) I

(c) I, V

(d) III

(e) II, IV

Justifique as afirmações julgadas erradas (Assim: *VII está errada porque ...*):

26. Em relação aos impasses:

I. Impasses ocorrem porque vários processos tentam usar o processador ao mesmo tempo.

II. O algoritmo de detecção de impasses deve ser executado com a maior freqüência possível, a fim de evitar que um impasse já formado se alastre.

III. O principal problema com a quebra da condição de posse e espera é que a taxa de uso dos recursos pode se tornar bastante baixa.

IV. Os sistemas operacionais atuais provêem vários recursos de baixo nível para o tratamento de impasses.

V. Podemos encontrar impasses em sistemas de processos que interagem unicamente por mensagens.

As asserções corretas são:

(a) I, II

(b) II

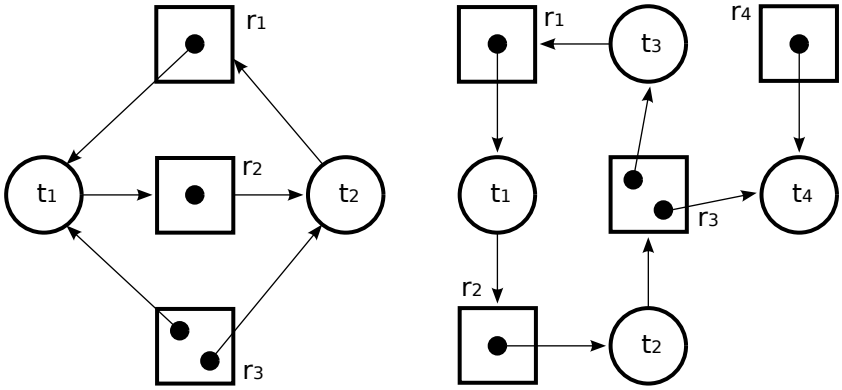
(c) III, V

(d) V

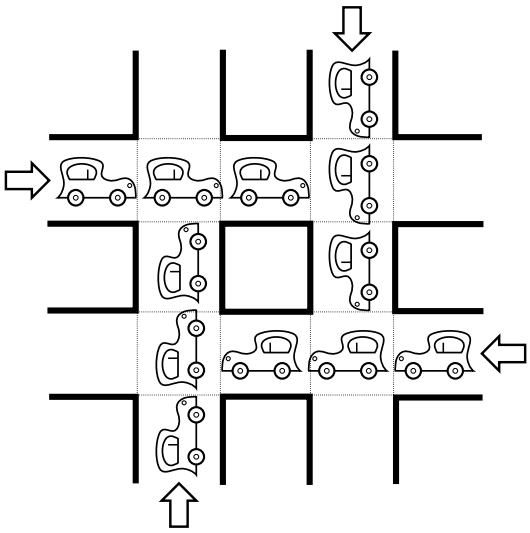
(e) III, IV

Justifique as afirmações julgadas erradas (Assim: *VII está errada porque ...*):

27. Nos grafos de alocação de recursos da figura a seguir, indique o(s) ciclo(s) onde existe um impasse:



28. A figura a seguir representa uma situação de impasse em um cruzamento de trânsito. Todas as ruas têm largura para um carro e sentido único. Mostre que as **quatro condições necessárias** para a ocorrência de impasses estão presentes nessa situação. Em seguida, defina uma regra simples a ser seguida por cada carro para **evitar** essa situação; regras envolvendo algum tipo de informação centralizada não devem ser usadas.



Capítulo 5

Gerência de memória

1. Explique a diferença entre endereços *lógicos* e endereços *físicos* e as razões que justificam seu uso.

2. Explique em que consiste a resolução de endereços nos seguintes momentos: *codificação*, *compilação*, *ligação*, *carga* e *execução*.

3. Como é organizado o espaço de memória de um processo?

4. O que é uma MMU – *Memory Management Unit*?

5. Seria possível e/ou viável implementar as conversões de endereços realizadas pela MMU em software, ao invés de usar um hardware dedicado? Por que?

6. Analise as seguintes afirmações relativas ao uso da memória RAM pelos processos:

   I. Os endereços físicos gerados pelo processador são convertidos em endereços lógicos através da MMU - *Memory Management Unit*.

   II. O acesso a endereços de memória inválidos é notificado ao processador através de interrupções geradas pela MMU.

   III. A área de memória `TEXT` contém o código-fonte a ser compilado e executado pelo processo.

   IV. A área de memória `DATA` é usada para armazenar todas as variáveis e constantes usadas pelo processo.

V. A área de memória `HEAP` é usada para as alocações dinâmicas de memória, sendo usada através de funções como `malloc` e `free`.

VI. A área de memória `STACK` contém as pilhas do programa principal e das demais *threads* do processo.

Indique a alternativa correta:

(a) As afirmações II e III estão corretas.

(b) As afirmações I e V estão corretas.

(c) Apenas a afirmação III está correta.

(d) As afirmações II e V estão corretas.

(e) As afirmações IV e VI estão corretas.

Justifique as afirmações julgadas erradas (Assim: *XIV está errada porque ...*):

7. Explique as principais formas de alocação de memória.

8. Explique como é feita a translação entre endereços lógicos e físicos e o mecanismo de tratamento de falta de página em um sistema de memória virtual paginada.

9. Por que os tamanhos de páginas e quadros são sempre potências de 2?

10. Analise as seguintes afirmações relativas às técnicas de alocação de memória:

I. Na alocação em partições fixas, a MMU é composta basicamente de um registrador e um somador.

II. Na alocação contígua, a área de memória acessível a cada processo é definida por um registrador *base* e um registrador *limite*.

III. A técnica de alocação contígua é imune a problemas de fragmentação externa.

IV. A alocação por segmentos resolve o problema da fragmentação externa.

V. Na alocação por segmentos, cada endereço de memória é composto de duas partes: *segmento* e *deslocamento*.

VI. A alocação por páginas resolve o problema da fragmentação externa.

Indique a alternativa correta:

(a) As afirmações II, III e VI estão corretas.
(b) As afirmações I, II, V e VI estão corretas.
(c) Apenas a afirmação V está correta.
(d) As afirmações IV e VI estão corretas.
(e) Todas as afirmações estão corretas.

Justifique as afirmações julgadas erradas (Assim: *XIV está errada porque ...*):

11. Considerando a tabela de segmentos abaixo (com valores em decimal), calcule os endereços físicos correspondentes aos endereços lógicos 0:45, 1:100, 2:90, 3:1.900 e 4:200.

Segmento	0	1	2	3	4
Base	44	200	0	2.000	1.200
Limite	810	200	1.000	1.000	410

12. Considerando a tabela de páginas abaixo, com páginas de 500 bytes[1], informe os endereços físicos correspondentes aos endereços lógicos 414, 741, 1.995, 4.000 e 6.633, indicados em decimal.

página	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
quadro	3	12	6	–	9	–	2	–	0	5	–	–	–	7	–	1

13. Considere um sistema com endereços físicos e lógicos de 32 bits, que usa tabelas de páginas com três níveis. Cada nível de tabela de páginas usa 7 bits do endereço lógico, sendo os restantes usados para o *offset*. Cada entrada das tabelas de páginas ocupa 32 bits. Calcule, indicando seu raciocínio:

[1] Um tamanho de página de 500 bytes permite fazer os cálculos mentalmente, sem a necessidade de converter os endereços para binário e vice-versa, bastando usar divisões inteiras (com resto) entre os endereços e o tamanho de página.

(a) O tamanho das páginas e quadros, em bytes.

(b) O tamanho máximo de memória que um processo pode ter, em bytes e páginas.

(c) O espaço ocupado pela tabela de páginas para um processo com apenas uma página de código, uma página de dados e uma página de pilha. As páginas de código e de dados se encontram no inicio do espaço de endereçamento lógico, enquanto a pilha se encontra no final do mesmo.

(d) Idem, caso todas as páginas do processo estejam mapeadas na memória.

14. Analise as seguintes afirmações relativas à alocação paginada:

I. Um endereço lógico com N bits é dividido em P bits para o número de página e $N - P$ bits para o deslocamento em cada página.

II. As tabelas de páginas multiníveis permitem mais rapidez na conversão de endereços lógicos em físicos.

III. O bit de referência R associado a cada página é "ligado" pela MMU sempre que a página é acessada.

IV. O cache TLB é usado para manter páginas frequentemente usadas na memória.

V. O bit de modificação M associado a cada página é "ligado" pelo núcleo sempre que um processo modificar o conteúdo da mesma.

VI. O cache TLB deve ser esvaziado a cada troca de contexto entre processos.

Indique a alternativa correta:

(a) As afirmações I, III e IV estão corretas.

(b) As afirmações II, V e VI estão corretas.

(c) Apenas a afirmação III está correta.

(d) As afirmações I, III e VI estão corretas.

(e) Todas as afirmações estão corretas.

Justifique as afirmações julgadas erradas (Assim: *XIV está errada porque ...*):

15. Explique o que é TLB, qual a sua finalidade e como é seu funcionamento.

16. Por que é necessário limpar o cache TLB após cada troca de contexto entre processos? Por que isso não é necessário nas trocas de contexto entre threads?

17. Um sistema de memória virtual paginada possui tabelas de página com três níveis e tempo de acesso à memória RAM de 100 ns. O sistema usa um cache TLB de 64 entradas, com taxa estimada de acerto de 98%, custo de acerto de 10 ns e penalidade de erro de 50 ns. Qual o tempo médio estimado de acesso à memória pelo processador? Apresente e explique seu raciocínio.

18. Explique o que é *fragmentação externa*. Quais formas de alocação de memória estão livres desse problema?

19. Explique o que é *fragmentação interna*. Quais formas de alocação de memória estão livres desse problema?

20. Em que consistem as estratégias de alocação *first-fit*, *best-fit*, *worst-fit* e *next-fit*?

21. Considere um sistema com processos alocados de forma contígua na memória. Em um dado instante, a memória RAM possui os seguintes "buracos", em seqüência e isolados entre si: 5K, 4K, 20K, 18K, 7K, 9K, 12K e 15K. Indique a situação final de cada buraco de memória após a seguinte seqüência de alocações: $12K \rightarrow 10K \rightarrow 5K \rightarrow 8K \rightarrow 10K$. Considere as estratégias de alocação *first-fit*, *best-fit*, *worst-fit* e *next-fit*.

22. Considere um banco de memória com os seguintes "buracos" não-contíguos:

B1	B2	B3	B4	B5	B6
10MB	4MB	7MB	30MB	12MB	20MB

Nesse banco de memória devem ser alocadas áreas de 5MB, 10MB e 2MB, nesta ordem, usando os algoritmos de alocação *First-fit*, *Best-fit* ou *Worst-fit*. Indique a alternativa correta:

(a) Se usarmos *Best-fit*, o tamanho final do buraco B4 será de 6 Mbytes.

(b) Se usarmos *Worst-fit*, o tamanho final do buraco B4 será de 15 Mbytes.

(c) Se usarmos *First-fit*, o tamanho final do buraco B4 será de 24 Mbytes.

(d) Se usarmos *Best-fit*, o tamanho final do buraco B5 será de 7 Mbytes.

(e) Se usarmos *Worst-fit*, o tamanho final do buraco B4 será de 9 Mbytes.

23. Considerando um sistema de 32 bits com páginas de 4 KBytes e um TLB com 64 entradas, calcule quantos erros de cache TLB são gerados pela execução de cada um dos laços a seguir. Considere somente os acessos à matriz `buffer` (linhas 5 e 9), ignorando páginas de código, *heap* e *stack*. Indique seu raciocínio.

```
1 unsigned char buffer[4096][4096] ;
2
3 for (int i=0; i<4096; i++)  // laço 1
4   for (int j=0; j<4096; j++)
5     buffer[i][j] = 0 ;
6
7 for (int j=0; j<4096; j++)  // laço 2
8   for (int i=0; i<4096; i++)
9     buffer[i][j] = 0 ;
```

24. Considerando um sistema com tempo de acesso à RAM de 50 ns, tempo de acesso a disco de 5 ms, calcule quanto tempo seria necessário para efetuar os acessos à matriz do exercício anterior nos dois casos (*laço 1* e *laço 2*). Considere que existem 256 quadros de 4.096 bytes (inicialmente vazios) para alocar a matriz e despreze os efeitos do cache TLB.

25. O que é uma falta de página? Quais são suas causa possíveis e como o sistema operacional deve tratá-las?

26. Calcule o tempo médio efetivo de acesso à memória se o tempo de acesso à RAM é de 5 ns, o de acesso ao disco é de 5 ms e em média

ocorre uma falta de página a cada 1.000.000 (10^6) de acessos à memória. Considere que a memória RAM sempre tem espaço livre para carregar novas páginas. Apresente e explique seu raciocínio.

27. Repita o exercício anterior, considerando que a memória RAM está saturada: para carregar uma nova página na memória é necessário antes abrir espaço, retirando outra página.

28. Considere um sistema de memória com quatro quadros de RAM e oito páginas a alocar. Os quadros contêm inicialmente as páginas 7, 4 e 1, carregadas em memória nessa seqüência. Determine quantas faltas de página ocorrem na seqüência de acesso {0, 1, 7, 2, 3, 2, 7, 1, 0, 3}, para os algoritmos de escalonamento de memória FIFO, OPT e LRU.

29. Repita o exercício anterior considerando um sistema de memória com três quadros de RAM.

30. Um computador tem 8 quadros de memória física; os parâmetros usados pelo mecanismo de memória virtual são indicados na tabela a seguir:

página	carga na memória	último acesso	bit R	bit M
p_0	14	58	1	1
p_1	97	97	1	0
p_2	124	142	1	1
p_3	47	90	0	1
p_4	29	36	1	0
p_5	103	110	0	0
p_6	131	136	1	1
p_7	72	89	0	0

Qual será a próxima página a ser substituída, considerando os algoritmos LRU, FIFO, segunda chance e NRU? Indique seu raciocínio.

31. Considere um sistema com 4 quadros de memória. Os seguintes valores são obtidos em dez leituras consecutivas dos bits de referência desses quadros: 0101, 0011, 1110, 1100, 1001, 1011, 1010, 0111, 0110 e 0111. Considerando o algoritmo de envelhecimento, determine o valor final do contador associado a cada página e indique que quadro será substituído.

32. Em um sistema que usa o algoritmo *WSClock*, o conteúdo da fila circular de referências de página em $t_c = 220$ é indicado pela tabela a seguir. Considerando que o ponteiro está em p_0 e que $\tau = 50$, qual será a próxima página a substituir? E no caso de $\tau = 100$?

página	último acesso	bit R	bit M
p_0	142	1	0
p_1	197	0	0
p_2	184	0	1
p_3	46	0	1
p_4	110	0	0
p_5	167	0	1
p_6	97	0	1
p_7	129	1	0

33. Considere as seguintes afirmações sobre memória virtual:

I. Por "Localidade de referências" entende-se o percentual de páginas de um processo que se encontram na memória RAM.

II. De acordo com a anomalia de Belady, o aumento de memória de um sistema pode implicar em pior desempenho.

III. A localidade de referência influencia significativamente a velocidade de execução de um processo.

IV. O algoritmo LRU é implementado na maioria dos sistemas operacionais, devido à sua eficiência e baixo custo computacional.

V. O compartilhamento de páginas é implementado copiando-se as páginas a compartilhar no espaço de endereçamento de cada processo.

VI. O algoritmo ótimo define o melhor comportamento possível em teoria, mas não é implementável.

As afirmações corretas são:

(a) II, III e VI

(b) I, II e IV

(c) II, IV e V

(d) I, IV e V

(e) IV, V e VI

Justifique as afirmações julgadas erradas (Assim: *XIV está errada porque ...*):

34. Construa um simulador de algoritmos de substituição de página. O simulador deve receber como entrada a sequência de referências a páginas de memória e gerar como saída o número de faltas de página geradas, para os algoritmos OPT, FIFO e LRU.

35. Construa um simulador de algoritmos de alocação de memória contígua. O simulador deve produzir aleatoriamente uma sequência de blocos de memória de tamanhos diferentes, simular sua alocação e gerar como saída o número de fragmentos livres de memória, os tamanhos do menor e do maior fragmentos e o tamanho médio dos fragmentos. Devem ser comparadas as estratégias de alocação *first-fit*, *next-fit*, *best-fit* e *worst-fit*.

Capítulo 6

Gerência de arquivos

1. Enumere os principais atributos de um arquivo.
2. Enumere as principais operações sobre arquivos.
3. Apresente e comente as principais formas de atribuição de tipos aos arquivos. Quais são as vantagens e desvantagens de cada uma?
4. Analise as seguintes afirmações relativas a formatos de arquivos:

   I. Um *magic number* consiste de um atributo numérico separado que identifica o tipo de arquivo.
   II. A forma mais comum de identificação de tipo de arquivo é o uso de extensões ao seu nome.
   III. Arquivos de texto em sistemas DOS e UNIX diferem nos caracteres de controle usados para identificar o fim de arquivo.
   IV. Para a maioria dos núcleos de sistema operacional, arquivos são quase sempre vistos como meras sequências de bytes.
   V. ELF e PE são dois formatos típicos de arquivos de configuração.
   VI. O padrão MIME é usado no Linux para identificação de tipos de arquivos pelo sistema operacional.

   As alternativas corretas são:

   (a) II e IV
   (b) II e V

(c) I e III

(d) IV e V

(e) III e VI

Justifique as afirmações julgadas erradas (Assim: *XIV está errada porque ...*):

5. O que é um ponteiro de arquivo? Para que ele serve?

6. Comente as principais formas de acesso a arquivos. Qual o uso mais apropriado para cada uma delas?

7. Quais as principais estruturas de diretórios empregadas em sistemas operacionais?

8. Do ponto de vista lógico, quais as principais diferenças entre a estrutura de diretórios Unix e Windows?

9. Explique os tipos de referências possíveis a arquivos em uma estrutura de diretórios.

10. Explique as formas de referência a arquivos direta, absoluta e relativa.

11. Analise as seguintes afirmações relativas ao uso de arquivos:

    I. No acesso sequencial, o ponteiro de posição corrente do arquivo é reiniciado a cada operação.

    II. O acesso direto pode ser implementado usando o acesso sequencial e operações de posicionamento do ponteiro do arquivo.

    III. No acesso mapeado em memória, o conteúdo do arquivo é copiado para a memória RAM durante a sua abertura.

    IV. O acesso indexado é raramente implementado pelo núcleo em sistemas operacionais *desktop*, sendo mais frequente em ambientes *mainframe*.

    V. Travas de uso exclusivo e compartilhado implementam um modelo de sincronização de tipo *produtor/consumidor* no acesso ao arquivo.

VI. Segundo a semântica de compartilhamento UNIX, o conteúdo de um arquivo é considerado imutável durante um compartilhamento.

As alternativas corretas são:

(a) II e IV
(b) II e V
(c) III e V
(d) I e IV
(e) III e VI

Justifique as afirmações julgadas erradas (Assim: *XIV está errada porque ...*):

12. Um conjunto de processos p_1, p_2, p_3 e p_4 abrem em leitura/escrita um arquivo compartilhado contendo um número inteiro, cujo valor inicial é 34. As operações realizadas pelos processos são indicadas na tabela a seguir no formato $[t, op]$, onde t é o instante da operação e op é a operação realizada:

p_1	p_2	p_3	p_4
[0, *open*]	[3, *open*]	[7, *open*]	[9, *open*]
[2, *write* 41]	[6, *write* 27]	[8, *read X*]	[11, *read Y*]
[6, *close*]	[8, *close*]	[9, *write* 4]	[12, *close*]
		[10, *close*]	

Considerando a semântica de sessão para o compartilhamento de arquivos, determine os valores de X e Y, **explicando seu raciocínio**. Cada operação de escrita no arquivo substitui o valor anterior.

13. Enumere principais problemas a resolver na implementação de um sistema de arquivos.

14. Apresente a arquitetura de gerência de arquivos presente em um sistema operacional típico, explicando seus principais elementos constituintes.

15. Explique o que é alocação contígua de arquivos, apresentando suas vantagens e desvantagens.

16. No contexto de alocação de arquivos, o que significa o termo *best-fit*?

17. Explique a alocação de arquivos em listas encadeadas, apresentando suas principais vantagens e desvantagens.

18. Explique a estrutura do sistema de arquivos conhecido como FAT, comentando sobre suas qualidades e deficiências.

19. Por que a alocação de arquivos em listas encadeadas é considerada pouco robusta? O que pode ser feito para melhorar essa característica?

20. Explique o esquema de alocação indexada de arquivos usando índices multi-níveis.

21. O que é fragmentação interna e fragmentação externa? Por que elas ocorrem?

22. Analise o impacto das fragmentações interna e externa nos sistemas de alocação contígua, indexada e por lista encadeadas.

23. Considere um sistema operacional hipotético que suporte simultaneamente as estratégias de alocação contígua, encadeada e indexada para armazenamento de arquivos em disco. Que critérios devem ser considerados para decidir a estratégia a usar para cada arquivo em particular?

24. Avalie as seguintes afirmações sobre as técnicas de alocação de arquivos:

    I. A alocação contígua é muito utilizada em sistemas desktop, por sua flexibilidade.

    II. A alocação FAT é uma alocação encadeada na qual os ponteiros de blocos foram transferidos para um vetor de ponteiros.

    III. Na alocação indexada os custos de acesso seqüencial e aleatório a blocos são similares.

    IV. Na alocação contígua, blocos defeituosos podem impedir o acesso aos demais blocos do arquivo.

V. Na alocação contígua, o custo de acesso a blocos aleatórios é alto.

VI. Apesar de complexa, a alocação indexada é muito usada em *desktops* e servidores.

As afirmações corretas são:

(a) II, III e VI

(b) I, III e IV

(c) I, IV e V

(d) II, IV e V

(e) IV, V e VI

Justifique as afirmações julgadas erradas (Assim: *XIV está errada porque ...*):

25. Considerando um arquivo com 500 blocos em disco, calcule quantas leituras e quantas escritas em disco são necessárias para (a) inserir um novo bloco no início do arquivo ou (b) inserir um novo bloco no final do arquivo, usando as formas de alocação de blocos contígua, encadeada e indexada.

Alocação	Contígua		Encadeada		Indexada	
Operações	leituras	escritas	leituras	escritas	leituras	esc
Inserir um novo bloco no início do arquivo						
Inserir um novo bloco no final do arquivo						

Observações:

(a) Considere somente as operações de leitura e escrita nos blocos do próprio arquivo e no *i-node*; a tabela de diretório sempre está em memória;

(b) Para a alocação contígua, assuma que não há espaço livre depois do arquivo, somente antes dele;

(c) Para a alocação encadeada, assuma que a tabela de diretório contém apenas um ponteiro para o início do arquivo no disco. Os ponteiros dos blocos estão contidos nos próprios blocos;

(d) Para a alocação indexada, considere *i-nodes* com somente um nível, contendo somente os ponteiros para os blocos de dados. O *i-node* está no disco.

26. Considere um disco rígido com capacidade total de 1 Mbyte, dividido em 1.024 blocos de 1.024 bytes cada. Os dez primeiros blocos do disco são reservados para a tabela de partições, o código de inicialização (*boot*) e o diretório raiz do sistema de arquivos. Calcule o tamanho máximo de arquivo (em bytes) que pode ser criado nesse disco para cada uma das formas de alocação a seguir, explicando seu raciocínio:

(a) Alocação contígua.

(b) Alocação encadeada, com ponteiros de 64 bits contidos nos próprios blocos.

(c) Alocação indexada, com *i-nodes* contendo somente ponteiros diretos de 64 bits; considere que o *i-node* não contém meta-dados, somente ponteiros, e que ele ocupa exatamente um bloco do disco.

27. Considerando a tabela FAT (*File Allocation Table*) a seguir, indique:

(a) o número de blocos ocupados pelo arquivo `relat.pdf`;

(b) o tamanho (em blocos) do maior arquivo que ainda pode ser criado nesse disco;

(c) quais arquivos estão íntegros e quais estão corrompidos por blocos defeituosos (*bad blocks*);

(d) quantos blocos do disco estão perdidos, ou seja, não são usados por arquivos nem estão marcados como livres ou defeituosos.

Na tabela, a letra `R` indica bloco reservado (*Reserved*), `F` indica bloco livre (*Free*), `L` indica o último bloco de um arquivo (*Last*) e `B` indica bloco defeituoso (*Bad*).

arquivo	início
readme.txt	76
icone.gif	14
retrato.jpg	29
relat.pdf	6
format.exe	31
carta.doc	67
programa.c	73

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
R	R	R	R	R	F	17	F	15	68	13	53	F	L	63	L	F	26	F	F
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
33	L	F	38	L	F	11	55	F	36	F	35	43	B	F	B	20	F	8	F
40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
21	32	F	50	B	L	F	F	40	F	L	45	F	58	F	B	F	F	72	F
60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
44	F	F	51	F	F	F	60	24	F	F	F	10	27	F	F	41	F	L	F

28. O sistema de arquivos indexado do sistema *Minix* possui os seguintes campos em cada *i-node*:

    - meta-dados (tipo, dono, grupo, permissões, datas e tamanho)
    - 7 ponteiros diretos
    - 1 ponteiro indireto
    - 1 ponteiro duplamente indireto

    A implementação básica desse sistema de arquivos considera blocos de 1.024 bytes e ponteiros de 32 bits. Desenhe o diagrama do sistema de arquivos e calcule o tamanho máximo de arquivo que ele suporta, indicando seu raciocínio.

29. O sistema de arquivos indexado `ext2fs`, usado no *Linux*, possui os seguintes campos em cada *i-node*:

    - meta-dados (tipo, dono, grupo, permissões, datas e tamanho)
    - 12 ponteiros diretos
    - 1 ponteiro indireto
    - 1 ponteiro duplamente indireto
    - 1 ponteiro triplamente indireto

    A implementação básica do `ext2fs` considera blocos de 1.024 bytes e ponteiros de 64 bits. Desenhe o diagrama do sistema de arquivos e determine o tamanho máximo de arquivo que ele suporta, indicando seu raciocínio.

30. Explique como é efetuada a gerência de espaço livre através de *bitmaps*.

Capítulo 7

Gerência de entrada/saída

1. Considere um escalonador de disco com os seguintes pedidos de leitura de blocos em sua fila, nessa ordem: 95, 164, 36, 68, 17 e 115. Determine todos os deslocamentos da cabeça de leitura do disco para atender esses pedidos e o número total de blocos percorridos, para as políticas FCFS, SSTF, SCAN, C-SCAN, LOOK e C-LOOK. O disco tem 200 setores, numerados de 0 a 199, e a cabeça de leitura acabou de percorrer os blocos 76 e 50, nessa ordem.

Capítulo 8

Segurança de sistemas

1. Analise as seguintes afirmações relativas às propriedades da segurança:

   I. A *Confidencialidade* consiste em garantir que as informações do sistema estarão criptografadas.

   II. A *Integridade* consiste em garantir que as informações do sistema só poderão ser modificadas por usuários autorizados.

   III. A *Disponibilidade* implica em assegurar que os recursos do sistema estarão disponíveis para consulta por qualquer usuário.

   IV. A *Autenticidade* implica em assegurar que os dados das entidades atuantes no sistema sejam verdadeiros e correspondam às informações do mundo real que elas representam.

   V. A *Irretratabilidade* implica em garantir que nenhuma ação possa ser desfeita no sistema.

   As alternativas corretas são:

   (a) II e IV
   (b) II e V
   (c) I e III
   (d) I e IV
   (e) III e V

   Justifique as afirmações julgadas erradas (Assim: *XIV está errada porque ...*):

2. Analise as seguintes afirmações relativas aos princípios de segurança:

   I. Princípio do *Privilégio Mínimo*: os processos devem receber o mínimo possível de privilégios, para minimizar os riscos em caso de bugs ou erros.

   II. Princípio do *Default Seguro*: os acessos permitidos devem ser explicitados; caso um acesso não seja explicitamente permitido, ele deve ser negado.

   III. Princípio da *Separação de Privilégios*: os privilégios dos usuários comuns devem ser separados dos privilégios do administrador do sistema.

   IV. Princípio do *Projeto Aberto*: a robustez do mecanismo de proteção não deve depender de segredos de programação.

   V. Princípio da *Facilidade de Uso*: o uso dos mecanismos de segurança deve ser fácil e intuitivo para os usuários.

   As alternativas corretas são:

   (a) I, II, III e IV
   (b) I, II, IV e V
   (c) II, III, IV e V
   (d) I, III, IV e V.
   (e) Todas

   Justifique as afirmações julgadas erradas (Assim: *XIV está errada porque ...*):

3. Relacione as situações abaixo a ataques diretos à (C)onfidencialidade, (I)ntegridade, (D)isponibilidade ou (A)utenticidade, justificando suas escolhas.

   [ ] Um programa que permite injetar pacotes falsos na rede.
   [ ] Um ataque de negação de serviços através da rede.
   [ ] Um processo *spyware* que vasculha os arquivos do sistema em busca de senhas.
   [ ] `int main { while (1) fork(); }`

[ ] Um site malicioso que imita um site bancário.

[ ] Um programa quebrador de senhas.

[ ] Um processo que modifica o arquivo de sistema `/etc/hosts` para redirecionar acessos de rede.

[ ] Um programa baixado da Internet que instala um *malware* oculto no sistema operacional.

[ ] Uma página Web cheia de arquivos *Flash* para sobrecarregar o processador.

[ ] Um programa de captura de pacotes de rede.

4. O código a seguir apresenta uma vulnerabilidade de segurança. Indique qual é essa vulnerabilidade e explique como ela pode ser usada por um atacante.

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <ctype.h>
4
5  int confirma (char * pergunta)
6  {
7    char resp[20] ;
8
9    printf ("%s (sim/nao): ", pergunta) ;
10   scanf ("%s", &resp[0]) ;
11   if (! strcmp (resp, "sim")) return (1) ;
12   return (0) ;
13 }
14
15 int main ()
16 {
17   ...
18
19   if (confirma ("Devo apagar os valores?"))
20   {
21     ...
22   }
23   ...
24 }
```

5. Relacione os tipos de *malwares* às suas respectivas descrições:

(V)írus, (W)orm, (T)rojan, (R)ootkit, (B)ackdoor, (E)xploit

[ ] Pode operar no nível dos comandos, das bibliotecas, do núcleo do sistema operacional ou mesmo abaixo dele.

[ ] Técnicas de engenharia social geralmente são empregadas para induzir o usuário a executar esse tipo de programa.

[ ] É um programa que se propaga entre sistemas usando vulnerabilidades em seus serviços.

[ ] É um trecho de código que se infiltra em programas executáveis, usando-os como suporte para sua execução e propagação.

[ ] É um programa construído para demonstrar ou explorar uma vulnerabilidade de um sistema.

[ ] Pode usar sistemas de e-mail ou de mensagens instantâneas para sua propagação.

[ ] É um programa que facilita a entrada do intruso em um sistema já invadido, ou que permite seu comando remotamente.

[ ] Programa usado para esconder a presença de um intruso no sistema.

[ ] Sua execução depende da execução do programa hospedeiro.

[ ] É um programa usado para enganar o usuário e fazê-lo instalar outros *malwares*.

[ ] Pode usar suportes de execução internos (macros) de editores de texto para sua propagação.

[ ] Costuma infectar *pendrives* plugados em portas USB.

6. Na série de TV Futurama (escrita por Matt Groening, o mesmo autor dos Simpsons) é usada uma escrita cifrada denominada *Alien Language*. Essa codificação pode ser decifrada através da tabela a seguir:

Explique qual o tipo de criptografia empregado na *Alien Language* e indique qual o tamanho do espaço de chaves da mesma.

7. O texto em português a seguir foi cifrado usando o cifrador de César. Encontre o **texto original** e a **chave** usada para cifrá-lo; explique seu procedimento.

```
Kjqne fvzjqj vzj ywfsxkjwj t vzj
xfgj j fuwjsij t vzj jsxnsf.
Htwf Htwfqnsf.
```

Para facilitar seu trabalho, a tabela a seguir traz a frequência de caracteres típica de textos na língua portuguesa:

letra	freq%	letra	freq%	letra	freq%	letra	freq%	letra	freq
A	14,63	B	1,04	C	3,88	D	4,99	E	12,5
F	1,02	G	1,30	H	1,28	I	6,18	J	0,4
K	0,02	L	2,78	M	4,74	N	5,05	O	10,7
P	2,52	Q	1,20	R	6,53	S	7,81	T	4,3
U	4,63	V	1,67	W	0,01	X	0,21	Y	0,0
Z	0,47								

8. O cifrador de Vigenère é um método de cifragem que combina vários cifradores de César em sequência. Ele constitui um exemplo simples de *cifrador polialfabético*. Para as operações de cifragem/decifragem é usada uma tabela denominada *tabula rasa*:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Para cifrar uma mensagem, primeiro se escolhe uma palavra-chave qualquer, que é repetida até ter o mesmo comprimento da mensagem. Em seguida, cada caractere da mensagem original é codificado usando uma cifra de substituição (cifrador de César). A cifra a usar para um caractere é definida pela linha da *tabula rasa* indicada pela letra correspondente da palavra-chave. Um exemplo de cifragem usando a palavra-chave "bicicleta" é indicado a seguir:

Mensagem aberta	`Atacare`**`m`**`os ao amanhecer de sexta-feira`
Palavra-chave	`bicicle`**`t`**`ab ic icletabic ic letab icicl`
Mensagem cifrada	`bbckcci`**`f`**`ot iq iolraedmt lg diqtb-ngqtl`

Use o cifrador de Vigenère para cifrar a mensagem secreta "Encontramos aliens" usando a palavra-chave "missao":

Msg aberta	E	n	c	o	n	t	r	a	m	o	s		a	l	i	e	n
Palavra-chave																	
Msg cifrada																	

9. Alice precisa enviar a imagem ISO de um CD confidencial a seus amigos Bob, Carol e David. Como o arquivo é muito grande, ela o carregou em um servidor de arquivos acessível remotamente. Contudo, esse servidor pode ser invadido e as comunicações entre eles podem ser capturadas. Como Alice pode cifrar o arquivo ISO de forma que somente Bob, Carol e David possam abri-lo, cada um com sua própria chave?

10. Recentemente foi noticiado na imprensa que certificados digitais emitidos pela Autoridade Certificadora holandesa *DigiNotar* haviam sido falsificados e estavam sendo usados por um governo do oriente médio para monitorar as comunicações de seus cidadãos. Considerando o certificado falso do serviço de e-mails do *Google* (`mail.google.com`), explique:

    (a) Neste contexto, em que consiste um certificado falso?

    (b) Qual a utilidade de um certificado falso na interceptação de comunicações?

    (c) Por que somente os usuários do navegador *Chrome* (produzido pelo próprio *Google*) detectaram o certificado falso, enquanto usuários de outros navegadores não perceberam nada?

11. O provedor de conteúdo TOL (*Tabajara OnLine*) decidiu implementar um novo mecanismo de segurança em suas páginas web. Esse mecanismo consiste em adicionar uma etiqueta oculta (*HTML tag*) em cada página, contendo o nome do autor (*name*), a data de produção (*date*) e uma assinatura digital *s*. Essa assinatura é constituida pelo hash criptográfico do nome do autor e da data ($hash(name + date)$), cifrado usando a chave privada do autor da página. O conteúdo da página Web em si não é cifrado. As chaves públicas dos autores registrados podem ser obtidas em `http://www.tol.com.br/pubkeys.html`.

    Responda:

    (a) Que objetivo tinham em mente os proponentes desse mecanismo?

    (b) Esse esquema é seguro? Por que?

    (c) Se o esquema não for seguro, indique um possível ataque ao mesmo; caso seja seguro, explique por que esse mesmo ataque não funcionaria.

12. Analise as seguintes afirmações relativas às técnicas de autenticação:

    I. Nas estratégias de autenticação SYK, o sistema autentica o usuário com base em informações fornecidas pelo mesmo.

    II. Nas estratégias de autenticação SYH, o sistema usa dados coletados do usuário para fazer sua autenticação.

    III. Nas estratégias de autenticação SYA, o usuário é autenticado com base em suas características físicas.

    As alternativas corretas são:

    (a) I e II
    (b) II e III
    (c) I e III
    (d) Nenhuma delas
    (e) Todas elas

    Justifique as afirmações julgadas erradas (Assim: *XIV está errada porque ...*):

13. Analise as seguintes afirmações relativas às técnicas de autenticação:

    I. Para estar devidamente protegidas, as senhas armazenadas no sistema devem ser cifradas com criptografia simétrica.

    II. A autenticação multi-fator consiste em autenticar o usuário usando duas senhas simultaneamente.

    III. A autenticação por técnicas biométricas deve usar características físicas *universais*, *singulares*, *permanentes* e *mensuráveis* dos usuários.

    IV. Os *tokens* de segurança usados no acesso a serviços bancários pela Internet implementam um esquema de senhas baseado em desafio-resposta.

    V. PAM e SSPI são infraestruturas de autenticação modulares usadas em sistemas operacionais de mercado.

    As alternativas corretas são:

(a) II e IV
(b) II e V
(c) I e III
(d) IV e V
(e) III e V

Justifique as afirmações julgadas erradas (Assim: *XIV está errada porque ...*):

14. Qual a função do "sal" usado em sistemas de autenticação por senhas? Explique como o "sal" é usado; sua explicação deve conter um diagrama.

15. Analise as seguintes afirmações relativas aos modelos de controle de acesso:

    I. Nos modelos de controle de acesso obrigatórios, o controle é definido por regras globais incontornáveis, que não dependem das identidades dos sujeitos e objetos nem da vontade de seus proprietários ou mesmo do administrador do sistema.
    II. Os modelos de controle de acesso discricionários se baseiam na atribuição de permissões de forma individualizada, ou seja, pode-se conceder ou negar a um sujeito específico a permissão de executar uma ação sobre um dado objeto.
    III. O Modelo da matriz de controle de acesso é uma forma de representação lógica de políticas discricionárias.
    IV. O modelo de Bell-LaPadula é uma forma de representar políticas de controle de acesso obrigatórias que tenham foco em confidencialidade de dados.
    V. O modelo de Biba é uma forma de representar políticas de controle de acesso obrigatórias que tenham foco em integridade de dados.
    VI. Os modelos de controle de acesso baseados em papéis permitem desvincular os usuários das permissões sobre os objetos, através da definição e atribuição de papéis.

    As alternativas corretas são:

(a) I, II, IV

(b) II, III, VI

(c) I, II, III, V

(d) Nenhuma delas

(e) Todas elas

Justifique as afirmações julgadas erradas (Assim: *XIV está errada porque ...*):

16. Analise a seguinte matriz de controle de acesso:

	$file_1$	$file_2$	$program_1$	$socket_1$
Alice	*read* *write* *remove* ***owner***	*read* *write*	*execute*	*write*
Beto	*read* *write*	*read* *write* *remove* ***owner***	*read* ***owner***	
Carol		*read*	*execute*	*read* *write*
Davi	*read*	*write*	*read*	*read* *write* ***owner***

Assinale a alternativa correta:

(a) O usuário *Beto* pode alterar as permissões dos recursos $file_1$ e $program_1$

(b) O usuário *Alice* tem a seguinte lista de capacidades: $\{file_1 : (read, write, remove, owner), file_2 : (read, write), program_1 : (read, execute), socket_1 : (write)\}$

(c) A lista de controle de acesso de $file_2$ é: $\{Alice : (read, write), Beto : (read, write, remove), Carol : (read), Davi : (write)\}$

(d) A lista de capacidades de *Beto* é: $\{file_1 : (read, write), file_2 : (read, write, remove, owner), program_1 : (read, owner)\}$

(e) Nenhuma das anteriores

17. Escreva as listas de controle de acesso (*ACLs*) equivalentes às listas de capacidades a seguir:

$$
\begin{aligned}
CL(Alice) &= \{\ file_1 : (read, write, remove, owner),\\
&\quad\ file_2 : (read),\\
&\quad\ program_1 : (execute),\\
&\quad\ socket_1 : (read, write)\ \}\\
CL(Beto) &= \{\ file_1 : (read),\\
&\quad\ file_2 : (read, write, remove, owner),\\
&\quad\ program_1 : (read, execute, owner)\ \}\\
CL(Carol) &= \{\ file_2 : (read, write),\\
&\quad\ program_1 : (execute),\\
&\quad\ socket_1 : (read, write)\ \}\\
CL(Davi) &= \{\ file_1 : (read),\\
&\quad\ file_2 : (write),\\
&\quad\ program_1 : (read, execute),\\
&\quad\ socket_1 : (read, write, owner)\ \}
\end{aligned}
$$

18. Relacione as expressões a seguir aos modelos de controle de acesso de Bell (L)aPadula, (B)iba ou da (M)atriz de controle de acesso. Considere s um sujeito, o um objeto, $h(s)$ o nível de habilitação ou de integridade do sujeito e $c(o)$ a classificação do objeto.

[ ] $request(s_i, o_j, write) \iff h(s_i) \geq c(o_j)$

[ ] $request(s_i, o_j, write) \iff write \in M_{ij}$

[ ] $request(s_i, o_j, read) \iff h(s_i) \geq c(o_j)$

[ ] $request(s_i, o_j, read) \iff read \in M_{ij}$

[ ] $request(s_i, o_j, write) \iff h(s_i) \leq c(o_j)$

[ ] $request(s_i, o_j, read) \iff h(s_i) \leq c(o_j)$

19. Preencha a matriz de controle de acesso que corresponde à seguinte listagem de arquivo em um ambiente UNIX:

```
-rwxr-x---  2 maziero   prof      14321   2010-07-01 16:44
-rw-------  2 lucas     aluno    123228   2008-12-27 08:53
-rwxr-x--x  2 daniel    suporte    3767   2010-11-14 21:50
-rw-rw-r--  2 sheila    prof      76231   2009-18-27 11:06
-rw-r-----  2 mariana   aluno      4089   2010-11-09 02:14
```

Observações:

- Composição do grupo `prof`: {`maziero`, `sheila`}
- Composição do grupo `suporte`: {`maziero`, `daniel`}
- Composição do grupo `aluno`: {`lucas`, `daniel`, `mariana`}
- Preencha os campos da matriz com os caracteres "`r`", "`w`", "`x`" e "`-`".

	script.sh	relat.pdf	backup.py	cmmi.xml	teste1
maziero					
lucas					
daniel					
sheila					
mariana					

20. Em um sistema de documentação militar estão definidos os seguintes usuários e suas respectivas habilitações:

Usuário	Habilitação
Marechal Floriano	Ultrassecreto
General Motors	Secreto
Major Nelson	Confidencial
Sargento Tainha	Restrito
Recruta Zero	Público

Considerando operações sobre documentos classificados, indique quais das operações a seguir seriam permitidas pelo modelo de controle de acesso de Bell-LaPadula:

[ ] Sargento Tainha cria o documento secreto `comunicado.txt`

[ ] Recruta Zero lê o documento ultrassecreto `salarios-dos-generais.`

[ ] General Motors escreve um memorando público `aviso-sobre-feria`

[ ] Major Nelson escreve um documento confidencial `avarias-no-submarino.doc`.

[ ] Marechal Floriano lê o documento restrito `comunicado.txt`.

[ ] General Motors lê o documento secreto `vendas-de-carros-2010.do`

[ ] Sargento Tainha lê o documento restrito `plano-de-ataque.pdf`.

[ ] Major Nelson lê o documento confidencial `processos-navais.html`.

[ ] Marechal Floriano escreve o documento secreto `novas-avenidas.do`

[ ] Recruta Zero escreve o documento ultrassecreto `meu-diario.txt`.

21. As listas de controle de acesso (*ACLs*) e as listas de capacidades (*CLs*) a seguir são complementares, mas estão incompletas. Complete-as com as regras faltantes.

$$
\begin{array}{lcl}
ACL(o_1) & = & \{ \quad (u_1 : rwx) \qquad\qquad\qquad\qquad\qquad\quad \} \\
ACL(o_2) & = & \{ \quad (u_2 : r) \qquad\qquad\qquad\qquad\qquad\quad \} \\
ACL(o_3) & = & \{ \quad (u_1 : r) \qquad (u_4 : rw) \qquad\qquad\quad \} \\
ACL(o_4) & = & \{ \quad (u_2 : rw) \qquad (u_3 : r) \qquad\qquad\quad \}
\end{array}
$$

$$
\begin{array}{lcl}
CL(u_1) & = & \{ \quad (o_2 : rw) \qquad (o_4 : r) \qquad\qquad\quad \} \\
CL(u_2) & = & \{ \quad (o_1 : rx) \qquad\qquad\qquad\qquad\qquad\quad \} \\
CL(u_3) & = & \{ \quad (o_1 : rx) \qquad\qquad\qquad\qquad\qquad\quad \} \\
CL(u_4) & = & \{ \quad (o_4 : rwx) \qquad\qquad\qquad\qquad\qquad\quad \}
\end{array}
$$

22. Considerando o modelo de controle de acesso de Bell & LaPadula, indique que tipo de acesso (R, W, RW ou –) um usuário u pode ter sobre os documentos abaixo identificados. Considere que $h(u) = secreto$ e que $\mathbb{C}(u) = \{vendas, rh\}$.

[ ] d_1: $c(d_1) = ultrassecreto$ e $\mathbb{C}(d_1) = \{vendas\}$

[ ] d_2: $c(d_2) = publico$ e $\mathbb{C}(d_2) = \{rh, financeiro\}$

[ ] d_3: $c(d_3) = secreto$ e $\mathbb{C}(d_3) = \{rh\}$

[ ] d_4: $c(d_4) = reservado$ e $\mathbb{C}(d_4) = \{rh, vendas\}$

[ ] d_5: $c(d_5) = confidencial$ e $\mathbb{C}(d_5) = \{\ \}$

23. Muitas vezes, usuários precisam executar ações que exigem privilégios administrativos, como instalar programas, reconfigurar serviços, etc. Neste contexto, analise as seguintes afirmações:

I. No mecanismo UAC – *User Access Control* – dos sistemas Windows, um usuário administrativo inicia sua seção de trabalho com seus privilégios de usuário normal e recebe mais privilégios somente quando precisa efetuar ações que os requeiram.

II. Alguns sistemas operacionais implementam mecanismos de *mudança de credenciais*, através dos quais um processo pode mudar de proprietário.

III. As "POSIX Capabilities" são uma implementação do mecanismo de *capabilities* para sistemas operacionais que seguem o padrão POSIX.

IV. Alguns sistemas operacionais separam os usuários em *usuários normais* ou *administrativos*, atribuindo aos últimos permissões para efetuar tarefas administrativas, como instalar programas.

V. Alguns sistemas operacionais implementam processos *monitores* que recebem pedidos de ações administrativas vindos de processos com baixo privilégio, que são avaliados e possivelmente atendidos.

VI. Os flags `setuid` e `setgid` do UNIX implementam um mecanismo de permissões temporárias.

As alternativas corretas são:

(a) I, II, IV, VI

(b) II, III, VI

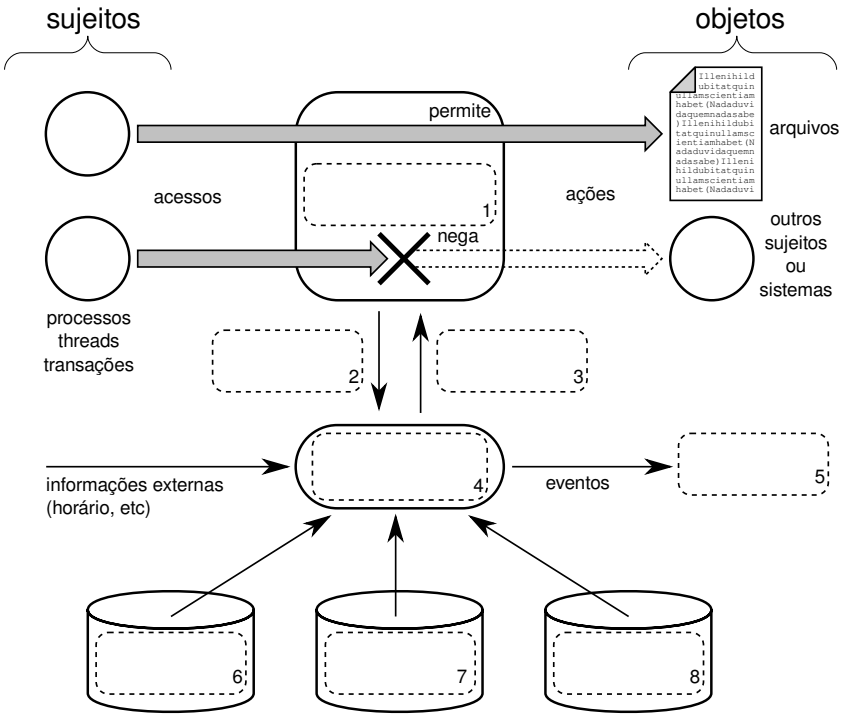
(c) I, II, III, V

(d) I, II, IV, V

(e) Todas elas

Justifique as afirmações julgadas erradas (Assim: *XIV está errada porque ...*):

24. O diagrama abaixo representa os principais componentes da infraestrutura de controle de acesso de um sistema operacional típico. Identifique e explique elementos representados pelas caixas tracejadas.



25. A listagem a seguir apresenta alguns programas executáveis e arquivos de dados em um mesmo diretório de um sistema UNIX, com suas respectivas permissões de acesso:

```
- rwx r-s ---  2 marge    family      indent
- rwx r-x --x  2 homer    family      more
- rws r-x --x  2 bart     men         nano
- rwx r-x ---  2 lisa     women       sha1sum

- rw- r-- ---  2 lisa     women       codigo.c
- rw- rw- ---  2 marge    family      dados.csv
- rw- r-- ---  2 bart     men         prova.pdf
- rw- rw- ---  2 homer    family      relatorio.txt
- rw- --- ---  2 bart     men         script.sh
```

Os programas executáveis precisam das seguintes permissões de acesso sobre os arquivos aos quais são aplicados para poder executar:

- `more`, `sha1sum`: leitura

- `nano`, `indent`: leitura e escrita

Considerando os grupos de usuários $men = \{bart, homer, moe\}$, $women = \{marge, lisa, maggie\}$ e $family = \{homer, marge, bart, lisa, maggie\}$, indique quais dos comandos a seguir serão permitidos e quais serão negados. O *prompt* indica qual usuário/grupo está executando o comando:

[ ] `lisa:women> nano codigo.c`

[ ] `lisa:women> more relatorio.txt`

[ ] `bart:men> nano relatorio.txt`

[ ] `bart:men> sha1sum prova.pdf`

[ ] `marge:women> more relatorio.txt`

[ ] `marge:women> indent codigo.c`

[ ] `homer:men> sha1sum prova.pdf`

[ ] `homer:men> nano dados.csv`

[ ] `moe:men> sha1sum relatorio.txt`

[ ] `moe:men> nano script.sh`

26. Sistemas de detecção de intrusão (IDS - *Intrusion Detection Systems*) podem ser classificados sob várias perspectivas. Explique como os IDSs são classificados segundo:

    (a) A origem dos dados analisados;

    (b) O momento da análise dos dados;

    (c) A forma de análise dos dados.