01. O Sistema Operacional Linux possui um método simples de definição de permissão de acesso a arquivos. Tais permissões liberam o acesso à escrita, execução e leitura dos arquivos. Os tipos de permissão escrita, execução e leitura são, respectivamente,

a) “w”, “r” e “x”.
b) “e”, “x” e “l”.
c) “r”, “w” e “x”.
d) “w”, “x” e “r”.

02. Os espaços de memória são alocados pelo Sistema Operacional Linux, levando em consideração

a) a paginação do programa.
b) o processo em execução.
c) o usuário que requisitou a operação.
d) a operação que requisitou o sistema de memória.

03. Com relação à comunicação entre processos, Os Sistemas Distribuídos podem adotar como formas de comunicação entre seus processos

a) trocas de mensagens, apenas.
b) trocas de sinais, apenas.
c) tanto trocas de mensagens como trocas de sinais.
d) tecnologias específicas para comunicação em sistemas distribuídos, tais como RML e PCS.

04. Considerando processos em sistemas operacionais, o fato de múltiplas execuções poderem ocorrer simultaneamente, independentemente e com ganhos consideráveis de desempenho deve-se, principalmente, a uma característica acrescida a esse modelo por

a) processadores com tecnologia Xextreme.
b) processadores com tecnologia Dual Core.
c) threads.
d) compiladores.

05. É responsável por armazenar informações importantes sobre processos, tais como identificação do processo, estado do processo, prioridade do processo e gerenciamento de instruções do processo,

a) o PID – Process Identification Number.
b) o Escalonador.
c) o PCB – Process Control Block.
d) o sistema de alocação de memória.

06. Os principais arquivos executáveis no sistema operacional Linux normalmente devem ser organizados no diretório

a) /root.
b) /home.
c) /bin.
d) /lib.

07. Considerando a comunicação entre processos, se um sistema operacional utiliza-se exaustivamente de operações I/O Bound (operações de entrada e saída), isso significa que

a) existe pouca comunicação entre processos.
b) existe muita comunicação entre processos.
c) o sistema apresenta alguma falha de execução.
d) ocorreu um *deadlock*.

08. Qual o tipo de escalonamento que, a partir do momento em que um processo recebe o direito de utilização da CPU, nenhum outro processo, ou até mesmo o escalonador, pode lhe tirar esse recurso?

a) Não Preemptivo.
b) Preemptivo.
c) Preemptivo auto-gestionado.
d) Não preemptivo auto-gestionado.

09. Arquivos armazenam dados. Para serem usados, esses dados devem ser lidos para a memória do computador. Quando nos referimos a métodos de acesso a dados, podemos destacar duas formas de obtenção destes junto a unidades de armazenamento. Quais são esses dois métodos?

a) Acesso indireto e acesso direto.
b) Acesso indireto e acesso sequencial.
c) Acesso por ordenação e acesso sequencial.
d) Acesso direto e acesso sequencial.

10. Um processo pode gerar um novo processo. Se o fizer, o processo criador é denominado processo-pai, e o processo criado denominado processo-filho. Quando o processo pai é destruído, os sistemas operacionais respondem de uma dessas duas formas:

a) destrói todos os processos-filho desse processo-pai ou vincula os processos-filho a outro processo-pai.
b) destrói todos os processos-filho desse processo-pai ou permite que os processos-filho prossigam de forma independente.
c) permite que os processos-filho prossigam de forma independente ou vincula os processos-filho a outro processo-pai.
d) cria processos RPI ou permite que os processos-filho prossigam de forma dependente.

11. Um processo no Linux interage com o sistema operacional via chamada ao sistema. Qual o procedimento que ela realiza quando a chamada ao sistema é do tipo *fork*?

a) Termina o processo que está chamando.
b) Faz com que o processo que está chamando fique bloqueado até que o processo-filho termine sua execução.
c) Carrega as instruções e dados de um processo no seu espaço de endereço em um arquivo.
d) Cria um processo filho.

12. Qual comando é utilizado no Linux para alterar as permissões de arquivos e diretórios?

a) chmod.
b) cat.
c) pwd.
d) du.

13. Um processo ou *thread* está em estado de *Deadlock* (travado) se estiver à espera de um determinado evento que não ocorrerá. Porém, para que aconteça o *Deadlock,* quatro condições necessitam ocorrer. Quanto a condição de exclusão mútua, essa consiste em

a) dois ou mais processos ficarem travados em uma ‘cadeia circular’, na qual um processo está aguardando um ou mais recursos que o outro processo detém e vice-versa.
b) uma vez que o processo obtém um recurso, o sistema não poder retirá-lo do controle até que ele (o processo) tenha terminado de utilizar o recurso.
c) um recurso poder ser adquirido exclusivamente por um único processo por vez.
d) uma *thread* não poder gerar processos filhos.

14. Visando obter uma maior taxa de transferência entre a placa-mãe e as placas de vídeo, foi desenvolvido um barramento especial para a comunicação com o vídeo. Que barramento é este?

a) PCI.
b) PCI-Express.
c) ISA.
d) AGP.

15. Um computador paralelo no qual todas as CPU´s compartilham uma memória comum é denominado

a) multicomputador.
b) multiacesso.
c) multiprocessador.
d) microcontrolador.

16. Leia as afirmações abaixo sobre o esquema RAID.

I. o RAID consiste em um agrupamento de unidades de discos lógicos, visto pelo sistema operacional como uma única unidade de disco físico.
II. os dados são distribuídos pelas unidades de discos lógicos do agrupamento.
III. a capacidade de armazenamento redundante é utilizada para armazenar informação de paridade, garantindo a recuperação dos dados em caso de falha em algum disco.

Está(ão) correta(s) as afirmativas

a) I, apenas.
b) I e II, apenas.
c) II e III, apenas.
d) III, apenas.

17. Sendo um bom exemplo de interface para dispositivos periféricos externos, a SCSI tem como características:

I. ser uma interface padrão para unidades de CD-ROM, equipamentos de áudio e dispositivos externos de armazenamento em massa.
II. usar uma interface paralela com 8 linhas de dados somente.
III. ser geralmente referida como um barramento, embora os dispositivos sejam, de fato, conectados em uma cadeia circular.

Estão corretas as afirmativas

a) I e II,apenas.
b) I e III, apenas.
c) I, II e III.
d) II e III, apenas.

18. O *Firewire* é considerado um barramento serial especificado pelo padrão IEEE 1394. Com relação a este barramento, é correto afirmar que

a) tem sido usado não apenas em sistemas de computação, mas também em produtos eletrônicos, tais como câmeras digitais e televisores.
b) apresenta algumas desvantagens com relação ao SCSI, principalmente pelo fato de utilizar transmissão serial (um bit de cada vez) e não paralela.
c) oferece várias interfaces de E/S para manipular um grande número de dispositivos.
d) usa uma configuração para conexão de dispositivos em forma de barramento, possibilitando a conexão de até 9 dispositivos.

19. A técnica de Acesso Direto à Memória (DMA) envolve um módulo adicional no barramento do sistema. Este módulo ou controlador de DMA é capaz de

I. imitar o processador e, de fato, controlar o seu sistema.
II. permitir que o módulo de DMA possa transferir dados diretamente de para a memória por meio do barramento do sistema.
III. usar o barramento do sistema apenas quando este não estiver sendo usado pelo processador, bem como forçar o processador a suspender sua operação temporariamente.

Estão corretas a(as) afirmativa(as)

a) I e II, apenas.
b) II e III, apenas.
c) I e III, apenas.
d) I, II e III.

20. À medida que os sistemas de computação evoluíram, seus componentes individuais tornaram-se mais complexos e sofisticados, principalmente aqueles voltados para funções de E/S. Parte desta evolução deve-se também

a) aos barramentos SCSI e *firewire*.
b) à CPU, que controla diretamente cada dispositivo periférico.
c) aos sistemas RAID.
d) ao uso de instruções *pipeline* na memória RAM dos computadores.

21. A alternativa correta onde se descreve os estágios de um *pipeline* que reduz o tempo de execução de 9 instruções de 54 para 14 unidades de tempo é:

a) Busca de Instrução; Decodificação da Instrução; Cálculo de Operandos; Busca de Operandos; Execução da Instrução e Escrita de Operando.
b) Busca de Instrução e Execução da Instrução.
c) Busca de Operandos; Contador de programa; Registrador de Instrução; Registrador de Endereçamento à Memória e Registrador de Armazenamento Temporário de Dados.
d) Busca de Instrução; Registro da Instrução; Armazenamento da Instrução e Execução da Instrução.

22. Quase todos os computadores possuem um mecanismo pelo qual componentes distintos do processador (E/S, memória) podem pausar a sequência normal de execução de suas instruções. A alternativa que corresponde a este mecanismo é

a) *Delay*
b) Interrupção
c) *DeadLock*
d) *Fork*

23. Um barramento do sistema contém, tipicamente, de 50 a 100 linhas distintas. Cada linha possui uma função. Embora existam diferentes projetos de barramentos, estas linhas podem ser classificadas em três grupos funcionais. A alternativa correta que corresponde a esta classificação é:

a) Linhas de dados, linhas de endereço e linhas de controle.
b) Linhas de instruções, linhas de operandos e linhas de execução.
c) Linhas de código, linhas de dados e linhas de execução.
d) Linhas de dados, linhas de instruções e linhas de execução.

24. A sigla do barramento de dados conhecido como PCI significa

a) interrupção de Controle do Processador.
b) informação de Controle de Periféricos.
c) interconexão de Componentes Periféricos.
d) interrupção de Componentes Periféricos.

25. São características da tecnologia de memória de semicondutor flash:

a) Memória de leitura e escrita; Mecanismo de apagamento por luz UV; Volátil.
b) Memória apenas de leitura; Mecanismo de apagamento eletrônico; Não-Volátil.
c) Memória de leitura e escrita; Mecanismo de apagamento eletrônico; Volátil.
d) Memória principalmente de leitura; Mecanismo de apagamento por luz UV; Não-Volátil.

26. Com o aumento da densidade dos circuitos integrados, foi possível incluir a memória *cache* na mesma pastilha do processador. Com base nesta afirmação, a alternativa que representa as vantagens ocasionadas por este processo é

a) redução da atividade do processador no barramento externo e, portanto, uma diminuição do tempo de execução e aumento do desempenho global do sistema.
b) redução da atividade do processador no barramento interno, ocasionando a diminuição do tempo de execução e aumentando o desempenho global do sistema.
c) rantagens econômicas, visto que houve uma redução nos custos de produção do processador.
d) o acesso mais rápido aos barramentos que têm um estado de espera de tempo nulo.

27.
```
int *p, i[10];
p = i;
p[5] = 100;
*(p+5)  = 100;
```

Considere o fragmento de um programa escrito na linguagem de programação C acima.
A afirmativa correta e relativa a esse fragmento é

a) a atribuição p[5] = 100 coloca o valor 100 no quinto elemento da matriz *i* e a atribuição *(p+5) = 100 faz com que o ponteiro p aponte para o valor 105 na memória.
b) a atribuição p[5] = 100 está correta, mas a operação *(p+5) = 100 não pode ser feita com variáveis do tipo ponteiro.
c) os dois comandos de atribuição, p[5] = 100 e *(p+5) = 100, colocam o valor 100 no sexto elemento da matriz *i*.
d) a atribuição *(p+5) = 100 faz com que o ponteiro *p* receba o endereço 105 da memória.

28. Considerando as funções de manipulação de *strings* mais comuns na linguagem C, **NÃO** é verdadeira a afirmativa:

a) A função strlen(s1) retorna o tamanho de s1.
b) A função strchr(s1, ch) inclui o caracter ch na string s1.
c) A função strstr(s1, s2) retorna um ponteiro para a primeira ocorrência de s2 em s1.
d) A função strcat(s1, s2) concatena s2 ao final de s1.

29. Avalie os seguintes comandos de expressões da linguagem de programação C e assinale a alternativa **FALSA**

a) func( ); /* uma chamada a uma função */
b) a = b + c; /* um comando de atribuição */
c) b + f( ); /* este comando é inválido na linguagem C */
d) i--; /* comando de atribuição */

30.
```
do {
        scanf("%d", &num);
        } while (num > 100);
```

Analise o fragmento do código de um programa na linguagem C acima e assinale a afirmativa correta:

a) O laço *do-while* repete até que a condição se torne verdadeira.
b) O laço *do-while* verifica a condição sempre no início do laço.
c) O laço *do-while* lerá números inteiros do teclado até que encontre um número menor ou igual a 100.
d) O laço *do-while* lerá números inteiros do teclado até que encontre um número maior do que 100.

31. Considerando as estruturas de repetição na linguagem C, **NÃO** é correto afirmar:

a) O comando `for( ; ; )` é um comando válido na linguagem C utilizado para criar um laço infinito.
b) O seguinte laço *for* remove os primeiros espaços da *stream* apontada por str.
`for ( ; *str == ‘  ‘ ; str++).`
c) O seguinte laço *while* verifica se ch não é igual a A:
`while ( ch != ‘A’ ) ch = getchar().`
d) Uma vez digitado o caracter A, a condição se torna falsa, porque ch fica igual a A, e o laço continua. Considerando o mesmo laço da alternativa C:
`while ( ch != ‘A’ ) ch = getchar().`

32. Avalie as tabelas seguintes que mostram a precedência relativa dos operadores relacionais e lógicos e marque a opção que representa a precedência correta dos operadores na linguagem C.

a)

Maior	> >= < <=
	== !=
	!
	&&
Menor	||

b)

Maior	!
	> >= < <=
	== !=
	&&
Menor	||

c)

Maior	== !=
	> >= < <=
	!
	||
Menor	&&

d)

Maior	> >= < <=
	!
	== !=
	||
Menor	&&

33.
```
int func(int n)
  {
    int r;
    if (n == 1) return (1);
    r = func(n-1) * n;
    return(r);
  }
```

Considerando conceitos de uma função recursiva na linguagem C descrito acima, a alternativa correta a respeito do funcionamento do fragmento de código é

a) Tendo n = 3, a função retornará como resposta final 6.
b) Tendo n = 4, a função retornará como resposta final 16.
c) Tendo n = 3, a função retornará como resposta final 9.
d) Tendo n = 2, a função retornará como resposta final 1.

34.
```
struct {
        char nome[30];
        char endereco[40];
        char cidade[20];
        char uf[2];
        unsigned long int cep;
   } info;
```

Observe o fragmento de código na linguagem C acima e marque a alternativa correta com relação à esta estrutura:

a) Está correta porque declara apenas uma variável.
b) Está incorreta, pois sua sintaxe obriga a existência de um identificador.
c) Está incorreta, pois o tipo *unsigned long* não é permitido na linguagem C.
d) Está correta porque declara seis variáveis, sendo elas: nome, endereço, cidade, uf, cep e info.

35. Sempre que um programa em C começa a sua execução, três *streams* são abertas automaticamente. São elas:

a) A entrada padrão (*stdin*), a saída padrão (*stdout*) e a saída de erro padrão (*stderr*).
b) A entrada e saída padrão (*stdio.h*) e a saída de erro padrão (*errno.h*).
c) A entrada padrão (*stdio.h*), a saída padrão (*stdlib.h*) e a saída de erro padrão (*errno.h*).
d) Qualquer *stream* que seja definida pelo desenvolvedor.

36.
```
void main(void)
{
int t, i, num[3][4];

for(t=0; t<3; ++t)
   for(i=0; i<4; i++)
      num[t][i] = (t*4)+i+1;
}
```

Analise o fragmento de código de um programa na linguagem C expresso acima, é correto afirmar que o código

a) Não funcionará corretamente, uma vez que os índices *t* e *i* deveriam começar com o número 1.
b) Carrega uma matriz bidimensional com os números de 1 a 12.
c) Atribui à matriz *num* números inteiros de 3 dígitos nas suas quatro posições.
d) Carrega a matriz *num* com os valores 1, 2, 3, 4 no laço interno e 6, 7, 8 e 9 no laço externo.

37. A função *fclose*(), na linguagem C, fecha uma *stream* que foi aberta por meio da chamada *fopen*(). Esta função escreve qualquer dado que ainda permaneça no *buffer* de disco no arquivo e, então, fecha o arquivo em nível de sistema operacional. O que pode ocasionar se houver uma falha ao fechar uma *stream?*

a) Nenhum tipo de problema, uma vez que as *streams* são automaticamente fechadas, sendo esta característica uma vantagem em sua utilização.
b) Perda de dados; porém, os arquivos se mantêm intactos.
c) Perda de dados, arquivos destruídos e possíveis erros intermitentes no programa.
d) O próprio sistema operacional se encarregaria do fechamento da *stream*, no caso de não existir um comando para fechá-la.

38. O fragmento de código que contém a declaração da *string* **str** **`char str[14] = "Eu gosto de C";`** inicializa **str** com a frase "Eu gosto de C". Sobre esta declaração é correto afirmar que

a) é o mesmo que escrever char str[14] = {'E','u',' ','g','o','s','t','o',' ','d','e',' ','C','\0'};
b) o acesso a str[4] retornará o caracter "g".
c) a declaração da string deveria ser str[13] e não str[14].
d) não podemos declarar e imediatamente inicializar uma *string*.

39. Todas as variáveis utilizadas para receber valores por meio de *scanf*() devem ser passadas por seus endereços. O que isto significa?

a) Que esta é a maneira da linguagem C criar uma chamada por valor, o que permite a uma função alterar o conteúdo de um argumento.
b) Que para ler um inteiro para a variável cont seria usada a chamada `scanf("%d", cont).`
c) Que esta é a maneira da linguagem C criar uma chamada por referência, o que permite a uma função alterar o conteúdo de um argumento.
d) Que para ler uma *string str* poderemos utilizar a chamada `scanf("%s", &str).`

40. Os especificadores de formato de entrada são precedidos por um sinal % e informam à *scanf*() que tipo de dado deve ser lido imediatamente após. Com base nesta informação, declarando-se uma variável do tipo *string* e considerando que uma *string* é uma cadeia de caracteres, qual o especificador de formato que deveríamos utilizar?

a) %s
b) %c
c) %d
d) %f