MIEIC **Sistemas Operativos – 2020/21** **FEUP**

Prova com consulta apenas da documentação fornecida **Exame da Época Normal**
Duração: 1,5 horas **8.Junho.2021**
Cotação máxima: 50 pontos ; peso na nota final da disciplina: 50%
Estrutura da prova: escolha múltipla

Utilização: para cada pergunta só há uma resposta correcta; indique-a (com a letra correspondente) na folha de respostas, completando uma tabela semelhante à que se segue; se não souber a resposta correcta, nada preencha ou faça um traço nessa alínea.

Cotação: cada resposta certa vale 1 ponto; cada resposta errada vale – 0,5 ponto (note o sinal menos!); cada resposta ambígua, ininteligível ou não assinalada vale 0 ponto. O total é 50 pontos, que irão equivaler a 20 valores.

Se desejar, *poderá* fazer um pequeno comentário em alguma pergunta que lhe pareça ambígua. No caso de a sua resposta ser considerada errada, o seu comentário *poderá* ajudar a perceber a sua ideia e *poderá* minorar a penalização da classificação.

1	2	3	4	5	6	7	8	9	10	11	12			13	14	15	16	17	18
											a	b	c						

19	20			21	22	23	24	25			26	27	28	29	30	31	32	33	34
	T1	T2	T3					a	b	c									

35			36			37	38	39	40
.1	.2	.3	a	b	c				

---

1. Nas operações efectuadas com o comando: "`gcc -Wall -o exec prog.c`" para a criação de um executável,

A) foi utilizada uma biblioteca estática.

B) foi utilizada uma biblioteca dinâmica.

C) não foi utilizada qualquer biblioteca.

2. O utilitário `make` funciona mediante regras de dependência e de datação de ficheiros.

A) Certo! Essencialmente, é isso que traduz o funcionamento de `make`.

B) Certo! Mas tais regras só podem ser utilizadas no desenvolvimento de programas.

C) Errado! O que importa é o que estiver estipulado nas primeiras linhas da *Makefile*.

3. *Shell*, em sistemas operativos, é

A) um mero programa muito utilizado em Unix.

B) um programa que faz de interlocutor entre o utilizador e o sistema operativo.

C) uma linguagem de programação compilada

4. Um sistema operativo é um "gestor de recursos". Isso quer dizer que:

A) permite, por exemplo, a partilha da memória disponível no computador.

B) faz a gestão dos recursos que as aplicações lhe submeterem.

C) é apropriado a programas gestão (de bases de dados, por exemplo).

5. Sistemas multi-programação e sistemas multi-utilizador

A) são sinónimos, pois têm o mesmo significado.

B) são antagónicos, pois não podem ambos qualificar um dado sistema operativo.

C) são coisas diferentes, apesar de normalmente virem associados.

6. Num sistema operativo, uma "chamada ao sistema"

A) é a maneira como um sistema operativo aceita pedidos.

B) é a maneira de se comunicar com um sistema do tipo Unix.

C) é a maneira como, num terminal, um utilizador pede um recurso.

7. O barramento (*bus*) de um computador

A) é o principal componente de computadores de elevado desempenho.

B) permite a interligação dos dispositivos electrónicos do computador.

C) é um dispositivo do tipo Entrada/Saída (I/O).

8. Um processo tem associado, pelo menos:

A) um programa, uma zona de memória partilhada e três ficheiros (*stdin, stdout, stderr*).

B) um programa, uma zona de memória e uma entrada na tabela de processos.

C) um programa, um processador e um dono.

9. Na sequência da invocação da chamada `fork()`, o novo processo irá começar imediatamente a executar.

A) Não, pois o processo-pai é sempre o primeiro a executar.

B) Sim, se for escolhida a opção certa para `fork()`.

C) Não é possível afirmar isso com certeza.

10. Considere o excerto de código junto. O que será impresso na saída padrão?

A) -1 1

B) 0 0

C) 1 -1

```
int global = 0;
void main() {
   int pid = fork();
   if (pid == 0) { global++; }
   else  { wait(NULL); global--; }
   printf("%d ",global);
}
```

11. Os três principais estados de um processo reconhecido pelo sistema operativo são:

A) prontidão (*ready*), bloqueamento (*blocked*), execução (*running*).

B) dormência (*sleeping*), bloqueamento (*blocked*), execução (*running*).

C) prontidão (*ready*), bloqueamento (*blocked*), sinalização (*signaling*).

12. Considere o seguinte excerto de código, relativo ao tratamento de sinais. O programa é executado e não há erros na invocação dos serviços de sistema.

```
void handler(int signo) { printf("Hello "); }

void main() {
   struct sigaction new;
   sigset_t smask; sigemptyset(&smask);
   new.sa_handler = handler; new.sa_mask = smask; new.sa_flags = 0;
   sigaction(SIGUSR1, &new, NULL);
   pause(); printf("World.\n");
}
```

O que aparecerá na saída padrão assim que for enviado ao processo o sinal

a) `SIGUSR1`?

A) Hello World.

B) World. Hello

C) World.

b) `SIGINT`?

A) [*nada e o programa continua*]

B) [*nada e o programa termina*]

C) World.

c) `SIGCHLD`?

A) [*nada e o programa continua*]

B) [*nada e o programa termina*]

C) World.

13. Deve optar-se por programação com *threads* relativamente à programação por processos,

A) quando se pretender escrever um servidor.

B) quando se pretender facilitar a troca de informação entre as partes do programa.

C) quando se pretender aumentar o isolamento entre as partes do programa.

14. Na criação de um *thread*, com `pthread_create()`, terá de se especificar uma função que irá estar ligada à execução do novo *thread*.

A) É certo, mas essa função tem de ter um protótipo do tipo `void* function (void*)`.

B) É certo, mas essa função apenas necessita de receber como argumentos um apontador para `void`.

C) É certo, mas não há qualquer restrição ao tipo do valor de retorno da função.

15. Para se poder atender um grande número de processos, basta usar um *quantum* pequeno, que melhora a rotatividade de utilização do processador.

A) Isso é importante, especialmente porque o custo das mudanças de contexto é mínimo.

B) Isso parece precipitado, pois há que considerar o custo das mudanças de contexto.

C) Isso só pode ser feito nos sistemas de grande utilização do processador.

16. Uma condição de competição (*race condition*) surge quando dois ou mais processos

A) tentam partilhar de forma eficiente um mesmo conjunto de recursos.

B) tentam partilhar de forma sincronizada um mesmo conjunto de recursos.

C) tentam partilhar de forma dessincronizada um mesmo conjunto de recursos.

17. Pode um processo fora de uma zona crítica impedir outro de entrar nessa zona?

A) Pode, mas não deve, porque o mecanismo de reserva antecipada é complicado.

B) Pode, e deve, pois isso até melhora a eficiência do sistema de exclusão mútua partilhada.

C) Pode, mas não deve, pois estaria a tornar o sistema ineficiente e com tendência a encravar.

18. Em programação POSIX, uma variável de condição deve ser sempre usada em associação com um mutex.

A) Não, tal não é necessário, pois a variável de condição é suficiente.

B) Talvez, se não se puder usar outro tipo de semáforo.

C) Sim, para garantir o acesso exclusivo à zona onde se vai usar a variável de condição.

19. Um encravamento (*deadlock*) típico dá-se quando vários processos de um grupo se vêm impossibilitados de prosseguir a sua actividade normal por

A) todos terem esgotado o seu quantum de processador.

B) todos estarem aguardando a libertação do processador.

C) todos necessitarem de recursos detidos por diferentes processos do grupo.

20. Um programa é composto por 3 actividades (*threads*) , T1, T2 e T3 que, de vez em quando, invocam uma função `work(...)` cujos argumentos, em número variável, identificam os recursos necessários à sua execução, que devem ser acedidos de forma exclusiva.
Assim, por exemplo, T3 poderá a certa altura invocar `work(R2,R3)`, querendo isto dizer que deverá haver código que garanta que os recursos R2 e R3 são unicamente usados por T3 durante a execução da função.
Uma forma de conseguir isso, usando 2 mutexes, m2 e m3, seria:

```
lock(m2); lock(m3); work(R2,R3); unlock(m2); unlock(m3);
```

Supondo que cada um de 3 recursos R1, R2 e R3 é protegido individualmente por um mutex, respectivamente, m1, m2 e m3, escolha, para cada um dos *threads*, a opção mais apropriada que impeça o programa de encravar, executando com o máximo de eficiência.
NOTA: tenha em atenção que o que escolher para um *thread* pode afectar a escolha para os outros!

T1 :

A)
```
lock(m1); lock(m2);
lock(m3);
 work(R1,R2);
unlock(m3); unlock(m2);
unlock(m1);
```

B)
```
lock(m1); lock(m2);
  work(R1,R2);
unlock(m2); unlock(m1);
```

C)
```
lock(m2); lock(m1);
  work(R1,R2);
unlock(m2); unlock(m1);
```

T2 :

A)
```
lock(m1); lock(m3); lock(m2);
  work(R1,R2,R3);
unlock(m1); unlock(m3);
unlock(m2);
```

B)
```
lock(m3); lock(m1); lock(m2);
  work(R1,R2,R3);
unlock(m1); unlock(m2);
unlock(m3);
```

C)
```
lock(m1); lock(m2); lock(m3);
  work(R1,R2,R3);
unlock(m3); unlock(m2);
unlock(m1);
```

T3 :

A)
```
lock(m3); lock(m2);
  work(R2,R3);
unlock(m1); unlock(m3);
```

B)
```
lock(m2); lock(m3);
  work(R2,R3);
unlock(m3); unlock(m2);
```

C)
```
lock(m3);
  work(R2,R3);
unlock(m3);
```

21. A hierarquia de memória de um computador

A) consiste em registos de processador, zonas temporárias de sistema (*buffers*) e memória dinâmica.

B) existe porque se deseja usar vários tipos diferentes de componentes de memória.

C) é necessária porque não se consegue obter a baixo custo toda a memória rápida que se deseja.

22. Considere um sistema em que a memória, numa dada ocasião, apresenta os seguintes espaços vazios (*holes*), por ordem crescente de posição: 10KiB, 4KiB, 20KiB, 18KiB, 7KiB e 9KiB. Nessa situação, foram feitos 3 pedidos sequenciais de memória: 10KiB, 12KiB e 9KiB, após o que a lista ordenada de espaços vazios ficou: 4KiB, 20KiB, 6KiB e 7KiB. Qual foi o método de alocação utilizado?

A) Primeiro a servir (*first fit*).

B) Melhor a servir (*best fit*).

C) Pior a servir (*worst fit*)

23. Uma entrada de uma tabela de páginas de um sistema de memória virtual contém, entre outros dados,

A) um número de moldura (*frame*), um bit de presença e um bit de alteração.

B) um número de moldura (*frame*), um número de página e um bit sinalização.

C) um número de página, um bit de falha de página e um número de bloco de disco.

24. Num sistema de memória virtual, a escolha do algoritmo de substituição de página é especialmente importante por questões de

A) economia do espaço de memória disponível.

B) melhoria do desempenho do sistema.

C) minimização do desgaste do disco de apoio (*swap*).

25. Considere um sistema que implementa memória paginada com (apenas!) 4 quadros (*page frames*). A tabela junto mostra a informação de paginação acessível ao sistema operativo na altura em que uma decisão de libertação de quadro tem de ser tomada.
Qual das páginas mostradas dará lugar à nova página, quando o algoritmo usado for o da "página que...

quadro	pág.	carregada há (*clock ticks*)	referenciada há (*clock ticks*)	R *flag*	M *flag*
0	25	136	122	1	0
1	81	290	260	0	0
2	19	145	145	1	1
3	97	280	250	0	1

a) ...não é usada recentemente" (NRU, Not Recently Used)?

A) 19

B) 25

C) 81

b) ...entra primeiro, sai primeiro" (FIFO)?

A) 25

B) 81

C) 97

c) ...entra primeiro, sai primeiro; com 2ª chance" (FIFO, second chance)?

A) 25

B) 81

C) 97

26. Nos sistemas reais que usam paginação de memória, a *cache* de tradução de endereços de memória (*Translation Lookaside Buffer*, TLB), que é exemplificada na figura junto,

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

A) acelera o acesso à memória porque mantém informação essencial de um conjunto de registos da Tabela de Páginas.

B) acelera o acesso à memória porque substitui integralmente a Tabela de Páginas nos acessos de leitura.

C) é especialmente eficaz na aceleração do acesso à memória quando se dá uma "falha de página" (*page fault*).

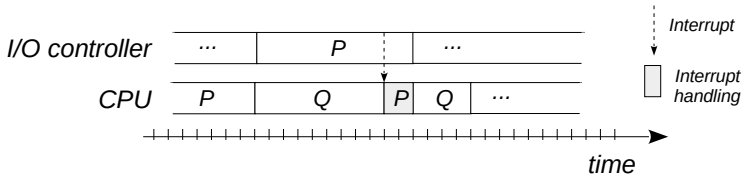
27. Os dispositivos de Entrada/Saída (I/O) dos computadores modernos costumam

A) ter uma componente de *software* de controlo.

B) poder aceder facilmente ao processador principal.

C) ser do tipo bloco (*block devices*), de acesso aleatório.

28. Na figura junto (tirada das folhas de apoio às aulas teóricas), é patente a vantagem do acesso a dispositivos de Entrada/Saída (I/O) usando interrupções.



A) Sim, porque é o dispositivo que fica com todo o trabalho de Entrada/Saída.

B) Sim, porque o processador precisa ser usado por outro processo mais prioritário.

C) Sim, porque outro processo pode aproveitar o processador, sem prejuízo do acesso de Entrada/Saída.

29. O "*tick*" do relógio interno de um computador está associado a

A) um segundo.

B) um sinal de alarme.

C) uma interrupção de relógio.

30. Os discos de estado sólido (SSD, Solid State Drive) são muitas vezes preferidos a discos clássicos, magnéticos (HDD, Hard Disk Drive) porque, para igual capacidade de armazenamento,

A) são mais rápidos.

B) são mais baratos.

C) são mais programáveis.

31. Dependendo do contexto, todos os nomes de ficheiro que se apresentam a seguir podem ser utilizados para identificar o mesmo ficheiro. Qual deles é um nome absoluto?

A) `/lib/dict`

B) `./lib/dict`

C) `../lib/dict`

32. Os ficheiros costumam ser guardados nos discos em blocos de igual tamanho. O valor do bloco afecta a rapidez de leitura e o aproveitamento do espaço de armazenamento.

A) Sim, pois com grandes blocos o espaço é melhor aproveitado.

B) Sim, pois com grandes blocos as leituras são mais rápidas.

C) Sim, pois com grandes blocos as leituras são mais rápidas e o espaço melhor aproveitado.

33. Os directórios têm, no sistema de ficheiros, uma estrutura interna específica.

A) Sim, e é semelhante à dos ficheiros regulares estruturados.

B) Sim, por isso é que têm chamadas ao sistema específicas, como `opendir()`.

C) Não, porque são apenas sequências de bytes.

34. Em Unix, um nó de indexação (*i-node*) contém os atributos de um ficheiro e os seus blocos de dados.

A) Sim, o *i-node* contém o nome do ficheiro, a data da sua criação, os seus blocos de dados, etc.

B) Só será assim se o ficheiro for pequeno; se for grande, o *i-node* não tem blocos de dados.

C) Não, o *i-node* além dos atributos só tem endereços que conduzem aos blocos de dados.

35. A função `search()`, cujo esqueleto é apresentado a seguir, é a parte essencial de um programa que, recursivamente, percorre a directoria corrente (ou de trabalho) e as suas sub-directorias identificando todos os *links* simbólicos e apresentado-os, pelo nome simples, na saída padrão.

```
void search() {
   DIR *dir = opendir(".");
   struct dirent *f;
   while ((f = ( .1 )) != NULL) {
      if ( .2 )
         printf("%s\n",f->d_name);
      else if (f->d_type == DT_DIR && strcmp(f->d_name,"..") && strcmp(f->d_name,"."))
         { .3 }
   }
} //search()
```

Cada local assinalado por '`.n`' (por exemplo, `.2`) deve ser preenchido com código apropriado, que deverá escolher , de entre as opções a seguir mostradas para cada um.

SUGESTÃO IMPORTANTE: tente completar o código de cada local antes de escolher de entre os extractos mostrados!

.1 :	**.2 :**	**.3 :**
A) `open(dir);`	A) `f->d_type == DT_LNK;`	A) `chdir(f->d_name); search(); chdir(dir);`
B) `readdir(dir);`	B) `f->d_type != DT_LNK;`	B) `chdir(f->d_name); search(); chdir("..");`
C) `read(dir);`	C) `f->d_type && DT_LNK;`	C) `chdir(f->d_name); search(); chdir(".");`

36. Considere os seguintes excertos simplificados de código, que respeitam a uma aplicação cliente-servidor do género da usada no 2º mini-projecto. Suponha também que não se dão situações inesperadas de erro (além do que estiver previsto nas perguntas).

// servidor

```
mkfifo(fifoname, 0666);
fifo = open(fifoname, O_RDONLY);
while (read(fifo, &mgs, sizeof(msg)) > 0)
       { ... // processa msg }
close(fifo);
unlink(fifoname);
```

// cliente

```
fifo = open(fifoname,O_WRONLY|O_NONBLOCK
);
       { ... // constrói msg}
write(fifo, &mgs, sizeof(msg);
close(fifo);
```

a) Suponha que o servidor arrancava primeiro, uns segundos antes do cliente. Quando o cliente arrancasse,

A) o servidor estaria bloqueado no `mkfifo()`.

B) o servidor estaria bloqueado no `open()`.

C) o servidor estaria bloqueado no `read()`.

b) Se a FIFO não existir quando o `open()` no cliente é executado

A) o programa dá imediatamente erro no `open()`.

B) o programa dá imediatamente erro no `write()`.

C) o programa bloqueia no `open()` até o FIFO ser criado.

c) Suponha que vários clientes (do tipo mostrado) eram lançados concorrentemente. Para garantir que o servidor não recebia mensagens de conteúdo misturado,

A) bastava que `sizeof(msg)` fosse inferior a `PIPE_BUF`.

B) não era preciso ter-se um cuidado especial.

C) era sempre necessário usar-se mutexes no acesso ao FIFO.

37. Um sistema operativo de tempo-real é usado especialmente em aplicações que processam dados à medida que lhe são entregues, sem atrasos devidos a armazenamento temporário (e.g. em *buffers*).

A) Assim é porque o processamento desses dados tem de ser feito num tempo específico.

B) Assim é porque o utilizador de tais aplicações não quer estar à espera: por exemplo, se liga um interruptor, a lâmpada do tipo "*smart*" tem de acender de imediato.

C) Assim é porque tais aplicações operam com equipamento muito simples sem provisão para memória temporária (*buffers*).

38. Das muitas ameaças à segurança de utilização de um computador, a do tipo "cavalo de Tróia" pode ser evitada pelo sistema operativo.

A) Sim, é certo, porque o sistema operativo pode detectar a instalação do "cavalo".

B) Talvez possa ser mitigada, caso o administrador faça uma configuração adequada do sistema.

C) Não, este é um tipo de ameaça em que só o utilizador pode evitar o problema.

39. Num terminal de texto em Unix, deu o comando "`ls -l`" e obteve

```
-rwxr-xr-x 1  user grp   8335  jan 18 2019  test1
```

A) Concluiu que o sistema devia usar algo semelhante a listas de controlo de acesso (ACL) na protecção dos ficheiros.

B) Concluiu que o sistema devia usar algo semelhante a credenciais de acesso (*capabilities*) na protecção dos ficheiros.

C) Nada lhe permitiu concluir sobre a forma como o sistema guardava a Matriz de Protecção.

40. Usamos Unix/Linux na disciplina porque, entre outras coisas:

A) Tem uma interface "seca" para o utilizador, o que é um bom treino para um futuro Engenheiro Informático.

B) Tem uma interface programática excepcionalmente simples e reconfigurável, característica de um sistema monolítico.

C) Pode ser explorado sem restrições de licenciamento e de acesso aos pormenores de implementação.

Respostas e Comentários noutra folha, que é fornecida.