Submeter | Limpar

1.

A diferença fundamental entre arquiteturas de 32 e 64 bits consiste no facto de as segundas permitirem:

a. ☒ a execução de processos com espaços de endereçamento maiores.
b. ☐ manter informação sobre mais processos na memória física.
c. ☐ transferir processos completos para a memória física.

2.

No CPU, a componente designada por "Memory Management Unit" (MMU) é responsável por:

a. ☐ atribuir espaço aos processos que aguardam execução mantendo a integridade da memória física.
b. ☒ garantir a integridade da memória e traduzir endereços virtuais em endereços na memória física.
c. ☐ detectar acessos indevidos a zonas de memória e transferir controlo para o sistema operativo.

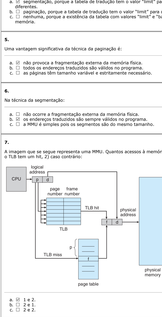
3.

A priori, uma desvantagem das técnicas de segmentação e paginação relativamente a técnicas que mapeiam processos completos em zonas contíguas de memória consiste no facto da tradução de endereços virtuais em endereços físicos:

a. ☐ não poder ser feita pelo hardware (MMU).
b. ☒ envolver a consulta de tabelas em memória.
c. ☐ as MMUs respectivas serem muito complexas.

4.

A imagem que se segue representa uma MMU. O circuito suporta segmentação, paginação ou nenhuma das duas? Porquê?



a. ☒ segmentação, porque a tabela de tradução tem o valor "limit" para cada entrada, indicando que os blocos de memória podem ter tamanhos diferentes.
b. ☐ paginação, porque a tabela de tradução tem o valor "limit" para cada entrada, indicando que os limites das páginas estão a ser verificados.
c. ☐ nenhuma, porque a existência da tabela com valores "limit" e "base" por entrada indica que está guarda a posição de processos completos na memória.

5.

Uma vantagem significativa da técnica da paginação é:

a. ☒ não provoca a fragmentação externa da memória física.
b. ☐ todos os endereços traduzidos são válidos no programa.
c. ☐ as páginas têm tamanho variável e extremamente reconfigurável.

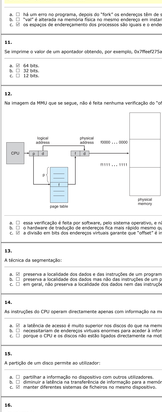
6.

Na técnica da segmentação:

a. ☐ não ocorre a fragmentação externa da memória física.
b. ☒ os endereços traduzidos são sempre válidos no programa.
c. ☐ a MMU é simples pois os segmentos são do mesmo tamanho.

7.

A imagem que se segue representa uma MMU. Quantos acessos à memória estão envolvidos na tradução de um endereço de memória virtual quando: 1) o TLB tem um hit, 2) caso contrário:



a. ☒ 1 e 2.
b. ☐ 2 e 1.
c. ☐ 2 e 2.

8.

Numa arquitectura de 32 bits, um sistema operativo com páginas de 4 Kibytes produz tabelas de páginas com quantas entradas?

a. ☐ 2^{20}.
b. ☒ 2^{32}.
c. ☐ 2^{12}.

9.

Uma vantagem/desvantagem, respectivamente, de usar páginas mais pequenas num sistema operativo, é:

a. ☒ diminuir a fragmentação interna / aumentar o "swapping" de páginas.
b. ☐ aumentar a localidade de instruções / aumentar a fragmentação interna.
c. ☐ aumentar a localidade de informação / aumentar a fragmentação externa.

10.

Como explica que o seguinte programa dê valores distintos para a variável "var" que está localizada no mesmo endereço?

```
/* includes & defines */

int main(int argc, char* argv[]) {
  int var = 0;
  int * ptr;

  ptr = fork();

  if (ptr == 0) {
    var = 0;
    printf("child: val = %d, at addr = %lx",var, &var);
    return EXIT_SUCCESS;
  } else {
    var = 1;
    wait(NULL);
    printf("parent: val = %d, at addr = %lx",var, &var);
    return EXIT_SUCCESS;
  }
}

$ gcc -Wall v1.c -o v1
$ ./v1
child: val = 0, at addr = 0x7ffeef175a38
parent: val = 1, at addr = 0x7ffeef175a38
```

a. ☐ há um erro no programa, depois do "fork" os endereços têm de ser diferentes.
b. ☒ "var" é alterada na memória física no mesmo endereço em instantes diferentes.
c. ☐ os espaços de endereçamento dos processos são iguais e o endereço é virtual.

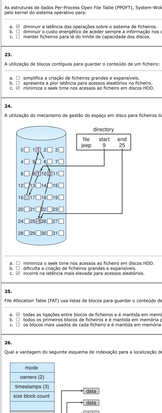
11.

Se imprimir o valor de um apontador obtendo, por exemplo, 0x7ffeef175a38, quantos bits tem a arquitectura em que o programa foi executado?

a. ☒ 64 bits.
b. ☐ 32 bits.
c. ☐ 12 bits.

12.

Na imagem da MMU que se segue, não é feita nenhuma verificação do "offset" do endereço para garantir que este cai sempre dentro da página porque:



a. ☐ essa verificação é feita por software, pelo sistema operativo, e não pelo hardware da MMU.
b. ☒ o hardware da tradução de endereços foi mais rápido mesmo que possa gerar erros ocasionais.
c. ☒ a dividir em bits dos endereços virtuais garante que "offset" é inferior ao tamanho duma página.

13.

A técnica da segmentação:

a. ☒ preserva a localidade dos dados e das instruções de um programa.
b. ☐ preserva a localidade dos dados mas não das instruções de um programa.
c. ☐ em geral, não preserva a localidade dos dados nem das instruções.

14.

As instruções do CPU operam directamente apenas com informação na memória física, e nunca com informação guardada em discos, porque:

a. ☒ a latência de acesso é muito superior nos discos do que na memória.
b. ☐ necessitariam de endereços virtuais enormes para aceder à informação.
c. ☐ porque o CPU e os discos não estão ligados directamente na motherboard.

15.

A partição de um disco permite ao utilizador:

a. ☐ partilhar a informação no dispositivo com outros utilizadores.
b. ☐ diminuir a latência na transferência de informação para a memória.
c. ☒ manter diferentes sistemas de ficheiros no mesmo dispositivo.

16.

Um volume é:

a. ☒ uma partição de um disco onde foi instalado um sistema de ficheiros.
b. ☐ o conjunto de todos os sistemas de ficheiros instalados num disco.
c. ☐ uma partição de um disco usada para transferir páginas da/para memória.

17.

A estrutura de dados utilizada para organizar a informação no Unix File System (UFS) e afins é:

a. ☐ uma lista ligada de ficheiros simples, sem directórios.
b. ☐ uma árvore com directórios nos nós e com ficheiros nas folhas.
c. ☒ um grafo dirigido acíclico com directórios nos nós e ficheiros nas folhas.

18.

Um sistema de ficheiros virtual (VFS) é:

a. ☐ é um sistema de ficheiros desenhado especificamente para ser utilizado em máquinas virtuais, suportando vários sistemas operativos.
b. ☐ é um sistema de ficheiros tradicional mantido em memória e que fornece ao utilizador a ilusão de um sistema de ficheiros em disco.
c. ☒ uma abstracção do sistema operativo que permite oferecer aos utilizadores uma API uniforme para operações sobre o sistema de ficheiros.

19.

A System-Wide Open File Table (SWOFT) é uma tabela mantida pelo kernel em memória contendo informação relativa a todos os ficheiros:

a. ☒ guardados actualmente nos discos.
b. ☐ abertos por processos em execução.
c. ☐ que alguma vez estiveram nos discos.

20.

Para cada processo, a Per-Process Open File Table (PPOFT) mantém informação sobre os ficheiros por ele abertos, porque há informação dos ficheiros específica aos processos, por exemplo:

a. ☐ a localização do cursor de leitura/escrita.
b. ☐ as permissões de leitura, escrita e execução.
c. ☒ a localização dos blocos respectivos em disco.

21.

A estrutura em C seguinte (exemplo do MACOS X) contém informação normalmente guardada:

```
struct stat {
    dev_t    st_dev;    /* device inode resides on */
    ino_t    st_ino;    /* inode's number */
    mode_t   st_mode;   /* inode protection mode */
    nlink_t  st_nlink;  /* number of hard links to file */
    uid_t    st_uid;    /* user ID of owner */
    gid_t    st_gid;    /* group ID of owner */
    dev_t    st_rdev;   /* device type (if inode special file inode */
    struct timespec st_atimespec;  /* time of last access */
    struct timespec st_mtimespec;  /* time of last data modification */
    struct timespec st_ctimespec;  /* time of last file status change */
    off_t    st_size;   /* file size, in bytes */
    quad_t   st_blocks; /* number of blocks for file */
    u_long   st_blksize;/* optimal file sys I/O ops blocksize */
    u_long   st_flags;  /* user defined flags for file */
    u_long   st_gen;    /* file generation number */
};
```

a. ☒ numa entrada da Per-Process Open File Table.
b. ☐ numa entrada da System-Wide Open File Table.
c. ☐ num File Control Block (inode no Unix).

22.

As estruturas de dados Per-Process Open File Table (PPOFT), System-Wide Open File Table (SWOFT) e File Control Block (FCB) são mantidas na memória pelo kernel do sistema operativo para:

a. ☒ diminuir a latência das operações sobre o sistema de ficheiros.
b. ☐ diminuir a latência envolvida em aceder a informação nos discos.
c. ☐ manter ficheiros para lá do limite de capacidade dos discos.

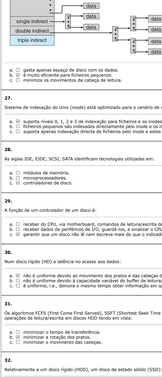
23.

A utilização de blocos contíguos para guardar o conteúdo de um ficheiro:

a. ☐ simplifica a criação de ficheiros grandes e expansíveis.
b. ☐ apresenta a pior latência para acessos aleatórios no ficheiro.
c. ☒ minimiza o seek time nos acessos ao ficheiro em discos HDD.

24.

A utilização do mecanismo de gestão do espaço em disco para ficheiros ilustrado na figura seguinte:



a. ☐ minimiza o seek time nos acessos ao ficheiro em discos HDD.
b. ☐ minimiza a latência nos acessos aleatórios.
c. ☒ ocorre na tabela mais elevada para acessos aleatórios.

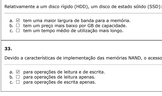
25.

File Allocation Table (FAT) usa listas de blocos para guardar o conteúdo de ficheiros mas é mais eficiente porque a tabela FAT contém:

a. ☒ todas as ligações entre blocos de ficheiros e é mantida em memória pelo kernel.
b. ☐ todos os primeiros blocos de ficheiros e é mantida em memória pelo kernel.
c. ☐ os blocos mais usados de cada ficheiro e é mantida em memória pelo kernel.

26.

Qual a vantagem do seguinte esquema de indexação para a localização de blocos de ficheiros em disco?



a. ☐ gasta apenas espaço de disco com os dados.
b. ☒ é muito eficiente para ficheiros pequenos.
c. ☐ minimiza os movimentos da cabeça de leitura.

27.

Sistema de indexação do Unix (inode) está optimizado para o cenário de muitos ficheiros de pequeno tamanho porque:

a. ☒ suporta níveis 0, 1, 2 e 3 de indexação para ficheiros e os inodes têm identificadores de 32 ou 64 bits.
b. ☐ ficheiros pequenos são indexados directamente pelo inode e os inodes têm identificadores de 32 ou 64 bits.
c. ☐ suporta apenas indexação directa de ficheiros pelo inode e estes têm identificadores no máximo de 8 bits.

28.

As siglas IDE, EIDE, SCSI, SATA identificam tecnologias utilizadas em:

a. ☐ módulos de memória.
b. ☐ microprocessadores.
c. ☒ controladores de disco.

29.

A função de um controlador de um disco é:

a. ☐ receber do CPU, via motherboard, comandos de leitura/escrita de dados e executá-los.
b. ☐ receber dados de periféricos de I/O, guardá-los, e analisar o CPU através de interrupção.
c. ☒ garantir que um disco não lê nem escreve mais do que o indicado pelos comandos do CPU.

30.

Num disco rígido (HD) a latência no acesso aos dados:

a. ☒ não é uniforme devido ao movimento dos pratos e das cabeças de leitura/escrita.
b. ☐ não é uniforme devido à capacidade variável do buffer de leitura/escrita do controlador.
c. ☐ é uniforme, i.e., demora o mesmo tempo obter informação em qualquer posição do disco.

31.

Os algoritmos FCFS (First Come First Served), SSFT (Shortest Seek Time First), SCAN e outros são usados pelo sistema operativo para organizar as operações de leitura/escrita em discos HDD tendo em vista:

a. ☐ minimizar o tempo de transferência.
b. ☒ minimizar a rotação dos pratos.
c. ☐ minimizar o movimento das cabeças.

32.

Relativamente a um disco rígido (HDD), um disco de estado sólido (SSD):

a. ☒ tem uma maior largura de banda para a memória.
b. ☐ tem um preço mais baixo por GB de capacidade.
c. ☐ tem um tempo médio de utilização mais longo.

33.

Devido a características de implementação das memórias NAND, o acesso a dados em discos de estado sólido (SSD) é uniforme:

a. ☒ para operações de leitura e de escrita.
b. ☐ para operações de leitura apenas.
c. ☐ para operações de escrita apenas.