Teoria

1) - Para que o acesso às páginas de um processo se torne mais eficiente, o ideal é que a tabela de páginas possa estar toda em memória principal. Quando a tabela de páginas é muito grande para ser totalmente carregada em memória há algumas formas de organização da memória que permitem um maior ganho em desempenho. Este ganho pode ser alcançado:

[x] utilizando múltiplos níveis de tabelas de páginas
[ ] utilizando segmentação
[ ] mantendo parte da tabela de página em memória, parte no disco
[ ] utilizando múltiplos níveis de indireção

2) - Assuma a seguinte ordem de chegada de processos no sistema: (números mais baixos de prioridade indicam alta prioridade)

#	Tempo de chegada	Tempo de execução	Prioridade
P0	0.0	1	4
P1	0.2	0.5	1
P2	1.0	2	2
P3	1.5	1	3

A seguinte afirmação é verdadeira:

[x] o algoritmo round-robin com quantum de tempo igual a 1 tem um tempo médio de espera maior do que o algoritmo SJF (Shortest Job First)
[ ] o algoritmo que utiliza prioridades vai escalonar os jobs na seguinte ordem: P1, P2, P3 e P0
[ ] o algoritmo round-robin com quantum de tempo igual a 1 executa estes processos em menos tempo do que o algoritmo SJF (Shortest Job First)
[ ] o algoritmo que utiliza prioridades vai escalonar os jobs na seguinte ordem: P0, P3, P2 e P1

3) - Em Unix, um socket do tipo "datagram":

[ ] estabelece uma conexão entre cliente e servidor
[ ] estabelece uma ligação do tipo correio
[x] estabelece uma ligação não fiável
[ ] estabelece uma ligação com uma porta específica

4) - Considere a seguinte implementação do problema dos leitores e escritores:

```
semaforo mutex=1;
semaforo bd=1;
int num_leitores=0;

escritor() {
  while (true) {
    produz_dado()
    down(&bd);
    escreve_dado_bd()
    up(&bd);
  }
}
```

```
leitor() {
  while (true) {
    down(&mutex);
    num_leitores++;
    if (num_leitores==1) down(&bd);
    up(&mutex);
    le_dado();
    down(&mutex);
    num_leitores--;
    if (num_leitores==0) up(&bd);
    up(&mutex);
    processa_dado_lido();
  }
}
```

Esta implementação dá prioridade:

[x] aos leitores
[ ] aos escritores
[ ] a muitos escritores
[ ] igual aos leitores e escritores

5) - O que é o tempo de resposta (response time) de um processo?
[ ] tempo de execução de um processo que é interativo
[ ] tempo para obter todas as respostas
[ ] tempo de execução de um processo que executa em "batch"
[x] tempo para obter a primeira resposta

6) - Considere um sistema de memória. Qual das seguintes afirmações é mais adequada:
[ ] fragmentação interna ocorre quando as partições são de tamanho muito grande (e.g., maior do que 32 KBytes)
[ ] fragmentação interna ocorre em sistemas que utilizam partições de tamanho variável
[x] fragmentação externa ocorre em sistemas que utilizam partições de tamanho variável
[ ] fragmentação externa ocorre quando as partições são de tamanho muito grande (e.g., maior do que 32 KBytes)

7) - Considere um sistema de gestão de memória virtual com paginação de um nível. O tamanho das páginas é de 1024 bytes, a memória física máxima é de 2 megabytes e o tamanho máximo do espaço de endereçamento é de 16 megabytes. Considere agora o trecho de tabela de páginas de um processo, mostrado abaixo.

Página	Moldura
0	4
1	8
2	16
3	17
4	9
...	...

A que endereço físico corresponde o endereço virtual 1524 e a que endereço virtual corresponde o endereço físico 10020?
[ ] 1524, 10020
[x] 8692, 4900
[ ] 8692, 10020
[ ] 1524, 4900

8) - Considere um sistema de ficheiros. Qual das seguintes afirmações é mais adequada:
[ ] quanto menor o tamanho de um bloco do sistema de ficheiros pior é a taxa de utilização do espaço em disco
[x] quanto menor o tamanho de um bloco do sistema de ficheiros maior é o "overhead" de entrada e saída
[ ] quanto maior o tamanho de um bloco do sistema de ficheiros maior é o "overhead" de entrada e saída
[ ] quanto maior o tamanho de um bloco do sistema de ficheiros melhor é a taxa de utilização do espaço em disco

9) - Considere um sistema que utiliza endereços virtuais de 32 bits e páginas de tamanho 4 KBytes. O programa e os dados iniciam na página de endereço mais baixo. A pilha inicia na página de endereço mais alto. Quantos bits são necessários para endereçar toda a tabela de páginas para este programa?

[ ] 4 bits
[ ] 32 bits
[x] 20 bits
[ ] 12 bits

10) - Considere um sistema com 4 molduras de memória. O tempo de primeiro acesso, tempo do último acesso e os bits R e M para cada página são mostrados na tabela abaixo:

Página	1º acesso	último acesso	R	M
0	126	280	1	0
1	230	265	0	1
2	140	270	0	0
3	110	285	1	1

Qual página será substituída pelos algoritmos NRU, FIFO, LRU e segunda tentativa, respectivamente?

[x] 2, 3, 1, 2
[ ] 1, 3, 1, 2
[ ] 2, 3, 1, 1
[ ] 2, 3, 2, 1

Prática

1) - Considere o procedimento get_hierarchical_level() que calcula o nível hierárquico de um dado directório na estrutura de directórios do sistema de gestão de ficheiros desenvolvido no Trabalho II.

```
int get_hierarchical_level(int dblock) {
  int level = 0;
  while (dblock != sb->root_block) {
    dir = (dir_entry *) BLOCK(dblock);

    // código em falta

    level++;
  }
  return level;
}
```

O nível hierárquico de um directório diz respeito ao número de directórios pai que existem no seu caminho absoluto. Por exemplo, o directório ``/dir/subdir'' é de nível 2, o directório ``/dir'' é de nível 1, e o directório raiz (``/'') é de nível 0. O procedimento recebe como argumento o número do primeiro data block (first_block) relativo ao directório pretendido e retorna o valor correspondente ao seu nível hierárquico. O código em falta no procedimento pode ser implementado pela sequência de instruções:

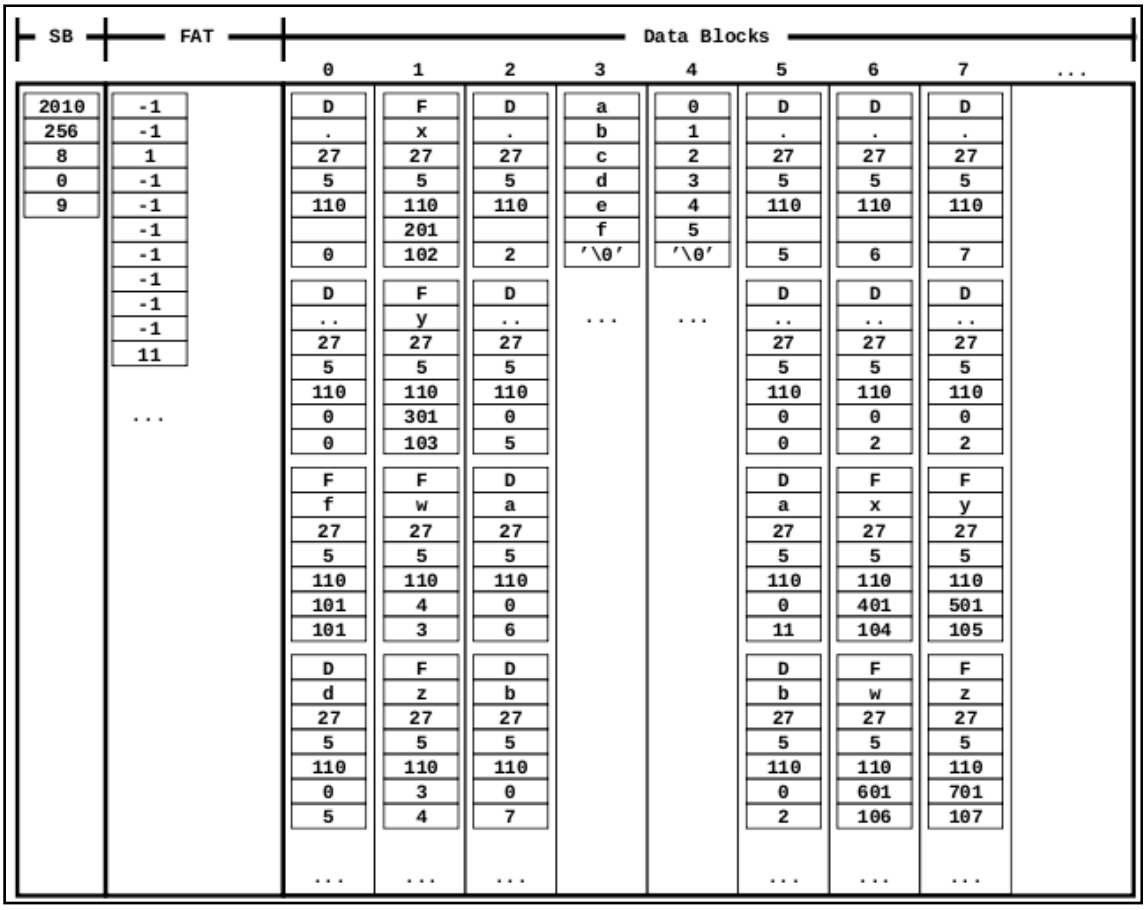
[ ] dblock = fat[dir[0].first_block];
[x] dblock = dir[1].first_block;
[ ] dblock = fat[dir[1].first_block];
[ ] dblock = dir[dir[0].size].first_block;

A imagem e código seguintes são válidos para todas as perguntas.

```
#define TYPE_DIR  'D'
#define TYPE_FILE 'F'
#define TYPE_FREE '-'
#define MAX_NAME_LENGHT 20

typedef struct superblock_entry {
  int check_number;
  int block_size;
  int fat_type;
  int root_block;
  int free_block;
} superblock;

typedef struct directory_entry {
  char type;
  char name[MAX_NAME_LENGHT];
  unsigned char day;
  unsigned char month;
  unsigned char year;
  int size;
  int first_block;
} dir_entry;
```



2) - Considerando que o bloco de dados 6 representa o directório corrente e que este possui 8 entradas (dir_entry), qual será o resultado da execução do comando `mkdir d' (assuma que `d' não existe no directório corrente):

[ ] não existem dados suficientes para responder à pergunta.

[x] **dá um erro do tipo 'mkdir: não existe espaço livre no disco'.**

[ ] é criada uma nova dir_entry no bloco de dados 6 e o novo directório `d' fica no bloco de dados 9.

[ ] é criada uma nova dir_entry no bloco de dados 9 e o novo directório `d' fica no bloco de dados 10.

3) - Considerando que '/d' é o directório corrente, quantos novos blocos de dados poderão ser ocupados pela execução do comando `mv a b':

[ ] 0 no mínimo e 0 no máximo.

[ ] 1 no mínimo e 1 no máximo.

[ ] não existem dados suficientes para responder à pergunta.

[x] 0 no mínimo e 1 no máximo.

4) - Considerando que '/d' é o directório corrente, quantos blocos de dados serão libertados pela execução do comando `rm a':

[ ] 1

[ ] 11

[x] 0

[ ] não existem dados suficientes para responder à pergunta.

5) - Quantos blocos de dados estão em utilização:
[ ] 9
[ ] 247
[ ] 8
[x] 255

6) - Considerando que o bloco de dados 2 representa o directório corrente, qual será o resultado da execução do comando `cat w':
[ ] não existem dados suficientes para responder à pergunta.
[x] abcd
[ ] abcdef
[ ] dá um erro do tipo 'cat: o ficheiro não existe'.

7) - Considerando que o bloco de dados 2 representa o directório corrente, qual será o resultado da execução do comando `cp y a' (assuma que `y' não existe em `a'):
[ ] dá um erro do tipo 'cp: o ficheiro de origem não existe'.
[ ] o ficheiro `y' é copiado para o directório `a'.
[x] dá um erro do tipo 'cp: não existe espaço livre no disco'.
[ ] o ficheiro `y' é copiado para o ficheiro `a'.

8) - Qual é o número total de blocos do sistema de ficheiros (considere inteiros de 4 bytes):
[ ] 1 + 8 + (8 * 256)
[ ] 1 + (8 * 4) + (8 * 256)
[ ] 1 + (2^8 / 256) + 2^8
[x] 1 + (2^8 * 4 / 256) + 2^8

9) - Quantas entradas (dir_entry) cabem num bloco de dados (considere inteiros de 4 bytes):
[ ] 16
[ ] 32
[x] 8
[ ] 4

10) - Considerando que o bloco de dados 6 representa o directório corrente, qual será o resultado da execução do comando `pwd':
[x] /d/b/a
[ ] /d/a/a
[ ] /a/b/d
[ ] /d/a/b