

- Por norma não se esclarecem dúvidas. Se tiver dúvidas, indique na folha de teste a sua interpretação.
- Escala 0:20 Valores. Sem Consulta .Utilize uma caligrafia legível.

Grupo A: História, Estrutura e Arquitetura dum Sistema Operativo (2 Valores)

1. Explique os termos "multiprogramação" e "multi-processamento".
2. O que é uma instrução privilegiada e como é executada. Dê alguns exemplos de Instruções Privilegiadas.

Grupo B: Programas, Processos e Escalonamento (4 Valores)

3. Qual é o output do seguinte programa? Justifique! (Deverá mostrar o trace do funcionamento do programa).

```
int x = 1;
if ( 0 == (pid = fork()) ) {
    pid = fork();
    x--;
    if ( 0 == pid )
        execl( "/bin/date", "date", NULL );
} else{
    fork();
    x=x+5;
}
printf("x=%d\n",x);
```

4. Usando o modelo de execução (explicado nas aulas teóricas), desenhe o layout (exposição) da memória quando a execução do programa indicado em baixo atinge o ponto indicado pelos asteriscos, logo a seguir o *printf* mas antes do retorno da função. O estado inicial de memória com o texto das funções é indicado na tabela em baixo.

Nota - Quando o valor numa cell de memória é um endereço escreva no formato &XY

Nota - Inicialmente utilize os seguintes endereços: Heap Pointer &40 e Stack Pointer &50

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	funções
0																	main
1																	func1
2																	func2
3																	
4																	
5																	
6																	

```
int main(){
    funk1('U');
}
void funk2( char *ptr )
{
    *(ptr+3)='N'+1;
    ptr = realloc(ptr,7);
    strcpy( ptr+4,"24");
    printf("%s",ptr);
    *****
}
```

```
void funk1( char MyP )
{
    char *ptrLocal;
    ptrLocal=malloc(4);
    *ptrLocal='E';
    *(ptrLocal+1)=MyP;
    *(ptrLocal+2)='Q'+1;
    funk2(ptrLocal);
}
```

Inserir aqui o

-output do PRINTF

-valor Heap Ptr

-valor Stack Ptr

Grupo C: Gestão de Memória (4 Valores)

5. Page Replacement

Num sistema de memória virtual paginada, quantas faltas de página aconteceriam usando os algoritmos de substituição de página "LRU" (least recently used) com a seguinte string de referência: 2, 3, 4, 2, 1, 5, 4, 3, 5, 2, 1 nos seguintes casos (i) três molduras (ii) quatro molduras. Este exercício exhibe a anomalia de Belady? Explique a sua resposta. (Todas as molduras estão inicialmente vazias)

6. Segurança e Eficiência

(i) Data Execution Prevention (DEP) é uma funcionalidade de segurança incluída nos sistemas operativos Microsoft Windows. O seu objetivo é o de impedir a execução de instruções vindas de certas zonas do espaço de endereçamento dum processo. Explique como é que este mecanismo poderia ser implementado usando um sistema de memória paginada e quais são as zonas de memória que devem ser protegidas.

(ii) Explique o conceito de COW (Copy-on-Write) usando no Linux na criação dum novo processo usando `fork()`. Para que serve e como é que poderá ser implementado usando paginação?

Grupo D: Concorrência, Sincronização e Bloqueio (6 valores)

7. Considere o programa seguinte:

```
int x=2;
void * work( void * ) { x=x+3; }
main(){
    int i; thread_t t[3];
    for (i=0;i<3;i++) pthread_create( &t[i], NULL, work, NULL);
    for (i=0;i<3;i++) pthread_join( t[i], NULL );
    printf("x=%d\n",x);
}
```

- Qual o output, ou outputs possíveis, do programa? Justifique!
 - Explique em **pormenor** como o valor final de `x` poderá ser igual ao 5. Deverá fazer uso duma linguagem assembler (NASM ou outro) ou um pseudocódigo apropriado.
 - Utilizando um semáforo ou variável de exclusão mútua (lock) altere o programa e a função `work` para executar a instrução `x=x+3` com exclusão mútua entre as threads. (deverá utilizar a sintaxe POSIX)
8. Considere a seguinte situação com 3 processos (P1, P2 e P3) e 3 recursos (R1, R2 e R3) - os recursos R1 e R3 tem apenas uma instância, mas o R2 tem duas instâncias do mesmo recurso.
- O recurso R1 está atribuído ao processo P2.
 - Ao processo P1 está atribuído uma das instâncias do recurso R2 e processo P1 está a espera do R1.
 - O processo P2 está a espera da atribuição duma instância do R2
 - O P3 está à espera do recurso R3.
- Desenhe o grafo de atribuição/alocações de recursos para esta situação.
 - Existe uma situação de bloqueio mútua (*Deadlock*) entre os processos ou não? Justifique.
 - Considere o seguinte: o sistema evolve da situação descrita em cima e o P3 pede e é atribuído uma instância do recurso R2 e o processo P2 pede o recurso R3. Desenhe o grafo atualizado de atribuição/alocações de recursos e explicar se agora há ou não bloqueio mútua (*Deadlock*) e/ou em que condição poderá ocorrer.

Grupo E: Exercício Prático (4 valores). Fazer APENAS um dos seguintes exercícios

9. Neste exercício irá escrever uma versão da função, `diff()`, que analise as diferenças entre dois ficheiros binários usando I/O de baixo-nível. Para cada "bloco" de bytes o programa deverá dizer o número de bytes que estão diferentes. Exemplo para dois ficheiros de 2100 bytes com block size de 1024

Block 1 : size 1024 : 0

Block 1 : size 1024 : 2

Block 2 : size 52 : 1

Pode supor que os dois ficheiros têm o mesmo tamanho. O programa deve abrir os ficheiros em modo leitura e o tamanho de bloco para a leitura dos ficheiros deverá ser definido através do atributo "st_blksize" obtido através a função "stat" ou "fstat" do ficheiro fileA

```
int diff ( char *fileA, char *fileB );
```

Caso qualquer dos ficheiros não puder ser aberto ou criado ou em caso de qualquer outro erro a função deverá terminar e devolver o valor -1. No caso de ter sucesso a função deverá devolver o valor 1.

10. Utilizando pipetas de baixo nível (pipes) é possível criar uma comunicação bidirecional entre dois processos!

Neste exercício escreva um programa tipo "cliente-servidor" para ajudar os milhões de pessoas que querem apostar no Euro2024. Todos querem adivinhar os resultados dos jogos, portanto vamos perguntar ao Mourinho! No início de cada etapa o Mourinho envia um ficheiro com os seus palpites para os jogos – o ficheiro "palpites.txt". Este tem o seguinte formato: nome da equipa A nome da equipa B golos da equipa A e golos da equipa B

Exemplo duma linha no ficheiro de texto: `portugal germany 1 1`

- O seu programa deverá criar 2 processos. O processo "progénito" (*child*) será o cliente e pedirá ao utilizador do programa duas strings – os nomes das duas equipas (de máximo 30 caracteres) que serão enviadas para o servidor através duma pipeta de baixo nível (pipe). O processo "progenitor" (*parent*) será o servidor. Este depois de receber os dois nomes abrirá o ficheiro fornecido pelo Mourinho, encontrará a linha respetiva e enviará o resultado do jogo para o processo cliente que imprimirá o palpite do Mourinho no ecrã. Se o utilizador introduzir um par de nomes inválidos o servidor enviará o resultado "-1 -1" para o cliente.

Exemplos dumas Execuções do Programa

Ask Mourinho Euro 2024 – Introduza os nomes das equipas! *albania italia* ↵
Mourinho acha que o resultado será *albania 1 italia 2*

Ask Mourinho Euro 2024 – Introduza os nomes das equipas! *turquia portugal* ↵
Mourinho acha que o resultado será *turquia 1 portugal 1*

Ask Mourinho Euro 2024 - Introduza os nomes das 2 equipas! *grecia usa* ↵
Não se brinca com Mourinho!

Notas: Pode assumir que os caracteres são sempre minúsculos. As sintaxes das funções necessárias são dadas numa folha anexa ao teste.

Notas

Não é necessário especificar as bibliotecas padrão (`#include <fcntl.h>` etc.) no seu programa.

Sintaxe Suplementar

int pipe(int *fildes); The pipe() function creates an anonymous pipe -- which is an object allowing unidirectional data flow for interprocess communication, and allocates a pair of file descriptors