UNIVERSIDADE DA BEIRA INTERIOR
Departamento de Informática
SISTEMAS OPERATIVOS

Frequência, 4 de Junho de 2021, 14.15 Horas Escala 0:20 Sem Consulta Duração: 1h45m

- Não é permitida a consulta de livros ou de apontamentos.
- Por norma não se esclarecem dúvidas durante a prova.
- Se tiver dúvidas, indique na folha de teste a sua interpretação.
- Utilize uma caligrafia legível.

1. Explique a diferença entre "User Mode" e "Kernel Mode" (Modo de Utilizador e Modo de Kernel) e as diferenças entre as funçoes sbrk() printf(), malloc(), sqrt(), strcmp(), write() (ver folha anexo para os sintaxes) **(2)**

2. Qual é a diferença essencial entre multi-programação e multi-processamento? **(1)**

3. Qual a relação entre "Processos", "Threads" e "Jobs" no sistema operativo Linux? Explique. **(1)**

4. Qual é o output do seguinte programa ? O output é determinístico? Justifique! (Deverá mostrar o *trace* do funcionamento do programa). **(2)**

```
char * args[2]={"date",NULL};
int  status, pid, x = 9;
pid = fork();
if (0==pid ) {
        x++;
        pid = fork();
       if (0==pid) execvp( *args, args );
        else        wait(&status);
}
else {
    x--;
}
printf("x=%d\n",x);
```

5. Considere um sistema de tempo real com 2 processos periódicos com períodos de 5 e 8 unidades e tempos de execução de 2 e 5 unidades respectivamente. **(3)**
   (i) Para este sistema poderá existir um escalonamento competente e viável ou não ? Explique .
   (ii) Explique em que tempo o sistema operativo poderá detetar um falha no prazo dum dos processos quando o algoritmo de Escalonamento é "Rate Monotonic". (Faça o diagrama de Gantt até cerca de 20 Unidades de tempo)

6. Gestão de Memoria **(3)**
   (i) Num sistema de memória paginada, faz algum sentido colocar a tabela de páginas em memória cache? Explique.
   (ii) 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 memoria que devem ser protegidas.
   (iii) 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 ?

7. Responde as perguntas seguintes considerando o seguinte programa em baixo.
Nota que a instrução "*x = x + k*" onde "*x*" é uma variável inteira global e *k* um constante em *assembler* (x86 GNU) é a sequencia de três instruções: *movl x, %eax ; addl $k, %eax ; movl %eax, x . For sintaxe NASM ver folha em anexo)* **(4)**

```
int x=0;
void *maisx(void *args)   {
    int id = *(int*)args ;
    x=x+id;
}
main() {
   pthread_t  th[3];
   int i,ids[3]={1,3,5};
   for (i=0; i<3; i++) pthread_create( &th[i],NULL, maisx, &ids[i]);
   for (i=0; i<3; i++) pthread_join( th[i], NULL );
   printf("x=%d\n",x);
}
```

a) Quais são os outputs possíveis do programa ?

b) Explique detalhadamente (faça uso dum *program trace*, pseudo-código, fluxograma etc) como é que o output poderá ser "x=6"

a) Usando a sintaxe de Posix *Threads* explique como é que pode usar um trinco logico de ezclusão mutuá para garantir que o resultado deste programa seja "x=9"

8. Quais são as condições necessárias para haver *deadlock* (bloqueio mútuo) entre um conjunto de processos que necessitam de recursos dum sistema computacional. Deverá especificar e explicar quaisquer pressupostos sobre os processos e sobre os recursos. **(1)**

9. Considere um sistema que gere contas bancárias e que serve clientes, identificados através dum número inteiro único, que efetuam pedidos de transferência de dinheiro entre contas do mesmo sistema. Tais pedidos podem ocorrer concorrentemente, sendo cada pedido servido por uma ***thread*** diferente. Cada cliente tem associado a sua conta uma trinco de exclusão mutua. Considere agora a seguinte implementação da função *transfere* (que cada thread executa para servir cada pedido de transferência): **(3)**

```
sem_t trincos[MAX_NUMERO_CONTAS];

transfere(int de, int para, int quantia) {
   sem_wait( &trincos[de] )
   sem_wait( &trincos[para] )
     saldo[de]  -= quantia;
     saldo[para] += quantia;
   sem_post( &trincos[de] )
   sem_post( &trincos[para] )
}
```

- Discute e Explique como este sistema possa entrar em "Deadlock" entre 3 clientes diferentes. A sua discussão devia incluir um grafo de alocação de recursos e deve indicar os passos necessários para Deadlock acontecer.