Licenciatura em Engenharia Informática
Sistemas Operativos 1- 1ª frequência – 7 de Abril de 2016
Departamento de Informática - Universidade de Évora

Justifique as suas respostas aplresentando os cálculos, quando aplicável.

1. Descreva graficamente o modelo de 7 estados.

2. Indique a hipótese correta. Um processo transita do estado RUN para o estado BLOCKED porque...
A – Terminou o tempo que estava reservado para correr no CPU e por isso o processo é interrompido.
B – O Processo precisa de esperar na fila de WAIT.
C – O processo executou uma instrução de I/O e fica à espera de um evento.
D – Ocorreu um evento enquanto esperava por dados.

3. Indique a hipótese correta.
A – O uso de threads só é vantajoso com CPUs múltiplos.
B – Com apenas um CPU o uso de threads permite aumentar a velocidade de resposta usando hardware de modo paralelo.
C – O uso de threads não é aplicável com CPUs múltiplos.
D – O uso de threads com CPUs múltiplos, torna-se mais lento.

4. Assinale quais dos seguintes são dados partilhados entre *threads* dum mesmo processo: *Program counter; Registos temporários do CPU; Variáveis globais; Código; Process ID; Estado; Ficheiros Abertos.*

5. Considere a seguinte tabela com o instante de chegada de cada processo à fila *ready* e com a duração do tempo de serviço no CPU:

processos	*T chegada*	*T serviço*
1	0	100
2	10	50
3	20	30
4	30	20

Calcule o tempo médio para terminar um processo (*turnaround time*) para os algoritmos:
5.1 - RR – round robin, quantum Q=20.
5.2 - SRT shortest remaining time
Nota: admita (se necessário) que num instante em que se interrompe um processo (se o algoritmo de escalonamento o impuser), primeiro passa-se o processo do CPU (*RUN*) para a fila de *READY* e só depois se testa se há processos novos para entrar na fila de *ready (de NEW para READY).*

6. Considere a seguinte tabela com o período e o tempo de serviço de três tarefas de tempo-real.

processos	*T serviço*	*Período*
A	30	120
B	10	30
C	20	60

6.1 Defina o escalonamento dos processos num período 120 ms, com o algoritmo de escalonamento "RMS-rate monotonic schedulling"

6.2 Indique Justificando se é garantido que exista escalonamento RMS sem violação de deadlines.

7. Considere o semáforo x, com as funções usuais wait(x) e signal(x) inicializado a 1.
Considere o seguinte programa que é lançado por vários processos em paralelo

```
While(true) do {
  N=N+1
  wait(x)
  P=P+1
  função()
  P=P-1
  signal(x)
  N=N-1
}
```

Indique justifica do se:
7.1 se os valor de N é sempre previsível
7.2 se os valor de P é sempre previsível

Licenciatura em Engenharia Informática
Sistemas Operativos 1- 2ª frequência – 19 de Maio de 2016
Departamento de Informática - Universidade de Évora

Justifique cuidadosamente todas as suas respostas

1. Considere um sistema com as seguintes matrizes de alocação; matriz dos pedidos; vector dos recursos totais; e vector das disponibilidades:

Request Matrix (Pedidos)

	A	*B*	*C*	*D*
P1	0	1	0	0
P2	3	4	0	0
P3	0	0	0	3
P4	1	1	4	1

Aloc Matrix (alocação)

	A	*B*	*C*	*D*
P1	1	0	2	1
P2	0	3	3	3
P3	2	2	2	0
P4	0	2	2	1
+	3	7	9	5

Rec	*tot*		
3	8	9	7

Disp

Indique se existe deadlock

0,5 a) Indique os recursos disponíveis. = Recursos totais − Recursos alocados 1 0 1 0 3

1 b) Indique se existe deadlock.

1 c) Após a detecção de deadlocks que acções e que critérios podem ser aplicados, de modo a resolver a situação ?

2. Usando semáforos, e indicando a sua inicialização, implemente um solução para o seguinte problema: considere um elevador panoramico com uma lotação de 15 pessoas, onde existe uma máquina de bilhetes. com um máquina e com espaço para 20 pessoas. Cada pessoa entra no elevador, vai à máquina, e, se houver bilhetes na máquina paga e retira um; a máquina imprime os bilhetes sempre que é retirado uma. Implemente em pseudo-código os processos "**máquina**" e "**cliente**", cumprindo as restrições enunciadas. Considere os seguintes procedimentos que pode usar: **entrar_elevador(), sair_elevador(), tirar_e_pagar_bilhete(), imprimir_bilhete()** .

3. Considere um sistema de gestão de memória paginado com page table de 4 níveis; com TLB de 10 ns de tempo de acesso, com um *Hit Ratio de 99%*, qual o tempo de acesso da RAM que garante um tempo médio de acesso inferior a 100 ns?

1

4. Num sistema de gestão de memória virtual com paginação, admita que o número de frames reservadas para as páginas é de 4 por processo.

1 2 3 4 2 1 6 4 2 6 4 2 3 1 4 6 2 4 1 3

1 a)Aplique o algoritmo de substituição algoritmo LRU aos pedidos.

1 b)Aplique o algoritmo de substituição algoritmo ótimo aos pedidos.

5. Consider um sistema de ficheiros indexado com i-nodes, com: blocos de 1024 Bytes; dimensão de endereços (de i-nodes e blocos) de 2 bytes; cada entrada num diretório tem 14 bytes para o nome e 2 para o endereço.

1 a) proponha uma estrutura para o i-node de modo que cada diretório comporte pelo menos 2500 ficheiros ou subdiretórios.

1 b) qual a dimensão máxima de um ficheiro no sistema que propôs ?

6. Indique a hipótese **correta**. Um sistema de memória paginada...

A – pode ter fragmentação interna

0,5 **B** – tem processos com dimensão superior à memória física RAM.

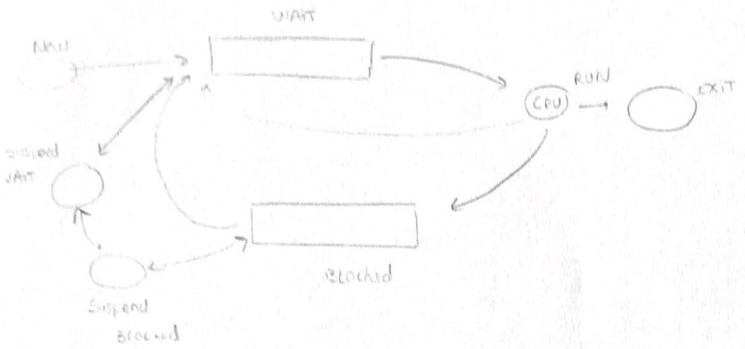
C – usa ou, o algoritmo BEST FIT ou o NEXT FIT

D – pode ter uma TLB (Table Lookaside Buffer) para diminuir a diensão das tabelas de paginação multinível

SO1

1ºfrequência

1).



RUN → EXIT : Quando um Processo termina

RUN → Blocked Quando o Processo precisa de uma interrupção

RUN → WAIT Devido ao escalonamento

2. A informação que existe num Processo são os Dados, o Código e o PCB. O PCB é composto pelos estados, o Program Counter, o I.D, os registos, I/O, os Ficheiros, o Prior e o Tempheap, este ultimos 2 que são dois tipos de escalonamento

3. As Threads de Kernell tem vantagem em relação às Threads de User level, porque as Threads de user level são coordenadas pelo sistema operativo, então as Threads de user level apenas podem correr em 1 CPU e não com as CPUs e são bloqueantes, ou seja quando precisam de interrupção suspostamente o processo vai ser obrigado a passar e a ir para o Blocked, o que faz com que a principal vantagem das Threads deixe de existir, em Kernell consigo aproveitar o Paralelismo. As Threads de user level tem vantagem em relação as Kernell na troca dos Processos, a troca de Processos em User level é mais rapida que em Kernel porque não possui a intervenção do sistema operativo enquanto que a alteração de Mudança de Processos em Kernell necessita da intervenção do Sistema Operativo.

4. Partilhados

Dinâmicos (variáveis globais)

Código

Dados

Não Partilhado

Ficheiros Abertos

o que está no PCB

ID

Estado

o PC

variáveis locais

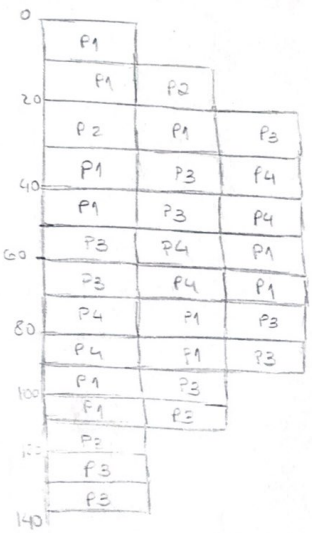
Registo de Ativação das funções

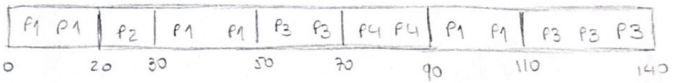
STACK
registos do CPU

5.

Processo	t chegada	t serviço
1	0	60
2	10	10
3	20	50
4	30	20

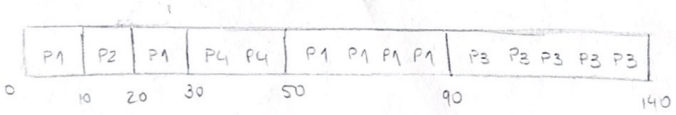
5. a) RR , TQuantum = 20





$$\frac{110 + 20 + 120 + 60}{4} = 77{,}5 \text{ m}\lambda$$

5. b) SRT



$$\frac{90 + 10 + 20 + 120}{4} = 60 \text{ m}\lambda$$

6.

Processo	T serviço	Período	
A	30	120	$3/12 = 1/4$
B	10	30	$1/3$
C	20	60	$1/3$
			1

Período de 8 mλ

$n \times (2^{1/n} - 1)$

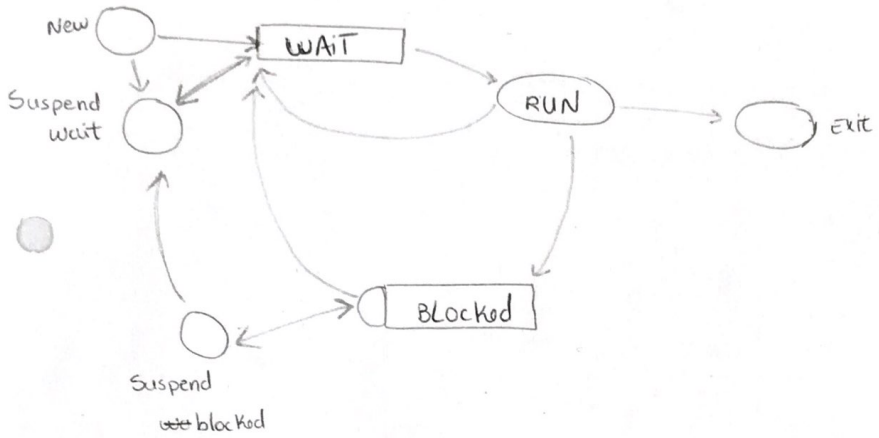
$3 \times (2^{1/3} - 1) = 0{,}78$

Para que o Algoritmo RTES funcione garantidamente o espaço que os processos ocupam no CPU tem de ser igual ou menor a 0,78

Como o espaço que os Processos ocupam do CPU é 1 não sabemos se o Algoritmo funciona ou não neste caso, visto que só sabemos que funciona no caso o valor ocupado for inferior ou igual a 0,78

então se a memória tiver sobrecarregado o processo pode ir para stand by dessa maneira o processo não irá ocupar memória por estar no estado WAIT e só quando a memória estiver numa sobrecarregado é que os processos vão para o estado WAIT. Quando os Processos acabam de ser executados, ou seja, de CORRER o processo pode desaparecer ou pode ir para o estado exit, onde irão ficar um bocado de tempo até poderem desaparecer.

Modelo de 7 estados



Quando um processo está no estado Suspend blocked por vezes a informação que está em memória do processo pode ser passado para o disco de modo a que a memoria fique com mais espaço, para que não ~~baisturamos~~ misturemos processos que precisam de algo com aqueles que estão a espera do "espaço" no CPU criamos 2 estados Suspend.

Poderiamos Construir um modelo de 9 estados com o mesmo esquema o que mudaria iria ser a Criação de mais estados (2) antes de os Processos irem para o wait, consoante a prioridade dos Processos, sendo que os Processos Prioritarios iriam primeiros que os restantes para o estado wait.

Transição de Estados

RUN → BLOCKED (Quando o processo precisa de uma informação qualquer)

RUN → EXIT (Quando o processo termina)

RUN → WAIT (Quando está a Correr á demasiado tempo e o escalonamento decide interromper o processo)

Kilo processo quando transita de um estado qualquer para o EXIT

mil quando passa do new para Suspend

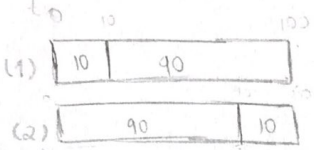
New → escalonamento de longo prazo

suspend → escalonamento de medio prazo

Block, wait, run → escalonamento de curto prazo (é o escalonamento mais utilizado)

Processos

se existirem 2 processos, 1 com 10 ms e outro com 90 ms, como fica mais eficiente a execução dos Processos



os processos independentemente do qual o 1º a ser executado vão demorar o mesmo tempo a correr os dois processos, mas qual será o mais eficiente.

(1)

$$\text{média} = \frac{110}{2} = 55$$

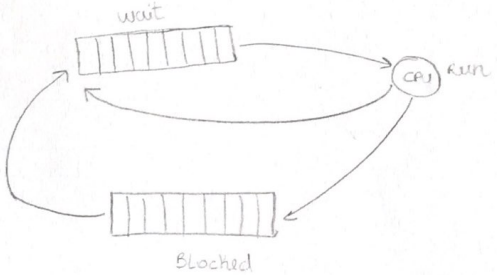
(2)

$$\text{média} = \frac{990}{2} = 95$$

Através da média de execução dos processos percebemos que era mais eficiente se os processos com um tempo mais curto fossem os primeiros a serem executados era mais eficiente.

Porem alguns Processos podem ser partidos aos bocados e dessa maneira já não iriamos pensar o mesmo que os mais curtos se fossem os primeiros a serem executados era mais eficaz. Logo os Processos podem estar divididos em bocados (Preemptivos) ou podem estar inteiros (não Preemptivos), ou seja, quando os Processos não são interrumpido.

Escalonamento de Processos (Cáp. 9 do livro)



O que o escalonador faz é a transição do wait para o Run. o escalonador escolhe qual dos Processos que estão em fila, no wait, vão para o CPU, caso a escolha seja pela ordem da fila temos um