

**Licenciatura em Engenharia Informática**  
**Sistemas Operativos 1- 2<sup>a</sup> 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

Disp

--

A B C D  
10 4 0 3

Indique se existe deadlock

- 0,5 1 a) Indique os recursos disponíveis. = recursos totais - recursos alocados  
 b) Indique se existe deadlock.  
 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 panorâmico com uma lotação de 15 pessoas, onde existe uma máquina de bilhetes, com um máquina e com espaço para 20 bilhetes. Cada pessoa entrará 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?

4. Num sistema de gestão de memória virtual com paginação, admita que o número de frames reservados 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 LRU aos pedidos.  
 1 b) Aplique o algoritmo de substituição 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

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 dimensão das tabelas de paginação multinível

B|B|A|A|A|A|B|A|A|A

3B

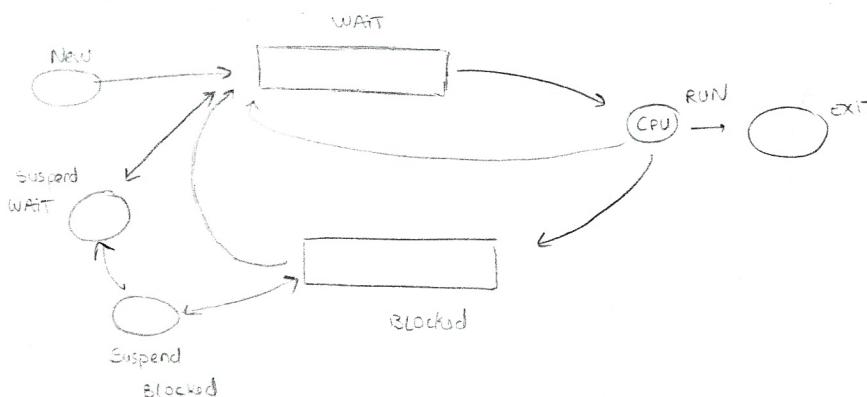
6A



504

## 1º frequência

2.



RUN → EXIT : Quando um Processo Termina  
RUN → Blocked : Quando o Processo precisa de uma informação  
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 ID, os registos, I/O, os Registros, o Rebanho e o Temporizador; estes últimos 2 que são dois tipos de escalonamento

3. As Threads do Kernel tem vantagem em relação às Threads do User Level, porque as Threads do user level são consideradas propriedade pelo sistema operativo, então as Threads do user level apenas podem correr em 1 CPU e não com n CPUs e são bloqueantes, ou seja, quando precisam de informações subsequentemente o processo terá de ser obrigado a parar e a ir para o Blocked, o que faz com que a principal vantagem das Threads deixe de existir em Kernel conseguindo aproveitar o Parallelismo. As Threads do user level tem vantagem em relação as kernel na troca dos processos, a troca de processos em user level é mais rápida que em kernel porque não possui a intervenção do sistema operativo enquanto que a ação da mudança de processos em kernel necessita da intervenção do sistema operativo.

## 4. Partilhado:

Dinâmicos (variáveis globais)

Código

Dados

## Não partilhado

Ficheiros Abertos

O que está no PCB

ID

Estado

O PC

Variáveis locais

Realiza a Ativação das funções

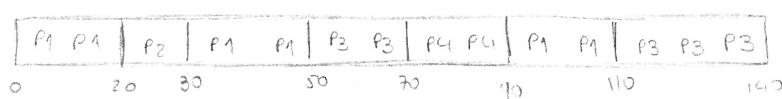
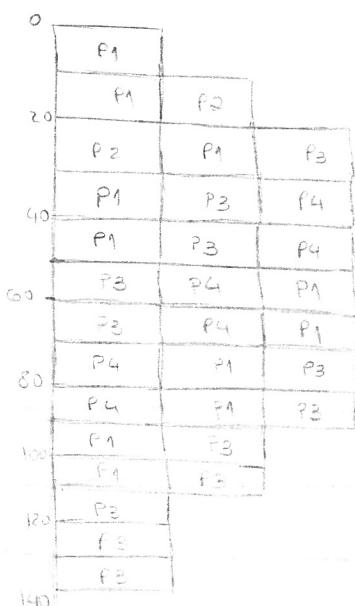
STACK

registos da CPU

5.

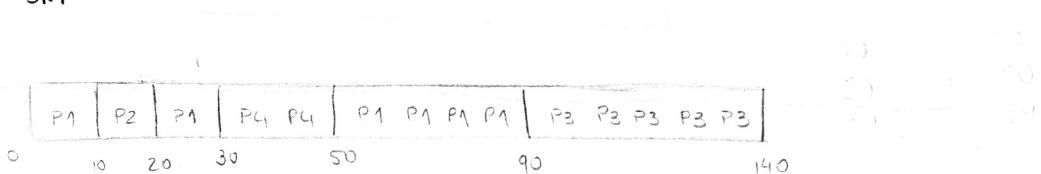
Processo	t chegada	t serviço
1	0	60
2	10	10
3	20	50
4	30	20

5.a) RR, T Quantum = 20



$$\frac{110 + 20 + 120 + 60}{4} = 72,5 \text{ ms}$$

5.b) SRT



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

Processo	T serviço	Período	Período de 8 ms	$n \times (2^{1/n} - 1)$
A	30	120	$\frac{3}{12} = \frac{1}{4}$	
B	10	30	$\frac{1}{3}$	$3 \times (2^{1/3} - 1) = 0,78$
C	20	60	$\frac{1}{3}$	

1

Para que o Algo

ritmo RRS funcione garantida

mente o espaço que os processos

ocupam na CPU tem de ser ig

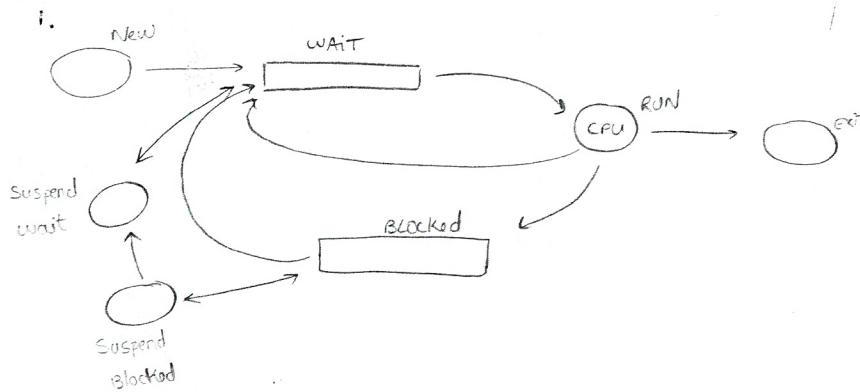
ual menor a 0,78.

Como o espaço que os processos ocupam na CPU é 1 não sabemos se o Algoritmo funciona ou não neste caso, visto que só sabermos que funcionava caso o valor acima fosse menor que 1.

• S01,

## 1.2 frequência

(2014)



RUN → EXIT ocorre quando um processo termina

RUN → blocked : Quando o processo precisa de uma informação, que pode ser do disco, I/O, etc..

RUN → WAIT : ocorre devido ao escalonamento, quando o processo está a correr à demasiado tempo ou porque chegou um processo mais prioritário

2. A informação que é guardada no PCB é as estatísticas program Counter, o ID, os registos, I/O, os

ficheiros e fechado e o Terphash, sendo que o Feichen e o Terphash são dois tipos de escalonamento. Um Processo possui também Código e Dados.

3. As vantagens das Threads em relação aos processos são a utilização de menos recursos, porque as threads são processos que partilham uma parte do código, por isso a mudança de contexto e a criação das threads é mais rápida que a criação e a mudança de contexto são mais lentas. As threads são processos que possuem uma parte do mesmo código e que foram pensados para trabalhar em paralelo em n CPU's, o que é uma vantagem em relação aos processos clássicos. Outra vantagem no uso de threads é quando é necessária uma informação se o processo precisa-se de uma informação este para de correr e tinha de ir para o blocked, mas threads isso não acontece, este não para nem vai para blocked mas continua a correr enquanto espera a informação.

4. Dados Partilhados:

Código

Dinâmicos (variáveis globais)

Dados

Dados não partilhados:

Ficheiros abertos

o que está no PCB

ID  
Estado

O PC

Variáveis locais

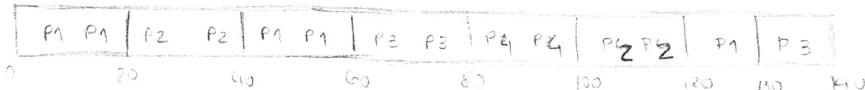
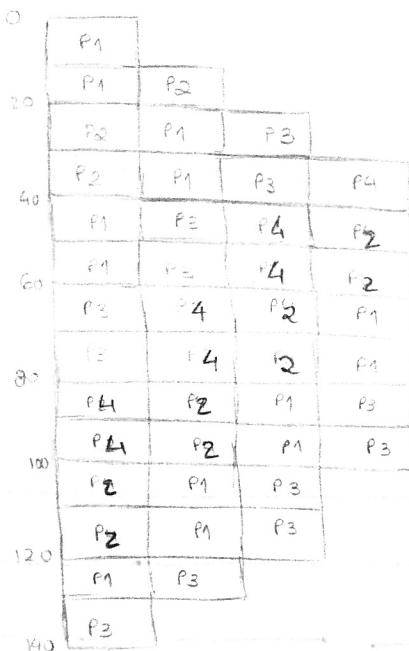
Registo de Ativações das funções

STACK

Registros do CPU

5.

Processo	Tchegada	t serviço	T <sub>Quantum</sub> = 20
1	0	50	
2	10	40	
3	20	30	
4	30	20	



$$\frac{130 + 110 + 120 + 70}{4} = \frac{430}{4} = 107,5 \text{ ms}$$

6. Período = 80 ms

Processos	T serviço	Período
A	20	80
B	10	20
C	10	40
		1

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

$$2/8 = 1/4$$

$$20/2 = 1/2$$

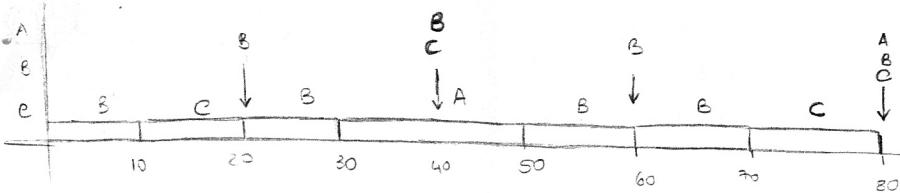
$$40/4 = 1/4$$

$3 \times (2^{1/3} - 1) = 0,78$ , ou seja, para o escalonamento ser de certeza possível só pode utilizar 0,78 da CPU.

Como a percentagem da CPU que os processos vão utilizar é 1 e como apenas vamos ter a garantia que o escalonamento RQS funciona se a utilização da CPU utilizada for 0,78 então neste caso o escalonamento pode ou não resultar no resultado que queremos. No entanto, se o escalonamento for menor que 0,78, não teremos garantia de que o escalonamento funcionará.

O processo que possui o período menor é o mais prioritário por isso vai ser o P1 a correr, neste caso é o PB.

Caso em que todos ao mesmo tempo.



Neste caso o escalonamento RQS funciona porque todos os processos terminam antes da sua deadline.

PCB

~~ficheiros Abertos~~

ficheiros

PC

I/O

JD

Registros

tipo de escalonamento

Estado S

Partilha

Código

Dados

Variáveis globais

N Partilha

ficheiros Abertos

PCB

registro da CPU

registro de Ativação de funções

STACK

Variáveis locais

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

**Justifique as suas respostas apresentando os cálculos, quando aplicável.**

- 1 ✓ 1. Descreva graficamente o modelo de 7 estados.
- 1 ✓ 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.
- 1 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.
- 1 ✓ 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.
- 1 ✓ 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        |
- white
- 2 6. Calcule o tempo médio para terminar um processo (*turnaround time*) para os algoritmos:  
 5.1 - RR – round robin, quantum Q=20. *fazer conta*  
 5.2 - SRT shortest remaining time *fazer conta*  
 Nota: admite (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*).
- 2 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      |
- 2 ✓ 6.1 Defina o escalonamento dos processos num período 120 ms, com o algoritmo de escalonamento "RMS-rate monotonic schedulling"
- 2 ✓ 6.2 Indique Justificando se é garantido que existe escalonamento RMS sem violação de deadlines.
- 2 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
- ```

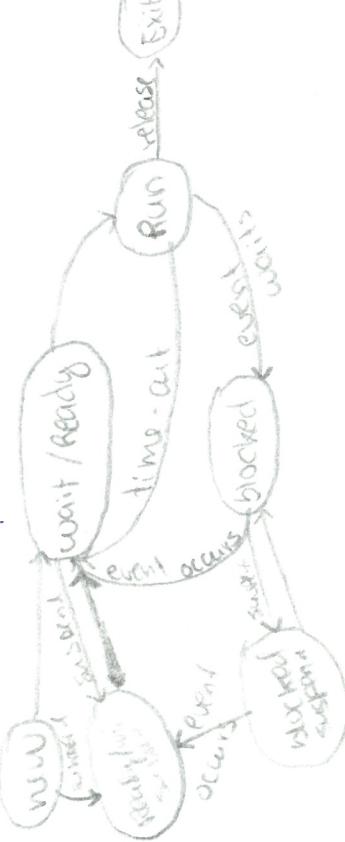
    While(true) do {
        N=N+1
        wait(x) → - no x
        P=P+1
        função()
        P=P-1
        signal(x) → + no x
        N=N-1
    }

```
- b n° de pessoas
- Indique justificando se:  
 7.1 se os valor de N é sempre previsível  
 7.2 se os valor de P é sempre previsível



18) desenhar gráfica do modelo + estados.

- desenhar os estados
- identificar as transições



(2) um processo transita do estado **run** para o estado **Blocoed**.

- C - O processo executa uma instrução 2/0 e pega à espera de um evento
- D - Dados partilhados entre threads em um mesmo processo

-> código

-> variáveis globais

a) RR  
Quantum = 20

| Processo | t serviço | t chegada |
|----------|-----------|-----------|
| P1       | 0         | 0         |
| P2       | 10        | 10        |
| P3       | 20        | 20        |
| P4       | 30        | 30        |

b) SRT

| Processo | tempo | tempo | tempo |
|----------|-------|-------|-------|
| P1       | 0     | 10    | 20    |
| P2       | 10    | 20    | 30    |
| P3       | 20    | 30    | 40    |
| P4       | 30    | 40    | 50    |

$$(90 - 0) + (20 - 10) + (160 - 90) + (50 - 30) = 92,5 \text{ ms}$$

| Processos | tempo | tempo | tempo         |
|-----------|-------|-------|---------------|
| P1        | 30    | 30    | 120 L.P 0,175 |
| P2        | 10    | 10    | 30 H.P 0,133  |
| P3        | 20    | 20    | 60            |

6

| Processos | tempo | tempo | tempo |
|-----------|-------|-------|-------|
| RMS       | 0     | 0     | 0     |
| P1        | 10    | 10    | 10    |
| P2        | 20    | 20    | 20    |
| P3        | 30    | 30    | 30    |

$$\tau_{A,T} = (110 - 0) + (30 - 10) + (160 - 20) + (90 - 30) = 271,5 \text{ ms}$$

| Processos | tempo | tempo | tempo |
|-----------|-------|-------|-------|
| P1        | 0     | 0     | 0     |
| P2        | 10    | 10    | 10    |
| P3        | 20    | 20    | 20    |
| P4        | 30    | 30    | 30    |
| P5        | 40    | 40    | 40    |
| P6        | 50    | 50    | 50    |
| P7        | 60    | 60    | 60    |
| P8        | 70    | 70    | 70    |
| P9        | 80    | 80    | 80    |
| P10       | 90    | 90    | 90    |
| P11       | 100   | 100   | 100   |
| P12       | 110   | 110   | 110   |
| P13       | 120   | 120   | 120   |
| P14       | 130   | 130   | 130   |
| P15       | 140   | 140   | 140   |

$$n(\tau_{A,T}) = 3 \times (2^{13} - 1) = 0,741$$

Se pode dizer que 74,1% das execuções resultam em 271,5 ms.

Portanto, os processos que têm o menor tempo de execução fazem 74,1% das execuções.

ANEXO

Matriz pedidos

|    | A | B | C | D | $\Sigma$ |
|----|---|---|---|---|----------|
| P1 | 0 | 1 | 0 | 0 | 1        |
| P2 | 3 | 4 | 0 | 0 | 9        |
| P3 | 0 | 0 | 3 | 3 | 6        |
| P4 | 1 | 1 | 1 | 1 | 4        |

Matriz Alocação

|    | A | B | C | D | $\Sigma$ |
|----|---|---|---|---|----------|
| P1 | 1 | 0 | 2 | 1 | X        |
| P2 | 0 | 3 | 3 | 3 | X        |
| P3 | 2 | 2 | 2 | 0 | X        |
| P4 | 0 | 2 | 2 | 1 | X        |

2016 - 2º Período

Processos = 15 init S = 1

init C = 15

maquina

cliente

wait(E)

wait(C)

ip((C=1) wait(S))

c = c + 1

signal(C)

signal(R)

c = c - 1

ip((C=1) then

signal(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then

wait(S)

wait(C)

c = c - 1

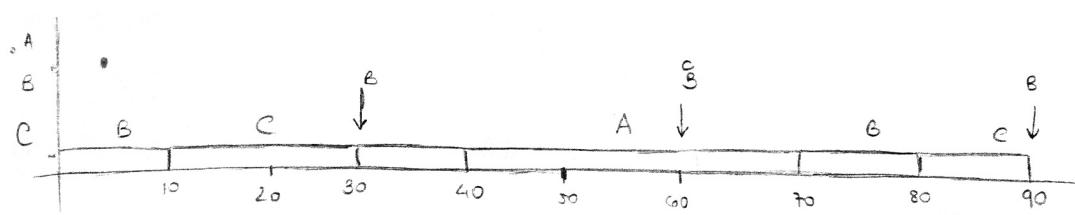
ip((C=1) then

wait(S)

wait(C)

c = c - 1

ip((C=1) then



O Processo que possui um período mais curto é o mais prioritário

Neste caso o Algoritmo de escalonamento FIFS funciona porque todos os Processos terminam antes da sua deadline

met

### Partilhados

código  
dados  
variáveis globais

### N Partilhados

ficheiros Abertos  
o que está no PCB  
registo da CPU  
registos de Ativação de funções  
STACK  
variáveis locais

### PCB

PC  
I/O  
ID  
Registros  
ficheiros  
Estado  
Tipo de Estacionamento

$$lotação = 100$$

$$Guarda = 0$$

$$Porta = 1$$

bit signal (Guarda)

wait (Guarda)

if n=0 then wait (Porta)  
wwait (lotação)  
n=n+1

entra ()

estar\_no\_jardim ()

$$n=n-1$$

saírl;

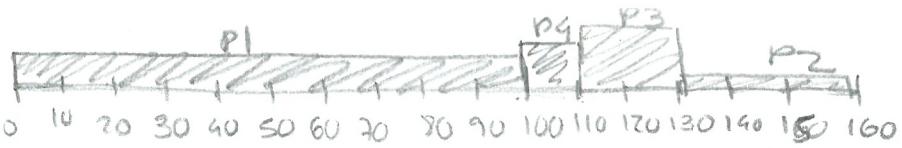
signal (lotação)

if n=0 then signal (Porta)

Signal (Guarda)

wait (Guarda)

SPN



$$t.a.t = (100 - 0) + (160 - 130) + (130 - 110) + (110 - 100)$$

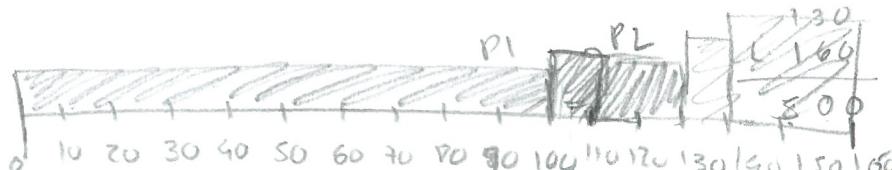
$$= \frac{100 + 30 + 20 + 10}{4} = \frac{160}{4}$$

E

$$\frac{100 + 110 + 130 + 160}{4} = \frac{100}{110}$$

HRRN

(não preemptivo) R=1



| P | Tchegada | Tservice |
|---|----------|----------|
| 1 | 0        | 100      |
| 2 | 10       | 30       |
| 3 | 30       | 20       |
| 4 | 40       | 10       |

$$100 + 30 + 20 + 10$$

$$\text{int } S = 1$$

wait(A)

if(i=1) then  
wait(B)

signal(A)

wait(B)

~sig(A)

|    | A | B | C | D |
|----|---|---|---|---|
| P1 | 1 | 1 | 0 | 1 |
| P2 | 1 | 0 | 1 | 1 |
| P3 | 1 | 1 | 0 | 2 |
| P4 | 1 | 0 | 1 | 1 |

|  | A | B | C | D |
|--|---|---|---|---|
|  | 3 | 3 | 0 | 1 |
|  | 0 | 0 | 1 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 0 | 0 | 3 | 1 |

RQ: Total  
| 5 | 5 | 5 | 5 |

| Fallas |   |   |   |
|--------|---|---|---|
| 1      | 1 | 0 | 0 |
| 2      | 0 | 3 | 3 |
| 3      | 3 | 0 | 1 |
| 4      | 0 | 1 | 1 |

Rec disponíveis

$$= 5355 - 4444 - [111]$$

74 → Red. de Recurso D(0,0,0,1)

(P1)

1111

+ 1001

-----

1112

P2

+ 6011

-----

1123

1111

+ 1011

-----

2122

+ 0031

-----

1153

1111

+ 0031

-----

1143

+ 0011

-----

1154

+ 0011

-----

1159

P3

P4

+ 1132

-----

2243

+ 0011

-----

2233

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11

11</p

1111

$$\begin{array}{r} +1101 \\ \hline \end{array} \quad \underline{\text{PS}}$$

2212

$$\begin{array}{r} +10031 \\ \hline 2243 \end{array} \quad \underline{\text{P4}}$$

$$\begin{array}{r} +10011 \\ \hline 2254 \end{array} \quad \underline{\text{P2}}$$