UFES - DEPARTAMENTO DE INFORMÁTICA

2ª. Prova de Sistemas Operacionais - Período: 2018/2 - Profª. Roberta Lima Gomes

Aluno: _____

1) **(2,0)** No primeiro trabalho, foi especificado que a fsh deveria criar uma "família de processos" a cada linha de comando

```
fsh> comando1 @ comando2 @ comando3
```

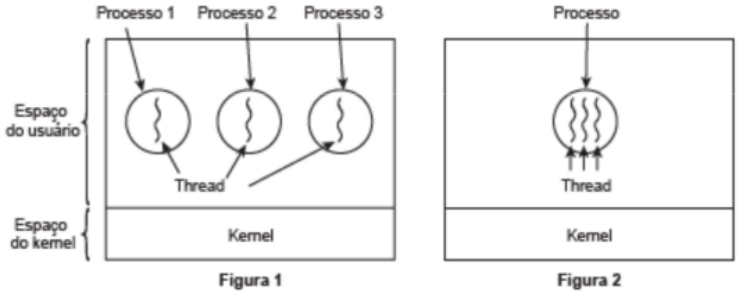
criando processos P1, P2 e P3 (nesse exemplo) para executar os comandos `comando1`, `comando2` e `comando3`, respectivamente, em background. E que que "Quando um dos processos Px morre devido a um sinal (no exemplo, P1 ou P2 ou P3) , todos os demais processos da família devem morrer juntos". Explique como seu grupo implementou esse comportamento.

2) **(3,0)** Especifique (em pseudocódigo) como um monitor seguindo a abordagem de Hoare pode ser implementado usando semáforos. Você deve definir (i) as variáveis necessárias, (ii) indicar como deve ser implementado cada procedimento de entrada do monitor (*entry procedure*), e (iii) indicar como deve ser implementada cada "variável de condição" (especificando igualmente suas respectivas operações *wait* e *signal*).

OBS: De acordo com Hoare, o processo sinalizador *P* bloqueia e espera pelo término da operação de monitor realizada pelo processo sinalizado *Q*.

3) **(2,5)** (ENADE)

Um processo tem um ou mais fluxos de execução, normalmente denominado apenas por *threads*.



A partir das figuras 1 e 2 apresentadas, avalie as afirmações a seguir.

I. Tanto na figura 1 quanto na figura 2, existem três *threads* que utilizam o mesmo espaço de endereçamento.
II. Tanto na figura 1 quanto na figura 2, existem três *threads* que utilizam três espaços de endereçamento distintos.
III. Na figura 2, existe um processo com um único espaço de endereçamento e três *threads* de controle.
IV. Na figura 1, existem três processos tradicionais, cada qual tem seu espaço de endereçamento e uma única *thread* de controle.
V. As *threads* permitem que várias execuções ocorram no mesmo ambiente de processo de forma independente uma das outras.

Para cada uma delas, indique se V ou F explicando sua resposta.

4) **(2,5)** (ENADE)

O problema do *buffer* limitado de tamanho *N* é um problema clássico de sincronização de processos: um grupo de processos utiliza um *buffer* de tamanho *N* para armazenar temporariamente itens produzidos; processos produtores produzem os itens, um a um, e os armazenam no *buffer*; processos consumidores retiram os itens do *buffer*, um a um, para processamento. O problema do *buffer* limitado de tamanho *N* pode ser resolvido com a utilização de semáforos, que são mecanismos de *software* para controle de concorrência entre processos. Duas operações são definidas para um semáforo *s*: `wait(s)` e `signal(s)`.

Considere o problema do *buffer* limitado de tamanho *N* cujos pseudocódigos dos processos produtor e consumidor estão mostrados na tabela abaixo. Pode-se resolver esse problema com a utilização dos semáforos ***mutex***, *cheio* e *vazio*, inicializados, respectivamente, com 1, 0 e *N*.

processo produtor	**processo consumidor**
produz item	`comando_e`
	`comando_f`
`comando_a`	
`comando_b`	**retira do *buffer***
coloca no *buffer*	`comando_g`
	`comando_h`
`comando_c`	
`comando_d`	**consome o item**

A partir dessas informações, para que o problema do *buffer* limitado de tamanho *N* cujos pseudocódigos foram apresentados possa ser resolvido a partir do uso dos semáforos *mutex*, *cheio* e *vazio*, é necessário que `comando_a`, `comando_b`, `comando_c`, `comando_d`, `comando_e`, `comando_f`, `comando_g` e `comando_h` correspondam, respectivamente, às operações

`comando_a` : _____ `comando_b` : _____

`comando_c` : _____ `comando_d` : _____

`comando_e` : _____ `comando_f` : _____

`comando_g` : _____ `comando_h` : _____

obs.: cada comando deve corresponder a uma chamada *down(semaforo)* OU *up(semaforo), em cima dos semáforos citados (mutex, cheio, vazio).*

Boa Prova !!!