ESCOLA POLITÉCNICA DE PERNAMBUCO

ALUNO: JOÃO VICTOR DOS SANTOS PEREIRA

DISCIPLINA: SISTEMAS OPERACIONAIS

AVALIAÇÃO 06

1º) Partindo dessa situação, pode-se dizer que não deve acontecer um impasse. Pois um impasse, são múltiplas tarefas bloqueadas que aguardam eventos que dependem delas. Ainda dentro do contexto, se outra tarefa for pedir o recurso que já está em utilização por outra tarefa e que está bloqueada, essa tarefa anterior não vai ficar bloqueada, pois pode ser devolvida da seguinte para a anterior.

2º) Se esse único processo não utilizar múltiplos threads, **não**. Também, são necessários quatro condições para que o deadlock aconteça**: Exclusão mútua, Posse e espera, Não-preempção e Espera circular.**

3º)

4º)

a) Um argumento para isso é que se pode garantir que o impasse nunca ocorresse. Mesmo tendo um aumento no tempo de resposta, os 5.000 trabalhos poderiam ser executados.
b) Um argumento contra isso, é que ocorrem com pouca frequência e custem pouco quando ocorrem.

5º) Se trata de um problema de impasse.

A mudança é um sinal depois de esperar no barrier. Conforme a thread passa, indica ao semáforo para que o próximo thread passe.

```
mutex.wait()
 count = count + 1;
mutex.signal()

if count == n: barrier.signal()
barrier.wait()
barrier.signal()

critical point
```

6º)

```
int n;                                          // n = num de cadeiras
int numClientes = 0;                            // num de clientes
Sem_t mutex = Semaphore(1);
Sem_t cliente = Semaphore(0);
Sem_t barbeiro = Semaphore(0);

cliente() {
sem_down(&mutex);                               // acessa a região crítica
 if (numClientes == n) {
  Volta();
  sem_up(&mutex);                               // sai da região critica
} else {
   numClientes++;                 // aumenta o número de clientes na espera
   sem_up(&cliente);               // chegou um cliente, avisa ao barbeiro
   sem_up(&mutex);                // sai da região critica
   sem_down(&barbeiro);            // espera o barbeiro cortar o cabelo
   ObterCorteCabelo();            // pede pra cortar o cabelo
}
}
barbeiro(){
while(1){
   sem_down(&cliente);             // se não tem cliente, dorme
   sem_down(&mutex);               // acessa a região critica
   numeroClientes--;              // diminui o número de clientes em espera
   sem_up(&mutex);                // barbeiro pronto
   sem_up(&barbeiro);             // sai do estado inseguro
   CortarCabelo();                // corta o cabelo
}
}
```