Laboratórios de Sistemas Operativos

Tutorial #5: Coordenação entre tarefas usando variáveis de condição

Este guião foca-se na implementação de situações de coordenação entre tarefas usando variáveis de condição.

1. Descarregue o *zip* e analise o programa *coordination.c.*

   Este programa implementa uma situação simples de coordenação entre tarefas. Neste caso, a tarefa inicial cria uma nova tarefa. Durante a execução do programa, há dois momentos de coordenação (em que uma tarefa espera até que outra tarefa execute alguma alteração ao estado partilhado):

   - A nova tarefa **espera** até que a variável partilhada (*value*) seja incrementada pela tarefa inicial.
   - A tarefa inicial, após ter modificado a variável partilhada, **espera** pela terminação da tarefa nova antes de imprimir o valor final no *stdout*.

   a) Compile usando a Makefile fornecida e experimente correr o programa usando o comando *time*:

   ```
   time ./coordination
   ```

   Analise os tempos apresentados no ecrã, em especial a componente *real* (tempo que passou até ao processo terminar) e a componente *user* (tempo de processador consumido pelo programa).

   b) Os tempos *user* e *real* são muito próximos. Isso significa que o processo esteve em execução (com uma ou outra tarefa) durante quase todo o tempo — apesar de grande parte desse tempo ter sigo gasto em espera! Ou seja, este programa **recorre a espera ativa**.
   Considerando as duas esperas referidas acima, qual/quais são esperas ativas e qual/quais são esperas bloqueadas?

2. Pretende-se corrigir o programa, substituindo a espera ativa que identificou na alínea anterior por uma espera bloqueada. Para tal, usaremos uma variável de condição.

   a) Abra o ficheiro *coordination_condvar.c*. Nele já encontra uma variável de condição declarada (*pthread_cond_t cond*). No entanto, este programa está incompleto pois a variável de condição não é usada.

   b) Inicialize a variável de condição no início da função *main*, usando a função *pthread_cond_init.*

   c) No local onde antes havia uma espera ativa, implemente uma espera bloqueada usando a função *pthread_cond_wait*. Não se esqueça que esta função deve ser sempre chamada de acordo com este padrão:

   ```
   while (! condicaoParaSairDaEspera)
   ```

```
pthread_cond_wait(variável de condição, trinco);
```

d) Não se esqueça de, inversamente, chamar a função *pthread_cond_signal* no(s) local(is) onde a condição de espera seja modificada.

e) Compile o novo programa e execute-o de novo usando o comando *time.*

```
time ./coordination_condvar
```

O que mudou nos tempos reportados pelo comando *time* (relativamente aos tempos reportados para a versão original)?

Como explica essa mudança?

Esta mudança é positiva? Porquê?

3. Agora aplique estes conhecimentos no seu projeto, para implementar as situações de coordenação entre tarefas que lá existam!