Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Sistemas Operativos, Verão de 2018/19

Segunda Série de Exercícios

I Parte

Biblioteca UThread

Para cada questão onde não for exigido explicitamente, apresente pelo menos um programa de teste que suporte a correção da solução proposta.

1. Modifique a biblioteca `UThread` para suportar as seguintes funcionalidades:

   a) Acrescente um campo ao descritor das *uthreads* para indicar o seu estado corrente. Os estados podem ser: `Running`, `Ready` e `Blocked`. Adicione à API a função `INT UtThreadState(HANDLE thread)` que retorna o estado da *thread* passada por parâmetro. Faça as alterações necessárias para manter o estado actualizado.

   b) Realize a função `BOOL UtAlive(HANDLE thread)` que retorna *true* se o *`handle`* passado como argumento corresponder ao de uma *thread* em actividade. Entende-se por *thread* em actividade qualquer *thread* que tenha sido criada e ainda não tenha terminado (não tenha invocado a função `UtExit`), independentemente do seu estado. Sugestão: mantenha uma lista de todas as *threads* em actividade.

   c) Acrescente suporte para medir o tempo que uma *thread* gasta nos estados `Running`, `Ready` e `Blocked`. Realize as funções em baixo que retornam, respectivamente, o tempo que a *thread* `h` gastou no estado `Ready`, no estado `Blocked` e no estado `Alive` até ao momento da chamada. Para medição de tempos utilize a função `GetTickCount()` da API Windows.

   ```
   ULONG UtGetTimeReady(HANDLE h);
   ULONG UtGetTimeBlocked(HANDLE h);
   ULONG UtGetTimeAlive(HANDLE h);
   ```

   d) Realize a função `VOID UtFinish()` que termina uma aplicação *UThread*. A implementação deve garantir que cada *thread* activa no momento da chamada deve terminar normalmente.

2. Escreva programas para determinar o tempo de comutação de *threads* no sistema operativo Windows. Teste o tempo de comutação entre *threads* do mesmo processo e entre *threads* de processos distintos. Para a medição de tempos, utilize a função da Windows API `GetTickCount`. Comente a relação do tempo de comutação obtido entre *threads* Windows e o tempo de comutação expectável entre *threads* da biblioteca `UThread`.

3. Implemente na forma de biblioteca o contentor associativo *ConcurrentHashMap* de pares chave/valor do tipo `ULONG`/`DOUBLE` que permite acessos concorrentes por múltiplas *threads* do mesmo processo e que define as operações descritas em baixo. Utilize os testes unitários para validar a sua solução.

```
// Cria e inicia um novo HashMap concorrente com capacidade capacity
ConcurrentHashMap * CHashMapCreate  (DWORD capacity);


// Elimina os pares associados à coleção map e o próprio map
VOID CHashMapDestroy  (ConcurrentHashMap * map);

// Retorna o valor em pval correspondente à chave key ou FALSE se não existir
BOOL CHashMapGet   (ConcurrentHashMap * map, ULONG key, DOUBLE * pval);

// Acrescenta ou substitui o par chave/valor. Retorna TRUE em caso de adição,
// FALSE em caso de substituição
BOOL CHashMapPut   (ConcurrentHashMap * map, ULONG key, DOUBLE val);

// Remove o par chave/valor retornando FALSE no caso da chave não existir
BOOL CHashMapRemove(ConcurrentHashMap * map, ULONG key);

// Retorna o número de pares chave/valor da coleção
ULONG CHashMapSize  (ConcurrentHashMap * map);

// Retorna uma lista formada por todos os pares chave/valor presentes na coleção map.
// Na implementação deverá tirar partido da multiplicidade de processadores do sistema.
LIST_ENTRY CHashMapToList(ConcurrentHashMap * map);
```

4. A DLL `MessageQueue` presente na solução em anexo contém a implementação de um mecanismo de comunicação entre *threads* do mesmo processo baseado em fila de mensagens. Complete a DLL com a implementação das funções com o prefixo `SysMq` que permitirão a utilização deste mecanismo entre *threads* de processos distintos. Os projetos `MqEchoServerTest` e `MqClientTest` contêm a implementação de um servidor de eco e do respetivo cliente.

João Patriarca e Jorge Martins
ISEL, 11 de Maio de 2019
Data de entrega: 31 de Maio de 2019