Recursive mutex

Implementing recursive mutexes.

(Note that Symbian 9 mutexes are recursive, so this applies to pre-v9)

If you come from a POSIX environment, you'll find Symbian's support for synchronization primitives rather simplistic. This is in some way due to Symbian's preference for single threaded applications using the Active Object idiom, thus making the use of multithreading often unnecessary. Sometimes, the need for recursive (reentrant) mutexes arises (note though that POSIX mutexes aren't recursive by default). Here is a possible implementation of a wrapper class over a RMutex:

```
class CRecursiveMutex : public CBase
{
public:
    CRecursiveMutex();
    ~CRecursiveMutex();

    void Acquire();
    void Release();

private:
    RMutex iMutex;
    TThreadId iOwner;
    TInt iCount;
};

CRecursiveMutex::CRecursiveMutex() : iCount(0)
{
    iMutex.CreateLocal();
}

CRecursiveMutex::~CRecursiveMutex()
{
    iMutex.Close();
}

void CRecursiveMutex::Acquire()
{
    TThreadId id = RThread().Id();
    if (iOwner == id)
    {
        ++iCount;
    }
    else
    {
        iMutex.Wait();
        iCount = 1;
        iOwner = id;
    }
}

void CRecursiveMutex::Release()
{
    if (--iCount == 0)
    {
        iOwner = 0;
```

```
        iMutex.Signal();
    }
}
```

You'll notice this is a bare bones implementation. Potential things to be added (at least on debug builds) could be checking thread ownership and iCount being 0 upon destruction.