condvar.cm

```
/*
  ==========================================================================

    Copyright (c) 2012-2015 Seppo Laakko
    http://sourceforge.net/projects/cmajor/

    Distributed under the GNU General Public License, version 3 (GPLv3).
    (See accompanying LICENSE.txt or http://www.gnu.org/licenses/gpl.html
       )

 ==========================================================================
    */

// Copyright (c) 1994
// Hewlett-Packard Company
// Copyright (c) 1996
// Silicon Graphics Computer Systems, Inc.
// Copyright (c) 2009 Alexander Stepanov and Paul McJones

using System;

namespace System.Threading
{
    public class ConditionVariable
    {
        public ConditionVariable(): handle(allocate_cond_handle())
        {
            int result = init_cond(handle);
            if (result != 0)
            {
                string reason = strerror(result);
                throw ThreadingException("could not initialize a
                    condition variable", reason);
            }
        }
        public ~ConditionVariable()
        {
            int result = destroy_cond(handle);
            free_cond_handle(handle);
        }
        suppress ConditionVariable(const ConditionVariable&);
        suppress void operator=(const ConditionVariable&);
        suppress ConditionVariable(ConditionVariable&&);
        suppress void operator=(ConditionVariable&&);
        public void NotifyOne()
        {
            int result = signal_cond(handle);
            if (result != 0)
            {
                string reason = strerror(result);
                throw ThreadingException("could not signal a condition
                    variable", reason);
            }
        }
        public void NotifyAll()
        {
            int result = broadcast_cond(handle);
            if (result != 0)
            {
                string reason = strerror(result);
                throw ThreadingException("could not broadcast a condition
                     variable", reason);
            }
        }
        public void Wait(Mutex& m)
        {
            mutex_t* mutexHandle = m.Handle();
            int result = wait_cond(handle, mutexHandle);
            if (result != 0)
            {
                string reason = strerror(result);
                throw ThreadingException("could not wait on a condition
                    variable", reason);
            }
        }
        public bool WaitUntil(Mutex& m, TimePoint tp)
        {
            bool timedOut = false;
            mutex_t* mutexHandle = m.Handle();
            long secs = tp.Rep() / 1000000000;
            int nanosecs = cast<int>(tp.Rep() % 1000000000);
            int result = timedwait_cond(handle, mutexHandle, secs,
                nanosecs, timedOut);
            if (result != 0)
            {
                string reason = strerror(result);
                throw ThreadingException("could not wait on a condition
                    variable", reason);
            }
            return timedOut;
        }
        public bool WaitFor(Mutex& m, Duration d)
        {
            return WaitUntil(m, Now() + d);
        }
        private cond_t* handle;
    }
}
```