
Subject: Interprocess communication with U++
Posted by [Mindtraveller](#) on Sun, 25 May 2008 18:27:36 GMT
[View Forum Message](#) <> [Reply to Message](#)

Are there any classes to help making interprocess communication?
Maybe something like Windows` named kernel objects.

For example, I`m updating a number of files and I want other copies of my process to wait for i/o operations to complete. Is there anything in U++ to help me with?

Subject: Re: Interprocess communication with U++
Posted by [Mindtraveller](#) on Mon, 26 May 2008 06:47:37 GMT
[View Forum Message](#) <> [Reply to Message](#)

I`ve discovered that both Windows and UNIX standards support named semaphores. So the existing Semaphore class could be extended towards the named semaphores.

Research continues.

Subject: Re: Interprocess communication with U++
Posted by [Mindtraveller](#) on Mon, 26 May 2008 07:16:53 GMT
[View Forum Message](#) <> [Reply to Message](#)

OK, patch seemes to be trivial.
Here is what I propose:

```
Mt.h: class Semaphore {  
#ifdef PLATFORM_WIN32  
    HANDLE    handle;  
#else  
    sem_t      sem;  
    sem_t      *namedSem;  
#endif
```

```
public:  
    void      Wait();  
    void      Release();
```

```
    Semaphore();  
    Semaphore(const char *name);  
    ~Semaphore();  
};
```

```
Win32: void Semaphore::Release()  
{
```

```

    ReleaseSemaphore(handle, 1, NULL);
}

void Semaphore::Wait()
{
    WaitForSingleObject(handle, INFINITE);
}

Semaphore::Semaphore()
{
    handle = CreateSemaphore(NULL, 0, INT_MAX, NULL);
}

Semaphore::Semaphore(const char *name)
{
    handle = CreateSemaphore(NULL, 0, INT_MAX, name);
}

Semaphore::~Semaphore()
{
    CloseHandle(handle);
}

POSIX: void Semaphore::Wait()
{
    namedSem ? sem_wait(namedSem) : sem_wait(&sem);
}

Semaphore::Semaphore()
: namedSem(NULL)
{
    sem_init(&sem, 0, 0);
}

Semaphore::Semaphore(const char *name)
{
    namedSem = sem_open(name, O_CREAT);
}

Semaphore::~Semaphore()
{
    namedSem ? sem_close(namedSem) : sem_destroy(&sem);
}

```

Didn't have opportunity to test on Linux.
