U++ - Feature #811
LocalProcess & AProcess should have "WaitForExit" synchronization method

07/19/2014 01:27 PM - Zbigniew Rebacz

Status:	Rejected	**Start date:**	07/19/2014
Priority:	Normal	**Due date:**	
Assignee:	Zbigniew Rebacz	**% Done:**	0%
Category:	Core	**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour

Description

I think it will be nice if LocalProcess & AProcess will posses one very usful method. It is "void WaitForExit()" method. The main goal of it is to wait until the process is finished.

Implementation:

```
void LocalProcess::WaitForExit()
{
  if(!IsRunning())
    return;

  bool isSuccess = false;
#ifdef PLATFORM_POSIX
  int status;
  if(waitpid(pid, &status, 0) == pid) {
    isSuccess = true;
    pid = 0;
  }
#endif
#ifdef PLATFORM_WIN32
  if(WaitForSingleObject(hProcess, INFINITE)) {
    isSuccess = true;
    hProcess = NULL;
  }
#endif
  if(isSuccess)
    CloseInputAndOutput();
}
```

Of course this patch also intoduce new private method "CloseInputAndOutput" which target is to close all pipes related to LocalProcess. (All implementation is included in source files).

Test case (I tested this solution with valgrind to make sure that we don't have any memory leaks):

```
#include <Core/Core.h>

using namespace Upp;

void startApp(LocalProcess& process) {
  #ifdef PLATFORM_POSIX
```

```
        process.Start("/usr/bin/xterm");
    #endif
    #ifdef PLATFORM_WIN32
        process.Start("E:\\Programy\\Notepad++\\notepad++.exe");
    #endif
}

CONSOLE_APP_MAIN {
    for (int i = 0; i < 1; i++) {
        // Test if "WaitForExit" works.
        LocalProcess myProcess;
        startApp(myProcess);
        myProcess.WaitForExit();

        // Test if we can reuse the same instance of LocalProcces for new task.
        startApp(myProcess);
        myProcess.WaitForExit();
    }
}
```

History

#1 - 07/19/2014 01:30 PM - Zbigniew Rebacz

We can also expand "WaitForExit" method by timeout variable.

#2 - 07/21/2014 07:36 PM - Miroslav Fidler

- Status changed from Patch ready to New

- Assignee changed from Miroslav Fidler to Zbigniew Rebacz

This has a problem: if slave process produces a lot of output, it will block forever (pipe buffer will get full as we are not reading it, process will be blocked).

#3 - 07/21/2014 08:56 PM - Zbigniew Rebacz

- Assignee changed from Zbigniew Rebacz to Miroslav Fidler

Should we close pipes befor we start waiting and throw exception if whole operation fails?

```
LocalProcess localProcess("/usr/bin/xterm");
try {
    localProcess.Wait();
}
catch(const EnviromentException& e) {
    // Pipes are not available...
    Cout() << e.what() << "\n";
```

```
        localProcess.Kill();
    }
```

Or, we can just do this inside WaitForExit method. But, in above case client programer can decide what to do in exceptional situation. What do you think about this solution?

P.S.
It will be good if we will possess a set of exceptions for different situations.

#4 - 07/22/2014 08:51 PM - Zbigniew Rebacz

- Status changed from New to Rejected

- Assignee changed from Miroslav Fidler to Zbigniew Rebacz

#5 - 07/22/2014 09:05 PM - Zbigniew Rebacz

It seems that it is bad idea, because to make it works perfect we will need more changes in Core.

Files

LocalProcess.cpp	12.6 KB	07/19/2014	Zbigniew Rebacz
LocalProcess.h	2.68 KB	07/19/2014	Zbigniew Rebacz
Process.cpp	561 Bytes	07/19/2014	Zbigniew Rebacz