Subject: Process Class wrapper for pipes in windows/linux
Posted by yeus on Fri, 24 Aug 2007 13:56:38 GMT
View Forum Message <> Reply to Message

does something like this already exist in ultimate++? If not, will it be added some time?

Some class, which wraps redirecting of streams between pipes stdout/int, sterr, creating your won streams to communicate between processes using pipes..

(I am mainly talking about a wrapper for the windows-api-function _popen(...), and the posix standart functions popen(...)

Greetings, Tom

Subject: Re: Process Class wrapper for pipes in windows/linux
Posted by mirek on Sat, 25 Aug 2007 08:16:59 GMT
View Forum Message <> Reply to Message

Search for "SlaveProcess".

Also, look into ide sources, Console.cpp or GDB interface.

Mirek

Subject: Re: Process Class wrapper for pipes in windows/linux
Posted by yeus on Sun, 26 Aug 2007 15:53:35 GMT
View Forum Message <> Reply to Message

thx... i have looked through the sources.. but i still have no clue how to use SlaveProcess... Somehow it gives me An error all the time, becuase Slave process is pure virtual, although iI am using "StartLocalProcess" and such stuff... I must admit: I have no clue, how SlaveProcess works ^^.

What would i need to do to achieve something like this with UPP:

FILE pipe=_popen("gnuplot.exe","w"); //start process

fprintf(pipe,"plot sin(x) \n"); //execute command in remote process
fflush(pipe);

_pclose(pipe);

greetings, Tom

Subject: Re: Process Class wrapper for pipes in windows/linux
Posted by mirek on Sun, 26 Aug 2007 17:52:36 GMT
View Forum Message <> Reply to Message

Should be something like:

```
One<SlaveProcess> p = StartLocalProcess("gnuplot w");
if(p) // started OK
    p.Write("plot sin(x)\n");
```

Mirek

Subject: Re: Process Class wrapper for pipes in windows/linux
Posted by yeus on Mon, 27 Aug 2007 11:14:34 GMT
View Forum Message <> Reply to Message

ahh i figured what my problem was: I #included the <web/sproc.h>-file and not the <web/web.h>. For some Reason the compiler had problems linking all that stuff...

Here is my code so far:

```
One<SlaveProcess> pipe;
pipe=StartLocalProcess("pgnuplot");

if(pipe){
   pipe->Write("plot sin(x)\n");

   cout<<"running...\n";
   String s;
   Sleep(10000);
   pipe->Kill();
   cout<<"gnuplot returned: "<<pipe->GetExitCode();
}//*/
```

I got the problem, that In this code "pipe" is NEVER invalid, as long as I have pgnuplot as starting command. Even if there appears a problem with starting pgnuplot for example, if I use invalid options and similar stuff.

Second Problem is, that for some reason I can not use the Kill()-Command. I am just not able to finish gnuplot when I started it...

Any idea on this?

Greetings, Tom...

Subject: Re: Process Class wrapper for pipes in windows/linux
Posted by mirek on Mon, 27 Aug 2007 13:29:53 GMT
View Forum Message <> Reply to Message

Maybe a PATH problem?

Mirek

Subject: Re: Process Class wrapper for pipes in windows/linux
Posted by yeus on Mon, 27 Aug 2007 14:14:29 GMT
View Forum Message <> Reply to Message

hmmm.... I made some further research and it seems that the Slaveprocess functions somehow interfere with the Sleep() function of U++...

All i can say is: as soon as I avoid using the Sleep function everything works just as expected...

Form the moment on, when I call the sleep function, the "pipe" object somehow gets disconnected from the process it owns, and I can not use the write, kill, read or whatever command then anymore....

greetings, Tom

Subject: Re: Process Class wrapper for pipes in windows/linux
Posted by yeus on Tue, 28 Aug 2007 17:00:57 GMT
View Forum Message <> Reply to Message

luzr is this a bug or something that you`d expect? Or Am I doing something wrong?

Greetings, Tom

Subject: Re: Process Class wrapper for pipes in windows/linux
Posted by mirek on Tue, 28 Aug 2007 17:07:58 GMT
View Forum Message <> Reply to Message

Well, is not it actually possible that process is finished while sleeping? (OTOH, in that case, it should still be kept as zombie).

All I can say is that SlaveProcess is extensively used in theide and (obviously) it seems to work.

Mirek