Subject: Re: capturing stdout/err/in of subprocess
Posted by mirek on Sat, 18 Mar 2006 07:47:38 GMT
View Forum Message <> Reply to Message

lundman wrote on Fri, 17 March 2006 19:01
I must agree I am not quite following why you need to know if your pipe-child is listening or not. You'll know if the child has gone away (EPIPE). The whole line buffering and echo is a tty/terminal feature which you can disable with ioctl(), that is quite trivial.

Just to remind: we are trying to create own terminal here.

Quote:
But if you create pipes to your child you need not worry about linemode/echo as it is direct without the tty/terminal layer.

Yes, that is why we need to implement echo/linemode to get the terminal/console equivalent

I am really not sure how to explain it, but the trouble is that terminal runs in "dialog" mode - child process prints something, then waits for input data, then again prints something, you simply need to know when it waits for input data in order to allow user enter them (I am describing user experience here, not implementation !)

You cannot allow user to enter the data all the time, at least that is not what terminals/consoles do...

Another attempt of explanation: You are implementing the terminal/console. You are executing the child process. User presses some key. How your terminal/console application should react to this event? If you would just echo it at the current output position, your characters would be "inlined" with child process output on the screen. And you need to echo that character somehow as it is terminal responsibility. (Of course, at some moment you will send characters down to the stdin pipe, but that is not what we are solving here).

The moral of the story probably is that you cannot implement the terminal just by redirecting stdin/stdout... (makes sense, curses or Win32 console functions seem to work on another level anyway).

Mirek