Subject: capturing stdout/err/in of subprocess
Posted by wilho on Mon, 13 Mar 2006 13:21:14 GMT
View Forum Message <> Reply to Message

Hi!

Just got the idea (meaning I haven't yet really done any reasearch) but; how complicated it would be create terminal emulator upp-control based on SDL?

Whats best platform-independent way to spawn subprocess and capture its output and feed its input? How to render ANSI with SDL reseived from shell/cmd?

I have none of SDL and only some c++ experience, so whats I'm really after is some understandable resources and maybe comments to make a conclusion if I'm able to begin with this.

Subject: Re: capturing stdout/err/in of subprocess
Posted by mirek on Mon, 13 Mar 2006 13:50:01 GMT
View Forum Message <> Reply to Message

wilho wrote on Mon, 13 March 2006 08:21Hi!

Just got the idea (meaning I haven't yet really done any reasearch) but; how complicated it would be create terminal emulator upp-control based on SDL?

Whats best platform-independent way to spawn subprocess and capture its output and feed its input? How to render ANSI with SDL reseived from shell/cmd?

I have none of SDL and only some c++ experience, so whats I'm really after is some understandable resources and maybe comments to make a conclusion if I'm able to begin with this.

Capturing output is relatively easy; feeding input is IMHO a trouble - the general problem is that current APIs do not give you a hint when user is allowed to feed that input (in other words, so far I have not found a way how to detect that slave process is waiting for the user input).

In fact, I would be glad if some such way would existed, that would allow me to improve "console mode" of executing programs in TheIDE....

Miriek

Subject: Re: capturing stdout/err/in of subprocess
Posted by gprentice on Tue, 14 Mar 2006 07:31:07 GMT
View Forum Message <> Reply to Message

Quote:
Capturing output is relatively easy; feeding input is IMHO a trouble - the general problem is that current APIs do not give you a hint when user is allowed to feed that input (in other words, so far I have not found a way how to detect that slave process is waiting for the user input).

In fact, I would be glad if some such way would existed, that would allow me to improve "console mode" of executing programs in TheIDE....

I'm lost. Don't you just "feed the input" when the user presses a key or sends a file - it's up to the user to know when the slave process will accept input ???

(except when the user explicitly enables XON/XOFF (software flow control) or RTS/CTS/DTR (hardware flow control))

Graeme

Subject: Re: capturing stdout/err/in of subprocess
Posted by wilho on Tue, 14 Mar 2006 07:53:55 GMT
View Forum Message <> Reply to Message

I suppose that miriek foresaw my intention not to create real terminal emulator but cmd.exe -wrapper.

If I redirect streams between myprocess/cmd-subprocess, is there a flow-control-alike mechanism - meaning do my input get lost if subprocess is not able to read it immediately?

Subject: Re: capturing stdout/err/in of subprocess
Posted by mirek on Tue, 14 Mar 2006 08:22:12 GMT
View Forum Message <> Reply to Message

Quote:
I'm lost. Don't you just "feed the input" when the user presses a key or sends a file - it's up to the user to know when the slave process will accept input ???

(except when the user explicitly enables XON/XOFF (software flow control) or RTS/CTS/DTR (hardware flow control))

Graeme

I believe that the trouble is with "echoing" user input on console: think, if keys are entered when the suprocess is running and printing the stuff, you have to echo keys entered by user, but simply placing them to the console would mix them with characters captured from subprocess stdout.

Even worse, user is supposed to edit the line...

But maybe my worries are void - perhaps console should always print edited input at the end until Enter is pressed and then send it to subprocess...

Mirek

Subject: Re: capturing stdout/err/in of subprocess
Posted by gprentice on Tue, 14 Mar 2006 09:43:49 GMT
View Forum Message <> Reply to Message

wilho wrote on Tue, 14 March 2006 20:53I suppose that miriek foresaw my intention not to create real terminal emulator but cmd.exe -wrapper.

If I redirect streams between myprocess/cmd-subprocess, is there a flow-control-alike mechanism - meaning do my input get lost if subprocess is not able to read it immediately?

As far as I can tell, the mechanism by which characters appear in the input stream (e.g. cin or stdin) is compiler, application and OS dependent but there's usually at least a small buffer either in the OS or the application or both (see setbuf()). I do not know whether an application asks the OS for a single character at a time or a "line" at a time or whether it's an event loop style "push" operation as in Win32 GUI apps, however there's no flow control provided by C/C++ runtime or the operating system, so an application that needs to have character input suspended or slowed at any time has to provide it's own flow control.

Graeme

Subject: Re: capturing stdout/err/in of subprocess
Posted by mirek on Tue, 14 Mar 2006 09:54:06 GMT
View Forum Message <> Reply to Message

I guess you still do not understand my worries

The problem is not with feeding characters down the subprocess stdin - that is simple. And subprocess will wait if stdin is empty.

The problem is with user interface implementation - it is terminal responsibility to echo user input. Try to start a long process in cmd.exe (or linux terminal) and try to hit some character keys while process is running - the characters will not appear until subprocess blocks waiting for input (e.g. getchar). (Of course, it can appear after it ends too).

The problem is that there is no method known to me how to detect this "child process is blocked as it is waiting for user input - time to display user input line".

Mirek

Subject: Re: capturing stdout/err/in of subprocess
Posted by gprentice on Tue, 14 Mar 2006 10:16:35 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 14 March 2006 22:54I guess you still do not understand my worries

The problem is not with feeding characters down the subprocess stdin - that is simple. And subprocess will wait if stdin is empty.

The problem is with user interface implementation - it is terminal responsibility to echo user input. Try to start a long process in cmd.exe (or linux terminal) and try to hit some character keys while process is running - the characters will not appear until subprocess blocks waiting for input (e.g. getchar). (Of course, it can appear after it ends too).

The problem is that there is no method known to me how to detect this "child process is blocked as it is waiting for user input - time to display user input line".

Mirek

You're right, I don't understand (though I was trying to respond to Wilho). But however, I'm curious. I've only used Linux terminal a little but like many people, I use cmd.exe a lot and I'm struggling to see where cmd.exe performs the echoing of user input you refer to e.g. if I do CL /?, the "CL" application is fed keys and does its own echoing (if any) - cmd.exe doesn't do the echoing. I would have thought any echoing is done by the application itself, via stdout. Any way, not to worry. I don't know how to build an atomic bomb either ...

Graeme

Subject: Re: capturing stdout/err/in of subprocess
Posted by lundman on Tue, 14 Mar 2006 11:00:48 GMT
View Forum Message <> Reply to Message

No idea about cmd.exe, but for Unix, if you type on the commandline and you are talking about if things echo or not is all down to tty mode, if its cooked or raw, in line mode, or char by char.

Nearest cmd equivalents seem to be SetConsoleMode(), or WaitForInputIdle().

For those spawning processes and sending/receiving IO to them which to do so portably, I can

recommend my own LiON library.

http://www.lundman.net/wiki/index.php/LiON

which we are using in our U++ we have just started.

Lund

Subject: Re: capturing stdout/err/in of subprocess
Posted by mirek on Tue, 14 Mar 2006 11:04:31 GMT
View Forum Message <> Reply to Message

Quote:
You're right, I don't understand (though I was trying to respond to Wilho). But however, I'm curious. I've only used Linux terminal a little but like many people, I use cmd.exe a lot and I'm struggling to see where cmd.exe performs the echoing of user input you refer to e.g. if I do CL /?, the "CL" application is fed keys and does its own echoing (if any) - cmd.exe doesn't do the echoing. I would have thought any echoing is done by the application itself, via stdout.

void main() { getchar(); }

Start that. The program will stop, allowing you to enter the line of input characters. Now this "enter the line of characters" is that "echoing" I am speaking about. And, AFAIK, this is not done by program, but terminal (in any case, it is terminal responsibility that if you press backspace, last character is removed from the input line).

Quote:
Any way, not to worry. I don't know how to build an atomic bomb either ...

Explaining to you always pays off... Usually it leads to solving the problem. I am not sure why, but it is almost a rule

Mirek

Subject: Re: capturing stdout/err/in of subprocess
Posted by gprentice on Tue, 14 Mar 2006 23:09:41 GMT
View Forum Message <> Reply to Message

Quote:void main() { getchar(); }

Start that. The program will stop, allowing you to enter the line of input characters. Now this "enter the line of characters" is that "echoing" I am speaking about. And, AFAIK, this is not done by program, but terminal (in any case, it is terminal responsibility that if you press backspace, last character is removed from the input line).

Wow, I never noticed that. (I don't write console programs much.)

I tried the following program (with Borland compiler on Win XP). If I run it, the first key that gets pressed doesn't get echoed but its numeric value does get printed, then I can press a bunch of keys that do get echoed (and backspace works just as you say) and when I finally press return, the program exits.

getch() is Borland extension (and kbhit) - I think VC has similar ones. I wonder why the first keypress doesn't get echoed. If I change getch() to getchar(), the first character and following characters (until I press return) do get echoed, after which the numeric value gets printed. Maybe getchar() hooks up to a blocking OS function that runs in "line buffer" mode and getch() doesn't run in line mode.

Graeme

```
#include <stdio.h>
#include <conio.h>

int main()
{
   while (1)
   {
      if (kbhit())
      {
         printf(" yes");
         int k = getch();
         printf( "%d", k);
         break;
      }
   }
   getchar();
}
```

Subject: Re: capturing stdout/err/in of subprocess
Posted by mirek on Wed, 15 Mar 2006 08:59:17 GMT
View Forum Message <> Reply to Message

I believe that getch uses different API - console api (Win32 curses equivalent).

The problem there is that getch is different from standard input.

Mirek

Subject: Re: capturing stdout/err/in of subprocess
Posted by mr_ped on Fri, 17 Mar 2006 09:52:57 GMT
View Forum Message <> Reply to Message

I don't see problem with knowing if the subprocess is looking for input.

If it does, it must call "getchar()" and similar things, i.e. listen to the standart input. If you (as an console) own the standart input line, you will know when the subprocess listens to it, so you know when to echo the input.

I.e.
OS input -> your console listen to it, and fetch the input into internal buffer, which will be standart input for subprocess -> subprocess connected to input from your console.

I don't know how this should be done in C++ and Windows, but that's imho the proper model how it should be done.

The cmd.exe allows you to choose the input for the subprocess, so you can run the subprocess and feed him with input from file or from output from other process trough pipe, and the reason why it works is because it knows when the subprocess calls getchar(), so the whole file is not feeded to the subprocess right at the start, but it's sended char by char whenever subprocess asks for next one.
(actually having ability to redirect input/output is very important in unix world, as most of the standart tools are highly specialized, and to fullfill your task you need usually to chain several tools to produce the final result, like "ps | grep gcc" to see only gcc processes. So the "ps" tool does not need to contain some "filter" code, as the filter code is included in grep, etc...

Subject: Re: capturing stdout/err/in of subprocess
Posted by mirek on Fri, 17 Mar 2006 11:08:13 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Fri, 17 March 2006 04:52
If you (as an console) own the standart input line, you will know when the subprocess listens to it

Well, this is the problem - HOW you will know?

There are no pipe functions I am aware of (in both Linux and Win32) that would indicate the "waiting for the stdin" (or, in fact "subprocess is blocked because of empty stdin") state.

Standard input behaves (from master process point of view) just like output file.

Then again, I might have missed something.

Mirek

Subject: Re: capturing stdout/err/in of subprocess
Posted by fudadmin on Fri, 17 Mar 2006 11:11:47 GMT
View Forum Message <> Reply to Message

What if to use a socket?

Subject: Re: capturing stdout/err/in of subprocess
Posted by mirek on Fri, 17 Mar 2006 11:23:28 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Fri, 17 March 2006 06:11What if to use a socket?

AFAIK sockets do not have anything to do with pipes (and redirecting stdin).

Actually, I suspect that there are needed functions in Linux, just that I have missed them... For Win32, I am less sure.

Mirek

Subject: Re: capturing stdout/err/in of subprocess
Posted by gprentice on Fri, 17 Mar 2006 22:11:33 GMT
View Forum Message <> Reply to Message

Quote:
There are no pipe functions I am aware of (in both Linux and Win32) that would indicate the "waiting for the stdin" (or, in fact "subprocess is blocked because of empty stdin") state.

Standard input behaves (from master process point of view) just like output file.

Then again, I might have missed something.

Why do you need to know the process is waiting for input? If you send characters to stdin the OS is going to buffer them until the process asks for them. From what I can tell, if you were to try and hold off sending a character until the previous character had been processed, the application would run too slowly because of all the context switching. It seems to me it's the applications responsibility to keep up with the input if it knows there's a lot of it. Maybe you could try PeekConsoleInput if you have the stdin handle, to see if stdin is "empty" ?

Graeme

Subject: Re: capturing stdout/err/in of subprocess
Posted by mirek on Fri, 17 Mar 2006 22:25:32 GMT
View Forum Message <> Reply to Message

gprentice wrote on Fri, 17 March 2006 17:11
Quote:
There are no pipe functions I am aware of (in both Linux and Win32) that would indicate the "waiting for the stdin" (or, in fact "subprocess is blocked because of empty stdin") state.

Standard input behaves (from master process point of view) just like output file.

Then again, I might have missed something.

Why do you need to know the process is waiting for input? If you send characters to stdin the OS is going to buffer them until the process asks for them. From what I can tell, if you were to try and hold off sending a character until the previous character had been processed, the application would run too slowly because of all the context switching. It seems to me it's the applications responsibility to keep up with the input if it knows there's a lot of it. Maybe you could try PeekConsoleInput if you have the stdin handle, to see if stdin is "empty" ?

Graeme

Unfortunatly, PeekConsoleInput seems to deal with existing Win32 console - not something would help us as we are here trying to implement ours...

Mirek

Subject: Re: capturing stdout/err/in of subprocess
Posted by gprentice on Fri, 17 Mar 2006 22:44:57 GMT
View Forum Message <> Reply to Message

Quote:
Unfortunatly, PeekConsoleInput seems to deal with existing Win32 console - not something would

help us as we are here trying to implement ours...

PeekNamedPipe then ...

Subject: Re: capturing stdout/err/in of subprocess
Posted by lundman on Sat, 18 Mar 2006 00:01:31 GMT
View Forum Message <> Reply to Message

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.

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

You can check the number of bytes buffered in the pipe, set the low and high water marks, and maybe even the process flags. But why?

In LiON, to make sure to be portable, and allow for the pipe to be in select() non-blocking loops (HANDLES can't be in select(), and nonblocking polling is hard if you want to work on Win95/98/Me), I had to create read/write threads to translate between HANDLE and SOCKET.

But it works well, the same code will spawn processes on Unix, as Windows, and talk to it. I had to implement fork() and execv() for Windows though.

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

Subject: Re: capturing stdout/err/in of subprocess
Posted by wilho on Sat, 18 Mar 2006 13:33:45 GMT
View Forum Message <> Reply to Message

Lot's of discussion and information, thanks guys. While consoles don't regularlary allow wirting before they're able to receive doesn't IMO mean that it couldn't do so even though it might be a bit messy. If someone is interest to see the problems involved, here'sa nice sample which spawns cmd and redirects its I/O to itself via anonymous pipes.

Lundman, how do I compile LiON with TheIDE? I had to give the win32 define compiler by switch. It compiles, but I get lots of linking errors. As a matter of fact, if someones knows good article about common linker errors and how to resolve them, I'd be itnterested...

Subject: Re: capturing stdout/err/in of subprocess
Posted by lundman on Sat, 18 Mar 2006 14:05:31 GMT
View Forum Message <> Reply to Message

You know, I should make a project for TheIDE too.. I just made project files for vc6 and vc8. We

just link against the library (and wsock32.lib etc) in TheIDE. I'll work on it.
