Base Language - Bug #5538

fix readByte() for DirStream, ProcessStream, and others

07/10/2021 04:21 AM - Constantin Asofiei

Status:	Closed	**Start date:**	
Priority:	Low	**Due date:**	
Assignee:	Theodoros Theodorou	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

Related issues:

Related to Base Language - Feature #7842: Make FWD behave the same as OE rega...	**New**

History

#2 - 07/10/2021 04:23 AM - Constantin Asofiei

FileStream and Stream were reworked to read characters or bytes as needed.

Currently Stream.readByte() defaults to readCh(), and only FileStream.readByte() implements it properly. We need proper fixes for DirStream, ProcessStream and others.

See 3821c rev 12652.

#3 - 07/05/2023 02:53 PM - Greg Shah

- Assignee set to Theodoros Theodorou

- Start date deleted (07/10/2021)

#4 - 08/03/2023 08:30 PM - Theodoros Theodorou

- Status changed from New to WIP

#5 - 08/04/2023 05:48 PM - Theodoros Theodorou

- % Done changed from 0 to 100

- Status changed from WIP to Review

The implementations of the abstract Stream.readByte() are the following:

ClipboardStream:

```
public int readByte() { throw new UnsupportedOperationException(); }
```

DirStream:

```
public int readByte() { return InputStream.read(); }
```

FileStream:

```
public int readByte() { int b = readByteWorker(false); }
```

NullStream:

```
public int readByte() { return -2; }
```

PrinterStream:

```
public int readByte() { return -2; }
```

ProcessStream:

```
public int readByte() { int res = readChWorker(false); }
```

TerminalStream:

```
public int readByte() { return -2; }
```

As I understand, the only implementation needed to be fixed is ProcessStream.readByte(). I did the changes in 5538a. Please review and let me know if I missed anything.

#6 - 08/07/2023 07:01 PM - Ovidiu Maxiniuc

- Status changed from Review to Test

Review of 5538a/14681.

I agree with the code. Just two notes about code:

- please move comments from lines 462 and 816 on separate lines;
- add a comment in catch block from 838 that the expected return -2 is coalesced with the case when the child is no longer alive and data from both its stream is no longer available (that is when the while loop exits naturally).

#7 - 08/11/2023 10:56 AM - Greg Shah

Once these items are addressed, what testing is needed before this can be merged?

#8 - 08/11/2023 10:56 AM - Greg Shah

- Status changed from Test to Review

#9 - 08/14/2023 10:09 AM - Theodoros Theodorou

Ovidiu Maxiniuc wrote:

> Review of 5538a/14681.
>
> I agree with the code. Just two notes about code:
>
> - please move comments from lines 462 and 816 on separate lines;
> - add a comment in catch block from 838 that the expected return -2 is coalesced with the case when the child is no longer alive and data from both its stream is no longer available (that is when the while loop exits naturally).

Done, please review 5538a rev no 14682.

Greg Shah wrote:

> Once these items are addressed, what testing is needed before this can be merged?

I think a unit test would be great for testing ProcessStream.readByte(). I tried to create an object but without success. If a test is needed, please provide an example on how to create a new ProcessStream and how to use connectOut().

#10 - 08/14/2023 11:20 AM - Greg Shah

As with most of what we do, it makes most sense to implement 4GL code that can test this. Pure Java unit tests don't work well.

Please make sure you have written some 4GL code to test this code path and then convert/run it. We are working on using ABLUnit to automate unit test of 4GL testcases but the testcases are not complete yet. This child process support would be part of our I/O and streams tests in base language (see #6854). For now, just write the code as non-interactive and run it manually.

#11 - 08/14/2023 01:01 PM - Ovidiu Maxiniuc

I had some thought about the implementation of this issue over the weekend. The current implementation scans both out and err streams for available bytes and returns the byte from any of them, preferably the former. I think (but not sure, this must be checked with specific testcase) this is not correct.

More specifically, I intuition says that, once we started to read a stream, we should read all available bytes from it at that specific time, before deciding to switch to the other stream. Otherwise, the reader will get scrambled results. Here is an example:

event	status	result
startup	out: (empty), err: (empty)	
program outputs White on err	out: (empty), err: White	
readByte() invoked	out: (empty), err: hite	will return W
program outputs Black on out	out: Black, err: White	
readByte() invoked	out: lack, err: hite	will return B
..		
readByte() invoked	out: empty, err: hite	will return k
readByte() invoked	out: empty, err: ite	will return h
..		

My point is that, instead of WhileBlack the result would be WBlackhite. If I am right, once the data is available on err we should stick to reading the whole chuck of available data (5 bytes in this case) before re-scanning both streams. If both of them have bytes in buffer, the out stream will probably be preferred.

I repeat, this is just a supposition and must be validated with a testcase to have a confirmation before switching to new implementation.

#12 - 08/14/2023 04:13 PM - Theodoros Theodorou

Ovidiu Maxiniuc wrote:

> My point is that, instead of WhileBlack the result would be WBlackhite. If I am right, once the data is available on err we should stick to reading the whole chuck of available data (5 bytes in this case) before re-scanning both streams. If both of them have bytes in buffer, the out stream will probably be preferred.

I see your point and you are right, although I am afraid to change the logic as I do not have the full picture of the implementation and its use cases. I just copied the logic from readChWorker() and modified it a bit. Moreover, without preexisting unit tests, it will be difficult to know that our changes will not break anything. I think Greg has to take a look at it.

#13 - 08/14/2023 05:41 PM - Ovidiu Maxiniuc

In these situations we do our testcases and run them against 4GL to see the actual behaviour. Along the years we encountered a lot of undocumented issues only by running targetted testcases. These edge-case testcases remain documented here and eventually in testcase project(s).

#14 - 08/15/2023 05:16 PM - Theodoros Theodorou

It looks like the developer who implemented the ProcessStream was aware of the issue. I found in the javadoc (ProcessStream.java line 141) the following:

> The readCh will always read from STDOUT as long as there are bytes available. However this means that there is a natural race condition with STDERR. If at one moment STDOUT has no bytes available for reading, then a read will occur from STDERR. If bytes arrive on STDOUT then the very next call to readCh will first read from STDOUT even if there are bytes that arrived previously on STDERR. This may make the output appear in a different order than expected.
> A polling approach is used for all reading to implement the semantic of combining the two pipes.
> This class does not support getting/setting the length or read/write position of this stream.

Can you please provide a basic 4GL example that uses ProcessStream?

#15 - 08/16/2023 06:18 AM - Greg Shah

```
def var txt as char init "This is some bogus data.".

/* open unnamed output process stream */
output through cat > some_output_file.txt.

/* write the text to the process stream */
display txt.

/* close unnamed output process stream */
```

```
output close.
```

#16 - 08/16/2023 06:19 AM - Greg Shah

Anything using INPUT THROUGH, INPUT-OUTPUT THROUGH or OUTPUT THROUGH creates a child process and uses that as a stream.

#17 - 08/16/2023 06:20 AM - Greg Shah

Since the indeterminate order of input is an existing issue, we will ignore it here.

#18 - 08/16/2023 09:00 AM - Theodoros Theodorou

The branch 5538a has been rebased to trunk with rev no 14697.

#19 - 08/21/2023 03:13 PM - Theodoros Theodorou

Can I get into the queue for merging this branch?

#20 - 08/22/2023 07:28 AM - Greg Shah

How has it been tested?

#21 - 09/18/2023 03:39 AM - Theodoros Theodorou

- Status changed from Review to WIP

#22 - 09/19/2023 03:41 PM - Theodoros Theodorou

Greg Shah wrote:

> Anything using INPUT THROUGH, INPUT-OUTPUT THROUGH or OUTPUT THROUGH creates a child process and uses that as a stream.

I wrote the following test example:

```
DEFINE VARIABLE dir-name AS CHARACTER.

INPUT THROUGH 'echo test1234'.
 SET dir-name.
INPUT CLOSE.

message dir-name.
```

Indeed, it uses ProcessStream under the hood but it uses ProcessStream.readLn() instead of ProcessStream.readByte(), and I couldn't find any way to test the latter. Can you please check if there is any way to check ProcessStream.readByte()?

#23 - 09/19/2023 04:39 PM - Greg Shah

> Can you please check if there is any way to check ProcessStream.readByte()?

Try the IMPORT statement with a MEMPTR variable.

#24 - 09/21/2023 10:32 AM - Theodoros Theodorou

Greg Shah wrote:

> Try the IMPORT statement with a MEMPTR variable.

It worked for FWD, it triggered ProcessStream.readByte().

```
def var m as memptr no-undo.
SET-SIZE(m) = 3.

input through 'echo abc'.
import m.
input close.

message get-byte(m, 1). // In OE it returns 0
message get-byte(m, 2). // In OE it returns 0
message get-byte(m, 3). // In OE it returns 0
```

Now I have an issue writing a test in OE. If I run the above code, it doesn't add 'abc' in m. If I use def var m as char, it works fine.

#25 - 09/22/2023 02:38 PM - Ovidiu Maxiniuc

Indeed, in OE the content of memptr variable is not filled with data from process stream. I do not know why. I tried replacing it with a longchar instead (because, according to OE reference manual, they are sharing a dedicated IMPORT syntax) but I got a very specific error:
Can't import a longchar through a pipe (16504)

However, FWD reported the variable as empty. This is because the Stream.assignDatum() does not handle the longchar. It does not need to as the error should occur before that. (To note: there is a symmetric error: Can't export a longchar through a pipe (16503))

Anyway, I was unable to understand why the memptr does not work on OE :(. I tried several variations without any luck.

#26 - 09/22/2023 03:08 PM - Constantin Asofiei

This is an example where 'something' ends up in the memptr:

```
def var m as memptr no-undo.
SET-SIZE(m) = 6.

def var ch as char.
def var r as raw.

input through 'dir'.
import m.
input close.

message get-string(m, 1) get-byte(m, 1) get-byte(m, 2) get-byte(m, 3) get-byte(m, 4) get-byte(m, 5).
```

Only first byte is populated, with the byte from the position of the process output's the same as the memptr's length. So, if the length is 6, the 6'th byte is retrieved, etc. Note that dir starts with a space and after that Volume text. In the example above, it prints 'm'.

raw I can't make it work.

import unformatted, unbuferred, binary, have no effect from my testing.

#27 - 09/22/2023 03:29 PM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> This is an example where 'something' ends up in the memptr:
> [...]
>
> Only first byte is populated, with the byte from the position of the process output's the same as the memptr's length. So, if the length is 6, the 6'th byte is retrieved, etc. Note that dir starts with a space and after that Volume text. In the example above, it prints 'm'.

Nice observation. Really strange. And it gets even stranger. The testcase from #5538-24 works similarly if the process is replaced with echo /?. However, it will SKIP the first line (stderr maybe ?) and processed the second and following.

> raw I can't make it work.

Neither could I. It requires a different dataserver.

> import unformatted, unbuferred, binary, have no effect from my testing.

Same for me.

#28 - 09/25/2023 01:55 PM - Theodoros Theodorou

Thank you for confirming the bug Constantin and Ovidiu. I have created a test under testcases\oo\progress\io\input_stream\test_input_through.p which tests ProcessStream.readByte(). It passes with FWD but it fails with OE. Do you have any suggestions? Should I keep the part which fails with OE?

#29 - 09/25/2023 05:23 PM - Theodoros Theodorou

- Status changed from WIP to Review

#30 - 09/25/2023 05:32 PM - Ovidiu Maxiniuc

Although the natural instinct is to create good code, which behaves as one would expect (in this case to fully populate the memptr buffer), we must mimic the OE / 4GL behaviour. If a OE programmer is aware of this quirk and intentionally use this to achieve some goal, FWD should be transparent and the core should work as he (that programmer) expects.

So the answer is yes, you should keep the part which fails, not as a failure, but as a positive test. And we need to modify FWD to perform in the same manner.

#31 - 09/26/2023 11:06 AM - Theodoros Theodorou

Ok. Should we close this ticket? Should we create another ticket for implementing the same behavior with OE? Is the implementation of this 'buggy' feature important?

#32 - 09/26/2023 11:12 AM - Greg Shah

Theodoros Theodorou wrote:

> Ok. Should we close this ticket? Should we create another ticket for implementing the same behavior with OE? Is the implementation of this 'buggy' feature important?

I agree we can postpone the generation of compatible errors in these cases. Since it doesn't work in OE, it is unlikely to hit this code so it is lower priority. Please open another ticket in the Base Language project for these failures.

#33 - 09/26/2023 05:53 PM - Theodoros Theodorou

- Related to Feature #7842: Make FWD behave the same as OE regarding input through added

#34 - 09/27/2023 08:52 AM - Theodoros Theodorou

Please review branch 5538a rev no 14756. It has been rebased to trunk.

#35 - 09/27/2023 08:54 AM - Greg Shah

Ovidiu: Please review.

#36 - 09/27/2023 07:58 PM - Ovidiu Maxiniuc

- Start date deleted (09/26/2023)

- Status changed from Review to Test

Review branch 5538a / 14756.

There are minor changes (only some comments) since my last review. One thing which I failed to notice then is the throws clause of the new readByteWorker() which should go on separate line.

Do not forget to rebase to latest trunk.

#37 - 09/28/2023 09:28 AM - Greg Shah

Code Review Task Branch 5538a Revision 14756

It seems like we could rewrite the @@ as follows:

```
   private int readChWorker(boolean isConvert)
   throws StopConditionException
   {
      int ch = readByteWorker();

      // convert back to the raw value if needed since the out or err stream
      // handled translation to a Unicode char for us
      if (isConvert && ch >= 0)
      {
         ch = Utils.toInt((char) ch, cset);
      }

      return ch;
   }
```

Am I missing some reason this won't work?

Also, please rebase. When you post the notice of rebase, make sure you mention which trunk revision was used for the rebase.

#38 - 09/28/2023 03:28 PM - Theodoros Theodorou

Ovidiu Maxiniuc wrote:

> There are minor changes (only some comments) since my last review. One thing which I failed to notice then is the throws clause of the new readByteWorker() which should go on separate line.
>
> Do not forget to rebase to latest trunk.

Done

Greg Shah wrote:

> Code Review Task Branch 5538a Revision 14756
>
> It seems like we could rewrite the @@ as follows: [...]
> Am I missing some reason this won't work?

It will work. I copied the logic from readChWorker() and did some refactoring. Also, should I remove readChWorker completely and put the 5 lines of logic into readCh()?

> Also, please rebase. When you post the notice of rebase, make sure you mention which trunk revision was used for the rebase.

Done. The branch 5538a has been rebased to trunk rev no 14755. The new rev no is 14758.

#39 - 09/29/2023 11:15 AM - Greg Shah

Code Review Task Branch 5538a Revision 14758

No objections.

#40 - 09/29/2023 11:16 AM - Greg Shah

You can merge to trunk now.

Should we set %Done to 100?

#41 - 09/29/2023 12:32 PM - Theodoros Theodorou

- % Done changed from 0 to 100

#42 - 09/29/2023 12:33 PM - Theodoros Theodorou

Branch 5538a was merged to trunk rev. 14758 and archived.

#43 - 09/29/2023 12:33 PM - Greg Shah

- Status changed from Test to Closed