Standard output streams default behavior in terminal sessions

Didier Verna <didier@lrde.epita.fr>

Copying

This work may be distributed and/or modified under the conditions of the LaTeX Project Public License, either version 1.3 of this license or (at your option) any later version. The latest version of this license is in http://www.latex-project.org/lppl.txt and version 1.3 or later is part of all distributions of LaTeX version 2005/12/01 or later.

This work has the LPPL maintenance status 'maintained'.

The Current Maintainer of this work is Didier Verna.

1 Motivation

The Common Lisp standard mandates the existence of several streams such as `*STANDARD-OUTPUT*`, `*ERROR-OUTPUT*` and `*QUERY-IO*`. The purpose of these streams, however, is only informally described, leading to implementation-specific behavior.

This can be problematic for Lisp sessions started from a terminal (without a graphical user interface) and standalone command-line executables. As illustrated in the next section, the current behavior of some standard output streams, notably with respect to shell redirection may not only be different across implementations, but also contrary to the user's expectations.

The purpose of this document is hence to illustrate the problem and suggest that all Common Lisp implementations agree on one particular scheme (one actually already adopted by two of them).

2 Analysis

In order to analyze the effects of these underspecifications, some tests were conducted on 8 Common Lisp implementations on May 28th 2012, with the help of the code snippet depicted below.

```
(format *standard-output* "This goes to standard output.~%")
(format *error-output* "This goes to error output.~%")
(format *query-io* "This goes to query io.~%")
```

This code was stored in a test file, and loaded in three different stream redirection contexts. Care was taken to avoid launching any graphical user interface for implementations providing them. The contexts were as follows (adapt the exact command-line settings to every tested implementation):

```
cl --quit --load test.lisp
cl --quit --load test.lisp > log
cl --quit --load test.lisp > log 2>&1
```

The results of these tests are depicted in the table below. They exhibit 3 different sets of behaviors.

Compiler	Test case #1	Test case #2	Test case #3
SBCL	std-output -> tty	std-output -> log	std-output -> log
CMU-CL	err-output -> tty	err-output -> tty	err-output -> log
	query-io -> tty	query-io -> tty	query-io -> tty
ECL	std-output -> tty	std-output -> log	std-output -> log
	err-output -> tty	err-output -> tty	err-output -> log
	query-io -> tty	query-io -> log	query-io -> log
CLISP	std-output -> tty	std-output -> log	std-output -> log
CCL	err-output -> tty	err-output -> log	err-output -> log
ABCL	query-io -> tty	query-io -> log	query-io -> log
LispWorks			
Allegro			

We believe that the behavior of SBCL and CMU-CL is the most intuitive one. Shell redirections of the system's `stdin` and `stdout` are honored by `*STANDARD-OUTPUT*` and `*ERROR-OUTPUT*`, and `*QUERY-IO*` always stays on the terminal.

The current behavior of SBCL and CMU-CL is informally described as follows by Nikodemus Siivola:

> There are streams `*STDIN*`, `*STDOUT*`, `*STDERR*` and `*TTY*`. If '/dev/tty' cannot be opened, `*TTY*` is simply (`make-two-way-stream *stdin* *stdout*`).
>
> `*STANDARD-INPUT*`, `*STANDARD-OUTPUT*` and `*ERROR-OUTPUT*` start out as synonym streams for `*STDIN*`, `*STDOUT*` and `*STDERR*`. `*TERMINAL-IO*`, `*QUERY-IO*` and `*DEBUG-IO*` start out as synonym streams for `*TTY*`.

ECL behaves almost the same. The difference is with `*QUERY-IO*` which follows the behavior of `*STANDARD-OUTPUT*`. The Common Lisp standard stipulates that this stream "should be used when asking questions to the user". Consequently, the current behavior of ECL is problematic because in cases #2 and #3, the user would never see the questions asked.

The remaining implementations suffer from the same problem as ECL with respect to `*QUERY-IO*`. An additional problem lies in the fact that `*ERROR-OUTPUT*` follows the behavior of `*STANDARD-OUTPUT*`, even in case #2 where the system's `stderr` is not redirected. We think that this behavior is counter-intuitive as well.

3 Proposal

In light of this analysis, we think that CMU-CL and SBCL offer the most intuitive behavior, as it closely matches what one would expect from a regular command-line application operating in a POSIX environment. Therefore, we suggest that all implementations conform to this behavior when run in a non-graphical mode.