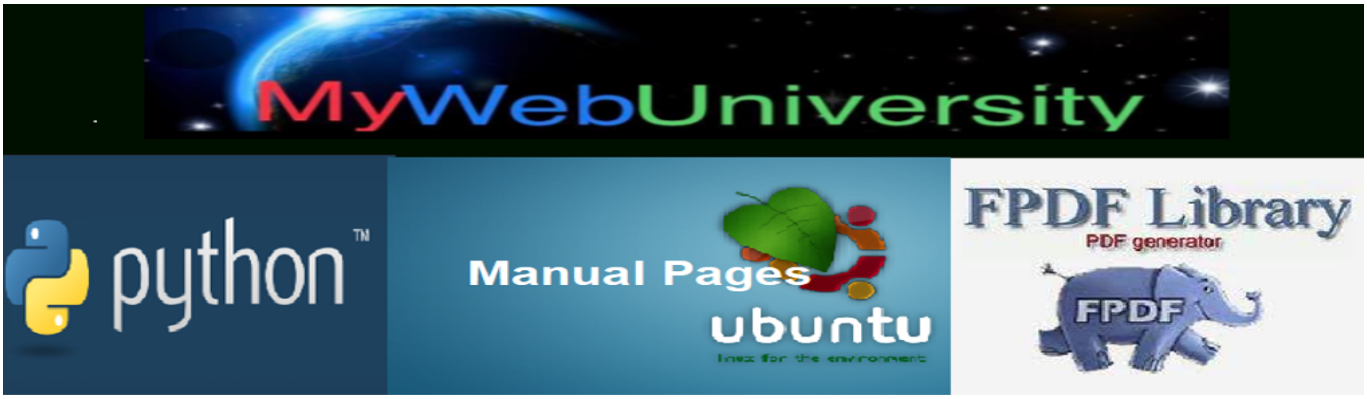

Rocky Enterprise Linux 9.2 Manual Pages on command 'Test2::IPC::Driver.3perl'

$ man Test2::IPC::Driver.3perl

NAME

Test2::IPC::Driver - Base class for Test2 IPC drivers.

SYNOPSIS

```
package Test2::IPC::Driver::MyDriver;

use base 'Test2::IPC::Driver';

...
```

METHODS

$self->abort($msg)

If an IPC encounters a fatal error it should use this. This will print the message to STDERR with 'IPC Fatal Error: ' prefixed to it, then it will forcefully exit 255. IPC errors may occur in threads or processes other than the main one, this method provides the best chance of the harness noticing the error.

$self->abort_trace($msg)

This is the same as "$ipc->abort($msg)" except that it uses "Carp::longmess" to add a stack trace to the message.

LOADING DRIVERS

Test2::IPC::Driver has an "import()" method. All drivers inherit this import method. This import method registers the driver.

In most cases you just need to load the desired IPC driver to make it work. You should load this driver as early as possible. A warning will be issued if you load it too late for it to be effective.

```
use Test2::IPC::Driver::MyDriver;
...
```

WRITING DRIVERS

```
package Test2::IPC::Driver::MyDriver;
use strict;
use warnings;

use base 'Test2::IPC::Driver';

sub is_viable {
    return 0 if $^O eq 'win32'; # Will not work on windows.
    return 1;
}

sub add_hub {
    my $self = shift;
    my ($hid) = @_;

    ... # Make it possible to contact the hub
}

sub drop_hub {
    my $self = shift;
    my ($hid) = @_;
```

```
    ... # Nothing should try to reach the hub anymore.
}

sub send {
    my $self = shift;
    my ($hid, $e, $global) = @_;

    ... # Send the event to the proper hub.

    # This may notify other procs/threads that there is a pending event.
    Test2::API::test2_ipc_set_pending($uniq_val);
}

sub cull {
    my $self = shift;
    my ($hid) = @_;

    my @events = ...; # Here is where you get the events for the hub

    return @events;
}

sub waiting {
    my $self = shift;

    ... # Notify all listening procs and threads that the main
    ... # process/thread is waiting for them to finish.
}

1;
```

$ipc->is_viable

This should return true if the driver works in the current environment. This should return false if it does not. This is a CLASS method.

$ipc->add_hub($hid)

This is used to alert the driver that a new hub is expecting events. The driver should keep track of the process and thread ids, the hub should only be dropped by the proc+thread that started it.

```
sub add_hub {
    my $self = shift;
    my ($hid) = @_;

    ... # Make it possible to contact the hub
}
```

$ipc->drop_hub($hid)

This is used to alert the driver that a hub is no longer accepting events. The driver should keep track of the process and thread ids, the hub should only be dropped by the proc+thread that started it (This is the drivers responsibility to enforce).

```
sub drop_hub {
    my $self = shift;
    my ($hid) = @_;

    ... # Nothing should try to reach the hub anymore.
}
```

$ipc->send($hid, $event);

$ipc->send($hid, $event, $global);

Used to send events from the current process/thread to the specified hub in its process+thread.

```
    sub send {
        my $self = shift;
        my ($hid, $e) = @_;

        ... # Send the event to the proper hub.

        # This may notify other procs/threads that there is a pending event.
        Test2::API::test2_ipc_set_pending($uniq_val);
    }
```

If $global is true then the driver should send the event to all hubs in all processes and threads.

@events = $ipc->cull($hid)

Used to collect events that have been sent to the specified hub.

```
    sub cull {
        my $self = shift;
        my ($hid) = @_;

        my @events = ...; # Here is where you get the events for the hub

        return @events;
    }
```

$ipc->waiting()

This is called in the parent process when it is complete and waiting for all child processes and threads to complete.

```
    sub waiting {
        my $self = shift;

        ... # Notify all listening procs and threads that the main
```

```
    ... # process/thread is waiting for them to finish.
}
```

METHODS SUBCLASSES MAY IMPLEMENT OR OVERRIDE

$ipc->driver_abort($msg)

This is a hook called by "Test2::IPC::Driver->abort()". This is your chance to cleanup when an abort happens. You cannot prevent the abort, but you can gracefully except it.

SOURCE

The source code repository for Test2 can be found at http://github.com/Test-More/test-more/.

MAINTAINERS

Chad Granum <exodist@cpan.org>

AUTHORS

Chad Granum <exodist@cpan.org>

COPYRIGHT