

**NAME**

IPC::Run::Debug – debugging routines for IPC::Run

**SYNOPSIS**

```
##
## Environment variable usage
##
## To force debugging off and shave a bit of CPU and memory
## by compile-time optimizing away all debugging code in IPC::Run
## (debug => ...) options to IPC::Run will be ignored.
export IPCRUNDEBUB=none

## To force debugging on (levels are from 0..10)
export IPCRUNDEBUB=basic

## Leave unset or set to "" to compile in debugging support and
## allow runtime control of it using the debug option.
```

**DESCRIPTION**

Controls IPC::Run debugging. Debugging levels are now set by using words, but the numbers shown are still supported for backwards compatibility:

0	none	disabled (special, see below)
1	basic	what's running
2	data	what's being sent/received
3	details	what's going on in more detail
4	gory	way too much detail for most uses
10	all	use this when submitting bug reports
	noopts	optimizations forbidden due to inherited STDIN

The none level is special when the environment variable IPCRUNDEBUB is set to this the first time IPC::Run::Debug is loaded: it prevents the debugging code from being compiled in to the remaining IPC::Run modules, saving a bit of cpu.

To do this in a script, here's a way that allows it to be overridden:

```
BEGIN {
    unless ( defined $ENV{IPCRUNDEBUB} ) {
        eval 'local $ENV{IPCRUNDEBUB} = "none"; require IPC::Run::Debug'
        or die $@;
    }
}
```

This should force IPC::Run to not be debuggable unless somebody sets the IPCRUNDEBUB flag; modify this formula to grep @ARGV if need be:

```
BEGIN {
    unless ( grep /^--debug/, @ARGV ) {
        eval 'local $ENV{IPCRUNDEBUB} = "none"; require IPC::Run::Debug'
        or die $@;
    }
}
```

Both of those are untested.

**AUTHOR**

Barrie Slaymaker <barries@slaysys.com>, with numerous suggestions by p5p.