```
use IPC::Open3;
use Symbol qw(gensym);
use IO::File;
local *CATCHOUT = IO::File->new_tempfile;
local *CATCHERR = IO::File->new_tempfile;
my $pid = open3(gensym, ">&CATCHOUT", ">&CATCHERR", "cmd");
waitpid($pid, 0);
seek $_, 0, 0 for \*CATCHOUT, \*CATCHERR;
while( <CATCHOUT> ) {}
while( <CATCHERR> ) {}
```

But there's no real need for *both* to be tempfiles... the following should work just as well, without deadlocking:

```
use IPC::Open3;
use Symbol qw(gensym);
use IO::File;
local *CATCHERR = IO::File->new_tempfile;
my $pid = open3(gensym, \*CATCHOUT, ">&CATCHERR", "cmd");
while( <CATCHOUT> ) {}
waitpid($pid, 0);
seek CATCHERR, 0, 0;
while( <CATCHERR> ) {}
```

And it'll be faster, too, since we can begin processing the program's stdout immediately, rather than waiting for the program to finish.

With any of these, you can change file descriptors before the call:

```
open(STDOUT, ">logfile");
system("ls");
```

or you can use Bourne shell file-descriptor redirection:

```
$output = `$cmd 2>some_file`;
open (PIPE, "cmd 2>some_file |");
```

You can also use file-descriptor redirection to make STDERR a duplicate of STDOUT:

```
$output = `$cmd 2>&1`;
open (PIPE, "cmd 2>&1 |");
```

Note that you *cannot* simply open STDERR to be a dup of STDOUT in your Perl program and avoid calling the shell to do the redirection. This doesn't work:

```
open(STDERR, ">&STDOUT");
$alloutput = `cmd args`;  # stderr still escapes
```

This fails because the open() makes STDERR go to where STDOUT was going at the time of the open(). The backticks then make STDOUT go to a string, but don't change STDERR (which still goes to the old STDOUT).

Note that you *must* use Bourne shell (sh(1)) redirection syntax in backticks, not csh(1)! Details on why Perl's system() and backtick and pipe opens all use the Bourne shell are in the *versus/csh.whynot* article in the "Far More Than You Ever Wanted To Know" collection in http://www.cpan.org/misc/olddoc/FMTEYEWTK.tgz . To capture a command's STDERR and STDOUT together:

```
$output = `cmd 2>&1`;                       # either with backticks
$pid = open(PH, "cmd 2>&1 |");              # or with an open pipe
while (<PH>) { }                            #    plus a read
```