1 Apache::SubProcess -- Executing SubProcesses from mod_perl

1.1 SYNOPSIS

```
use Apache::SubProcess ();

use Config;
use constant PERLIO_IS_ENABLED => $Config{useperlio};

# pass @ARGV / read from the process
$command = "/tmp/argv.pl";
@argv = qw(foo bar);
$out_fh = Apache::SubProcess::spawn_proc_prog($r, $command, \@argv);
$output = read_data($out_fh);

# pass environment / read from the process
$command = "/tmp/env.pl";
$r->subprocess_env->set(foo => "bar");
$out_fh = Apache::SubProcess::spawn_proc_prog($r, $command);
$output = read_data($out_fh);

# write to/read from the process
$command = "/tmp/in_out_err.pl";
($in_fh, $out_fh, $err_fh) =
    Apache::SubProcess::spawn_proc_prog($r, $command);
print $in_fh "hello\n";
$output = read_data($out_fh);
$error  = read_data($err_fh);

# helper function to work w/ and w/o perlio-enabled Perl
sub read_data {
    my($fh) = @_;
    my $data;
    if (PERLIO_IS_ENABLED || IO::Select->new($fh)->can_read(10)) {
        $data = <$fh>;
    }
    return defined $data ? $data : '';
}
```

1.2 DESCRIPTION

`Apache::SubProcess` provides the Perl API for running and communicating with processes spawned from mod_perl handlers.

1.3 API

1.3.1 spawn_proc_prog()

```
$out_fh =
    Apache::SubProcess::spawn_proc_prog($r, $command, [\@argv]);
($in_fh, $out_fh, $err_fh) =
    Apache::SubProcess::spawn_proc_prog($r, $command, [\@argv]);
```

spawn_proc_prog() spawns a sub-process which exec()'s `$command` and returns the output pipe filehandle in the scalar context, or input, output and error pipe filehandles in the list context. Using these three pipes it's possible to communicate with the spawned process.

The third optional argument is a reference to an array which if passed becomes ARGV to the spawned program.

It's possible to pass environment variables as well, by calling:

```
  $r->subprocess_env->set($key => $value);
```

before spawning the subprocess.

There is an issue with reading from the read filehandle (`$in_fh`)):

A pipe filehandle returned under perlio-disabled Perl needs to call select() if the other end is not fast enough to send the data, since the read is non-blocking.

A pipe filehandle returned under perlio-enabled Perl on the other hand does the select() internally, because it's really a filehandle opened via `:APR` layer, which internally uses APR to communicate with the pipe. The way APR is implemented Perl's select() cannot be used with it (mainly because select() wants fileno() and APR is a crossplatform implementation which hides the internal datastructure).

Therefore to write a portable code, you want to use select for perlio-disabled Perl and do nothing for perlio-enabled Perl, hence you can use something similar to the read_data() wrapper shown in the SYNOPSIS section.

Table of Contents:

1 Apache::SubProcess -- Executing SubProcesses from mod_perl 1
1.1 SYNOPSIS 2
1.2 DESCRIPTION 2
1.3 API 2
1.3.1 spawn_proc_prog() 2