Writing Tests with `Apache-Test` Part II

Geoffrey Young

geoff@modperlcookbook.org

Apache-Test

- Framework for testing Apache-based application components
- Part of the mod_perl ASF project
- Provides tools to make testing Apache related things simple
- Written in Perl
- Familiar interface
  - `Test.pm`, `Test::More`, `LWP`, etc

Last Year...

- I introduced `Apache-Test` mechanics
- Everyone was impressed
- Some people even started using it
- There's so much I didn't tell you
- But first, let's review...

Testing, Testing... 1, 2, 3

1. Perl `Makefile.PL` foo
2. Apache `httpd.conf` foo
3. Write the tests

Step 1 - Makefile.PL

```
use Apache::TestMM qw(test clean);
use Apache::TestRunPerl ();

# configure tests based on incoming arguments
Apache::TestMM::filter_args();

# generate the test harness (t/TEST)
Apache::TestRunPerl->generate_script();
```

Step 2 – Apache Foo

- Apache needs a basic configuration to service requests

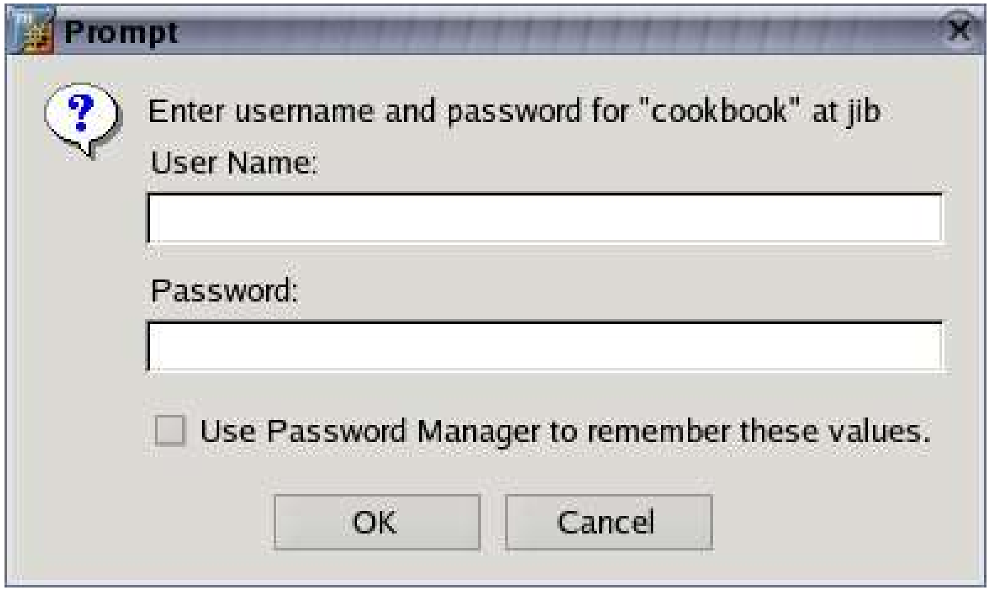
– `ServerRoot`	`t/`
– `DocumentRoot`	`t/htdocs`
– `ErrorLog`	`t/logs/error_log`
– `Listen`	`8529`
– `LoadModule`	

- `Apache-Test` "intuits" these and creates its own `httpd.conf`
- Configures all that is required to `GET`

```
http://localhost:8529/index.html
```

More Apache Foo

- When testing we generally need more than the default
- Instead of altering `httpd.conf` we augment it with a special file
- `t/conf/extra.conf.in`
  - pulled in via `Include`
  - provides handy variable substitution


Prompt

Enter username and password for "cookbook" at jib

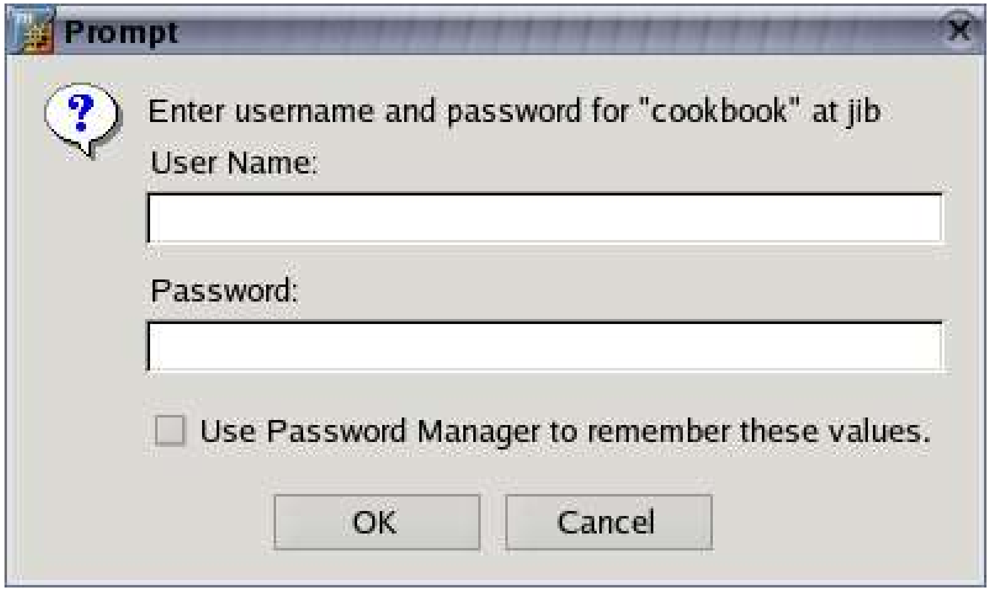
User Name:

Password:

t/conf/extra.conf.in

```
Alias /basic @DocumentRoot@

<Location /basic>
  Require valid-user
  AuthType Basic
  AuthName myrealm

  PerlAuthenHandler Apache::BasicAuth
</Location>
```

BasicAuth.pm


Enter username and password for "cookbook" at jib

User Name:

Password:

```
package My::AuthenHandler;

use Apache2::Const -compile => qw(OK HTTP_UNAUTHORIZED);

use Apache2::RequestRec ();
use Apache2::Access ();

sub handler {

  my $r = shift;

  # Get the client-supplied credentials.
  my ($status, $password) = $r->get_basic_auth_pw;

  return $status unless $status == Apache2::Const::OK;

  # Perform some custom user/password validation.
  return Apache2::Const::OK if $r->user eq $password;

  # Whoops, bad credentials.
  $r->note_basic_auth_failure;
  return Apache2::Const::HTTP_UNAUTHORIZED;
}

1;
```

Whew!

- At this point we have...
  - given Apache a minimal config
  - configured a protected URL
  - created a mod_perl handler to do the authentication
- *Now* we write the tests

Step 3 – The Tests

- Standard Perl foo

```
plan tests => 3;

ok ($condition);

etc...
```

t/01basic.t

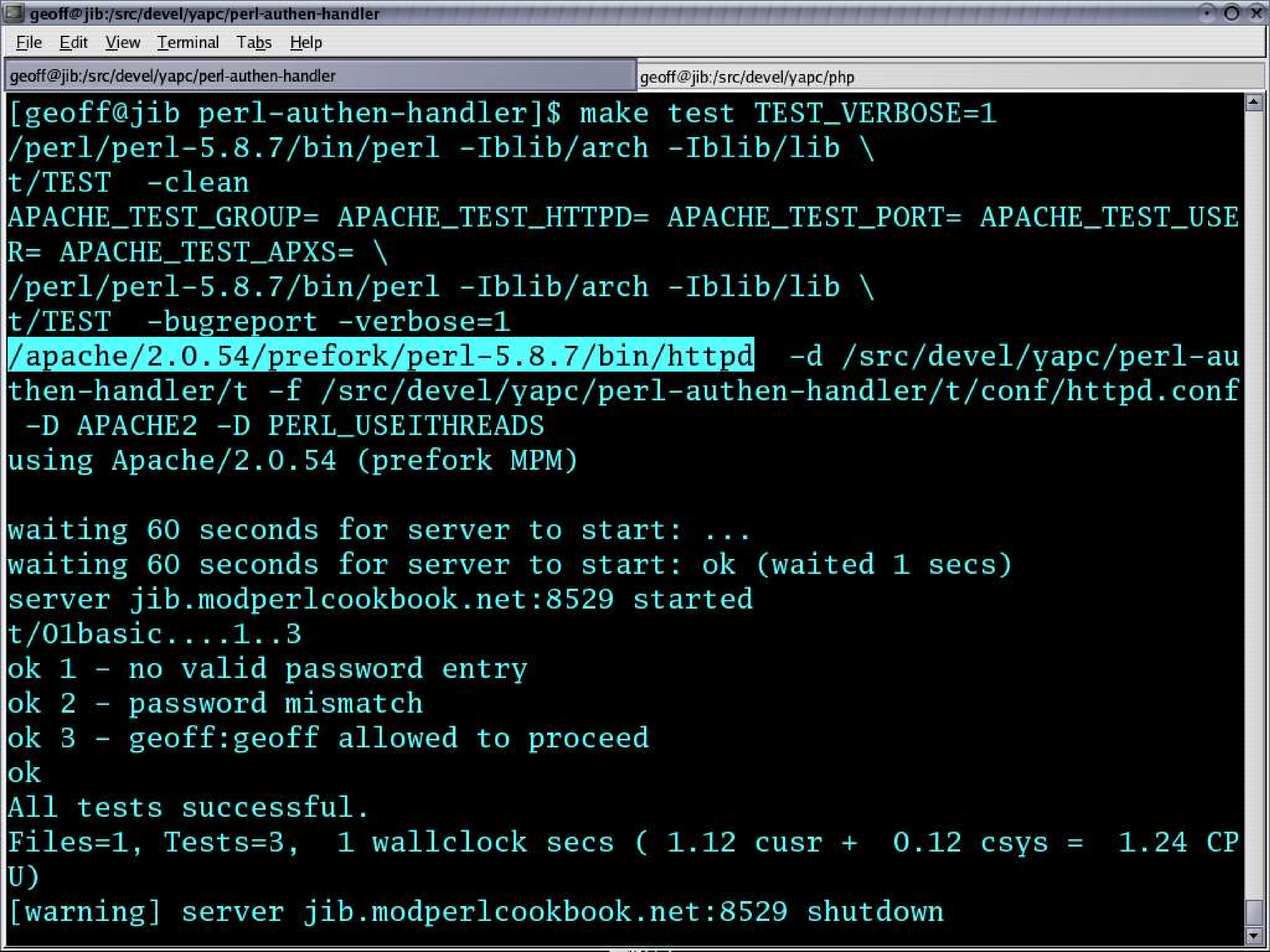
```
{
  my $response = GET '/basic/index.html';

  is ($response->code,
      401,
      "no valid password entry");
}

{
  my $response = GET '/basic/index.html',
                     username => 'geoff',
                     password => 'geoff';

  is ($response->code,
      200,
      "geoff:geoff allowed to proceed");
}
```

Testing, Testing... 1, 2, 3

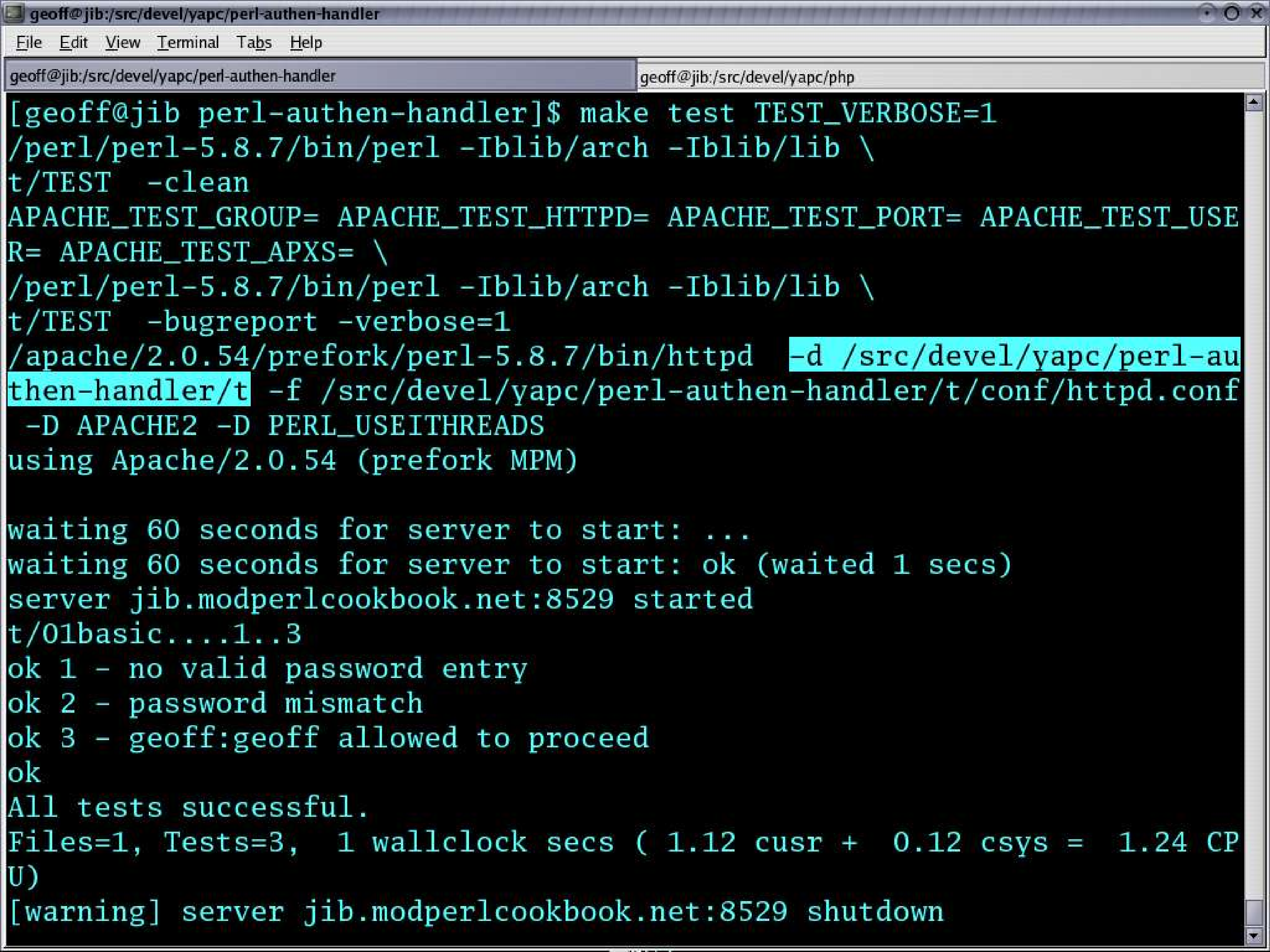
1. Generate the test harness
2. Configure Apache
3. Write the tests
4. Run the tests

Running the Tests

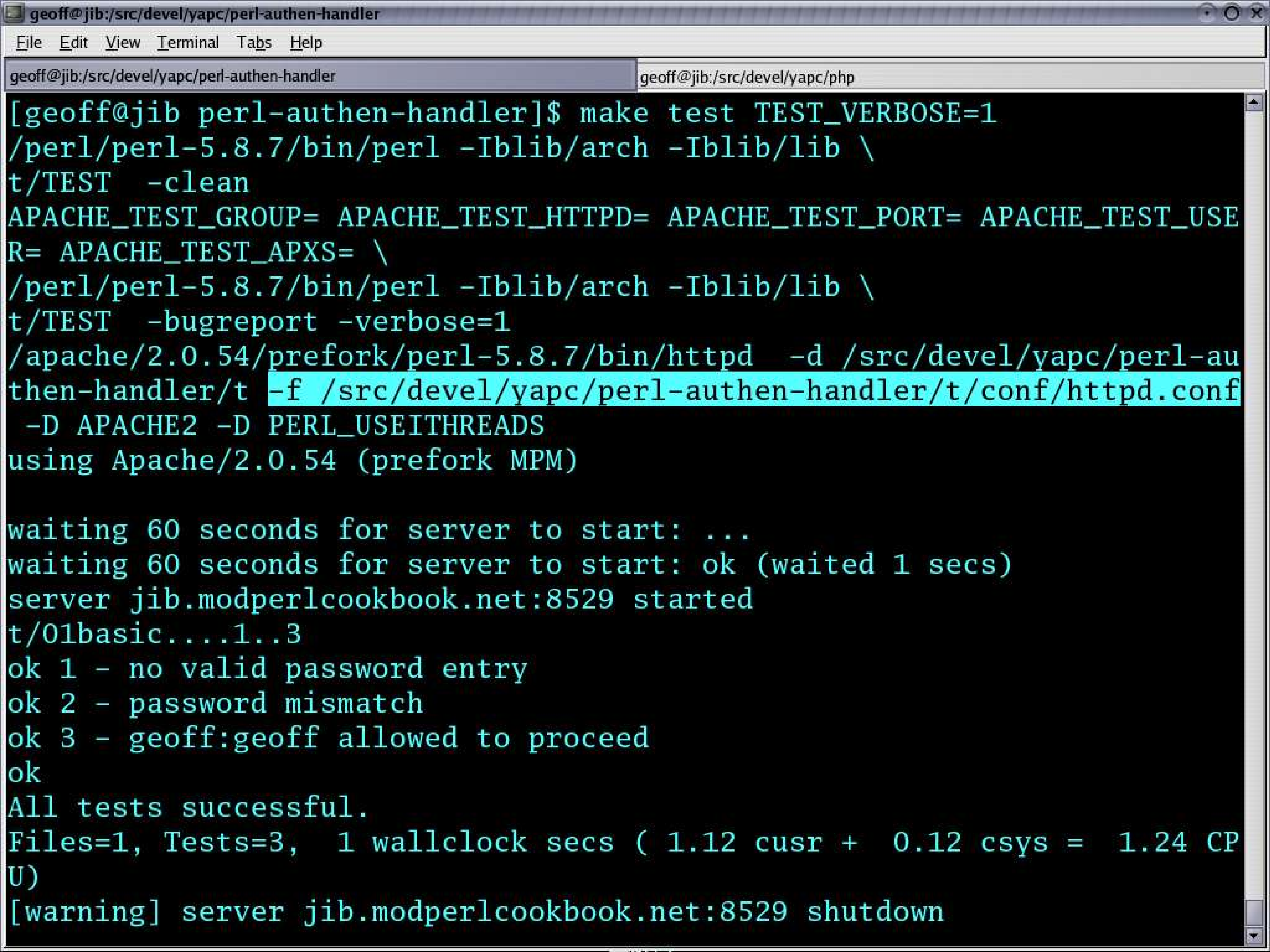
- Initial setup is a bit complex
  - you only do it once
- Running the tests is simple
  - `make test`
  - `t/TEST`


geoff@jib:/src/devel/yapc/perl-authen-handler
File Edit View Terminal Tabs Help
geoff@jib:/src/devel/yapc/perl-authen-handler
geoff@jib:/src/devel/yapc/php

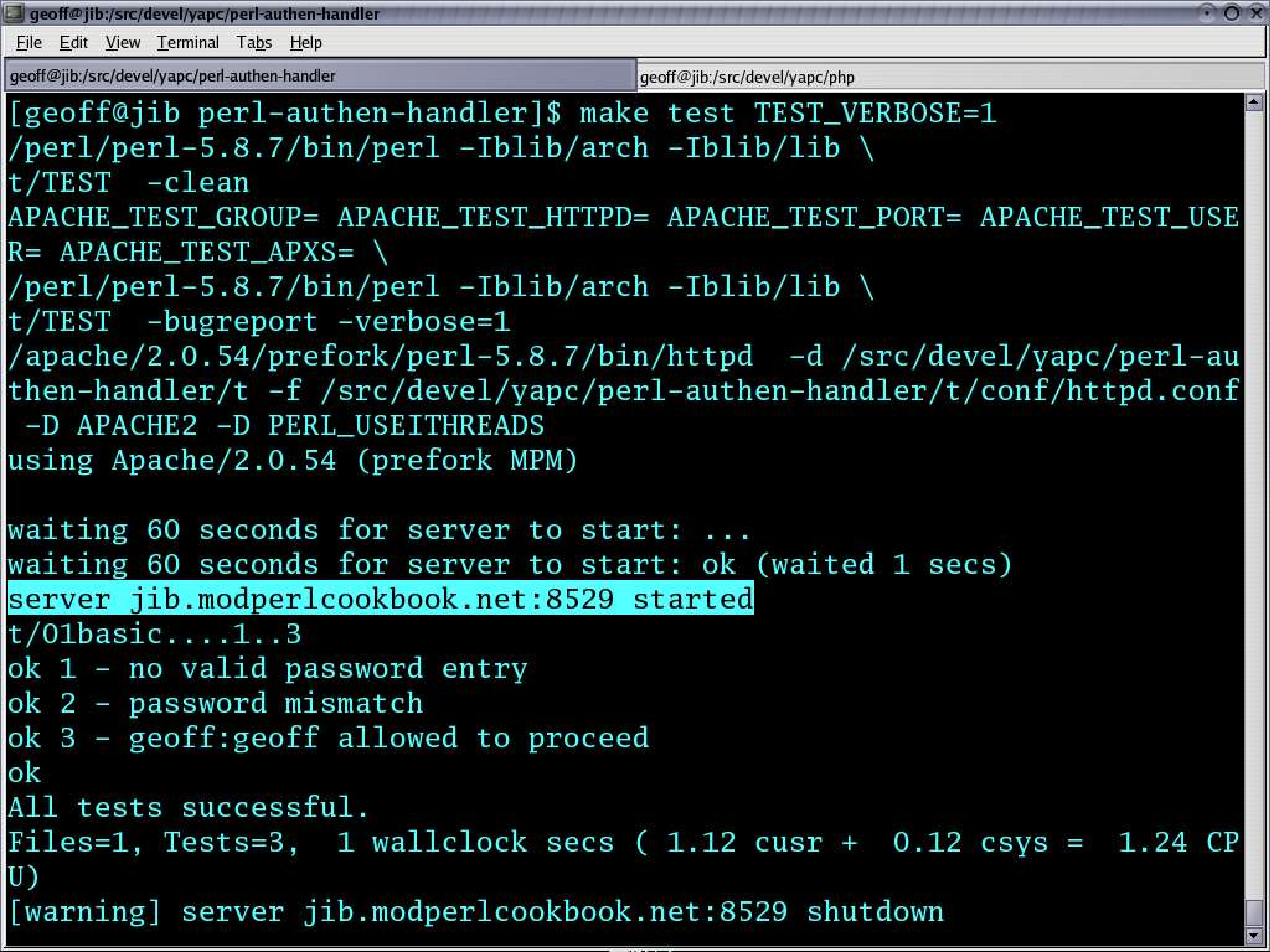
```
[geoff@jib perl-authen-handler]$ make test TEST_VERBOSE=1
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST  -clean
APACHE_TEST_GROUP= APACHE_TEST_HTTPD= APACHE_TEST_PORT= APACHE_TEST_USE
R= APACHE_TEST_APXS= \
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST  -bugreport -verbose=1
/apache/2.0.54/prefork/perl-5.8.7/bin/httpd  -d /src/devel/yapc/perl-au
then-handler/t -f /src/devel/yapc/perl-authen-handler/t/conf/httpd.conf
 -D APACHE2 -D PERL_USEITHREADS
using Apache/2.0.54 (prefork MPM)

waiting 60 seconds for server to start: ...
waiting 60 seconds for server to start: ok (waited 1 secs)
server jib.modperlcookbook.net:8529 started
t/01basic....1..3
ok 1 - no valid password entry
ok 2 - password mismatch
ok 3 - geoff:geoff allowed to proceed
ok
All tests successful.
Files=1, Tests=3,  1 wallclock secs ( 1.12 cusr +  0.12 csys =  1.24 CP
U)
[warning] server jib.modperlcookbook.net:8529 shutdown
```


geoff@jib:/src/devel/yapc/perl-authen-handler
File Edit View Terminal Tabs Help
geoff@jib:/src/devel/yapc/perl-authen-handler
geoff@jib:/src/devel/yapc/php

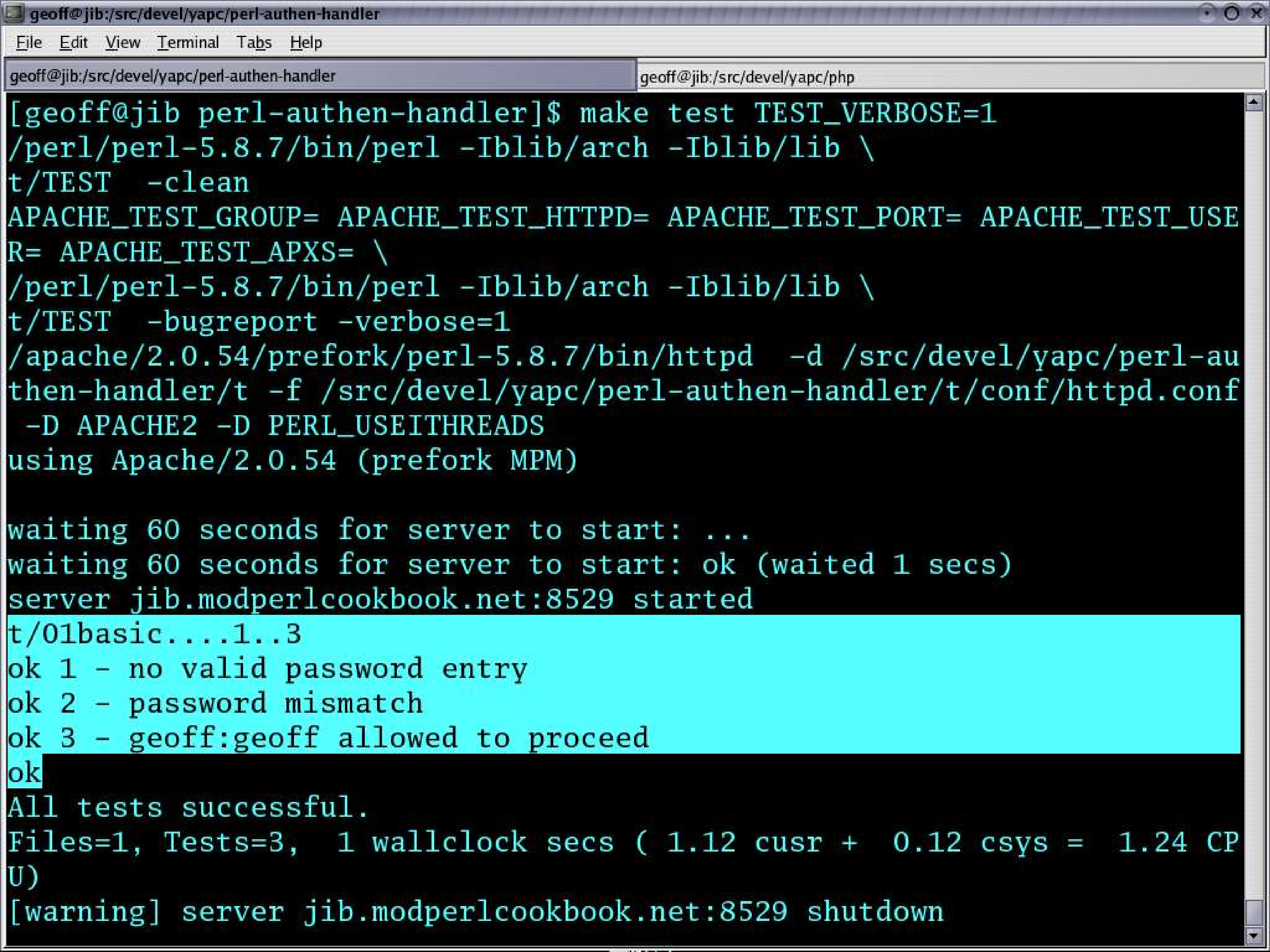
```
[geoff@jib perl-authen-handler]$ make test TEST_VERBOSE=1
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST  -clean
APACHE_TEST_GROUP= APACHE_TEST_HTTPD= APACHE_TEST_PORT= APACHE_TEST_USE
R= APACHE_TEST_APXS= \
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST  -bugreport -verbose=1
/apache/2.0.54/prefork/perl-5.8.7/bin/httpd  -d /src/devel/yapc/perl-au
then-handler/t -f /src/devel/yapc/perl-authen-handler/t/conf/httpd.conf
 -D APACHE2 -D PERL_USEITHREADS
using Apache/2.0.54 (prefork MPM)

waiting 60 seconds for server to start: ...
waiting 60 seconds for server to start: ok (waited 1 secs)
server jib.modperlcookbook.net:8529 started
t/01basic....1..3
ok 1 - no valid password entry
ok 2 - password mismatch
ok 3 - geoff:geoff allowed to proceed
ok
All tests successful.
Files=1, Tests=3,  1 wallclock secs ( 1.12 cusr +  0.12 csys =  1.24 CP
U)
[warning] server jib.modperlcookbook.net:8529 shutdown
```


geoff@jib:/src/devel/yapc/perl-authen-handler
File Edit View Terminal Tabs Help
geoff@jib:/src/devel/yapc/perl-authen-handler
geoff@jib:/src/devel/yapc/php

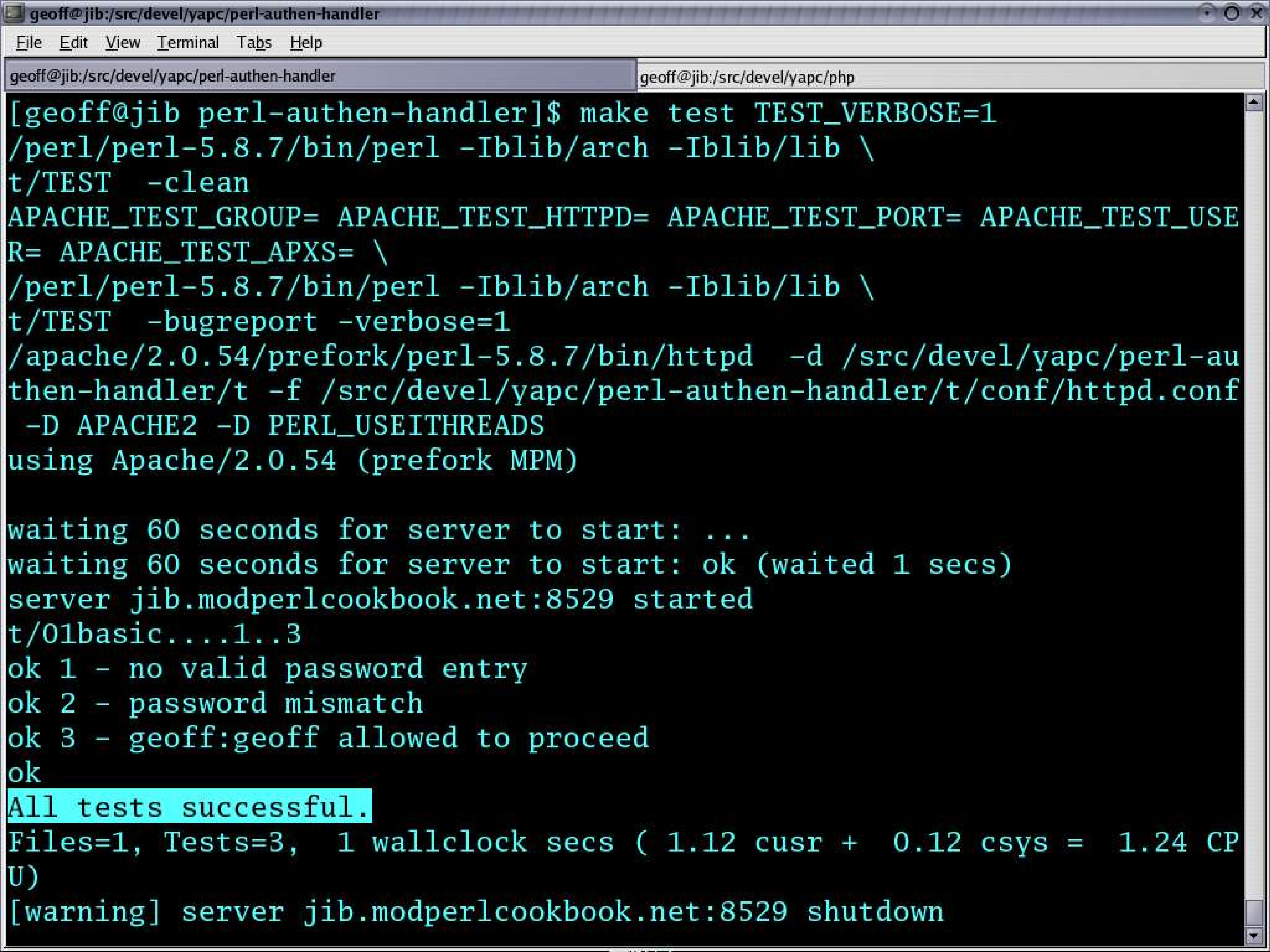
```
[geoff@jib perl-authen-handler]$ make test TEST_VERBOSE=1
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST  -clean
APACHE_TEST_GROUP= APACHE_TEST_HTTPD= APACHE_TEST_PORT= APACHE_TEST_USE
R= APACHE_TEST_APXS= \
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST  -bugreport -verbose=1
/apache/2.0.54/prefork/perl-5.8.7/bin/httpd  -d /src/devel/yapc/perl-au
then-handler/t -f /src/devel/yapc/perl-authen-handler/t/conf/httpd.conf
 -D APACHE2 -D PERL_USEITHREADS
using Apache/2.0.54 (prefork MPM)

waiting 60 seconds for server to start: ...
waiting 60 seconds for server to start: ok (waited 1 secs)
server jib.modperlcookbook.net:8529 started
t/01basic....1..3
ok 1 - no valid password entry
ok 2 - password mismatch
ok 3 - geoff:geoff allowed to proceed
ok
All tests successful.
Files=1, Tests=3,  1 wallclock secs ( 1.12 cusr +  0.12 csys =  1.24 CP
U)
[warning] server jib.modperlcookbook.net:8529 shutdown
```


geoff@jib:/src/devel/yapc/perl-authen-handler
File Edit View Terminal Tabs Help
geoff@jib:/src/devel/yapc/perl-authen-handler
geoff@jib:/src/devel/yapc/php

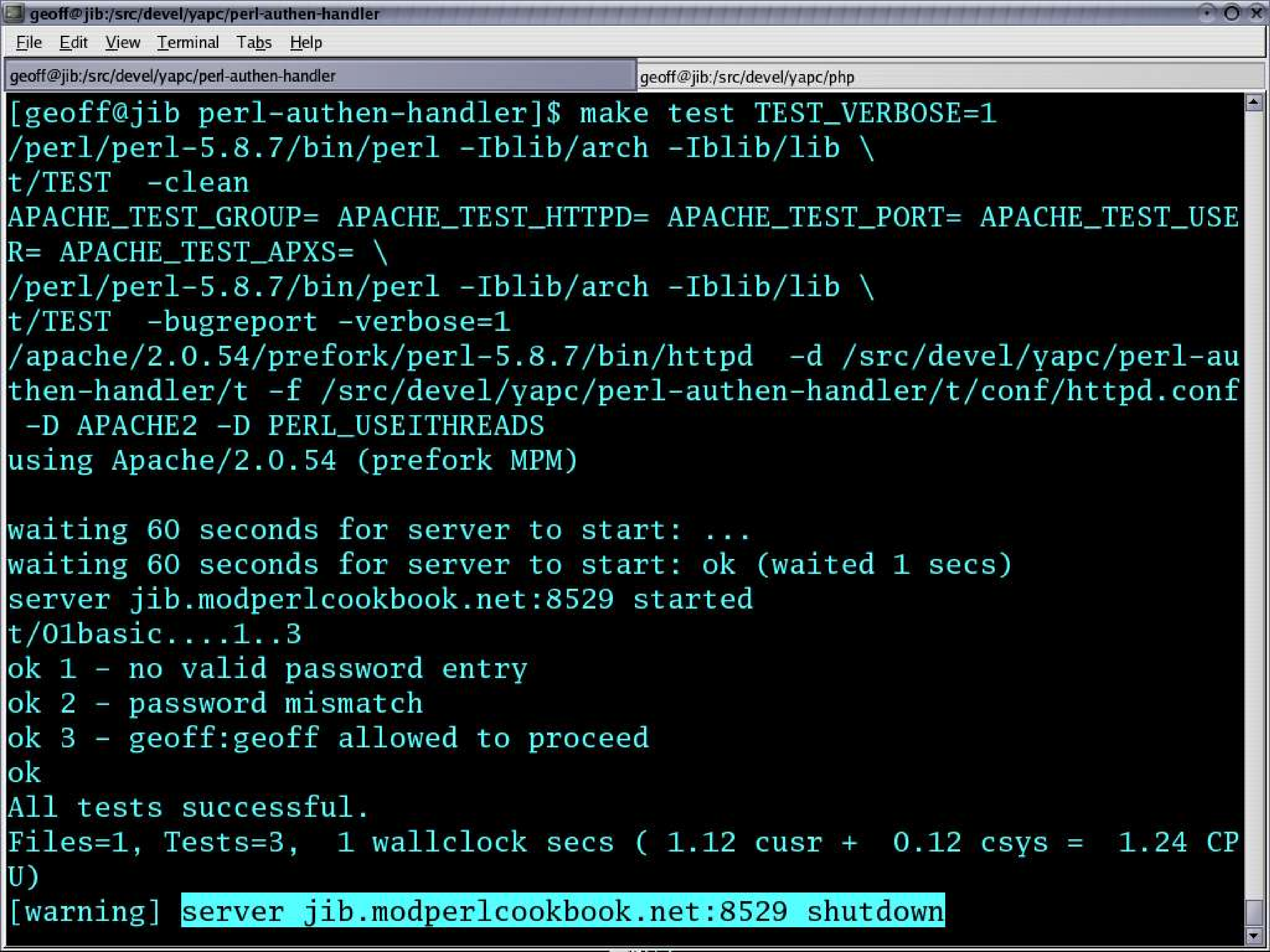
```
[geoff@jib perl-authen-handler]$ make test TEST_VERBOSE=1
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST  -clean
APACHE_TEST_GROUP= APACHE_TEST_HTTPD= APACHE_TEST_PORT= APACHE_TEST_USE
R= APACHE_TEST_APXS= \
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST  -bugreport -verbose=1
/apache/2.0.54/prefork/perl-5.8.7/bin/httpd  -d /src/devel/yapc/perl-au
then-handler/t -f /src/devel/yapc/perl-authen-handler/t/conf/httpd.conf
 -D APACHE2 -D PERL_USEITHREADS
using Apache/2.0.54 (prefork MPM)

waiting 60 seconds for server to start: ...
waiting 60 seconds for server to start: ok (waited 1 secs)
server jib.modperlcookbook.net:8529 started
t/01basic....1..3
ok 1 - no valid password entry
ok 2 - password mismatch
ok 3 - geoff:geoff allowed to proceed
ok
All tests successful.
Files=1, Tests=3,  1 wallclock secs ( 1.12 cusr +  0.12 csys =  1.24 CP
U)
[warning] server jib.modperlcookbook.net:8529 shutdown
```


geoff@jib:/src/devel/yapc/perl-authen-handler
File Edit View Terminal Tabs Help
geoff@jib:/src/devel/yapc/perl-authen-handler
geoff@jib:/src/devel/yapc/php

```
[geoff@jib perl-authen-handler]$ make test TEST_VERBOSE=1
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST  -clean
APACHE_TEST_GROUP= APACHE_TEST_HTTPD= APACHE_TEST_PORT= APACHE_TEST_USE
R= APACHE_TEST_APXS= \
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST  -bugreport -verbose=1
/apache/2.0.54/prefork/perl-5.8.7/bin/httpd  -d /src/devel/yapc/perl-au
then-handler/t -f /src/devel/yapc/perl-authen-handler/t/conf/httpd.conf
 -D APACHE2 -D PERL_USEITHREADS
using Apache/2.0.54 (prefork MPM)

waiting 60 seconds for server to start: ...
waiting 60 seconds for server to start: ok (waited 1 secs)
server jib.modperlcookbook.net:8529 started
t/01basic....1..3
ok 1 - no valid password entry
ok 2 - password mismatch
ok 3 - geoff:geoff allowed to proceed
ok
All tests successful.
Files=1, Tests=3,  1 wallclock secs ( 1.12 cusr +  0.12 csys =  1.24 CP
U)
[warning] server jib.modperlcookbook.net:8529 shutdown
```


geoff@jib:/src/devel/yapc/perl-authen-handler
File Edit View Terminal Tabs Help
geoff@jib:/src/devel/yapc/perl-authen-handler
geoff@jib:/src/devel/yapc/php

```
[geoff@jib perl-authen-handler]$ make test TEST_VERBOSE=1
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST  -clean
APACHE_TEST_GROUP= APACHE_TEST_HTTPD= APACHE_TEST_PORT= APACHE_TEST_USE
R= APACHE_TEST_APXS= \
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST  -bugreport -verbose=1
/apache/2.0.54/prefork/perl-5.8.7/bin/httpd  -d /src/devel/yapc/perl-au
then-handler/t -f /src/devel/yapc/perl-authen-handler/t/conf/httpd.conf
 -D APACHE2 -D PERL_USEITHREADS
using Apache/2.0.54 (prefork MPM)

waiting 60 seconds for server to start: ...
waiting 60 seconds for server to start: ok (waited 1 secs)
server jib.modperlcookbook.net:8529 started
t/01basic....1..3
ok 1 - no valid password entry
ok 2 - password mismatch
ok 3 - geoff:geoff allowed to proceed
ok
All tests successful.
Files=1, Tests=3,  1 wallclock secs ( 1.12 cusr +  0.12 csys =  1.24 CP
U)
[warning] server jib.modperlcookbook.net:8529 shutdown
```


geoff@jib:/src/devel/yapc/perl-authen-handler
File Edit View Terminal Tabs Help
geoff@jib:/src/devel/yapc/perl-authen-handler
geoff@jib:/src/devel/yapc/php

```
[geoff@jib perl-authen-handler]$ make test TEST_VERBOSE=1
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST  -clean
APACHE_TEST_GROUP= APACHE_TEST_HTTPD= APACHE_TEST_PORT= APACHE_TEST_USE
R= APACHE_TEST_APXS= \
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST  -bugreport -verbose=1
/apache/2.0.54/prefork/perl-5.8.7/bin/httpd  -d /src/devel/yapc/perl-au
then-handler/t -f /src/devel/yapc/perl-authen-handler/t/conf/httpd.conf
 -D APACHE2 -D PERL_USEITHREADS
using Apache/2.0.54 (prefork MPM)

waiting 60 seconds for server to start: ...
waiting 60 seconds for server to start: ok (waited 1 secs)
server jib.modperlcookbook.net:8529 started
t/01basic....1..3
ok 1 - no valid password entry
ok 2 - password mismatch
ok 3 - geoff:geoff allowed to proceed
ok
All tests successful.
Files=1, Tests=3,  1 wallclock secs ( 1.12 cusr +  0.12 csys =  1.24 CP
U)
[warning] server jib.modperlcookbook.net:8529 shutdown
```

Client-Side Tests

- The test file acts as a web browser
- The client is scripted to
  - query the server
  - compare server response to expected results
  - indicate success or failure
- In short, the granularity rests in the client
  - that's where all the calls to `ok()` are

Server-Side Tests

- `Apache-Test` provides a mechanism for achieving granularity in the Apache runtime
- Highly magical (not really)
- Let's see it in action...

Apache::SSLLookup

- mod_ssl exposes a few optional functions in C
  - `is_https                 # true if https://`
  - `ssl_var_lookup           # SSL_CLIENT_VERIFY`
- `Apache::SSLLookup` provides Perl glue
  - `Apache::SSLLookup->new()`
  - `is_https()`
  - `ssl_lookup()`

What to Test?

- Class
  - compiles
- Constructor
  - defined
  - returns an object of the proper class
  - returns an object with proper attributes
- Methods
  - defined
  - do something useful

Where To Test

- Clearly this API needs a real Apache + mod_perl runtime

```
$ perl -MApache::SSLLookup -e0

Can't load 'Apache/SSLLookup/SSLLookup.so'
  for module Apache::SSLLookup:  undefined
  symbol: MP_debug_level
```

- `Apache-Test++`
- But even here we have choices

Options

- Client-side test
  - do a bunch of stuff on the server and simply return
    - `OK` if it went OK
    - `500` if it didn't
  - testing in aggregate
- Server-side test
  - much more granular
  - each test can individually pass or fail
- It's all about where you call `ok()`

```
sub handler {

  my $r = shift;

  plan $r, tests => 2;

  {
    use_ok('Apache::SSLLookup');
  }

  {
    can_ok('Apache::SSLLookup', 'new');
  }

  return Apache2::Const::OK
}
```

```
package TestSSL::01new;

use Apache::Test qw(-withtestmore);

use Apache::Const -compile => qw(OK);

sub handler {

  my $r = shift;

  plan $r, tests => 2;

  {
    use_ok('Apache::SSLLookup');
  }

  {
    can_ok('Apache::SSLLookup', 'new');
  }

  return Apache2::Const::OK
}
1;
```

A Brief Digression…

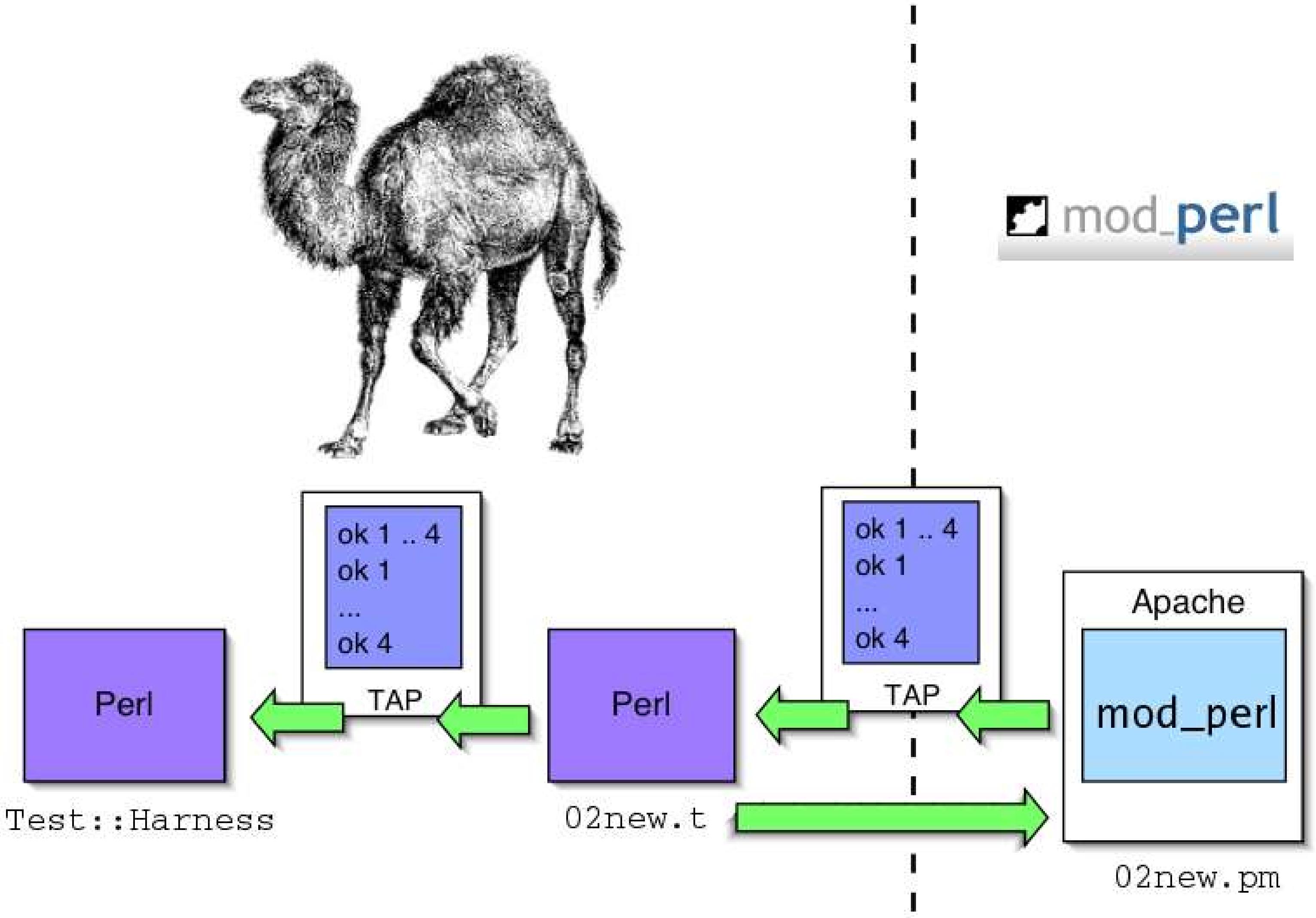
- TAP – the Test Anything Protocol
  - aka

```
1..2
ok 1
# this is a comment
not ok 2
```

- Documented in `Test::Harness::TAP`

Marketing++

- Once TAP was properly branded things started happening
- There are now TAP implementations in
  - PHP (`test-more.php`)
  - C (`libtap`)
  - JavaScript (`TestSimple.js`)
- Once you can generate TAP all you need to do is feed it to `Test::Harness`


mod_perl
ok 1 .. 4
ok 1
...
ok 4
TAP
ok 1 .. 4
ok 1
...
ok 4
TAP
Perl
Perl
Apache
mod_perl
Test::Harness
02new.t
02new.pm

Where's Your TPS Report?

- Other names for TAP that didn't make the cut...
  - TPS
  - Universal Testing Format, Version 1 (UTF-1)
  - Simple Test Output Protocol, Perl (STOP-Perl)
  - Simple Test Reporting Engine Protocol (STREP)
  - probe
  - Practical Output Of Pass/fails (POOP)
  - A Regular System Emitting Bulleted Ancillary Normative Declarations of Idempotent Tests
- See Andy's `use.perl` journal for more

Magic

- Magical things happen if you follow a specific filesystem pattern
- In our case

```
t/response/TestSSL/02new.pm
```

automagically generates

```
t/ssl/02new.t
```

and an entry in

```
t/conf/httpd.conf
```

t/ssl/02new.t

```
# WARNING: this file is generated, do not edit
# 01: Apache/TestConfig.pm:898
# 02: /Apache/TestConfig.pm:916
# 03: Apache/TestConfigPerl.pm:138
# 04: Apache/TestConfigPerl.pm:553
# 05: Apache/TestConfig.pm:584
# 06: Apache/TestConfig.pm:599
# 07: Apache/TestConfig.pm:1536
# 08: Apache/TestRun.pm:501
# 09: Apache/TestRunPerl.pm:80
# 10: Apache/TestRun.pm:720
# 11: Apache/TestRun.pm:720
# 12: t/TEST:28

use Apache::TestRequest 'GET_BODY_ASSERT';
print GET_BODY_ASSERT "/TestSSL__02new";
```

t/conf/httpd.conf

```
<Location /TestSSL__02new>
    SetHandler modperl
    PerlResponseHandler TestSSL::02new
</Location>
```

```
sub handler {

  plan shift, tests => 7;

  use_ok('Apache::SSLLookup');

  can_ok('Apache::SSLLookup', 'new');

  eval { my $r = Apache::SSLLookup->new() };

  like ($@, qr/Usage:/,  'new() requires arguments');

  eval { my $r = Apache::SSLLookup->new({}) };

  like ($@, qr/invoked by a 'unknown' object with no 'r' key/,
        'new() requires an object');

  eval { my $r = Apache::SSLLookup->new(bless {}, 'foo') };

  like ($@, qr/invoked by a 'foo' object with no 'r' key/,
        'new() requires an Apache2::RequestRec object');

  my $r = Apache::SSLLookup->new($r);

  isa_ok($r, 'Apache::SSLLookup');
  isa_ok($r, 'Apache2::RequestRec');

  return Apache2::Const::OK;
}
```

More Magic

- Take this code

```
package My::NeedsToBePreLoaded;

BEGIN {
  Apache->push_handlers(
     PerlChildInitHandler => sub { ... }
  );
}
```

- We need `PerlModule` or the `PerlChildInitHandler` won't run for our tests
  - or `startup.pl`, of course

The Solution

```
package TestMy::NeedsToBePreloaded::01compile;

use Apache::Test qw(-withtestmore);

sub handler {

  plan shift, tests => 1;

  use_ok (My::NeedsToBePreloaded);

  return 0;
}
1;

__DATA__
# make sure that the PerlChildInitHandler
# invokes the BEGIN block
PerlModule My::NeedsToBePreloaded
SetEnv PRELOAD test
```

t/conf/httpd.conf

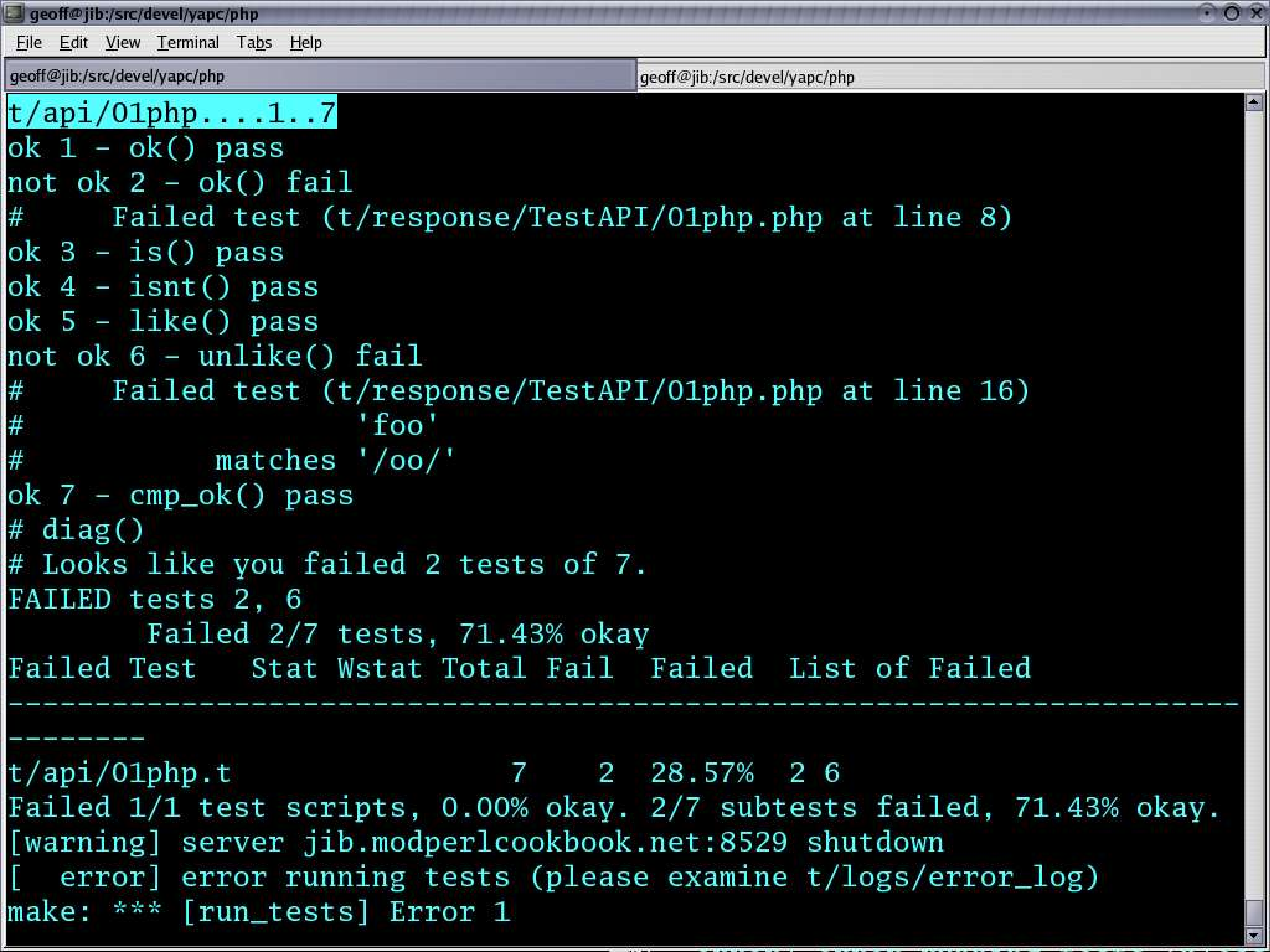
```
# make sure that the PerlChildInitHandler
# invokes the BEGIN block
PerlModule My::NeedsToBePreloaded

<Location /TestMy__NeedsToBePreloaded__01compile>
    SetEnv PRELOAD test
    SetHandler perl-script
    PerlHandler TestMy::NeedsToBePreloaded::01compile
</Location>
```

EOP

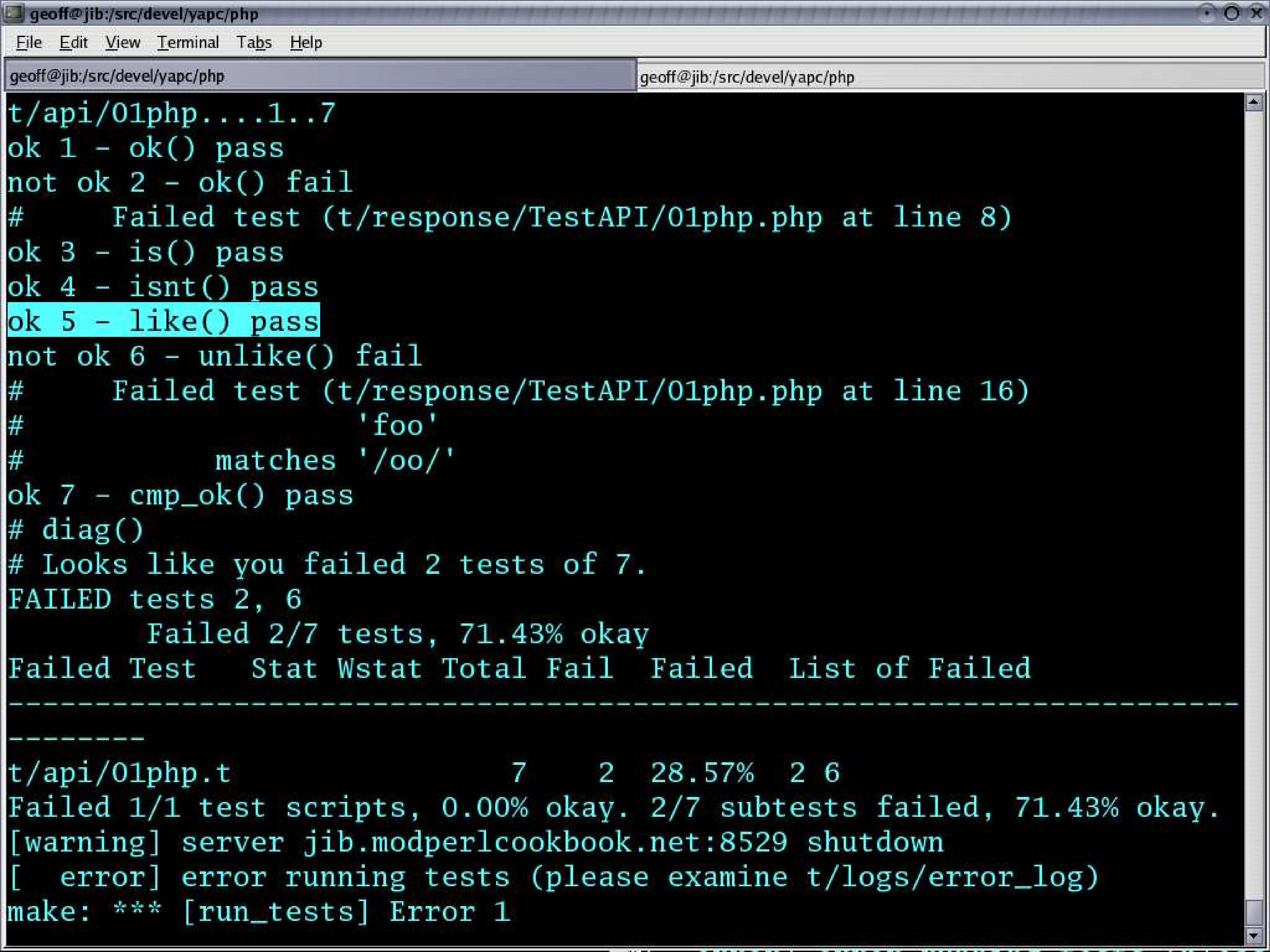
- `Apache-Test` is an Equal Opportunity Platform
- You can now use `Apache-Test` to test PHP too!
  - server-side
  - client-side
- `make test` runs PHP!

t/response/TestAPI/01php.php

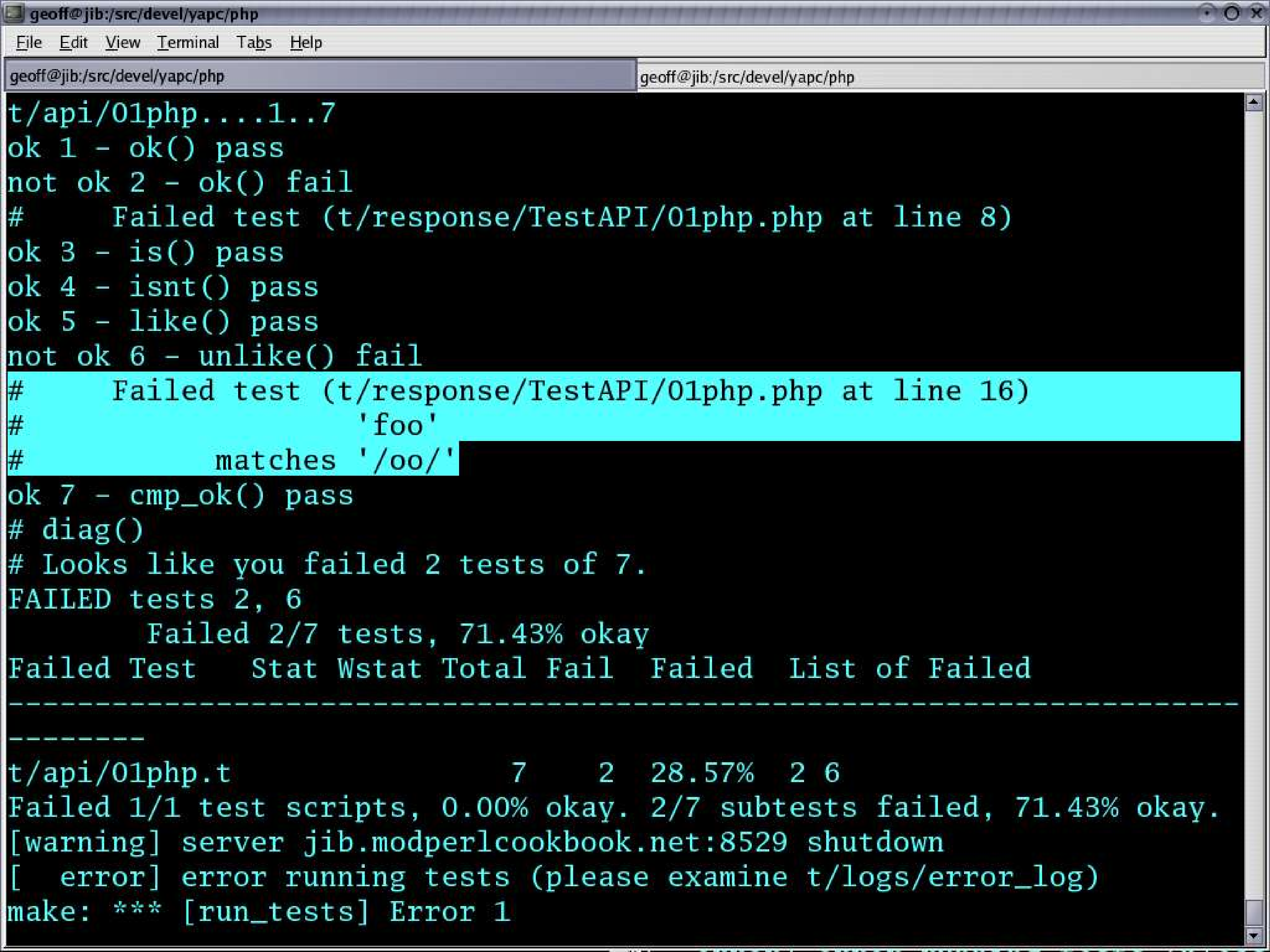
```
<?php

  include 'test-more.php';

  plan(7);

  ok(true, 'ok() pass');
  ok(false, 'ok() fail');

  is('foo', 'foo', 'is() pass');

  isnt('foo', 'bar', 'isnt() pass');

  like('foo', '/oo/', 'like() pass');

  unlike('foo', '/oo/', 'unlike() fail');

  diag('diag()');
?>
```


geoff@jib:/src/devel/yapc/php
File Edit View Terminal Tabs Help
geoff@jib:/src/devel/yapc/php
geoff@jib:/src/devel/yapc/php

```
t/api/01php....1..7
ok 1 - ok() pass
not ok 2 - ok() fail
#     Failed test (t/response/TestAPI/01php.php at line 8)
ok 3 - is() pass
ok 4 - isnt() pass
ok 5 - like() pass
not ok 6 - unlike() fail
#     Failed test (t/response/TestAPI/01php.php at line 16)
#                   'foo'
#           matches '/oo/'
ok 7 - cmp_ok() pass
# diag()
# Looks like you failed 2 tests of 7.
FAILED tests 2, 6
        Failed 2/7 tests, 71.43% okay
Failed Test    Stat Wstat Total Fail  Failed  List of Failed
-----------------------------------------------------------------------
--------
t/api/01php.t                  7    2  28.57%  2 6
Failed 1/1 test scripts, 0.00% okay. 2/7 subtests failed, 71.43% okay.
[warning] server jib.modperlcookbook.net:8529 shutdown
[  error] error running tests (please examine t/logs/error_log)
make: *** [run_tests] Error 1
```


geoff@jib:/src/devel/yapc/php
File Edit View Terminal Tabs Help
geoff@jib:/src/devel/yapc/php
geoff@jib:/src/devel/yapc/php

```
t/api/01php....1..7
ok 1 - ok() pass
not ok 2 - ok() fail
#     Failed test (t/response/TestAPI/01php.php at line 8)
ok 3 - is() pass
ok 4 - isnt() pass
ok 5 - like() pass
not ok 6 - unlike() fail
#     Failed test (t/response/TestAPI/01php.php at line 16)
#                   'foo'
#           matches '/oo/'
ok 7 - cmp_ok() pass
# diag()
# Looks like you failed 2 tests of 7.
FAILED tests 2, 6
        Failed 2/7 tests, 71.43% okay
Failed Test   Stat Wstat Total Fail  Failed  List of Failed
-----------------------------------------------------------------------
--------
t/api/01php.t                7    2  28.57%  2 6
Failed 1/1 test scripts, 0.00% okay. 2/7 subtests failed, 71.43% okay.
[warning] server jib.modperlcookbook.net:8529 shutdown
[  error] error running tests (please examine t/logs/error_log)
make: *** [run_tests] Error 1
```


geoff@jib:/src/devel/yapc/php
File Edit View Terminal Tabs Help
geoff@jib:/src/devel/yapc/php
geoff@jib:/src/devel/yapc/php

```
t/api/01php....1..7
ok 1 - ok() pass
not ok 2 - ok() fail
#     Failed test (t/response/TestAPI/01php.php at line 8)
ok 3 - is() pass
ok 4 - isnt() pass
ok 5 - like() pass
not ok 6 - unlike() fail
#     Failed test (t/response/TestAPI/01php.php at line 16)
#                   'foo'
#           matches '/oo/'
ok 7 - cmp_ok() pass
# diag()
# Looks like you failed 2 tests of 7.
FAILED tests 2, 6
        Failed 2/7 tests, 71.43% okay
Failed Test   Stat Wstat Total Fail  Failed  List of Failed
------------------------------------------------------------------------
--------
t/api/01php.t                 7    2  28.57%  2 6
Failed 1/1 test scripts, 0.00% okay. 2/7 subtests failed, 71.43% okay.
[warning] server jib.modperlcookbook.net:8529 shutdown
[  error] error running tests (please examine t/logs/error_log)
make: *** [run_tests] Error 1
```

The Server Side

- mod_perl and PHP are just examples
- As long as you can get TAP from the server to `Test::Harness` you're golden
- Automagic client generation is helpful but not required
  - write a CGI script that calls `plan()` and `ok()`
  - `GET` the script from `t/foo.t`
  - `print` CGI script response to `STDOUT`
- `Apache-Test` takes care of the hard parts

Slides

- These slides freely available at some long URL you will never remember…

`http://www.modperlcookbook.org/~geoff/slides/YAPC`

- Linked to from my homepage

`http://www.modperlcookbook.org/~geoff/`