Writing Tests with Apache-Test

Geoffrey Young
geoff@modperlcookbook.org

Apache-Test

- Framework for testing Apache-based application components
- Provides tools to make testing Apache related things simple
  - configures, starts, and stops Apache
  - lets you focus on writing tests
  - provides HTTP-centric tools
- Gives you a self-contained, pristine Apache environment for testing

Apache-Test by Example

- Write a simple Perl handler
- Integrate Apache-Test
- Show all kinds of cool stuff

```
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;
```

Voila!


Prompt
Enter username and password for "cookbook" at jib
User Name:
Password:
Use Password Manager to remember these values.
OK
Cancel

Testing, Testing... 1, 2, 3

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

Step 1 - The Test Harness

- Generally starts from `Makefile.PL`
- There are other ways as well

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();
```

t/TEST

- `generate_script()` creates the special file `t/TEST`
- `t/TEST` is the actual harness that coordinates testing activities
- called via `make test`
- can be called directly

```
$ t/TEST t/foo.t
```

Step 1 - The Test Harness

- Don't get bogged down with `Makefile.PL` details
- Lather, Rinse, Repeat

Testing, Testing... 1, 2, 3

1. Generate the test harness
2. Configure Apache

Step 2 - Configure Apache

- 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
```

Adding to the Default Config

- 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`

extra.conf.in

- Same directives as `httpd.conf`
- Pulled into `httpd.conf` via `Include`
- Provides handy variable substitution

Create the Configuration

- Our handler is an authentication handler

- Let's set up a protected URI using `t/conf/extra.conf.in`

extra.conf.in

```
Alias /authen @DocumentRoot@

<Location /authen>
  Require valid-user
  AuthType Basic
  AuthName "my test realm"

  PerlAuthenHandler My::AuthenHandler
</Location>
```

Whew!

- At this point we have...
  - autogenerated a minimal Apache configuration
  - configured a protected URL
  - created a mod_perl handler to do the authentication
- Now...

Testing, Testing... 1, 2, 3

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

What Should We Test?

- Tests need to be meaningful
- Tests ought to be thorough
- Tests should align the server response with our expected results

In Other Words...

- Our handler is an authentication handler

- Unknown users should fail
- Known users should pass

What Exactly is a Test?

- Tests are contained within a test file
- The test file acts as a client
- The client is scripted to
  - query the server
  - compare server response to expected results
  - indicate success or failure

The t/ Directory

- Tests live in t/
  - t/01basic.t
- t/ is the ServerRoot
  - t/htdocs
  - t/cgi-bin
  - t/conf

Anatomy of a Test

- `Apache-Test` works the same way as `Test.pm`, `Test::More` and others
- `plan()` the number of tests
- call `ok()` for each test you plan
  - where `ok()` is any one of a number of comparison functions
- All the rest is up to you

t/01basic.t

```
use Apache::Test;
use Apache::TestRequest;

plan tests => 1, (need_lwp &&
                  need_auth &&
                  need_module('mod_perl.c'));
```

Apache::Test

- Provides basic `Test.pm` functions
  - `ok()`
  - `plan()`
- Also provides helpful `plan()` functions
  - `need_lwp()`
  - `need_module()`
  - `need_min_apache_version()`

plan()

- plan() the number of tests in the file

```
plan tests => 5;
```

- Preconditions can be specified

```
plan tests => 5, need_module('mod_foo');
```

- Failed preconditions will skip the entire test file

```
server localhost.localdomain:8529 started
t/01basic....skipped
        all skipped: cannot find module 'mod_foo.c'
All tests successful, 1 test skipped.
```

t/01basic.t

```
use Apache::Test;
use Apache::TestRequest;

plan tests => 1, (need_lwp &&
                  need_auth &&
                  need_module('mod_perl.c'));

{
  my $uri = '/authen/index.html';

  my $response = GET $uri;
  ok $response->code == 401;
}
```

Apache::TestRequest

- Provides a basic LWP interface
  - GET()
  - POST()
  - HEAD()
  - GET_OK()
  - GET_BODY()
  - more
- Note that these functions know which host and port to send the request to

```
GET '/authen/index.html';
```

  - request URI can be relative

HTTP::Response

- LWP base class
- Provides accessors to response attributes
  - code()
  - content()
  - content_type(), content_length(), etc
  - headers()
    - authorization()
- as well as some useful utility methods
  - as_string()
  - previous()

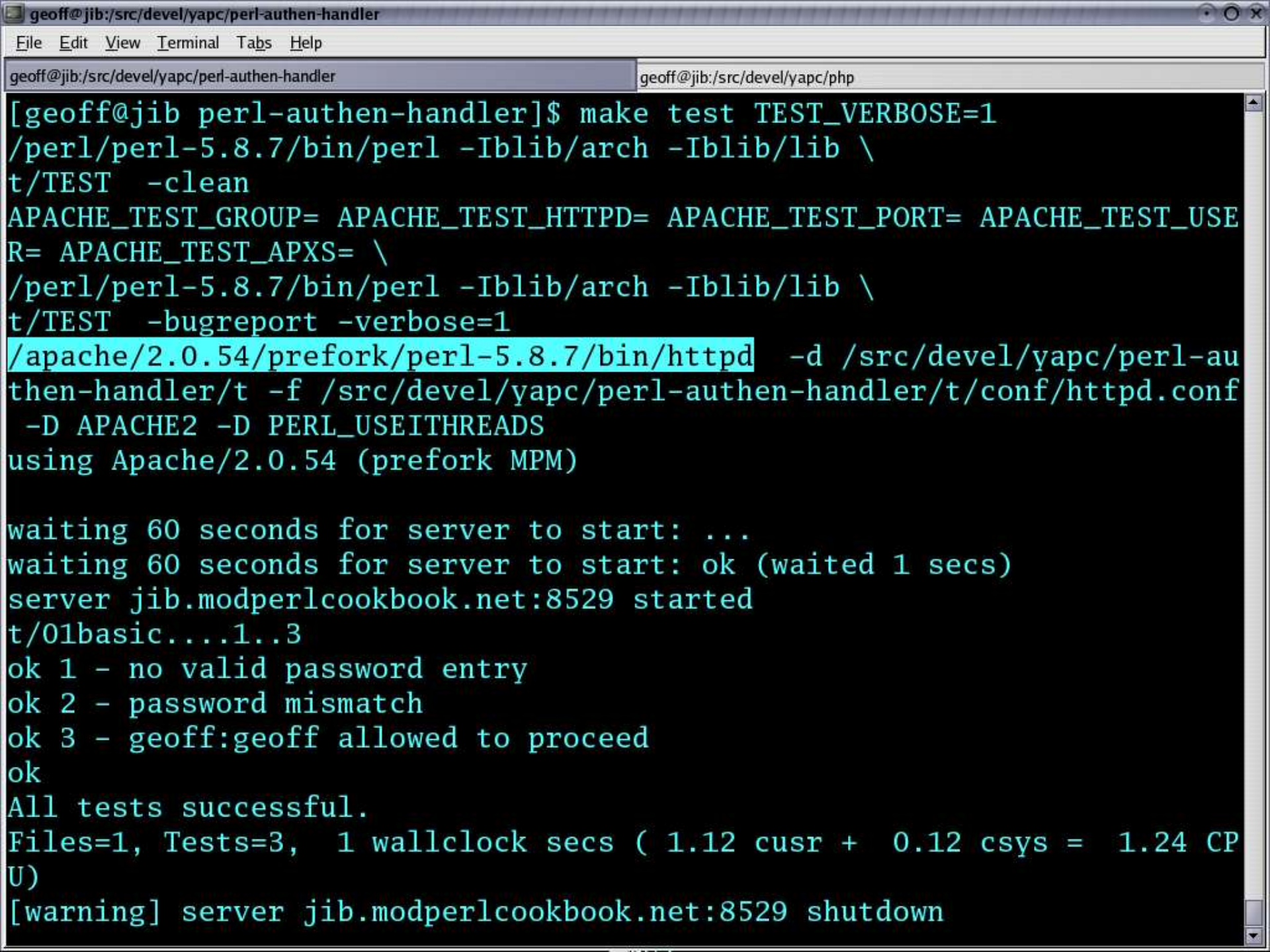
Testing, Testing... 1, 2, 3

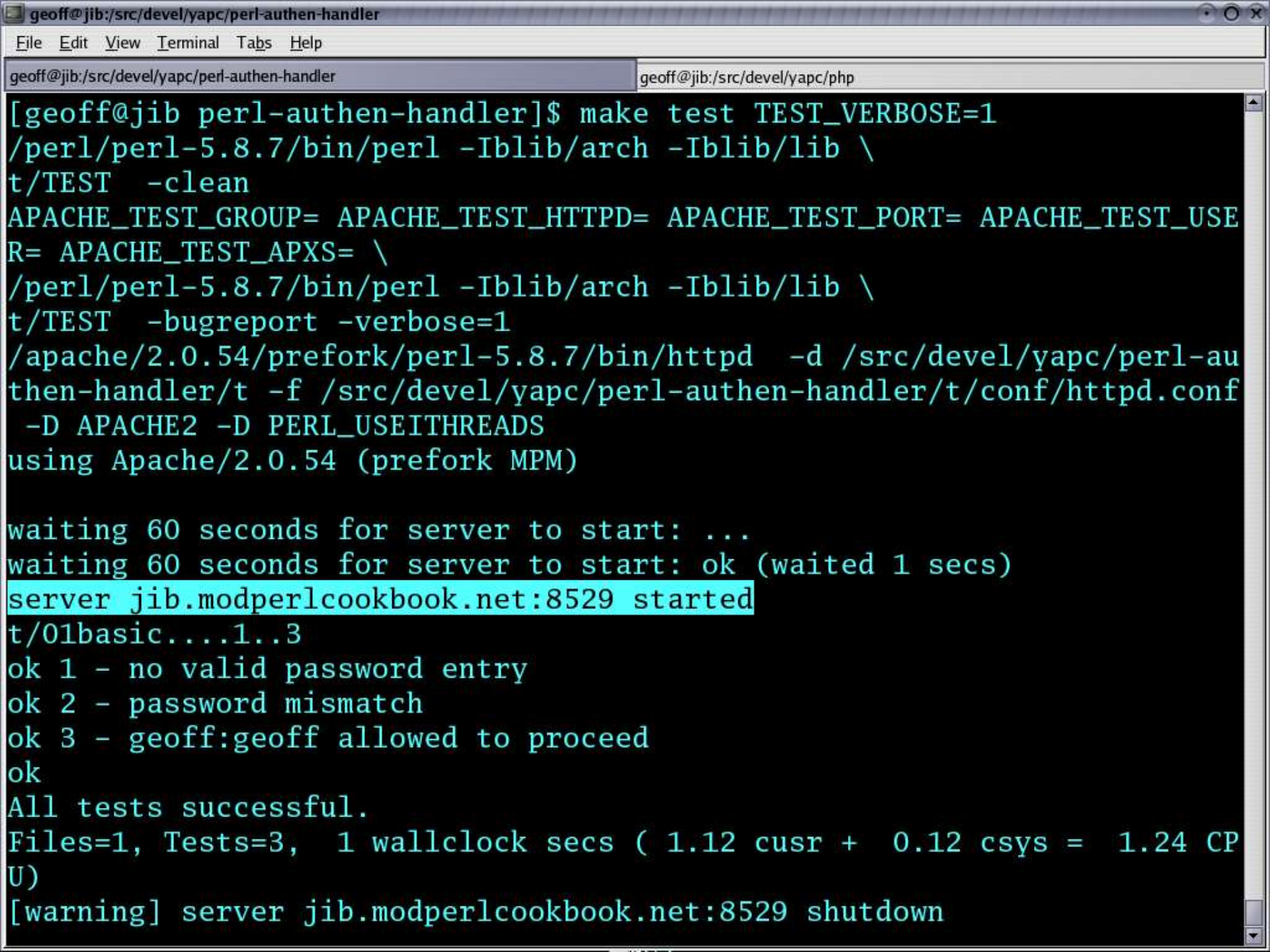
1. Generate the test harness
2. Configure Apache
3. Write the tests
4. Run the tests

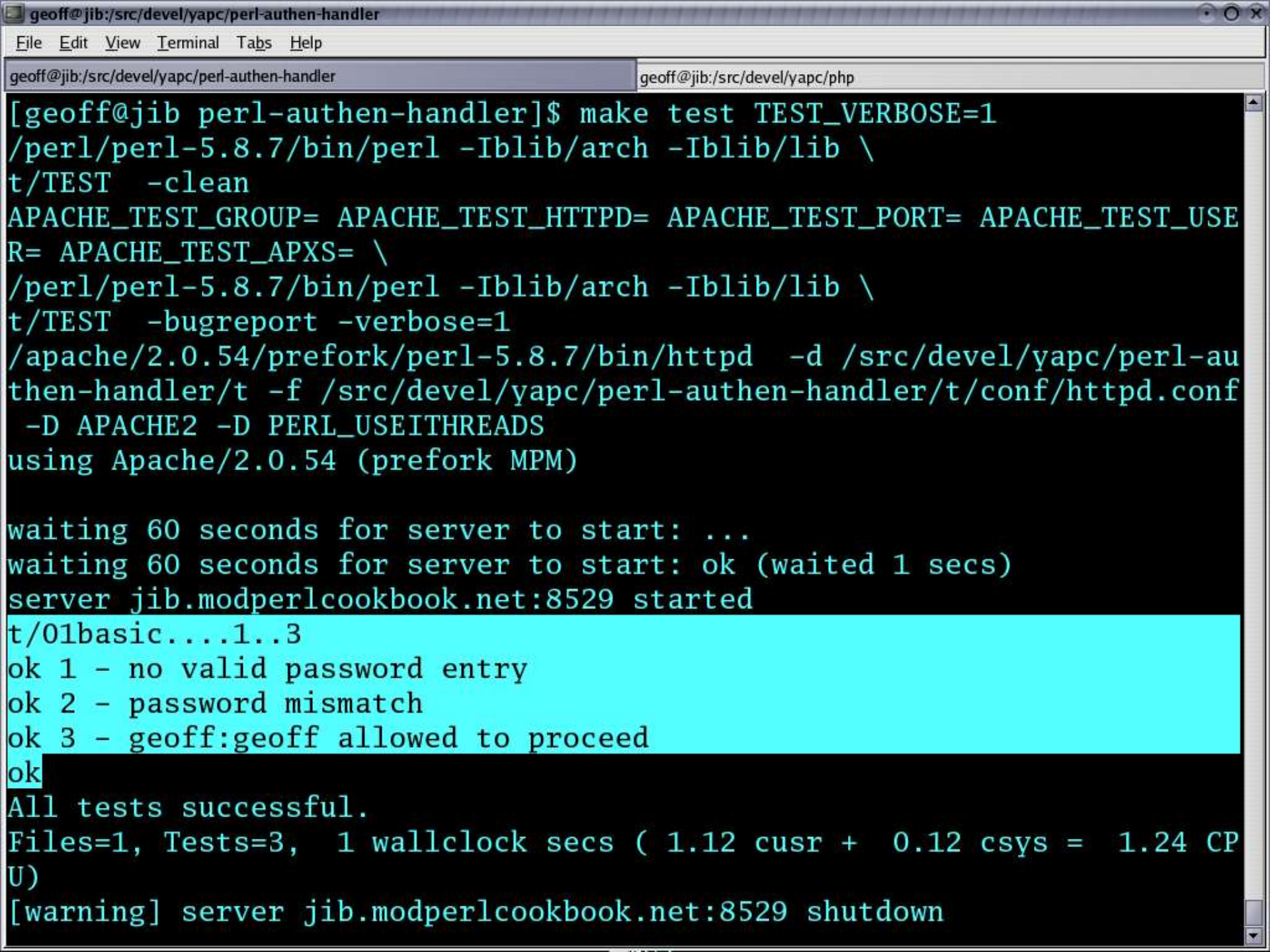
The Pain is Over

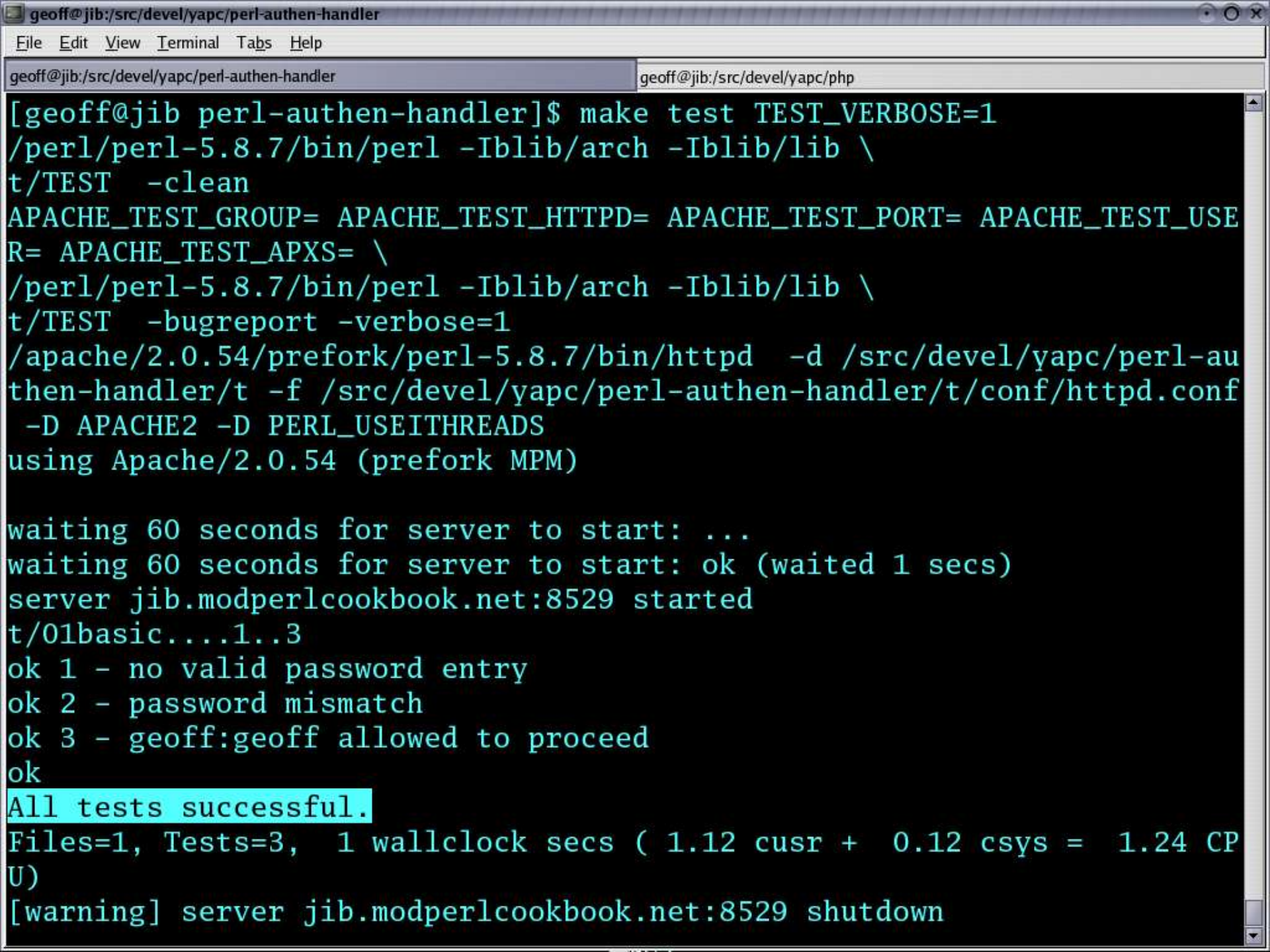
- Initial setup is a bit complex
  - you only do it once
- Running the tests is simple

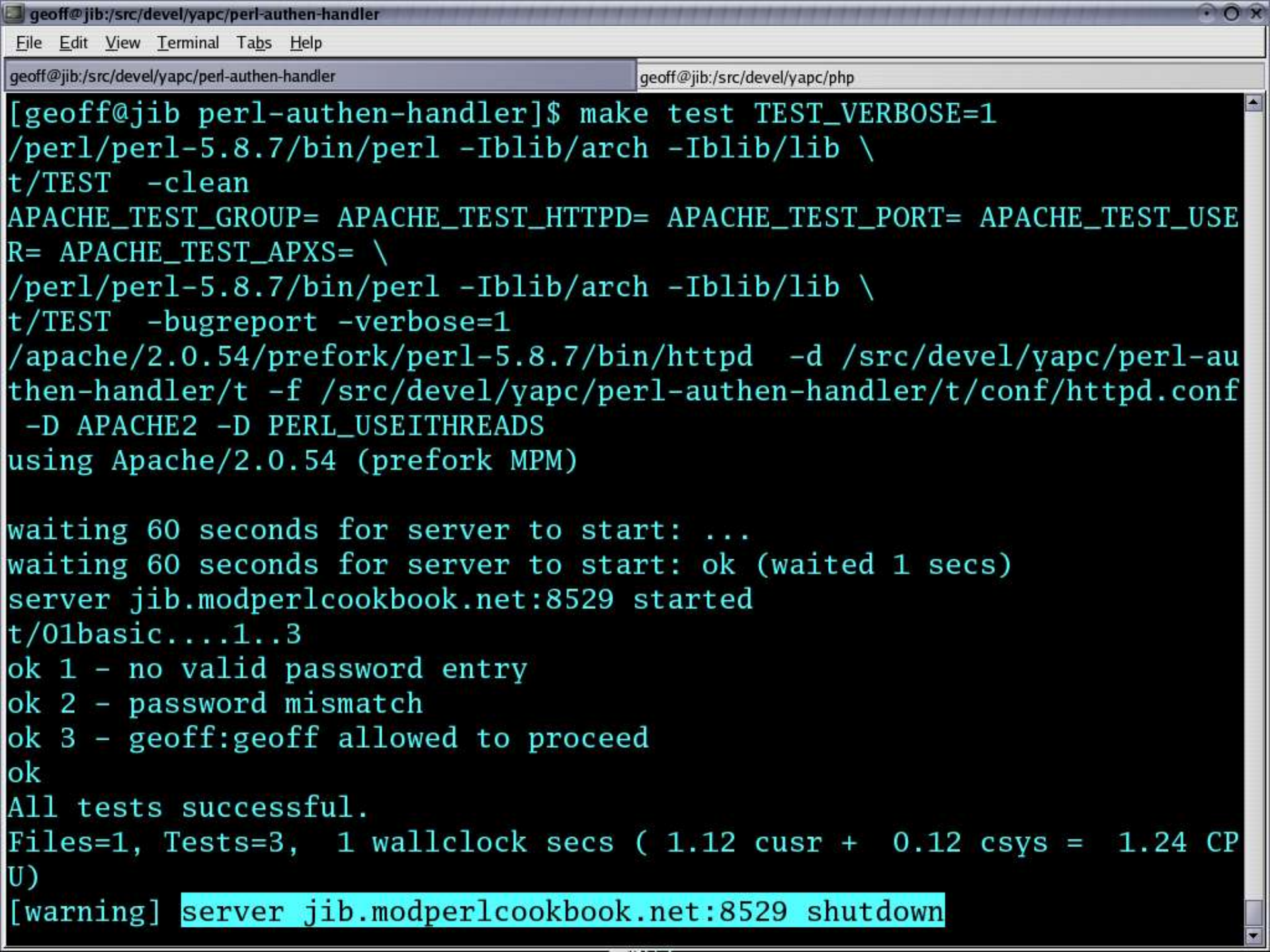
```
$make test
$t/TEST
```


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

Are you ok?

- `ok()` works, but is not descriptive
- luckily, we have options
  - `Apache::TestUtil`
  - `Test::More`

Apache::TestUtil

- Chocked full of helpful utilities
- `t_cmp()`

```
t_cmp($foo, $bar, 'foo is bar');
t_cmp($foo, qr/bar/, 'foo matches bar');
```

- `t_write_file($file, @lines);`
  - write out a file
  - clean it up after script execution completes
- `t_write_perl_script($file, @lines);`
  - same as `t_write_file()`
  - with compilation-specific shebang line
  - useful for writing out CGI scripts

```
use Apache::Test;
use Apache::TestRequest;


plan tests => 1, (need_lwp &&
                  need_auth &&
                  need_module('mod_perl.c'));

{
  my $uri = '/authen/index.html';

  my $response = GET $uri;

  ok $response->code == 401;
}
```

But Wait, There's More!

- Test::More is the Perl testing standard
- Goes beyond Apache-Test's t_cmp()

Test::More functions

- Basic comparisons
  - ok()
  - is()
  - like()
- Intuitive comparisons
  - isnt()
  - unlike()
- Complex structures
  - is_deeply()
  - eq_array()

```
use Apache::Test qw(-withtestmore);
use Apache::TestRequest;


plan tests => 1, (need_lwp &&
                  need_auth &&
                  need_module('mod_perl.c'));

{
  my $uri = '/authen/index.html';

  my $response = GET $uri;

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

```
server localhost.localdomain:8529 started
t/authen03....1..1
ok 1 - no valid password entry
ok
All tests successful.


server localhost.localdomain:8529 started
t/authen03....1..1
not ok 1 - no valid password entry

#     Failed test (t/authen03.t at line 18)
#          got: '200'
#     expected: '401'
# Looks like you failed 1 test of 1.
```

Using Test::More

- Test::More is the Perl-world standard
- You should use Test::More whenever possible
- Eventually -withtestmore will not be required

Getting Back to the Point...

- So far, we haven't actually tested anything useful
  - no username or password
- Let's add some real tests

```
my $uri = '/authen/index.html';

{
  my $response = GET $uri;

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

{
  my $response = GET $uri, username => 'geoff', password => 'foo';

  is ($response->code,
      401,
      "password mismatch");
}

{
  my $response = GET $uri, username => 'geoff', password => 'geoff';

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

Take Advantage of LWP

```
# check cache-friendliness
$last_modified = $res->header('Last-Modified');
$res = GET $uri, 'If-Modified-Since' => $last_modified;

ok $res->code == 304;
```

It's not just for mod_perl

- `Apache-Test` was designed with the mod_perl API in mind
- Generic enough to test components of any Apache-centric environment

SOAP::Lite-based API

```
use SOAP::Lite
  +autodispatch =>
  uri   => 'http://intranet/Foo/Email/SOAP',
  proxy => 'http://intranet/Email';

my $rc = add_user('foo', 'foo.com', 'Pete', 'bar');
```

extra.conf.in

```
<Location /Email>
  SetHandler perl-script
  PerlHandler Apache::SOAP

  PerlSetVar dispatch_to "Foo::Email::SOAP"
</Location>
```

Perl

```
use Apache::Test;

plan tests => 1, need_module('Apache::SOAP');

use SOAP::Lite
  +autodispatch =>
  uri   => 'http://localhost:8529/Foo/Email/SOAP',
  proxy => 'http://localhost:8529/Email';

my $rc = add_user('foo', 'foo.com', 'Pete', 'bar');
```

Non-Perl

```
use Apache::Test;

plan tests => 1, need_module('Apache::SOAP');

my $rc = qx!java email_soap_client
            'http://localhost:8529/' Email
            add_user foo foo.com Pete bar!;
```

But Wait, There's More!

- Letting `Apache-Test` run *all* your tests
- Automagic configuration via `__DATA__`
- Integrated Apache C-module support
- Server-side tests