```
package Test::Pod::Coverage;

=head1 NAME

Test::Pod::Coverage - Check for pod coverage in your distribution.

=head1 VERSION

Version 1.08

=cut

our $VERSION = "1.08";

=head1 SYNOPSIS

Checks for POD coverage in files for your distribution.

    use Test::Pod::Coverage tests=>1;
    pod_coverage_ok( "Foo::Bar", "Foo::Bar is covered" );

Can also be called with L<Pod::Coverage> parms.

    use Test::Pod::Coverage tests=>1;
    pod_coverage_ok(
        "Foo::Bar",
        { also_private => [ qr/^[A-Z_]+$/ ], },
        "Foo::Bar, with all-caps functions as privates",
    );

The L<Pod::Coverage> parms are also useful for subclasses that don't
re-document the parent class's methods.  Here's an example from
L<Mail::SRS>.

    pod_coverage_ok( "Mail::SRS" ); # No exceptions

    # Define the three overridden methods.
    my $trustme = { trustme => [qr/^(new|parse|compile)$/] };
    pod_coverage_ok( "Mail::SRS::DB", $trustme );
    pod_coverage_ok( "Mail::SRS::Guarded", $trustme );
    pod_coverage_ok( "Mail::SRS::Reversable", $trustme );
    pod_coverage_ok( "Mail::SRS::Shortcut", $trustme );

Alternately, you could use L<Pod::Coverage::CountParents>, which always allows
a subclass to reimplement its parents' methods without redocumenting them.  For
example:

    my $trustparents = { coverage_class => 'Pod::Coverage::CountParents' };
    pod_coverage_ok( "IO::Handle::Frayed", $trustparents );

(The C<coverage_class> parameter is not passed to the coverage class with other
parameters.)

If you want POD coverage for your module, but don't want to make
Test::Pod::Coverage a prerequisite for installing, create the following
as your F<t/pod-coverage.t> file:

    use Test::More;
    eval "use Test::Pod::Coverage";
    plan skip_all => "Test::Pod::Coverage required for testing pod coverage" if $@;

    plan tests => 1;
    pod_coverage_ok( "Pod::Master::Html");

Finally, Module authors can include the following in a F<t/pod-coverage.t>
file and have C<Test::Pod::Coverage> automatically find and check all
```

```
modules in the module distribution:

    use Test::More;
    eval "use Test::Pod::Coverage 1.00";
    plan skip_all => "Test::Pod::Coverage 1.00 required for testing POD coverage" i
f $@;
    all_pod_coverage_ok();

=cut

use strict;
use warnings;

use Pod::Coverage;
use Test::Builder;

my $Test = Test::Builder->new;

sub import {
    my $self = shift;
    my $caller = caller;
    no strict 'refs';
    *{$caller.'::pod_coverage_ok'}       = \&pod_coverage_ok;
    *{$caller.'::all_pod_coverage_ok'}   = \&all_pod_coverage_ok;
    *{$caller.'::all_modules'}           = \&all_modules;

    $Test->exported_to($caller);
    $Test->plan(@_);
}

=head1 FUNCTIONS

All functions listed below are exported to the calling namespace.

=head2 all_pod_coverage_ok( [$parms, ] $msg )

Checks that the POD code in all modules in the distro have proper POD
coverage.

If the I<$parms> hashref if passed in, they're passed into the
C<Pod::Coverage> object that the function uses.  Check the
L<Pod::Coverage> manual for what those can be.

The exception is the C<coverage_class> parameter, which specifies a class to
use for coverage testing.  It defaults to C<Pod::Coverage>.

=cut

sub all_pod_coverage_ok {
    my $parms = (@_ && (ref $_[0] eq "HASH")) ? shift : {};
    my $msg = shift;

    my $ok = 1;
    my @modules = all_modules();
    if ( @modules ) {
        $Test->plan( tests => scalar @modules );

        for my $module ( @modules ) {
            my $thismsg = defined $msg ? $msg : "Pod coverage on $module";

            my $thisok = pod_coverage_ok( $module, $parms, $thismsg );
            $ok = 0 unless $thisok;
        }
    }
    else {
        $Test->plan( tests => 1 );
```

```
        $Test->ok( 1, "No modules found." );
    }

    return $ok;
}


=head2 pod_coverage_ok( $module, [$parms, ] $msg )

Checks that the POD code in I<$module> has proper POD coverage.

If the I<$parms> hashref if passed in, they're passed into the
C<Pod::Coverage> object that the function uses.  Check the
L<Pod::Coverage> manual for what those can be.

The exception is the C<coverage_class> parameter, which specifies a class to
use for coverage testing.  It defaults to C<Pod::Coverage>.

=cut

sub pod_coverage_ok {
    my $module = shift;
    my %parms = (@_ && (ref $_[0] eq "HASH")) ? %{(shift)} : ();
    my $msg = @_ ? shift : "Pod coverage on $module";

    my $pc_class = (delete $parms{coverage_class}) || 'Pod::Coverage';
    eval "require $pc_class" or die $@;

    my $pc = $pc_class->new( package => $module, %parms );

    my $rating = $pc->coverage;
    my $ok;
    if ( defined $rating ) {
        $ok = ($rating == 1);
        $Test->ok( $ok, $msg );
        if ( !$ok ) {
            my @nakies = sort $pc->naked;
            my $s = @nakies == 1 ? "" : "s";
            $Test->diag(
                sprintf( "Coverage for %s is %3.1f%%, with %d naked subroutine$s:",
                    $module, $rating*100, scalar @nakies ) );
            $Test->diag( "\t$_" ) for @nakies;
        }
    }
    else { # No symbols
        my $why = $pc->why_unrated;
        my $nopublics = ( $why =~ "no public symbols defined" );
        my $verbose = $ENV{HARNESS_VERBOSE} || 0;
        $ok = $nopublics;
        $Test->ok( $ok, $msg );
        $Test->diag( "$module: $why" ) unless ( $nopublics && !$verbose );
    }

    return $ok;
}

=head2 all_modules( [@dirs] )

Returns a list of all modules in I<$dir> and in directories below. If
no directories are passed, it defaults to F<blib> if F<blib> exists,
or F<lib> if not.

Note that the modules are as "Foo::Bar", not "Foo/Bar.pm".

The order of the files returned is machine-dependent.  If you want them
sorted, you'll have to sort them yourself.
```

```
=cut

sub all_modules {
    my @starters = @_ ? @_ : _starting_points();
    my %starters = map {$_,1} @starters;

    my @queue = @starters;

    my @modules;
    while ( @queue ) {
        my $file = shift @queue;
        if ( -d $file ) {
            local *DH;
            opendir DH, $file or next;
            my @newfiles = readdir DH;
            closedir DH;

            @newfiles = File::Spec->no_upwards( @newfiles );
            @newfiles = grep { $_ ne "CVS" && $_ ne ".svn" } @newfiles;

            push @queue, map "$file/$_", @newfiles;
        }
        if ( -f $file ) {
            next unless $file =~ /\.pm$/;

            my @parts = File::Spec->splitdir( $file );
            shift @parts if @parts && exists $starters{$parts[0]};
            shift @parts if @parts && $parts[0] eq "lib";
            $parts[-1] =~ s/\.pm$// if @parts;

            # Untaint the parts
            for ( @parts ) {
                if ( /^([a-zA-Z0-9_\.\-]+)$/ && ($_ eq $1) ) {
                    $_ = $1;  # Untaint the original
                }
                else {
                    die qq{Invalid and untaintable filename "$file"!};
                }
            }
            my $module = join( "::", @parts );
            push( @modules, $module );
        }
    } # while

    return @modules;
}

sub _starting_points {
    return 'blib' if -e 'blib';
    return 'lib';
}

=head1 BUGS

Please report any bugs or feature requests to
C<bug-test-pod-coverage at rt.cpan.org>, or through the web interface at
L<http://rt.cpan.org/NoAuth/ReportBug.html?Queue=Test-Pod-Coverage>.
I will be notified, and then you'll automatically be notified of progress on
your bug as I make changes.

=head1 SUPPORT

You can find documentation for this module with the perldoc command.

    perldoc Test::Pod::Coverage
```

```
You can also look for information at:

=over 4

=item * AnnoCPAN: Annotated CPAN documentation

L<http://annocpan.org/dist/Test-Pod-Coverage>

=item * CPAN Ratings

L<http://cpanratings.perl.org/d/Test-Pod-Coverage>

=item * RT: CPAN's request tracker

L<http://rt.cpan.org/NoAuth/Bugs.html?Dist=Test-Pod-Coverage>

=item * Search CPAN

L<http://search.cpan.org/dist/Test-Pod-Coverage>

=back

=head1 AUTHOR

Written by Andy Lester, C<< <andy at petdance.com> >>.

=head1 ACKNOWLEDGEMENTS

Thanks to Ricardo Signes for patches, and Richard Clamp for
writing Pod::Coverage.

=head1 COPYRIGHT & LICENSE

Copyright 2006, Andy Lester, All Rights Reserved.

You may use, modify, and distribute this package under the
same terms as Perl itself.

=cut

1;
```