NEW MODULES

App::sh2p

module by Clive Darke, CLIVE@cpan.org

This application provides a shell to Perl script converter, although you shouldn't get too excited too soon: it does its best to make the conversion but might leave some it up to you.

```
sh2pl.pl script.sh > script.pl
```

If you have to convert shell scripts, this can take some of the tedium out of the process.

VideoLAN::Client

module by Cyrille Hombecq, ELLIRYC@cpan.org

You can launch VLC (*http://www.videolan.org/vlc/*) and control it from Perl. Once you enable the telnet interface, you can connect to it with `VideoLan::Client`. Once connected, you can issue commands to it. The help command returns a list of things that you can do:

```
use VideoLan::Client;
my $vlc = VideoLan::Client->new(
  HOST => 'localhost',
  PORT => 4212,
  );

if( eval { $vlc->login } )
  {
  print $vlc->cmd( "help" );

  $vlc->logout;
  }
else
  {
  die "Couldn't log in!\n";
  }
```

Iterator::Diamond

module by Johan Vromans, JV@cpan.org

The `Iterator::Diamond` module is a replacement for the diamond operator, <>, but it doesn't use any magic to process the file names. It stills reads from all of the files you specify on the command line, sets `$ARGV`, and can handle in-place editing. Instead of using a bare <>, get the iterator from the module and use that in the line-input operator:

```
use Iterator::Diamond;

$input = Iterator::Diamond->new;

while ( <$input> ) {
  print "$ARGV: $_";
  }
```

Sub::Delete

module by Father Chrysostomos, SPROUT@cpan.org

If you undefine a subroutine, you don't really delete it. It's still there, messing up things like `AUTOLOAD`. If you really want to get rid of a subroutine, use `Sub::Delete` to do it. Once you delete the subroutine, you can't reference it anymore. Compiled code is just fine though:

```
use Sub::Delete;
sub hello { print "Hello World\n" }
delete_sub 'hello';

#just fine, already compiled
hello();

#not fine, can't compile any more
eval 'hello();1' or die;
```

App::CCSV

module by Karlheinz Zöchling, GARGAMEL@cpan.org

Work with comma-separated value files with this Perl application. It works with `perl`'s `-a` switch, but is smarter about the CSV format. Instead of looking in `@F` for the autosplit values, look in `@f`:

```
% perl -MApp::CCSV -ne 'print @f' < f.csv
```

You can even change the CSV by specify different sets of options for the input and output. Use the `cprint` subroutine to print an array as CSV:

```
% perl -MApp::CCSV=in,out -ne 'cprint @f' <
f.csv
```

Scalar::Util::Refcount

module by Jeremy Nixon, JNIXON@cpan.org

You're not supposed to have to care about the reference count of a reference, but if you can't help peeking, the `Scalar::Util::Refcount` module just gives it to you. You don't have to play with the output from `Devel::Peek` anymore:

```
use Scalar::Util::Refcount;
use 5.010;

my $ua = LWP::UserAgent->new;
say refcount($ua); # says "1"

my $ua2 = $ua;
say refcount($ua2); # says "2"
```