```
no MyFilter;

print "red\n";   # this code is not filtered, will print "red\n"
```

- `File::Temp`, by Tim Jenness, allows one to create temporary files and directories in an easy, portable, and secure way. See *File::Temp*. [561+]
- `Filter::Util::Call`, by Paul Marquess, provides you with the framework to write *source filters* in Perl. For most uses, the frontend Filter::Simple is to be preferred. See *Filter::Util::Call*.
- `if`, by Ilya Zakharevich, is a new pragma for conditional inclusion of modules.
- *libnet*, by Graham Barr, is a collection of perl5 modules related to network programming. See *Net::FTP*, *Net::NNTP*, *Net::Ping* (not part of libnet, but related), *Net::POP3*, *Net::SMTP*, and *Net::Time*.

  Perl installation leaves libnet unconfigured; use *libnetcfg* to configure it.
- `List::Util`, by Graham Barr, is a selection of general-utility list subroutines, such as sum(), min(), first(), and shuffle(). See *List::Util*.
- `Locale::Constants`, `Locale::Country`, `Locale::Currency` `Locale::Language`, and *Locale::Script*, by Neil Bowers, have been added. They provide the codes for various locale standards, such as "fr" for France, "usd" for US Dollar, and "ja" for Japanese.

  ```
  use Locale::Country;

  $country = code2country('jp');               # $country gets 'Japan'
  $code    = country2code('Norway');           # $code gets 'no'
  ```

  See *Locale::Constants*, *Locale::Country*, *Locale::Currency*, and *Locale::Language*.
- `Locale::Maketext`, by Sean Burke, is a localization framework. See *Locale::Maketext*, and *Locale::Maketext::TPJ13*. The latter is an article about software localization, originally published in The Perl Journal #13, and republished here with kind permission.
- `Math::BigRat` for big rational numbers, to accompany Math::BigInt and Math::BigFloat, from Tels. See *Math::BigRat*.
- `Memoize` can make your functions faster by trading space for time, from Mark-Jason Dominus. See *Memoize*.
- `MIME::Base64`, by Gisle Aas, allows you to encode data in base64, as defined in RFC 2045 - *MIME (Multipurpose Internet Mail Extensions)*.

  ```
  use MIME::Base64;

  $encoded = encode_base64('Aladdin:open sesame');
  $decoded = decode_base64($encoded);

  print $encoded, "\n"; # "QWxhZGRpbjpvcGVuIHNlc2FtZQ=="
  ```

  See *MIME::Base64*.
- `MIME::QuotedPrint`, by Gisle Aas, allows you to encode data in quoted-printable encoding, as defined in RFC 2045 - *MIME (Multipurpose Internet Mail Extensions)*.

  ```
  use MIME::QuotedPrint;

  $encoded = encode_qp("\xDE\xAD\xBE\xEF");
  $decoded = decode_qp($encoded);
  ```