91.3.48 Improved security features

More potentially unsafe operations taint their results for improved security.

The `passwd` and `shell` fields returned by the getpwent(), getpwnam(), and getpwuid() are now tainted, because the user can affect their own encrypted password and login shell.

The variable modified by shmread(), and messages returned by msgrcv() (and its object-oriented interface IPC::SysV::Msg::rcv) are also tainted, because other untrusted processes can modify messages and shared memory segments for their own nefarious purposes.

91.3.49 More functional bareword prototype (*)

Bareword prototypes have been rationalized to enable them to be used to override builtins that accept barewords and interpret them in a special way, such as `require` or `do`.

Arguments prototyped as * will now be visible within the subroutine as either a simple scalar or as a reference to a typeglob. See Prototypes in *perlsub*.

91.3.50 `require` and do may be overridden

`require` and `do 'file'` operations may be overridden locally by importing subroutines of the same name into the current package (or globally by importing them into the CORE::GLOBAL:: namespace). Overriding `require` will also affect `use`, provided the override is visible at compile-time. See Overriding Built-in Functions in *perlsub*.

91.3.51 $ ˆX variables may now have names longer than one character

Formerly, $ˆX was synonymous with ${"\cX"}, but $ˆXY was a syntax error. Now variable names that begin with a control character may be arbitrarily long. However, for compatibility reasons, these variables *must* be written with explicit braces, as `${^XY}` for example. `${^XYZ}` is synonymous with ${"\cXYZ"}. Variable names with more than one control character, such as `${^XY^Z}`, are illegal.

The old syntax has not changed. As before, 'ˆX' may be either a literal control-X character or the two-character sequence 'caret' plus 'X'. When braces are omitted, the variable name stops after the control character. Thus `"$^XYZ"` continues to be synonymous with `$^X . "YZ"` as before.

As before, lexical variables may not have names beginning with control characters. As before, variables whose names begin with a control character are always forced to be in package 'main'. All such variables are reserved for future extensions, except those that begin with `^_`, which may be used by user programs and are guaranteed not to acquire special meaning in any future version of Perl.

91.3.52 New variable $ ˆC reflects `-c` switch

`$^C` has a boolean value that reflects whether perl is being run in compile-only mode (i.e. via the `-c` switch). Since BEGIN blocks are executed under such conditions, this variable enables perl code to determine whether actions that make sense only during normal running are warranted. See *perlvar*.

91.3.53 New variable $ ˆV contains Perl version as a string

`$^V` contains the Perl version number as a string composed of characters whose ordinals match the version numbers, i.e. v5.6.0. This may be used in string comparisons.

See `Support for strings represented as a vector of ordinals` for an example.

91.3.54 Optional Y2K warnings

If Perl is built with the cpp macro `PERL_Y2KWARN` defined, it emits optional warnings when concatenating the number 19 with another number.

This behavior must be specifically enabled when running Configure. See *INSTALL* and *README.Y2K*.