cinoptions - VIM Documentation

Bruce Dearing
100036623
Acadia University
036623d@acadiau.ca

November 9, 2013

cinoptions-values The 'cinoptions' option sets how Vim performs indentation. In the list below, N represents a number of your choice (the number can be negative). When there is an 's' after the number, Vim multiplies the number by 'shiftwidth': 1s is 'shiftwidth' 2s is two times 'shiftwidth', etc. You can use a decimal point, too: "-0.5s" is minus half a 'shiftwidth'. The examples below assume a 'shiftwidth' of 4.

>N Amount added for "normal" indent. Used after a line that should increase the indent (lines starting with "if", an opening brace, etc.). (default 'shiftwidth').

```
cino=            cino=>2        cino=>2s
  if (cond)        if (cond)      if (cond)
  {                {              {
      foo;             foo;               foo;
  }                }              }
```

eN Add N to the prevailing indent inside a set of braces if the opening brace at the End of the line (more precise: is not the first character in a line). This is useful if you want a different indent when the '{' is at the start of the line from when '{' is at the end of the line. (default 0).

```
cino=            cino=e2        cino=e-2
  if (cond) {      if (cond) {    if (cond) {
      foo;               foo;       foo;
  }                }              }
  else             else           else
  {                {              {
      bar;             bar;           bar;
  }                }              }
```

nN Add N to the prevailing indent for a statement after an "if", "while", etc., if it is NOT inside a set of braces. This is useful if you want a different indent when there is no '{' before the statement from when there is a '{' before it. (default 0).

```
cino=          cino=n2          cino=n-2
  if (cond)      if (cond)        if (cond)
      foo;             foo;         foo;
  else                else        else
  {                   {           {
      bar;                bar;        bar;
  }                   }           }
```

fN Place the first opening brace of a function or other block in column N. This applies only for an opening brace that is not inside other braces and is at the start of the line. What comes after the brace is put relative to this brace. (default 0).

```
cino=          cino=f.5s           cino=f1s
  func()         func()              func()
  {                 {                  {
      int foo;         int foo;            int foo;
```

{N Place opening braces N characters from the prevailing indent. This applies only for opening braces that are inside other braces. (default 0).

```
cino=            cino={.5s         cino={1s
  if (cond)          if (cond)           if (cond)
  {            {               {
      foo;          foo;               foo;
```

}N Place closing braces N characters from the matching opening brace. (default 0).

```
cino=            cino={2,}-0.5s     cino=}2
  if (cond)        if (cond)           if (cond)
  {                  {                 {
      foo;             foo;                foo;
  }                }                     }
```

ˆN Add N to the prevailing indent inside a set of braces if the opening brace is in column 0. This can specify a different indent for whole of a function (some may like to set it to a negative number). (default 0).

```
cino=              cino=^-2          cino=^-s
  func()             func()            func()
  {                  {                 {
      if (cond)        if (cond)       if (cond)
      {                {               {
          a = b;           a = b;          a = b;
      }                }               }
  }                  }                 }
```

LN Controls placement of jump labels. If N is negative, the label will be placed at column 1. If N is non-negative, the indent of the label will be the prevailing indent minus N. (default -1).

```
cino=               cino=L2             cino=Ls
  func()              func()              func()
  {                   {                   {
      {                   {                   {
          stmt;               stmt;               stmt;
  LABEL:                      LABEL:            LABEL:
      }                   }                   }
  }                   }                   }
```

:N Place case labels N characters from the indent of the switch(). (default 'shiftwidth').

```
cino=              cino=:0
  switch (x)         switch(x)
  {                  {
      case 1:        case 1:
          a = b;         a = b;
      default:       default:
  }                  }
```

=N Place statements occurring after a case label N characters from the indent of the label. (default 'shiftwidth').

```
cino=            cino==10
  case 11:         case 11:  a = a + 1;
      a = a + 1;             b = b + 1;
```

lN If N != 0 Vim will align with a case label instead of the statement after it in the same line.

```
cino=                    cino=l1
  switch (a) {             switch (a) {
      case 1: {                case 1: {
                  break;           break;
              }                        }
```

bN If N != 0 Vim will align a final "break" with the case label, so that case..break looks like a sort of block. (default: 0). When using 1, consider adding "0=break" to cinkeys.

```
cino=               cino=b1
  switch (x)          switch(x)
  {                   {
      case 1:             case 1:
          a = b;              a = b;
          break;          break;

      default:            default:
          a = 0;              a = 0;
          break;          break;
  }                   }
```

gN Place C++ scope declarations N characters from the indent of the block they are in. (default 'shiftwidth'). A scope declaration can be "public:", "protected:" or "private:".

```
cino=               cino=g0
  {                 {
      public:       public:
          a = b;        a = b;
      private:      private:
  }                 }
```

```
    hN    Place statements occurring after a C++ scope declaration N
          characters from the indent of the label.  (default
          'shiftwidth').

        cino=             cino=h10
           public:         public:   a = a + 1;
               a = a + 1;              b = b + 1;

    pN    Parameter declarations for K&R-style function declarations will
          be indented N characters from the margin.  (default
          'shiftwidth').

        cino=             cino=p0         cino=p2s
          func(a, b)          func(a, b)      func(a, b)
              int a;          int a;              int a;
              char b;          char b;              char b;

    tN    Indent a function return type declaration N characters from the
          margin.  (default 'shiftwidth').
        cino=           cino=t0        cino=t7
              int         int             int
          func()         func()          func()

    iN    Indent C++ base class declarations and constructor
          initializations, if they start in a new line (otherwise they
          are aligned at the right side of the ':').
          (default 'shiftwidth').

        cino=             cino=i0
          class MyClass :        class MyClass :
              public BaseClass      public BaseClass
          {}                 {}
          MyClass::MyClass() :        MyClass::MyClass() :
              BaseClass(3)         BaseClass(3)
          {}                 {}

    +N    Indent a continuation line (a line that spills onto the next)
              inside a function N additional characters.  (default
              'shiftwidth').
              Outside of a function, when the previous line ended in a
              backslash, the 2 * N is used.

        cino=             cino=+10
          a = b + 9 *            a = b + 9 *
              c;                  c;

    cN    Indent comment lines after the comment opener, when there is no
          other text with which to align, N characters from the comment
          opener.  (default 3).

        cino=             cino=c5
          /*                 /*
             text.              text.
           */                 */

    CN    When N is non-zero, indent comment lines by the amount specified
          with the c flag above even if there is other text behind the
          comment opener.  (default 0).

        cino=c0             cino=c0,C1
          /********            /********
            text.             text.
          ********/            ********/
           (Example uses ":set comments & comments-=s1:/* comments^=s0:/*")
```

/N   Indent comment lines N characters extra. (default 0).

```
cino=              cino=/4
  a = b;              a = b;
  /* comment */              /* comment */
  c = d;              c = d;
```

(N   When in unclosed parentheses, indent N characters from the line with the unclosed parentheses. Add a 'shiftwidth' for every unclosed parentheses. When N is 0 or the unclosed parentheses is the first non-white character in its line, line up with the next non-white character after the unclosed parentheses. (default 'shiftwidth' * 2).

```
cino=              cino=(0
  if (c1 && (c2 ||        if (c1 && (c2 ||
          c3))               c3))
      foo;            foo;
  if (c1 &&            if (c1 &&
      (c2 || c3))        (c2 || c3))
     {                   {
```

uN   Same as (N, but for one level deeper. (default 'shiftwidth').

```
cino=              cino=u2
  if (c123456789      if (c123456789
      && (c22345          && (c22345
          || c3))            || c3))
```

UN   When N is non-zero, do not ignore the indenting specified by (or u in case that the unclosed parentheses is the first non-white character in its line. (default 0).

```
cino= or cino=(s      cino=(s,U1
  c = c1 &&            c = c1 &&
      (                (
       c2 ||               c2 ||
       c3                c3
      ) && c4;            ) && c4;
```

wN   When in unclosed parentheses and N is non-zero and either using "(0" or "u0", respectively, or using "U0" and the unclosed parentheses is the first non-white character in its line, line up with the character immediately after the unclosed parentheses rather than the first non-white character. (default 0).

```
    cino=(0              cino=(0,w1
      if (   c1            if (   c1
         && (   c2        && (   c2
            || c3))            || c3))
          foo;            foo;

WN    When in unclosed parentheses and N is non-zero and either
      using "(0" or "u0", respectively and the unclosed parentheses is
      the last non-white character in its line and it is not the
      closing parentheses, indent the following line N characters
      relative to the outer context (i.e. start of the line or the
      next unclosed parentheses).  (default: 0).

    cino=(0               cino=(0,W4
      a_long_line(            a_long_line(
              argument,        argument,
              argument);    argument);
      a_short_line(argument,    a_short_line(argument,
               argument);         argument);

mN    When N is non-zero, line up a line starting with a closing
      parentheses with the first character of the line with the
      matching opening parentheses.  (default 0).

    cino=(s              cino=(s,m1
      c = c1 && (            c = c1 && (
          c2 ||            c2 ||
          c3            c3
          ) && c4;            ) && c4;
      if (                if (
          c1 && c2            c1 && c2
         )                )
          foo;            foo;

MN    When N is non-zero, line up a line starting with a closing
      parentheses with the first character of the previous line.
      (default 0).

    cino=              cino=M1
      if (cond1 &&            if (cond1 &&
         cond2               cond2
         )                   )


                *java-cinoptions* *java-indenting*
jN    Indent java anonymous classes correctly.  The value '<a href="pattern.html#N">
```

```
        currently unused but must be non-zero (e.g. 'j1').  'j1' will
        indent for example the following code snippet correctly:

      object.add(new ChangeListener() {
          public void stateChanged(ChangeEvent e) {
          do_something();
          }
      });


                *javascript-cinoptions* *javascript-indenting*
  JN    Indent JavaScript object declarations correctly by not confusing
        them with labels.  The value 'N' is currently unused but must be
        non-zero (e.g. 'J1').

      var bar = {
          foo: {
          that: this,
          some: ok,
          },
          "bar":{
          a : 2,
          b: "123abc",
          x: 4,
          "y": 5
          }
      }

  )N    Vim searches for unclosed parentheses at most N lines away.
        This limits the time needed to search for parentheses.  (default
        20 lines).

  *N    Vim searches for unclosed comments at most N lines away.  This
        limits the time needed to search for the start of a comment.
        (default 70 lines).

  #N    When N is non-zero recognize shell/Perl comments, starting with
        '#'.  Default N is zero: don't recognizes '#' comments.  Note
        that lines starting with # will still be seen as preprocessor
        lines.


The defaults, spelled out in full, are:
    cinoptions=>s,e0,n0,f0,{0,}0,^0,L-1,:s,=s,l0,b0,gs,hs,ps,ts,is,+s,
            c3,C0,/0,(2s,us,U0,w0,W0,m0,j0,J0,)20,*70,#0
```

```
Vim puts a line in column 1 if:
- It starts with '#' (preprocessor directives), if cinkeys contains '#'.
- It starts with a label (a keyword followed by ':', other than "case" and
  "default") and 'cinoptions' does not contain an 'L' entry with a positive
  value.
- Any combination of indentations causes the line to have less than 0
  indentation.
```