C−C++ Beautifier HOW−TO

Al Dev (Alavoor Vasudevan)

< alavoor[AT]yahoo.com >

v16.4, 03 May 2003

This document will help you to format (beautify) the C/C++ programs so that it is more readable and confirms to your site C/C++ coding standards. The information in this document applies to all the operating sytems that is − Linux, MS DOS, Apple Macintosh, Windows 95/NT/2000, BeOS, OS/2, IBM OSes, all flavors of Unix like Solaris, HPUX, AIX, SCO, Sinix, BSD, UnixWare, etc.. and to all other operating systems which support "C" compiler (it means almost all the operating systems on this planet!).

Table of Contents

1. Introduction

(The latest version of this document is at "http://www.milkywaygalaxy.freeservers.com" . You may want to check there for changes).

Coding standards for C/C++ or any language is required in order to make the programs more readable/understandable by programmers. There are C/C++ beautifiers (formating tools) to accomplish this goal. Formatted (beautified) code improves the *productivity* of programmers by *2 times* !!

On Linux/Unixes there is a command called *"indent"* and *"cb"* . Refer to 'man indent' and 'man cb'. Note that *indent* and *cb* work for only "C" programs. For "C++" programs use *"bcpp"* .

Important NOTE: To compile bcpp under unix, unpack bcpp.tar.gz and you MUST change directory to "code" and give a make. Do not change to "unix" directory and give a make. That will give lots of errors.

Download the beautifier program from one of the following

- If you are having trouble downloading software from any of the sites below, then download for a small cost from my site at "http://www.milkywaygalaxy.freeservers.com" . The cost is very small to maintain this web site. Some of the free sites below are not maintained properly.
- C++ : BCPP site is at "http://dickey.his.com/bcpp/bcpp.html" or at "http://www.clark.net/pub/dickey" . BCPP ftp site is at "ftp://dickey.his.com/bcpp/bcpp.tar.gz"
- C++ : "http://www.consultix−inc.com/c++b.html"
- C : "http://www.chips.navy.mil/oasys/c/" and mirror at Oasys
- C++ : "http://www.semdesigns.com/Products/DMS/DMSToolkit.html"
- C++, C, Java and Oracle Pro−C Beautifier "http://www.geocities.com/~starkville/main.html"
- C++, C beautifier "http://users.erols.com/astronaut/vim/ccb−1.07.tar.gz" and site at "http://users.erols.com/astronaut/vim/#vimlinks_src"
- C++, C, Java, Perl beautifier CBP "http://www.prismtk.de/docs/cbp"
- GC! GreatCode! is a powerful C/C++ source code beautifier Windows 95/98/NT/2000 "http://perso.club−internet.fr/cbeaudet"
- CbVan for C, C++ and Java at "http://www.geocities.com/~starkville/main.html"
- Artistic Style beautifier for C, C++, Java at "http://sourceforge.net/projects/astyle" "http://astyle.sourceforge.net" .

I used BCPP to format the C++ programs and it worked fine for me. You may want to check other tools and use the one which you may like the most.

BCPP was written by Steven De Toni at steve@alpha.ocbbs.gen.nz

---

2. Installing BCPP

To install BCPP

```
        Unpacking bcpp:
                tar zxvf bcpp.tar.gz
        Building bcpp:

                cd code
                make
                cp bcpp ~/local/bin/
                cp bcpp.cfg ~/local/bin
        Usage:
                bcpp -h  or bcpp -?
        Recommended: Always use spaces instead of tabs in indenting during beautifying.
        Use the option -s so that the code looks the same in all types of editors like vi,
        emacs, MS DOS edit, Notepad, Wordpad etc..
                bcpp -s *.cpp
        In Solaris, you can also use:
                ls *.cpp | xargs -I{} -t bcpp -s {}
        in Linux, you can also use:
                ls *.cpp | xargs -i{} -t bcpp -s {}
```

3. How can I trust Beautifier programs??!!

For 100[percnt] assurance you need a *SCIENTIFIC* way to validate and trust a beautifier program. The method described in this section will enable the beautifier program to be accepted as "trust−worthy" and reliable.

In order to verify that beautifier programs like *bcpp* , *indent* or *cb* is not damaging or changing the input source−code after formatting, you can use one of the following technique −

3.1. Method 1: Verfication Program for C++/C

```
        bash$ man diff
        bash$ diff -b --ignore-all-space originalfile formattedfile
```

3.2. Method 2: Verfication Program for C++/C

Generate the object code from the original input source code using the compiler −

```
          g++ -c myprogram.cpp
```

Here g++ is GNU C++ compiler. This will create object output myprogram.o

Save this file −

```
           mv myprogram.o myprogram_orig.o
```

Now run bcpp −

```
           bcpp myprogram.cpp
```

This will create the formatted output program file myprogram.cpp and move the original file to myprogram.cpp.orig. Compile the new file with −

```
           g++ -c myprogram.cpp
```

Now use the unix 'diff' command to compare the two object files −

```
           diff myprogram.o myprogram_orig.o
```

Both these files *MUST BE IDENTICAL* . This verifies that bcpp is working perfectly. On DOS or Windows 95 you may want to use the free Cygnus Cygwin 'diff' or 'MKS' utilities.

If for some reason you are not able to diff the object files then you MUST use the assembly output as described below.

You can use the assembler output instead of object output from the C++ compiler for doing the comparison. Like −

```
            g++ −S myprogram.cpp
```

This creates myprogram.s. Verify with −

```
            diff myprogram.s myprogram_orig.s
```

This step gives 100[percnt] guarantee that your valuable source code is intact and bcpp is JUST doing ONLY formatting and is NOT changing or damaging your code in any way. This method gives you 100[percnt] quality assurance and life term or long term *WARRANTY* on beautifier programs like 'bcpp', 'cb' or 'indent'.

It is strongly recommended that you do these two steps every time you run beautifier programs like *bcpp* , *indent* or *cb* .

3.3. Method 3: Verfication Program for Java/C++/Others

Since you cannot compile the Java source code to machine code and you can compile Java source to byte−codes you cannot use the technique given in Method 2 above. When you do diff on Java class files it will always be different.

In this method, a different technique will be given which can be used to validate any beautifier program for Java. Also this method is quite powerful and can be used to validate any beautifier program for any language like C, C++, PERL, SQL, HTML or Java. Since all beautifier program simply rearrange or insert whitespaces , you can strip all the whitespaces from original source file and dump it to a file called verify1.out and strip all the whitespaces from beautified source file and dump it to a file called verify2.out. Now, do a diff on verify1.out and verify2.out. If there is no difference, then beautifier program is working properly. The method is not 100[percnt] perfect and can catch atleast 98[percnt] of the errors/bugs in the beautifier program. Use this method in conjunction with other methods. But this method is better than not having a verification at all and blindly trusting the beautifier program!!

Note: A whitespace can be one of following − blank space ' ', form−feed '\f', newline '\n', carriage return '\r', horizontal tab '\t' or vertical tab '\v'.

```
        bash$ java StripWhitespaces  sample.java > verify1.out
        bash$ java StripWhitespaces  sample_beutified.java > verify2.out
        bash$ diff verify1.out verify2.out
        bash$ java StripWhitespaces  sample.cpp > verify1.out
        bash$ java StripWhitespaces  sample_beutified.cpp > verify2.out
        bash$ diff verify1.out verify2.out
        bash$ java StripWhitespaces  sample.sql > verify1.out
        bash$ java StripWhitespaces  sample_beutified.sql > verify2.out
        bash$ diff verify1.out verify2.out
```

The source code of StripWhitespaces Java program is not given here. It is left as an exercise for students (you) to write a small program in Java which will simply strip whitespaces from the input text file and output to standard console output. Students are also urged to write this small program (StripWhitespaces) in C, PERL, Unix shell script (Korn, Bourne) and AWK script. Students can see howto the same task can be accomplished in these five different languages and can do comparison of ease of programing. You should *put a newline '\n'* *character after every 50 characters* while generating verify1.out and verify2.out so that when you do a diff you can see on which lines differences are coming up. Otherwise, verify*.out files will just contain one line and it will be difficult to pin−point where exactly the beautifier program is failing (got this point ???).

3.4. Method 4: Shell script: Verfication Program for C++/C

This is a Korn shell script to verify beautifier program. Requires "pdksh*.rpm" from Linux 'contrib' cdrom. Save this file as 'text' file and chmod a+rx on it. You can re−write this shell script in PERL so that you can use it on Window 95/NT or MSDOS. Uncomment the PRGM variable to point to *bcpp* , *cb* or *indent*

```
        #!/bin/ksh
        # Verification program to check C++ Beautifiers 'bcpp', 'indent' or cb
        ############################################################
        # Copyright
        # The copyright policy is GNU/GPL.
        # Author: Al Dev (Alavoor Vasudevan) alavoor[AT]yahoo.com
        ############################################################
        check_beautify_now()
        {
                # Remove all the temp files....
                \rm −f ${TMP_FILE}
                \rm −f ${TMP_CPPFILE}*.*
                FNAME=$1
                if [ ! −f ${FNAME} ]; then
                        print "\nError: The file ${FNAME} does not exist!!. Aborting now ...."
                        exit
                fi
                \cp  −f ${FNAME} ${TMP_CPPFILE}.cpp
                ${COMPILER} −c ${TMP_CPPFILE}.cpp
                if [ ! −f ${TMP_CPPFILE}.o ]; then
                        print "Fatal Error: Failed to compile ${FNAME}. Aborting now... "
                        exit
                fi
                \mv −f ${TMP_CPPFILE}.o ${TMP_CPPFILE}_orig.o
                aa=`basename $PRGM`
                print "\nRunning, verifying $aa on ${FNAME}"
                ${PRGM} ${TMP_CPPFILE}.cpp
                ${COMPILER} −c ${TMP_CPPFILE}.cpp
                \rm −f $TMP_FILE
                diff ${TMP_CPPFILE}.o ${TMP_CPPFILE}_orig.o 1> $TMP_FILE 2>> $TMP_FILE
                result=""
                result=`wc −c $TMP_FILE | awk '{print $1}' `
                if [ "$result" = "0" ]; then
                        print "Success!! Beautifier $aa is working properly!!\n"
                else
                        print "Fatal Error: Something wrong!! Beautifier is not working!!"
                        exit
                fi
        #       ${COMPILER} −S ${TMP_CPPFILE}.cpp
        #       diff ${TMP_CPPFILE}.s ${TMP_CPPFILE}_orig.s
                # Remove all the temp files....
                \rm −f ${TMP_FILE}
                \rm −f ${TMP_CPPFILE}*.*
        }
        ########## Main of program begins here ##################3
        #PRGM=/usr/bin/bcpp
        #PRGM=/usr/bin/cb
        PRGM=/usr/bin/indent
        COMPILER=/usr/bin/g++
        TMP_FILE=beautify.tmp
        TMP_CPPFILE=beautify−tmp_cppfile
        print −n "Enter the C++ file name <default is *.cpp> : "
        read ans
        if [ "$ans" = "" −o "$ans" = " " ]; then
                ans="ALL"
        else
                FILENAME=$ans
        fi
        # Remove all the temp files....
        \rm −f ${TMP_FILE}
        \rm −f ${TMP_CPPFILE}*.*
        if [ "$ans" != "ALL" ]; then
                check_beautify_now ${FILENAME}
        else
                ls *.cpp |
                while read FILENAME
                do
                        check_beautify_now ${FILENAME}
                done
        fi
```

4. Beautifiers for other Languages

Visit the following sites to get beautifiers for other languages like HTML, SQL, Java, Perl, Fortran.

- HTML : "http://www.digital−mines.com/htb/"
- HTML : "http://www.datacomm.ch/mwoog/software/perl/beautifier.html"
- HTML : "http://www.watson−net.com/free/perl/s_fhtml.asp"
- SQL : "http://www.netbula.com/products/sqlb"
- Oracle PLSQL : http://www.revealnet.com
- GPL "http://www.geocities.com/~starkville/vancbj.html"
- GPL "http://kevinkelley.mystarband.net/java/dent.html"
- Free "http://www.tiobe.com/jacobe.htm"
- Free "http://www.mmsindia.com/JPretty.html"
- Free "http://members.magnet.at/johann.langhofer/products/jxbeauty/overview.html" (has JBuilder support)
- Free "http://www.semdesigns.com/Products/Formatters/JavaFormatter.html"
- Commercial $24.99 "http://smartbeautify.com"
- Commercial $129 "http://www.jindent.com"
- Google "http://directory.google.com/Top/Computers/Programming/Languages/Java/Development_Tools/Code_Beauti
- Java, SQL, HTML, C++ : "http://www.semdesigns.com/Products/DMS/DMSToolkit.html"
- Java JIndent "http://home.wtal.de/software−solutions/jindent"
- Java Pat "http://javaregex.com/cgi−bin/pat/jbeaut.asp"
- Java JStyle "http://www.redrival.com/greenrd/java/jstyle"
- Java JPrettyPrinter "http://www.epoch.com.tw/download/ms/java/java.htm"
- Java JxBeauty "http://members.nextra.at/johann.langhofer/download/jxbeauty" and the JxBeauty Home
- Java beautify percolator
- Java list "http://www.java.about.com/compute/java/library/weekly/aa102499.htm"
- Java html present VasJava2HTML
- Java code colorifier and beautifier "http://www.mycgiserver.com/~lisali/jccb"
- Perl : "http://www.consultix−inc.com/www.consultix−inc.com/talk.htm"
- Perl : "http://www.consultix−inc.com/www.consultix−inc.com/perl_beautifier.html"
- Fortran beautifier : "http://www.aeem.iastate.edu/Fortran/tools.html"
- C++ : BCPP site is at "http://dickey.his.com/bcpp/bcpp.html" or at "http://www.clark.net/pub/dickey" . BCPP ftp site is at "ftp://dickey.his.com/bcpp/bcpp.tar.gz"
- C++ : "http://www.consultix−inc.com/c++b.html"
- C : "http://www.chips.navy.mil/oasys/c/" and mirror at Oasys
- C++, C, Java, Oracle Pro−C Beautifier "http://www.geocities.com/~starkville/main.html"
- C++, C beautifier "http://users.erols.com/astronaut/vim/ccb−1.07.tar.gz" and site at "http://users.erols.com/astronaut/vim/#vimlinks_src"
- GC! GreatCode! is a powerful C/C++ source code beautifier Windows 95/98/NT/2000 "http://perso.club−internet.fr/cbeaudet"
- C++ beautifier 'SourceStyler' "http://www.ochre.com.au"

- White paper on beautifier : "http://www.consultix−inc.com/www.consultix−inc.com/talk.htm"

To create presentation of codes to display using HTML −

- Presentation (C,C++,Java) to html : "http://www.perlstudio.de/cbindex.html"

- The GNU project: Source−Highlight: "http://www.gnu.org/software/src−highlite" and its KDE frontend Ksrc2html "http://murphy.netsolution−net.de/Ksrc2.html"

Also search the search engines like "http://www.yahoo.com" or "http://www.lycos.com" and search for keyword "beautfier".

5. Beautifiers For SGML and XML

To beautify SGML and XML use one of these perl scripts. I used perl script from Kevin and it works fine for me.

5.1. SGML Auto−Indenter By Kevin

This script was originally written by Kevin M. Dunn kdunn@hsc.edu Department of Chemistry Hampden−Sydney College HSC, VA 23943 (804) 223−6181 (804) 223−6374 (Fax). And this script here was modified and enhanced by Al Dev alavoor[AT]yahoo.com.

Several people have discussed the use of Tidy to indent sgml and xml sources, but does not work for SGML documents, as Tidy did not recognize the entities. Rather than fix Tidy, here is the perl script to indent anything with sgml−type tags. Only non−empty tags are indented, and text is justified at 80 characters/line (easily changed).

Known problems: will break line−specific enviroments. So far, the script is quite general−−it does not recognize specific tags and so could be used for any xml or sgml, not just docbook. Is there any way to recognize literal text independent of DTD? Leading whitespace, for example? Trailing whitespace? Or I could indent tags only, and leave all non−tag text unjustified and unindented.

```
#!/usr/bin/perl -w
#
# sb: the sgml beautifier
# indents non-empty sgml tags
# usage: sb filename or sb < filename or | sb
# author: Kevin M. Dunn (kdunn@hsc.edu), Modified by Al Dev (alavoor[AT]yahoo.com)
# license: anyone is free to use this for any purpose whatever
#
use strict;
use diagnostics;

sub separate_tags
{
        @_ < 1 ?  die "\nInsufficient args .. " : 0 ;
        my ($tmpfile) = @_;
        my ($current_line);
        open(FILETMP, ">$tmpfile");
        while (<>)
        {
                $current_line = $_;
                #if ($current_line =~ /^\s+$/)
                if ($current_line eq "\n")
                {
                        # Pad spaces to distinguish/identify this line with other newlines
                        # so that this line is printed and not bypassed in indent_tags()
                        $current_line = "\t  " . $current_line;  # Prepend with spaces
                        #print "\ndone padding\n";
                        #sleep 5;
                }
                #$_ =~ s/^\s+//;  # Left trim the leading white spaces - ltrim
                #$_ =~ s/\s+$//;  # Right trim the trailing white spaces - rtrim
                $current_line =~ s/</\n</g;  # Put newline before start of tag "<"
                $current_line =~ s/>/>\n/g;  # Put newline after end of tag ">"
                print FILETMP "$current_line";
        }
        close(FILETMP);
}

sub get_tags
{
        @_ < 1 ?  die "\nInsufficient args .. " : 0 ;
        my ($tmpfile) = @_;
        open(FILETMP, "$tmpfile");
        my ($word);
        while (<FILETMP>)
        {
                $word = $_;
                $word =~ s/[> ].*//;
                chomp($word);
                if ( $word =~ /^<\/.*/ )
                {
                        $sgb::tag2{$word} = 1; # here the word has something like '</TITLE'
                        $word =~ s/\///;
                        $sgb::tag1{$word} = 1; # here the word has something like '<TITLE'
                }
        }
}

sub indent_tags
{
        @_ < 1 ?  die "\nInsufficient args .. " : 0 ;
        my ($tmpfile) = @_;
        my $jl = 80; #text will be justified to 80 characters/line
        my $nl = 0;
        my $sp = 0;
        my @space;
        $space[0] = "";

        my $newline = ""; # hack to prevent extraneous blank first line

        open(FILETMP, "$tmpfile");
        my ($current_line, $word, $saveword);
        while (<FILETMP>)
        {
                chomp($_); # avoid \n on last field
                $current_line = $_;
                $word = $current_line;
                $word =~ s/[> ].*//;  # truncate trailing "> " and spaces therafter
                if ( $sgb::tag1{$word} )
                {
                        $saveword = $word;
                        print "\n$space[$sp]$current_line";
                        $nl = $jl; # force new line on next line of input
                        $sp++;
                        if ( ! $space[$sp] )
                        {
                                $space[$sp] = $space[$sp−1] . "  ";
                        }
                }
                elsif ( $sgb::tag2{$word} )
                {
                        $saveword = $word;
                        $sp−−;
                        # If the tag is <ProgramListing> then do not justify...
                        if (lc($word) eq "</programlisting")
                        {
                                print "$current_line";
                        }
                        else
                        {
                                print "\n$space[$sp]$current_line";
                        }
                        $nl = $jl; # force new line on next line of input
                }
                elsif ( $word =~ /<.*/ )
                {
                        $saveword = $word;
                        print "$newline$space[$sp]$current_line";
                        $newline = "\n"; # hack to prevent extraneous blank first line
                        $nl = $jl; # force new line on next line of input
                }
                elsif ( length($current_line) > 0 )
                {
                        # If the tag is <ProgramListing> then do not justify...
                        if (lc($saveword) eq "<programlisting")
                        {
                                #print "\nthe tag1 word is $saveword----eof \n";
                                #print "$newline$space[$sp]$current_line";
                                # DO NOT put any tabs or spaces, because repeated running of this
                                # on same file will keep putting tabs or spaces.
                                print "$newline$current_line";
                                $newline = "\n"; # hack to prevent extraneous blank first line
                                $nl = $jl; # force new line on next line of input
                        }
                        else
                        {
                                $nl = justify($jl, $nl, $sp, $current_line, @space);
                        }
                }
        }
}

sub justify
{
        @_ < 4 ?  die "\nInsufficient args .. " : 0 ;
        my ($jl, $nl, $sp, $current_line, @space) = @_;

        my @words = split;
        my $nw = @words;
        for (my $i = 0; $i < $nw; $i++ )
        {
                $sgb::ll += length($words[$i]) + 1 + $nl; # line length if this word is added
                if ($sgb::ll < $jl) # if short enough, print it
                {
                        print "$words[$i] ";
                        $nl = 0;
                }
                else # if line is too long, start a new one
                {
                        print "\n$space[$sp]$words[$i] ";
                        $nl = 0;
                        $sgb::ll = length($space[$sp] . $words[$i]) + 1;
                }
        }
        return $nl;
}

$sgb::ll = 0; # global var
```

```
my $tmpfile = "$$.tmp";
separate_tags($tmpfile);
get_tags($tmpfile);
indent_tags($tmpfile);
unlink ("$tmpfile"); # remove temporary file
print "\n"; # add final line to output
```

5.2. SGML Auto−Indenter By Hector

Download from "http://www.olea.org/tmp/indent−sgml−xml" . The author is at hector@debian.org And this script here was modified and enhanced by Al Dev alavoor[AT]yahoo.com.

The program below uses the XML::Parser. Read the online manual page with 'man XML::Parser::Expat' and also 'man XML::Parser'.

```
#!/usr/bin/perl -w
#

# Author: Hector (hector@debian.org). Modified by Al Dev (alavoor[AT]yahoo.com)

# For documentation please see 'man XML::Parser::Expat' and
# also see 'man XML::Parser'

use diagnostics;
use XML::Parser::Expat;

$|=1;

if ( !$ARGV[0] )
{
       print "Argument missing\n";
       exit 1;
}

$inline_tags = "acronym|ulink|link|citetitle|firstname|surname|application|guimenu|guisubmenu|gui
$one_line = "title|member";

$todo = "";
$temp = "";
$ancho = "  ";
$indentacion = 0;

#open IN , "<$ARGV[0]";
#my $todo = join ('', <IN>);
#close (IN);

$parser = new XML::Parser::Expat;
$parser->setHandlers('Start'   => \&inicio,
                     'End'     => \&fin,
                                 'Char'    => \&cadena,
                         'Comment' => \&comentario);

open(FOO, "$ARGV[0]") or die "Couldn't open";

# If you get this type of error:
# syntax error at line 1, column 0, byte 0 at ../sgml-beautifier-indentar.pl line 37
# Then edit input file $ARGV[0] and change put this line -
#       <?xml version="1.0" encoding="utf−8"?>
$parser−>parse(*FOO);
close(FOO);


$todo =~ s/\n+/\n/gm;
$todo =~ s/\n *\n/\n/gm;
print "$todo\n";

exit 1;

sub inicio
{
        my ($p, $el, %atts) = @_;
        my $tag = "<$el";
        foreach my $key ( sort %atts)
        {
                if ( $atts{$key} )
                {
                        $tag .= " $key=\"$atts{$key}\"";
                }
        }
        $tag .= ">";

        if ( !($el =~ /$inline_tags|$one_line/) )
        {
                $temp = &indentar ($temp, $indentacion);
                if ( $temp ) {
                        $todo .= "$temp\n";
                }
                my $pad = $ancho x $indentacion;
                $todo .= "$pad$tag\n";
                $temp = "";
                $indentacion++;
        }
        else
        {
                $temp .= $tag;
        }
}

sub fin
{
        my ($p, $el) = @_;
        my $tag = "</$el>";
        if ( !($el =~ /$inline_tags/) )
        {
                $temp = &indentar ($temp, $indentacion);
                $temp =~ s/\n$// ;
                $todo .= "$temp";
                if ( !($el =~ /$one_line/) )
                {
                        $indentacion−−;
                        if ( !($todo =~ /\n$/) ) {
                                $todo .= "\n";
                        }
                        my $pad = $ancho x $indentacion;
                        $todo .= "$pad";
                        #$indentacion−−;
                }
                $todo .= "$tag\n";
                $temp = "";
                #               $indentacion++;
        }
        else
        {
                $temp .= "$tag";
        }
}

sub cadena
{
        my ($p, $str) = @_;
        $str =~ s/ +/ /g;
        #$str =~ s/^ //;
        #$str =~ s/ $//;
        $temp .= "$str";
}

sub comentario
{
        my ($p, $str) = @_;
        $todo .= "<!--\n $str \n-->\n";
}


sub indentar ()
{
        my $linea = $_[0] ;
        #       print ("Indentacion es $_[1]\nLinea $_[0]\n");
        my $indentacion = $_[1];
        my $cantidad = 75 − ( $indentacion * length($ancho));
        my $pad = $ancho x $indentacion;

        my $temp = &cortar_linea ( $linea, $cantidad);
        $temp =~ s/\n/\n$pad/g;
        $temp =~ s/^ //;
        my $resultado = "$pad$temp\n";
        return $resultado;
}

sub cortar_linea ()
{
        my $linea = $_[0];
        #$linea =~ s/\n/ \n/;
        $linea .= " ";
        my $cantidad = $_[1];
        $temp = "";
        $temp2 = "";
        #print "Llega $linea\n";
        while ( $linea =~ /(.+?) / )
        {
                if ( (length ($temp) + length ($+)) <= $cantidad )
                {
                        $temp .= "$+ ";
                        $linea = $';
                }
                elsif ( length ($+) >= $cantidad )
                {
                        $linea = $';
                        $temp2 .= "$temp\n$+";
                        $temp = "";
                }
                else
                {
                        $temp2 .= "$temp\n";
                        $temp = "$+ ";
                        $linea = $';
                }
        }
        $temp2 .= "$temp\n";
        $temp2 =~ s/\n$//;
        $temp2 =~ s/ $//;
        #       print "Sale\n##$temp2##\n";
        return $temp2;
}
```

6. Related URLs

Visit following locators which are related to C, C++ −

- Vim color text editor for C++, C
- C++ Programming HOWTO
- CVS HOWTO for C++ programs
- Linux goodies "http://www.milkywaygalaxy.freeservers.com" and mirrors at "http://aldev0.webjump.com" , angelfire , geocities , virtualave , 50megs , theglobe , NBCi , Terrashare , Fortunecity , Freewebsites , Tripod , Spree , Escalix , Httpcity , Freeservers .

7. Other Formats of this Document

This document is published in 14 different formats namely − DVI, Postscript, Latex, Adobe Acrobat PDF, LyX, GNU−info, HTML, RTF(Rich Text Format), Plain−text, Unix man pages, single HTML file, SGML (Linuxdoc format), SGML (Docbook format), MS WinHelp format.

This howto document is located at −

- "http://www.linuxdoc.org" and click on HOWTOs and search for howto document name using CTRL+f or ALT+f within the web−browser.

You can also find this document at the following mirrors sites −

- "http://www.caldera.com/LDP/HOWTO"
- "http://www.linux.ucla.edu/LDP"
- "http://www.cc.gatech.edu/linux/LDP"
- "http://www.redhat.com/mirrors/LDP"
- Other mirror sites near you (network−address−wise) can be found at "http://www.linuxdoc.org/mirrors.html" select a site and go to directory /LDP/HOWTO/xxxxx−HOWTO.html

- You can get this HOWTO document as a single file tar ball in HTML, DVI, Postscript or SGML formats from − "ftp://www.linuxdoc.org/pub/Linux/docs/HOWTO/other−formats/" and "http://www.linuxdoc.org/docs.html#howto"
- Plain text format is in: "ftp://www.linuxdoc.org/pub/Linux/docs/HOWTO" and "http://www.linuxdoc.org/docs.html#howto"
- Single HTML file format is in: "http://www.linuxdoc.org/docs.html#howto" Single HTML file can be created with command (see man sgml2html) − sgml2html −split 0 xxxxhowto.sgml
- Translations to other languages like French, German, Spanish, Chinese, Japanese are in "ftp://www.linuxdoc.org/pub/Linux/docs/HOWTO" and "http://www.linuxdoc.org/docs.html#howto" Any help from you to translate to other languages is welcome.

The document is written using a tool called "SGML−Tools" which can be got from − "http://www.sgmltools.org" Compiling the source you will get the following commands like

- sgml2html xxxxhowto.sgml (to generate html file)
- sgml2html −split 0 xxxxhowto.sgml (to generate a single page html file)
- sgml2rtf xxxxhowto.sgml (to generate RTF file)
- sgml2latex xxxxhowto.sgml (to generate latex file)

7.1. Acrobat PDF format

PDF file can be generated from postscript file using either acrobat *distill* or *Ghostscript* . And postscript file is generated from DVI which in turn is generated from LaTex file. You can download distill software from "http://www.adobe.com" . Given below is a sample session:

```
        bash$ man sgml2latex
        bash$ sgml2latex filename.sgml
        bash$ man dvips
        bash$ dvips −o filename.ps filename.dvi
        bash$ distill filename.ps
        bash$ man ghostscript
        bash$ man ps2pdf
        bash$ ps2pdf input.ps output.pdf
        bash$ acroread output.pdf &
```

Or you can use Ghostscript command *ps2pdf* . ps2pdf is a work−alike for nearly all the functionality of Adobe's Acrobat Distiller product: it converts PostScript files to Portable Document Format (PDF) files. *ps2pdf* is implemented as a very small command script (batch file) that invokes Ghostscript, selecting a special "output device" called *pdfwrite* . In order to use ps2pdf, the pdfwrite device must be included in the makefile when Ghostscript was compiled; see the documentation on building Ghostscript for details.

7.2. Convert Linuxdoc to Docbook format

This document is written in linuxdoc SGML format. The Docbook SGML format supercedes the linuxdoc format and has lot more features than linuxdoc. The linuxdoc is very simple and is easy to use. To convert linuxdoc SGML file to Docbook SGML use the program *ld2db.sh* and some perl scripts. The ld2db output is not 100[percnt] clean and you need to use the *clean[lowbar]ld2db.pl* perl script. You may need to manually correct few lines in the document.

- Download ld2db program from "http://www.dcs.gla.ac.uk/~rrt/docbook.html" or from Milkyway Galaxy site
- Download the cleanup[lowbar]ld2db.pl perl script from from Milkyway Galaxy

The ld2db.sh is not 100[percnt] clean, you will get lots of errors when you run

```
                bash$ ld2db.sh file-linuxdoc.sgml db.sgml
                bash$ cleanup.pl db.sgml > db_clean.sgml
                bash$ gvim db_clean.sgml
                bash$ docbook2html db.sgml
```

And you may have to manually edit some of the minor errors after running the perl script. For e.g. you may need to put closing tag < /Para> for each < Listitem>

7.3. Convert to MS WinHelp format

You can convert the SGML howto document to Microsoft Windows Help file, first convert the sgml to html using:

```
                bash$ sgml2html xxxxhowto.sgml     (to generate html file)
                bash$ sgml2html -split 0   xxxxhowto.sgml (to generate a single page html file)
```

Then use the tool HtmlToHlp . You can also use sgml2rtf and then use the RTF files for generating winhelp files.

7.4. Reading various formats

In order to view the document in dvi format, use the xdvi program. The xdvi program is located in tetex−xdvi*.rpm package in Redhat Linux which can be located through ControlPanel [verbar] Applications [verbar] Publishing [verbar] TeX menu buttons. To read dvi document give the command −

```
        xdvi −geometry 80x90 howto.dvi man xdvi
```

And resize the window with mouse. To navigate use Arrow keys, Page Up, Page Down keys, also you can use 'f', 'd', 'u', 'c', 'l', 'r', 'p', 'n' letter keys to move up, down, center, next page, previous page etc. To turn off expert menu press 'x'.

You can read postscript file using the program 'gv' (ghostview) or 'ghostscript'. The ghostscript program is in ghostscript*.rpm package and gv program is in gv*.rpm package in Redhat Linux which can be located through ControlPanel [verbar] Applications [verbar] Graphics menu buttons. The gv program is much more user friendly than ghostscript. Also ghostscript and gv are available on other platforms like OS/2, Windows 95 and NT, you view this document even on those platforms.

- Get ghostscript for Windows 95, OS/2, and for all OSes from "http://www.cs.wisc.edu/~ghost"

To read postscript document give the command −

```
        gv howto.ps ghostscript howto.ps
```

You can read HTML format document using Netscape Navigator, Microsoft Internet explorer, Redhat Baron Web browser or any of the 10 other web browsers.

You can read the latex, LyX output using LyX a X−Windows front end to latex.

8. Copyright