Numdiff User Manual, version 5.0

“und der eignen Kraft vertrauend steigt ein frei Geschlecht empor”

This manual describes how to install and use Numdiff, a program which compares putatively similar files line by line and field by field, ignoring small numeric differences or/and different numeric formats.

Table of Contents

1 Copying

2 Acknowledgments

I want to thank Mr. Norman Clerman norm(dot)opcon(at)fuse(dot)net for several suggestions he gave me to improve the readability and the effectiveness of the output produced by Numdiff. Moreover, he pointed out the need to implement a filter to resynchronize the lines between two files in case of addition or deletion of one or more lines. I have to give him credit for the urge to prepare the version 4.x of Numdiff.

Moreover, I want to thank my friend Mariapia Palombaro since she removed some errors while reviewing the first version of this document.

3 Overview

Computer users often find occasion to ask how two files differ. Perhaps one file is a newer version of the other file. Or maybe the two files started out as identical copies but were changed by different people.

There are several ways to think about the differences between two files. One way to think of the differences is as a series of lines that were deleted from, inserted in, or changed in one file to produce the other file. The well-known `diff` program compares two files line by line, finds groups of lines that differ, and reports each group of differing lines. Without particular options, the `diff` program considers any change in the amount or in the type of the characters as a relevant difference. However, trough some command line options it also provides ways to suppress certain kinds of differences that are not important to the user. For instance, `diff` provides ways to ignore differences in the amount of white space between words or lines, or differences in alphabetic case.

Another way to think of the differences is as a series of words that were deleted from, inserted in, or changed in one file to produce the other file. Here "word" refers to a sequence of non white-space characters delimited by a couple of white-spaces, one before and the other one after the word.

The less known `wdiff` program by Franc,ois Pinard `<pinard@iro.umontreal.ca>` compares words in two files and reports the differences.

At last, one can think of the differences between two files as a sequence of pairs of bytes that can be either identical or different. The `cmp` program reports the differences between two files byte by byte, instead of line by line or word by word. As a result, it is often more useful than `diff` or `wdiff` for comparing binary files.

However, none of these approaches turns out to be good when you want to compare a couple of text files composed partially or entirely by numerical fields. Indeed, when you compare a couple of such files, what you want to obtain usually is a list of the numerical fields in the second file which **numerically** differ from the corresponding fields in the first file. But, as you probably knows, a same number can be written using different notations and programs like `diff` or `wdiff` can not recognize whether a difference between two numeric fields is only due to the notation or is actually a difference of numerical values.

For instance, 11.23 and 11.2300000 are the same number but represented in different ways. While, if you are interested in numerical values, it is obvious that the difference in the representation is not meaningful and then it should be ignored, however `diff` and `wdiff` consider the previous one as a relevant difference and there is no way for you to tell these programs to ignore it !

Another example of this type is given by 98765.4321 and 9.87654321E04 where the difference is only due to the use of the scientific notation in place of the common notation.

Moreover, depending on your country you could stick to different conventions in writing numbers. For instance, the amount "three hundred millions and fifty-two thousands of dollars and forty-six cents" is usually written by an Italian accountant as 300.052.000,46$ while an American accountant would write 300,052,000.46$. Of course, 300.052.000,46$ and 300,052,000.46$ represent the same amount of money but `diff` and `wdiff` would report a difference, which probably is not what you would in a similar case.

At last, sometimes you could want to ignore even differences in numerical values as long as they do not overcome a certain threshold. In other words, you could desire to suppress all "small" numerical differences too.

For instance, it could happen that you want to ignore all numerical differences whose absolute value is not greater than 0.0001. If this is the case, then the numerical fields 33 and 33.00009 must be considered equal, while 33 and 33.00011 must be reported as different.

However, `diff` and `wdiff` can not be used to ignore "small" numerical differences, since they do not even know what a numerical difference is.

What I have been saying till now explains why I decided to implement a new program with the capability to "appropriately" compare files containing numerical fields. In writing this program I was inspired by `ndiff`, a GPL'ed software by Nelson H. Baabe of the Salt Lake City University. The author of `ndiff` had the same good reasons as me to write `ndiff`. Actually `ndiff` is a good tool and I used it for a while. But I did not completely like the way it works and so `numdiff` was born. Although `ndiff` inspired `numdiff`, they are completely different from the viewpoint of the source code: `numdiff` has been entirely written from scratch.

`numdiff` can be used to compare putatively similar files line by line and field by field, ignoring small numeric differences or/and different numeric formats. `numdiff` takes two mandatory arguments, the paths of the two files to compare, and, after splitting them into lines and the lines into fields according to a given list of field delimiters, it compares every field of every line of the first file with the field of the second file at the same position (here position refers both to the line number and to the location within the line). If the compared fields are both legal numerical values, then `numdiff` performs a numerical comparison between them, else it performs a literal comparison, that is to say the usual byte by byte comparison. In case of literal comparison, two fields are regarded as equal if they are formed by the same sequence of characters. In case of numerical comparison and without specific command line options, two fields are regarded as equal if their numerical difference is zero. Be careful ! If you do not explicitly specify a list of field delimiters by the option '`-s`', then `numdiff` takes as field delimiters the characters newline ('`\n`', ASCII code 0x0A), horizontal tabulation ('`\t`', ASCII code 0x09), and blank (' ', ASCII code 0x20).

For instance, if the file '`list1`' contains the data

```
accident        123     23Joshua          34.55           +3+4i          water
dog       -3455.321     cat             2.345678e-9     .0005-6.23e2i
```

and file '`list2`' contains the data

```
Accident        123     23456          34.5500         +3.0001+4i
dog       -3455.320098  Cat     +2.345678e-9            -6.23e2i    $$$
A new line
```

then the output of the command '`numdiff list1 list2`' will be:

```
----------------
##1       #:1   <== accident
                ==> Accident
                                                      @
##1       #:3   <== 23Joshua
                ==> 23456
                                                      @
##1       #:5   <== +3+4i
                ==> +3.0001+4i
 Absolute error = 1.0000000000e-4, Relative error = 2.0000000000e-5
##1       #>6   <== water
                ==>
 Line 1 in file "list2" is shorter!
----------------
##2       #:2   <== -3455.321
                ==> -3455.320098
 Absolute error = 9.0200000000e-4, Relative error = 2.6104672633e-7
```

```
##2       #:3   <== cat
                ==> Cat
                                                        @
##2       #:5   <== .0005-6.23e2i
                ==> -6.23e2i
 Absolute error = 5.0000000000e-4, Relative error = 8.0256821830e-7
##2       #>6   <==
                ==> $$$
 Line 2 in file "list1" is shorter!
----------------
##3       #>1   <==
                ==> A new line
 Line 3 in file "list1" is shorter!
----------------
##4       <==
          ==>


+++  File "list1" differs from file "list2"
```

At the same time `numdiff` will print the following error message on stderr:

```
***  End of file "list1" reached
     Likely the files "list1" and "list2" do not have the same number of lines !
```

I have to remark that `numdiff` can recognize complex numbers, provided that they are written in the form $a + bi$ or $a - bi$ with no extra characters between the values a, b and the sign $+$ or $-$ (actually the symbol i used to represent the imaginary unit can be changed by a suitable command line option, but we shall see it in Chapter 5 [Invoking], page 11). If you do not know what complex numbers are, do not worry ! In this case probably you will never manage files containing complex numbers and so you can happily continue to ignore them. :) Even if the output of `numdiff` is self-explanatory, in the next section I will explain in details all you have to know about it.

3.1 Output format

Let us go back to the previous example. If the files '`list1`' and '`list2`' contain the data

```
accident        123     23Joshua          34.55           +3+4i          water
dog       -3455.321     cat             2.345678e-9     .0005-6.23e2i
```

and

```
Accident        123     23456          34.5500         +3.0001+4i
dog       -3455.320098  Cat     +2.345678e-9            -6.23e2i    $$$
A new line
```

respectively, then the output of the command '`numdiff list1 list2`' will be:

```
----------------
##1       #:1   <== accident
                ==> Accident
                                                        @
##1       #:3   <== 23Joshua
                ==> 23456
                                                        @
```

```
##1       #:5   <== +3+4i
                ==> +3.0001+4i
 Absolute error = 1.0000000000e-4, Relative error = 2.0000000000e-5
##1       #>6   <== water
                ==>
 Line 1 in file "list2" is shorter!
----------------
##2       #:2   <== -3455.321
                ==> -3455.320098
 Absolute error = 9.0200000000e-4, Relative error = 2.6104672633e-7
##2       #:3   <== cat
                ==> Cat
                                                          @
##2       #:5   <== .0005-6.23e2i
                ==> -6.23e2i
 Absolute error = 5.0000000000e-4, Relative error = 8.0256821830e-7
##2       #>6   <==
                ==> $$$
 Line 2 in file "list1" is shorter!
----------------
##3       #>1   <==
                ==> A new line
 Line 3 in file "list1" is shorter!
----------------
##4       <==
          ==>


+++  File "list1" differs from file "list2"
```

`numdiff` prints a report on the standard output for every field of the first file which differs from the corresponding field of the second one.

First this report indicates the location of the field, that is to say the number of the line where the field appears and its position within the line (this is "1" if it is the first field of its line, "2" if it is the second field of the line, "3" if it is the third one and so on. Fields are numerated starting from the left hand of the line and proceeding towards the right hand). For each report the line number is introduced by the symbol "##", while the field number by "#:". Then `numdiff` shows in what the difference consists. For instance,

```
##1       #:1   <== accident
                ==> Accident
                                                        @
```

means that the first field of the first line is "accident" in the first file, while in the second file it appears as "Accident". This difference could then be canceled by removing "accident" from the first file and inserting "Accident" in place of it. The arrows "<==" and "==>" try to visualize this idea. Analogously,

```
##2       #:2   <== -3455.321
                ==> -3455.320098
 Absolute error = 9.0200000000e-4, Relative error = 2.6104672633e-7
```

means that the second field of the second line is "-3455.321" in the first file and "-3455.320098" in the second one. Since the contents of the field are numerical in both files, `numdiff` also prints the absolute and relative errors.

The absolute error (or absolute difference) is given by the absolute value of the difference between the values appearing in the two files.

The relative error (or relative difference) is actually defined in a more complicated way. If "n1" is the value appearing in the first file and "n2" is the value in the second file, then the absolute error is given by the formula "A=|n1-n2|", while the relative error "R" is given by:

- "R = 0" if "n1" and "n2" are equal,
- "Inf" (infinity) if "n2" differs from "n1" and at least one of them is zero,
- "R = A/ min(|n1|, |n2|)" if "n1" and "n2" are both non zero and "n2" differs from "n1". "min(|n1|, |n2|)" denotes the minimum between the absolute value of "n1" and the absolute value of "n2".

I have to remark that, with these definitions of absolute and relative error it turns out that A(n2, n1) = A(n1, n2) and R(n2, n1) = R(n1, n2). In other words, the absolute/relative error does not change if you only change the order of the compared values.

If the contents of a field are, in at least one of the compared files, non-numerical, then the output line reporting absolute and relative errors is replaced by the separator:

```
@                                                    @@
```

It can happen that a line in one of the two files to compare contains more fields than the corresponding line of the other file. When this is the case, `numdiff` reports this difference by telling that a certain line (identified by its line number) appears to be shorter in one of the two files, just as in

```
##1       #>6   <== water
                 ==>
 Line 1 in file "list2" is shorter!
```

or in

```
##3       #>1   <==
                 ==> A new line
 Line 3 in file "list1" is shorter!
```

When this is the case, `numdiff` also shows the *tail* of the line as it appears in one of the compared files. Moreover, it uses the notation "#>n" to indicate the number n of the first field in the longer line for which there is no corresponding field in the shorter line. For instance,

```
##1       #>6   <== water
                 ==>
 Line 1 in file "list2" is shorter!
```

means that all the fields of the first line starting from the sixth one are empty in the second file ('`list2`'). In this context, the symbol `<<*>>` (when it appears) is used to denote the End-Of-File, i.e. a line or the tail of a line which is located at the end of the corresponding file and does not have a terminating **newline** character.

It can also happen that one of the two files to compare has less lines than the other one. In this case `numdiff` prints the number of the first line which compares in only one of the two files. Moreover, it prints on the standard error a message telling in which of the two files the end has been prematurely reached:

```
***  End of file "list1" reached
     Likely the files "list1" and "list2" do not have the same number of lines !
```

At last, `numdiff` prints on standard output a message reporting the final status of the comparison. This message says either the two files are equal or they are different, just as in the example we are considering:

```
+++  File "list1" differs from file "list2"
```

I have to remark that the user can make `numdiff` avoid to print, partially or totally, the messages that it would otherwise send to standard output. This can be achieved by some suitable command line options, see Chapter 5 [Invoking], page 11.

4 Installing

To successfully compile, build and install Numdiff some tools are required. The first one is an ANSI C compiler. This compiler should at least accept the option '`-o`' in order to place its output in a specified file. Numdiff has been successfully compiled and tested on Slackware GNU/Linux 10.2 with the version 3.3.6 of the GNU C Compiler (GCC), on Slackware GNU/Linux 11 with GCC 3.4.6, on Debian GNU/Linux 4.0 with GCC 4.1.2 20061115 (prerelease) (Debian 4.1.1-21; the installation on Debian required however a tricky modification of the standard procedure, see below for more details), and on SunOS 5.8 with the version 2.95.3 of the same compiler.

Moreover, you need a POSIX implementation of the `make` utility (I used both GNU make and smake by Joerg Schilling to compile Numdiff) and a POSIX implementation of the commands `rm` and `find`. At last, you need a proper installation of GNU Texinfo (in order to install the info documentation) and a shell sh-compatible.

Configuration, building and installation of Numdiff can be performed through the standard three steps:

```
./configure
make
make install
```

If you want to install Numdiff on Debian GNU/Linux, you may have to manually correct the Makefile before doing '`make install`': if you find the line

```
INSTINFO=
```

then you have to add `ginstall-info` at the end of this line to correctly define the variable `INSTINFO`

```
INSTINFO=ginstall-info
```

If you leave enabled the Natural Language Support and you also want to install the localization files (at the moment only the italian localization is supplied), then, after '`make`', you will have to type and run

```
make install-nls
```

By default, '`make install`' will install all the files in '`/usr/local/bin`', '`/usr/local/info`' etc. You can specify an installation prefix other than '`/usr/local`' using the option '`--prefix`' in the `configure` step, for instance '`--prefix=$HOME`':

```
./configure --prefix=$HOME
```

For better control, you can use the options '`--bindir`', '`--infodir`', and so on. Type '`./configure --help`' to obtain the complete list of all the available options.

Anyway, the documentation files, including a full User Manual available in several formats (HTML, PDF and plain ASCII text), will always be put in '*`DOCDIR`*`/numdiff`', where *DOCDIR* is the path specified by the option '`--docdir`' or, if this option has not been given to `configure`, '*`PREFIX`*`/local/doc`'. Here *PREFIX* is the installation prefix specified by the option '`--prefix`' or the default '`/usr/local`'.

Once Numdiff has been installed you can remove all the files previously installed by a simple '`make uninstall`'. If you have also installed the localization files trough '`make install-nls`', then, in order to remove also these ones, use '`make uninstall-nls`' in place of '`make uninstall`'.

Between the options accepted by `configure` there are '`--enable-debug`', '`--enable-optimization`', and '`--enable-nls`'.

The option '`--enable-debug`' turns on debugging when compiling the source code. This is obtained by passing to the compiler the '`-g`' option, but you can change this default debugging flag (which could not even be recognized by your compiler) by setting the environment variable `DBGFLAGS` before calling `configure`.

The option '`--enable-optimization`' turns on basic optimization when compiling the source code. This is obtained by passing to the compiler the '`-O`' option, but you can change this default flag (which could not even be recognized by your compiler) by setting the environment variable `OPTFLAGS` before calling `configure`.

The option '`--enable-nls`' turns on Natural Language Support. But you do not need to use it explicitly, since Natural Language Support is enabled by default. However, you can disable it by using '`--disable-nls`'.

5 Invoking

SYNOPSIS

```
numdiff -h|v
```

or

```
numdiff [-a maxerr][-r maxerr][-2][-P][-N][-s ifs][-b][-V][-q][-D][-E]
[-I][-S][-# prec][-d c1c2][-t c1c2][-g n1n2][-p c1c2][-n c1c2][-e c1c2]
[-i c1c2][-F f1-f2][-L l1-l2][-l path][-o path] file1 file2
```

where *file1* and *file2* are the names of the two files to compare.

In the first case `numdiff` prints a short help (actually not so short) or/and version number, Copyright, NO-Warranty disclaimer and some information about the way it was built. In the second case `numdiff` compares the files specified by the two (mandatory) arguments following the list of the options. The complete path of a file should be given, a directory name is not accepted. Moreover, the two arguments cannot refer to the same file but one of them can be `"-"`, which refers to stdin.

OPTIONS

- '`-a maxerr`' Specify the maximum absolute error permitted before that two numerical values are regarded as different (The default value is zero)
- '`-r maxerr`' Specify the maximum relative error permitted before that two numerical values are regarded as different (The default value is zero)
- '`-2`' Order that two numerical values are regarded as equal only if both absolute and relative error do not exceed the corresponding tolerance threshold (The default behavior is considering equal two numerical values if at least one between absolute and relative error does not exceed the corresponding tolerance threshold, or zero if no tolerance threshold has been specified)
- '`-P`' Ignore all differences due to numeric fields of the second file that are less than the corresponding numeric fields in the first file (consider only positive errors)
- '`-N`' Ignore all differences due to numeric fields of the second file that are greater than the corresponding numeric fields in the first file (consider only negative errors)
- '`-E`' While printing the differences between the two compared files show only the numerical ones
- '`-D`' While printing the differences between the two compared files neglect all the numerical ones
- '`-I`' Ignore changes in case while doing literal comparisons
- '`-s ifs`' Specify the set of characters to use to split the input lines into fields (The default set of characters is white space, tab and newline)
- '`-b`' Suppress all messages concerning the differences discovered in the structures of the two files
- '`-V`' For every line differing in at least one field print an header to show how this line appears in the two compared files
- '`-q`' Suppress all the standard output of the program
- '`-S`' Add some statistics to the standard output
- '`-# prec`' Specify the number of digits in the significands used in multiple-precision arithmetic
- '`-d c1c2`' Specify the characters representing the decimal point in the two files to compare

- '`-t c1c2`' Specify the characters representing the thousands separator in the two files to compare
- '`-g n1n2`' Specify the number of digits forming each group of thousands in the two files to compare
- '`-p c1c2`' Specify the (optional) prefixes for positive values used in the two files to compare
- '`-n c1c2`' Specify the prefixes for negative values used in the two files to compare
- '`-e c1c2`' Specify the exponent letters used in the two files to compare
- '`-i c1c2`' Specify the characters representing the imaginary unit in the two files to compare
- '`-F f1-f2`' Select the fields that have to be compared (The default behavior is comparing all the fields)
- '`-L l1-l2`' Select the lines whose fields have to be compared (The default behavior is comparing all the fields in all the lines)
- '`-l path`' Redirect warning and error messages from stderr to the indicated file
- '`-o path`' Redirect output from stdout to the indicated file
- '`-h`' Show help message and predefined settings
- '`-v`' Show version number, Copyright and NO-Warranty

DIAGNOSTICS

The exit status is 1 if the two given files differ, 0 if they are equal, -1 (255) in case of error.

DEFAULT NUMERIC FORMAT (for both files to compare):

Decimal point = '.'

Thousands separator = ','

Number of digits in each thousands group = 3

Positive sign = '+'

Negative sign = '-'

Prefix for decimal exponent = 'e'

Symbol used to denote the imaginary unit = 'i'

SOME EXPLANATIONS

The options '`-D`', '`-E`', '`-b`' and '`-q`' are used to hide part of the standard output of the program according to some rule.

The option '`-D`' triggers the "dummy mode". In this mode `numdiff` does not print the numerical differences. A numerical difference occurs when a given field turns out to be of numerical type in both files to compare, but it has in the second file a value differing from the one contained in the first file. The "dummy mode" is so called since when it is active, `numdiff` does not perform the job for which I created it.

The option '`-E`' triggers the "essential mode". In this mode `numdiff` only prints the numerical differences between the two files and, if there are some, the differences in the structure, which occur either when a line of text comes out to be formed by a different number of fields in the two files to compare or when the two files have a different number of lines.

The option '`-b`' triggers the "brief mode". In this mode `numdiff` does not print the differences in the structure of the two files, i.e. the ones consisting in a different number of fields on a line, or in a different number of lines in the two files.

Finally, the option '`-q`' triggers the "quiet mode". When in this mode `numdiff` does not print anything on the standard output. The "quiet mode" is useful if you only want to know whether a couple of files are equal or not. This information can be obtained by looking at the exit status of the program.

The option '`-V`' triggers the "verbose mode". In this mode `numdiff` produces a richer report by printing an header for every line differing in at least one field in order to show how this line appears in the two compared files. For instance, if the files '`data1`' and '`data2`' contain the data

```
12      33
22      44.5
0.008   1.002
221.12  -34.56  water
2101.21 boats
```

and

```
12      33
22.3      44.5
0.008   1.202
221.12  -34.56
2101.21 boats   dogs
```

respectively, then the command '`numdiff -V data1 data2`' will print the following output:

```
----------------
##2       <== 22      44.5
          ==> 22.3      44.5

##2       #:1   <== 22
                ==> 22.3
 Absolute error = 3.0000000000e-1, Relative error = 1.3636363636e-2
----------------
##3       <== 0.008   1.002
          ==> 0.008   1.202

##3       #:2   <== 1.002
                ==> 1.202
 Absolute error = 2.0000000000e-1, Relative error = 1.9960079840e-1
----------------
##4       <== 221.12  -34.56  water
          ==> 221.12  -34.56

##4       #>3   <== water
                ==>
 Line 4 in file "data2" is shorter!
----------------
##5       <== 2101.21 boats
          ==> 2101.21 boats   dogs

##5       #>3   <==
                ==> dogs
 Line 5 in file "data1" is shorter!

+++  File "data1" differs from file "data2"
```

You must care that the options '`-b`' and '`-V`' will be overridden if '`-q`' is also set.

Moreover, the amount of additional information printed by the option '`-V`' is trivially influenced by the options eventually set between the ones altering the way `numdiff` performs the comparisons between fields (for instance '`-a`', '`-r`', '`-2`', '`-N`', '`-P`', '`-D`', '`-E`', '`-I`', '`-F`'). In the headers printed by `numdiff` when in "verbose mode" can also appear the symbol `<<*>>`. This symbol, if present, is always located at the end of a line to mean that the line is at the end of the corresponding file and does not have a terminating **newline** character.

The option '`-S`' adds to the standard output of `numdiff` the following information:

- the largest absolute error in the set of relevant numerical differences and the corresponding relative error, and
- the largest relative error in the set of relevant numerical differences together with the corresponding absolute error.

For relevant numerical differences I mean those ones appearing in the output of `numdiff` when the options '`-D`' and '`-q`' are not used. The information printed by the option '`-S`' is not removed when this option is used together with '`-q`'.

The options '`-a`', '`-r`', '`-2`', '`-P`' and '`-N`' affect the way `numdiff` performs the comparisons between numerical values. Without any of these options, `numdiff` considers two numerical fields as equal when their difference is zero.

The option '`-a`' can be used to specify that two numerical fields must be considered equal as long as their absolute difference does not exceed a given threshold, which is supplied by the argument following the '`-a`' option.

The option '`-r`' can be used to specify that two numerical fields must be considered equal as long as their relative difference does not exceed a given threshold, which is supplied by the argument following the '`-r`' option.

The option '`-2`' is only meaningful when both '`-a`' and '`-r`' are present on the command line. If the user specifies a non-zero tolerance threshold for both absolute and relative error by using both '`-a`' and '`-r`', `numdiff` adopts this behavior: it considers equal two numerical fields as long as at least one between absolute and relative error does not exceed the corresponding threshold. With the option '`-2`' `numdiff` regards two numerical values as equal only if both absolute and relative error do not exceed the threshold of tolerance. For instance, if *file1* contains the unique line

```
100
```

and *file2* the line

```
100.00012
```

then the output of the command '`numdiff` *file1 file2*' will be

```
----------------
##1       #:1   <== 100
               ==> 100.00012
@ Absolute error = 1.2000000000e-4, Relative error = 1.2000000000e-6

+++  File "file1" differs from file "file2"
```

the output of the commands '`numdiff -a 1.0e-4` *file1 file2*' and '`numdiff -r 1.0e-6` *file1 file2*' will be the same, while '`numdiff -a 1.0e-4 -r 1.3e-6` *file1 file2*' and '`numdiff -a 1.3e-4 -r 1.0e-6` *file1 file2*' will print the message

```
+++  Files "file1" and "file2" are equal
```

since the actual relative error is 1.2e-6 < 1.3e-6, the actual absolute error is 1.2e-4 < 1.3e-4, and it is sufficient that one of them does not exceed the tolerance specified on the command line to make `numdiff` consider equal the two compared values. However, the commands '`numdiff -a`

`1.0e-4 -r 1.3e-6 -2 file1 file2`' and '`numdiff -a 1.3e-4 -r 1.0e-6 -2 file1 file2`' will print the message

```
----------------
##1       #:1   <== 100
               ==> 100.00012
@ Absolute error = 1.2000000000e-4, Relative error = 1.2000000000e-6

+++  File "file1" differs from file "file2"
```

since the option '`-2`' makes `numdiff` regard two values as equal only if both absolute and relative difference do not exceed the corresponding threshold of tolerance.

The option '`-P`' makes `numdiff` consider two values equal whenever the second one, i.e. the one coming from the file specified as last on the command line, is less or equal than the first one, which is the value coming from the file specified as first on the command line. If the values to compare are complex numbers, saying that the second one is less or equal than the first one means that both real and imaginary part of the second one are not greater than the real part and, respectively, the imaginary part of the first one.

Finally, the option '`-N`' makes `numdiff` consider two values equal whenever the second one, i.e. the one coming from the file specified as last on the command line, is greater or equal than the first one, which is the value coming from the file specified as first on the command line. If the values to compare are complex numbers, saying that the second one is greater or equal than the first one means that both real and imaginary part of the second one are not less than the real part and, respectively, the imaginary part of the first one.

The options '`-I`', '`-l`', '`-o`', '`-h`' and '`-v`' do not require further explanations. The options '`-l`' and '`-o`' are only supplied for the users of some poorly designed operating systems (like MSDog or MSWindoze), whose default shell does not allow the redirection of standard error and standard output. The option '`-I`' has no effect on the outcome of numerical comparisons.

Several things must be told about the option '`-s`'. First, it will be better if you will always quote the set of the delimiters, just as in the next examples:

```
numdiff -s ' \t\n,;:.' file1 file2
numdiff -s ' \t\n\r\f\v"\:;' file1 file2
numdiff -s `` \t\n''' file1 file2
```

I recommend you actually to always use the single quote character (') to enclose the list of the delimiters, since in this way you will prevent any substitution or handling of characters by the shell.

`numdiff` recognizes and interprets the following sequences of characters within the argument passed to the option '`-s`':

- '`\f`' form feed
- '`\n`' newline
- '`\r`' carriage return
- '`\t`' horizontal tab
- '`\v`' vertical tab

since these characters are often used as delimiters in files containing numerical data and they could not be included directly in the set of delimiters. Then, by passing the string ' `\t\n,;:.`' as argument for the option '`-s`', one tells `numdiff` use as field delimiters the characters **blank**, **horizontal tab**, **newline**, **comma**, **semicolon**, **colon** and **dot**. Passing ' `\t\n`' as argument to the option '`-s`' is the same as not using at all the option '`-s`', since **blank**, **horizontal tab** and **newline** are the default field delimiters. In the list of field delimiters the character **backslash** ('\') is always treated in a special way. If followed by '`f`', '`n`', '`r`', '`t`' and '`v`' it is combined with the

subsequent character and interpreted in the way we have just seen. Otherwise, the **backslash** is coupled with the following character and then removed. In particular, if you want to specify the **backslash** itself as field delimiter, you have to put **two backslashes** (‘\\’) in the list of delimiters. Therefore, the delimiters specified by the command line

```
numdiff -s' \t\n\\\"' file1 file2
```

are **blank**, **horizontal tab**, **newline**, **backslash** and **double quote** since ‘\\’ and ‘\"’ are interpreted by `numdiff` as ‘\’ and ‘"’.

Even if I have recommended to enclose the set of delimiters in single quotes, there are cases in which you will be constrained to use the double quote character (‘"’) to enclose the set of field delimiters, just as in one of the previous examples. However you must take into account that in this case the shell could make some substitutions on the command line before executing `numdiff`. For instance, if your shell is GNU bash, then (citing the man page of GNU bash)

> Enclosing characters in double quotes preserves the literal value of all characters within the quotes, with the exception of ‘$’, ‘`’, and ‘\’. The characters ‘$’ and ‘`’ retain their special meaning within double quotes. The backslash retains its special meaning only when followed by one of the following characters: ‘$’, ‘`’, ‘"’, ‘\’, or **<newline>**. A double quote may be quoted within double quotes by preceding it with a backslash . . . The special parameters * and `@` have special meaning when in double quotes . . .

Therefore, if the set of delimiters is formed by ‘ ’, ‘\t’, ‘\n’, ‘\’ and ‘"’ and you decide to enclose them in double quotes, then the `numdiff` command line should be

```
numdiff -s'' \t\n\\\\\"'' file1 file2
```

and not

```
numdiff -s'' \t\n\\\"'' file1 file2
```

Indeed, in this last case the shell would replace the string

‘ \t\n\\\"’

with

‘ \t\n\"’

and then `numdiff` would take ‘ ’, ‘\t’, ‘\n’ and ‘"’ as field delimiters.

A last advice about the use of the option ‘`-s`’. I recommend you to always put **newline** (on Unix© and Unix-like operating systems, like GNU©) and **carriage return** (on MS-Dog/MSWindoze) in the set of field delimiters. Otherwise, these characters would be included in all the fields staying at the end of a line and this would cause some undesirable effects. For instance, a number put at the end of a line would not be considered as a numerical field by `numdiff`, since `numdiff` would consider the final **newline** as part of the field which then would be qualified as non-numerical. Maybe in the future I will modify `numdiff` in order to remove the mandatory specification of **newline** as field delimiter.

The option ‘`-#`’ lets the user specify the number of digits in the significands used in multiple-precision arithmetic. The default value is 35, the largest admissible value is 180. Take into account that an higher precision makes the execution of `numdiff` slower. This is particularly true when the files to compare contain a lot of numerical fields. Moreover, you have to care that `numdiff` can truncate the value of a numerical field if it has *too much* digits with respect to the current precision. To be precise, denoted by P the current value of the precision:

- If the number is written in common notation, then `numdiff` will consider, in addition to all the digits of the integer part, only the first P digits of the fractional part.
- If the value is written in scientific notation, then `numdiff` will only consider the first P digits of the fractional part of the mantissa.

The options '`-d`', '`-t`', '`-g`', '`-p`', '`-n`', '`-e`' and '`-i`' can be used to instruct `numdiff` about the numeric formats used in the files which it is going to compare. The two files to compare do not have to adopt the same numeric format, and then `numdiff` allows to specify different numeric formats for them. Indeed each of the options '`-d`', '`-t`', '`-g`', '`-p`', '`-n`', '`-e`' and '`-i`' can have as argument one or two characters (one or two digits if the option is '`-g`'). In the first case, the argument refers to both files to compare, in the second one the first character is for the file specified as first on the command line, the second character is for the file specified as last one on the command line. For instance, the option '`-d`' can be used to tell `numdiff` which character(s) is(are) used to mean the decimal point in the two files to compare. If you give the command '`numdiff -d_ file1 file2`', then `numdiff` will understand that both in *file1* and *file2* the character **underscore** ('`_`') is used in place of the default one ('`.`') to indicate the position of the decimal point in the numerical values. But if the command is '`numdiff -d_: file1 file2`', then `numdiff` will understand that the decimal point is indicated by the character **underscore** in *file1*, and by **colon** ('`:`') in *file2*.

Naturally, if you omit to use one of the options '`-d`', '`-t`', '`-g`', '`-p`', '`-n`', '`-e`' and '`-i`', then the corresponding attribute will take its default value, see [Default Numeric Format], page 12.

You must be really careful when you use one or more of these options. First, not all characters can be passed to them as arguments. For instance, the arguments of the option '`-g`' must be digits. The arguments of the options '`-d`' and '`-t`' must be punctuation marks (the punctuation marks are all the characters of the ASCII set for which the standard C function `ispunct` returns a non zero value), those ones of the options '`-p`', '`-n`', '`-e`' and '`-i`' must be graphical characters but digits (graphical characters are all the characters of the ASCII set for which the standard C function `isgraph` returns a non zero value).

Moreover, it is not possible to set the decimal point, the thousands separator, the positive sign, the negative sign, the prefix for decimal exponent and the symbol of the imaginary unit so that, for a same file, two or more of these characters come out to be equal. This rule also applies when you miss to explicitly set one of the previous arguments through the appropriate option. For instance, the command '`numdiff -d,. file1 file2`' will make `numdiff` abnormally terminate after printing the error message:

```
The numeric format specified for the first file is illegal,
the following symbols should be all different
while two or more of them are actually equal:

Decimal point = ‘,’
Thousands separator = ‘,’
Positive sign = ‘+’
Negative sign = ‘-’
Prefix for decimal exponent = ‘e’
Symbol used to denote the imaginary unit = ‘i’
```

Indeed, through the option '`-d`' we have told to `numdiff` that in the first file the decimal point is indicated by the character **comma**, but at the same time we have not modified the character in use to separate the groups of thousands, which has remained the default one, i.e. **comma**, for both files to compare. In this way, we have implicitly told to `numdiff` that in *file1* the character **comma** represents both decimal point and thousands separator. Since this is not reasonable, `numdiff` refuses to work. To avoid this problem it would be sufficient to explicitly notify to `numdiff` the thousands separator through the option '`-t`': '`numdiff -d,. -t., file1 file2`'. Of course, here we are supposing that the decimal point and the thousands separator are represented in *file1* by **comma** and **dot** respectively, in *file2* by **dot** and **comma**. I strongly suggest you that in writing a file you avoid the use of the same symbol to mean two different things (like would be using **comma** for both decimal point and thousands separator): it would be dummy, would not it ? :)

At last, it is possible (but it is stupid) to use as argument for the options '`-d`', '`-t`', '`-g`', '`-p`', '`-n`', '`-e`' and '`-i`' one of the characters used to separate the fields in the files to compare. In a such case `numdiff` does not complain but you have to consider that first it uses the set of field delimiters in order to split the files into fields and then it takes into account the numeric formats specified for the two files when it has to distinguish between numerical and non-numerical fields. However, it should never happen to specify as argument for one of the options '`-d`', '`-t`', '`-g`', '`-p`', '`-n`', '`-e`' and '`-i`' a character which is also used as field delimiter: again, in writing a file you should avoid (and people usually avoid it) to use the same symbol to mean two different things.

The options '`-L`' and '`-F`' can be used to restrict the comparison between files to a certain range of lines or to a certain group of fields. Without these options `numdiff` compares all the fields of all the lines.

With the option '`-L`' the user can make `numdiff` restrict the comparison to a certain line or to a certain range of lines. For instance '`numdiff -L 5 file1 file2`' will make `numdiff` compare only the fields in the fifth line of *file1* with the corresponding fields in the fifth line of *file2*.

Similarly '`numdiff -L 5-10 file1 file2`' will make `numdiff` compare only the fields which are contained in the lines from 5 to 10.

As you can see, you can specify a range of lines by using the notation '*n1*`-`*n2*', where *n1* and *n2* are the line numbers of the first and of the last line in the range of lines that you want to compare.

If you use two or more times the option '`-L`' with different specifications, then `numdiff` will only consider the last specification. Therefore '`numdiff -L 5-10 -L 6 -L 10-20 file1 file2`' will make `numdiff` compare the fields in the lines from 10 to 20.

With the option '`-F`' the user can make `numdiff` restrict the comparison to a certain field or to a certain group of fields. For instance '`numdiff -F 3 file1 file2`' will make `numdiff` compare only the third field of each line of *file1* with the third field of the corresponding line of *file2*.

Similarly '`numdiff -F 3-7 file1 file2`' will make `numdiff` compare, for every line in *file1* and *file2*, only the fields from the third one to the seventh one (both included).

As you can see, you can specify a range of fields by using the notation '*n1*`-`*n2*', where *n1* and *n2* are the field numbers of the first and of the last field in the range of fields that you want to compare.

Remember that the fields of a line are numerated starting from the left hand of the line and proceeding towards the right hand.

By using two or more times the option '`-F`' you can extend the comparison to a group of fields formed by more ranges and/or single fields. For instance '`numdiff -F 5-10 -F 6 -F 12-20 -F 4 file1 file2`' will make `numdiff` compare the fourth field, the fields from the 5th one to the 10th one and the fields from the 12th one to the 20th one of every line. Care the difference with respect to the option '`-L`', since, as I told above, if you use two or more times the option '`-L`' only the last specification will be considered by `numdiff`.

Moreover, take into account that the largest field number that you can use while writing a specification for the option '`-F`' is 32768.

Of course, you can use the options '`-F`' and '`-L`' together. In this way you can restrict the comparison to a certain group of fields within a certain range of lines.

If you use the option '`-F`' and/or the option '`-L`' the exit status of `numdiff` will reflect the outcome of the restricted comparison. For instance, the exit status of '`numdiff -L 1-7 file1 file2`' will be 1 only if `numdiff` will have found a difference in the first seven lines of *file1* and *file2*. If the two files differ only in the lines after the seventh one, then `numdiff` will end with a zero exit status.

6 Tools

Together with the version 5.x of Numdiff are shipped a couple of useful tools, `ndselect` and `ndfilter`. These tools are provided because Numdiff lacks the capability to resynchronize the lines between two files in case of addition or deletion of one or more lines. When I wrote down the first stable version of Numdiff, I wanted just to create a program able to perform numerical comparisons between files having the same structure, i.e. same number of lines and, on each line, same number and type of fields. Then I added the capability to restrict the comparison to a given range of lines (through the option '`-L`') or to some specific fields (through the option '`-F`'). However, even with these additions, Numdiff is not able to deal with some important issues related to the comparison between files.

The first tool shipped with Numdiff is `ndselect`. I decided to create this utility in order to deal with a situation that comes out often in Numerical Analysis. Here I present a very simple example of such a situation. Let us suppose that file *list1* contains the values of the square root, rounded to the 20th decimal digit, for all the integer numbers between 10 and 20:

```
12      3.46410161513775458705
13      3.60555127546398929312
14      3.74165738677394138558
15      3.87298334620741688518
16      4
17      4.12310562561766054982
18      4.24264068711928514641
19      4.35889894354067355224
20      4.47213595499957939282
21      4.58257569495584000659
22      4.69041575982342955457
23      4.79583152331271954l6
24      4.89897948556635619639
```

while *list2* contains *suitable* approximations of the square root only for the numbers between 12 and 21 which are multiple of 3:

```
12      3.46410162002945508100
15      3.87298387096774193548
18      4.24264705882352941176
21      4.58260869565217391304
```

These approximations could have been obtained for instance by using the famous Heron's algorithm, which, given an approximation `a` for the square root of a number `x`, computes a better approximation by the formula `a := 0.5 * (x/a + a)`. What we want now is to understand how good are the approximations contained in file *list2* using `numdiff`. Of course, we cannot execute directly the command '`numdiff list1 list2`', since in this way we would compare the approximations provided for the square roots of 15, 18, and 21 with the square roots of 13, 14, and 15 respectively. To make the comparison in the right way, one could open *list1* in a text editor and remove from it all the non interesting lines till to leave only the ones related to the numbers 12, 15, 18, and 21. But this approach is practicable because we have to remove few lines, and one can easily figure out how boring and inefficient would be to manually remove hundreds or thousands of lines from a file in order to compare it with another one.

An expert GNU/Linux© user would suggest that it is possible to make this removal automatic by the well known utilities `head` and `sed`, in this particular case '`head -n 10 list1 | sed -n -e '1~3 p' > List1`' A quick explanation for the ones who do not know how to use `head` and `sed`: the previous command extracts from *list1* the first 10 lines, namely the ones containing the square roots of the numbers from 12 to 21, then it picks every third line starting from the

first one between the extracted lines, in order to select only the ones related to 12, 15, 18, and 21. Finally, these lines are printed on the file *List1*, which then looks like:

```
12      3.46410161513775458705
15      3.87298334620741688518
18      4.24264068711928514641
21      4.58257569495584000659
```

Once obtained *List1*, we can perform with `numdiff` the comparison between the values we are interested in: '`numdiff` *`List1 list2`*' . Unfortunately, this trick only works if you have installed the GNU version of `sed`, which, as far as I know, is the only one to provide the extension *first~step* to specify line addresses. That is way I decided to implement `ndselect`, which allows to obtain the same result as before with the simpler command: '`ndselect -b 1 -e 10 -s 3` *`list1 > List1`*'

The meaning of the arguments passed to the options '`-b`', '`-e`', and '`-s`' is the following: we are telling `ndselect` to print every third line of file *list1* (the option '`-s`' specifies the step), starting from the first one (the option '`-b`' specifies the begin) and ending within the tenth one possibly inclusive (the option '`-e`' specifies the end). Because of the presence of the redirection operator `>`, the previous command sends to the file *List1* what `ndselect` would print on the screen (standard output).

Using `ndselect` is very simple and elegant, other than extremely useful in the situations just outlined. A complete description of its use follows.

SYNOPSIS

```
ndselect -h|v
```

or

```
ndselect [-b FIRST][-e LAST][-s STEP][-l path][-o path] [file]
```

where *file* is the name of the file to scan.

In the first case `ndselect` prints a short help or/and version number, Copyright and NO-Warranty disclaimer. In the second case `ndselect` prints on the standard output every STEPth line of *file* starting with line FIRST and ending within line LAST. The complete path of *file* should be given, a directory name is not accepted. If no input file is specified, the program reads from the standard input.

OPTIONS

- '`-b FIRST`' Specify the first line to print (The default behavior is to start with line number 1)
- '`-e LAST`' Specify the last line that can be printed (The default behavior is to arrive till to the end of the file)
- '`-s STEP`' Specify the step to use when selecting the lines to print (The default value for the step is 1)
- '`-l path`' Redirect warning and error messages from stderr to the indicated file
- '`-o path`' Redirect output from stdout to the indicated file
- '`-h`' Show this help message
- '`-v`' Show version number, Copyright and NO-Warranty

Passing 0 as argument to the option '`-e`' is equivalent to omit this option and leave enabled the default behavior (which consists in scanning till to the end of the file).

DIAGNOSTICS

The exit status is 0 in case of normal termination, -1 (255) in case of error.

Numdiff does not work well if one of the two files to compare contains in the middle some lines more or less than the other one. For instance, if you have one file that is 1000 lines long that you are comparing to a second file 1001 lines long, and except for that one extra line, located, let us say, at line 500, the files are identical, then `numdiff` will **not** show only the one line difference: once the files are out of synchronization `numdiff` reports every line as different. Although the ultimate solution would be adding to Numdiff the capabilities of the `diff` command, since I do not have time to do that for the moment, I decided to supply a tool, called `ndfilter`, which can turn out useful in similar cases. A simple example will explain how to employ `ndfilter`. Let us suppose that *file1* and *file2* have the following contents,

`file1:`

```
 Day            Expenses
-------------------------
  1             $  233.56
  2             $  850.77
  3             $   12.55
  4             $  524.00
  5             $   78.25
 10             $  230.00
 11             $  443.10
 12             $   67.65
 13             $   10.00
 14             $  201.45
 15             $  110.00
-------------------------
Total           $ 2761.33
```

and

`file2:`

```
 Day            Expenses
  1             $  233.56
  2             $  850.77
  3             $   12.55
  4             $  524.00
  5             $   78.25
  6             $  432.90
  7             $   60.88
  8             $  759.00
  9             $  322.54
 10             $  230.00
 11             $  443.10
 12             $   67.65
 13             $   10.00
 14             $  201.45
-------------------------
Total           $ 4226.65
```

The differences between *file1* and *file2* are given by

- the insertion of the separator `-------------------------` in *file1* before the list of the days,
- the insertion in *file2* of the lines related to the expenses for the days 6,7,8, and 9,

- the addition in *file1* of the entry for the day 15, and
- the different values for the total sum of the expenses.

However, if you compare directly *file1* and *file2* using `numdiff`, then the result will be completely misleading:

```
----------------
##1       #:1   <== file1:
               ==> file2:
                                                      @
----------------
##4       #:1   <== -------------------------
               ==> 1
                                                      @
##4       #>2   <==
               ==> $  233.56
 Line 4 in file "file1" is shorter!
----------------
##5       #:1   <== 1
               ==> 2
 Absolute error = 1.0000000000e+0, Relative error = 1.0000000000e+0
##5       #:3   <== 233.56
               ==> 850.77
 Absolute error = 6.1721000000e+2, Relative error = 2.6426185991e+0
----------------
##6       #:1   <== 2
               ==> 3
 Absolute error = 1.0000000000e+0, Relative error = 5.0000000000e-1
##6       #:3   <== 850.77
               ==> 12.55
 Absolute error = 8.3822000000e+2, Relative error = 6.6790438247e+1
----------------
##7       #:1   <== 3
               ==> 4
 Absolute error = 1.0000000000e+0, Relative error = 3.3333333333e-1
##7       #:3   <== 12.55
               ==> 524.00
 Absolute error = 5.1145000000e+2, Relative error = 4.0752988048e+1
----------------
##8       #:1   <== 4
               ==> 5
 Absolute error = 1.0000000000e+0, Relative error = 2.5000000000e-1
##8       #:3   <== 524.00
               ==> 78.25
 Absolute error = 4.4575000000e+2, Relative error = 5.6964856230e+0
----------------
##9       #:1   <== 5
               ==> 6
 Absolute error = 1.0000000000e+0, Relative error = 2.0000000000e-1
##9       #:3   <== 78.25
               ==> 432.90
 Absolute error = 3.5465000000e+2, Relative error = 4.5322683706e+0
----------------
```

```
##10      #:1   <== 10
               ==> 7
 Absolute error = 3.0000000000e+0, Relative error = 4.2857142857e-1
##10      #:3   <== 230.00
               ==> 60.88
 Absolute error = 1.6912000000e+2, Relative error = 2.7779237845e+0
----------------
##11      #:1   <== 11
               ==> 8
 Absolute error = 3.0000000000e+0, Relative error = 3.7500000000e-1
##11      #:3   <== 443.10
               ==> 759.00
 Absolute error = 3.1590000000e+2, Relative error = 7.1293161814e-1
----------------
##12      #:1   <== 12
               ==> 9
 Absolute error = 3.0000000000e+0, Relative error = 3.3333333333e-1
##12      #:3   <== 67.65
               ==> 322.54
 Absolute error = 2.5489000000e+2, Relative error = 3.7677753141e+0
----------------
##13      #:1   <== 13
               ==> 10
 Absolute error = 3.0000000000e+0, Relative error = 3.0000000000e-1
##13      #:3   <== 10.00
               ==> 230.00
 Absolute error = 2.2000000000e+2, Relative error = 2.2000000000e+1
----------------
##14      #:1   <== 14
               ==> 11
 Absolute error = 3.0000000000e+0, Relative error = 2.7272727273e-1
##14      #:3   <== 201.45
               ==> 443.10
 Absolute error = 2.4165000000e+2, Relative error = 1.1995532390e+0
----------------
##15      #:1   <== 15
               ==> 12
 Absolute error = 3.0000000000e+0, Relative error = 2.5000000000e-1
##15      #:3   <== 110.00
               ==> 67.65
 Absolute error = 4.2350000000e+1, Relative error = 6.2601626016e-1
----------------
##16      #:1   <== -------------------------
               ==> 13
                                                    @
##16      #>2   <==
               ==> $   10.00
 Line 16 in file "file1" is shorter!
----------------
##17      #:1   <== Total
               ==> 14
                                                    @
```

```
##17      #:3   <== 2761.33
               ==> 201.45
 Absolute error = 2.5598800000e+3, Relative error = 1.2707272276e+1
----------------
##18      <==
          ==> -------------------------


***  End of file "file1" reached
     Likely the files "file1" and "file2" do not have the same number of lines !

+++  File "file1" differs from file "file2"
```

The reason for that is the inability of `numdiff` to deal correctly with the insertion and the deletion of one or more lines in the files being compared. `ndfilter` can be very useful in similar situations. It takes two mandatory arguments, the names of the files to compare, and, after comparing the given files by recalling the standard Unix© utility `sdiff`, it uses the results of this comparison to create a new couple of files. Each of these new files is identical to the corresponding original, except for the eventual addition of one or more empty lines. Namely, the command '`ndfilter` *`file1 file2`*' will create two new files in the current working directory named respectively '*`file1`*`.new`' and '*`file2`*`.new`'. The file '*`file1`*`.new`' is identical to *file1* with the eventual exception of one or more empty lines, which are exactly put to compensate the presence of lines in *file2* which do not have a correspondence in *file1*. Analogously, '*`file2`*`.new`' is identical to *file2* if we except the empty lines that could have been inserted to compensate the presence in *file1* of lines not having a correspondence in *file2*.

To be more precise, `ndfilter` uses the output of the command '`sdiff` *`file1 file2`*' to establish if and which lines were deleted from, inserted in, or changed in one file to produce the other file. Then `ndfilter` knows for every line of *file1* and *file2* if it is present only in *file1*, or only in *file2*, or in both files maybe with some changes. The files '*`file1`*`.new`' and '*`file2`*`.new`' are created using this information according to the following rules. `ndfilter` reads line by line the output of '`sdiff` *`file1 file2`*' and if a line is marked as present both in *file1* and in *file2*, then it is copied both in '*`file1`*`.new`' and in '*`file2`*`.new`'. If there are changes between the *version* in *file1* and the one in *file2*, then `ndfilter` prints on '*`file1`*`.new`' the version coming from *file1* and on '*`file2`*`.new`' the one coming from *file2*. If a line is present in *file1* but has no correspondence in *file2*, then `ndfilter` adds this line to '*`file1`*`.new`' and at the same time puts an empty line in '*`file2`*`.new`'. On the other hand, if a line turns out to be present only in *file2*, then `ndfilter` adds this line to '*`file2`*`.new`' and an empty line to '*`file1`*`.new`'.

The new files created by `ndfilter` have the same number of lines and through the addition of empty lines they are often correctly synchronized to be compared by `numdiff`.

If *file1* and *file2* are the same shown above, then the output of '`sdiff` *`file1 file2`*' is (I used GNU sdiff from GNU diffutils 2.8.1 to obtain it)

```
file1:                                | file2:

 Day            Expenses                Day            Expenses
-------------------------             <
  1             $  233.56                 1             $  233.56
  2             $  850.77                 2             $  850.77
  3             $   12.55                 3             $   12.55
  4             $  524.00                 4             $  524.00
  5             $   78.25                 5             $   78.25
                                        >   6             $  432.90
                                        >   7             $   60.88
```

```
                                    >   8             $  759.00
                                    >   9             $  322.54
 10             $  230.00              10             $  230.00
 11             $  443.10              11             $  443.10
 12             $   67.65              12             $   67.65
 13             $   10.00              13             $   10.00
 14             $  201.45              14             $  201.45
 15             $  110.00           <
-------------------------             -------------------------
Total           $ 2761.33           | Total           $ 4226.65
```

The special symbols `<`, `>`, and `|` in the *gutter* column are used to mark the lines which are present only in *file1*, or only in *file2*, or in both files but with changes. When no symbol is present, then the corresponding line appears with no changes both in *file1* and in *file2*. According to the output of '`sdiff` *`file1 file2`*', the files '*`file1`*`.new`' and '*`file2`*`.new`' created by '`ndfilter` *`file1 file2`*' have the following contents (side by side view):

```
file1:                          #    file2:
                                #
 Day            Expenses        #     Day            Expenses
-------------------------       #
  1             $  233.56       #      1             $  233.56
  2             $  850.77       #      2             $  850.77
  3             $   12.55       #      3             $   12.55
  4             $  524.00       #      4             $  524.00
  5             $   78.25       #      5             $   78.25
                                #      6             $  432.90
                                #      7             $   60.88
                                #      8             $  759.00
                                #      9             $  322.54
 10             $  230.00       #     10             $  230.00
 11             $  443.10       #     11             $  443.10
 12             $   67.65       #     12             $   67.65
 13             $   10.00       #     13             $   10.00
 14             $  201.45       #     14             $  201.45
 15             $  110.00       #
-------------------------       #    -------------------------
Total           $ 2761.33       #    Total           $ 4226.65
```

Now, if we perform the comparison of the *synchronized* files '*`file1`*`.new`' and '*`file2`*`.new`' instead of the original ones, we obtain the right result. The output of '`numdiff` *`file1`*`.new` *`file2`*`.new`' is indeed

```
----------------
##1       #:1   <== file1:
               ==> file2:
                                                        @
----------------
##4       #>1   <== -------------------------
               ==>
 Line 4 in file "/home/ivano/file2.new" is shorter!
----------------
##10      #>1   <==
               ==> 6             $  432.90
 Line 10 in file "/home/ivano/file1.new" is shorter!
```

```
----------------
##11      #>1   <==
                ==> 7             $   60.88
 Line 11 in file "/home/ivano/file1.new" is shorter!
----------------
##12      #>1   <==
                ==> 8             $  759.00
 Line 12 in file "/home/ivano/file1.new" is shorter!
----------------
##13      #>1   <==
                ==> 9             $  322.54
 Line 13 in file "/home/ivano/file1.new" is shorter!
----------------
##19      #>1   <== 15             $  110.00
                ==>
 Line 19 in file "/home/ivano/file2.new" is shorter!
----------------
##21      #:3   <== 2761.33
                ==> 4226.65
 Absolute error = 1.4653200000e+3, Relative error = 5.3065732817e-1

+++  File "/home/ivano/file1.new" differs from file "/home/ivano/file2.new"
```

Unfortunately, `ndfilter` is currently implemented so that it gets easily confused. As we have seen above, `ndfilter` relies on the `sdiff` utility to work, but `sdiff` does not know anything about numerical differences: whenever `sdiff` compares a couple of files, it performs a literal comparison. Therefore, when `sdiff` has to establish if a line was deleted from, inserted in, or changed in one of the two files, it decides according to the results of a literal comparison, which is not in line with the way `numdiff` compares the files. To work right, `ndfilter` should use the numerical comparison whenever it is appropriate.

For instance, if we suppose that *file1* and *file2* have the following contents

`file1:`

```
 Day            Expenses
-------------------------
  1             $  233.56
  2             $  850.77
  3             $   12.55
  4             $  524.00
  5             $   78.25
 10             $  230.05
 11             $  443.10
 12             $   67.65
 13             $   10.00
 14             $  201.45
 15             $  110.00
-------------------------
Total           $ 2761.38
```

and

`file2:`

```
 Day            Expenses
```

```
 1             $  233.54
 2             $  850.80
 3             $   12.65
 4             $  524.00
 5             $   78.25
 6             $  432.90
 7             $   60.88
 8             $  759.00
 9             $  322.54
10             $  230.00
11             $  443.10
12             $   67.65
13             $   10.00
14             $  201.45
-------------------------
Total          $ 4226.76
```

then the differences between *file1* and *file2* are given by

- the insertion of the separator `-------------------------` in *file1* before the list of the days,
- the slightly different values for the expenses related to the days 1, 2, and 3,
- the insertion in *file2* of the lines related to the expenses for the days 6,7,8, and 9,
- the slightly different values for the expenses of the day 10,
- the addition in *file1* of the entry for the day 15, and
- the different values for the total sum of the expenses.

In this case the output of '`sdiff` *`file1 file2`*' is (when using `sdiff` from GNU diffutils 2.8.1)

```
file1:                          | file2:

 Day           Expenses             Day           Expenses
-------------------------       |   1           $  233.54
 1             $  233.56        |   2           $  850.80
 2             $  850.77        |   3           $   12.65
 3             $   12.55        <
 4             $  524.00            4           $  524.00
 5             $   78.25            5           $   78.25
10             $  230.05        |   6           $  432.90
                                >   7           $   60.88
                                >   8           $  759.00
                                >   9           $  322.54
                                >  10           $  230.00
11             $  443.10            11          $  443.10
12             $   67.65            12          $   67.65
13             $   10.00            13          $   10.00
14             $  201.45            14          $  201.45
15             $  110.00        <
-------------------------           -------------------------
Total          $ 2761.38        | Total         $ 4226.76
```

and it is easy to deduce that `ndfilter` will fail in resynchronizing the files. The files '*`file1`*`.new`' and '*`file2`*`.new`' created by '`ndfilter` *`file1 file2`*' are actually (side by side view):

```
file1:                      #    file2:
                            #
 Day            Expenses    #    Day            Expenses
-------------------------   #    1             $  233.54
  1             $  233.56   #    2             $  850.80
  2             $  850.77   #    3             $   12.65
  3             $   12.55   #
  4             $  524.00   #    4             $  524.00
  5             $   78.25   #    5             $   78.25
 10             $  230.05   #    6             $  432.90
                            #    7             $   60.88
                            #    8             $  759.00
                            #    9             $  322.54
                            #   10             $  230.00
 11             $  443.10   #   11             $  443.10
 12             $   67.65   #   12             $   67.65
 13             $   10.00   #   13             $   10.00
 14             $  201.45   #   14             $  201.45
 15             $  110.00   #
-------------------------   #   -------------------------
Total           $ 2761.38   #   Total          $ 4226.76
```

and the output of '`numdiff` *`file1`*`.new` *`file2`*`.new`' will be then completely misleading.

Until a new improved version of `ndfilter` will be made available, I strongly recommend to inspect the files created by `ndfilter` before comparing them with `numdiff`. At the moment it is quite likely that `ndfilter` does not manage to perform a correct resynchronization.

When you have a couple of files which need to be resynchronized to be correctly compared by `numdiff`, try to do that cleverly using `ndselect` and `grep` before turning to `ndfilter`.

SYNOPSIS

```
ndfilter -h|v
```

or

```
ndfilter [-s suffix][-e "opts"][-p col][-t tabwd][-n][-d][-l path][-o path]
file1 file2
```

where *file1* and *file2* are the names of the files to compare. The complete paths of the files should be given, a directory name is not accepted. These paths cannot refer to the same file.

In the first case `ndfilter` prints a short help or/and version number, Copyright and NO-Warranty disclaimer. In the second case `ndfilter` compares the given files by recalling `sdiff`, and use the results of this comparison to create a new couple of files which *should* be correctly synchronized for a later comparison through `numdiff`.

OPTIONS

- '`-s suffix`' Specify the suffix to add to the old name when creating each of the two new files (The default suffix is ".new")
- '`-e "opts"`' Specify the options to put on the command line when calling `sdiff` (The default behavior is executing `sdiff` with no options)
- '`-p col`' Specify the column number of the gutter in the '`sdiff`' output (The default value is the one for GNU sdiff, i.e. 62)
- '`-t tabwd`' Specify the width of the tabulations in the `sdiff` output (The default value is 8)

- '`-n`' Print the line number at the head of each line when writing the two new files
- '`-d`' Print debug information on stderr
- '`-l path`' Redirect warning and error messages from stderr to the indicated file
- '`-o path`' Redirect output from stdout to the indicated file
- '`-h`' Show this help message
- '`-v`' Show version number, Copyright and NO-Warranty

The option '`-s`' never accepts an empty suffix ("") as argument.

The option '`-e`' is particularly useful to ignore case differences. In this case, just use the option '`-e`' with the string ''`-I`'' as argument: '`ndfilter -e "-I" file1 file2`' . Look at the man page of the `sdiff` utility to know which options are supported by the version of `sdiff` installed on your system.

The options '`-p`' and '`-t`' are useful to cope with the differences in the output of the different versions of `sdiff`. Not all the versions of `sdiff` place the gutter in the same column, for instance the `sdiff` utility included in the GNU diffutils package (version 2.8.1) puts the gutter in the column number 62, while the `sdiff` utility included in SunOS® 5.10 puts the gutter in the column 64 (if I well remember :)). The option '`-p`' can be used to notify to `ndfilter` the position of the gutter in the `sdiff` output, if it differs from the default value (which is the one for GNU `sdiff`, namely 62). If `ndfilter` ends abnormally with an error message sounding like ''`Error occurred while creating the new files`'', then you probably omitted to specify the right value for the column position of the gutter.

The option '`-t`' is only useful if the `sdiff` utility does not expand automatically the tabulations present in the files when printing its output. Normally, the default value for the width of a tabulation (namely 8) is the right one. Otherwise, you can change it with the option '`-t`'.

The option '`-d`' is only used for debugging by the author of `ndfilter`, so you should never use it.

The options '`-l`' and '`-o`' have the same use as for `numdiff` and `ndselect`.

If the option '`-n`' is set, then, whenever `ndfilter` copies a line from one of the two original files in the new corresponding one, it adds at the head the line number referring to the original position of the line. It is only useful if you want to keep trace of the positions of the lines in the original files.

DIAGNOSTICS

Exit status: 0 in case of normal termination, -1 (255) in case of error.

7 Warnings

- After reading a numerical field, Numdiff can truncate its value if this number has *too much* digits with respect to the current precision. To be precise, denoted by P the current value of the precision:

  If the number is written in common notation, then `numdiff` will consider, in addition to all the digits of the integer part, only the first P digits of the fractional part.

  If the value is written in scientific notation, then `numdiff` will only consider the first P digits of the fractional part of the mantissa.

  By current value of the precision I mean the integer value specified by the option '`-#`', or the default one (35) when this option is not in use.
- At the moment Numdiff can only manage text files with an 8-bit encoding (ASCII and ISO 8859-* text files). Sooner or later Numdiff might support UTF-8 encoding.
- This manual describes the version 5.x of Numdiff. The versions preceding 4.0.0 (i.e. 3.0.2 and below) used a slightly different format for the output. Some changes were then introduced in order to make the output more consistent. These changes included a better use of the separator `----------------`, the use of the symbol `<<*>>` to mark the last line of a file when it does not have a terminating **newline** character, and the way Numdiff signals that a line has in one of the two files less fields than in the other file. While before these reports had the following format

  ```
  Line 1 in file "list2" is shorter:
  <== water
  ==>
  ```

  the current format is given by

  ```
  ##1       #>6   <== water
                  ==>
   Line 1 in file "list2" is shorter!
  ```

  Moreover, the versions of Numdiff preceding 4.0.0 did not provide the "verbose mode" with the related option '`-V`'. Finally, in the passage from the version 3.0.2 to the 4.0.0 one a minor bug related to the handling of the last line of a file has been corrected.
- Bug reports have to be sent to the address `ivprimi(at)libero(dot)it` . Please, put Numdiff in the subject and indicate the version of Numdiff you are using, the version of the operating system you are running and, if you know it, the version of the compiler used to build Numdiff.

Appendix A GNU Free Documentation License

GNU Free Documentation License
Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means

the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit

reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add

to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of

following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index