The mdwlist* package

Mark Wooding

2 May 1996

Contents

1 User guide

This package provides some vaguely useful list-related commands and environments:

- A way of building description-like environments.
- Commands for making 'compacted' versions of list environments
- A method for suspending and resuming enumerated lists.

1.1 Description list handling

Different sorts of description-type lists require different sorts of formatting: I think that's fairly obvious. There are essentially three different attributes which should be changable:

- the indentation of the items being described,
- the handling of labels which don't fit properly, and
- the style used to typeset the label text.

*The mdwlist package is currently at version 1.1, dated 2 May 1996.

The first two items should usually be decided for all description-like lists in the document, to ensure consistency of appearance. The last depends much more on the content of the labels.

basedescript

The basedescript environment acts as a 'skeleton' for description environments. It takes one argument, which contains declarations to be performed while constructing the list. I'd consider it unusual for the basedescript environment to be used in the main text: it's intended to be used to build other environments.

The declarations which can be used to define description-type environments include all of those which are allowed when setting up a list (see the LaTeX book for information here). Some others, which apply specifically to description lists, are also provided:

\desclabelwidth

- The `\desclabelwidth{`⟨*length*⟩`}` declaration sets labels to be left-aligned, with a standard width of ⟨*length*⟩; the item text is indented by ⟨*length*⟩ plus the value of `\labelsep`.

\desclabelstyle

- The label style determines how overlong labels are typeset. A style may be set using the `\desclabelstyle{`⟨*style*⟩`}` declaration. The following ⟨*style*⟩s are provided:

  `\nextlinelabel` If the label is too wide to fit next to the first line of text, then it is placed on a line by itself; the main text is started on the next line with the usual indentation.

  `\multilinelabel` The label is typeset in a parbox with the appropriate width; if it won't fit on one line, then the text will be split onto subsequent lines.

  `\pushlabel` If the label is too wide to fit in the space allocated to it, the start of the item's text will be 'pushed' over to the right to provide space for the label. This is the standard LaTeX description behaviour.

\makelabel

- The `\makelabel` command is responsible for typesetting a label. It is given one argument, which is the text given as an argument to the `\item` command; it should typeset it appropriately. The text will then be arranged appropriately according to the chosen label style. This command should be redefined using `\renewcommand`.

\defaultdesc

To allow document designers to control the global appearance of description lists, the `\defaultdesc` command may be redefined; it is called while setting up a new basedescript list, before performing the user's declarations. By default, it attempts to emulate the standard LaTeX description environment:[1]

```
\providecommand{\defaultdesc}{%
  \desclabelstyle{\pushlabel}%
  \renewcommand{\makelabel}[1]{\bfseries##1}%
  \setlength{\labelwidth}{0pt}%
}
```

[1] This is a slightly sanitised version of the real definition, which is given in the implementation section of this document.

Various labelling styles

Short label This is a short item, although it has quite a lot of text attached to it.

Slightly longer label text
This is a rather longer piece of text, with a correspondingly slightly longer label.

Short label This is a short item, although it has quite a lot of text attached to it.

Slightly longer label text This is a rather longer piece of text, with a correspondingly slightly longer label.

Short label This is a short item, although it has quite a lot of text attached to it.

Slightly longer label text This is a rather longer piece of text, with a correspondingly slightly longer label.

```
\begin{basedescript}{\desclabelstyle{\nextlinelabel}}
\item [Short label] This is a short item, although it has quite a
      lot of text attached to it.
\item [Slightly longer label text] This is a rather longer piece
      of text, with a correspondingly slightly longer label.
\end{basedescript}
\medskip
\begin{basedescript}{\desclabelstyle{\multilinelabel}}
\item [Short label] This is a short item, although it has quite a
      lot of text attached to it.
\item [Slightly longer label text] This is a rather longer piece
      of text, with a correspondingly slightly longer label.
\end{basedescript}
\medskip
\begin{basedescript}{\desclabelstyle{\pushlabel}}
\item [Short label] This is a short item, although it has quite a
      lot of text attached to it.
\item [Slightly longer label text] This is a rather longer piece
      of text, with a correspondingly slightly longer label.
\end{basedescript}
```

Unfortunately, LaTeX doesn't provide a means for overriding a command which may or may not have been defined yet; in this case, I'd probably recommend using the TeX primitive `\def` to redefine `\defaultdesc`.

If you want to redefine the description environment in terms of the commands in this package, the following method is recommended:

```
\renewenvironment{description}{%
  \begin{basedescript}{%
    \renewcommand{\makelabel}[1]{\bfseries##1}%
  }%
}{%
  \end{basedescript}%
}
```

This ensures that labels are typeset in bold, as is usual, but other properties of the list are determined by the overall document style.

1.2 Compacted lists

LaTeX tends to leave a certain amount of vertical space between list items. While this is normally correct for lists in which the items are several lines long, it tends to look odd if all or almost all the items are only one line long.

`\makecompactlist` The command `\makecompactlist{`⟨*new-env-name*⟩`}{`⟨*old-env-name*⟩`}` defines a new environment ⟨*new-env-name*⟩ to be a 'compacted' version of the existing environment ⟨*old-env-name*⟩; i.e., the two environments are the same except that the compacted version leaves no space between items or paragraphs within the list.

`itemize*` `enumerate*` `description*` So that the most common cases are already handled, the package creates compacted ∗-variants of the itemize, enumerate and description environments. These were created using the commands

```
\makecompactlist{itemize*}{itemize}
\makecompactlist{enumerate*}{enumerate}
\makecompactlist{description*}{description}
```

Some list environments accept arguments. You can pass an argument to a list environment using an optional argument to its compact variant. For example,

```
\begin{foolist*}[{someargument}]
```

1.3 Suspending and resuming list environments

`\suspend` `\resume` The `\suspend` and `\resume` commands allow you to temporarily end a list environment and then pick it up where you left off. The syntax is fairly simple:

⟨*suspend-cmd*⟩ ::= `\suspend` [`[` ⟨*name*⟩ `]`] `{` ⟨*env-name*⟩ `}`

⟨*resume-cmd*⟩ ::= `\resume` [`[` ⟨*name*⟩ `]`] `{` ⟨*env-name*⟩ `}` [`[` ⟨*text*⟩ `]`]

The ⟨*env-name*⟩ is the name of the environment; this will more often than not be the enumerate environment. The ⟨*name*⟩ is a magic name you can use to identify the suspended environment; if you don't specify this, the environment name is used instead.

Suspended environments

Here's some initial text. It's not very interesting.

1. This is an item.
2. This is another.

Some more commentry text.

3. Another item.

```
Here's some initial text.  It's
not very interesting.
\begin{enumerate*}
\item This is an item.
\item This is another.
\suspend{enumerate*}
Some more commentry text.
\resume{enumerate*}
\item Another item.
\end{enumerate*}
```

You can pass arguments to a resumed list environment through the second optional argument of the `\resume` command. If, for example, you're using David Carlisle's enumerate package, you could say something like

```
\begin{enumerate}[\bfseries{Item} i]
\item An item
\item Another item
\suspend{enumerate}
Some intervening text.
\resume{enumerate}[{[\bfseries{Item} i]}]
\item Yet another item
\end{enumerate}
```

2 Implementation

1 ⟨∗package⟩

2.1 Description lists

2.1.1 Label styles

`\nextlinelabel` The idea here is that if the label is too long to fit in its box, we put it on its own line and start the text of the item on the next. I've used `\sbox` here to capture colour changes properly, even though I have deep moral objections to the use of LaTeX boxing commands. Anyway, I capture the text in box 0 and compare its width to the amount of space I have in the label box. If there's enough, I can just unbox the box; otherwise I build a vbox containing the label text and an empty hbox – `\baselineskip` glue inserted between the two boxes makes sure we get the correct spacing between the two lines, and the vboxness of the vbox ensures that the baseline of my strange thing is the baseline of the *bottom* box. I then bash the vbox on the nose, so as to make its width zero, and leave that as the result. Either way, I then add glue to left align whatever it is I've created.

```
2 \def\nextlinelabel#1{%
3   \sbox\z@{#1}%
4   \ifdim\wd\z@>\labelwidth%
```

```
    \setbox\z@\vbox{\box\z@\hbox{}}%
    \wd\z@\z@%
    \box\z@%
  \else%
    \unhbox\z@%
  \fi%
  \hfil%
}
```

`\multilinelabel` A different idea – make the label text wrap around onto the next line if it's too long. This is really easy, actually. I use a parbox to contain the label text, set to be ragged right, because there won't be enough space to do proper justification. There's also a funny hskip there – this is because TEX only hyphenates things it finds sitting *after* glue items. The parbox is top-aligned, so the label text and the item run downwards together. I put the result in box 0, and remove the depth, so as not to make the top line of the item text look really strange.

All this leaves a little problem, though: if the item text isn't very long, the label might go further down the page than the main item, and possibly collide with the label below. I must confess that I'm not actually sure how to deal with this possibility, so I just hope it doesn't happen.

By the way, I don't have moral objections to `\parbox`.

```
\def\multilinelabel#1{%
  \setbox\z@\hbox{%
    \parbox[t]\labelwidth{\raggedright\hskip\z@skip#1}%
  }%
  \dp\z@\z@%
  \box\z@%
  \hfil%
}
```

`\pushlabel` Now we implement the old style behaviour – if the label is too wide, we just push the first line of the item further over to the right. This is really very easy indeed – we just stick some `\hfil` space on the right hand side (to left align if the label comes up too short). The 'push' behaviour is handled automatically by LATEX's item handling.

```
\def\pushlabel#1{{#1}\hfil}
```

2.1.2 The main environment

`\desclabelstyle` This is a declaration intended to be used only in the argument to the basedescript environment. It sets the label style for the list. All we do is take the argument and assign it to a magic control sequence which basedescript will understand later.

```
\def\desclabelstyle#1{\def\desc@labelstyle{#1}}
```

`\desclabelwidth` We set the label width and various other bits of information which will make all the bits of the description line up beautifully. We set `\labelwidth` to the value we're given (using `\setlength`, so that people can use the calc package if they so wish), and make the `\leftmargin` equal `\labelwidth` + `\labelsep`.

```
\def\desclabelwidth#1{%
  \setlength\labelwidth{#1}%
  \leftmargin\labelwidth%
```

```
  \advance\leftmargin\labelsep%
}
```

basedescript This is the new description environment. It does almost everything you could want from a description environment, I think. The argument is a collection of declarations to be performed while setting up the list.

This environment isn't really intended to be used by users – it's here so that you can define other description environments in terms of it,

The environment is defined in two bits – the 'start' bit here simply starts the list and inserts the user declarations in an appropriate point, although sensible details will be inerted if the argument was empty.

```
\def\basedescript#1{%
```

We must start the list. If the `\item` command's optional argument is missing, we should just leave a blank space, I think.

```
  \list{}{%
```

So far, so good. Now put in some default declarations. I'll use a separate macro for this, so that the global appearance of lists can be configured.

```
    \defaultdesc%
```

Now we do the user's declarations.

```
    #1%
```

Now set up the other parts of the list. We set `\itemindent` so that the label is up against the current left margin. (The standard version actually leaves the label hanging to the left of the margin by a distance of `\labelsep` for a reason I can't quite comprehend – there's an `\hspace{\labelsep}` in the standard `\makelabel` to compensate for this. Strange...)

To make the label start in the right place, the text of the item must start a distance of `\labelwidth`+`\labelsep` from the (pre-list) left hand margin; this means that we must set `\itemindent` to be `\labelwidth` + `\labelsep` − `\leftmargin`. Time for some TEX arithmetic.

```
    \itemindent\labelwidth%
    \advance\itemindent\labelsep%
    \advance\itemindent-\leftmargin%
```

Now we must set up the label typesetting. We'll take the `\makelabel` provided by the user, remember it, and then redefine `\makelabel` in terms of the `\desclabelstyle` and the saved `\makelabel`.

```
    \let\desc@makelabel\makelabel%
    \def\makelabel##1{\desc@labelstyle{\desc@makelabel{##1}}}%
```

I can't think of anything else which needs doing, so I'll call it a day there.

```
  }%
}
```

Now we define the 'end-bit' of the environment. Since all we need to do is to close the list, we can be ever-so slightly clever and use `\let`.

```
\let\endbasedescript\endlist
```

Note that with these definitions, the standard description environment can be emulated by saying simply:

```
\renewenvironment{description}{%
  \begin{basedescript}{}%
}{%
  \end{basedescript}
}
```

\defaultdesc Now to set up the standard description appearance. In the absence of any other declarations, the label will 'push' the text out the way if the text is too long. The standard \labelsep and \leftmargin are not our problem. We typeset the label text in bold by default. Also, \labelwidth is cleared to 0 pt, because this is what LaTeX's usual description does.

```
\providecommand\defaultdesc{%
  \desclabelstyle\pushlabel%
  \def\makelabel##1{\bfseries##1}%
  \labelwidth\z@%
}
```

2.1.3 An example

note The note environment is a simple application of the general description list shown above. It typesets the label (by default, the text '**note**') at the left margin, and the note text indented by the width of the label.

The code is simple – we take the environment's argument (which may have been omitted), store it in a box (using \sbox again, to handle colour changes correctly), set the label width from the width of the box, and then create a single item containing the label text. The text of the environment then appears in exactly the desired place.

I've not used \newcommand here, for the following reasons:

- I don't like it much, to be honest.
- Until very recently, \newcommand only allowed you to define 'long' commands, where new paragraphs were allowed to be started in command arguments; this removes a useful check which traps common errors like missing out '}' characters. I'd prefer to be compatible with older LaTeXs than to use the new \newcommand which provides a $*$-form to work around this restriction.

```
\def\note{\@ifnextchar[\note@i{\note@i[Note:]}}
\def\note@i[#1]{%
  \basedescript{%
    \sbox\z@{\makelabel{#1}}%
    \desclabelwidth{\wd\z@}%
  }%
  \item[\box\z@]%
}
\let\endnote\endbasedescript
```

2.2 Compacted environments

Normal lists tend to have rather too much space between items if all or most of the item texts are one line or less each. We therefore define a macro \makecompactlist whuch creates 'compacted' versions of existing environments.

\makecompactlist We're given two arguments: the name of the new environment to create, and the name of the existing list environment to create.

The first thing to do is to ensure that the environment we're creating is actually valid (i.e., it doesn't exist already, and it has a sensible name). We can do this with the internal LaTeX macro `\@ifdefinable`.

```
\def\makecompactlist#1#2{%
  \expandafter\@ifdefinable\csname#1\endcsname%
    {\makecompactlist@i{#1}{#2}}%
}
```

We also ought to ensure that the other environment already exists. This isn't too tricky. We'll steal LaTeX's error and message for this.

```
\def\makecompactlist@i#1#2{%
  \@ifundefined{#2}{\me@err{Environment `#2' not defined}\@ehc}{}%
```

The main work for starting a compact list is done elsewhere.

```
  \@namedef{#1}{\@compact@list{#2}}%
```

Now to define the end of the environment; this isn't terribly difficult.

```
  \expandafter\let\csname end#1\expandafter\endcsname%
                  \csname end#2\endcsname%
```

That's a compacted environment created. Easy, no?

```
}
```

The general case macro has to try slurping some arguments, calling the underlying environment, and removing vertical space.

```
\def\@compact@list#1{\@testopt{\@compact@list@i{#1}}{}}
\def\@compact@list@i#1[#2]{%
  \@nameuse{#1}#2%
  \parskip\z@%
  \itemsep\z@%
}%
```

itemize* enumerate* description* Let's build some compacted environments now. These are easy now that we've done all the work above.

```
\makecompactlist{itemize*}{itemize}
\makecompactlist{enumerate*}{enumerate}
\makecompactlist{description*}{description}
```

2.3 Suspending and resuming lists

This is nowhere near perfect; it relies a lot on the goodwill of the user, although it seems to work fairly well.

\suspend The only thing that needs saving here is the list counter, whose name is stored in `\@listctr`. When I get a request to save the counter, I'll build a macro which will restore it when the environment is restored later.

The first thing to do is to handle the optional argument. `\@dblarg` will sort this out, giving me a copy of the mandatory argument if there's no optional one provided.

```
\def\suspend{\@dblarg\suspend@i}
```

That's all we need to do here.

```
\def\suspend@i[#1]#2{%
```

Now I have a little problem; when I `\end` the environment, it will close off the grouping level, and the counter value will be forgotten. This is bad. I'll store all my definitions into a macro, and build the `\end` command into it; that way, everything will be expanded correctly. This requires the use of `\edef`, which means I must be a little careful.

```
  \edef\@tempa{%
```

The first thing to do is to end the environment. I don't want `\end` expanded yet, so I'll use `\noexpand`.

```
    \noexpand\end{#2}%
```

Now I must define the 'resume' macro. I'll use `\csname` to build the named identifier into the name, so it won't go wrong (maybe). There's a little fun here to make the control sequence name but not expand it here.

```
    \def\expandafter\noexpand\csname resume.#1\endcsname{%
```

The counter name is hidden inside `\@listctr`, so the actual counter is called '`\csname c@\@listctr\endcsname`'. I'll use `\the` to read its current value, and assign it to the counter when the macro is used later.

```
      \csname c@\@listctr\endcsname\the\csname c@\@listctr\endcsname%
```

That's all we need to do there. Now close the macros and run them.

```
    }%
  }%
  \@tempa%
}
```

`\resume` Resuming environments is much easier. Since I use `\csname` to build the name, nothing happens if you try to resume environments which weren't suspended. I'll trap this and raise an error. Provide an optional argument for collecting arguments to the target list.

```
\def\resume{\@dblarg\resume@i}
\def\resume@i[#1]#2{\@testopt{\resume@ii{#1}{#2}}{}}
\def\resume@ii#1#2[#3]{%
  \begin{#2}#3%
  \@ifundefined{resume.#1}{\ml@err@resume}{\@nameuse{resume.#1}}%
}
```

That's all there is.

```
⟨/package⟩
```

Mark Wooding, 2 May 1996

Appendix

A The GNU General Public Licence

The following is the text of the GNU General Public Licence, under the terms of which this software is distrubuted.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

A.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

A.2 Terms and conditions for copying, distribution and modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source

code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program 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.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public 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.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. Because the Program is licensed free of charge, there is no warranty for the Program, to the extent permitted by applicable law. except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the Program is with you. Should the Program prove defective, you assume the cost of all necessary servicing, repair or correction.

12. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the Program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

END OF TERMS AND CONDITIONS

A.3 Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) 19yy  <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider

it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Index

Numbers written in italic refer to the page where the corresponding entry is described, the ones underlined to the code line of the definition, the rest to the code lines where the entry is used.