The at* package

Mark Wooding

2 May 1996

Contents

1 User guide

The at package is an attempt to remove a lot of tedious typing that ends up in LaTeX documents, by expanding the number of short command names available. The new command names begin with the '@' character, rather than the conventional '\', so you can tell them apart.

The package provides some general commands for defining @-commands, and then uses them to define some fairly simple ones which will be useful to most people.

The rules for @-command names aren't terribly complex:

- If the first character of the name is a letter, then the command name consists of all characters up to, but not including, the first nonletter. Spaces following the command name are ignored.
- If the first character of the name is a backslash, then the @-command name consists of the control sequence introduced by the backslash.

*The at package is currently at version 1.3, dated 2 May 1996.

- Otherwise, the command name consists only of that first character. Spaces following the name are not ignored, unless that character was itself a space character.

Usually, digits are not considered to be letters. However, the `at` package will consider digits to be letters if you give it the `digits` option in the `\usepackage` command. (Note that this only affects the `at` package; it won't change the characters allowed in normal command names.)

`\atallowdigits` `\atdisallowdigits`

You can enable and disable digits being considered as letters dynamically. The `\atallowdigits` command allows digits to be used as letters; `\atdisallowdigits` prevents this. Both declarations follow LaTeX's usual scoping rules. Both of these commands have corresponding environments with the same names (without the leading '\', obviously).

1.1 Defining @-commands

`\newatcommand` `\renewatcommand`

The `\newatcommand` command will define a new @-command using a syntax similar to `\newcommand`. For example, you could define

```
\newatcommand c[1]{\chapter{#1}}
```

to make `@c{`⟨*name*⟩`}` equivalent to `\chapter{`⟨*name*⟩`}`.

A `\renewatcommand` is also provided to redefine existing commands, should the need arise.

`\atdef`

For TeX hackers, the `\atdef` command defines @-commands using a syntax similar to TeX's built-in `\def`.

As an example, the following command makes `@/`⟨*text*⟩`/` write its argument ⟨*text*⟩ in italics:

```
\atdef/#1/{\textit{#1}}
```

The real implementation of the `@/.../` command is a bit more complex, and is given in the next section.

You can use all of TeX's features for defining the syntax of your command. (See chapter 20 of *The TeXbook* for more details.)

`\atlet`

Since `\atdef` is provided to behave similarly to `\def`, `at` provides `\atlet` which works similarly to `\let`. For example you can say

```
\atlet!=\index
```

to allow the short `@!` to behave exactly like `\index`.

Note that all commands defined using these commands are robust even if you use fragile commands in their definitions. Unless you start doing very strange things, @-commands never need `\protecting`.

1.2 Predefined @-commands

A small number of hopefully useful commands are provided by default. These are described in the table below:

Command	Meaning
`@@`	Typesets an '@' character.
`@/`⟨*text*⟩`/`	In text (LR or paragraph) mode, typesets its argument emphasised. In maths mode, it always chooses italics.
`@*`⟨*text*⟩`*`	Typesets its argument ⟨*text*⟩ in bold.
`@i{`⟨*text*⟩`}`	Equivalent to '`\index{`⟨*text*⟩`}`'.
`@I{`⟨*text*⟩`}`	As for `@i`, but also writes its argument to the document.

Package writers should not rely on any predefined @-commands – they're provided for users, and users should be able to redefine them without fear of messing anything up. (This includes the 'standard' commands provided by the `at` package, by the way. They're provided in the vague hope that they might be useful, and as examples.)

2 Implementation

1 ⟨*package⟩

2.1 Options handling

We need a switch to say whether digits should be allowed. Since this is a user thing, I'll avoid `\newif` and just define the thing by hand.

```
2 \def\atallowdigits{\let\ifat@digits\iftrue}
3 \def\atdisallowdigits{\let\ifat@digits\iffalse}
```

Now define the options.

```
4 \DeclareOption{digits}{\atallowdigits}
5 \DeclareOption{nodigits}{\atdisallowdigits}
6 \ExecuteOptions{nodigits}
7 \ProcessOptions
```

2.2 How the commands work

Obviously we make the '@' character active. It inspects the next character (or argument, actually – it can be enclosed in braces for longer commands, although this is a bit futile), and builds the command name from that.

The `\at` command is equivalent to the active '@' character always.

2.3 Converting command names

We need to be able to read an @-command name, and convert it to a normal TEX control sequence. First, we declare some control sequences for braces, which we need later.

```
8 \begingroup
9 \catcode`\<1
10 \catcode`\>2
11 \catcode`\{12
```

```
\catcode`\}12
\gdef\at@lb<{>
\gdef\at@rb<}>
\gdef\at@spc< >
\endgroup
```

I'll set up some helper routines now, to help me read the command names. The way this works is that we `\futurelet` the token into `\@let@token`. These routines will then sort out what to do next.

`\at@test` Given an `\if...` test, does its first or second argument.

```
\def\at@test#1\then{%
  #1\expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi%
}
```

`\at@ifcat` Checks the category code of the current character. If it matches the argument, it does its second argument, otherwise it does the third.

```
\def\at@ifcat#1{\at@test\ifcat#1\noexpand\@let@token\then}
```

`\at@ifletter` This routine tests the token to see if it's a letter, and if so adds it to the token list and does the first argument; otherwise it does the second argument. It accepts digits as letters if the switch is turned on.

There's some fun later, so I'll describe this slowly. First, we compare the category code to a letter, and if we have a match, we know we're done; we need to pick up the letter as an argument. If the catcode is 'other', we must compare with numbers to see if it's in range.

```
\def\at@ifletter#1#2{%
  \at@ifcat x%
    {\at@ifletter@ii{#1}}%
    {\at@ifcat 0%
      {\at@ifletter@i{#1}{#2}}%
      {#2}%
    }%
}
```

Right. It's 'other' (so it's safe to handle as a macro argument) and we need to know if it's a digit. This is a little tricky: I use `\if` to compare two characters. The first character is '1' or '0' depending on the 'digit' switch; the second is '1' or 'x' depending on whether it's actually a digit. They'll only match if everything's worked out OK.

```
\def\at@ifletter@i#1#2#3{%
  \at@test\if%
    \ifat@digits1\else0\fi%
    \ifnum`#3<`0x\else\ifnum`#3>`9x\else1\fi\fi%
  \then%
    {\at@ifletter@ii{#1}{#3}}%
    {#2#3}%
}
```

Right; we have the character, so add it to the list and carry on.

```
\def\at@ifletter@ii#1#2{\toks@\expandafter{\the\toks@#2}#1}
```

Now we define the command name reading routines. We have *almost* the same behaviour as TEX, although we can't support '%' characters for reasons to do with TEX's tokenising algorithm.

`\at@read@name` The routine which actually reads the command name works as follows:

1. Have a peek at the next character. If it's a left or right brace, then use the appropriate character.
2. If the character is not a letter, just use the character (or whole control sequence.
3. Finally, if it's a letter, keep reading letters until we find one that wasn't.

First, we do some setting up and read the first character

```
\def\at@read@name#1{%
  \let\at@next=#1%
  \toks@{}%
  \futurelet\@let@token\at@rn@i%
}
```

Next, sort out what to do, based on the category code.

```
\def\at@rn@i{%
  \def\@tempa{\afterassignment\at@rn@iv\let\@let@token= }%
  \at@ifletter%
    {\futurelet\@let@token\at@rn@iii}%
    {\at@ifcat\bgroup%
      {\toks@\expandafter{\at@lb}\@tempa}%
      {\at@ifcat\egroup%
        {\toks@\expandafter{\at@rb}\@tempa}%
        {\at@ifcat\at@spc%
          {\toks@{ }\@tempa}%
          {\at@rn@ii}%
        }%
      }%
    }%
}
```

Most types of tokens can be fiddled using `\string`.

```
\def\at@rn@ii#1{%
  \toks@\expandafter{\string#1}%
  \at@rn@iv%
}
```

We've found a letter, so we should check for another one.

```
\def\at@rn@iii{%
  \at@ifletter%
    {\futurelet\@let@token\at@rn@iii}%
    {\@ifnextchar.\at@rn@iv\at@rn@iv}%
}
```

Finally, we need to pass the real string, as an argument, to the macro. We make `\@let@token` relax, since it might be something which will upset TEX later, e.g., a # character.

```
\def\at@rn@iv{%
  \let\@let@token\relax%
  \expandafter\at@next\csname at.\the\toks@\endcsname%
}
```

`\at@cmdname` Given a control sequence, work out which @-command it came from.

```
\def\at@cmdname#1{\expandafter\at@cmdname@i\string#1\@@foo}
```

Now extract the trailing bits.

```
\def\at@cmdname@i#1.#2\@@foo{#2}
```

`\at@decode` The `\at@decode` macro takes an extracted @-command name, and tries to execute the correct control sequence derived from it.

```
\def\at@decode#1{%
  \at@test\ifx#1\relax\then{%
    \PackageError{at}{Unknown @-command `@\at@cmdname#1'}{%
      The @-command you typed wasn't recognised, so I've ignored it.
    }%
  }{%
    #1%
  }%
}
```

`\@at` We'd like a measure of compatibility with amsmath. The @-commands provided by amsmath work only in maths mode, so this gives us a way of distinguishing. If the control sequence `\Iat` is defined, and we're in maths mode, we'll call that instead of doing our own thing.

```
\def\@at{%
  \def\@tempa{\at@read@name\at@decode}%
  \ifmmode\ifx\Iat\not@@defined\else%
    \let\@tempa\Iat%
  \fi\fi%
  \@tempa%
}
```

2.4 Defining new commands

`\at@buildcmd` First, we define a command to build these other commands:

```
\def\at@buildcmd#1#2{%
  \expandafter\def\csname\expandafter
    \@gobble\string#1@decode\endcsname##1{#2##1}%
  \edef#1{%
    \noexpand\at@read@name%
    \expandafter\noexpand%
      \csname\expandafter\@gobble\string#1@decode\endcsname%
  }%
}
```

`\newatcommand` `\renewatcommand` `\provideatcommand` `\atdef` `\atshow` Now we define the various operations on @-commands.

```
\at@buildcmd\newatcommand\newcommand
\at@buildcmd\renewatcommand\renewcommand
\at@buildcmd\provideatcommand\providecommand
```

```
\at@buildcmd\atdef\def
\at@buildcmd\atshow\show
```

\atlet `\atlet` is rather harder than the others, because we want to allow people to say things like `\atlet`⟨*name*⟩`=@`⟨*name*⟩. The following hacking does the trick. I'm trying very hard to duplicate `\let`'s behaviour with respect to space tokens here, to avoid any surprises, although there probably will be some differences. In particular, `\afterassignment` won't work in any sensible way.

First, we read the name of the @-command we're defining. We also open a group, to stop messing other people up, and make '@' into an 'other' token, so that it doesn't irritatingly look like its meaning as a control sequence.

```
\def\atlet{%
  \begingroup%
  \@makeother\@%
  \at@read@name\atlet@i%
}
```

Put the name into a scratch macro for later use. Now see if there's an equals sign up ahead. If not, this will gobble any spaces in between the @-command name and the argument.

```
\def\atlet@i#1{%
  \def\at@temp{#1}%
  \@ifnextchar=\atlet@ii{\atlet@ii=}%
}
```

Now we gobble the equals sign (whatever catcode it is), and peek at the next token up ahead using `\let` with no following space.

```
\def\atlet@ii#1{\afterassignment\atlet@iii\global\let\at@gnext=}
```

The control sequence `\at@gnext` is now `\let` to be whatever we want the @-command to be, unless it's picked up an '@' sign. If it has, we've eaten the `@` token, so just read the name and pass it on. Otherwise, we can `\let` the @-command directly to `\at@gnext`. There's some nastiness here to make `\the\toks@` expand before we close the group and restore its previous definition.

```
\def\atlet@iii{%
  \if @\noexpand\at@gnext%
    \expandafter\at@read@name\expandafter\atlet@iv%
  \else%
    \expandafter\endgroup%
    \expandafter\let\at@temp= \at@gnext%
  \fi%
}
```

We've read the source @-command name, so just copy the definitions over.

```
\def\atlet@iv#1{%
  \expandafter\endgroup%
  \expandafter\let\at@temp=#1%
}
```

2.5 Robustness of @-commands

We want all @-commands to be robust. We could leave them all being fragile, although making robust @-commands would then be almost impossible. There are two problems which we must face:

- The '\@at' command which scans the @-command name is (very) fragile. I could have used \DeclareRobustCommand for it (and in fact I did in an earlier version), but that doesn't help the other problem at all.
- The 'name' of the @-command may contain active characters or control sequences, which will be expanded at the wrong time unless we do something about it now.

We must also be careful not to introduce extra space characters into any files written, because spaces are significant in @-commands. Finally, we have a minor problem in that most auxiliary files are read in with the '@' character set to be a letter.

\at Following the example of LaTeX's 'short' command handling, we'll define \at to decide what to do depending on what \protect looks like. If we're typesetting, we just call \@at (above) and expect it to cope. Otherwise we call \at@protect, which scoops up the \fi and the \@at, and inserts other magic.

```
\def\at{\ifx\protect\@typeset@protect\else\at@protect\fi\@at}
```

\at@protect Since we gobbled the \fi from the above, we must put that back. We then need to do things which are more complicated. If \protect is behaving like \string, then we do one sort of protection. Otherwise, we assume that \protect is being like \noexpand.

```
\def\at@protect\fi#1{%
  \fi%
  \ifx\protect\string%
    \expandafter\at@protect@string%
  \else%
    \expandafter\at@protect@noexpand%
  \fi%
}
```

\at@protect@string When \protect is \string, we don't need to be able to recover the original text particularly accurately – it's for the user to look at. Therefore, we just output a $\texttt{@}_{11}$ and use \string on the next token. This must be sufficient, since we only allow multi-token command names if the first token is a letter (code 11).

```
\def\at@protect@string{@\string}
```

\at@protect@noexpand This is a little more complex, since we're still expecting to be executed properly at some stage. However, there's a cheeky dodge we can employ since the \at command is thoroughly robustified (or at least it will be by the time we've finished this). All \@unexpandable@protect does is confer repeated robustness on a fragile command. Since our command is robust, we don't need this and we can get away with just using a single \noexpand, both for the \@at@ command and the following token (which we must robustify, because no-one else can do it for us – if anyone tries, they end up using the @\protect command which is rather embarassing).

I'll give the definition, and then examine how this expands in various cases.

```
\def\at@protect@noexpand{\noexpand\@at@ @\noexpand}
\def\@at@#1{\at}
```

A few points, before we go into the main examination of the protection. I've inserted a $@_{11}$ token, which is gobbled by `\@at@` when the thing is finally expanded fully. This prevents following space tokens in an `\input` file from being swallowed because they follow a control sequence. (I can't use the normal $@_{13}$ token, because when files like the `.aux` file are read in, `@` is given code 11 by `\makeatletter`.)

Now for a description of why this works. When `\at` is expanded, it works out that `\protect` is either `\noexpand` or `\@unexpandable@protect`, and becomes `\at@protect@noexpand`. Because of the `\noexpand` tokens, this stops being expanded once it reaches `@at@` $@_{11}\,x$ (where x is the token immediately following the $@_{13}$ character). If this is expanded again, for example in another `\edef`, or in a `\write` or a `\mark`, the `\@at@` wakes up, gobbles the following `@` (whatever catcode it is – there may be intervening `\write` and `\input` commands) and becomes `\at`, and the whole thing can start over again.

2.6 Enabling and disabling @-commands

`\aton` We define the `\aton` command to enable all of our magic. We store the old catcode in the `\atoff` command, make '@' active, and make it do the stuff.

```
\def\aton{%
  \ifnum\catcode`\@=\active\else%
    \edef\atoff{\catcode`\noexpand\@\the\catcode`\@}%
    \catcode`\@\active%
    \lccode`\~`\@%
    \lowercase{\let~\at}%
  \fi%
}
```

`\atoff` The `\atoff` command makes '@' do the stuff it's meant to. We remember the old catcode and revert to it. This is largely unnecessary.

```
\def\atoff{\catcode`\@12}
```

`\makeatother` Now we make our active '@' the default outside of package files.

```
\let\makeatother\aton
```

And we must make sure that the user can use all of our nice commands. Once the document starts, we allow @-commands.

```
\AtBeginDocument{\aton}
```

`\dospecials` We must add the '@' character to the various specials lists.
`\@sanitize`

```
\expandafter\def\expandafter\dospecials\expandafter{\dospecials\do\@}
\expandafter\def\expandafter\@sanitize\expandafter{%
  \@sanitize\@makeother\@}
```

2.7 Default @-commands

We define some trivial examples to get the user going.

```
\expandafter\chardef\csname at.@\endcsname=`\@
\atdef*#1*{\ifmmode\mathbf{#1}\else\textbf{#1}\fi}
\atdef/#1/{\ifmmode\mathit{#1}\else\emph{#1}\fi}
\atlet i=\index
\atdef I#1{#1\index{#1}}
⟨/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

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
675 Mass Ave, Cambridge, MA 02139, USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

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.