GNU Info: a Decade of Hypertext Experience

Roland H. Pesch
Cygnus Support
1937 Landings Drive
Mountain View, CA 94043

pesch@cygnus.com
+1 415 903 1410

Introduction

Proprietary software vendors are beginning to awaken to the benefits of online hypertext documentation. Meanwhile, for over a decade, users and developers of free software have been exploiting the hypertext Info documentation format (first introduced by Richard Stallman around 1976).

GNU[1] Info formatters and readers are highly portable, running (at least) on all Unix systems, DOS, VMS, Commodore Amiga, and Atari ST; they support hypertext links both within and between documents, including a rich structure of automatic links between the sections of a manual; they provide both integrated index generation and dynamic, arbitrary text search; and they permit generating both printed manuals and online hypertext from a single source file.

[1] "GNU" refers to the system of free software originated and disseminated by the Free Software Foundation, founded by Richard Stallman. The leading "G" is sounded.

This paper itself was first written, and circulated for review, as Info hypertext; if formatted and installed for the Info system, its references to other documents (and its own internal structure) are live hypertext links.

History of the GNU Info format

In the early 1970s, the Massachusetts Institute of Technology (MIT) sponsored a research project called "project MAC". Many seminal ideas originated with this project, including much of the basis of today's operating systems. More to our point, Richard Stallman, who was part of this project, wrote the first version of a text editor named Emacs. Originally, Emacs was written in the command language of another editor, TECO; it rapidly became the editor of choice, relegating TECO to a vehicle for its implementation language. To provide on-line help for Emacs, Stallman also invented a straightforward method of representing hypertext: Info files.

In the spirit of sharing that was originally typical of software development, the programs from project MAC—including Emacs and its Info files—were available to anyone who wished to copy them. In particular, they were ported to the TENEX and TOPS-20 operating systems, which were extremely popular in universities and research establishments. By this time, Info files were available for many other useful and interesting topics, ranging from Cambridge restaurants to the system calls for the TOPS-20 operating system itself. This last application alone guaranteed the spread of Info in the TOPS-20 envi-

ronment, since hard copy documentation for this essential topic was often hard to come by. For local extensions to the system calls, the Info files were sometimes the *only* documentation.

In the meantime, as the cooperative environment fostered by MIT fell victim to attempts to keep secrets in order to make software development proprietary, Richard Stallman resigned his post at MIT in order to establish the Free Software Foundation (FSF). The FSF creates and disseminates copyrighted, high-quality software under a license designed to promote software sharing. Among the earliest and most popular of these programs was a new implementation of Emacs: GNU Emacs, largely written in a dialect of Lisp (it includes Stallman's own implementation of a Lisp interpreter).

GNU Emacs continued to use Info files for online documentation. But there was also demand for printed documentation. Richard Stallman noticed that there were many parallels between the structures of printed text and the structures needed for hypertext—chapters and sections corresponded more or less to hypertext nodes, cross references to links. Stallman exploited these parallels by writing the first Emacs manual in a version of Scribe modified to produce Info files. Stallman also wrote Botex (a formatter based on the freely available TeX program by Don Knuth of Stanford University) for other Lisp documentation. For the FSF, Stallman combined these ideas, designing a new dual-purpose formatting markup language called Texinfo.

Texinfo permits using a single file of text to generate *either* printed documentation (via TeX), or online Info files (generated by a separate formatting program). By the first issue of the fledgling FSF's newsletter, Stallman was able to announce

> I now have a truly compatible pair of programs which can convert a file of texinfo format documentation into either a printed manual or an Info file.
>
> *GNUS Bulletin* #1, February 1986

Since that point, the FSF has continued to exploit (and develop!) Texinfo as the documentation standard for all its programs. Since the FSF is dedicated to promoting software sharing, Info documents have penetrated far and wide—and the Info style has been adopted by many organizations besides the FSF for their own needs.

Today, the FSF maintains two programs for browsing Info hypertext: Stallman's Info mode within GNU Emacs[2], and a compact but powerful stand-alone program called info, written by Brian Fox. For generating Info files, there are likewise two FSF standard programs: a Lisp formatter (originally by Stallman, now maintained by Robert Chassell), for use within GNU Emacs (used, for example, as the command `texinfo-format-buffer`), and the standalone C program `makeinfo`, also by Brian Fox. Others have also written Info browsers and formatters. Among the other stand-alone browsers, the most notable are `tkinfo` and `xinfo`, which exploit the graphical environments current on today's workstations. Among alternative formatters, latexinfo is fairly widespread; it generates Info files from source files that resemble a subset of LaTeX, rather than from standard Texinfo source.

Who uses Info today?

Subjectively, it is clear that Info is extremely widespread; for one thing, it goes everywhere with the very popular GNU free software. Moreover, there is a continual flow of contributed improvements and suggestions for the Info formatters and readers in the GNU mailing lists. But a subjective impression is hard to communicate.

In an attempt to quantify the spread of Info today, I broadcast an open-ended query ("How do you and your organization use Info?") to several Internet news-groups that carry discussions related to the GNU project.

The categories tabulated here are not usually mutually exclusive; for example, someone who uses Info for all documents certainly uses it to read GNU documents! I have *not* counted such uses in both categories, however—my intent is to show the most inclusive use. Those counted under "GNU and other outside documentation" reported using Info *only* for that purpose.

[2] A somewhat enhanced alternative Info mode for Emacs, by Dave Gillespie, is also freely available.

Internet Survey: Use of Info

Total respondents: 159

Count	*% users*	
12	8.8	Primary form of online documentation
69	50.4	Some locally written documentation
2	1.4	Locally converted FAQ lists
5	3.6	Personal documentation
49	35.8	GNU and other outside documentation
137	100	Total respondents using Info
8		We avoid it!
11		What is it?
3		Response could not be classified

One distortion is immediately apparent: people who did not feel they had something "interesting" to say did not, for the most part, respond. As a result, readers are under-represented. Of the 35.8% who reported using Info only for reading documents distributed by others, most also added something further: a desire to use Info more, or a former use of Info, or some sort of query to me.

Nevertheless, the responses demonstrate the extent to which Info is adopted actively, rather than passively! The largest fraction (59.2%—the sum of the first two categories in the table) of respondents use Info to publish internal documentation; a sizable fraction (8.8%) use it for *all* documentation within their organization.

Counting the apparent locations of respondents reflects the current preponderance of North Americans on the Internet, but the geographic spread may nevertheless be of interest. The following table includes only those respondents who use Info:

Info user survey: Geographic Distribution

Count	*Location*
1	Austria
5	Australia
1	Finland
3	France
8	Germany
1	Greece
1	Iceland
1	Italy
2	Netherlands
1	Russia
1	Switzerland
6	United Kingdom
32	Europe (total)
5	Canada
48	North America, Commercial
41	North America, Educational
3	North America, Government
5	North America, NGO
102	North America (total)
1	Venezuela
1	Japan
1	Israel

The reader: using Info

Readers are always in full control of how they wish to read Info documentation. A reader can

- rely on the authors' organization;
- follow cross-reference links back and forth to pursue a particular subject;
- use additional menus provided by the author as alternative structures;
- search the index or indices;
- search for arbitrary words, phrases, or patterns.

And, of course, a reader can switch among these modes of navigation at any time—using them to supplement rather than supplant each other.

A major strength, and the main weakness, of GNU Info both stem from its long history: GNU Info uses no special graphics display capabilities to display online documentation. This means that everyone in an organization can use a common collection of Info files, whether they sit at a graphics workstation, a new or old microcomputer, or a dumb terminal. It also means that Info displays look somewhat old fashioned—they are hyper*text*, not "multimedia", and do not exploit rich typography for online displays. (Of course, Info files are often easier to read than "sexier" online documentation formats, thanks to using fonts designed for easy viewing on a screen, rather than imitating the appearance of fonts that are attractive on paper!)

Lacking the ability to incorporate graphics, however, becomes a significant handicap—at least to documenting graphically-oriented software. Another serious handicap, at least in countries where the primary language is not English, has been the limitation to the seven-bit ASCII character set.

However, the Info system is evolving to surmount these difficulties.

The major virtue of free software is that it is freely *modifiable*; you have all the source code to the software, and you can study it and make it do more. At the University of Florida Mathematics department, G. R. Fischer and Christopher W. Stark[3] have discovered an undocumented feature in the Emacs version of the Info browser: "active nodes". With some minor modifications to the software, they were able to design documents with Info nodes that, when visited, pop up graphical windows or even controlled interactive displays with running examples of other programs.

As for the richer character sets required by other languages, the same virtue of free software comes to the rescue. Although the FSF itself has not yet settled on a complete solution, here is what Ken'ichi Handa, Satoru Tomura, and Mikiko Nisikimi of Japan's Electrotechnical Laboratory[4] have done about the problem:

> Mule is a MULtilingual Enhancement to GNU Emacs. It can handle not only ASCII characters (7 bits) and ISO Latin-1 (8 bits), but also Japanese, Chinese, Korean (16 bits) coded in the ISO2022 standard and its variants (e.g. EUC, Compound Text). For Chinese there is support for both GB and Big5. In addition, Thai (based on TIS620) and Vietnamese (based on VISCII and VSCII) are also supported.
>
> A text buffer in Mule can contain a mixture of characters from these languages. To input any of these characters, you can use various input methods provided by Mule itself. In addition, if you use Mule under some terminal emulator (kterm, cxterm, or exterm), you can use any input methods supported by the emulator.
>
> ...
>
> Many kinds of European character inputting methods are provided by QUAIL system (bundled with Mule). For the moment, QUAIL provides inputting methods for: Latin1, Latin2, . . . Latin5, Greek (ISO8859-7), Hebrew (ISO8859-8), Cyrillic (ISO8859-5). See 'doc/QUAIL' for the usage of this system. As for Hebrew, right-to-left writing is supported. See 'doc/R2L'.

Mule is also freely available; the quotation is from the README.Mule file that accompanies the Mule distribution on the Internet.

Multiple-language support (probably based on Unicode) will eventually become part of standard Emacs, and of the Info system. The latest version of GNU Emacs (Emacs 19) can already support eight-bit character display using the ISO Latin-1 character set.

Structure of an Info file

Info files are built using very simple conventions; they are very nearly plain text files. (See Info file `info.info`, node `Add`.)

Nodes are the central notion: small blocks of text that fit into the network of links to make up the complete document. A single Info file may contain any number of nodes. The beginning of a node is marked by a special two-character sequence—the control character ^_ followed by a newline. On the immediately following line, the string "`Node:` *nodename*" identifies the node. This *nodename* identifier must be unique in the file; it provides the means to link other nodes to this one.

The same line contains up to three standard links: a `Next` pointer, a `Prev` pointer, and an `Up` pointer. These links define how the node fits into the main structure envisioned by the writer of the document, and provide for quick navigation analogous to "flipping the pages" of a book. For example, in the Info version of this paper, this section begins:

```
^_
File: sd93, Node: Structure, Next:
Writer, Prev: Reader, Up: Top
```

[3] fischer@math.ufl.edu and cws@math.ufl.edu on the Internet, respectively.

[4] handa@etl.go.jp, tomura@etl.go.jp, and nisikimi@etl.go.jp respectively.

`Node: Structure`
: allows the Info browser to identify this section when other sections of this document, or other documents altogether, have a link to it.

`Next: Writer`
: is a link identifying the node I expect most readers may want to read after this one.

`Prev: Reader`
: is a link identifying the node that normally precedes this one.

`Up: Top`
: means that the main *menu* that leads to this node is from a node called Top (conventionally, the beginning of Info documents is a node called Top.)

A *menu* is the next most important device for collecting pointers in Info files. It is simply a list of pointers to other nodes. Menus are most often used analogously to a table of contents (or a segment of one):

```
* Menu:

* Intro::       Introduction
* History::     History of the GNU Info
                format
* Today::       Who uses Info today?
* Structure::   Structure of an Info file
* Writer::      The writer-Texinfo files
* Experience::  Hypertext style after a
                decade
* Thanks::      Acknowledgements
```

The example shows the Info menu for this paper. There are no magic characters or other devices involved; everything is visible—the start of the menu is marked by the line "`* Menu:`", and each line with a * in the left margin thereafter is an entry in the menu—a pointer to one of the nodes of this paper. There are other possible fields in Info menus (for instance, to point to nodes of other documents); this style is the simplest. The first text field of each line is the link; the trailing field simply provides more of a clue for the (human) reader about the topic of each node.

This paper, a relatively simple document, has only the menu shown above. In general, however, Info files contain at least one menu per "level" of node, to link each higher-level node to all its subordinates (where "levels" correspond directly to the hierarchical levels of a book: chapters, sections, subsections and so on). Moreover, you can use additional menus anywhere that a collection of pointers seems appropriate to the context at hand.

Cross references are the most general-purpose kind of link in Info files; they may be embedded anywhere in running text. As for other links, cross references are marked by simple conventions in the text. For example, the cross reference at the beginning of this section looks like this in the Info version of the paper:

```
(*note Adding a New Topic to "DIR":
(info.info)Add.)
```

The string "*note" alerts the Info browsing programs that a link follows; the link itself is to another file in this case—the string "(info.info)Add" identifies the target as node Add in the file info.info, the main online documentation for the Info system as distributed by the FSF.

These are the highlights of an Info file's structure, but this is not an exhaustive treatment. I will only mention two other important facilities: *index entries*, corresponding directly to indices in a printed book, establish an alternate collection of links; and a *tag table* at the end of an Info file acts as a quick index to the locations of nodes within the file, thus helping the Info browsing programs to navigate quickly.

Inverse links are only partially represented in an Info file itself, and even that is only if an author chooses to follow recommended usage. In that case, one node's Prev field is the inverse link corresponding to the previous node's Next field, and each node's Up field is the inverse link corresponding to one of the links in a containing node's menu. However, even this is a matter of convention, and there are no conventional inverses to cross references, index entries, or to links from "extra" menus. Nevertheless, there is no risk that the reader may become lost in a maze of links; the Info readers themselves maintain a history of the nodes visited in a session, and the reader can retrace a path by successively using the Info command 'l' (return to the last node visited).

The writer: Texinfo files

You can write Info files directly, if you wish. But most writers use the Texinfo markup language, or something similar. Using Texinfo, you first prepare a source file that describes your document's structure (as well as including its content). Then you can process that file with TeX to make a printed book, or with a formatter like makeinfo to build the Info file.

Texinfo is an intentional markup language—that is, rather than specifying formatting details (such as 12-point boldface Times, flush left), you specify what you intend (for instance, a chapter heading). You can use many of the commands simply to describe the structure of a printed book: @title, @chapter, @section, @example, and the like. However, certain commands are available to define the node, pointer, and menu structure of a document. These commands correspond to the Info structural elements described above:

@node

Mark the beginning of a node with this command (typically, followed immediately by a sectioning command). You can simply use @node to declare the node name, and let the Info formatting program fill in the node pointers. For example, one of the nodes in this paper is coded like this in the source file:

```
@node Structure
@unnumbered Structure of an Info
file
```

(@unnumbered is a sectioning command—in both the printed book and the Info file, it makes a heading out of the remaining text on the line.)

Alternately, if you prefer—or if you want to use unconventional node pointers which cannot be generated automatically—you can specify the node pointers explicitly, as in

```
@node Structure, Writer, Reader,
Top
@unnumbered Structure of an Info
file
```

The additional comma-separated fields (in order) are used to fill in the Next, Prev, and Up pointers in the Info file.

@menu

Menus appear only in the Info file. They are analogous to a table of contents,[5] but they appear much more often: usually there is at least one menu for every section with subordinate sections. In the source document, you can specify a menu as text between a @menu command and an @end menu command; in between, you simply write the lines of the menu as they are to appear in the Info file.

@ref

There are several related commands to generate slightly different kinds of cross reference. @ref is the most general-purpose case. In the simplest instance, writing @ref{*nodename*} generates a cross reference to the node named *nodename*. There are additional optional fields (separated by commas) you can use to specify the full description of the nodename, a different document where the target node resides, and the name of that document as a printed work. (A rarely used field allows you to use a different full description in the Info file than in the printed form of the cross reference. This field is in the second position of the @ref argument when present, so it most often shows up as two successive commas, to permit specifying the later optional fields.)

The cross reference that concludes this section is an example of using @ref to specify a node in another document. This is how it looks in the source file:

```
For a comprehensive explanation of
the Texinfo markup language, start
with @ref{Overview, , Overview of
Texinfo, texi.info, Texinfo: the
GNU Documentation Format}, by Robert
J. Chassell and Richard M. Stallman.
```

[5] Texinfo can also automatically generate the ordinary sort of table of contents, for the printed form of a book.

For a comprehensive explanation of the Texinfo markup language, start with section "Overview of Texinfo" in *Texinfo: the GNU Documentation Format*, by Robert J. Chassell and Richard M. Stallman.

Hypertext style after a decade

In older Info documents, you can still find elaborate experiments involving non-linear text; documents with two disconnected halves reached by different means, or links that define two alternative "cycles" through a document (somewhat like Julio Cortazar's paper hypertext novel from 1963, *Rayuela* [6]).

This is evidence that Info writers went through some of the same soul-searching found in today's discussions of hypertext: what implications does this new non-linear documentation form have for writing style and presentation? How can we deal with the new level of control on the part of the user?

Over the intervening decade, Texinfo documents have settled on a style and structure remarkable in how *little* it differs from the style of good printed technical documentation. There is a clear linear structure unifying the whole; there are helpful introductory discussions preceding more detailed explanations; and of course there are good indices, and rich cross references.

I believe this is ultimately because non-linear access to books—especially technical books—is in fact nothing new. Some readers, of course, are interested enough to read such books cover-to-cover. Many others, however, are more apt to look over some of the introductory material (or skip it entirely, for books in areas they are already conversant with) for a general orientation, then jump directly to chapters on especially interesting subjects. If it turns out a critical explanation was missed, the index, or in the case of especially well-organized books a timely cross reference, can be relied on to light the way. In any case, after the initial reading, all these kinds of readers—even the cover-to-cover crew—are likely to use technical works for reference. This hardly ever involves linear access! Again, the structures most like hypertext in the linear book—the table of contents, index or indices, and embedded cross references—are exploited to lead directly to the relevant section of the book.

Nevertheless the linear organization has always been useful. To the writer, because it makes it possible to plan and monitor coverage of the subject. To the reader, because the linearity makes the book's organization more predictable. In fact, linearity constitutes yet *another* support for "hypertext" access! If (having skipped ahead) you discover yourself in deep water, you know roughly where to turn for supporting explanations: *back*, to the introductory matter that must have preceded the overly technical explanation.

The dual nature of Texinfo—the fact that a single file can generate both the online hypertext and the printed versions of documentation—has helped us discover this, by encouraging us to think of the hypertext and the printed book together. (This also tends to yield printed manuals with better cross references and indices, since these correspond directly to important links in the online hypertext!)

Incidentally, recall that this dual nature was motivated by user demand for printed books, even *after* hypertext Info files were already in widespread use! This constitutes another valuable lesson: printed books are not obsolete. Hypertext is an exceedingly convenient way to read—given an elaborate technological substructure. Books are complementary; they are still another exceedingly convenient way to read—whether or not the technological substructure is available.

In short, the established structure of books remains effective for hypertext; and the same writing practices lead to good printed technical documentation, and to good hypertext.

Acknowledgements

For the Texinfo system, thanks to Richard Stallman, its inventor; to his collaborators at the Free Software Foundation, notably Robert Chassell, Brian Fox, and Karl Berry; to Dave Gillespie for his enhancements to Info mode in Emacs; to the many users of free software who have contributed bug reports and

[6] *Hopscotch* in the English translation.

improvements over the years; and to the writers of GNU and other documentation in Texinfo format, who have explored the possibilities of the medium.

For this paper, particular thanks to Robert Chassell and Jim Kingdon for many fruitful discussions; to David Vinayak Henkel-Wallace, Stu Grossman, Jym Dyer, and Don Bashford for historical clues; and to the respondents to my online survey.