Word Processing with GNU/Linux
Part 1: Document Processors and Output Formats

Ben Pfaff <pfaffben@msu.edu>

8 Jan 2000

Contents

1 Introduction

If you're accustomed to word processing in GUI environments such as those provided by Microsoft Windows, you will find the facilities provided by Unix-like environments unfamiliar, even alien. But in fact, GNU/Linux tools for word processing are just as powerful, or more so, than their Windows equivalents.

The aim of this article is to provide an overview of the free tools available under GNU/Linux for word processing. No particular tool will be emphasized. Instead, the purpose is to introduce the reader to the concepts behind each of the most popular tools, and to explain their respective strengths and weaknesses in various tasks.

This is the first part of two in a series. Whether the second part will be written depends on the reception of this part, so stay tuned.

2 Document processors

In GNU/Linux, the most popular word processing tools all fall into the category of "document processors." These tools all take a file written in a special document language as input, and output a processed version of the document in one of several formats.

Document processor input files are typically written using a text editor such as `vi` or `Emacs`. The languages used to write the input files vary greatly in style and complexity from one document processor to another. Some document languages (TEX, `nroff`) are Turing complete, that is, they are complete programming languages specialized for writing documents, whereas others (SGML, nroff `-man`) are intentionally circumscribed to ease the construction of tools for translation.

Output formats also vary between document processors. Some offer only a single output format, but most tools these days offer a choice of at least a few

formats. Output formats are covered in more detail later in this article.

The sections below discuss individual document processors.

2.1 TEX

TEX is arguably the most important document processor available today. Other than `nroff` (see below), it is also the most popular.

Donald E. Knuth began designing TEX in 1978 in response to the declining quality of typesetting in his seminal textbook series, *The Art of Computer Programming*. The language has evolved since, but its design is now frozen, and no more changes will be made. For additional information on TEX's history, see the Jargon File [1].

Besides a document processor, TEX includes a complete font-generation system called METAFONT. Fonts in METAFONT are **parameterized**, meaning that their appearance can be modified based on a number of user-controllable settings. For instance, Knuth's own Computer Modern typeface, used in this article, has over 50 parameters. It is not normally necessary to set all these parameters by hand, but the available versatility can occasionally come in handy.

Distinctive features of TEX include its macro-based programming environment and its strong support for typesetting equations.

Input files to TEX typically have a `.tex` extension. TEX writes output files in its own `.dvi` format. Modern versions can also output PDF files. Use the `tex` program to translate TEX to DVI, or `pdftex` to translate to PDF.

There is an enormous amount of third-party support for TEX. As a result, TEX "distributions," which are to TEX as GNU/Linux distributions are to the Linux kernel, have been developed. For GNU/Linux systems, the most popular of these is currently teTEX, which includes a large number of useful TEX "packages."

It is possible to use raw TEX code to produce documents, but most users use a TEX document preparation system. By simplifying and specializing TEX, such systems allow the user to worry less about typesetting and more about writing. These systems exist for general-purpose typesetting and for specialized needs, such as the styles used by particular magazines or journals. The most commonly encountered TEX document preparation systems are LATEX and Texinfo, discussed in more detail below.

There is little documentation for plain TEX on the net. Knuth's *TEXbook*[2] is the authoritative guide.

2.1.1 LATEX

LATEX is a system designed for writing technical documents, including articles, reports, and books. It is also useful for writing less formal documents such as letters and even the occasional slide presentation. This article was written using LATEX.

Some of the features of LATEX, in addition to those provided by TEX itself, include strong support for figures and tables, tables of contents, indexes, graphics, and bibliographies. Due to its popularity, there are numerous extension packages available for use with LATEX: if LATEX doesn't support what you're trying to do directly, it is probably possible to find a package to help you do it.

LATEX is squarely aimed at paper publishing. It provides little support for online publishing in HTML format, although it is easy to produce PDF files. Add-ons for web publishing are available, but their output often needs to be edited by hand to provide the same quality as a tool designed for HTML output.

LATEX input files typically have `.tex` extensions. Unfortunately, this is the same as plain TEX. They can be distinguished from plain TEX by their first line, which typically contains a `\documentclass` or (old-style) `\documentstyle` directive. Use `latex` to translate LATEX to DVI, or `pdflatex` to translate to PDF.

Basic documentation for LATEX is included with the system, but the most authoritative guide to using LATEX (besides the source code, of course!) is the book *LATEX: A Document Preparation System*[3] by LATEX's author.

2.1.2 Texinfo

Texinfo is the GNU Project's documentation system. Its goals are different from LATEX's. Whereas LATEX is designed for producing paper documents such as articles and books, Texinfo is designed to provide both online and paper documentation for GNU software.

Texinfo documents are designed for easy parsing by software programs besides TEX itself. Specifically, the Free Software Foundation supplies a program called `makeinfo` for translating Texinfo documents to Info format. In turn, Info format is designed for online viewing with an Info viewer (see 3.6 below).

Tools also exist to translate Texinfo to HTML. The

best of these is currently `texi2html`, but later versions of `makeinfo` also support HTML output.

Texinfo input files have `.texinfo`, `.texi`, or `.txi` extensions. They can be identified by their first line, typically `\input texinfo`, or by the large number of `@`-signs scattered around their contents. `tex` is used to translate Texinfo to DVI, or `pdftex` for translation to PDF.

Full documentation for Texinfo is included within its distribution.

2.2 nroff

`nroff` is the oldest document processor still in common use in GNU/Linux. It was originally written in the mid-1970s in PDP-11 assembler by Joseph Ossanna. For additional history, see the Jargon File[1].

`nroff` is used under GNU/Linux and other Unix-like systems as the basic system documentation tool, used for formatting manpages. The documentation for some important GNU/Linux programs, among them XFree86, is also written using `nroff`.

Strictly speaking, `nroff` refers to a particular program that reads a document language and produces output in a plain text format, useful for online viewing with a text viewer such as `more` or `less`. Similarly, `troff` is a program that reads the same document language and produces a device-independent output format designed for easy translation into printer-specific formats.

The GNU project's `nroff` implementation, called `groff`, is more versatile than older versions. It can produce output in several formats: PostScript, TEX DVI, plain text, HP PCL (supported by HP LaserJet printers among others), HTML, and on-screen previews for X.

Analogous to TEX's document preparation systems, `nroff` has macro packages for different purposes. These are referred to by the command line option passed to `nroff` in order to select them, so on a typical system one would find `-man` for manpages, `-ms` for "manuscripts," and so on.

Document files for `nroff` typically end in an extension that is a single digit, indicating the Unix manual section that it should be installed in. `nroff` documents are also seen with `.nroff` and `.troff`, or extensions based on the macro package used. `nroff` files can also be identified based by noticing the large number of periods in the first column in a typical document file.

`nroff` is poorly documented. However, this may be okay, since it has few new users now that TEX has achieved wide acceptance. Except for a few diehard users, little new documentation is being written using `nroff`. The except is manpages for new programs, which fortunately use a small subset of `nroff` syntax, again to ease parsing by other programs.

Several preprocessors are included with `nroff`. These preprocessors take `nroff` source and pass through most of it unchanged. They recognize special directives and translate them into `nroff` commands. Common preprocessors include `eqn`, for typesetting equations; `tbl`, for tables; `pic`, for drawing pictures; `soelim`, for handling include files; and `refer`, for bibliographic citations.

2.3 SGML

SGML, the Standard Generalized Markup Language, and closely related XML, the Extensible Markup Language, are the fastest-growing documentation format in the GNU/Linux world. SGML and XML (hereafter, simply "SGML") are both formats designed to be easily parseable by programs.

SGML is not a document language in itself. Instead, it describes a standard format for specifying document languages, called a **document type definition**, or DTD. These document languages are sometimes called **SGML applications**. In contrast, individual documents are written against particular DTDs.

Because of the existence of these DTDs, programs can be written to deal with any SGML application, not just particular DTDs. For instance, several tools exist for validating SGML documents against their DTDs and analyzing their structure. (However, tools for translation of SGML into other formats must be customized for the particular DTD in use.)

SGML document files typically have `.sgml` extensions or extensions based on the name of their associated DTD. SGML DTDs can be identified based the first line of the SGML file, starting with `<!DOCTYPE`. In addition, SGML files contain lots of < and > characters.

Some popular SGML applications are described in more detail in the sections below.

2.3.1 HTML

The various versions of HTML are by far the most popular SGML application. However, HTML is too primitive a format for use in general word proc-

cessing. For instance, it lacks direct support for footnotes, indexes, figures, mathematical typesetting, columns of text, hyphenation, tables of content, bibliographies, and many other features expected of a serious word processing tool. As a result, HTML is rarely used directly for word processing. Instead, it is used as an output format of other tools better suited for word processing.

2.3.2 Docbook

Docbook is an SGML DTD for technical documentation. It provides a feature set reminiscent of Texinfo, which is unsurprising since their purposes are the same.

An increasing number of programs provide their documentation in Docbook format. Tools exist for converting Docbook documents into LaTeX, HTML, PostScript, `nroff`, and plain text format, possibly others as well.

2.3.3 Linuxdoc

Linuxdoc is the SGML DTD used by the Linux Documentation Project for writing Linux documentation. It is also used by other projects and organizations. Linuxdoc is a significantly simpler format than Docbook.

Tools exist to convert Linuxdoc documents into at least LaTeX, HTML, Texinfo, LyX, RTF, and plain text formats.

2.3.4 Debiandoc

Debiandoc is an SGML DTD used by some Debian projects for writing Debian manuals. Debiandoc is a format even simpler than Linuxdoc.

Tools exist to convert Debiandoc documents into at least LaTeX, HTML, Texinfo, plain text, and text with overstrikes (see section 3.7 below for explanation).

3 Output formats

So you have a carefully written document in whatever document language you ended up choosing. You've run it through your document processor, and it processed cleanly. Now you have... some output format. Exciting, huh?

Oh, you wanted to *do* something with your output? What you can do with the output of your document processor depends on what format the output ended up in. Let's take a look at the most common output formats:

3.1 PostScript

PostScript is a programming language designed and largely controlled by Adobe Systems. It happens to be particularly good at putting marks on paper. PostScript is understood directly by high-end laser and inkjet printers, among others. Implementations also exist as software products.

There are three main varieties of PostScript: Level 1, Level 2, and Level 3. Level 1 is found in very old printers. Level 2 is the current standard in printer and software products. Level 3 is the newly anointed successor to Level 2, but due to Adobe's increasingly proprietary attitude toward PostScript, it is unlikely to ever achieve the market penetration of Level 2.

Products that support one level of PostScript can handle documents designed for lower levels, but not those which use features from higher levels. As a result, PostScript Level 1 documents are the most generic and can print on any PostScript printer.

PostScript files typically begin with a line of the form `%!PS-Adobe-`$x.y$, where x and y are version numbers.

A subtype of PostScript document is Encapsulated PostScript, or EPS. EPS files are designed specifically to be embedded in other documents. They are often used as figures within larger documents. EPS figures are typically vector-based so that they can be scaled at high quality.

GNU/Linux has strong support for PostScript. A few of the most useful utilities for PostScript are described below.

3.1.1 Ghostscript

Ghostscript is a tool for executing PostScript code. It can output the equivalent in a particular printer language such as PCL, or display an on-screen preview. Ghostscript also includes utilities for converting PostScript to PDF and vice versa, converting PostScript files to plain text (with necessarily poor quality), and for more esoteric purposes.

Sites with non-PostScript printers usually install Ghostscript between the printer and the print queue to allow PostScript files to be conveniently printed.

3.1.2 gv

'gv' is a handy X-based front-end to Ghostscript. It makes previewing printouts much more pleasant than using Ghostscript directly. Recommended.

'gv' can also display PDF files (see below).

3.1.3 psutils

PostScript documents which are structured according to Adobe's Document Structuring Conventions (DSC) can be manipulated by programs. `psutils` includes programs to extract particular pages from PostScript files, rearrange pages, perform n-up and booklet printing, combine and split files, and more. It also has programs to fix up the PostScript output of various programs which are known to be broken in particular ways.

3.2 PDF

PDF is Adobe's Portable Document Format. PDF is closely related to PostScript, but it is optimized for online viewing rather than for printing. Tools exist to convert PostScript to PDF and vice versa; see the previous section for details.

PDF has special features for online viewing: Tables of contents can be displayed alongside document text; hyperlinks can be made between related sections; and PostScript figures can be replaced by GIFs or JPEGs for faster display.

Unfortunately, even with these concessions to online users, PDF is still a fundamentally flawed format for online viewing. It forces the user to adapt to the format of the printed work, instead of adapting the work to the user's environment.

PDF also has provisions for encrypted documents. The usefulness of this feature in practice is questionable.

3.2.1 xpdf

`xpdf` is a standalone viewer for PDF. In its international viewer, it supports encrypted PDFs as well. `xpdf` can convert PDFs to PostScript for printing.

Note that 'gv', described in the previous section, also supports viewing and printing PDF files.

3.3 DVI

DVI is TEX's standard output format, though some other tools (such as GNU Groff) can also output it now.

DVI is almost as good as PDF for online viewing. Its only shortcoming is the lack of hyperlinks, but its great speed of display compared to PDF is a big advantage. Pages displayed in `xdvi` refresh almost instantly, but it can take a few seconds in `gv` or `xpdf`. DVI files are also smaller than the corresponding PDF files, typically one-twentieth to one-third of their size.

For online viewing, DVI is flawed in the same way as PDF.

A few of the most commonly used DVI utilities are described below.

3.3.1 xdvi

The `xdvi` program is used to view the contents of DVI files under X. When used on a system that has Ghostscript installed (see above), it can even display PostScript figures included as part of TEX documents.

3.3.2 dvips

Converts DVI files to PostScript format for printing.

3.3.3 dvihp

Converts DVI files to HP's PCL format for printing.

3.4 RTF

RTF is Rich Text Format, and it is somewhat of an enigma. RTF was originally designed by Microsoft. Despite this history, RTF files are in an ASCII format, not binary, and they are somewhat readable with a text viewer. RTF appears to be an open, documented format.

RTF is primarily an output format. However, there is support for reading and writing RTF files in at least two products: Pathetic Writer and Ted.

3.4.1 Pathetic Writer

Pathetic Writer is part of the Siag Office suite, which also contains a word processor and an animation package. It is an X-based word processor with support for the usual things expected of such.

3.4.2 Ted

Ted is a standalone X-based editor designed for use with RTF.

3.5 HTML

HTML format is a popular choice for online viewing, since HTML browsers are available for every modern computing platform. HTML has been chosen as the documentation format for numerous projects, including the Debian project.

HTML can be read and written by many programs, but its utility as an input format for documentation or general word processing is limited. For more information, see 2.3.1 above.

3.6 Info

Outside the GNU Project, Info is controversial. Some say that it should be replaced by HTML. Its proponents argue that Info is more useful than HTML for online viewing, since Info documents include a full index and their viewers support full-text search for entire documents, not just individual sections, which are features lacking in HTML browsers. Info also has next, previous, and 'up' pointers from each page, which eases browsing considerably in many situations.

The most popular browsers for Info format are Emacs and Info.

3.6.1 Emacs

All flavors of Emacs derived from GNU Emacs are able to browse Info format.

3.6.2 Info

Info is the standalone GNU browser for Info format. Its interface is strongly reminiscent of Emacs. As a result, those who don't like Emacs don't like Info, either.

3.7 Plain text

Plain text is just that. Plain text, in a file.

But there are sometimes-troublesome variations, described below.

3.7.1 Character sets

The simplest, and most common, form of plain text is in more-or-less universal 7-bit ASCII. This is sufficient for English prose, but not for most other languages.

The next most common format is ISO Latin-1 format, an 8-bit format which specifies additional characters for use in languages other than English. ISO Latin-1 is sufficient for writing languages used in western Europe as well as English.

Additional 'national character sets' exist as well, but these are not as common.

Unicode is a character set which contains all the characters in every human language. Unlike the other character sets discussed here, which are 8-bit, Unicode characters are 16 bits in width. It is being slowly adopted across computerdom, including GNU/Linux. You may encounter it, especially in its UTF-8 coding format, which is used for coding 16-bit characters in contexts where 8-bit characters are expected.

GNU `recode` is useful for translating between these character sets and others as well (such as various flavors of EBCDIC). See its manual [4] for more details about character sets.

3.7.2 Overstrikes

On a dot matrix printer, bold and underlined text can be produced by overstriking using backspace characters. Some online text viewing tools such as `less` can also interpret these backspace sequences, which is how many Linux manpages are displayed on a text console complete with colored text to represent bold and underlines.

The `col`, `colcrt`, and `ul` utilities are useful for dealing with files that contain overstrikes.

3.8 Device-specific formats

There are many formats which are more-or-less specific to particular devices. There is often little tha

The most common of these is HP PCL (Printer Control Language). Variants of PCL can be found on many inkjet and laser printers, but incompatibilities between implementations are common, so it is often better to consider each of these printers as having a different command set.

Another example is the Epson/IBM command set for driving dot-matrix line printers. This command set is also rather different between manufacturers and even between particular models from one manufacturer.

More and more printers are now using completely proprietary command sets. Most of this new breed of printers are "dumb framebuffers"; that is, they have little or no intelligence, simply spraying pixels

on paper where indicated. These have no internal fonts or support for other drawing primitives.

Such printers are commonly known as "WinPrinters" due to their proliferation under the Microsoft Windows platform. However, they are not typically Windows specific[1], although sometimes documentation on their command sets is not available. Ghostscript (see section 3.1.1) supports a number of these printers under GNU/Linux.

4 Intermission

In the next part we'll discuss the use of graphics and figures in document processors, how to construct your own document processing tools, and how to tie it all together. We'll also take a brief look at how GUI-based WYSIWYG tools can help to construct documents.

References

[1] Eric S. Raymond. *The Jargon File*. Online at `http://sagan.earthspace.net/jargon`.

[2] Donald E. Knuth. *The TEXbook*. Addison-Wesley 1988. ISBN 0-201-13448-9.

[3] Leslie Lamport. *LATEX, A Document Preparation System: User's Guide and Reference Manual*. 2nd ed. Addison-Wesley 1994. ISBN 0-201-52983-1.

[4] François Pinard, et al. *The GNU Recode Manual*. Current version 3.5 at time of this writing. Online at `ftp://ftp.gnu.org/pub/gnu/recode`.

[1] Historically, there did exist a short-lived line of printers which implemented the Windows GDI graphics-drawing API.