groff

The GNU implementation of `troff`
Edition 1.19.3
Spring 2006

by Trent A. Fisher
and Werner Lemberg (bug-groff@gnu.org)

Table of Contents

1 Introduction 1
1.1 What Is `groff`? 1
1.2 History 1
1.3 `groff` Capabilities 3
1.4 Macro Packages 4
1.5 Preprocessors 4
1.6 Output Devices 4
1.7 Credits 5

1 Introduction

GNU `troff` (or `groff`) is a system for typesetting documents. `troff` is very flexible and has been used extensively for some thirty years. It is well entrenched in the UNIX community.

1.1 What Is `groff`?

`groff` belongs to an older generation of document preparation systems, which operate more like compilers than the more recent interactive WYSIWYG[1] systems. `groff` and its contemporary counterpart, TEX, both work using a *batch* paradigm: The input (or *source*) files are normal text files with embedded formatting commands. These files can then be processed by `groff` to produce a typeset document on a variety of devices.

`groff` should not be confused with a *word processor*, an integrated system of editor and text formatter. Also, many word processors follow the WYSIWYG paradigm discussed earlier.

Although WYSIWYG systems may be easier to use, they have a number of disadvantages compared to `troff`:

- They must be used on a graphics display to work on a document.
- Most of the WYSIWYG systems are either non-free or are not very portable.
- `troff` is firmly entrenched in all UNIX systems.
- It is difficult to have a wide range of capabilities within the confines of a GUI/window system.
- It is more difficult to make global changes to a document.

> "GUIs normally make it simple to accomplish simple actions and impossible to accomplish complex actions." –Doug Gwyn (22/Jun/91 in `comp.unix.wizards`)

1.2 History

`troff` can trace its origins back to a formatting program called `runoff`, written by J. E. Saltzer, which ran on MIT's CTSS operating system in the mid-sixties. The name came from the use of the phrase "run off a document", meaning to print it out. Bob Morris ported it to the 635 architecture and called the program `roff` (an abbreviation of `runoff`). It was rewritten as `rf` for the PDP-7 (before having UNIX), and at the same time (1969), Doug McIllroy rewrote an extended and simplified version of `roff` in the BCPL programming language.

The first version of UNIX was developed on a PDP-7 which was sitting around Bell Labs. In 1971, the developers wanted to get a PDP-11 for

[1] What You See Is What You Get

further work on the operating system, and to justify the cost, proposed the development of a document formatting system for the AT&T patents division. This first formatting program was a reimplementation of McIllroy's `roff`, written by J. F. Ossanna.

When they needed a more flexible language, a new version of `roff` called `nroff` ("Newer `roff`") was written. It had a much more complicated syntax, but provided the basis for all future versions. When they got a Graphic Systems CAT Phototypesetter, Ossanna wrote a version of `nroff` that would drive it. It was dubbed `troff`, for "typesetter `roff`", although many people have speculated that it actually means "Times `roff`" because of the use of the Times font family in `troff` by default. As such, the name `troff` is pronounced 't-roff' rather than 'trough'.

With `troff` came `nroff` (they were actually the same program except for some '`#ifdef`'s), which was for producing output for line printers and character terminals. It understood everything `troff` did, and ignored the commands which were not applicable (e.g. font changes).

Since there are several things which cannot be done easily in `troff`, work on several preprocessors began. These programs would transform certain parts of a document into `troff`, which made a very natural use of pipes in UNIX.

The `eqn` preprocessor allowed mathematical formulæ to be specified in a much simpler and more intuitive manner. `tbl` is a preprocessor for formatting tables. The `refer` preprocessor (and the similar program, `bib`) processes citations in a document according to a bibliographic database.

Unfortunately, Ossanna's `troff` was written in PDP-11 assembly language and produced output specifically for the CAT phototypesetter. He rewrote it in C, although it was now 7000 lines of uncommented code and still dependent on the CAT. As the CAT became less common, and was no longer supported by the manufacturer, the need to make it support other devices became a priority. However, before this could be done, Ossanna died by a severe heart attack in a hospital while recovering from a previous one.

So, Brian Kernighan took on the task of rewriting `troff`. The newly rewritten version produced device independent code which was very easy for postprocessors to read and translate to the appropriate printer codes. Also, this new version of `troff` (called `ditroff` for "device independent `troff`") had several extensions, which included drawing functions.

Due to the additional abilities of the new version of `troff`, several new preprocessors appeared. The `pic` preprocessor provides a wide range of drawing functions. Likewise the `ideal` preprocessor did the same, although via a much different paradigm. The `grap` preprocessor took specifications for graphs, but, unlike other preprocessors, produced `pic` code.

James Clark began work on a GNU implementation of `ditroff` in early 1989. The first version, `groff` 0.3.1, was released June 1990. `groff` included:

- A replacement for `ditroff` with many extensions.
- The `soelim`, `pic`, `tbl`, and `eqn` preprocessors.
- Postprocessors for character devices, PostScript, TEX DVI, and X Windows. GNU `troff` also eliminated the need for a separate `nroff` program with a postprocessor which would produce ASCII output.
- A version of the '`me`' macros and an implementation of the '`man`' macros.

Also, a front-end was included which could construct the, sometimes painfully long, pipelines required for all the post- and preprocessors.

Development of GNU `troff` progressed rapidly, and saw the additions of a replacement for `refer`, an implementation of the '`ms`' and '`mm`' macros, and a program to deduce how to format a document (`grog`).

It was declared a stable (i.e. non-beta) package with the release of version 1.04 around November 1991.

Beginning in 1999, `groff` has new maintainers (the package was an orphan for a few years). As a result, new features and programs like `grn`, a preprocessor for gremlin images, and an output device to produce HTML output have been added.

1.3 `groff` Capabilities

So what exactly is `groff` capable of doing? `groff` provides a wide range of low-level text formatting operations. Using these, it is possible to perform a wide range of formatting tasks, such as footnotes, table of contents, multiple columns, etc. Here's a list of the most important operations supported by `groff`:

- text filling, adjusting, and centering
- hyphenation
- page control
- font and glyph size control
- vertical spacing (e.g. double-spacing)
- line length and indenting
- macros, strings, diversions, and traps
- number registers
- tabs, leaders, and fields
- input and output conventions and character translation
- overstrike, bracket, line drawing, and zero-width functions
- local horizontal and vertical motions and the width function

- three-part titles
- output line numbering
- conditional acceptance of input
- environment switching
- insertions from the standard input
- input/output file switching
- output and error messages

1.4 Macro Packages

Since `groff` provides such low-level facilities, it can be quite difficult to use by itself. However, `groff` provides a *macro* facility to specify how certain routine operations (e.g. starting paragraphs, printing headers and footers, etc.) should be done. These macros can be collected together into a *macro package*. There are a number of macro packages available; the most common (and the ones described in this manual) are '`man`', '`mdoc`', '`me`', '`ms`', and '`mm`'.

1.5 Preprocessors

Although `groff` provides most functions needed to format a document, some operations would be unwieldy (e.g. to draw pictures). Therefore, programs called *preprocessors* were written which understand their own language and produce the necessary `groff` operations. These preprocessors are able to differentiate their own input from the rest of the document via markers.

To use a preprocessor, UNIX pipes are used to feed the output from the preprocessor into `groff`. Any number of preprocessors may be used on a given document; in this case, the preprocessors are linked together into one pipeline. However, with `groff`, the user does not need to construct the pipe, but only tell `groff` what preprocessors to use.

`groff` currently has preprocessors for producing tables (`tbl`), typesetting equations (`eqn`), drawing pictures (`pic` and `grn`), and for processing bibliographies (`refer`). An associated program which is useful when dealing with preprocessors is `soelim`.

A free implementation of `grap`, a preprocessor for drawing graphs, can be obtained as an extra package; `groff` can use `grap` also.

There are other preprocessors in existence, but, unfortunately, no free implementations are available. Among them are preprocessors for drawing mathematical pictures (`ideal`) and chemical structures (`chem`).

1.6 Output Devices

`groff` actually produces device independent code which may be fed into a postprocessor to produce output for a particular device. Currently, `groff`

has postprocessors for PostScript devices, character terminals, X Windows (for previewing), TEX DVI format, HP LaserJet 4 and Canon LBP printers (which use CAPSL), and HTML.

1.7 Credits

Large portions of this manual were taken from existing documents, most notably, the manual pages for the `groff` package by James Clark, and Eric Allman's papers on the '`me`' macro package.

The section on the '`man`' macro package is partly based on Susan G. Kleinmann's '`groff_man`' manual page written for the Debian GNU/Linux system.

Larry Kollar contributed the section in the '`ms`' macro package.