# Groff Mission Statement
## 2014

As the most widely deployed implementation of troff in use today, groff holds an important place in the Unix universe. Frequently and erroneously dismissed as a legacy program for formatting Unix manuals (manpages), groff is in fact a sophisticated system for producing high-quality typeset material, from business correspondence to complex, technical reports and plate-ready books. With an impressive record for backward compatibility, it continues to evolve and play a leading role in the development of free typesetting software.

Future development will focus on groff's typesetting capabilities, the goal being to improve automatic formatting of documents with no corresponding loss of user control over typographic details. In addition to the core binary, where much of the work will be done, the system comprises a macro language, pre-built macro sets, pre- and post-processors, and drivers for outputting to various formats. The guiding principle for changes to these parts will be to improve and extend, not alter, especially with respect to established usage.

A key part of the project's mandate will be to redress issues that have historically discouraged widespread adoption of groff. Issues include the perceived superiority of TEX for general-purpose typesetting, the long and somewhat unfortunate association with manpages, and the difficulty of mastering groff's low-level typesetting requests.

Broadly speaking, development will be channelled into two areas:

- core groff
  - the typesetting backend and low-level formatting requests
- program usage
  - general-purpose macro sets and manpages

Improvements and enhancements to other areas of the system will continue to form a routine part of groff's growth, with active support for contributions provided by a team of groff mavens.

## Core groff

### The typesetting backend

Groff currently uses a greedy algorithm to format paragraphs one line at a time, but the Knuth-Plass algorithm, implemented inTEX, Heirloom troff and elsewhere, optimizes linebreaks throughout whole paragraphs and achieves a more uniform typographic grey. One of the most exciting challenges facing groff will be implementing an improved formatting algorithm, the most likely candidate being Knuth-Plass. It's a big job, but with groff's clean codebase and helpful community of experts, it is hoped additional developers will step forward and contribute their programming skills.

Equally important will be instituting native support for TrueType, OpenType and other non-Type1 PostScript fonts, and improving Unicode support.

### Low-level formatting requests

Supplementary low-level formatting requests will continue to be added, and the behaviour of some existing requests reviewed, with care taken to maintain backward compatibility whenever modifications are deemed worthwhile.

At the request level, groff's use of integer arithmetic and linear evaluation of expressions hearkens back to the stone age. While not top-priority, these and other historical encumbrances (read "annoyances") will be addressed—again, with a watchful eye toward backward compatibility.

## Program usage

### General-purpose macro sets

Macro sets form the primary user interface to groff. Well-designed, well-documented sets help make groff accessible to users—new users in particular. Adoption or rejection of the program is often based on whether available sets meet user needs without requiring mastery of low-level requests. Learning curve and ease of use are equally important.

The most active area of groff development in the past decade has been in macro sets, with **mom** alone adding over 13,000 lines of code and 1MB of documentation to the project. Support for existing macro sets and the design of new macro interfaces will therefore play a significant role in groff's future.

### Manpages

The need for Unix manuals to render cleanly to multiple output media favours structural rather than presentational markup, however the classical *man*(7) macros remain almost exclusively presentational. *mdoc*(7) provides a semantically superior alternative, but the use of *man*(7) is deeply rooted in Unix.

Future work on manpages will entail improving the semantic clarity of the *man*(7) macros, decoupling them as much as possible from low-level presentational requests. The aim will be to ease conversion of manpages to markup languages that do not rely on groff for display and printing, e.g. XML, while preserving the full presentational richness of manpages processed with groff.

Concurrent with work on *man*(7), *mdoc*(7) will be actively supported and its use promoted.

To summarize, this two-pronged strategy aims to foster manpage markup that:

- renders cleanly to the terminal
- respects presentational markup when output to PostScript
- allows semantic analysis
- is portable

## Looking forward, looking back

Backward compatibility with existing documents and usage will remain a top priority, as will avoiding feature-bloat and increased overheads. Groff's viability and vitality rest as much on these as on forward-looking development.

Finally, it is hoped that users of and contributors to groff will promote its use, providing unobtrusive advocacy to encourage more widespread adoption of the program, thereby increasing the pool of potential contributors and developers and furthering the cause of good typography.