μ, a simple macro package for troff

John Ankarström

28 June 2021

TABLE OF CONTENTS

Introduction

μ or *mu* is a simple macro package for troff with the following features:

1. It is designed to be easy to understand and to customize by editing the source code.
2. It makes use of the extended support for environments offered by modern troff implementations.
3. It is designed to be practically easy to use. Macros consist of a single lowercase letter.

While μ does provide macros for many common tasks, including footnotes, it is at the end of the day an idiosyncratic macro package, written to serve the author's personal needs. Users of μ are encouraged to

a) modify the source code according to their own needs, as well as
b) use built-in troff requests for some things that other packages might provide custom macros for.

All in all, μ aspires to abstract as little as possible from the underlying troff requests and registers. In its author's humble opinion, it is the ideal macro package for learning troff.

In this document, the fundamental concepts of μ are explained. The document itself also serves as a demonstration of μ. With a couple of exceptions, it uses μ's default settings. The reader is encouraged to inspect the document's source code in order to see how the macro package is used in practice.

And yes, that table of contents is automatically generated! See p. 8 for more information.

Environments

μ makes heavy use of named environments, supported by implementations such as GNU troff, Heirloom troff and Neatroff. Environments obliviate the need for many special registers that a macro package (and its user) would otherwise need to keep track of. For example, *ms* keeps track of the document's font size in the `PS` register. For the font size of headings, it has yet another register. μ has no such registers. If the user wishes to modify the default font size, he or she can simply switch to the relevant environment and set the font size as desired using regular troff requests:

```
.\" set heading font
.h
.fam H
.ps +1p
```

Troff saves the font settings in the environment, so that the next time the environment is invoked, the desired font family and point size are automatically restored.

The environments are initialized as soon as the first "block-level" macro is called. At the initialization of each environment, the default environment (0) is copied, meaning that all environment-relevant settings defined before the first macro call are applied to all μ environments. It is thus remarkably simple and intuitive to set, for example, the default font of a document:

```
.fam N
.t
Document title
.p
First paragraph.
```

In addition to the normal environment-relevant settings, μ manually associates a few special registers with the current environment:

1. `sp`, the amount of vertical space to add before an environment
2. `sq`, the amount of vertical space to add before a different type of environment
3. `ti`, the indentation of the first line in some environments (currently only `p`)

These can be set inside a given environment to control its behavior when invoked. The only exception are the margin and footnote environments (`@m`, `@f`), which are treated specially and do not support these registers.

External macros

μ defines a number of macros. Some of them are used internally by μ itself; these carry an at (@) prefix and are going to be explored later. For now, we will focus on the external macros provided by μ.

Inline macros

There is a group of macros that provide convenient inline formatting. All take three arguments: the text to be formatted, the text to be placed immediately after and the text to be placed immediately before. The inline macros are the following:

1. `"`, quotation (like "this")
2. `b`, bold font
3. `c`, constant-width font
4. `i`, italic font
5. `i`, bold italic font

For example, the following request outputs "μ.":

```
.i µ .
```

Note that `c` uses the font family and point size set in the `l` environment (see below).

Block-level macros

There is a large group of macros that provide block-level formatting:

1. `d`, centered date or text
2. `h`, heading
3. `l`, literal display (for source code)
4. `p`, paragraph
5. `q`, indented quotation
6. `s`, subheading
7. `t`, centered title

The `t` and `d` macros can be used at the beginning of a document to create a centered header:

```
.t
Document title
.d
First author
.br
Second author
.d i \" current date formatted as YYYY-MM-DD
```

In the example above, you can see that the `d` macro may be used for other things than just dates. This works because `d` displays the date only if given an argument describing the desired date format:

a) `i`: international date, like "2021-06-21"
b) `e`: English date, like "21 June 2021"

The formatted dates are defined in strings prefixed with a hash symbol (`#`): `#i` and `#e` are provided by default, but more may be added by the user.

Other macros

Finally, there are a few macros that belong to neither category:

1. `(`, begin footnote
2. `)`, end footnote
3. `w`, want space

The macros `(` and `)` create footnotes, placing a numerical reference at the place of their invocation. Both `(` and `)` take an optional argument, which is output either immediately before or immediately after the inline reference. For example, the code

```
.q
This is a quotation\c
.(
This is a footnote.
.)
```

generates the following output:

> This is a quotation[1].

If `(` and `)` are called at such a position that the collected footnotes cannot fit on the current page, μ will print the footnotes on the next page instead.

The `w` macro is an alternative to troff's `ne` request, which ensures that a given amount of space is available on the page – otherwise, a line break is issued – but unlike `ne`, which takes an exact amount of space as its argument, `w` takes a declarative specification describing the amount of desired space in terms of μ environments. For example:

```
.\" want space for...
.w s p \" a subheading of one line + a paragraph of one line
.w s pp \" a subheading of one line + a paragraph of two lines
.w ss p \" a subheading of two lines + a paragraph of one line
```

[1] This is a footnote.

Internal macros

μ's internal macros are generally not meant to be used outside of *u.tmac*. Documented in this section are the exceptions to this rule. For examples of how these macros are used in practice, see the FAQ section on page 6.

Page header and footer

The `@h`, `@f`, `@th` and `@tf` macros control the page header and footer.

At document initialization, μ automatically creates traps for `@th` and `@tf`. When sprung, `@th` and `@tf` call `@h` and `@t`, which control the text and spacing of the header and footer.

μ will avoid creating a trap for `@tf` if any trap has already been set before document initialization. This means that users can override the position of the `@tf` trap by setting it themselves. Likewise, users can redefine `@h` and `@f` to change the content of the header and footer.

Environments

The environment-related macros `(e`, `@e` and `@c` may be used in advanced documents to define custom environments.

At the present, `(e` is nothing more than a wrapper around troff's built-in `ev`, but it may eventually be redefined in order to offer compatibility with troff implementations without support for named environments.

`@e` and `@c` are equivalent to troff's `ev` and `evc`, but perform some extra work to keep track of μ's special environment variables (see *Environments*, p. 2).

Frequently anticipated questions

How do I customize the default appearance of a document?

All environment settings, like point size, font family and indentation[2], are configured with the standard troff requests. If you set the point size at the beginning of the document, before any μ macros have been called, it will be correctly set for the entire document. This works because μ's environments initially copy all their settings from 0, the default environment.

For example, if you wanted to write a document with the New Century Schoolbook font at 9 points and a vertical spacing of 12 points, you would start the document like this (before you call any μ macros):

```
.fam N
.ps 9p
.vs 12p
```

How do I customize the default appearance of a given environment?

To configure the layout and font settings of a specific environment, you can switch to that environment and use the relevant troff requests:

```
.q
.ps +1p
```

If you are merely configuring the environment without printing anything in it, you can also use `@e`:

```
.@e q
.ps +1p
```

How do I customize the default appearance of the margin text?

```
.(e @m \" margin environment
.ps +1p
```

You can also set such settings in your redefinition of `@h` or `@f`.

How do I customize the default appearance of footnotes?

```
.(e @f \" footnote environment
.ps +1p
```

How do I redefine the page header?

```
.de @h
. \" set position of header text
. sp |36p
. \" set header text
. tl 'left'center'right'
. \" set position of page text
. sp |1i
..
```

How do I redefine the page footer?

To change the position of the footer:

```
.\" set position of footer trap
.\" (must be done before any macros have been called)
.wh -1i @tf
.
```

[2] For a complete list of settings that are associated with the environment, see 5.26 *Environments* in the full GNU troff manual (`info 'groff.info)Environments'`).

To change the contents of the footer:

```
.de @f
. \" set position of footer text
. \" (must be below the footer trap)
. sp |\\n(.pu-48p
. \" set footer text
. tl 'left'center'right'
..
```

How do I define my own environments?

Environments are a feature built into troff, accessed via the `ev` request, but because μ extends environments with additional functionality, it provides special macros to be used instead of `ev`:

1. `(e`, set environment (same as `ev`)
2. `@e`, set extended environment
3. `@c`, copy environment (same as `evc`)

The `@e` macro can be used to activate any environment that supports μ's extensions (see *Environments*, p. 2). The following code configures an environment called `n` and defines a corresponding macro:

```
.@e n
. @c 0 \" copy default environment
. ps -1p
.@e
.
.de n
. br \" finish current environment
. @e n \" activate new environment
..
```

How do I prevent a section from being broken up by a page break?

Some macro packages have a concept of "keeps", sections that are kept together across page breaks. μ does not (at least yet) define any such macros by default. The simplest solution is to use troff's `ne` request:

```
.ne 7 \" break page if seven lines won't fit on this page
.\" ... some text ...
```

If you want to keep text of various styles together, you can use μ's own `w` macro:

```
.w s qq \" break page if a subheading and two lines of a quotation won't fit
.\" ... some text ...
```

How do I use alternative quotation marks?

Redefine `"`. The following code defines a Swedish quotation style:

```
.de "
\\&\\$3”\\$1”\\$2
..
```

Another option is to create new macros for alternative quotation styles:

```
.\" alternative swedish quotation styles
.de ><
\\&\\$3»\\$1«\\$2
..
.de >>
\\&\\$3»\\$1»\\$2
..
```

How do I create nested inline quotations?

Just use the backtick (`) and apostrophe (') directly in the argument to ".

If you need to change the quotation style, for example when converting a document from US to British English, you can redefine to translate backticks and apostrophes accordingly:

```
.eo
.de "
. char ` \(lq
. char ' \(rq
\&\$3\(oq\$1\(cq\$2
. char ` \\(oq
. char ' \\(cq
..
.ec
```

How do I include a table of contents in my document?

Included with the μ distribution is a package called *toc*, which includes the following files:

1. *toc.tmac*, which provides the `te` and `to` macros
2. *toc*, a script that invokes troff in three passes in order to allow a table of contents to be generated

Use `te` to register a line to be inserted into the table of contents. Use `to` to insert the lines registered by `te` into the document. Note that the argument given to `te` will be interpreted as a troff input line.

The following definitions provide a good starting point:

```
.eo
.de he
. h
\$*
. tm .the \$*\t\n%
..
.de the
. ta 0 \n(.luR
. tc .
\$*
. tc
. br
..
.ec
```

Now, instead of `h`, you may use `he` to create a heading to be included in the table of contents:

```
.he A heading
```

Wherever you want the table of contents to be inserted, you may simply invoke `to`.

Just remember to use `toc` to run your processing pipeline:

```
<example.t toc refer \| troff -Tps >example.ps
```

Thanks to the multiple passes performed by *toc*, `to` can be invoked at any place in the document, including the beginning.

The macro definition listed above is included in *ux.tmac*.

How do I create a reference to a later page?

Use the macros provided by *toc.tmac* in combination with the *toc* program. Near the beginning of your document, invoke `to`. Then, at each position you want to reference, invoke `.te` like this:

```
.te .nr &refname \n%
```

To refer to that position, interpolate the register:

```
See page \n[&refname].
```

(I prefer prefixing my references with an ampersand.)

How do I include source code without needing to escape it?

The μ distribution includes a troff preprocessor called *list*, which filters text delimited by `.l(` and `.l)`, escaping standard troff syntax.

To automatically add a `.l` request before each listing, set the *-p* (prefix) flag accordingly:

```
list -p.l
```

To transparently pass an input line to troff, prefix it with `\!`. (To disable this behavior, use the *-E* flag.)