**Reference**

**Tutorial**

**Developer**
1 Compile
2 Design
3 Implement
4 Modify
5 Misc
6 Licence

**SEARCH**

# FunnelWeb Developer Manual

## Version 3.2d (9 Jan 2000) for FunnelWeb V3.2

THIS DEVELOPER MANUAL is for hackers! Anyone who wants to bash, diddle, frob, grind, mangle, patch, poke, toggle, twiddle, zap, or generally hack FunnelWeb should at least take a look at this manual, as it explains how to compile FunnelWeb, and contains all kinds of other useful information about the FunnelWeb source code.

---

**FunnelWeb Developer Manual**

# 1 How To Compile FunnelWeb

FunnelWeb Developer Manual

# 1.1 Resources Required

This chapter describes how to obtain and compile FunnelWeb from scratch and set up for further development. In order to compile FunnelWeb, you will need:

- FTP access to the internet *OR* a FunnelWeb source distribution kit on disk.
- A computer running one of the FunnelWeb-supported operating systems *OR* lots of extra time to port FunnelWeb to a new platform.
- About five megabytes of free disk space.
- A C compiler.
- An acquaintance with the C programming language and the ability to compile and link C programs on your machine.
- Elementary systems programming knowlededge for your machine.
- About an hour.

You will *not* need any sort of system privileges to install FunnelWeb, unless you want the FunnelWeb command fw to be automatically available to everyone on your machine as well as yourself.

FunnelWeb Developer Manual

## 1.2 FTP The Source Files

The first step is to obtain a copy of the FunnelWeb source code and associated files. You can find FunnelWeb in the RossNet FTP archive at:

ftp://ftp.ross.net/clients/ross/funnelweb/

It is not clear at the time of writing whether FunnelWeb will be presented as a `".tar"` file, or as a directory tree, or both. Just sniff around and use your common sense.

# 1.3 Establish The Source Tree

At this stage, we will assume that you have somehow obtained a set of files that are supposed to be FunnelWeb, and that they are sitting on a disk on the machine on which you wish to compile FunnelWeb.

The next thing you have to do is to make sure that the FunnelWeb directory tree has been correctly unpacked. The directory tree should look like this.

```
   fwdir          - Root FunnelWeb directory.
      admin       - Administrative files.
      answers     - Answers to test suite.
      results     - For test results.
      scripts     - Test scripts.
      sources     - Source code.
      tests       - Test suite.
      web         - The FunnelWeb web.
```

The following sections describe the contents of each directory in alphabetical order. Check the contents to make sure that you have everything. Do not become fussed if your configuration is not quite as specified, as it is very easy for installation guides such as this one to go out of date.

## Admin Directory

The admin directory contains administrative files to do with licensing and such. It is also a catch-all directory for files that don't belong anywhere else. At the time of writing, it is not clear exactly what will be in the admin directory. Why not take a look?

## Answers Directory

The answers directory contains the "correct answers" to all the regression testing input files. The regression test scripts compare these files to the files generated in the results directory.

```
an01.lis ... an04.lis
ex01.lis ... ex16.lis
ex01.out ... ex10.out
ex11.tex ... ex16.tex
generate.lis
hi01.lis ... hi10.lis
```

```
hi01.out ... hi05.out
hi06a.out
hi06b.out
hi07a.out
hi07b.out
hi08.out ... hi10.out
pr01.lis ... pr10.lis
sc01.lis ... sc29.lis
tg01.lis ... tg09.lis
tg01.out ... tg09.out
wv01.lis ... wv06.lis
wv01.tex ... wv06.tex
```

## Results Directory

The results directory exists as a target directory for the output files generated by FunnelWeb during regression testing. This directory is distributed empty and should be empty at the start of regression testing. However, it is permissible for the results directory to contain files generated during a previous test run, as the regression testing scripts delete specific unwanted files before each test anyway.

## Scripts Directory

The scripts directory stores the FunnelWeb command shell scripts that are used to perform regression testing.

```
master.fws    - The master test script.
test_gen.fws  - Script to generate tricky input files.
test_l.fws    - Test FunnelWeb with +L.
test_ld.fws   - Test FunnelWeb with +L +B...
test_lo.fws   - Test FunnelWeb with +L +O.
test_lo2.fws  - Test FunnelWeb with +L +O (2 outfiles).
test_lot.fws  - Test FunnelWeb with +L +O +T.
test_lt.fws   - Test FunnelWeb with +L +T.
```

## Sources Directory

The sources directory contains *all* of the C source files required to build a FunnelWeb binary executable. In the following list, files given without an extension represent both .c and .h files.

```
analyse       - The analyser.
as            - Assertions.
```

```
clock          - A clock abstraction.
command        - The shell command interpreter.
data           - Shared data structures and global vars.
dump           - Dump internal data structures.
environ.h      - Machine-dependent, program-ind header.
help           - Module to write out help messages.
help_gnu       - Function to write out the GNU license.
help_gnu.txt   - The GNU license in text form.
help_gnu.ctx   - The GNU license in C code form.
list           - A list abstraction.
lister         - Module to manage the listing file.
machin         - Module to hold machine-dependent,
                 program-dependent stuff.
main.c         - The main() program.
mapper         - Module to read files into memory.
memory         - Memory management.
misc           - Miscellaneous functions.
option         - Command line option processing.
parser         - The parser.
scanner        - The scanner.
section        - A section number abstraction.
style.h        - A machine-independent,
                 program-independent header file.
table          - A table abstraction.
tangle         - The tangler.
texhead        - Module to write out TeX header
                 in documentation files.
texhead.ctx    - The TeX header in C code form.
texhead.tex    - The TeX header in TeX form.
weave          - The weaver.
writfile       - Output abstraction.
```

The "`.txt`", and "`.tex`" files do not participate in the compilation, but are considered part of the source code as they were used to generate the "`.ctx`" files. The "`.ctx`" files are included by .c files of the same name. They do not need to be compiled themselves.

## Tests Directory

The tests directory stores all the input files of the regression test suite. These come in two kinds: FunnelWeb input files with extensions of "`.fw`", and FunnelWeb include files with extensions of "`.fwi`".

```
FunnelWeb Input Files:
   an01.fw ... an04.fw  - Analyser tests.
   ex01.fw ... ex16.fw  - Examples from the tut man.
   generate.fw          - Generates tricky input files.
   hi01.fw ... hi10.fw  - Examples from hints.
   pr01.fw ... pr10.fw  - Parser tests.
   sc01_note.fw         - Explains absence of sc01.fw
   sc02.fw ... sc29.fw  - Scanner tests.
   tg01.fw ... tg09.fw  - Tangler tests.
   wv01.fw ... wv06.fw  - Weaver tests.

FunnelWeb Include Files:
   ex09a.fwi
   sc13a.fwi ... sc13f.fwi
   sc15a.fwi
   tg08a.fwi
```

## Web Directory

This directory is for the FunnelWeb webs if you want to keep a copy on your local disk so you can surf it quickly.

Reference

Tutorial

Developer
1 Compile
2 Design
3 Implement
4 Modify
5 Misc
6 Licence

SEARCH

**FunnelWeb Developer Manual**

# 1.4 Compiling FunnelWeb

The FunnelWeb source code is entirely contained within the sources directory. However, some simple script files and makefiles can be found in the admin directory.

There should be little difficulty compiling FunnelWeb for any of the currently supported platforms. If the machine on which you are compiling FunnelWeb is not one of the ones listed in the environ.h file, then choose the closest one you can. FunnelWeb contains some machine-dependent components, so if you are compiling for a currently-unsupported platform, you will probably need to go into the modules environ.h, machin.h and machin.c and make some changes.

Compile FunnelWeb by pointing your C compiler at all the "`.c`" files in the sources directory. The "`.txt`", and "`.tex`" files do not participate in the compilation, but appear in the sources directory because they were used to generate the "`.ctx`" files. The "`.ctx`" files are included by .c files of the same name and do not need to be compiled separately. Link the results.

```
   cc -c *.c
   cc -o fw *.o
```

The result of all this should be a binary executable called fw, or fw.exe, or fw.xxx where .xxx is whatever file extension is appropriate on the target machine. Clean up the sources directory by deleting all the listing and object files.

If you encounter any difficulties compiling FunnelWeb, you may wish to refer to the platform-specific notes below.

If you port FunnelWeb to a new platform, please email me any information that I can incorporate into the next release, or add here to help others porting to the same platform.

## A Make File

The FunnelWeb distribution does not provide a makefile as compilers are so fast now that the risk of omitting a dependency far exceeds the benefits provided by a makefile for a program this size.

However, some people have complained about the lack of a makefile, and in fact Dougal Scott (dwagon@nella9.cc.monash.edu.au) actually sent me one which he constructed using the GNU C -MM option. Here it is:

```
------------------------SLICE and DICE here---------------------------------
OBJS= analyse.o as.o clock.o command.o data.o dump.o help.o help_gnu.o list.o \
      lister.o machin.o mapper.o memory.o misc.o option.o parser.o scanner.o \
      section.o table.o tangle.o texhead.o weave.o writfile.o main.o
HDRS= analyse.h as.h clock.h command.h data.h dump.h environ.h help.h \
      help_gnu.h list.h lister.h machin.h mapper.h memory.h misc.h option.h \
      parser.h scanner.h section.h style.h table.h tangle.h texhead.h weave.h \
      writfile.h
CC=gcc
CFLAGS=-O -Wall

all: fw

fw: $(OBJS)
        $(CC) -o fw $(OBJS)

analyse.o : analyse.c style.h environ.h analyse.h as.h data.h clock.h list.h \
  table.h option.h machin.h help.h section.h writfile.h lister.h misc.h
as.o : as.c style.h environ.h as.h machin.h
clock.o : clock.c style.h environ.h as.h clock.h machin.h
command.o : command.c style.h environ.h analyse.h as.h command.h machin.h data.h \
  clock.h list.h table.h option.h help.h section.h writfile.h dump.h lister.h \
  memory.h mapper.h misc.h parser.h scanner.h tangle.h weave.h
data.o : data.c data.h style.h environ.h clock.h list.h table.h option.h machin.h \
  help.h section.h writfile.h
dump.o : dump.c style.h environ.h as.h clock.h data.h list.h table.h option.h \
  machin.h help.h section.h writfile.h dump.h misc.h
help.o : help.c style.h environ.h as.h help.h help_gnu.h misc.h data.h clock.h \
  list.h table.h option.h machin.h section.h writfile.h
help_gnu.o : help_gnu.c style.h environ.h help_gnu.h help_gnu.ctx
list.o : list.c style.h environ.h as.h machin.h memory.h list.h
lister.o : lister.c style.h environ.h as.h data.h clock.h list.h table.h option.h \
  machin.h help.h section.h writfile.h lister.h misc.h
machin.o : machin.c style.h environ.h as.h machin.h
main.o : main.c style.h environ.h as.h command.h machin.h data.h clock.h list.h \
  table.h option.h help.h section.h writfile.h memory.h
mapper.o : mapper.c style.h environ.h as.h machin.h mapper.h memory.h
memory.o : memory.c style.h environ.h as.h machin.h memory.h
misc.o : misc.c style.h environ.h as.h data.h clock.h list.h table.h option.h \
  machin.h help.h section.h writfile.h memory.h misc.h
option.o : option.c style.h environ.h as.h data.h clock.h list.h table.h option.h \
  machin.h help.h section.h writfile.h misc.h
parser.o : parser.c style.h environ.h as.h data.h clock.h list.h table.h option.h \
  machin.h help.h section.h writfile.h lister.h mapper.h memory.h misc.h parser.h
scanner.o : scanner.c style.h environ.h as.h clock.h data.h list.h table.h option.h \
  machin.h help.h section.h writfile.h dump.h lister.h mapper.h memory.h misc.h \
  scanner.h
section.o : section.c style.h environ.h as.h section.h
table.o : table.c style.h environ.h as.h machin.h memory.h table.h
tangle.o : tangle.c style.h environ.h as.h data.h clock.h list.h table.h option.h \
  machin.h help.h section.h writfile.h lister.h memory.h misc.h tangle.h
texhead.o : texhead.c style.h environ.h texhead.h writfile.h texhead.ctx
weave.o : weave.c style.h environ.h as.h data.h clock.h list.h table.h option.h \
  machin.h help.h section.h writfile.h lister.h misc.h texhead.h weave.h
writfile.o : writfile.c style.h environ.h as.h machin.h writfile.h
------------------------SLICE and DICE here---------------------------------
```

## Apollo

16-Oct-1992: David Barton (dlb@hudson.wash.inmet.com) reports that he has managed to compile FunnelWeb on an Apollo running Domain/OS SR10.2.

## HP-UX

15-Oct-1992: On an HP 9000/s300 running HP-UX 8.0, some users got FunnelWeb to compile, but found that they were running up against buffer size limitations. The problem is the definition of FILENAME_MAX in machin.h. When I wrote FunnelWeb, I assumed that FILENAME_MAX was the maximum length of an entire file specification including the path. However, on many machines, it is defined the be the maximum length of a filename without the path. The problem can be fixed by forcing the definition

```
   #define FILENAME_MAX 300
```

in machin.h. This is the only known problem encountered by those who have performed HP ports.

## Macintosh MPW C

09-Aug-1992: Tor Olaussen (olaussen@cc.uib.no) of the Nansen Remote Sensing Center, University of Bergen, Norway reports that he has managed to get FunnelWeb to compile and run under the Macintosh MPW environment. He writes:

```
"If you are interested in my changes, I should be able to generate
diff files, and send them to you together with a makefile. Otherwise
```

you can use me a reference if you receive questions about FunnelWeb and MPW."

## Next

16-Oct-1992: John Dawson (avsoft!john@cs.utexas.edu) has installed FunnelWeb on a NeXT machine and has got it to pass the regression tests. He writes:

> "I had to play with the machin.{h,m} and environ.h files to get FunnelWeb to compile correctly.  The problem area was in machin.h where you put all those prototypes for the Sun.  They didn't jive well with my system.  However, every other default for the Sun was ok.  (A NeXT only needs to be aligned on 2^1-byte boundaries, and I think 2^0-byte boundaries might even work, but aligning things on 2^2-byte boundaries doesn't hurt much.)."

The modified header files are available in the file:
ftp://ftp.ross.net/clients/ross/funnelweb/funnelweb300/funnelweb300_next_patch.c

## MS-DOS

09-Oct-1992: In the lead up to the release of FunnelWeb 3.0, the PC that had been available for me to use to test the port of FW to the PC became unavailable. Because of this, I took the shortcut of not compiling and testing the final released version of the code (V3.0) on a PC!!! Worse still, I did not provide any notes on what compiler flags should be used.

As a result, those who have attempted to compile FunnelWeb on PCs ran into a few difficulties.

If you still want to compile FunnelWeb on the PC, here are some notes that I have collected:

Most people who compile FunnelWeb on a PC seem to be using Borland C++. If you are using this compiler, you need to do the following:

- Insert "#define STDC 1" in environ.h.
- Compile with the HUGE memory model.
- Turn on the ANSI option.
- Turn on the BorlandC++ option.
- Increase the run time stack size from 4K (default) to 32K by inserting the line: extern unsigned _stklen = 32768; just above the minimain() declaration. Refer to page 608 of the borland library reference manual (V3.00).

You may run into troubles with MSDOS with the regression test input file "sc13a" which is a zero length file. It is just a zero length file so if you run into difficulties, create it any way you like.

**FunnelWeb Developer Manual**

# 1.5 Testing FunnelWeb

Once you have obtained a binary executable, you should test FunnelWeb before using it or making it available to users. FunnelWeb comes with a comprehensive regression test suite. Here's how to use it:

1. Set the default directory to be the scripts directory.
2. If you are on a Macintosh, copy the FunnelWeb executable into the scripts directory.
3. Edit the script master.fws. Locate the section called "Define Symbol For the Root Test Directory" and define the R symbol to point to the FunnelWeb root directory fwdir. The examples in the comments in the script should make it clear what is required.
4. Invoke FunnelWeb to execute the master test script with the command line fw +xmaster

The master.fws script should run for a minute or two. If all goes well, you will find a differences report on your screen reporting zero differences. If this happens, then FunnelWeb has been fully tested and is ready to be made available to users. You should delete all the files in the results directory.

If you run into any problems, please write a short report describing the problem and mail it to me (Ross Williams (ross@ross.net)). I may not be able to help you with it immediately, but I certainly want to know that a problem exists, so that it can be corrected in future releases of FunnelWeb.

Reference

Tutorial

Developer
1 Compile
2 Design
3 Implement
4 Modify
5 Misc
6 Licence

SEARCH

FunnelWeb Developer Manual

# 2 FunnelWeb Design Notes

FunnelWeb Developer Manual

# 2.1 Introduction

This chapter contains notes on the design of FunnelWeb. These notes are mostly the result of a complete design review that occurred in late 1991 as part of the process of preparing FunnelWeb V3.0 for public release.

Throughout the FunnelWeb design process I have tried to stick to the principles of simplicity and clarity. As a rule, it was considered more important that a feature be simple, and not allow the user to outsmart himself, than it was for the feature to be elegant. For example, the FunnelWeb macro calling syntax is not as clean as the C preprocessor syntax, but is simpler and more visible, and so is less likely to cause trouble.

**FunnelWeb Developer Manual**

## 2.2 Motivation For FunnelWeb

During 1986, I discovered Donald Knuth's WEB literate programming system in the form of Jon Bentley's *Programming Pearls* column in *Communications of the ACM* **[Bentley86]**. This prompted me to obtain a copy of the report on the Web system**[Knuth83]** and to try out the WEB program.

WEB was the best system that I had seen for producing printed and online, inline documentation. To me the most extraordinary aspect of the system was its success despite the fact that it had been built into the horribly antiquated file/batch processing systems that we all know and love (and use). I had imagined sophisticated documentation systems before this time, but had always assumed that they would be parts of complex programming environments. Knuth showed that, to some extent, it can be done using 1960s software technology (excluding the 1980s typesetting technology).

The WEB system was enticing and promising, but to me suffered from many drawbacks, many of which Knuth had categorized as advantages. The following highly subjective list of disadvantages formed a springboard for the construction of FunnelWeb.

- WEB can only process Pascal programs.
- WEB can produce only one output file. In many instances it is desirable to generate more than one output file.
- WEB enforces Knuth's individual style of indentation. WEB supplies commands to over-ride the automatic indentation, but it is an uphill battle and the code becomes clogged up with format directives.
- WEB does not cater for non-standard Pascal programs. In particular, all identifiers are truncated to about eight characters.
- WEB formats the program output file into a form that is unreadable to humans.
- WEB does not provide an include facility. This was considered a feature essential for supporting macro libraries.
- WEB provides macros with at most one parameter. Knuth describes a hack that can extract a multiple

parameter macro facility from a single parameter one, but it is hardly satisfactory.

- WEB does not provide conditionals.

Most of these objections boiled down to two points: that WEB is far too specialized, and that Knuth's "Occam's Razor" had cut too far. What I wanted was a documentation system that employed all the same principles as WEB but was far more general. The result was FunnelWeb V1 which could process programs in any language or any combination of languages at the cost of setting the text in tt font instead of typesetting it in a language-specific way, as WEB does.

Originally, it was intended that FunnelWeb would be typesetter independent as well as language independent. It was intended that a format file consisting of a set of productions describing how the document file was to be formatted would be handed to FunnelWeb along with the input file. However, time pressures forced me to take the back door and hack up a TeX document file generator. In version 3.2 an HTML generator was added.

www.ross.net

F W

Reference

Tutorial

Developer
1 Compile
2 Design
3 Implement
4 Modify
5 Misc
6 Licence

SEARCH

FunnelWeb Developer Manual

## 2.3 Indentation

How should FunnelWeb insert the text of macros that are not called in column one? A macro call that does not appear at the left margin is called an **indented macro call** and suggests three different alternatives for its expansion: **no indentation**, **blank indentation**, and **text indentation**. Here are examples of each kind of indentation. First the example problem.

```
@$@<Sloth@>==@{@-
Aardvark
Walrus@}

@O@<Output@>==@{@-
Zebra@<Sloth@>
Giraffe
@}
```

There are three ways that the second line of the Sloth macro can be indented in the product file.

**No indentation:**

```
     ZebraAardvark
     Walrus
     Giraffe
```

**Blank indentation:**

```
     ZebraAardvark
          Walrus
     Giraffe
```

**Text indentation:**

```
     ZebraAardvark
     ZebraWalrus
     Giraffe
```

No indentation is useful where the user wishes to view the output as a pure byte stream. Blank indentation is useful when the user wishes to generate indented computer programs. Text indentation is useful where the user wishes to prefix each line of an entire macro invocation with a string. This can be useful for commenting out code (e.g. in Ada using --), and for prepending constructs such as a dollar sign at the start of each line in an OpenVMS DCL script command file.

The design questions are as follows:

1. Which of the three kinds of indentation should FunnelWeb support?
2. What should be the granularity of swapping between indentation modes?
3. Are particular indentation modes dangerous?
4. Is the presence of particular combinations of indentation modes confusing to the user?
5. How and when should the choice of indentation be specified?

After a lot of thought, it was decided that the factor that should determine the design is the *clarity* in the user's mind of the indentation facility, and the *danger* associated with misunderstanding it. Here are two examples that show how easily a misunderstanding about the indentation model can cause a serious semantic error:

```
--Misuse of blank (and no) indentation.
--@<Sloth@>
```

In this first example, the user has assumed that text indentation is in action and has placed an Ada comment symbol "`--`" before the invocation of the macro @<Sloth@> in the expectation that every line in the expansion of the macro will be prefixed by "`--`". Unfortunately, if no-indentation or blank-indentation is active, only the first statement of the expansion of macro @<Sloth@> will be commented out.

```
--Misuse of text indentation:
a++; @<Sloth@>
```

In this second example, the user has assumed no-indentation or blank-indentation and has placed the call to @<Sloth@> after the incrementing of variable a. Under a text-indentation mode, this will result in the statement "`a++;`" appearing at the start of each line in the macro expansion!

These examples show that confusion about the indentation model could cause a serious semantic problem in a program written using FunnelWeb. In particular, the examples above pose a dilemma because they are symmetric. One cannot simply pin the blame on one particular indentation form. A little thought reveals that the greatest danger lies in *confusion* in the user's mind. If the user is confused between text or blank indenting, problems will arise.

There seems to be two ways to solve the problem. The first is to ban all macro calls that are preceded by non-blank text. This is not a good option because there are so many cases where it is desirable to expand more than one single line macro on the same line. A second option is to eliminate one of the two forms so as to reduce the potential for confusion in the user's mind. I choose the latter option. Of the three forms, the clear choice for elimination is text indenting for the following reasons:

1. It actually introduces extra text which gives it an a priori potential for problems.
2. It is harder to implement and would slow down Tangle.

The only other decision is the level of granularity of choice between the remaining options: no indentation and blank indentation. FunnelWeb V1 allowed this choice to be made in the command line, but this was a bad choice because it made the user responsible for a portion of the file's semantics. A better system, used in FunnelWeb V3, is to allow the user to specify the indentation mode in the input file itself.

Again, to avoid confusion, it seems sensible to allow the user only one indentation mode per FunnelWeb input file. In most cases, the user will be happy with blank indentation (the default) and there will be no need for change anyway.

**Decision:** Implement only "no indentation" and "blank indentation". Make the choice of indentation a static attribute of a particular FunnelWeb run that is specified in the input file.

FunnelWeb Developer Manual

# 2.4 A Review Of FunnelWeb Syntax

One of the disadvantages of FunnelWeb is its clumsy macro definition and calling syntax. Compared to (say) the C preprocessor, FunnelWeb's macro call syntax is like a freight train in a china shop. Unfortunately, all attempts to convert this syntax to something more elgant have failed, because the existing syntax is the simplest from a conceptual point of view and is the least likely to cause semantic confusion. This page discusses and justifies the FunnelWeb syntax.

## Macro Definition Syntax

FunnelWeb's macro definition syntax results in definitions that look like this:

```
@$@<Put out the cat@>==@{@-
Open the door
Say out
Close the door@}
```

Here's why this syntax was chosen:

> **Dollars:** The @$ is necessary to cue a definition. Without it, the definition might somehow be mistaken for an invocation.
>
> **Name delimiters:** The @< and @> are required to delimit the name and to resonate syntactically with macro calls.
>
> **Content delimiters:** The @{ and @} delimit the content of the macro.
>
> **EOL suppressor:** The @- is a result of the simplifying rule that the content of a macro is "exactly the text between the @{ and @}."

The only real target for a syntax purge the "==" which is optional anyway.

An alternative minimalist approach to macro definitions is:

```
@<Put out the cat@>
Open the door
```

```
Say out
Close the door@}
```

but this is too dangerous for my tastes as there is no syntax surrounding the name of the macro to indicate that it is a macro definition and not a macro call.

## Parameterized Macro Definition Syntax

The definitions of parameterized macros are even messier than the normal definitions.

```
@$@<Put out the cat@>@(@3@)==@{@-
Open the door
Say out
Close the door@}
```

This is messy, but the "natural" alternative is even worse:

```
@$@<Put out the cat@>@(@1@,@2@,@3@)==@{@-
Open the door
Say out
Close the door@}
```

Given that the parameters don't have names, it seems cleaner just to specify the number of parameters. pecifying the number of parameters seems sensible.

## Macro Call Syntax

Here are some syntaxes that were considered for FunnelWeb macro calls.

```
Open the door
@<Say Out@>         @! Current style.
Close the door

Open the door
@<Say out>@
Close the door

Open the door
@"Say out@"
Close the door
```

```
Open the door
@(Say out@)
Close the door

Open the door
@<Say out>
Close the door
```

Of these, the first was chosen.

## Parameterized Macro Call Syntax

FunnelWeb V1 provided a messy parameterized macro call syntax:

```
@<Say Out@>@(@"firstparam@"  @,
   @"Secondparam@" @,
   @"thirdparam@" @)
```

This syntax was cleaned up considerably in V3.0 by making the @" symbols optional:

```
@<Say Out@>@(firstparam@,@-
Secondparam@,thirdparam@)
```

Reference

Tutorial

Developer
1 Compile
2 Design
3 Implement
4 Modify
5 Misc
6 Licence

SEARCH

FunnelWeb Developer Manual

## 2.5 Document Structuring

Knuth's WEB system provided just two levels of headings, and this scheme was used in FunnelWeb V1. However, experience showed that there was a need for more levels. Here are some syntaxes that were considered for implementing this:

```
@*@<Main Program@>
@**@<Read the Message@>
@***@<Encrypt the Buffer@>

@*@<Main Program@>
@*@*@<Read the Message@>
@*@*@*@<Encrypt the Buffer@>

@s@<Main Program@>
@ss@<Read the Message@>
@sss@<Encrypt the Buffer@>

@s@<Main Program@>
@s@s@<Read the Message@>
@s@s@s@<Encrypt the Buffer@>

@S@<Main Program@>
@SS@<Read the Message@>
@SSS@<Encrypt the Buffer@>

@S@<Main Program@>
@S@S@<Read the Message@>
@S@S@S@<Encrypt the Buffer@>

@1@<Main Program@>
@2@<Read the Message@>
@3@<Encrypt the Buffer@>
(using @A..@I@ as macro
 parameters or overload @1..@9)

@*@1@<Main Program@>
@*@2@<Read the Message@>
@*@3@<Encrypt the Buffer@>
(using @A..@I@ as macro parameters
 or overload @1..@9)
```

```
@A Main Program
@B Read the Message
@C Encrypt the Buffer

@*A Main Program
@*B Read the Message
@*C Encrypt the Buffer

The following is the syntax finally chosen.
@A@<Main Program@>
@B@<Read the Message@>
@C@<Encrypt the Buffer@>
```

Choosing between these alternatives was not easy. The following thoughts contributed to the decision.

- Syntaxes that require visual counting are probably not a good idea.
- Syntaxes that do not delimit the heading name somehow are likely to cause problems where heading names are omitted. Users will be tempted to start paragraphs after the start of heading symbol, and the result is that the first line of the paragraph will be sucked into the heading.
- Overloading the @1, ..., @9 sequences is undesirable.

For these reasons, the following syntax was chosen:

```
@A@<Main Program@>
@B@<Read the Message@>
@C@<Encrypt the Buffer@>
@D@<Encrypt the Buffer@>
@E@<Encrypt the Buffer@>
```

A maximum of five levels was chosen because it's probably sufficient, and @F should be reserved for **F**ile.

**FunnelWeb Developer Manual**

## 2.7 Automated Regression Testing

Automated regression testing is extremely important for several reasons:

1. It allows portability problems to be pinpointed when the program is moved to a new machine.
2. It provides confidence that changes made to the program have not introduced bugs.
3. It provides ongoing confidence in the integrity of FunnelWeb as a programming tool.
4. It provides a practical confirmation of the semantics of FunnelWeb.

The simplest way to set up automated regression testing is to construct a suite of test cases, each of which consists of a test input file and a "correct answer" output file. To test FunnelWeb, the input files are processed and the resulting output files compared with the correct answer output files.

To automate such a list of tests, some kind of scripting language is required. Unfortunately, at the time of writing FunnelWeb (at least), there was no scripting language that was available on all the platforms to which FunnelWeb had to be ported (Macintosh, IBM-PC, Sun, and OpenVMS). So, after some thought, I decided that the best solution to the problem was to create a command language *within FunnelWeb* that could be used as a framework for the automation of the testing of the rest of FunnelWeb.

The resultant scripting language is described in the FunnelWeb Reference Manual.

**FunnelWeb Developer Manual**

# 2.8 File Name Management

File names present a host of problems for a program like FunnelWeb. First, FunnelWeb can generate so many different kinds of files that conventions must be adopted to prevent their names from colliding. Second, the constraints on file names, and even the structure of file names themselves varies considerably from machine to machine. These two problems have combined to result in the sophisticated and rather complicated way in which FunnelWeb handles filenames.

To summarize, the three problems are:

1. What filename extensions should be chosen for various kinds of file?
2. What filename inheritance should take place?
3. How should FunnelWeb cope with the variations in filename structure between platforms?

These problems are addressed in the following sections.

## Filename Extensions

FunnelWeb is capable of reading and writing a variety of different kinds of files. In particular, FunnelWeb must often operate in an environment where the same information is stored in many forms (e.g. prog.fw, prog.c, proc.exe). File extensions are an essential tool in managing this situation. The default filename extensions chosen for FunnelWeb are:

```
FunnelWeb     : .fw
Product       : None
Documentation : .tex or .html
Listing       : .lis
Journal       : .jrn
```

Lowercase will be used in systems that are case sensitive.

## Filename Inheritance

Inheritance in filenames refers to how input and output files inherit parts of their name from other filenames and their environment. For example if the command

```
fw sloth +J +L +T
```

were issued, you would expect to see output files sloth.jrn, sloth.lis, and sloth.tex. The output file names have inherited the "`sloth`" part of the argument. In fact, FunnelWeb implements a comprehensive filename inheritance scheme, which is shown in the following table. Each filename is built up by starting at the top of its column in the table and working down the column, inheriting filename fields (directory, name, extension) that are absent at that point.

| | **Script** | **Input** | **Include** | **Journal** | **List** | **Document** | **Product** |
|---|---|---|---|---|---|---|---|
| 1 | | | @i | | | | @o |
| 2 | +x | +f | +i | +j | +l | +t | +o |
| 3 | "`.fws`" | "`.fw`" | "`.fwi`" | "`.jrn`" | "`.lis`" | "`.tex`" | |
| 4 | | | +f | +f | +f | +f | |
| 5 | DefDir | Defdir | Defdir | Defdir | Defdir | Defdir | Defdir |

The following notes explain the table.

1. Level 1 has the highest priority because it is a direct specification by the user.
2. Level 2 comes next because this is also a direct specification from the user on the command line.
3. Level 3 provides the default file extensions. Product files do not inherit an extension.
4. Level 5 is built into most operating systems' file specification systems. If you specify file "`x.y`", it is taken to mean on the default disk in the default directory.
5. Level 4 looks straightforward, but secretly conceals a difficult design decision. By the time we get down to this level of inheritance, we know for sure that the filename has already picked up a file extension. So all that is left to inherit is the path and the filename proper. Obviously we have to inherit the filename proper (e.g. sloth in sloth.tex), but should we inherit the input file path? If we do inherit the input file path, files will be placed in the same directory as the input file. If we don't inherit the input file path, files will be placed in the current directory. The choice made is to send all the logging type files into the same directory as the input file. This means, for example, that sloth.lis and sloth.tex will generally land in the same directory as sloth.fw. However, product files are sent to the default directory (if not earlier

specified), as this is where the action is. In normal use, the main output of FunnelWeb will be product files, and so the user will expect them to appear in the current directory by default.

## Portable Structure of File Names

Another problem with file names is the variation of their structure between environments. Here are examples of some of the formats that prevail:

```
UNIX   /device/dir1/dir2/name
VMS    node::device:[dir1.dir2]name.ext;vn
MSDOS  device:\dir1\dir2\name.ext
MAC    device:dir1:dir2:name
```

The solution to dealing with these different formats is to classify them as non-portable and hide the functions that manipulate them in the machine-specific module of FunnelWeb. Luckily there are not many such functions.

The main problem is coping with file systems that do not explicitly support file extensions. With so many possible input and output files, FunnelWeb all but needs such extensions. Machines that do not support them pose difficult design decisions. If the user specifies "`sloth`" as an input file on such a non-extension-supporting system, should FunnelWeb look for sloth or sloth.fw? If the user specifies walrus as a listing file, should it generate walrus or walrus.lis?

Some possible solutions are:

1. Regard the filename sloth as having an empty extension. It will then default to sloth.fw.
2. Regard the filename sloth as having a blank but full extension. That is, it cannot be overwritten by inheritance, but it remains blank.
3. Provide an extra syntactic mechanism to allow the user to specify one or other of the two options above.

The solution adopted in FunnelWeb is the first option. Use of FunnelWeb results in lots of files lying around (e.g. sloth.lis) and it is hard to see how the user will cope with them all without some kind of naming discipline. If a naming discipline has to be used, it might as well be the FunnelWeb one.

Thus the names of all files read and written by FunnelWeb will have a file extension of from zero to three letters separated from the rest of the filename by a "`.`".

The only exception is product files, whose extension is left undefined. Product files need not contain a "`.`" and a file extension.

---

**FunnelWeb Developer Manual**

# 2.9 Constraints On The Number of Instantiations

Experience with FunnelWeb V1 demonstrated the need to be able to specify in each macro definitions how many times the macro was expected to be called. FunnelWeb V1 generated an error if a macro is not used, but permited macros to be called more than once. This caused problems for macro libraries, many of whose macros were not called.

By default, FunnelWeb V3.0 requires that each macro (except for the ones attached to output files) be called exactly once. However, it also provides syntax that allows you to specify that a macro be allowed to be called zero times or many times. This allows a macro to be specified with the following permissible ranges of numbers of calls depending on the presence or absence of "`@Z`" and "`@M`":

```
   0..1          @$@<Sloth@>@Z...
      1          @$@<Sloth@>...
      1..n       @$@<Sloth@>@M...
   0.....n       @$@<Sloth@>@Z@M...
```

The initial proposal for this syntax was to allow the user to insert zero, one, or both of @? and @M just after the @$ of a macro definition. Here are some other ideas that were considered for the syntax of the calling number constraint.

```
@?@M@$@<Slothy dogs@>@(@5@)==@{@-
This is a short macro.
With only a line or two@}

@$@<Slothy dogs@>@?@M@(@5@)==@{@-
This is a short macro.
With only a line or two@}

@$@<Slothy dogs@>@(@5@)@?@M==@{@-
This is a short macro.
With only a line or two@
```

The first form was rejected because it is a good visual rule to start all the macros with @$. The second form was rejected because it detaches the macro name from the parameter list, thus making it look less like a call, which is a desirable syntactic resonance. The third form is messy, but was adopted with the change that @? was changed to @M.

After some thought, it was decided that the third syntax was the best, but that the "@?" sequence be reserved for a possible future conditional facility. It was replaced by @Z.

Example of final syntax:

```
@$@<Slothy dogs@>@(@5@)@Z@M+=@{@-
This is a short macro.
With only a line or two@}
```

FunnelWeb Developer Manual

# 2.10 Document Structure Vs Macro Structure

Once the decision was made to adopt five levels of headings as a document structure, the next issue was to decide how this document structure would relate to the macro structure. The issues to be addressed were as follows:

- How should the hierarchical structure connect to the macro structure?
- How can backwards compatibility be achieved? Should it?
- Should the macros be cross referenced by section or by definition?
- Should nameless sections inherit macro names as headings?
- Should TeX macros be used to structure the document?

This page contains some thoughts on these issues:

## TeX File Document Basis

One idea is to make the FunnelWeb input file an actual TeX .tex file which is directly typesettable, and use FunnelWeb to extract code from the TeX file to generate the product files. This approach was rejected because it is too typesetter dependent. Instead, the separate FunnelWeb format was adopted.

## Hierarchical Heading Numbering

WEB and FunnelWeb V1 used two levels of section headings, but numbered them all sequentially. This is unacceptable in a fully hierarchical document which should have hierarchical section numbers such as 3.2.4. So FunnelWeb V3 uses hierarchical numbering for its headings in the documentation file. While hierarchical numbering is good for headings, it is messy and confusing when applied to macro names. In FunnelWeb V1's typeset output, each macro call has appended in square brackets the number of the section in which the macro is defined. However, hierarchical numbering would be somewhat messy. For example, in the typeset documentation, a macro call might look like.

```
Write out the output[6.7.4.3]
```

Similarly, cross reference lists would be messy:

```
This macro is used in 3.4.5, 1.2, 7.8.9, 7.4, 2.2.1.1.
```

These problems were solved by numbering headings hierarchically and numbering the macros sequentially and independently.

## Input Format Matters Most

In terms of legacy commitment, the critical issue is not how FunnelWeb formats programs for printing, but how the FunnelWeb .fw input file should be formatted. The typeset output can always be changed simply by fiddling with FunnelWeb's Weave module. However, as soon as the input syntax is defined, it will be used in dozens or hundreds of documents and it will be very difficult indeed to change it. Therefore, the important thing is to provide as sensible and expressive a .fw format as possible.

Thus, the issue of how to number headings and macros in the typeset documentation is not a serious architectural issue.

## Naming Sections

How should sections be named? In many cases (especially in the case of high-level sections) the writer will provide an explicit name for a section. In other cases, provision of such a name will merely duplicate the name of the macro contained within the section. It therefore makes sense to allow the user to omit the name from a section, allowing Weave to name the section after the first macro definition in the section. If a macro is unnamed and there is no macro in the section, an error can be generated.

## Summary

All these thoughts lead to the following scheme:

- Documents will be hierarchically structured using @A, @B etc.
- Each section can be given a name @<...>.
- Sections that do not have names inherit the name of their first macro.
- If a section does not have a name or a macro, it is erroneous.
- Sections will be numbered hierarchically either by FunnelWeb or by TeX.
- Macro body parts will be numbered sequentially by FunnelWeb and cross referenced by these numbers.

All this results in a system which:

- Provides a hierarchical document structuring capability.
- Is typesetter independent.
- Does not require duplication between heading and macro names.
- Separates the heading and macro systems so that Weave can be configured at a later date to cross reference in different ways without requiring input files to be reworked.

FunnelWeb Developer Manual

## 2.11 Diagnostic Messages

In FunnelWeb, all error messages commence with an indicator indicating the severity of the error message. Here are some of the formats that were considered:

```
W--Error creating sloth.
E--Error opening output file.
S--I'm a teapot.
F--Can't open output file.

W-Error creating sloth.
E-Error opening output file.
S-I'm a teapot.
F-Can't open output file.

W:Error creating sloth.
E:Error opening output file.
S:I'm a teapot.
F:Can't open output file.

W: Error creating sloth.       -- Format chosen.
E: Error opening output file.
S: I'm a teapot.
F: Can't open output file.

War-Error creating sloth.
Err-Error opening output file.
Sev-I'm a teapot.
Fat-Can't open output file.

W-Old fashioned feature.

W-Old fashioned feature.

W--Old fashioned feature.

W: Old fashioned feature.

W:Old fashioned feature.
```

FunnelWeb Developer Manual

# 3 FunnelWeb Implementation Notes

This chapter contains some notes on the FunnelWeb C code.

Reference

Tutorial

Developer
1 Compile
2 Design
3 Implement
4 Modify
5 Misc
6 Licence

SEARCH

FunnelWeb Developer Manual

# 3.1 History of FunnelWeb Implementations

## FunnelWeb V1 (1986)

The first implementation of FunnelWeb was written in Ada**[USDOD83]** in December 1986. I used FunnelWeb as a program to write to help me learn Ada. FunnelWeb V1 took about one month to write and was fairly VMS dependent.

I used FunnelWeb V1 to write all the Ada software for my Ph.D. project and it was very successful in this role, helping me to document some complicated code. However when I finished the Ph.D work in July 1989, I lost access to the VAX and hence to FunnelWeb. During my candidature, at least twenty thousand lines of code were written using FunnelWeb, but hardly anyone else used FunnelWeb. After losing access to Ada and the Vax (and hence to FunnelWeb), I was forced back to programming the non-literate way. I recognised a need to translate FunnelWeb into the more portable C programming language, but didn't have the time or energy to create one.

## FunnelWeb V2 (1991)

About this time (late 1989), David Hulse, at the time a student in Computer Science at the University of Adelaide, volunteered to translate the 4000 line Ada version of FunnelWeb into C. To my knowledge this translation process took about three weeks (in December 1989). The result was called FunnelWeb V2 and was formally signed into the public domain on 5 May 1992. However, it was never publicly released.

I would like to take this opportunity to thank David Hulse for translating FunnelWeb from Ada to C, as this effort was the basis of later work on FunnelWeb.

## FunnelWeb V3.0 (1992)

Lack of portability of the translated C code, combined with the need for a rather solid design review, combined with the need to strengthen the program to bring it up to production standard, resulted in my performing a complete reworking of the code in late 1991 and early 1992. The C code was entirely, but incrementally, replaced or reformatted. The code was also strengthened and new features were added. This process took about two months (November and December 1991). A further two months (approx) were spent writing documentation, constructing a regression test suite, porting the program to different machines, and sorting out the legal issues involved in its release. FunnelWeb V3.0 was publicly released on the Internet under a GNU licence on 6 June 1992.

Version 3.0 proved very solid, but had the following bugs:

- Bombs with an assertion error if the input file is at least 10000 lines long.

- Bombs with an assertion error if it's generating a listing file and an error occurs on a line longer than 200 characters.

```
   lister.dup: count>=DUPMAX
   An assertion has failed! See the line above.
   Press return to abort FunnelWeb>
```

- The FunnelWeb shell diff command bombs with an assertion error if it can't open the first file:

```
FunnelWeb>diff deleteme.list vctest01.ans x.x
E: DIFF: Error mapping "deleteme.list".
         Error fopen()ing input file
         (to determine its length).
do_diff: Image comparison succeeded,
         but text comparison failed.
An assertion has failed! See the line above.
Press return to abort FunnelWeb>
```

- Under VMS FunnelWeb overwrites previous output files with the newly generated output files, rather than generating a new version.

## FunnelWeb V3.1 (1993)

During 1993, the following changes were made, yielding V3.1:

```
13-Oct-1993:
   lister.c: Increased MAXLINES to one million to
             avoid mes_wri case default bomb.
             Modified bomb message in mes_wri.
   memory.*  Modified package to count blocks
             of various sizes.
   command.c Modified it to call memory to
             write out mem report.
13-Oct-1993
   parser.c  Change ty_name to ty_pname
             to save memory.
   data.h    ?
   weave.c   ?
```

These changes essentially fixed the "10,000 lines" bug and performed some memory tuning.

## FunnelWeb 3.05AC

Tony Coates created a FunnelWeb variant called FunnelWeb V3.05AC that fixes some bugs, and provides some extra useful features (including HTML output). This version has been officially classified as an unofficial version.

## FunnelWeb V3.2 (1999)

In April 1999 I decided to perform extensive housecleaning on FunnelWeb so as to make it more presentable and useable. This process involved:

- Eliminating all known bugs.
- Converting all FunnelWeb documentation to webs.
- Porting FunnelWeb to a wider range of platforms.
- Providing FunnelWeb executables online.
- Reworking the main FunnelWeb web.

The result was FunnelWeb V3.2.

FunnelWeb Developer Manual

## 3.2 Why FunnelWeb Wasn't Used to Write Itself

After Knuth created the WEB literate preprocessing system, he re-wrote it using WEB and distributed the source code in WEB source form. To allow the WEB source code to be tangled by users not yet having a copy of WEB, he also included the tangled Pascal code for the Tangler.

While this approach is heroic and serves to convey a commitment and a confidence in literate programming, it seemed to me that writing FunnelWeb in FunnelWeb would simply be asking for trouble. For a start, it would be very hard to modify any feature of FunnelWeb that had been used to write FunnelWeb. Also, the thought of what would happen if the working executable became inoperative for some reason does not bear thinking upon.

One million billion computer programs were written in the non-literate style before FunnelWeb was created. Why not one more?

**FunnelWeb Developer Manual**

## 3.3 Coding Style

Although FunnelWeb wasn't coded under any formal C coding standard, it was coded in accordance with a fairly strict personal style of C which developed during the development of FunnelWeb. This section aims to describe some of the more important elements of the coding style.

**Portability:** This was a major goal of the FunnelWeb implementation. Two excellent books guided this move to portability. They were **[Rabinowitz90]** (which deals with C code itself) and **[Horton90]** (which deals with the portability of various library calls). Other works such as **[Kernighan88]** and **[ANSI]** were also helpful.

**Identifiers: [Rabinowitz90]** specifies that for wide portability, identifiers of block and file scope should be unique to eight characters, and identifiers of program scope should be unique to six characters. I have gone further in FunnelWeb and actually made these restrictions actual limits on identifier length.

Because names must be so short, a system of abbreviations was developed to organize the identifiers used within FunnelWeb. Each abbreviation consists of a letter pair. Here are most of the abbreviations used:

```
bp - Body Part.
cm - Compare. Used to prefix comparison
     routines that return [-1,0,1].
dc - Document component.
dm - Dump package.
el - Element.
eq - Equal. Used to prefix comparison
     routines that return a boolean.
ex - Expression.
f  - Global files.
ll - List of lists.
ln - Line record.
ls - List Package.
lr - Lister package.
ma - Macro.
mc - Macro Call.
mn - Macro Name.
```

```
op - Options package.
pr - Parser.
ps - Position record.
sc - Scrap record.
sn - Section.
tb - Table package.
ty - Typesetter directive.
wf - Write file package.
wl - Write with EOL (misc.c).
wr - Write          (misc.c).
```

**Pointers:** Variables or types denoting pointers start with "`p_`".

**Types:** Names denoting types end in "`_t`". Thus, a type for a pointer to a table would be named p_tb_t.

**File names:** All FunnelWeb source files have file names that are from one to eight characters long and file extensions that are from one to three characters long. This ensures that the FunnelWeb source code can be portably moved to all kinds of machines, including MSDOS!

FunnelWeb Developer Manual

# 3.4 Use of Memory

FunnelWeb is not a memory-stressed program. However, during its development, problems with the management of memory seemed to crop up again and again. This section documents some of these problems and the solutions that were adopted.

There are three places where memory can be obtained: the heap, the stack, and from static variables. The following three sections deal with each of these areas.

## The Heap

One of the great frustrations of being a user is to find that a computer program is complaining about lack of memory when one knows full well that one has allocated at least ten times as much memory to the program as it would ever need to do its job. The reason for such error messages usually has to do with the programmer setting a fixed "reasonable" limit to a particular data structure and then locking it up into an array whose bound is specified by a constant. This malody is particularly common in old Pascal programs. While the use of arrays can increase the speed of a program, it also means that the user cannot increase the capacity of the program without obtaining the source code and recompiling it.

The alternative is to use the heap for all data structures that can grow in proportion to the size of the user's input. This rule has been followed rigorously in FunnelWeb. This means that as memory spaces increase, users will be able to hand their version of FunnelWeb more memory without having to recompile it.

---

Some problems arose early on the Macintosh in the use of the heap. For some obscure reason, many of the heap application malloc calls were failing. Whatever it was, it went away when I replaced direct calls to malloc with calls to a mini package I wrote (called memory) that allocated large chunks of memory and then doled out small pieces as required by the rest of the program.

Having a package to manage all the memory allocation had two other benefits.

First, only one check was required in the entire program to see if memory had run out (in the memory package), and if that failed, the program could be brought to a screaming halt. This organization was far preferable to having each piece of code that needed to allocate memory having to check to see if malloc had failed.

Second, the decision to construct a mini-shell within FunnelWeb to support regression testing meant that FunnelWeb proper (the FunnelWeb fw shell command) could be run many times in any given invocation of FunnelWeb. As a consequence it was necessary to make sure that there was no memory leakage between invocations of FunnelWeb proper. This was accomplished by reworking the memory package to operate a watermark system. The user of the package, when requesting memory, could request "temporary" or "permanent". If permanent, the memory package forgot that it had allocated the memory. If temporary, the memory package places the allocated block on a list. There was then a function in the memory package that could be called to deallocate all the temporary memory. All this meant that so long as all requests for memory within FunnelWeb proper were for temporary memory, and that memory was freed at the end of every run, one could be sure that there was no memory leakage.

## The Stack

For a while during the development of FunnelWeb a particularly nasty bug proved extremely hard to find. The symptom was that FunnelWeb would crash, sometimes at random, but more often upon entering a particular function. In the end, about a day of specific debugging was required before the problem was tracked down to a stack problem. It turned out that somehow (either the fault of the Macintosh or the THINK C language system), the compiler was allocating just 6K for stack space!!!!!!!

This experience led me immediately to go through the entire program and eliminate (or remove to the heap) any automatic variable declarations that used more than one hundred or so bytes.

The lesson is clearly that C programs that use more than a few thousand bytes of stack space are risking their portability. All large data structures should be placed in the heap.

## Static Variables

Static variables also proved a problem on the Macintosh. It turned out that the Macintosh THINK C compiler did not allow more than 32K of statics *in the entire program* . For a while this restriction was a serious threat to the program as it was discovered that constant strings were included in this total! However, some searching revealed a compiler option that removed the strings from the static category.

Nevertheless, the 32K limit is rather severe. Again, it seems that for portability reasons, C programs that use a lot of static variables are risking their portability. As a result, the FunnelWeb code avoids static variables where possible in favour of the heap.

**FunnelWeb Developer Manual**

## 3.5 Implementing Text Indentation

At one point during the development of FunnelWeb, text indentation was fully implemented. However, it was subsequently removed because it was considered a dangerous feature. This section records the way in which text indentation was implemented so that if the feature ever has to be put back, this technique can be used again.

1. Create a new field in the sc_t record call sc_postn.

```
      char *sc_postn; /* Pointer in the range [sc_first,sc_last+1].          */
                      /* It is the minimum possible value of sc_postn for    */
                      /* which EOL does not appear in *sc_postn..*sc_last.   */
                      /* i.e. Points to the byte following the first EOL in  */
                      /* the scrap or sc_first if EOL does not appear.       */
```

2. Modify the scanner so that it generates this field. Sendtext should be modified so that it accepts an argument for the p_postn value.

```
LOCAL void sendtext P_((p_ps_t,char *,char *,char *,bool));
LOCAL void sendtext(p_tkps,p_first,p_last,p_postn,is_white)
/* Appends a text token to the end of the token list.                       */
/* IN: p_ps is a pointer to a position structure giving the position of the   */
/*     first character of the token.                                        */
/* IN: p_first and p_last point to the first and last byte of the text scrap. */
/* IN: p_postn has the same definition as sc_postn (see fwdata.h).           */
/* IN: is_white should be set to TRUE iff scrap is entirely whitespace.      */
p_ps_t p_tkps;
char   *p_first;
char   *p_last;
char   *p_postn;
bool    is_white;
{
 tk_t token;

 /* Empty text scraps should never be generated. */
 assert(p_first<=p_last,"sendtext: Text scrap bounds are bad.");

 /* If ch=EOL then we should be scanning more text, not shipping it! */
 assert(ch!=EOL,"senttext: Shipping text while still more to scan.");

 /* Check that p_postn is in range. See definition in fwdata.h. */
 assert(p_first<=p_postn && p_postn<=p_last+1,
        "sendtext: p_postn is out of range.");

 /* Debug: Check the p_postn field using a brute force check. */
 {
  char *i,*j;
  j=p_first;
  for (i=p_first;i<=p_last;i++)
     if (*i==EOL)
        j=i+1;
  assert(j==p_postn,"sendtext: sc_postn field is incorrect.");
 }

 /* Load the text token. */
 token.tk_kind        = TK_TEXT;
 ASSIGN(token.tk_ps,*p_tkps);
 token.tk_sc.sc_first = p_first;
 token.tk_sc.sc_last  = p_last;
 token.tk_sc.sc_postn = p_postn;
 token.tk_white       = is_white;
 token.tk_parno       = 0;
 ls_add(token_list,PV &token);
}
```

Then all the calls to sendtext have to be changed:

```
/* @ instructs FunnelWeb to replace the special construct with the */
/* special character. Luckily one appears just before the @ !!     */
/* Note: FALSE is OK because space is not a legal specialch.       */
/* Note: Setting parameter p_postn to p_ch-1 is OK as EOL is not a */
/*       legal specialch.                                          */
sendtext(ps_spec,p_ch-1,p_ch-1,p_ch-1,FALSE);
break;

/* + instructs FunnelWeb to insert an EOL. We can't look to the end of */
/* the previous line to find an EOL as this might be the first line.   */
/* Running ahead to the end of the line is expensive, and having the   */
/* liner mini-package maintain a variable for it would be extra        */
/* housekeeping. Instead of all this, we just point to a static.       */
{CONST static char stateol = EOL;
 sendtext(&ps_spec,&stateol,&stateol,(&stateol)+1,TRUE);}
break;

/* If we hit something that ends a text token */
/* then we can transmit a white text token.   */
if (ch==specialch || ch==EOFCH)
   {sendtext(&ps_start,p_first,p_ch-1,MAX(p_sol,p_first),TRUE); return;}

/* Otherwise we have some more (non-white) text to scan. */
/* We can then send a non-white text token.              */
while (ch!=specialch && ch!=EOFCH)
   NEXTCH;
sendtext(&ps_start,p_first,p_ch-1,MAX(p_sol,p_first),FALSE);
```

The dump code needs to be changed too!

```
        wf_str(p_wf,"\"");
assert(token->tk_sc.sc_first !=NULL,"dm_tkls: NULL ptr1.");
assert(token->tk_sc.sc_last  !=NULL,"dm_tkls: NULL ptr2.");
for (i=token->tk_sc.sc_first; i<=token->tk_sc.sc_last; i++)
  {
   if (i==token->tk_sc.sc_postn)
      wf_str(p_wf,"<postn>");
   if (*i=='\n')
      wf_wl(p_wf,"");
   else
      dm_byte(p_wf,*((ubyte_ *) i));
  }
if (i==token->tk_sc.sc_postn)
   wf_str(p_wf,"<postn>");
wf_str(p_wf,"\"");
}
```

3. Over in the Tangle module, create a massive array of pointers to scraps to be used as a stack. Maintain pointers into the stack called current and *base* (similar to the blank indentation variables). Implement the following:

- To write out a scrap, scan it byte by byte. Output each byte. When you hit an EOL, pop the stack back to base. Then write out an EOL followed by the stack contents but writing each scrap only from postn to end end of each scrap. When you have finished the new scrap, push it on the stack.
- When you hit a new macro to expand, save base. Restore it later.

The postn field solves the big problem of how to cope with something like this:

```
The rain in Spain
```

```
falls mainly @<on the plain@>
```

The trouble is that we want to text indent the lines in @<on the plain@> with just "`falls mainly `". However, this string is only part of a scrap. The solution is to get the scanner to record, in the postn field of each scrap, the position of the first byte with a EOL-free run to the end of the scrap.

This scheme is very efficient because all we are doing is pushing and popping pointers to scraps on a stack array. The main disadvantage is that the array must necessarily be finite and would impose a limit on the depth of indentation nesting.

**FunnelWeb Developer Manual**

# 4 Modifying FunnelWeb

This chapter deals with modifications to FunnelWeb. Although the GNU license under which FunnelWeb is distributed allows anyone to modify FunnelWeb and distribute the modified versions, care should be taken before doing this. This chapter discusses some of the issues you should think about if you intend to distribute modified versions of FunnelWeb.

FunnelWeb Developer Manual

# 4.1 The Danger of Modifying Languages

FunnelWeb is a computer program that implements the semantics of the FunnelWeb language. As such, you should take great care when making changes to FunnelWeb that modify the syntax or semantics of its language:

**Removing features:** Removal of features (featurectomy) is extremely difficult once the legacy base is using the features. If a feature is removed, users will have to go through all their files and find a way to simulate the effect of the removed feature with other features. This is usually unthinkable.

**Modifying features:** Modification of features has less direct impact than the removal of features, but can cause more subtle errors. Anyone modifying features should be sure that they are not inadvertently laying semantic traps.

**Adding features:** Although the addition of features is generally the most painless for the user community, as Hoare points out, it is also the most dangerous in the long run.

"When any new language design project is nearing completion, there is always a mad rush to get new features added before standardization. The rush is mad indeed, because it leads into a trap from which there is no escape. A feature which is omitted can always be added later, when its design and its implications are well understood. A feature which is included before it is fully understood can never be removed later."**[Hoare80]**

The benefits of tight control over a language are enormous.

**Universal portability:** Source files can be treated as portable.

**Clear semantics:** Doubt about the semantics of the language are minimized.

For these reasons, I have decided to keep tight control over the FunnelWeb syntax and semantics. If you develop a variant of

FunnelWeb, please take these issues into account.

**FunnelWeb Developer Manual**

# 4.2 Authority vs User Security

There are a number of ways of providing the strong central design authority required to produce the portability and semantic security desired by users,

- Trade mark the name of the language. Publish a specification of the language under the trade name. Warn all users not to trust any implementation that does not guarantee that it implements the language. Then control implementations by only licensing the trade mark to conforming implementations.
- Create a single implementation of the language. Do not release the source code to the implementation. Release only binary executables.
- Release the source code to the implementation, but under a license that prohibits the distribution of modified versions.

Many other variations on these themes are possible, but they are all based on the idea of regulating either the "official" definition of the language or all of its existing implementations.

The solution that I (Ross Williams (ross@ross.net) have adopted (in 1992) is to release the FunnelWeb source code under a GNU license and then to write this chapter in this manual to encourage programmers to think hard if they make any changes to the language.

FunnelWeb Developer Manual

# 4.3 What I Want to Protect

The concerns expressed in the previous section about modifications to the FunnelWeb program do not exclude modifications. They merely suggest that some conditions be observed when modifications are made. In the end there are two things that I want to protect and maintain:

1. Restriction of the name "FunnelWeb" only to computer programs that exactly implement my "official" definition of the language.
2. Restriction of the FunnelWeb source file extensions "`.fw`" (input files) and "`.fwi`" (include files) only to source files that conform exactly to my "official" definition of the language.

So long as these conditions hold, .fw source file will be universally portable. Here are my suggestions for how to obey these rules. These suggestions are in addition to the GNU license rules about documenting any changes that you make in the source files.

**Level 0: Modifications that do not affect functionality:** If you change the FunnelWeb program in a manner that does not affect the functionality of the program in any way (e.g. port it to a new machine, or just speed it up), then you should modify the program to write out a message when it starts up saying that it is a modified version of FunnelWeb. No other actions need be taken.

**Level 1: Modifications that affect surface functionality:** If you make changes to FunnelWeb that affect its functionality without altering or extending the syntax or semantics of the FunnelWeb language, you should augment the name of the program to indicate that it is a FunnelWeb variant. For example, you could call it "FunnelWeb-Dave".

**Level 2: Modifications that affect the language:** If you make changes to FunnelWeb that affect the FunnelWeb language (e.g. by adding some new syntax), you should change the name of the program so that the name no longer contains the word "FunnelWeb", and should choose alternative input and include-file file extensions (the current ones are "`.fw`" and "`.fwi`"). For example, you might call your program "BananaWeb" and use the file extensions "`.bw`" and "`.bwi`".

These rules are not very restrictive. Basically you can do what you like so long as you change the name of the resulting program. I do not wish to restrict anyone who might want to use FunnelWeb as a foundation for a more sophisticated literate programming system. My sole aim here is to protect the integrity of what already exists.

---

**FunnelWeb Developer Manual**

## 4.4 Modifying the Manuals

The FunnelWeb documentation:

FunnelWeb
FunnelWeb Reference Manual
FunnelWeb Tutorial Manual
FunnelWeb Developer Manual

may be modified and distributed so long as you conform to the conditions of the licence notice associated with this documentation:

**FunnelWeb Developer Manual**

# 4.5 Contributions To The Official FunnelWeb

Contributions to the official version of FunnelWeb code are welcome, although it may be some time before I will be able to find the time to incorporate the changes into the official version. Here are some of the issues relating to contributions to the official version.

## Quality Issues

Contributions must be of sufficiently high standard to warrant inclusion in the official version. Whether a modification will be accepted will depend, amongst other things, on the following criteria.

- Does the modification fit in with the design goals of FunnelWeb, or is it just a case of creeping featurism?
- How well coded is the modification? Would it reduce the quality of the code?
- If the modification changes FunnelWeb's functionality, is this a desirable change? How will it impact on existing users? Does it change the semantics of existing FunnelWeb .fw files?
- Would it be quicker for me to make the modification myself than to work out how to incorporate the submitted modification?
- Does the change come with a corresponding addition to the regression test suite?

It is my goal to guard the integrity of the design and code of the official version of FunnelWeb and so I will probably be rather fussy about what I regard as a worthwhile modification.

## Legal Issues

For various reasons, I have decided to maintain complete copyright over the official version of FunnelWeb, but release it under a GNU license each time it is updated. This means that, while you are free to create and distribute different versions of FunnelWeb, if you want your modifications to be incorporated into the official version, you will have to license copyright of the modifications to me (Ross Williams). Here's how it works:

```
+------->------+----------<-------+
|(mods by      |                  |
| me)          V                  |
|     +------------------+        |
+--<--| My Official Copy |        ^
      |   Of FunnelWeb   |        |
      | Copyright (c) Me |        |
      +------------------+        |
 (Periodic GNU |                 / \
  release)     V                /   \  Programmer
      +-------------------+    /Legal\ signs away
      | Official GNU Copy |    \Filte/ copyright on
      +-------------------+     \ r /  modifications
 (Mods made    |                 \ /
  by random    |                  |
  programmer)  |                  ^
               V                  |
             +-----------------------+
             | Modified GNU Version  |
             +-----------------------+
```

This organization allows me to retain full use of the source code for any private, public or commercial purpose, and for the purpose of maintaining the integrity of the FunnelWeb language, while still assuring contributors that their contributions will become part of the public GNU version.

This policy is my current policy as at December 1999, but might change, depending on circumstances.

FunnelWeb Developer Manual

# 5 Miscellaneous

This chapter contains miscellenous development information.

Reference

Tutorial

Developer
1 Compile
2 Design
3 Implement
4 Modify
5 Misc
6 Licence

SEARCH

**FunnelWeb Developer Manual**

# 5.1 FunnelWeb Versions

## FunnelWeb V1 (1986)

FunnelWeb V1 was written in Ada on a VAX in 1986 by Ross Williams. FunnelWeb V1 was released as a VAX/VMS executable for use within the University of Adelaide, but was never publicly released.

## FunnelWeb V2 (1991)

In late 1991, David Hulse translated FunnelWeb V1 from Ada to C. This C version was designated V2. It was never publicly released.

## FunnelWeb V3.0 (June 1992)

In early 1992, Ross Williams rewrote and expanded V2, creating FunnelWeb V3.0. This version was publicly released on the Internet on 6 June 1992.

Version 3.0 proved very solid. However, the following bugs were eventually encountered:

- Bombs with an assertion error if the input file is 10000 or more lines long.
- Bombs with an assertion error (lister.dup: count>=DUPMAX) if a listing file is being generated, and an error occurs on a line longer than 200 characters.
- The FunnelWeb shell diff command bombs with an assertion error if it can't open the first file:
- VMS FunnelWeb overwrites previous output files with the newly generated output files, rather than generating a new version.

## FunnelWeb 3.05AC

Tony Coates created a FunnelWeb variant called FunnelWeb V3.05AC (a variant of FunnelWeb V3.0) that provides some additional useful features. As this is an unofficial version, it's features are not guaranteed to appear in future official versions of FunnelWeb.

## FunnelWeb V3.1 (October 1993)

In October 1993, the following changes were made, yielding V3.1:

- Lister: Increased MAXLINES to one million to avoid 10000 bug.
- Memory: Modified package to count blocks of various sizes.
- Command: Modified it to call memory to write out mem report.
- Parser: Change ty_name to ty_pname to save memory.
- Data: Some unrecorded changes.
- Weave: Some unrecorded changes.

These changes essentially just fixed the "10,000 lines" bug and performed some memory tuning. The other bugs in V3.0 remained. The source code for V3.1 was never released. However, some executables were.

## FunnelWeb 3.2 (May 1999)

In May 1999, Ross Williams revised and enhanced FunnelWeb to produce a new official version: FunnelWeb Version 3.2. The changes made were as follows:

- Fixed the >=10,000 lines bug.
- Fixed the DUPMAX bug.
- Fixed the bug in the diff command.
- Added HTML output using new +u option.
- Added the @L library macro feature.
- Now executes fwinit.fws if no command line arguments.
- Now gives a line number in "Expecting @)" error.
- Eliminated the 64K input file restriction in the MS-DOS

version.

- Macintosh version now sets creator of output files to BBEdit.
- Ported to more platforms.
- Redesigned the main FunnelWeb web.
- Converted the user manuals to webs and placed online.

---

FunnelWeb Developer Manual

# 5.2 Wish List

This page contains the wish list that was included in the original 1992 FunnelWeb Hacker's Manual. As hardly any changes have been made since then, I thought I'd just include it here verbatim.

## Documentation

**An official example:** A official example program written using FunnelWeb should be constructed and made available.

**Index program:** In order to typeset the documentation, a portable index sorting program is required. One should be written and added to the distribution kit. Perhaps this could be the official example!

## Command Line Interface

**Buffer length:** Currently the FunnelWeb shell uses the COMLINE_MAX constant to size its internal command line buffers. This is untenable. The maximum length of a shell command line should not be machine dependent.

**Antiquated Features:** As the FunnelWeb language develops, it is likely that some changes will have to be made that will render one or more language constructs obsolete. When this happens, it may be necessary to add a command line option that has the power to turn on and off warnings or errors flagging antiquated features.

## Shell Interpreter

**SETALL command:** When writing FunnelWeb scripts, it is sometimes desirable to set *all* of FunnelWeb's options to some particular value so that the script is not vulnerable to changes in FunnelWeb's default values which might occur from time to time. To this end, it may be worth creating a "`SETALL`" command that is identical to the "`SET`" command except that it will generate an error if the value of an option is not specified explicitly.

**Recursion test:** A test should be added to test for recursive invocation in a shellscript.

**Diagnostic counting:** The code for counting diagnostics in the script interpreter is rather messy and perhaps even buggy. It should be cleaned up and commented.

**Argument counting:** In the command.c module, there is a variable that counts the arguments to a command. Currently it takes the value of the number of parameters including the command verb. This has turned out to make the code less readable. It should be changed to be the number of arguments to the command verb.

**Make facility:** It may be worth building some sort of make facility into the script language so as to support machines such as the Macintosh that do not already have this facility.

**Signature file:** One problem with using FunnelWeb in conjunction with an external Make facility is that a user might change a FunnelWeb source file without making changes that will affect the files that it generates. If FunnelWeb is then run and the "`+D`" option is on, then the output files will be deleted (to avoid further Make propagations). If Make then has a production linking the .fw file to its output files, then it may conclude that the output files are still out of date. To solve the problem, FunnelWeb could be changed to write a .sig file whenever it processes a .fw file. The Make production could then be wired up to link the .fw file to the .sig file instead of to the output files.

## Language Design

Some proposed changes to FunnelWeb do not correspond to any particular component of FunnelWeb and are really to do with the design of the input language.

**Output or file?:** The "`@O`" special sequence for defining an output file is somewhat non-mnemonic and can be confused with zero ("`0`"). Perhaps it should be replaced by the "`@F`" sequence.

**Syntax of section names:** Currently section names use the same syntax as macro names. For example "`@`". It can be argued that angle brackets should be reserved only for macro names and that some other syntax should be found for delimiting section names. This is not a clear issue. It could also be argued that they are both names, and that because sections can inherit their names from the macros they contain, that the names are of the same "type".

**One macro per section:** One particular style of using FunnelWeb is to have at most one macro definition per section. It may be worth adding a pragma that instructs FunnelWeb to enforce this.

**Should @\{ suppress EOL?:** When defining a macro in FunnelWeb, it seems to be rule rather than the exception that the "`@\{`" be followed by "`@-`". This suppresses the EOL on the definition line, allowing the first line of the macro to be placed immediately above and in line with the other lines without introducing an EOL at the start of the macro text. One option is to introduce a pragma to determine whether to suppress EOLs following "`@\{`".

**Pragma syntax:** It is not clear how "loose" the syntax of pragmas should be. Perhaps they should be case insensitive.

**Conditionals:** Depending on demand, it may be worth reintroducing some sort of conditional feature into FunnelWeb. However, it would have to be very simple to compete with the various ways in which conditionals can already be fudged within FunnelWeb as it stands.

**File markers:** It might be worth modifying the language so that a special syntactical marker is required at the start and end of files. This will assist in detecting truncations and other corruptions.

**Formal parameter lists:** It might be worth changing over to a syntax for formal parameter lists that does not require the @( and @). However, they could be retained as optional for backward compatibility.

## Scanner/Mapper

**All non-contiguous mappings:** Currently FunnelWeb requires that all input files be mapped into a contiguous lump of memory. This caused problems for two reasons. First, to do this, one has to allocate the memory first, and to do that, you have to know how long the file is, and it turns out that finding out the length of a file in a portable manner is very inefficient. Second, although IBM PC compatibles may have megabytes of memory, it is segmented into blocks of at most 64K. This means that FunnelWeb currently cannot read a file longer than 64K on a PC. These problems could be avoided if the mapper and scanner were reorganized to allow input files to be read in and stored as a linked list of chunks of text rather than a contiguous block.

**EOL is unspecifiable:** FunnelWeb uses ASCII character decimal ten (10) internally to represent logical end-of-line and is currently organized so that if one of these is inserted into the text by the user using a "`@^D(10)`", it will be written out as a logical end of line, rather than as a single ASCII character 10. This should be fixed.

**Allow mnemonics for unprintables:** FunnelWeb allows users to insert unprintable characters into the output using the "`@^D(ddd)`" special sequence. Perhaps it would be changed so that it understands ASCII standard mnemonics such as "`LF`" as well as ASCII numbers.

**Version pragma:** A "`version`" pragma should be added that allows the user to specify in the input file the version of FunnelWeb that was around when the input file was created. At a later date, such a construct would be very useful for determining how an input file should be updated if the FunnelWeb language has changed between versions.

## Parser

There are no proposals to change the parser except as a consequence of other proposals.

## Analyser

**Recursion detection:** Currently the FunnelWeb analyser flags, with an error, all macros with an infinite expansion. This would be best changed to flagging all macros that are directly or indirectly recursive. To do this, Tarjan's algorithm**[Tarjan72]** for the detection of strong components should be installed.

**Once only macros:** By default FunnelWeb prevents a macro from being called more than once unless it has a "`@M`" associated with it. However, FunnelWeb does allow a macro that calls such a macro to be called more than once. Perhaps this "loophole" should be plugged somehow.

## Tangle

The Tangler is one of the cleanest components of FunnelWeb, as basically all it has to do is expand some very well-defined macros.

**Text indentation:** Currently FunnelWeb supports *no indentation* and *blank indentation* . A third form could be added if it was considered necessary. *Text indentation* is the same as *blank indentation* except that instead of indenting with blanks, FunnelWeb would indent with the text to the left of the called macro. This facility could be useful for commenting out large bodies of text in languages that do not have multi-line comments (e.g. Ada). A discussion of the pros and cons of this form of indentation appears earlier.

## Weave

Perhaps FunnelWeb's weakest aspect is its typesetting facility.

**Align table of contents:** When FunnelWeb generates a table of contents, the section numbers are not quite aligned with the start of the controlling heading above them.

**Typesetting strength:** It should be possible to specify the level of typesetting strength for headings so that short documents do not look overdone when typeset. A new pragma would be good for this.

**Typeset a portion:** Sometimes it is desirable to typeset just a portion of a program. A command line option could be added to do this. The option could accept as its argument, a string containing a list of section numbers or heading names.

**Generic typesetter option:** In addition to building in a number of different versions of Weave, one for each popular typesetter, it would be possible to add a special generic format where the typeset output is expressed in terms of *FunnelWeb macros* . The user could then specify macro definitions for a non-supported typesetter and run the output through FunnelWeb Tangle to get a typeset file in a format suitable for the unsupported typesetter.

**Suppression of include files:** It should be possible to specify in the input file that particular included files not appear in the typeset output. Currently, the fact that an inclusion has occurred is not even represented in the typeset output. Suppression of inclusions is particularly necessary where a library of macros has been included at the top of each of a group of source files.

**Cross reference of identifiers:** WEB provides a list of identifiers and a list of all the definitions in which they are used. A similar feature could be added to FunnelWeb.

**Support for non-printables:** Currently FunnelWeb does not provide support for typesetting the special "`@\circumflex(num)`" sequences. This should be added.

**Support for @+ sequences:** Currently Weave does not see "`@+`" sequences as such. Instead it perceives them as ordinary EOLs. However, there are arguments for typesetting them specially.

**Typeset text in macro bodies:** One of the much-loved features of WEB is the way that it allows the user to switch recursively between document and program formats. FunnelWeb does not allow this, but should. In FunnelWeb, the delimiters "`@{`" and "`@}`" are already used consistently to delimit macro text. The "`@[`" and "`@]`" sequences have been reserved for the delimitation of documentation text.

**Non-tt typesetting:** The current version of FunnelWeb sets all its macro text in tt font. This is both a blessing and a curse. It is a blessing because it connects the reader directly to the code, with no complicated intermediary. It is a curse because it looks ugly compared to the beautifully typeset programs produced by other literate programming tools.

The difficulty with adding such beautiful typesetting is that it is necessarily language-specific. Keywords and syntax differ from language to language and it would not be easy to come up with a set of language independent rules.

One approach is to write a set of Weave back-ends, one for each language. Another approach is to *generate* back ends. This is the approach taken in the *Spider* system**[Ramsey89]**. In the *Spider* system, the programmer writes production rules for converting lexical components of the program text into typesetter instructions. The *Spider* program reads these rules and generates a new version of WEB suited for the target language.

For FunnelWeb a slightly different system is proposed in which Spider-like rules appear in the input file and are used directly by Weave to perform the typesetting. An intermediate abstract typesetting language could be used so that the productions can be made language specific, but not typesetter specific.

## Lister

**Glue factor:** A glue factor could be added that determines how many lines can be in between two diagnostics in the listing before the two groups of lines are joined together in the listing with no intervening ellipsis.

## Diagnostics

**Advisory information:** Some of FunnelWeb's diagnostics provide a detailed explanatory paragraph. While this information might be useful the first time, it has the capacity to clog up a listing file if the user has made the same error many times. To solve this problem, FunnelWeb could be modified so that such explanations are only displayed the first time the error occurs.

**Abort after n errors:** A facility could be added to prevent FunnelWeb's scanning, parsing, and analysing phases from continuing if a certain number of errors have already been issued.

## Speed

**Measurement of speed:** Although FunnelWeb can generate a breakdown of where it is spending its time, it does not give a final rating in lines per minute. This should be added.

**Find the hot spots:** Although FunnelWeb has been designed to allow high speed, not much effort has so far been made to make it fast. This should be done.

**Change some declarations:** FunnelWeb is full of variable declarations where the variables are wider than they need be. Replacing these might speed it up.

## Correctness

**Bounds analysis:** Not much effort has gone into the design of FunnelWeb's input boundaries. An analysis should be made of FunnelWeb's behaviour when the following quantities are stretched:

- Input line length.
- Input file size.
- Number of macros.
- Length of macro.

In particular, FunnelWeb's behaviour at 32K and 64K boundaries should be observed.

**Stack detection:** Macintosh THINK-C provides just 6K of memory for the stack. It might be worth adding checks to make sure that the stack is not being blown.

## Test Suite

The following tests should be added to the test suite:

```
Lister
------
   LR01: Test with a full listing with no diagnostics.
   LR02: Test with a full listing with diagnostics.
   LR03: Test abbreviated listing with no diagnostics.
   LR04: Test abbreviated listing with diagnostics.
   LR05: Test error context with nearby diagnostics.

Boundary Cases
--------------
Static analysis might preclude the need for
most of these tests.
   BC01: Test what happens when memory runs out.
   BC02: File with a single line of a megabyte.
   BC03: File of a megabyte of EOLs.
   BC04: Output file with an extremely long line.
   BC05: Output file with one million lines.
   BC06: Test on a file with very many macros.

General
-------
   GN01: A large legal input file exercising
         as many features as possible.
         1. Test listing file.
         2. Test output files.
         3. Test typeset file.
   GN... A selection of ten real-life FunnelWeb files.
```

## Platform-Specific Changes

**Icon for the Macintosh:** Currently no icon is supplied for the Macintosh version of FunnelWeb. An icon depicting a spider or a funnelled web of some kind would seem appropriate.

FunnelWeb Developer Manual

## 5.3 Suggestions From Users

This page contains a list of features that FunnelWeb users have suggested. The fact that a suggestion has been made does not automatically mean that I agree with it or that it will be implemented one day. However, most of the suggestions are sound.

The number in square brackets after each suggestion indicates the number of users who have suggested the feature.

A quick look at this list suggests that future work on FunnelWeb should be mainly directed towards improving its typesetting facilities.

```
General
-------
S: Integrate FW into the Borland C++ development environment. [1]
S: Add a feature for updating FW source files after the product
   files have been modified. [1]

Documentation
-------------
S: Improve the typesetting of the index. [1]
S: Add more entries to the index. [1]
S: Add a quick reference guide. [1]

Scanner
-------
S: Fix the bug that makes FunnelWeb bomb on excceptionally long input
   lines. [1]
S: If a closing @} occurs at the start of a line, eliminate the
   preceeding end of line. [1]
S: Allow TABS [1].
S: The syntax is ugly. Redesign it. [2]

Parser
------
S: Add named parameters for macros. [1]
S: Add conditionals. [1]
S: Add a feature that enables the user to specify that all the
   macros called within a particular macros be defined in the order
   in which they are called. [1]

Tangle
------
S: Allow line-oriented substitution and so on rather than stream
   oriented. [1]

Weave
-----
S: Don't typeset non-leaf macros. This will avoid clutter. [1]
S: Teach FW C++ so it can typeset it nicely. [1]
S: Allow both FW text and TeX in the same document (two modes). [1]
S: Include greater support for typesetting a program as a book. [1]
   "What we need is a facility for TANGLING small WEB files
    into small source files, but WEAVING those same small
    WEB files into a single large documentation file, without
    the seams becoming apparent."
S: Add a new format: Microsoft Rich Text Format (RTF). [2]
S: Write a program to convert an RTF file into FunnelWeb source.
   Then people can edit programs in MSWord and run them through
   FunnelWeb. [1]
S: Detailed notes on indexing from one user: [1]
S: Add the ability to include typeset comments within program text. [2]
   R: The @[..@] syntax has been reserved for this.
S: Add a new format: LaTeX. [1]
S: Add a new format: Plain ASCII text. [1]
S: Add a GENERIC language typesetting facility: [1]

   "Have you seen the program 'vgrind' (for TeX) or 'psgrind' (for
    PostScript)?  It contains a termcap-like database of programming
    language features, e.g. format of a label, format of a
    producedure/function, keyword listings, character string quoting
    rules, etc.  Perhaps this would be useful for FW?"

S: Automatically generate an index. [1]
S: Allow @" and @" in documentation to typeset double quotes properly. [1]
S: Detailed comments from one correspondent:
   "The ability to generate a cross reference of identifiers
   (variables and functions) is essential.

   Constants       are generally defined only in 1 place
                   and so a cross reference to the definition
                   macro is useful.

   Variables       are generally declared in 1 place but
                   refered to and defined in multiple places.
                   A cross reference to the declaration site
                   is generally not useful.

                   A cross reference to the site of use is
                   confusing, since there may be many, equally
                   important sites where the variable is assigned
                   a value.

   functions       are generally unique and so a cross reference
                   to the site of definition (or declaration in
                   the absense of a definition) is useful.

   In FWEB (which I have used extensively recently) each instance
   of use of an identifier contains a subscript which points
   to the site of definition of the identifier (I think this
   is what you refer to in the documents as a feature of the
   original WEB.)  The fact that locating identifiers requires
   knowledge of the programming language is true."

Documentation
-------------
   Page 34, section 1.7.2, paragraph 2, last sentence:
           "free text be default" should be "free text by default".

   Page 34, section 1.7.3, paragraph 1, 1st sentence:
           "divide and conquor" should be "divide and conquer"

   Page 35, section 1.73, paragraph 2 on page 35:
           "section heading at level n cannot....at level (n-1) or less."
           I think this should be:
           "section heading at level n cannot.. at less than level (n-1)"
           Otherwise you are saying that level C cannot occur at B or less.

   Page 50, section 2.11, last paragraph, line 10 of para:
```

"excessibly" should be "excessively"

In the example on p. 21, "... the string ``Hello World''" looks like a TeX dependent hangover.

The caption under Fig. 4, p.111 came out badly spaced on our system.

Isn't it `holistic' rather than `wholistic?'

There is often a need to have version dependent documentation as well as version dependent code. It would be nice if there was a special kind of macro that could run the output to a file through the typesetter, i.e., generate TeX files. Meanwhile, you can get some interesting effects by letting FunnelWeb generate .fw files! You might like to mention this exciting possibility in V2 of the user manual.

Reference

Tutorial

Developer
1 Compile
2 Design
3 Implement
4 Modify
5 Misc
6 Licence

SEARCH

FunnelWeb Developer Manual

# 5.4 Known Bugs

## FunnelWeb 3.2 (May 1999)

A bug exists, at least in the Macintosh version, where FunnelWeb V3.2 hangs when fed the following file:

```
@O@<X@>@{@<Y@>@}
@$@<Y@>@M@{@<Z@>@(@<Y@>@)@}
@$@<Z@>@(@1@)@{@1@}
```

FunnelWeb has always (since V1 in 1986) had macro recursion detection, but it appears to fail in this case. You can workaround this bug by eliminating all recursive macro invocations.

---

A bug exists in the output line length monitoring of FunnelWeb V3.2. If you set the pragma indentation = none, FunnelWeb still counts the output line lengths as if they are being indented, even though they aren't. You can workaround this by ignoring the output file line length errors or by setting:

```
@p maximum_output_line_length = infinity
```

The following source code patch has been proposed, but not yet tested:

```
In tangle.c, the code:

     /* Output an EOL with indentation if desired. */
     if (tgindent)
        eolblank(ind_base);
     else
        wf_chr(&f_o,EOL);
     ind_curr=ind_base;

should be changed to:

     /* Output an EOL with indentation if desired. */
     if (tgindent)
       { eolblank(ind_base); ind_curr=ind_base; }
     else
```

```
{ wf_chr(&f_o,EOL); ind_curr=0; }
```

# 6 GNU General Public License Version 2

This page contains a verbatim copy of Version 2 of the GNU General Public License under which the FunnelWeb computer program is released. Note that the FunnelWeb documentation is released and under a much simpler license that does not allow modifications.

---

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this

License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

  4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License.  Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

  5. You are not required to accept this License, since you have not signed it.  However, nothing else grants you permission to modify or distribute the Program or its derivative works.  These actions are prohibited by law if you do not accept this License.  Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

  6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions.  You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

  7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License.  If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all.  For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices.  Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

  8. If the distribution and/or use of the Program is restricted in

convey the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

```
    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) 19yy  <name of author>

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

```
    Gnomovision version 69, Copyright (C) 19yy name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
    type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

```
  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.

**Reference**

**Tutorial**

**Developer**
1 Compile
2 Design
3 Implement
4 Modify
5 Misc
6 Licence

**SEARCH**

**FunnelWeb Developer Manual**

# Search FunnelWeb Documentation

Information about FunnelWeb is divided into the main FunnelWeb web and the Tutorial, Reference, and Developer manual webs. Choose a combination of manuals to search, and enter one or more keywords.

☐ FunnelWeb Main Web (General information)
☐ FunnelWeb Reference Manual (Official definition)
☐ FunnelWeb Tutorial Manual (Tutorial and hints)
☑ FunnelWeb Developer Manual (How to compile)

* Enter one or more words or word prefixes separated by spaces.
* Matching is case insensitive.
* Finds all pages containing at least one word.
* Add the word AND to find pages containing all the words.
* Searching will not work in offline copies of this web.

Search

FunnelWeb Developer Manual

# Copyright and Credits

## Copyright Of The FunnelWeb Program

The FunnelWeb is made available under the GNU General Public Licence Version 2. See the FunnelWeb Developer Manual for a complete copy of this licence.

## Copyright Of The FunnelWeb Webs

## Trademarks

**Macintosh** is a trademark of Apple Computer
**MS-DOS** is a trademark of Microsoft.
**Rocksoft** is a registered trademark of Rocksoft Pty Ltd, Australia.
**Unix** is a registered trademark of AT&T.
**VAX** and **OpenVMS** are trademarks of Digital Equipment Corporation.

## Questions

Please email me if you have any other questions about the intellectual property aspects of FunnelWeb.

## Credits

FunnelWeb was conceived, designed and implemented by Ross Williams in 1986, 1992, and 1999. The FunnelWeb webs were originally written by Ross Williams in 1992 in the form of a printed manual, and converted by him to webs in April 1999.

FunnelWeb was the main tool used to create these FunnelWeb webs. Each web was written as a single FunnelWeb .fw file which, when processed by FunnelWeb, generates all the .shtml files.

I would like to thank a number of people who assisted me (Ross Williams) during the creation of FunnelWeb.

Many thanks to David Hulse for translating the original version of FunnelWeb (FunnelWeb V1) from Ada into C (FunnelWeb V2) and getting it to work on Unix and a PC. The C code written by David (FunnelWeb V2) formed the basis of FunnelWeb V3.

Thanks go to Simon Hackett of Internode Systems for the use of his Sun, Mac, and PC, for assistance in porting FunnelWeb to the Sun and PC, and for helpful discussions.

Thanks go to Jeremy Begg of VSM Software Services for the use of his VAX, and for assistance with the VMS-specific code.

Thanks to Barry Dwyer and Roger Brissenden for trying out FunnelWeb Version 1 in 1987 and providing valuable feedback.

Thanks to Donald Knuth for establishing the idea of literate programming in the first place.

---