The CWEB System of Structured Documentation

(Version 3.0)

Donald E. Knuth and Silvio Levy

Система структурного документирования CWEB

(Версия 3.0)

Дональд Э. Кнут и Сильвио Леви

Система структурного документирования CWEB

Дональд Э. Кнут и Сильвио Леви

Этот документ является описанием версии системы `WEB` Д. Кнута, адаптированной к языку C Сильвио Леви. С момента своего создания в 1987 г., `CWEB` совместными усилиями Кнута и Леви во многом был модифицирован и улучшен. В настоящий момент мы полагаем его эволюцию близкой к завершению, но тем не менее, сообщения об ошибках, предложения и комментарии весьма желательны и могут направляться Леви (`levy@geom.umn.edu`).

Читатели, знакомые с меморандумом "The `WEB` System of Structured Documentation" Кнута, могут лишь бегло ознакомиться с этим документом, поскольку `CWEB` и `WEB` имеют общую философию и (в основном) синтаксис. В некоторых вопросах `CWEB` является упрощением `WEB`: например, `CWEB` не нуждается в механизмах определения макросов и обработки строк `WEB`, поскольку язык C и его препроцессор берут эту работу на себя. Аналогично, соглашения `WEB` относительно формата восьмиричных и шестнадцатиричных чисел `@'77` и `@"3f` заменены принятыми в языке C `077` и `0x3f`. Все прочие возможности `WEB` сохранены и расширены.

Мы выражаем свою благодарность всем, кто своими пожеланиями и замечаниями участвовал в разработке `CWEB`. Особую признательность мы приносим Steve Avery, Nelson Beebe, Hans-Hermann Bode, Klaus Guntermann, Norman Ramsey и Joachim Schnitter, которые предоставили программный код, а также Cameron Smith, за множество предложений по документации. Ramsey сделал literate programming доступным для других языков своей системой `SPIDER` [см. *Communications of the ACM* **32** (1989), 1051–1055]. Bode адаптировал `CWEB` к языку C++, так что далее по тексту все ссылки на C подразумевают и C++.

Введение

Суть философии `CWEB` в том, что программисты, желающие снабдить свою программу наилучшей документацией, должны одновременно пользоваться языком, подобным TeX, для подготовки документации и языком, подобным C, для программирования. Ни тот, ни другой сам по себе не предоставляет всего необходимого, но будучи должным образом объединены, оказываются куда более полезными, чем порознь.

Структура компьютерной программы может быть представлена, как "сеть" ("web" — "паутина"), состоящая из множества взаимно зависимых частей. Чтобы документировать такую программу, необходимо описать каждую такую часть и ее взаимоотношения с другими. Средства обработки текстов, предоставляемые TeX, позволяют нам описать локальную структуру каждой части программы, а средства программирования языка C — формально и однозначно специфицировать алгоритмы. Объединив их, мы получаем стиль программирования, максимально способствующий пониманию структуры сложных программ, и одновременно — хорошо документированные программы, для которых автоматически обеспечивается взаимное соответствие программы и документации.

Система `CWEB` состоит из двух программ, названных `CWEAVE` и `CTANGLE`. При программировании в системе `CWEB` программа на языке C и документация к ней объединены в одном файле, называемом `CWEB`-файл, с именем вроде `something.w`. Команда '`cweave something`' создает файл `something.tex`, который с помощью TeX превращается в красиво оформленную ("pretty printed") версию `something.w`, которая соблюдает все типографские соглашения, такие как дизайн страницы, использование отступов, курсивных и жирных шрифтов, математических символов. Дополнительно автоматически строится система перекрестных ссылок. Аналогично, команда '`ctangle something`' порождает программу на языке C `something.c`, которая может быть откомпилирована в исполнимый код.

Помимо предоставления удобного средства документирования, `CWEB` расширяет язык C возможностью менять местами части текста программы, так что большие программные системы могут быть восприняты целиком в терминах небольших секций и их локальных взаимосвязей. Программа `CTANGLE` названа так потому, что она расставляет секции web-структуры в порядке, требуемом C; преимуществом программирования в `CWEB` является то, что алгоритмы могут быть выражены в "развернутой" форме, в которой каждая секция объясняется отдельно от других. Программа `CWEAVE`, в соответствии своему названию, "сплетает" содержащиеся в каждой секции TeXи C-части в структурированный документ. Возможно, не случайно немецким эквивалентом "weave" является "*webe*", а соответствующий им латинский императив — "*texe*"!

Пользователь CWEB должен быть хорошо знаком с языком C. Минимальное знакомство с TeX также желательно, но фактически может быть получено уже в процессе работы с CWEB, поскольку простой текст подготавливается в TeX практически без какого-либо знания его языка. Тем, кто хорошо знаком с обеими системами потребуются совсем небольшие усилия, чтобы изучить команды CWEB.

Обзор

Содержимым файлов CWEB может быть TeX-текст и C-текст. CWEB-программист должен четко представлять все происходящее с ними как в процессе генерации документации, так и программы, т.е. знать суть всех тех действий, которые будут производиться над CWEB-файлом программами CWEAVE и CTANGLE. Текст в формате TeX полностью копируется программой CWEAVE и игнорируется CTANGLE, это "чистая документация". С другой стороны, C-программа форматируется CWEAVE и преобразуется CTANGLE по правилам, которые будут строго определены позднее. В настоящий момент достаточно понять, что существует две разновидности текста. Создание CWEB-программы в чем-то схоже с подготовкой документа TeX с использованием дополнительной "C-моды", подобной имеющимся в TeX горизонтальной, вертикальной и математической модам.

CWEB-файл содержит *секции*, которые более или менее самодостаточны. Каждая секция состоит из трех частей:

- TeX-часть, содержащая материал, разъясняющий суть содержимого секции.
- Промежуточная (middle) часть содержит макроопределения, служащие сокращениями для C-конструкций, которые неудобно выписывать каждый раз заново. Они преобразуются CTANGLE в макросы препроцессора.
- C-часть, содержащая часть программы, генерируемой CTANGLE. Длина C-части в идеале не должна превышать дюжины строчек, чтобы она составляла законченный фрагмент программы и была легко понимаемой.

Три части каждой секции должны располагаться в этом порядке, т.е. TeX-комментарий должен следовать первым, затем промежуточная часть и, наконец, C-код. Любая из этих частей может быть пустой.

Секция начинается либо '@␣', либо '@*', где '␣' означает символ "пробел". Секция заканчивается началом следующей секции, т.е. следующим '@␣' или '@*', либо концом файла. CWEB-файл может содержать произвольный текст, располагающийся перед началом первой секции и не входящий ни в одну из секций. Подобный текст далее будем именовать "преамбулой" ("in limbo"), он игнорируется CTANGLE и полностью копируется CWEAVE, так что его назначение состоит в определении произвольных инструкций TeX'а. Как правило, эти инструкции загружают требуемые шрифты, определяют макросы, изменяют размеры страницы и/или генерируют титульный лист.

Секции нумеруются последовательно, начиная с 1. Эти номера появляются в начале каждой секции в TeX-документе, полученном CWEAVE, а также в комментариях, отмечающих начало и конец каждой секции в C-программе, сгенерированной CTANGLE.

Имена секций

К счастью, нет никакой надобности нумеровать секции вручную. Просто поместите '@␣' либо '@*' в начале каждой новой секции, и их номера будут сгенерированы CWEAVE и CTANGLE автоматически. При необходимости секции может быть присвоено *имя* вместо номера. Имя начинается '@<', далее следует произвольный текст TeX, а затем — '@>'. Когда CWEAVE выводит имя секции, '@<' и '@>' заменяются угловыми скобками, и добавляется номер секции более мелким шрифтом. Поэтому, когда вы читаете вывод CWEAVE, легко проследить все взаимосвязи между секциями.

Для наглядности имя секции должно быть хорошим описанием содержимого секции, т.е. пояснять абстракцию, представленную секцией. Впоследствие секция может быть "встроена" в другие секции так, что малозначимые детали ее реализации окажутся скрытыми. Имя секции, следовательно, должно быть достаточно длинным, чтобы выразить при этом ее суть.

К сожалению, утомительно и чревато ошибками писать длинное имя секции снова и снова. Поэтому CTANGLE и CWEAVE позволяют использовать сокращения. Всякий раз, когда в CWEB-файле должно появиться имя секции, вы можете использовать '@<α...@>', где α — любая строка, являющаяся префиксом полного имени. Например, '@<Clear the arrays@>' может быть сокращено до '@<Clear...@>', если никакое другое имя секции не начинается с 'Clear'. Аналогично можно использовать сокращение '@<Clear t...@>' и т.д.

Имена секций в прочих случаях должны совпадать посимвольно, за исключением последовательностей пробельных символов (собственно пробела, символов табуляции, конца строки и/или страницы), которые трактуются как один пробел. В начале и конце имени все пробельные символы будут отброшены. Так, '@< Clear the arrays @>' будет соответствовать имени в предыдущем примере. Пробельные символы после многоточия также будут игнорироваться, но не перед ним, так что '@<Clear t ...@>' не соответствует '@<Clear the arrays@>'.

Что делает CTANGLE

Как было сказано выше, секция начинается '@␣' либо '@*', но ничего — о том, как она делится на TEX-часть, промежуточную и C-части. Промежуточная часть начинается с первым появлением в теле секции '@d' либо '@f', а C-часть начинается первым '@c' или '@<section name@>='. В последнем случае имя присваивается данной C-секции, если же секция начата '@c', то она будет *неименованной*.

Конструкция '@<section name@>' может появиться в теле C-секции произвольное количество раз: последующие ссылки на имя означают его "использование", а не "объявление", т.е. что именованная секция, определенная где-либо еще, должна быть вставлена в это место C-программы. Действительно, основная идея CTANGLE — создать программу, объединив некоторое количество именованных и неименованных секций. Точные правила такого объединения таковы: вначале все макроопределения, отмеченные '@d' преобразуются в инструкции препроцессора и собираются в начале файла. Далее туда копируются неименованные C-секции. Так образуется первое приближение текста программы (должна существовать хотя бы одна неименованная секция). Затем все имена секций, найденные в полученном тексте, заменяются C-частями соответствующих секций и этот процесс подстановки продолжается до разрешения всех ссылок. Все комментарии удаляются, поскольку полученная программа на C не предназначается для чтения человеком.

Если одно и то же имя было дано более чем одной секции, C-фрагмент получается объединением C-частей всех таких секций. Это оказывается полезным, например, если имеется секция с именем 'Global variables', в которой будут собраны все глобальные переменные, где бы они ни были объявлены. Когда несколько секций имеют одинаковое имя, CWEAVE полагает номер первой из них номером, соответствующим этому имени, и она вставляет в конец секции ссылку: 'See also sections ...' ("См. также секции ..."), в которой перечислены номера всех остальных секций, имеющих это же имя. Соответствующий C-текст форматируется CWEAVE так, что вместо знака равенства в выходном документе присутствует знак эквивалентности, т.е. '$\langle$ section name $\rangle \equiv$ C text'. При этом в последующих секциях с этим именем знак '$\equiv$' заменяется на '$+\equiv$', указывающим на то, что C-фрагмент этой секции будет добавлен к фрагменту другой секции.

Когда CTANGLE начинает и оканчивает обработку секции, она вставляет в выходной файл директивы препроцессора #line, так что номера строк в сообщениях об ошибках и при отладке программы будут ссылаться на исходный CWEB-файл. Благодаря этому, в большинстве случаев о получаемом C-файле можно просто забыть.

Что делает CWEAVE

Общая идея CWEAVE — создать из файла CWEB файл .tex таким образом: первая строка получаемого файла .tex указывает TEX'у загрузить файл макроопределений, описывающих формат CWEB-документов. Далее в tex-файл копируется текст преамбулы, расположенный перед первой секцией. Затем следует по очереди вывод каждой секции, возможно вперемешку с командами окончания страницы. Наконец, CWEAVE генерирует систему перекрестных ссылок: для каждого C-идентификатора указываются номера всех секций, в которых он используется, а также строится алфавитный список всех имен секций, а также оглавление, содержащее номера страниц и секций для всех "заглавных" секций.

Вы спросите, что такое "заглавная" секция ("starred-section")? Секции, которые начинаются '@*' вместо '@␣' обозначают начало новой группы секций. После '@*' указывается имя группы, оканчивающееся точкой. Такие группы в документе TEX всегда начинают новую страницу, и их имя используется в колонтитуле всех последующих страниц до появления очередной заглавной секции. Имя группы также заносится в оглавление и печатается в начале секции жирным шрифтом. Предостережение: не используйте в таких именах управляющих предложений TEX'а, если вы не уверены, что макроопределения из `cwebmac` не конфликтуют с ними. Причиной этому служит то, что имя группы будет преобразовано в верхний регистр, когда оно появляется в колонтитуле, а также будет выведено жирным шрифтом в начале секции и записано во временный файл-оглавление, так что используемые команды должны работать правильно во всех этих режимах.

TEX-файл, полученный с помощью CWEAVE, для каждой секции содержит следующее: во-первых, номер секции (т.е. '`\M123.`' в начале секции номер 123, за исключением заглавных секций, в начале которых вместо '`\M`' следует '`\N`'). Далее следует TEX-часть секции, скопированная почти дословно, как описано ниже. Вслед за этим идут промежуточная и C-части секции, разделенные небольшим промежутком, если они обе непустые. Эти две части получены вставкой множества затейливых TEX-макросов в C-программу. Эти макросы управляют типографскими подробностями, такими, как шрифты, правильная расстановка пробелов в математических формулах, разбиение на строки и отступы.

C-код в тексте TEX'а и наоборот

Когда вы набираете TEX-текст, возможно, что вам понадобится сослаться на переменные и другие объекты C-программы, причем желательно, чтобы их оформление соответствовало тому, что используется в программе. Для этого CWEB предусматривает возможность разместить в теле TEX-текста конструкции языка C, ограничив их символами '`|`'. Например, пусть мы хотим написать что-то вроде

If *pa* is declared as '**int** $*pa$', the assignment $pa = \&a[0]$ makes *pa* point to the zeroth element of *a*.

Текст на языке TEX в CWEB-файле будет таким:

```
If |pa| is declared as '|int *pa|', the
assignment |pa=&a[0]| makes |pa| point to the zeroth element of |a|.
```

И наконец, CWEAVE преобразует его в нечто такое, что никому не захотелось бы вводить вручную:

```
If \\{pa} is declared as '\&{int} ${}{*}\\{pa}$',
the assignment $\\{pa}\K{\AND}\|a[\T{0}]$
makes \\{pa} point to the zeroth element of \|a.
```

Отметим, что таблица перекрестных ссылок, созданная CWEAVE, для подобного комментария будет включать номер текущей секции в списке ссылок на *pa* несмотря на то, что *pa* может и не появиться в C-части этой секции. Действительно, индекс включает ссылки на идентификаторы в поясняющих комментариях так же, как и в самой программе; чтобы оценить эту возможность не потребуется много времени. В то же время, идентификаторы **int** и *a* в индекс не попадут, потому что CWEAVE не помещает в индекс зарезервированные слова и однобуквенные идентификаторы. Такие идентификаторы полагаются столь вездесущими, что нет смысла перечислять все места, где они встречаются.

Несмотря на то, что секция начинается TEX-частью и заканчивается C-частью, граница между ними точно не определена, поскольку C-код может быть включен в текст TEX, если он ограничен '`|...|`'. Аналогично, текст TEX зачастую появляется внутри C-кода, поскольку все, что считается комментарием (т.е. все внутри `/*` и `*/`, либо после `//`) интерпретируется как текст TEX. Точно так же имя секции содержит текст TEX, но конструкция `@< @>` целиком может использоваться в C-коде; следовательно, переход между режимами C и TEX осуществляется естественно, как в этих примерах:

```
if (x==0) @<Empty the |buffer| array@>
... using the algorithm in |@<Empty the |buffer| array@>|.
```

Первый из них может быть найден в C-части секции, в которую будет вставлен код из секции, названной "Empty the *buffer* array". Второй пример из TEX-части секции, которая лишь ссылается на другую секцию, но не определяет и не использует ее. (Обратите внимание на '`|...|`' вокруг имени секции в этом случае.)

Макроопределения

Управляющий код `@d`, за которым следует

identifier C-код либо *identifier*($par_1, \ldots, par_n$) C-код (*identifier* и скобка во втором случае не должны разделяться пробелами) преобразуется CTANGLE в директиву препроцессора `#define`, которые располагаются в начале генерируемой C-программы, как описано ранее.

Макроопределение '`@d`' может быть разбито на несколько строк, причем не нужно ставить в конце строки '\' — их автоматически вставит CTANGLE. Если по какой-то причине потребуется директива `#define` в конкретной точке C-файла, можно задать ее в C-коде, а не с помощью CWEB, но в этом случае придется маскировать символы перевода строки самостоятельно.

Строки и константы

Если вам нужна в C-коде строковая константа, ограниченная ' или ", ее можно задать как обычно, за исключением того, что символ '`@`' должен быть представлен '`@@`' (это единственный управляющий символ, который может появляться внутри строки, см. далее). Строковые константы не должны содержать символов перевода строки, если они не замаскированы '\'.

TEX и C предлагают различные способы задания восьмеричных и шестнадцатиричных констант, поскольку TEX ориентирован на подготовку текста человеком, а C — на машинную обработку. В TEX константа объявляется восьмеричной или шестнадцатиричной с помощью префикса ' либо " соответственно; в C префиксы имеют вид `0` и `0x`. В CWEB разрешено использовать оба соглашения там, где они приняты: в C-коде чтобы получить 40_8 задается '`040`' и CTANGLE без изменений копирует это в C-файл к удовольствию компилятора, а CWEAVE печатает как $^\circ\!40$. Аналогично, CWEAVE печатает шестнадцатиричную C-константу '`0x20`' как $^\#$20. Использование курсива для восьмеричных цифр и шрифта пишущей машинки для шестнадцатиричных улучшает восприятие этих констант в документе. Для сохранения общности можно использовать '`|040|`' и '`|0x20|`' в TEX-части секции.

Управляющие символы

Управляющий символ (control code) CWEB — это комбинация двух символов, из которых первый '`@`'. Мы уже ознакомились с назначением некоторых из них, а теперь изложим это более точно.
В нижеследующем списке буквы в квадратных скобках указывают контекст, в котором каждый управляющий символ имеет смысл. L означает, что символ допустим в преамбуле, T (TEX), M (middle) и C — в каждой из трех частей секции на верхнем уровне (т.е. вовне конструкции '`|...|`' и имен секций). Стрелка $\to$ обозначает, что управляющий символ завершает текущую часть CWEB-файла и начинает следующую часть, указанную буквой после стрелки. Например, $[LTMC \to T]$ после `@␣` значит, что управляющий код может появиться в преамбуле либо в любой из трех частей секции, и что он начинает TEX-часть (возможно пустую) текущей секции.

Еще два сокращения могут быть указаны в этих квадратных скобках: Буква r означает *ограниченный контекст (restricted context)*, то есть, содержимое C-комментариев, имен секций, C-строк и управляющих текстов (определены далее). Буква c — это *внутренний C-контекст (inner C context)* или C-код внутри'`|...|`' (включая комментарии внутри '`|...|`', но не в других ограниченных контекстах). Звездочка $*$ после скобок указывает, что этим управляющим символом начинается ограниченный контекст, который завершается парной `@>`

Буквы в управляющих символах не различаются по регистру. Так, `@d` означает то же самое, что и `@D`. Далее рассмотрены только символы с буквами нижнего регистра.

`@@` $[LTMCrc]$ Пара символов `@` представляет сам символ '`@`'. Это единственный управляющий код, который действует повсюду. Заметим, что вам придется использовать это соглашение, если вы указываете email-адрес в CWEB-файле (например, `levy@@geom.umn.edu`).

Следующие коды представляют TEX-часть секции.

`@␣` $[LTMC \to T]$ Означает начало новой (не заглавной) секции. Символ табуляции, прогона страницы или конца строки, следующие за знаком `@`, эквивалентны пробелу (как и в большинстве других случаев).

`@*` $[LTMC \to T]$ Отмечает начало новой заглавной секции, т.е. секции, начинающей новую группу. Название этой группы должно следовать за `@*` и завершаться точкой. Как указано выше, в нем не должны использоваться управляющие последовательности TEX'а, кроме достаточно простых. Когда CWEAVE и CTANGLE считывают `@*`, они выводят на терминал символ '*' и номер текущей секции, так что пользователь может визуально следить за продвижением работы. Самая первая секция желательно должна быть заглавной.

Вы можете указать "глубину" заглавной секции, вводя * либо десятиричное число после @*, отмечая этим относительное расположение текущей группы секций в иерархии программы. Части программы верхнего уровня, обозначенные @**, выделяются в оглавлении жирным шрифтом: будем говорить, что их глубина −1. В противном случае, глубина представляет собой неотрицательное число, которое определяет величину отступа в оглавлении. Подобные отступы помогают прояснить структуру длинной программы. Если глубина не указана явно, она полагается равной 0. Если ваша программа коротка, то, по-видимому, возможно будет оставить значение глубины нулевым. Заглавные секции всегда располагаются на новой странице вывода, если только глубина не превышает 1.

Промежуточная часть каждой секции содержит произвольное количество макроопределений (начинающихся с @d) и определений форматов (предваряемых @f либо @s), расположенных в произвольном порядке.

@d $[TM \to M]$ Макроопределение: начинается @d, за которым идут идентификатор, необязательные параметры и C-код, как описано ранее.

@f $[TM \to M]$ Определение формата: начинается @f; они указывают CWEAVE обрабатывать идентификаторы особым образом, когда они встречаются в C-коде. Общий вид определения формата — '@f l r', далее следует необязательный комментарий, ограниченный /* и */, где l и r — идентификаторы: CWEAVE в дальнейшем будет рассматривать идентификатор l так, как он это делает в настоящий момент для r. Это дает CWEB-программисту возможность вводить новые зарезервированные слова и/или отменять резервирование некоторых идентификаторов языком C. Если r — специальный идентификатор '*TeX*', идентификатор l будет отформатирован как управляющая последовательность TEX. Например, '@f foo TeX' в CWEB-файле приведет к тому, что идентификатор *foo* будет выводиться CWEAVE как \foo. Программист должен определить \foo по своему усмотрению, полагая, что TEX находится в математической моде. (Каждый символ подчеркивания преобразуется в x, когда создается управляющая последовательность TEX; так, *foo_bar* становится \fooxbar. Прочие символы, включая цифры, не транслируются, так что TEX будет полагать их параметрами макроса, а не частью самой управляющей последовательности. Например,

```
\def\x#1{x_{#1}} @f x1 TeX @f x2 TeX
```

будет выводить x1 и x2 не как *x1* и *x2*, а как x_1 и x_2.

CWEAVE автоматически форматирует идентификаторы, введенные посредством **typedef**, как зарезервированные слова, поэтому вам не придется определять форматы слишком часто.

@s $[TM \to M;\ L]$ Аналогично @f, но CWEAVE не отображает определения форматов в своем выводе, и необязательный C-комментарий недопустим. Используется главным образом в @i-файлах.

Следующие коды управляют C-частью секции.

@c @p $[TM \to C]$ C-часть неименованной секции начинается @c (либо @p, т.е. "program"; оба управляющих символа делают одно и то же). Это заставляет CTANGLE дополнить следующим C-кодом первое приближение текста программы, как описано на странице 3. Заметим, что CWEAVE не помещает '@c' в TEX-вывод, так что если вы создаете файл CWEB, основываясь на распечатанной TEX'ом CWEB-документации, не забудьте указать @c в соответствующих местах неименованных секций.

@< $[TM \to C;\ C;\ c]$ * Этот управляющий символ представляет имя секции (либо уникальный префикс, как указано выше), содержащее текст TEX и завершающееся @>. Конструкция @<...@> в целом концептуально является элементом C-кода. Ее поведение зависит от контекста:

Символ @< в контекстах T и M связывает следующее далее имя секции с текущей секцией, и открывает C-часть секции. Имя завершается @>, за которым следует = или +=.

В контексте C, @< указывает, что секция здесь используется — ее C-определения будут помещены сюда CTANGLE, как описано на странице 3. Для более полного выявления ошибок CTANGLE и CWEAVE предусматривают вывод сообщения в случае, если за таким именем секции следует =, поскольку это скорее всего означает определение новой секции, и поэтому должно предваряться @␣. Если вам действительно нужно написать $\langle\,\text{foo}\,\rangle = bar$, где $\langle\,\text{foo}\,\rangle$ используется, а не определяется, поместите = на новой строке.

Наконец, во внутреннем C-контексте (т.е., внутри '|...|' в TEX-части секции или в комментарии), @<...@> означает, что эта сеция цитируется. Такое использование игнорируется CTANGLE.

Отметим, что даже в этом случае мы трактуем имя секции как элемент C, так как оно ограничено `|...|`.

`@(` $[TM \to C;\ C;\ c]$ $*$ Имя секции может начинаться `@(`. Все будет работать, как и для `@<`, но с тем исключением, что C-код секции, названной `@(foo@>` будет записан `CTANGLE` в файл `foo`. Тем самым мы можем получить из одного файла `CWEB` несколько выходных файлов. (Определения `@d` не попадают в такие файлы, а лишь в основной `.c`-файл.) Одно из предназначений этой возможности в том, чтобы создать заголовочные файлы для программных модулей. Другим примером ее использования служит генерация тестов, распространяемых с вашей программой. Сохраняя исходные тексты программы, ее заголовочные файлы и тестовые процедуры совместно, мы имеем больше шансов поддержать их согласованность. Отметим, что именованная секция может быть выведена в несколько различных выходных файлов, поскольку вы можете упомянуть `@<foo@>` как в `@(bar1@>`, так и в `@(bar2@>`.

`@h` $[Cc]$ Предписывает `CTANGLE` вставить в эту точку определения `#define` из промежуточных частей всех секций и *не* записывать их в начало C-файла. Полезно, к примеру, при необходимости разместить макроопределения после включаемых файлов. (Игнорируется `CTANGLE` внутри '`|...|`'.)

Несколько следующих управляющих символов начинают "управляющие тексты" ("control texts"), завершающиеся '`@>`'. Закрывающая '`@>`' должна быть на той же строке `CWEB` файла, что и начало управляющего текста. Контекст, начинающийся этими управляющими символами и оканчивающийся `@>`, является ограниченным.

`@^` $[TMCc]$ $*$ Дальнейший управляющий текст, вплоть до '`@>`', будет помещен в индекс наряду с идентификаторами C-программы; этот текст будет набран романским шрифтом. Например, чтобы поместить фразу "system dependencies" (системные зависимости) в индекс, генерируемый `CWEAVE`, поместите '`@^system dependencies@>`' в каждой секции, котрую вы хотите отметить как системно-зависимую.

`@.` $[TMCc]$ $*$ Последующий управляющий текст будет помещен в индекс в стиле пишущей машинки (`typewriter type`).

`@:` $[TMCc]$ $*$ Соответствующий текст будет помещен в индекс в формате, описанном TEX-макросом '`\9`', который вы должны определить так, как это необходимо.

`@t` $[MCc]$ $*$ Дальнейший управляющий текст будет заключен в TEX `\hbox` и отформатирован как часть окружающей его C-программы. Этот текст игнорируется `CTANGLE`, но может использоваться в различных целях `CWEAVE`. Например, вы можете делать комментарии, которые сочетают C и классическую математику, вроде '$size < 2^{15}$', вводя '`|size < 2@t$^{15}$@>|`'.

`@=` $[MCc]$ $*$ Управляющий текст будет дословно выведен в C-программу.

`@q` $[LTMCc]$ $*$ Управляющий текст полностью игнорируется — это комментарий исключительно для читателей `CWEB`-файла. Файл, предназначенный для включения в преамбулу с помощью `@i`, может содержать информацию о себе самом в `@q` комментарии. Другим использованием этой конструкции может быть восстановление баланса скобок внутри C-строк, чтобы текстовые редакторы, отслеживающие их парность, "не падали в штопор" (да, именно так: "go into a tailspin" — *Прим. перев.*).

`@!` $[TMCc]$ $*$ Номер секции в индексе будет подчеркнут, если '`@!`' непосредственно предшествует идентификатору либо управляющему тексту, помещаемому в индекс. Это соглашение используют, чтобы различать секции, в которых идентификатор определен либо объясняется некоторым особым образом, от секций, где он использован. Ссылки на зарезервированные слова либо односимвольные идентификаторы в индекс не помещаются, за исключением подчеркнутых. '`@!`' неявно вставляется `CWEAVE` когда идентификатор объявляется либо определяется в C-коде; например, определение

$$\mathbf{int}\ \mathit{array}[\mathit{max\_dim}],\ \mathit{count} = \mathit{old\_count};$$

помещает в индекс имена *array* и *count*, выделяя их подчеркиванием. Метки, определения функций, подобные $\mathit{main}(\mathbf{int}\ \mathit{argc}, \mathbf{char}\ {*}\mathit{argv}[\,])$, и определения **typedef** также подразумевают подчеркивание. Определения функций старого стиля (без прототипа) не определяют их аргументов, тем не менее, аргументы будут полагаться определенными (и будут отображаться в индексе с подчеркиванием), если их типы объявлены перед телом функции обычным образом, (например, '$\mathbf{int}\ \mathit{argc};\ \mathbf{char}\ {*}\mathit{argv}[\,];\ \{\ldots\}$').

Перейдем, наконец, к управляющим символам, которые влияют только на работу CTANGLE.

@' [MCc] Использование этого управляющего символа опасно, поскольку его значение в CWEB существенно отличается от его прообраза в WEB. В CWEB он раскрывается в десятиричную константу, соответствующую ASCII-коду строки единичной длины (т.е. @'a' преобразуется CTANGLE в 97, а @'\t' — в 9). Вы можете воспользоваться этим, если хотите работать с ASCII на машине, не использующей эту кодировку, но в большинстве случаев средства языка C, определенные в <ctype.h>, вполне достаточны для устранения зависимости от кодировки символов.

@& [MCc] Операция @& сцепляет текст слева от себя с текстом справа и выводит в C-файл так, что ни пробелы, ни маркеры конца строки не будут вставлены между ними.

@l [L] CWEB-программисты вольны использовать в TEXте любой 8-битный символьный код из нередко запретного диапазона 128–255; эти символы также допустимы в строках и даже в идентификаторах C-программы. В различных расширениях базового стандарта ASCII нижняя часть таблицы (коды, большие 128) соответствует буквам с диакритическими знаками, символам не латинских алфавитов и т.д. Когда такие символы появляются в идентификаторах, CTANGLE заменяет их стандартными алфавитно-цифровыми символами ASCII либо _, чтобы имена соответствовали соглашениям языка C. При этом применяется таблица перекодировки, которая по умолчанию сопоставляет с каждым символом, имеющим ASCII-код $^{\#}ab$ (где a и b — шестнадцатеричные цифры и $a \geq 8$) строку Xab. Поместив в преамбуле конструкцию @l␣ab␣newstring, вы укажете CTANGLE заменять этот символ строкой newstring. Например, в кодировке ISO Latin-1 буква 'ü' имеет код #FC (либо '\374'), и CTANGLE по умолчанию заменяет в идентификаторах этот код последовательностью символов XFC. Если вы зададите @l fc ue, то далее вместо этой буквы будет подставляться ue.

CWEAVE пропускает 8-битные символы прямо в TEX без транслитерации; но при этом TEX должен быть, в свою очередь, подготовлен к их обработке. Если вы форматируете все нестандартные идентификаторы как особые ("custom") управляющие последовательности, следует настроить TEX так, чтобы он воспринимал все их символы как буквы. В противном случае следует либо объявить в TEX'е 8-битные символы "активными", либо загрузить шрифты, содержащие в соответствующей позиции нужные специальные символы. (Для идентификаторов используется шрифт, выбираемый командой TEX'а \it.) Ищите специальные макропакеты, написанные для использования в CWEB вашего языка либо, если не боитесь трудностей, напишите свой собственный.

Следующие восемь управляющих символов (а именно, '@,', '@/', '@|', '@#', '@+', '@;', '@[', и '@]') не влияют на генерацию C-программы CTANGLE, но в некоторых необычных случаях могут помочь улучшить читаемость ее TEX-представления, полученного CWEAVE. Встоенный алгоритм форматирования весьма хорош, когда применяется к синтаксически правильной программе, но не может учесть всех возможных ситуаций, потому что работает с фрагментами текста, включающими макросы и имена секций; такие фрагменты не всегда соблюдают синтаксис. Несмотря на то, что CWEB позволяет перекрывать автоматическое форматирование, лучшей стратегией будет начать с него, поскольку может оказаться, что понадобятся лишь несколько мелких корректив.

@, [MCc] Этот управляющий символ вставляет тонкий пробел (thin space) в вывод CWEAVE. Иногда дополнительный пробел может понадобиться при необычном использовании макросов, например, когда два идентификатора расположены рядом.

@/ [MC] Этот управляющий символ вызывает разрыв строки в C-программе, отформатированной CWEAVE. Разрывы строк автоматически выбираются TEX'ом по схеме, работающей корректно в 99% случаев, но иногда необходимо явное указание, чтобы разбиение соответствовало, помимо внешней привлекательности, логике программы. Например, используйте '@/@,' перед комментариями, чтобы они начинались с новой строки.

@| [MC] Этот управляющий символ задает необязательный разрыв строки посреди выражения. Например, если имеется длинное выражение в правой части присваивания, с помощью '@|' можно указать в нем возможные точки разрыва, которые будут лучше подчеркивать логику программы, нежели выбранные TEX'ом автоматически.

@# [MC] Этот управляющий символ вызывает разрыв строки подобно @/, а также вставляет небольшой дополнительный пробел в точку разрыва. Он может использоваться, например, между логически различными группами макросов, размещенными в одной секции. CWEB автоматически вставляет такой дополнительный пробел между функциями, внешними объявлениями и функциями, а также между объявлениями и операторами в теле функции.

`@+` [*MC*] Этот управляющий символ маскирует разрыв строки, который должен быть сделан CWEAVE, например, перед '**else**', если вы хотите разместить короткую if–else конструкцию на одной строке. Если указать '`{@+`' в начале составного оператора, являющегося телом функции, то первое объявление или оператор функции будут расположены на той же строке, что и открывающая скобка, и с отступом, который будет предшествовать второму оператору или объявлению на следующей строке.

`@;` [*MC*] Этот управляющий символ при форматировании аналогичен точке с запятой, но невидим. Вы можете воспользоваться ним, например, после имени секции или макроса, когда С-код, представленный в этой секции или макросе, есть составной оператор либо оканчивается точкой с запятой. Примером такой конструкции является

```
if (condition) macro @;
else break;
```

где определением *macro* служит составной оператор (заключенный в фигурные скобки). Это хорошо известный недостаток синтаксиса С.

`@[` [*MC*] См. `@]`.

`@]` [*MC*] Поместите квадратные скобки `@[...@]` вокруг текста программы, который должен быть отформатирован CWEAVE как выражение, если этого не произошло. (Такое иногда случается с необычными аргументами макросов.) Также вставляйте '`@[@]`' между именем типа и левой скобкой, когда объявляете указатель на функцию, вроде,

```
int @[@] (*f)();
```

в противном случае CWEAVE спутает первую часть этого определения с выражением С++ '**int**$(*f)$'.

Остальные управляющие символы CWEB руководят вводом данных.

`@x @y @z` [*change_file*] CWEAVE и CTANGLE предназначены для работы с двумя входными файлами, называемыми *web_file* и файл изменений *change_file*, где *change_file* содержит данные, замещающие отдельные части *web_file*. Результирующий текст и есть тот самый файл CWEB, который упоминается повсеместно в этом руководстве.

Как это работает: файл изменений содержит нуль и более "корректив", где корректива имеет вид '`@x`⟨старые строки⟩`@y`⟨новые строки⟩`@z`'. Специальные управляющие символы `@x`, `@y`, `@z`, допустимые только в файлах изменений, должны появляться в начале строки, остаток такой строки игнорируется. Содержимое ⟨старые строки⟩ представляет материал, который точно соответствует последовательным строкам *web_file*; а ⟨новые строки⟩ содержит нуль или более строк, которыми предполагается заменить старые. Как только первая "старая строка" коррективы оказывается совпадающей со строкой в *web_file*, все остальные строки этой коррективы также должны совпасть.

Между коррективами, перед первой коррективой и после последней коррективы файл изменений может содержать любое количество строк, не начинающихся '`@x`', '`@y`', либо '`@z`'. Такие строки пропускаются и не используются в сопоставлении.

Такая возможность полезна, когда работаешь с оригинальным CWEB-файлом, полученным откуда-либо (например, `tangle.w` и `weave.w` либо `tex.web`), который необходимо изменить для адаптации программы к особенностям вашей системы. Вы можете отлаживать свои системно-зависимые изменения, не исправляя основного файла; и как только они заработают, их можно будет легко включать в новые версии основного файла, которые будут появляться время от времени.

`@i` [*web_file*] В свою очередь, *web_file* сам по себе может быть комбинацией нескольких файлов. Когда CWEAVE либо CTANGLE, считывая файл, обнаруживают управляющий символ `@i` в начале строки, они прерывают чтение и начинают обработку файла, имя которого следует за `@i`, подобно препроцессору компилятора С, обрабатывающему директиву `#include`. После завершения чтения включаемого файла программа переходит к следующей строке исходного файла. Имя файла, следующее за `@i`, может окружаться символами ", но это необязательно. Включение может быть вложенным.

Файлы изменений могут иметь строки, начинающиеся `@i`. Таким способом вы можете подменять один включаемый файл другим. Концептуально, описанный выше механизм подстановки действует первым, а затем в его выводе обрабатываются директивы `@i`. Если `@i foo` появится между `@y` и `@z` в файле изменений, отдельные строки файла `foo` и файлы, которые он включает, не изменяются; модификации могут быть внесены в строки, котрые включались не измененной директивой input.

В системе UNIX (и других, поддерживающих переменные окружения), если установлена переменная `CWEBINPUTS`, либо если одноименное определение было задано при компиляции, `CWEB` будет использовать ее значение в качестве имени каталога, в котором будет происходить поиск включаемых файлов, если они не будут найдены в текущем каталоге.

Дополнительные возможности и предостережения

1. В некоторых реализациях `CWEB`, которые допускают использование расширенного символьного набора, символы ‘↑’, ‘↓’, ‘→’, ‘≠’, ‘≤’, ‘≥’, ‘≡’, ‘∨’, ‘∧’, ‘⊂’ и ‘⊃’ могут применяться как сокращения для ‘`++`’, ‘`--`’, ‘`->`’, ‘`!=`’, ‘`<=`’, ‘`>=`’, ‘`==`’, ‘`||`’, ‘`&&`’, ‘`<<`’ и ‘`>>`’ соответственно.

2. Если ваша система имеет расширенный набор символов, он может быть использован с минимальными ограничениями, как это описано выше при обсуждении `@l`. Но вам следует придерживаться стандартных символов ASCII, если вы хотите писать программы, которые окажутся полезными и тем, кому такое расширение недоступно.

3. Документ TeX, произведенный `CWEAVE`, разбивается на строки с максимальной длиной 80 символов. Когда текст TeX копируется, воспроизводятся и существующие разрывы строк. Если вы не делаете ничего слишком хитрого, `CWEAVE` распознает, когда комментарий TeX’а разбивается на несколько строк, и добавит ‘`%`’ в начало каждой такой строки.

4. С-текст транслируется “восходящей” (“bottom up”) процедурой, которая распознает каждую лексему как “часть речи” (“part of speech”) и комбинирует эти части во все бо́льшие и бо́льшие фразы, пока это возможно, согласно специальной грамматике, описанной в документации к `CWEAVE`. Легко понять схему, применяемую к простым конструкциям, таким как отдельные идентификаторы и короткие выражения, просто изучив несколько примером вывода `CWEAVE`, но общий механизм несколько сложнее, поскольку должен обрабатывать нечто большее, чем просто С. Помимо этого, вывод содержит команды, которые заставляют TeX делать отступы и разрывы строк где необходимо, соответственно использованным шрифтам и размеру страницы. Для достижения наилучших результатов благоразумно избегать заключения длинных С-текстов в `|...|`, поскольку отступы и маркеры конца строки игнорируются, когда содержимое `|...|` транслируется с С в TeX. Стремитесь к простоте выражений и операторов. Если директива препроцессора заключена в `|...|`, то начинающий ее `#` должен располагаться в начале строки, иначе `CWEAVE` не напечатает ее правильно.

5. Комментарии в `|...|`-тексте недопустимы. После ‘`|`’, указывающего на переход от текста TeX к С, следующий ‘`|`’, который не является частью строки, управляющего текста либо имени секции, завершает С-текст.

6. Комментарий должен иметь правильное вложение фигурных скобок, в противном случае `CWEAVE` выразит свое недовольство, но при этом сделает попытку восстановить баланс скобок, так что TeX’у не придется слишком трудно.

7. Если во время отладки программы возникает необходимость опустить некоторую часть С-кода, НЕ следует пытаться просто его “закомментировать”. Такие комментарии не в духе `CWEB`; читателям они покажутся пояснениями к остальным инструкциям. Кроме того, комментарии должны быть корректны с точки зрения TeX’а; следовательно, `CWEAVE` будет в затруднении, если вы заключите предложения С в `/*...*/` вместо `/*|...|*/`. Если часть С-кода должна быть “закомментирована”, вы можете ограничить ее директивами препроцессора, подобными `#if 0==1` и `#endif`.

8. При помощи `@f` возможно определить один идентификатор так, чтобы он вел себя подобно другому, и эти определения формата применяются последовательно. Вообще говоря, данный идентификатор имеет единственный формат вывода на протяжении всего документа, и этот формат используется даже до того, как будет определен посредством `@f`. Это объясняется тем, что `CWEAVE` работает в два прохода; на первом проходе обрабатываются директивы `@f` и перекрестные ссылки, а на втором производится вывод. (Тем не менее, идентификаторы, введенные `typedef`, которым неявно присваивается формат вывода жирным шрифтом, не следуют этому правилу: они выводятся по-разному до и после соответствующего `typedef`. Это разочаровывает, но что-либо исправить затруднительно. Вы можете обойти это ограничение, используя `@s`, до либо после `typedef`.)

9. Иногда желательно вставить в отформатированный C-код пробел, отличный от предоставляемого '@,'. Для этого можно воспользоваться `@t`; например, '`@t\hskip 1in@>`' дает пробел длиной 1 дюйм. Кроме того, '`@t\4@>`' может использоваться для возврата на одну единицу отступа, поскольку управляющая последовательность `\4` определена в `cwebmac` так, чтобы делать такой возврат. (Она используется, к примеру, в начале строк, которые содержат помеченные операторы, так что метка будет немного смещена влево.) Вы также можете использовать '`@t}\3{-5@>`' для принудительного разрыва строки посреди выражения.

10. Каждый идентификатор в CWEB имеет один стиль форматирования. Поэтому вы не должны использовать одно и то же имя для одновременного обозначения, например, имени типа и поля `struct`, несмотря на то, что C позволяет это.

Запуск программ

Командная строка UNIX как для CTANGLE, так и для CWEAVE имеет вид

```
ctangle [options] web_file[.w] [{change_file[.ch]|-} [out_file]]
```

Если файл изменений опущен либо задан '-', он полагается пустым. Расширения `.w` и `.ch` добавляются только в том случае, если заданные имена файлов не содержат точек. Если файл с таким именем не найден, будет испробовано расширение `.web`. Например, '`cweave cob`' попытается прочитать `cob.w`; если это не удается, будет испробовано `cob.web` перед аварийным завершением работы. Если имя выходного файла не задано, имя C-файла, генерируемого CTANGLE, получается добавлением расширения `.c`; имя TEX-файла, произведенного CWEAVE, получит расширение `.tex`. В индексных файлах, порождаемых CWEAVE, `.tex` заменяется `.idx` и `.scn`.

Приверженцы лаконичности могут так настроить свой командный интерпретатор, что '`wv`' будет означать '`cweave -bhp`'; это подавляет весь вывод CWEAVE на терминал, за исключением сообщений об ошибках.

Параметры-опции начинаются либо символом '-', запрещающим действие опции, либо '+', который его разрешает. Например, '`-fb`' запрещает опции `f` и `b`; '`+s`' разрешает опцию `s`. Опции могут задаваться перед именами файлов, после них, либо и то, и другое. В настоящий момент поддерживаются следующие опции:

- `b` Выводить сообщение при старте. (Разрешено по умолчанию.)
- `f` Принуждает завершать каждый оператор C, отформатированный CWEAVE, маркером конца строки. (Разрешено по умолчанию; `-f` экономит бумагу, но многие полагают, что такой стиль форматирования программы выглядит в C "неестественно".) (Не влияет на CTANGLE.)
- `h` Вывод сообщения о нормальном завершении программы. (Разрешен по умолчанию.)
- `p` Отображать во время работы программы сообщения о ее прогрессе. (Разрешено по умолчанию.)
- `s` Выдает статистику использования памяти по окончании работы программы. (Запрещено по умолчанию.) Если ваши CWEB-файлы либо секции велики, может оказаться полезным увидеть, насколько вы близки к исчерпанию ресурсов CTANGLE и/или CWEAVE.
- `x` Включать индекс и оглавление в TEX-файл, выводимый CWEAVE. (Разрешено по умолчанию.) (Не влияет на CTANGLE.)

Дополнительные сведения о форматировании

Вам может не понравиться, как CWEAVE обрабатывает некоторые ситуации. Если у вас нет другого выхода, вы можете настроить CWEAVE изменяя его грамматику. Это требует модификации исходных текстов, работы, которая может показаться занимательной. Таблица грамматических правил содержится в исходных текстах CWEAVE, и вы можете сделать отдельную копию этой таблицы, скопировав файл `prod.w`, найденный в исходных текстах CWEB и скомандовав '`cweave -x prod`' и '`tex prod`'.

Вы можете точно узнать, как CWEAVE разбирает ваш C-код, если поместите перед ним строку '`@ @c @2`'. (Управляющий код '`@2`' активизирует "следящий" ("peeping") режим работы, а '`@0`' отменяет его.) Например, если пропустить через CWEAVE файл

```
@ @c @2
main (argc,argv)
char **argv;
{ for (;argc>0;argc--) printf("%s\n",argv[argc-1]); }
```

на экран будет выведена следующая тарабарщина:

```
[...]
4:*exp ( +exp+ )...
11:*exp +exp+ int...
5:*+exp+ int +unorbinop+...
[...]
60: +fn_decl+*+{+ -stmt- +}-
55:*+fn_decl+ -stmt-
52:*+function-
[...]
```

Первая строка указывает, что было применено правило номер 4, и `CWEAVE` в настоящий момент содержит в своей памяти последовательность фрагментов (chunk) текста в формате TEX (называемых "вырезками", "scraps"), имеющих соответственно тип *exp* (для выражений), открывающая скобка, снова *exp*, закрывающая скобка, и последующие вырезки, которые не были еще считаны синтаксическим анализатором. (Знаки `+` и `-` указывают, что TEX должен войти в математическую моду либо выйти из нее на границе вырезки. Символ `*` отмечает текущую позицию анализатора.) Далее применяется правило 11, и последовательность (*exp*) свертывается до *exp* и так далее. В конце концов весь С-текст оказывается одной большой вырезкой типа *function*.

Иногда все идет не так гладко, и вы получаете группу строк, сваленных в одну кучу. Это значит, что `CWEAVE` не смог переварить что-то в вашем С-коде. Например, пусть '`@<Argument definitions@>`' займет место '`char **argv;`' в программе, упомянутой выше. В этом случае `CWEAVE` будет слегка озадачен, поскольку считает, что имена секций всего лишь только *exp*. Поэтому TEX'у будет предписано вывести '⟨ Argument declarations 2 ⟩' в той же самой строке, что и '*main*(*argc*, *argv*)'. В этом случае можно помочь `CWEAVE`, помещая '`@/`' после '`main(argc,argv)`'.

`CWEAVE` автоматически вставляет небольшой промежуток между описаниями и первым оператором блока. Вот один способ локально его отменить:

```
int x;@+@t}\6{@>
@<Other locals@>@;@#
```

'`@#`' поместит дополнительный пробел после '⟨ Other locals ⟩'.

Приложения

Как пример реальной программы, написанной в `CWEB`, Приложение A содержит фрагменты программ, входящих в состав самой системы `CWEB`. Те из читателей, кто тщательно изучат листинги в этом приложении, смогут глубоко прочувствовать основные идеи `CWEB`.

Приложение B — это файл определений, которые настраивают TEX на работу с выводом `CWEAVE`, а в Приложении C обсуждается, как использовать некоторые из этих макросов для изменения формата вывода.

"Полная" версия этого руководства, которая может быть сгенерирована из исходных текстов `CWEB` командой UNIX `make fullmanual`, содержит также приложения D, E, и F, представляющие собой полные исходные тексты `CTANGLE` и `CWEAVE`.

Приложение А: Фрагменты программ CWEB

Это приложение содержит четыре листинга. Первый содержит входные данные для CWEB, по которым генерируются секции 12–15 файла `common.w`, который содержит подпрограммы, общие для CWEAVE и CTANGLE. Отметим, что некоторые строки содержат отступы, подчеркивающие структуру программы; отступы эти игнорируются как CWEAVE, так и CTANGLE, но пользователи находят, что файлы CWEB гораздо легче воспринимаются, если в них имеются такие отступы.

Второй и третий листинги содержат соответствующие фрагменты C-кода, сгенерированного CTANGLE и текста в формате TeX, полученного CWEAVE при обработке `common.w`. Четвертый листинг демонстрирует, как будет выглядеть этот выходной файл на бумаге.

```
@ Procedure |prime_the_change_buffer|
sets |change_buffer| in preparation for the next matching operation.
Since blank lines in the change file are not used for matching, we have
|(change_limit==change_buffer && !changing)| if and only if
the change file is exhausted. This procedure is called only when
|changing| is 1; hence error messages will be reported correctly.

@c
void
prime_the_change_buffer()
{
  change_limit=change_buffer; /* this value is used if the change file ends */
  @<Skip over comment lines in the change file; |return| if end of file@>;
  @<Skip to the next nonblank line; |return| if end of file@>;
  @<Move |buffer| and |limit| to |change_buffer| and |change_limit|@>;
}

@ While looking for a line that begins with \.{@@x} in the change file, we
allow lines that begin with \.{@@}, as long as they don't begin with \.{@@y},
\.{@@z} or \.{@@i} (which would probably mean that the change file is fouled up).

@<Skip over comment lines in the change file...@>=
while(1) {
  change_line++;
  if (!input_ln(change_file)) return;
  if (limit<buffer+2) continue;
  if (buffer[0]!='@@') continue;
  if (xisupper(buffer[1])) buffer[1]=tolower(buffer[1]);
  if (buffer[1]=='x') break;
  if (buffer[1]=='y' || buffer[1]=='z' || buffer[1]=='i') {
    loc=buffer+2;
    err_print("! Missing @@x in change file");
@.Missing @@x...@>
  }
}

@ Here we are looking at lines following the \.{@@x}.

@<Skip to the next nonblank line...@>=
do {
  change_line++;
  if (!input_ln(change_file)) {
    err_print("! Change file ended after @@x");
@.Change file ended...@>
    return;
  }
} while (limit==buffer);

@ @<Move |buffer| and |limit| to |change_buffer| and |change_limit|@>=
{
  change_limit=change_buffer-buffer+limit;
  strncpy(change_buffer,buffer,limit-buffer+1);
}
```

Здесь приведена порция С-кода, сгенерированного CTANGLE, который соответствует исходному тексту на предыдущей странице. Обратите внимание, что секции 13, 14 и 15 были "вплетены" в секцию 12.

```
/*:9*//*12:*/
#line 247 "common.w"

void
prime_the_change_buffer()
{
change_limit= change_buffer;
/*13:*/
#line 261 "common.w"

while(1){
change_line++;
if(!input_ln(change_file))return;
if(limit<buffer+2)continue;
if(buffer[0]!='@')continue;
if(xisupper(buffer[1]))buffer[1]= tolower(buffer[1]);
if(buffer[1]=='x')break;
if(buffer[1]=='y'||buffer[1]=='z'||buffer[1]=='i'){
loc= buffer+2;
err_print("! Missing @x in change file");

}
}

/*:13*/
#line 252 "common.w"
;
/*14:*/
#line 278 "common.w"

do{
change_line++;
if(!input_ln(change_file)){
err_print("! Change file ended after @x");

return;
}
}while(limit==buffer);

/*:14*/
#line 253 "common.w"
;
/*15:*/
#line 288 "common.w"

{
change_limit= change_buffer-buffer+limit;
strncpy(change_buffer,buffer,limit-buffer+1);
}

/*:15*/
#line 254 "common.w"
;
}

/*:12*//*16:*/
```

Соответствующий отрывок common.tex выглядит так:

```
\M{12}Procedure \PB{\\{prime\_the\_change\_buffer}}
sets \PB{\\{change\_buffer}} in preparation for the next matching operation.
Since blank lines in the change file are not used for matching, we have
\PB{$(\\{change\_limit}\E\\{change\_buffer}\W\R\\{changing})$} if and only if
the change file is exhausted. This procedure is called only when
\PB{\\{changing}} is 1; hence error messages will be reported correctly.

\Y\B\&{void} \\{prime\_the\_change\_buffer}(\,)\1\1\2\2\6
${}\{{}$\1\6
${}\\{change\_limit}\K\\{change\_buffer}{}$;\C{ this value is used if the
change file ends }\6
\X13:Skip over comment lines in the change file; \PB{\&{return}} if end of file%
\X;\6
\X14:Skip to the next nonblank line; \PB{\&{return}} if end of file\X;\6
\X15:Move \PB{\\{buffer}} and \PB{\\{limit}} to \PB{\\{change\_buffer}} and %
\PB{\\{change\_limit}}\X;\6
\4${}\}{}$\2\par
\fi

\M{13}While looking for a line that begins with \.{@x} in the change file, we
allow lines that begin with \.{@}, as long as they don't begin with \.{@y},
\.{@z} or \.{@i} (which would probably mean that the change file is fouled up).

\Y\B\4\X13:Skip over comment lines in the change file; \PB{\&{return}} if end
of file\X${}\E{}$\6
\&{while} (\T{1})\5
${}\{{}$\1\6
${}\\{change\_line}\PP;{}$\6
\&{if} ${}(\R\\{input\_ln}(\\{change\_file})){}$\1\5
\&{return};\2\6
\&{if} ${}(\\{limit}<\\{buffer}+\T{2}){}$\1\5
\&{continue};\2\6
\&{if} ${}(\\{buffer}[\T{0}]\I\.{'@'}){}$\1\5
\&{continue};\2\6
\&{if} (\\{xisupper}(\\{buffer}[\T{1}]))\1\5
${}\\{buffer}[\T{1}]\K\\{tolower}(\\{buffer}[\T{1}]);{}$\2\6
\&{if} ${}(\\{buffer}[\T{1}]\E\.{'x'}){}$\1\5
\&{break};\2\6
\&{if} ${}(\\{buffer}[\T{1}]\E\.{'y'}\V\\{buffer}[\T{1}]\E\.{'z'}\V\\{buffer}[%
\T{1}]\E\.{'i'}){}$\5
${}\{{}$\1\6
${}\\{loc}\K\\{buffer}+\T{2};{}$\6
\\{err\_print}(\.{"!\ Missing\ @x\ in\ cha}\)\.{nge\ file"});\6
\4${}\}{}$\2\6
\4${}\}{}$\2\par
\U12.\fi

\M{14}Here we are looking at lines following the \.{@x}.

\Y\B\4\X14:Skip to the next nonblank line; \PB{\&{return}} if end of file\X${}%
\E{}$\6
\&{do}\5
${}\{{}$\1\6
${}\\{change\_line}\PP;{}$\6
\&{if} ${}(\R\\{input\_ln}(\\{change\_file})){}$\5
${}\{{}$\1\6
\\{err\_print}(\.{"!\ Change\ file\ ended}\)\.{\ after\ @x"});\6
\&{return};\6
\4${}\}{}$\2\6
\4${}\}{}$\2\5
\&{while} ${}(\\{limit}\E\\{buffer}){}$;\par
\U12.\fi

\M{15}\B\X15:Move \PB{\\{buffer}} and \PB{\\{limit}} to \PB{\\{change\_buffer}}
and \PB{\\{change\_limit}}\X${}\E{}$\6
${}\{{}$\1\6
${}\\{change\_limit}\K\\{change\_buffer}-\\{buffer}+\\{limit};{}$\6
${}\\{strncpy}(\\{change\_buffer},\39\\{buffer},\39\\{limit}-\\{buffer}+%
\T{1});{}$\6
\4${}\}{}$\2\par
\Us12\ET16.\fi
```

А вот так этот же самый фрагмент выглядит при печати.

12. Procedure *prime_the_change_buffer* sets *change_buffer* in preparation for the next matching operation. Since blank lines in the change file are not used for matching, we have ($change\_limit \equiv change\_buffer \land \neg changing$) if and only if the change file is exhausted. This procedure is called only when *changing* is 1; hence error messages will be reported correctly.

```
void prime_the_change_buffer()
{
  change_limit = change_buffer;      /* this value is used if the change file ends */
  ⟨ Skip over comment lines in the change file; return if end of file 13 ⟩;
  ⟨ Skip to the next nonblank line; return if end of file 14 ⟩;
  ⟨ Move buffer and limit to change_buffer and change_limit 15 ⟩;
}
```

13. While looking for a line that begins with `@x` in the change file, we allow lines that begin with `@`, as long as they don't begin with `@y`, `@z` or `@i` (which would probably mean that the change file is fouled up).

```
⟨ Skip over comment lines in the change file; return if end of file 13 ⟩ ≡
  while (1) {
    change_line++;
    if (¬input_ln(change_file)) return;
    if (limit < buffer + 2) continue;
    if (buffer[0] ≠ '@') continue;
    if (xisupper(buffer[1])) buffer[1] = tolower(buffer[1]);
    if (buffer[1] ≡ 'x') break;
    if (buffer[1] ≡ 'y' ∨ buffer[1] ≡ 'z' ∨ buffer[1] ≡ 'i') {
      loc = buffer + 2;
      err_print("!␣Missing␣@x␣in␣change␣file");
    }
  }
```

This code is used in section 12.

14. Here we are looking at lines following the `@x`.

```
⟨ Skip to the next nonblank line; return if end of file 14 ⟩ ≡
  do {
    change_line++;
    if (¬input_ln(change_file)) {
      err_print("!␣Change␣file␣ended␣after␣@x");
      return;
    }
  } while (limit ≡ buffer);
```

This code is used in section 12.

15. ⟨ Move *buffer* and *limit* to *change_buffer* and *change_limit* 15 ⟩ ≡

```
  {
    change_limit = change_buffer − buffer + limit;
    strncpy(change_buffer, buffer, limit − buffer + 1);
  }
```

This code is used in sections 12 and 16.

Приложение B: файл cwebmac.tex

Этот файл расширяет формат "plain TeX" определениями, необходимыми для обработки вывода CWEAVE.

```
% standard macros for CWEB listings (in addition to plain.tex)
% Version 3.1 --- September 1994
\ifx\documentstyle\undefined\else\endinput\fi % LaTeX will use other macros
\xdef\fmtversion{\fmtversion+CWEB3.1}

\let\:=\. % preserve a way to get the dot accent
 % (all other accents will still work as usual)

\parskip 0pt % no stretch between paragraphs
\parindent 1em % for paragraphs and for the first line of C text

\font\ninerm=cmr9
\let\mc=\ninerm % medium caps
\def\CEE/{{\mc C\spacefactor1000}}
\def\UNIX/{{\mc U\kern-.05emNIX\spacefactor1000}}
\def\TEX/{\TeX}
\def\CPLUSPLUS/{{\mc C\PP\spacefactor1000}}
\def\Cee{\CEE/} % for backward compatibility
\def\9#1{}
\font\eightrm=cmr8
\let\sc=\eightrm % small caps (NOT a caps-and-small-caps font)
\let\mainfont=\tenrm
\let\cmntfont\tenrm
%\font\tenss=cmss10 \let\cmntfont\tenss % alternative comment font
\font\titlefont=cmr7 scaled\magstep4 % title on the contents page
\font\ttitlefont=cmtt10 scaled\magstep2 % typewriter type in title
\font\tentex=cmtex10 % TeX extended character set (used in strings)
\fontdimen7\tentex=0pt % no double space after sentences

\def\\#1{\leavevmode\hbox{\it#1\/\kern.05em}} % italic type for identifiers
\def\|#1{\leavevmode\hbox{$#1$}} % one-letter identifiers look better this way
\def\&#1{\leavevmode\hbox{\bf
  \def\_{\kern.04em\vbox{\hrule width.3em height .6pt}\kern.08em}%
  #1\/\kern.05em}} % boldface type for reserved words
\def\.#1{\leavevmode\hbox{\tentex % typewriter type for strings
  \let\\=\BS % backslash in a string
  \let\{=\LB % left brace in a string
  \let\}=\RB % right brace in a string
  \let\~=\TL % tilde in a string
  \let\ =\SP % space in a string
  \let\_=\UL % underline in a string
  \let\&=\AM % ampersand in a string
  \let\^=\CF % circumflex in a string
  #1\kern.05em}}
\def\){\discretionary{\hbox{\tentex\BS}}{}{}}
\def\AT{@} % at sign for control text (not needed in versions >= 2.9)
\def\ATL{\par\noindent\bgroup\catcode`\_=12 \postATL} % print @l in limbo
\def\postATL#1 #2 {\bf letter \\{\uppercase{\char"#1}}
   tangles as \tentex "#2"\egroup\par}
\def\noATL#1 #2 {}
\def\noatl{\let\ATL=\noATL} % suppress output from @l
\def\ATH{\X\kern-.5em:Preprocessor definitions\X}
\let\PB=\relax % hook for program brackets |...| in TeX part or section name

\chardef\AM=`\& % ampersand character in a string
\chardef\BS=`\\ % backslash in a string
\chardef\LB=`\{ % left brace in a string
\chardef\RB=`\} % right brace in a string
\def\SP{{\tt\char`\ }} % (visible) space in a string
\chardef\TL=`\~ % tilde in a string
\chardef\UL=`\_ % underline character in a string
\chardef\CF=`\^ % circumflex character in a string

\newbox\PPbox % symbol for ++
\setbox\PPbox=\hbox{\kern.5pt\raise1pt\hbox{\sevenrm+\kern-1pt+}\kern.5pt}
\def\PP{\copy\PPbox}
\newbox\MMbox \setbox\MMbox=\hbox{\kern.5pt\raise1pt\hbox{\sevensy\char0
```

```
 \kern-1pt\char0}\kern.5pt}
\def\MM{\copy\MMbox}
\newbox\MGbox % symbol for ->
\setbox\MGbox=\hbox{\kern-2pt\lower3pt\hbox{\teni\char'176}\kern1pt}
\def\MG{\copy\MGbox}
\def\MRL#1{\mathrel{\let\K==#1}}
%\def\MRL#1{\KK#1}\def\KK#1#2{\buildrel\;#1\over{#2}}
\let\GG=\gg
\let\LL=\ll
\let\NULL=\Lambda
\mathchardef\AND="2026 % bitwise and; also \& (unary operator)
\let\OR=\mid % bitwise or
\let\XOR=\oplus % bitwise exclusive or
\def\CM{{\sim}} % bitwise complement
\newbox\MODbox \setbox\MODbox=\hbox{\eightrm\%}
\def\MOD{\mathbin{\copy\MODbox}}
\def\DC{\kern.1em{::}\kern.1em} % symbol for ::
\def\PA{\mathbin{.*}} % symbol for .*
\def\MGA{\mathbin{\MG*}} % symbol for ->*
\def\this{\&{this}}

\newbox\bak \setbox\bak=\hbox to -1em{} % backspace one em
\newbox\bakk\setbox\bakk=\hbox to -2em{} % backspace two ems

\newcount\ind % current indentation in ems
\def\1{\global\advance\ind by1\hangindent\ind em} % indent one more notch
\def\2{\global\advance\ind by-1} % indent one less notch
\def\3#1{\hfil\penalty#10\hfilneg} % optional break within a statement
\def\4{\copy\bak} % backspace one notch
\def\5{\hfil\penalty-1\hfilneg\kern2.5em\copy\bakk\ignorespaces}% optional break
\def\6{\ifmmode\else\par % forced break
  \hangindent\ind em\noindent\kern\ind em\copy\bakk\ignorespaces\fi}
\def\7{\Y\6} % forced break and a little extra space
\def\8{\hskip-\ind em\hskip 2em} % no indentation

\newcount\gdepth % depth of current major group, plus one
\newcount\secpagedepth
\secpagedepth=3 % page breaks will occur for depths -1, 0, and 1
\newtoks\gtitle % title of current major group
\newskip\intersecskip \intersecskip=12pt minus 3pt % space between sections
\let\yskip=\smallskip
\def\?{\mathrel?}
\def\note#1#2.{\Y\noindent{\hangindent2em\baselineskip10pt\eightrm#1~#2.\par}}
\def\lapstar{\rlap{*}}
\def\stsec{\rightskip=0pt % get out of C mode (cf. \B)
  \sfcode`;=1500 \pretolerance 200 \hyphenpenalty 50 \exhyphenpenalty 50
  \noindent{\let\*=\lapstar\bf\secstar.\quad}}
\let\startsection=\stsec
\def\defin#1{\global\advance\ind by 2 \1\&{#1 } } % begin `define' or `format'
\def\A{\note{See also section}} % xref for doubly defined section name
\def\As{\note{See also sections}} % xref for multiply defined section name
\def\B{\rightskip=0pt plus 100pt minus 10pt % go into C mode
  \sfcode`;=3000
  \pretolerance 10000
  \hyphenpenalty 1000 % so strings can be broken (discretionary \ is inserted)
  \exhyphenpenalty 10000
  \global\ind=2 \1\ \unskip}
\def\C#1{\5\5\quad$/\ast\,${\cmntfont #1}$\,\ast/$}
\let\SHC\C % "// short comments" treated like "/* ordinary comments */"
%\def\C#1{\5\5\quad$\triangleright\,${\cmntfont#1}$\,\triangleleft$}
%\def\SHC#1{\5\5\quad$\diamond\,${\cmntfont#1}}
\def\D{\defin{\#define}} % macro definition
\let\E=\equiv % equivalence sign
\def\ET{ and~} % conjunction between two section numbers
\def\ETs{, and~} % conjunction between the last two of several section numbers
\def\F{\defin{format}} % format definition
\let\G=\ge % greater than or equal sign
% \H is long Hungarian umlaut accent
\let\I=\ne % unequal sign
\def\J{\.{@\&}} % TANGLE's join operation
\let\K== % assignment operator
```

```
%\let\K=\leftarrow % "honest" alternative to standard assignment operator
% \L is Polish letter suppressed-L
\outer\def\M#1{\MN{#1}\ifon\vfil\penalty-100\vfilneg % beginning of section
  \vskip\intersecskip\startsection\ignorespaces}
\outer\def\N#1#2#3.{\gdepth=#1\gtitle={#3}\MN{#2}% beginning of starred section
  \ifon\ifnum#1<\secpagedepth \vfil\eject % force page break if depth is small
    \else\vfil\penalty-100\vfilneg\vskip\intersecskip\fi\fi
  \message{*\secno} % progress report
  \edef\next{\write\cont{\ZZ{#3}{#1}{\secno}% write to contents file
                 {\noexpand\the\pageno}}}\next % \ZZ{title}{depth}{sec}{page}
  \ifon\startsection{\bf#3.\quad}\ignorespaces}
\def\MN#1{\par % common code for \M, \N
  {\xdef\secstar{#1}\let\*=\empty\xdef\secno{#1}}% remove \* from section name
  \ifx\secno\secstar \onmaybe \else\ontrue \fi
  \mark{{{\tensy x}\secno}{\the\gdepth}{\the\gtitle}}}
% each \mark is {section reference or null}{depth plus 1}{group title}
% \O is Scandinavian letter O-with-slash
% \P is paragraph sign
\def\Q{\note{This code is cited in section}} % xref for mention of a section
\def\Qs{\note{This code is cited in sections}} % xref for mentions of a section
\let\R=\lnot % logical not
% \S is section sign
\def\T#1{\leavevmode % octal, hex or decimal constant
  \hbox{$\def\?{\kern.2em}%
    \def\$##1{\egroup_{\,\rm##1}\bgroup}% suffix to constant
    \def\_{\cdot 10^{\aftergroup}}% power of ten (via dirty trick)
    \let\~=\oct \let\^=\hex {#1}$}}
\def\U{\note{This code is used in section}} % xref for use of a section
\def\Us{\note{This code is used in sections}} % xref for uses of a section
\let\V=\lor % logical or
\let\W=\land % logical and
\def\X#1:#2\X{\ifmmode\gdef\XX{\null$\null}\else\gdef\XX{}\fi % section name
  \XX$\langle\,${#2\eightrm\kern.5em#1}$\,\rangle$\XX}
\def\Y{\par\yskip}
\let\Z=\le
\let\ZZ=\let % now you can \write the control sequence \ZZ
\let\*=*

%\def\oct{\hbox{\rm\char'23\kern-.2em\it\aftergroup\?\aftergroup}} % WEB style
%\def\hex{\hbox{\rm\char"7D\tt\aftergroup}} % WEB style
\def\oct{\hbox{$^\circ$\kern-.1em\it\aftergroup\?\aftergroup}}% CWEB style
\def\hex{\hbox{$^{\scriptscriptstyle\#}$\tt\aftergroup}} % CWEB style
\def\vb#1{\leavevmode\hbox{\kern2pt\vrule\vtop{\vbox{\hrule
        \hbox{\strut\kern2pt\.{#1}\kern2pt}}
      \hrule}\vrule\kern2pt}} % verbatim string

\def\onmaybe{\let\ifon=\maybe} \let\maybe=\iftrue
\newif\ifon \newif\iftitle \newif\ifpagesaved

\def\lheader{\mainfont\the\pageno\eightrm\qquad\grouptitle\hfill\title\qquad
  \mainfont\topsecno} % top line on left-hand pages
\def\rheader{\mainfont\topsecno\eightrm\qquad\title\hfill\grouptitle
  \qquad\mainfont\the\pageno} % top line on right-hand pages
\def\grouptitle{\let\i=I\let\j=J\uppercase\expandafter{\expandafter
                        \takethree\topmark}}
\def\topsecno{\expandafter\takeone\topmark}
\def\takeone#1#2#3{#1}
\def\taketwo#1#2#3{#2}
\def\takethree#1#2#3{#3}
\def\nullsec{\eightrm\kern-2em} % the \kern-2em cancels \qquad in headers

\let\page=\pagebody \raggedbottom
% \def\page{\box255 }\normalbottom % faster, but loses plain TeX footnotes
\def\normaloutput#1#2#3{\ifodd\pageno\hoffset=\pageshift\fi
 \shipout\vbox{
  \vbox to\fullpageheight{
  \iftitle\global\titlefalse
  \else\hbox to\pagewidth{\vbox to10pt{}\ifodd\pageno #3\else#2\fi}\fi
  \vfill#1}} % parameter #1 is the page itself
  \global\advance\pageno by1}
```

```
\gtitle={\.{CWEB} output} % this running head is reset by starred sections
\mark{\noexpand\nullsec0{\the\gtitle}}
\def\title{\expandafter\uppercase\expandafter{\jobname}}
\def\topofcontents{\centerline{\titlefont\title}\vskip.7in
  \vfill} % this material will start the table of contents page
\def\botofcontents{\vfill
  \centerline{\covernote}} % this material will end the table of contents page
\def\covernote{}
\def\contentspagenumber{0} % default page number for table of contents
\newdimen\pagewidth \pagewidth=6.5in % the width of each page
\newdimen\pageheight \pageheight=8.7in % the height of each page
\newdimen\fullpageheight \fullpageheight=9in % page height including headlines
\newdimen\pageshift \pageshift=0in % shift righthand pages wrt lefthand ones
\def\magnify#1{\mag=#1\pagewidth=6.5truein\pageheight=8.7truein
  \fullpageheight=9truein\setpage}
\def\setpage{\hsize\pagewidth\vsize\pageheight} % use after changing page size
\def\contentsfile{\jobname.toc} % file that gets table of contents info
\def\readcontents{\input \contentsfile}
\def\readindex{\input \jobname.idx}
\def\readsections{\input \jobname.scn}

\newwrite\cont
\output{\setbox0=\page % the first page is garbage
  \openout\cont=\contentsfile
       \write\cont{\catcode `\noexpand\@=11\relax}   % \makeatletter
  \global\output{\normaloutput\page\lheader\rheader}}
\setpage
\vbox to \vsize{} % the first \topmark won't be null

\def\ch{\note{The following sections were changed by the change file:}
  \let\*=\relax}
\newbox\sbox % saved box preceding the index
\newbox\lbox % lefthand column in the index
\def\inx{\par\vskip6pt plus 1fil % we are beginning the index
  \def\page{\box255 } \normalbottom
  \write\cont{} % ensure that the contents file isn't empty
       \write\cont{\catcode `\noexpand\@=12\relax}   % \makeatother
  \closeout\cont % the contents information has been fully gathered
  \output{\ifpagesaved\normaloutput{\box\sbox}\lheader\rheader\fi
    \global\setbox\sbox=\page \global\pagesavedtrue}
  \pagesavedfalse \eject % eject the page-so-far and predecessors
  \setbox\sbox\vbox{\unvbox\sbox} % take it out of its box
  \vsize=\pageheight \advance\vsize by -\ht\sbox % the remaining height
  \hsize=.5\pagewidth \advance\hsize by -10pt
    % column width for the index (20pt between cols)
  \parfillskip 0pt plus .6\hsize % try to avoid almost empty lines
  \def\lr{L} % this tells whether the left or right column is next
  \output{\if L\lr\global\setbox\lbox=\page \gdef\lr{R}
    \else\normaloutput{\vbox to\pageheight{\box\sbox\vss
        \hbox to\pagewidth{\box\lbox\hfil\page}}}\lheader\rheader
    \global\vsize\pageheight\gdef\lr{L}\global\pagesavedfalse\fi}
  \message{Index:}
  \parskip 0pt plus .5pt
  \outer\def\I##1, {\par\hangindent2em\noindent##1:\kern1em} % index entry
  \def\[##1]{$\underline{##1}$} % underlined index item
  \rm \rightskip0pt plus 2.5em \tolerance 10000 \let\*=\lapstar
  \hyphenpenalty 10000 \parindent0pt
  \readindex}
\def\fin{\par\vfill\eject % this is done when we are ending the index
  \ifpagesaved\null\vfill\eject\fi % output a null index column
  \if L\lr\else\null\vfill\eject\fi % finish the current page
  \parfillskip 0pt plus 1fil
  \def\grouptitle{NAMES OF THE SECTIONS}
  \let\topsecno=\nullsec
  \message{Section names:}
  \output={\normaloutput\page\lheader\rheader}
  \setpage
  \def\note##1##2.{\quad{\eightrm##1~##2.}}
  \def\Q{\note{Cited in section}} % crossref for mention of a section
  \def\Qs{\note{Cited in sections}} % crossref for mentions of a section
  \def\U{\note{Used in section}} % crossref for use of a section
```

```
  \def\Us{\note{Used in sections}} % crossref for uses of a section
  \def\I{\par\hangindent 2em}\let\*=*
  \readsections}
\def\con{\par\vfill\eject % finish the section names
% \ifodd\pageno\else\titletrue\null\vfill\eject\fi % for duplex printers
  \rightskip 0pt \hyphenpenalty 50 \tolerance 200
  \setpage \output={\normaloutput\page\lheader\rheader}
  \titletrue % prepare to output the table of contents
  \pageno=\contentspagenumber
  \def\grouptitle{TABLE OF CONTENTS}
  \message{Table of contents:}
  \topofcontents
  \line{\hfil Section\hbox to3em{\hss Page}}
  \let\ZZ=\contentsline
  \readcontents\relax % read the contents info
  \botofcontents \end} % print the contents page(s) and terminate
\def\contentsline#1#2#3#4{\ifnum#2=0 \smallbreak\fi
    \line{\consetup{#2}#1
      \rm\leaders\hbox to .5em{.\hfil}\hfil\ #3\hbox to3em{\hss#4}}}
\def\consetup#1{\ifcase#1 \bf % depth -1 (@**)
  \or % depth 0 (@*)
  \or \hskip2em % depth 1 (@*1)
  \or \hskip4em % depth 2 (@*2)
  \or \hskip6em % depth 3 (@*3)
  \or \hskip8em % depth 4 (@*4)
  \or \hskip10em % depth 5 (@*5)
  \else \hskip12em \fi} % depth 6 or more
\def\noinx{\let\inx=\end} % no indexes or table of contents
\def\nosecs{\let\FIN=\fin \def\fin{\let\parfillskip=\end \FIN}}
    % no index of section names or table of contents
\def\nocon{\let\con=\end} % no table of contents
\def\today{\ifcase\month\or
  January\or February\or March\or April\or May\or June\or
  July\or August\or September\or October\or November\or December\fi
  \space\number\day, \number\year}
\newcount\twodigits
\def\hours{\twodigits=\time \divide\twodigits by 60 \printtwodigits
  \multiply\twodigits by-60 \advance\twodigits by\time :\printtwodigits}
\def\gobbleone1{}
\def\printtwodigits{\advance\twodigits100
  \expandafter\gobbleone\number\twodigits
  \advance\twodigits-100 }
\def\TeX{{\ifmmode\it\fi
   \leavevmode\hbox{T\kern-.1667em\lower.424ex\hbox{E}\hskip-.125em X}}}
\def\,{\relax\ifmmode\mskip\thinmuskip\else\thinspace\fi}
\def\datethis{\def\startsection{\leftline{\sc\today\ at \hours}\bigskip
  \let\startsection=\stsec\stsec}}
  % say `\datethis' in limbo, to get your listing timestamped before section 1
\def\datecontentspage{%
  \def\topofcontents{\leftline{\sc\today\ at \hours}\bigskip
   \centerline{\titlefont\title}\vfill}} % timestamps the contents page
```

Приложение C: Как пользоваться макросами CWEB

Макроопределения, содержащиеся в `cwebmac`, обеспечивают получение вывода в различном формате без корректировки результата работы `CWEAVE`, и цель этого приложения — разъяснить некоторые из возможностей.

1. В дополнение к стандартным шрифтам формата `PLAIN` вводятся четыре новых: можно ввести '`{\mc UNIX}`' и получить UNIX, выведенное прописными буквами среднего размера (medium-size caps); также можно задать '`{\sc STUFF}`' для набора STUFF шрифтом "малые прописные" (small caps); помимо этого можно выбрать шрифты увеличенного размера `\titlefont` и `\ttitlefont` для заголовков документа, где `\ttitlefont` определяет шрифт типа "пишущая машинка". Макроопределения `\UNIX/` и `\CEE/` обозначают UNIX и C, набранные средними прописными буквами.

2. Когда в тексте TEX упоминается идентификатор, обычно это выглядит как '`|identifier|`'. Но также возможна конструкция '`\\{identifier}`'. Результат будет выглядеть одинаково, но второй вариант не помещает *identifier* в индекс, поскольку не будет обрабатываться `CWEAVE` как конструкция C. Кроме того, во втором случае придется поместить '\' перед каждым символом подчеркивания, входящим в идентификатор.

3. Для получения стиля "пишущая машинка", вроде '`CWEB`', можно воспользоваться макросом '`\.`' (например, '`\.{CWEB}`'). Задавая аргумент этого макроса, следует вставить '\' перед символами, которые перечислены в пункте 'special string characters' индекса `CWEAVE`, т.е., перед символом '\', знаком доллара и пр. Появление в тексте '\␣' приведет к вставке символа "видимого пробела"; чтобы получить обыкновенный пробел после командной последовательности, используйте '`{␣}`'. Если строка слишком длинная, она может быть разбита на части с помощью '`\)`'; если TEX будет вынужден перенести строку, он ее разделит на этом символе, используя '\' как символ переноса.

4. Три командных слова: `\pagewidth`, `\pageheight`, и `\fullpageheight` могут быть переопределены в секции-преамбуле, начинающей `CWEB`-файл, для задания нужных размеров страницы. Значения по умолчанию

```
\pagewidth=6.5in
\pageheight=8.7in
\fullpageheight=9in
```

использованы при подготовке этого руководства; `\fullpageheight` получается сложением `\pageheight` с размером дополнительного заголовка и номера наверху каждой страницы. После изменения хотя бы одного из этих параметров следует немедленно вызвать `\setpage`.

5. Макрос `\pageshift` определяет величину, на которую правосторонние страницы (т.е., с нечетными номерами) будут сдвинуты по отношению к левосторонним (четные номера). Изменяя эту величину, можно получить двусторонний вывод, в котором номера страниц будут располагаться на противоположных сторонах листа.

6. Содержимое `\title` печатается вверху каждой страницы малыми прописными буквами; по умолчанию это `\jobname`.

7. Обычно первой странице присваивается номер 1. Чтобы начать со страницы 16, поместив оглавление на странице 15, используйте '`\def\contentspagenumber{15} \pageno=\contentspagenumber \advance\pageno by 1`'.

8. Командное слово `\iftitle` будет подавлять вывод колонтитула, если задано '`\titletrue`'. Обычное значение — `\titlefalse`, за исключением оглавления, так что страницы оглавления как правило не нумеруются.

Для придания гибкости процессу генерации оглавления служат следующие макросы: `\topofcontents` вызывается непосредственно перед считыванием информации оглавления, и `\botofcontents` — после этого. Типичное определение:

```
\def\topofcontents{\null\vfill
  \titlefalse % include headline on the contents page
  \def\rheader{\mainfont The {\tt CWEAVE} processor\hfil}
  \centerline{\titlefont The {\ttitlefont CWEAVE} processor}
  \vskip 15pt \centerline{(Version 3.0)} \vfill}
```

Переопределить `\rheader`, являющееся колонтитулом правосторонних страниц, будет в этом случае достаточно, чтобы разместить нужную информацию в верхней части страницы оглавления.

9. Данные оглавления выводятся в файл, который считывается после обработки индекса; каждой заглавной секции соответствует одна строка. Файл `common.toc` может выглядеть так:

```
\ZZ {Introduction}{0}{1}{23}
\ZZ {The character set}{2}{5}{24}
```

и т.д. Макрос `\topofcontents` может переопределить `\ZZ`, так что информация может быть выведена в любом нужном формате.

10. Иногда бывает желательно либо необходимо разделить вывод CWEAVE на подфайлы, которые могут обрабатываться раздельно. Например, листинг TEX'а превышает 500 страниц, чего вполне достаточно для исчерпания возможностей многих печатающих устройств и/или программ. Когда очень большой документ не разделен на более мелкие части, попытка обработать его целиком может быть сведена на нет единственной ошибкой, после которой придется начать все сначала.

Ниже описан безопасный способ разделить выходной файл CWEAVE на три части: Пусть части называются α, β, и γ, где каждая часть начинается заглавной секцией. Все макросы должны определяться в преамбуле α, а копии этих предписаний TEX'у должны быть помещены в начало β и γ. Чтобы обрабатывать части раздельно, мы должны позаботиться о двух вещах: номера начальных страниц β и γ должны быть скорректированы, и данные оглавления всех трех частей должны быть объединены.

Файл `cwebmac` содержит два командных слова: `\contentsfile` и `\readcontents`, которые облегчают выполнение необходимых действий. Нам требуется включить '`\def\contentsfile{cont1}`' в преамбулу α, а '`\def\contentsfile{cont2}`' — в преамбулу β; это заставит TEX вывести данные оглавлений α и β в файлы `cont1.tex` и `cont2.tex`. Теперь в γ мы указываем

```
\def\readcontents{\input cont1 \input cont2 \input \contentsfile},
```

собирая все данные вместе в нужном порядке.

Тем не менее, по-прежнему своего решения требует проблема нумерации страниц. Одним из ее путей ее решения будет включить в преамбулу β следующее:

```
\message{Please type the last page number of part 1: }
\read -1 to \temp \pageno=\temp \advance\pageno by 1
```

Далее необходимая информация просто вводится по запросу TEX'а; аналогичная конструкция помещается в начале γ.

Этот метод, безусловно, может использоваться для разбиения выходного файла на произвольное число фрагментов.

11. Временами нужно включать в индекс элементы, оформленные особым образом. Например, нам может понадобиться иметь элемент индекса для 'TEX'. CWEAVE предоставляет два простых способа задать явно элемент индекса (помимо имен и ключевых слов): '`@^`' дает романский стиль, а '`@.`' — стиль "пишущая машинка". Но если мы попытаемся вывести 'TEX' в романском стиле, задав '`@^\TeX@>`', нам помешает символ '\', и этот элемент индекса не появится вместе с другими, начинающимися на T.

Решением будет использовать конструкцию '`@:`', определив макрос, который просто удаляет ключ:

```
\def\9#1{}
```

Вы можете задать, к примеру, '`@:TeX}{\TeX@>`' в своем CWEB-файле; CWEAVE поместит это в индекс в алфавитном порядке, соответственно ключу, и сгенерирует макровызов '`\9{TeX}{\TeX}`', который гарантирует, что ключ не будет напечатан.

Аналогично в имена секций может помещаться скрытый материал, так что они могут быть сортированы в любом необходимом порядке. Некоторые люди называют подобные трюки "special refinements"; другие зовут их "kludges".

12. Командному слову `\secno` присваивается номер текущей обрабатываемой секции.

13. Если вы хотите вывести на печать только секции, которые были изменены (с помощью файла корректив) и оглавление, поместите команду '`\let\maybe=\iffalse`' в преамбулу, перед первой секцией файла CWEB. Общепринятой практикой является сделать ее первой коррективой.

14. Для печати на других языках (не по-английски), переопределите макросы `\A`, `\ET`, `\Q`, `\U`, `\ch`, `\fin`, `\con`, `\today`, `\datethis`, и `\datecontentspage`. Собственно `CWEAVE` изменять не требуется.

15. Вывод отдельных частей документа может быть избирательно запрещен с помощью `\noatl`, `\noinx`, `\nosecs`, `\nocon`.

16. Все диакритические знаки и другие специальные текстовые символы формата plain TEX будут работать в документах `CWEB` в точности так, как это описано в главе 9 *The TEXbook*, с одним исключением. Акцент "точка" (обычно `\.`) должен вводиться как `\:`.

17. "Закомментированные" строки `cwebmac.tex` предлагают некоторые дополнительные возможности, которые могут пригодиться пользователям. Например, одна из них вставляет пустую страницу, если у вас дуплексный принтер. Приложения D, E, и F в полной версии этого руководства напечатаны с использованием закомментированной опции, которая заменяет '$\leftarrow$' на '$=$' в листингах программ. Вы можете просмотреть эти приложения, чтобы решить, какой формат вам больше нравится.