The CWEB System of Structured Documentation

(Version 3.64 — February 2002)

Donald E. Knuth and Silvio Levy

TEX is a trademark of the American Mathematical Society.
Acrobat Reader is a trademark of Adobe Systems Incorporated.

The printed form of this manual is copyright © 1994 by Addison-Wesley Publishing Company, Inc. All rights reserved.

The electronic form is copyright © 1987, 1990, 1993, 2000 by Silvio Levy and Donald E. Knuth.

Permission is granted to make and distribute verbatim copies of the electronic form of this document provided that the electronic copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of the electronic form of this document under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Individuals may make copies of the documentation from the electronic files for their own personal use.

Internet page `http://www-cs-faculty.stanford.edu/~knuth/cweb.html` contains current info about CWEB and related topics.

구조적 문서화를 위한 CWEB 시스템

(버전 3.64 — 2002년 2월)

도널드 E. 커누스, 실비오 레비

TEX은 American Mathematical Society의 등록상표 입니다.
Acrobat Reader는 Adobe Systems Incorporated의 등록상표 입니다.

저작권법에 의해, 인쇄 형식의 이 문서의 저작권은 Addison-Wesley Publishing Company에 있습니다. © 1994
전자 형식의 이 문서의 저작권은 실비오 레비와 도널드 E. 커누스에게 있습니다. © 1987, 1990, 1993, 2000

전자 형식의 이 문서는 자유롭게 복사하거나 배포할 수 있으나, 반드시 출처를 밝혀야 합니다.
전자 형식의 이 문서는 자유롭게 수정해서 복사하거나 배포할 수 있으나, 반드시 출처를 밝혀야 합니다.
개개인은 자신의 개인적인 용도로 전자 형식의 이 문서를 프린터로 출력하여 이용할 수 있습니다.
CWEB과 그와 관련된 토픽들의 최신 정보는 웹 페이지 http://www-cs-faculty.stanford.edu/~knuth/cweb.html 에서 찾을 수 있습니다.

이 문서의 한글 번역은 남수진이 하였습니다.
본 저작물은 한겨레결체로 만들어졌습니다.

구조적 문서화를 위한 CWEB 시스템

도널드 커누스, 실비오 레비

이 글은 도널드 커누스의 WEB 시스템에 C 프로그래밍 언어를 적용한 실비오 레비의 CWEB 시스템에 관한 문서이다. CWEB 시스템이 만들어진 1987년 이래로, 커누스와 레비는 다양한 방법으로 시스템을 수정하고 보완하여서 현재의 시스템은 거의 완벽한 시스템이 되었다고 볼 수 있지만, 여전히 버그리포트, 개선사항, 또는 이 시스템에 도움이 될만한 사항들이 있다면, 주저하지말고 레비(levy@math.berkeley.edu)에게 연락하길 바란다.

CWEB 시스템과 WEB 시스템은 같은 철학을 공유하고 (근본적으로는) 동일한 문법을 사용하고 있기 때문에 커누스의 "구조적 문서화를 위한 WEB 시스템" 이란 문서에 익숙한 독자는 이 문서를 빠르게 읽어 나갈 수 있을 것이다. 어떤 면에서 보면 CWEB 시스템은 WEB 시스템에서 필요로 했던 매크로 정의나 문자열 처리 같은 기능이 필요 없어진 단순화된 버전이기도 하다. 왜냐하면 파스칼 프로그래 언어를 기반으로 했던 WEB은 파스칼에는 없는 전처리 기능이 필요했던 반면, C언어를 기반으로 하는 CWEB 시스템은 C언어 자체가 전처리 기능을 가지고 있어서 그러한 기능을 가질 필요가 없기 때문이다. 더구나, WEB 시스템은 8진수와 16진수를 표시하기 위해서 @'77과 @"3f와 같은 별도의 표기 방식이 필요했지만, CWEB에서는 그러한 숫자를 나타내기 위한 별도의 표기 방식의 필요 없이 C언어에서 사용하는 그대로 077이나 0x3f처럼 쓰면 된다. CWEB은 WEB의 다른 모든 기능들을 가지고 있으면서 새로운 기능이 더해진 시스템이라고 할 수 있다.

CWEB 시스템 개발에 있어서 많은 제안과 비평을 해준 모든 분들께 감사의 말을 전한다. 특히, 코드 작성에 기여한 Steve Avery, Nelson Beebe, Hans-Hermann Bode, Klaus Guntermann, Norman Ramsey, Joachim Schnitter, Saroj Mahapatra, 그리고 이 문서 작성에 많은 도움을 준 Cameron Smith에게 감사의 말을 전한다. Ramsey는 그의 SPIDER 시스템 [*Communications of the ACM* **32** (1989), 1051–1055]에서, 파스칼이나 C언어 이외의 다른 프로그래밍 언어를 이용하는 문학적 프로그래밍을 가능하게 해주었다. 커누스의 책인 *Literate Programming* (1992)은 문학적 프로그래밍과 관련된 많은 참고 문헌들과 초창기 작업에 대한 풍부하고 흥미로운 내용을 담고 있다. Bode, Schnitter, Mahapatra는 CWEB에서 C언어 뿐만아니라, C++도 사용할 수 있도록 하여서, 독자에 따라서 이 문서를 C에 대한 글이 아닌 C++에 대한 글로 읽어도 된다.

서 론

CWEB의 배경이 되는 철학은, 자신이 작성하는 프로그램과 더불어 가능한 최상의 문서를 제공하고자 하는 프로그래머는 문서 작성을 위한 TEX과 같은 언어와 프로그래밍을 위한 C와 같은 프로그래밍 언어 두가지가 동시에 필요하다는 것이다. 두가지 언어 중 어느 한 가지만 가지고는 최상의 문서를 제공할 수 없다. 하지만 이 두가지 언어를 적절히 조합하면, 각각의 언어가 발휘할 수 있는 기능보다 더욱 유용한 기능을 갖춘 시스템을 구성할 수 있다.

소프트웨어 프로그램의 구조는 내부적으로 많은 작은 부분들이 서로 긴밀히 연결되어 있는 "웹(web)"구조라고 생각할 수 있다. 그러한 구조의 프로그램을 문서화하는 좋은 방법은 웹을 구성하는 각각의 개별적인 부분들을 설명하고, 그 부분들이 이웃들과 어떠한 연관이 있는지를 설명하는 것일 것이다. TEX이라는 조판 언어를 이용해서 각 부분을 읽기 좋게 만들어서 웹의 구조를 쉽게 파악 할 수 있고, C 프로그래밍 언어를 이용해서 프로그램에 사용된 알고리즘들을 수학적으로 명확하게 기술할 수 있다. 이 두가지 언어를 잘 조합해서 소프트웨어의 복잡한 구조를 쉽게 파악 할 수 있도록하는 프로그래밍이 가능함과 동시에 그 문서화된 프로그램은 기계적으로 소프트웨어 시스템으로 변환되어 실행 가능한 시스템을 만들 수 있다.

CWEB 시스템은 CWEAVE와 CTANGLE이라고 불리는 두개의 프로그램으로 구성되어 있다. CWEB 프로그래밍을 할때는, CWEB 파일 이라고 불리는 하나의 파일에 C 코드와 문서화를 위한 TEX 코드를 동시에 작성하고 그 파일의 이름은 일반적으로 something.w과 같이 확장자를 .w로 한다. 명령어 'cweave something'를 통해서 something.tex 파일을 생성하고, 그 파일은 TEX으로 컴파일하여 페이지 레이아웃과 프로그램의 들여쓰기, 그리고 예약어나 변수명 혹은 수학 기호들을 각각 다른 폰트로 표현하는등 우리들이 쉽게 읽을 수 있도록 멋진 출력물로 바뀐다. 이 출력물에는 cweave 명령어가 자동적으로 만든 색인 리스트도 들어있다. 비슷한 방법으로, 'ctangle something' 이란 명령을 실행하면 something.c란 이름의 C 파일을 얻을 수 있고, 이 파일을 컴파일하여 실행 파일을 얻을 수 있다.

CWEB은 C 프로그래밍에 단순히 문서화 도구를 제공한다는 측면 보다는 C 언어를 보다 발전시키는 측면이 더 크다. 보통의 C 프로그래밍에서는 불가능한, 하나의 프로그램을 여러 작은 조각들로 나누고 그 조각들을 사람들이 생각하고 읽기 편하도록

새롭게 배치하는 것이 가능하다. 이와같이 하면, 프로그램을 컴파일러가 이해할 수 있는 방식대로의 순차적으로 설명하는 것이 아니라, 알고리즘 혹은 자료 구조적으로 의미적으로 관련있는 부분들을 한 곳에 배치해서 그 부분을 집중적으로 보다 자세히 설명할 수 있다. 그리고나서 그러한 부분들이 다른 부분들과 서로 어떠한 연관이 있는지에 대한 전체적인 그림을 그리면 프로그램을 보다 쉽게 이해할 수 있다. CTANGLE이라는 프로그램은 인간의 정서에는 맞지 않는, 인간이 보기에 다소 복잡해 보이는 C 프로그램을 인간의 기준에서 잘 정돈된(untangle) 순서로 작성했다가 이를 다시 컴파일러에게는 맞지만 인간에게는 순서가 엉망인 C 프로그램을 만든다고 하여 ctangle 이라는 이름을 얻게 되었다. CWEB 프로그래밍을 하여 얻을 수 있는 장점은 프로그램을 사람의 기준에서 사람이 읽기 좋은 형태로 작성할 수 있다는 것이다. CWEAVE라는 프로그램은 CWEB 파일을 입력 받아서 각 부분을 구성하는 TEX 코드와 C 코드를 보기 좋게 구성하여 뜨개질이나 직물을 짜내듯이 하나의 멋진 구조화된 문서를 만들어 내기 때문에 cweave라는 이름을 얻었다. 영어 단어 "weave"에 해당하는 독일 단어가 "webe"이고, 그에 해당하는 라틴어가 "texe"라는 것에는 아마도 일종의 깊은 연관이 있을 것이다.

CWEB을 하기 위해서는 반드시 C 프로그래밍 언어를 잘 알아야 한다. 그리고 최소한의 TEX에 대한 지식도 있으면 좋겠지만, CWEB 프로그래밍을 하면서 TEX에 대한 지식은 자동적으로 얻게 될 것이므로 TEX에 대한 지식은 필요 없는데, 이는 TEX에서 사용되는 단순한 텍스트는 TEX에 대한 지식이 거의 없어도 가능하기 때문이다. 물론 C언어와 TEX에 모두 익숙한 독자는 CWEB을 매우 쉽게 배울 수 있다.

개 요

CWEB 파일에는 TEX 텍스트와 C 텍스트라는 두 종류의 텍스트가 있다. CWEB 프로그래머는 만들고자하는 문서와 C 프로그램을 동시에 생각해야 한다. 즉, 프로그래머는 본능적으로 CWEAVE와 CTANGLE 프로그램이 CWEB 파일을 어떻게 처리할지를 미리 알고 있어야 한다. CWEAVE는 TEX 파일을 만들 때, CWEB 파일 내의 TEX 텍스트를 그대로 TEX 파일에 복사하지만, CTANGLE은 C 파일을 만들 때, TEX 텍스트를 모두 제거하여 전혀 포함시키지 않는다. TEX 텍스트는 순전히 문서화를 위한 텍스트로써, CTNALGE은 이를 완전히 무시하는 반면에, CWEAVE는 C 텍스트를 보기좋게 문서화 하고, CTNALGE은 그것을 본래 C 문법에 맞게 재배치한다. 재배치 규칙은 나중에 자세히 설명하게 될 것이다. 지금 이 순간, 기억해야 할 것은 두 종류의 텍스트가 있다는 것이다. CWEB 프로그램을 작성한다는 것은 보통의 TEX 문서를 만든다는 것과 동일하고, 단지 TEX의 기본적인 모드인 수평 모드, 수직 모두, 수학 모드에 "C 모드"가 더해진 것이라고 생각하면 된다.

CWEB 파일은 마디(*section*)라고 하는 하나의 온전한 의미를 가진 단위들로 구성된다. 각 마디는 다음과 같은 세개의 부(部, part)를 가지고 있다.

- TEX부, 마디가 어떤 일을 하는지에 대한 설명을 담고 있다.
- 중간부, C언어에서 제공하는 매크로 정의 부분을 담고 있다. 이 부분은 CTNAGLE에 의해서 전처리기 매크로 정의로 변환된다.
- C언어부, CTANGLE이 만들어내는 C 프로그램 코드이다. C 코드는 하나의 단위로 쉽게 이해될 수 있을 정도의 십여줄 내외의 프로그램 텍스트이다.

위의 세 부분은 마디에서 반드시 위의 순서대로 나열되어야 한다; 즉 TEX부가 가장 먼저 오고, 그다음에 중간부, 마지막으로 C언어부가 나타난다. 세개의 부분들 중 어느 것이나 생략할 수 있다.

마디는 '@␣' 또는 '@*'와 같은 기호로 시작된다. 여기서 '␣'는 빈칸을 나타낸다. 마디는 그다음 마디가 시작하는 곳에서 끝나거나 (즉, 다음의 '@␣' 또는 '@*') 파일의 끝에서 끝난다. 또한 CWEB 파일에는 어떠한 마디에도 포함되지 않는 텍스트가 있다. 이른바 맨 첫번째 마디 앞에 나오는 텍스트가 그것인데, 그러한 텍스트를 "in limbo"라고 부른다. CTNAGLE은 이 텍스트를 완전히 무시하지만 CWEAVE는 TEX 파일에 그대로 복사한다. CWEB 프로그램을 작성하는 것은 TEX 파일을 작성하는 것과 크게 다르지 않으므로 TEX 파일을 작성할 때와 마찬가지로 in limbo 부분에 특별한 글꼴을 불러오거나 매크로를 정의 하거나 페이지의 크기를 정한다거나 혹은 타이틀 페이지를 만드는 코드등을 담아서 파일의 맨 윗부분에 두는 것이 관례이다.

마디들은 1부터 시작하는 순차적인 번호가 붙는다. 이 번호들은 CWEAVE가 만들어 내는 TEX 문서의 각 마디의 맨 앞에 붙어서 마디의 시작을 나타낸다. 그리고 이 번호들은 CTANGLE이 만들어내는 C 프로그램에서 이 마디에 의해서 만들어진 코드의 시작부와 끝부분에 C의 주석문으로 나타나기도 한다.

마디 이름

마디 이름과 관련해서 좋은 소식과 나쁜 소식이 있다. 먼저 좋은 소식: CWEB 파일을 작성할 때, 마디 앞에 붙는 마디의 번호를 매기느라 수고할 필요가 전혀 없다는 것이다. 그저 단순히 각 마디의 시작부에 '@␣' 혹은 '@*'를 붙여주면, 마디 번호는 CWEAVE와 CTANGLE이 알아서 자동적으로 매겨준다. 마디는 번호로 인식되는 것이 아니라 이름(*name*)으로 인식된다. 마디 이름은 '@<', TEX 텍스트, '@>' 순으로 구성된다. CWEAVE가 마디 이름을 출력할 때는 '@<'와 '@>'를 꺽쇠 괄호로 바꾸고 그 안에 작은 글꼴로 마디 번호를 매겨 넣는다. 따라서 CWEAVE가 만들어낸 출력물을 읽을 때 하나의 마디에서 언급된 다른 마디를 번호를 이용해서 쉽게 찾을 수 있다.

마디가 어떤 일을 하는지 한 눈에 알아보기 위해서, 마디 이름은 그 마디의 내용을 잘 설명할 수 있는 것이어야 한다; 즉, 마디 이름은 그 이름만 보고도 이 마디가 무슨 일을 하는 마디인지 알 수 있을 정도의 일목요연한 문구이어야 한다. 마디는 다른 마디에 포함될 수 있는데, 이때 마디를 포함하는 마디에서 중요하지 않지만 상세한 설명이 필요한 부분에 그 설명에 해당하는 마디를 끼워 넣어서 마디를 보다 읽기 좋고 간결하게 할 수 있다. 따라서 마디 이름은 그 마디가 전달하고자 하는 의미를 분명히 반영할 수 있을 정도의 충분한 길이이어야 한다.

나쁜 소식: 위 설명대로 마디가 하는 일을 요약해서 마디 이름을 정하면, 마디 이름이 다소 길어질 경우가 생기는데, 마디 이름이 길어지면 나중에 혹은 다른 곳에서 그 마디를 언급할 필요가 있을 때마다 마디의 그 긴 이름을 매번 자판을 두드려 입력해야 하는데, 이것은 매우 수고로운 작업이 된다. 이런 수고를 덜기 위해서, CWEAVE와 CTANGLE은 마디 이름을 간단하게 하는 기능을 가지고있다. CWEB 파일 내의 어디에선가 정확한 마디 이름이 한번이라도 언급이 되면, 단순히 '@<α...@>'처럼 입력해 넣으면 된다. 여기서 α는 하나의 마디를 정확하게 구분해 낼 수 있는 최소 길이의 접두어(prefix)이다. 예를 들면, '@<Clear the arrays@>'라는 마디 이름은 'Clear'라는 다섯 글자로 시작되는 다른 마디 이름이 없다면 '@<Clear...@>'로 간단하게할 수 있다. 만약에 'Clear'로 시작하는 다른 마디가 있다면, '@<Clear t...@>'등과 같이 이 마디를 다른 마디 이름과 구별 할 수 있는 최소의 길이로 간략화하면 된다.

두 개의 마디 이름이 같은 것인지 비교 할때는 마디 이름을 구성하는 문자 하나하나 서로 매치해가면서 비교하게 된다. 이때 한 가지 예외가 있는데, 여러 개의 연속적인 공백 문자(스페이스, 탭, 개행문자 또는 용지먹임(form feed)들은 하나의 빈칸으로 인식되고 마디 이름의 앞과 뒤에 나오는 공백 문자들은 제거된다. 따라서 마디 이름 '@< Clear the arrays@>'는 앞서 예와 동일한 이름이다. 간략화된 이름에서 생략기호(...) 다음에 나오는 공백도 마찬가지로 무시되지만, 생략기호 앞에 나오는 공백문자는 공백으로 인식된다. 따라서 '@<Clear t ...@>'는 '@<Clear the arrays@>'와 서로 다른 마디이다.

CTANGLE이 하는 일

마디 이름이 '@␣' 또는 '@*'로 시작된다고 하였는데, 마디가 어떻게 TEX부, 중간부, C언어부의 세 부분으로 나뉘는지에 대해서는 설명하지 않았다. 마디 내에서 중간부는 '@d' 또는 '@f'가 처음 나타나는 곳부터 시작되고, C언어부는 '@c' 또는 '@<마디 이름@>='이 처음 나타나는 곳 부터 시작된다. 후자의 경우로 C언어부를 시작한다는 것은 마디 이름이 C 텍스트가 하는 일을 요약 설명한다는 것을 의미한다. C언어부가 마디 이름 대신에 '@c'로 시작하면 그 마디는 마디 이름이 없다는 (*unnamed*) 것을 뜻한다.

마디 이름을 구성하는 '@<마디 이름@>'은 다른 마디들의 C언어부에 몇 번이고 나타날 수 있다: 최초에 마디의 C언어부를 정의하기 위해 쓰인 '@<마디 이름@>'를 제외한 나머지의 쓰임은 그 마디가 "정의"된다는 의미가 아니라 "사용"된다는 의미이다. 다른 말로 하면, 이미 다른 곳에서 정의된 마디의 C 코드는 현재 '@<마디 이름@>'이 나타난 C 프로그램의 바로 그 지점에 삽입되어야 한다는 것이다. CTANGLE이 하는 주된 일은 각각의 이름을 가진 마디와 이름이 없는 마디들로 부터 C 프로그램을 만들어 낸다는 것이다. 그 방법은 다음과 같다. 먼저 '@d'로 표시되는 모든 매크로 정의들이 C 언어의 전처리기 매크로 정의들로 변환되면서 파일의 맨 처음으로 복사된다. 그리고 나서 이름 없는 마디의 C언어부들이 그 다음에 차례로 복사된다; CTANGLE은 여러번의 패스를 통하여 CWEB파일로 부터 C 프로그램을 만들어 내는데, 지금까지의 절차로 인하여 1차 패스의 결과물을 얻게된다. (CWEB 파일에는 이름 없는 마디가 적어도 하나는 있어야 한다. 그렇지 않으면 C 프로그램이 만들어지지 않는다.) 그리고 나서, 1차 패스 결과물 내의 모든 마디 이름들은 해당하는 마디의 C언어부로 치환되고, 이 치환 작업은 이름을 가진 마디가 하나도 남지 않을 때까지 계속된다. 또한 C언어부 내의 모든 주석문들은 제거된다. CTANGLE이 만들어 내는 C 프로그램은 오로지 C 컴파일러용으로 출력물을 만들기 때문이다.

만일 동일한 마디 이름이 다른 여러 마디의 이름으로 사용된다면, 그 이름을 가진 마디의 C언어부는 이 마디 이름을

사용하는 마디의 C언어부들의 합으로 이루어진다. CWEB의 이 기능은 매우 편리한 기능이다. C 언어에서 전역 변수는 대개 프로그램의 첫부분에 그 프로그램에서 사용되는 모든 전역 변수들을 한 곳에서 정의하거나 선언한다. 그런데, 모든 변수들을 한 곳에서 정의하기 보다는 그 변수가 사용되는 시점에서 그 변수에 대한 설명과 함께 정의를 하면 그 변수에 대한 의미가 확실해지고 프로그램을 읽어나가기가 더울 수월해 질 것이다. 하지만 C 언어는 이러한 기능을 제공하지 않는다. 이때, 방금 전에 설명한 CWEB의 기능을 사용할 수 있다. 예를들어 '전역 변수들' 이라는 이름을 가진 마디가 있다고하자. 이 마디가 여러 마디의 이름으로 사용된다면, 전역 변수들은 여러 마디에 걸쳐서 정의 되지만, 결국은 이 모든 전역 변수들이 처음에 정의한 '전역 변수들' 이란 이름을 가진 마디에 모이게 된다. 여러 개의 마디가 동일한 이름을 가질 때마다, CWEAVE는 그 이름의 마디가 처음 정의되는 곳에 그 마디의 번호를 할당하고, 그 마디 밑에 각주로써 이 마디가 다른 마디에서 사용된다는 것을 알려주기 위해서 'See also sections so-and-so.' 라고 써넣는다; 이 각주에는 이 마디를 C언어부로 사용하는 모든 마디들의 번호가 나타나게 된다. 마디의 C 텍스트는 CWEB 파일 내에서 등호(=)기호로 나타나는데, 이는 CWEAVE에 의해서 '≡' 기호로 바뀐다. 즉, '⟨ 마디 이름 ⟩ ≡ C 텍스트'로 바뀌게된다. 그러나 동일한 이름의 마디가 여러 번 나타날 때, 두번째부터는 현재 나타나는 마디는 다른 마디의 C 텍스트에 첨가된다는 의미로 '≡' 기호가 '+≡'로 바뀐다.

CTANGLE은 마디를 처리 할 때마다, CTANGLE의 출력물인 C 파일에 전처리기 명령어인 `#line`문을 써넣는다. 이는 C 파일을 컴파일 할때, 컴파일러가 에러 메시지를 내거나, 혹은 디버깅 할때, C 파일이 아니라 CWEB 파일의 라인 넘버가 나오므로 원래 작성한 파일을 가지고 작업을 할 수 있도록 하기 위함이다. 따라서 대개의 경우 CTANGLE이 만들어 낸 C 파일을 직접 다루는 일은 없다.

CWEAVE가 하는 일

CWEAVE가 하는 일은 CWEB 파일로 부터 `.tex` 파일을 만들어내는 것인데, 그 방법은 다음과 같다. 우선 `.tex`의 맨 첫줄은 다른 `.tex` 파일들이 그렇듯이, CWEB의 문서화를 위한 규칙들을 만들어 놓은 매크로 파일을 포함하는 것이다. 그리고 다음 줄부터 CWEB 파일의 "in limbo"에 정의된 TEX 텍스트들이 그대로 복사된다. 그 다음 부터 각 마디에 대한 출력 결과가 차례로 나온다. 이 과정에서 각 페이지에 마디들이 적절히 배치된다. 마지막으로, CWEAVE는 색인 페이지를 만는데, 이 색인 페이지에는 각각의 C 프로그램에 사용된 변수명 이라든가 여러 식별자 들이 나오고, 이 파일을 구성하는 모든 마디들이 마디 이름의 알파벳순으로 나온다. 또한 색인 페이지 뿐만 아니라 별표 마디와 그의 마디 번호를 나타내는 목차도 만든다.

"별표" 마디가 무엇인가 하고 의문을 가졌을 것이다. 앞에서 살펴보았듯이 보통의 마디는 '`@␣`'로 시작하는데, 별표 마디는 '`@*`'로 시작한다. 다른 보통의 마디와 크게 다른 점은 없다. 단순히 여러 마디들로 구성되는 그룹을 대표하는 의미를 갖는다. '`@*`' 다음에는 반드시 그룹의 제목이 나오고 그 뒤에 마침표가 나와야 한다. 이러한 별표 마디는 TEX 출력물에서 보면 반드시 새로운 페이지로 시작되고, 제목은 다음 별표 마디가 나오기 전까지 그 페이지와 그 뒤의 모든 페이지의 헤드라인에 사용된다. 그리고, 제목은 목차에도 나오고, 마디의 시작부에 굵은 글꼴로 출력된다. 예를 들어, 어떤 프로그램에서 스택을 이용하고 그 프로그램은 스택을 직접 구현해야 한다고 하자. 그러면 프로그램 어딘가에 스택을 구현하기 위한 많은 마디들이 있을 것이다. 이 때 이러한 마디들을 그룹으로 묶어서 "스택구현" 이라는 제목의 별표 마디를 만들 수 있는 것이다. 주의: 이러한 제목에 TEX의 명령어는 사용하지 말아야 한다. 만약에 사용한다면, 출력물이 원하는 모양이 나오지 않을 수도 있다. 왜냐하면 제목들이 각 페이지의 헤드라인에 사용 될 때는 제목의 각 글자들은 대문자로 바뀌고, 마디의 처음에 나올 때는 굵은 글꼴로 바뀌고, 또한 목차에도 사용되기 때문이다. 이 세가지 경우 모두 문제를 일으키지 않을 명령어라면 사용해도 된다.

각 마디에 대해서 CWEAVE가 만들어 내는 TEX 출력물은 다음과 같이 구성된다. 먼저 마디 번호가 나온다. (예를 들어, 123번 마디의 시작에 '`\M123.`'가 온다. 만약 이 마디가 별표 마디라면, '`\M`' 대신에 '`\N`'이 쓰인다.) 그리고나서 그 마디의 TEX부가 CWEB 파일에 입력해 넣은 그대로 복사되어 온다. 그 뒤로 중간부, C언어부가 그들 사이에 약간의 간격을 두고 온다. 중간부와 C언어부는 TEX부처럼 그냥 그대로 복사되어 오는 것은 아니고, 멋진 문서를 만들어 내기 위해서 매크로 파일에 정의된 대로 변환된다.

TEX 텍스트 내의 C 코드와 그의 역

TEX 텍스트를 입력하다보면, C 코드에서 사용된 변수들이나 다른 C의 요소들을 TEX 텍스트 내에서 사용할 일이 자주 생긴다. 이 경우에 그 변수들이나 요소들을 C 코드에서 사용되던 그대로의 글꼴이나 모양을 유지해야 한다. 그래서 CWEB 언어는 그러한 기능을 제공하는데, 표현하고자 하는 C 코드를 '|'로 둘러싸면 된다. 예를 들어 다음과 같은 TEX 문장을

얻고자 한다고 해보자.

변수 pa를 '**int** $*pa$'로 선언하면, 식 $pa = \&a[0]$는 pa가 a의 0번째 원소를 가리킨다.

위 문장을 얻기 위해서 CWEB 파일에 아래와 같은 TEX 텍스트를 입력해야 할것이다.

```
변수 |pa|를 '|int *pa|'로 선언하면,
식 |pa=&a[0]|는 |pa|가 |a|의 0번째 원소를 가리킨다.
```

그리고 CWEAVE는 위의 문장을 마치 암호처럼 보이는 아래와 같은 문장으로 변환한다.

```
변수 \\{pa}가 '\&{int} ${}{*}\\{pa}$'로 선언될 때,
식 $\\{pa}\K{\AND}\|a[\T{0}]$는 \\{pa}가 \|a의 0번째 원소를 가리킨다.
```

덧붙여 말하자면, 위와 같은 문장이 주어졌을때, 비록 이 마디의 C언어부에 pa가 없을지라도, CWEAVE가 생성하는 색인은 pa의 색인으로 현재 마디 번호를 포함한다. 색인은 프로그램에서 사용된 변수들 뿐만아니라, 보통의 설명을 하는 문장에 사용된 변수들까지도 모두 포함한다; 이 기능을 매우 유용한 기능이다. 그러나, **int**나 a 와 같은 것들은 색인에 포함되지 않는데, 이는 CWEAVE가 C 언어의 예약어나 한글자짜리 변수나 식별자들은 색인에 포함하지 않는다는 규칙때문이다. 그러한 변수나 식별자들은 너무나도 흔하고 널리 사용되는 것이라 그들이 사용되는 모든 마디 번호를 색인 포함하는 것은 자칫하면 초점을 흐릴 수 있다.

비록 마디가 TEX 텍스트로 시작해서 C 텍스트로 끝난다고 할지라도, C 텍스트가 '|...|'로 둘러싸여 있으면, 그 C 텍스트가 TEX 텍스트 내에 포함될 수 있기때문에, 각 텍스트를 구분하는 곳이 명확하지 않을 수 있다. 역으로, C의 주석문 (즉, /* 와 */ 사이에 있는 것들, 또는 // 다음에 나오는 것들)에 있는 모든 것들은 사실상 TEX 텍스트로 취급되기 때문에, C 텍스트 내에도 TEX 텍스트가 매우 빈번히 나타난다고 할 수 있다. 마찬가지로, 마디 이름은 TEX 텍스트이지만, @<마디 이름@>이 자체가 C 텍스트 내에서도 발견될 수 있는 것이다. 따라서 아래의 예와 같이 C와 TEX 환경을 자연스럽게 왔다갔다 할 수 있다.

```
if (x==0) @<배열 |buffer|를 초기화하라@>
```

```
|@<배열 |buffer|를 초기화하라@>|에 사용된 알고리듬을 이용해서 ...
```

위에서 첫번째 예는 어떤 마디의 C언어부에서 볼 수 있는 것인데, "배열 *buffer*를 초기화하라"라는 이름을 가진 마디에 사용된 코드가 이 부분에 삽입된다. 두번째 예는 어떤 마디의 TEX부에서 발견 할 수 있는 것인데, 그 이름의 마디가 사용이나 정의되는 것이 아니라 "인용"되는 것이다. (이 경우에 마디 이름을 감싸고 있는 '|...|'를 빠뜨리지 않도록 주의한다.)

매크로

CWEB 명령어, @d 뒤에는 다음과 같은 것들이 나오는데,

식별자 C 텍스트 또는 *식별자*(*par*$_1$, ..., *par*$_n$) C 텍스트

(위의 두번째의 경우에서 *식별자*와 괄호 사이에는 빈칸이 없다.) 이때, @d는 CTANGLE에 의해서 #define으로 시작하는 전처리기 명령어로 변환되고, 앞서 설명한대로, CTANGLE에 의해서 만들어지는 C 파일의 맨 위에 출력된다.

명령어 '@d'로 시작하는 매크로 정의는 여러 줄에 걸쳐서 나올 수가 있는데, CTANGLE이 알아서 각 줄의 맨 뒤에 백슬래쉬를 넣어주므로, C 언어에서 하듯이 각 줄의 끝을 백슬래쉬로 마감할 필요가 없다. 만일, 어떠한 이유로 해서 마디의 중간부에서 '@d' 매크로 정의를 하지 않고 C언어부에서 직접 #define을 사용해서 매크로 정의를 할 경우에, 그 정의는 CWEB의 매크로가 아니라, C 코드로 취급되므로, 그 경우는 반드시 백슬래쉬로 처리를 해주어야 한다.

문자열과 상수

C 파일 내에서 문자열을 나타내고자 하면, 보통 '나 "를 이용하게 된다. CWEB 파일에서도 마찬가지로 그렇게 하면된다. 단, 문자 '@'는 '@@'로 해야만 제대로 볼 수 있다. (사실 @@는 CWEB의 명령어인데, 문자열 내에서 사용할 수 있는 유일한 명령어이다. CWEB의 명령어에 대해서는 아래에서 자세히 살펴볼 것이다.) 문자열은 반드시 한 줄에 있어야 한다. 그렇지 않으면, 줄이 바뀌는 곳에 백슬래쉬를 넣어주어야 한다.

TEX은 문서를 만들기 위한 언어이고 C는 프로그래밍 언어이기 때문에 그 둘이 8진수와 16진수 상수를 다루는 방법에는 약간의 차이가 있다. TEX은 '와 "를 이용해서 각각 8진수와 16진수를 표시한다. C는 0과 0x를 이용한다. CWEB은 TEX과 C를 모두 포함하고 있기 때문에 각자의 영역에서 각자의 표기법을 사용할 수 있도록 해준다. 그래서 CWEB의 C 텍스트에서 40_8을 얻기 위해서 '040'라고 하면, CTANGLE은 그 그대로 C 파일에 복사를 하고, CWEAVE은 °*40*이라고 출력한다. 비슷한 방법으로, CWEAVE는 16진수 C 상수 '0x20'을 #200이라고 출력한다. CWEAVE은 8진수는 이탤릭체를 사용하고 16진수는 타자체를 사용해서 나타내기 때문에 문서에서 그 둘을 확실히 구분할 수 있다. 일관성을 위해서, 어떤 마디의 TEX부에서 8진수와 16진수 상수를 나타낼때는 '|040|' 또는 '|0x20|' 와 같이 입력해야 한다.

명령어

CWEB의 명령어(*control code*)는 첫번째 문자가 '@'인 두글자의 조합으로 이루어진다. 이미 앞에서 여러 명령어를 살펴봤는데, 이제 CWEB의 모든 명령어들을 체계적이고 자세히 알아 볼 시간이 됐다.

앞으로 설명할 명령어들에서 명령어 다음에 나오는 대괄호 안의 글자들은 그 명령어가 어떤 환경에서 사용될 수 있는가를 설명한다. L은 명령어가 in limbo에서 사용될 수 있다는 것을 뜻하고, T는 TEX부에서, M는 중간부에서, C는 C언어부와 같은 최상위 수준에서 사용될 수 있다는 것을 나타낸다. 최상위 수준이라 함은, '|...|'나 마디 이름과 같은 환경 이외의 것을 말한다. 화살표 →는 명령어를 사용함으로 해서 현재 환경을 마감하고 화살표 다음에 나오는 글자가 뜻하는 새로운 환경이 시작된다는 것을 의미한다. 따라서 명령어 @␣ 다음에 나오는 $[LTMC \to T]$가 뜻하는 바는 이 명령어는 in limbo와 마디의 TEX부, 중간부, C언어부에서 사용할 수 있고, 이 명령어를 사용하면, 새로운 마디의 TEX부가 시작된다는 의미이다.

대괄호 안에는 위에서 설명한 것들외에 두가지 다른 문자가 나올 수 있는데, 글자 r은 제한된 환경(*restricted context*)을 나타내는데, C 주석문 안의 환경, 마디 이름, C 문자열, 아래에서 설명될 명령 텍스트등이 이 환경에 속한다. 글자 c는 내부 C 환경(*inner* C *context*)을 나타내는데, 이에 속하는 환경은 '|...|'안에 있는 C 텍스트가 그것이다. 이때 '|...|'는 주석문 안에 사용된 '|...|'는 해당되지만, 다른 제한된 환경에서 나타나는 '|...|'는 제외된다. 대괄호 다음에 나오는 $*$는 현재 설명하는 명령어는 @>와 짝을 이루고 그러한 명령어들로 이루어지는 환경이 제한된(restricted) 환경이라는 뜻이다.

CWEB 명령어를 구성하는 두문자중 '@'를 제외한 나머지 한 문자가 기호가 아닌 알파벳인 명령어들은 CWEB이 대소문자를 구분하지 않기때문에 @d와 @D는 동일한 명령어가 된다. 이 글에서는 소문자만 다룰 것이다.

@@ $[LTMCrc]$ 두 개가 중복된 @는 하나의 '@'를 나타낸다. 이 명령어는 모든 환경에서 사용할 수 있는 유일한 명령어이다. 이 명령어는 주로 CWEB 파일 내에서 이메일 주소를 나타낼 때 사용된다. (예, levy@@math.berkeley.edu).

다음 명령어를 사용하면 마디의 TEX부가 시작된다.

@␣ $[LTMC \to T]$ 이 명령어를 사용하면 별표가 아닌 새로운 마디가 시작된다. 빈칸 대신에 탭, 용지먹임(form feed), 줄의 끝(end-of-line) 문자등이 @ 다음에 올 수 있고, 그것들은 빈칸과 동일한 효과를 나타낸다.

@* $[LTMC \to T]$ 이 명령어를 사용하면 새로운 별표 마디가 시작된다. 별표 마디는 앞에서 설명 했듯이, 여러 마디들로 이루어지는 그룹을 대표하는 마디이다. 이때 별표 마디는 제목을 갖는데, 그 제목은 @* 다음에 나오고 제목 다음에는 반드시 마침표가 나와야 한다. 가급적이면 이 제목에 TEX의 명령어들은 사용하지 않아야 한다. 그리고 CWEAVE와 CTANGLE은 이 명령어를 만나면, 모니터에 $*$와 현재 처리하고 있는 마디 번호를 출력해서 사용자가 모니터를 통하여 어떤 마디가 처리되고 있는지를 알 수 있다. CWEB 파일에서 가장 먼저 나오는 마디, 즉 1번 마디는 반드시 별표 마디이어야 한다.

명령어 @* 다음에 $*$ 혹은 숫자를 써서 별표 마디의 "깊이"를 지정할 수 있다. 이 깊이는 전체 프로그램의 구조에 있어서 현재 별표 마디의 상대적 순위를 나타낸다. 프로그램의 최상위 수준을 @**로 나타내면 그 마디의 제목은 목차에 굵은 글꼴로 표시되고 그 마디의 깊이는 -1이 된다. 이를 제외한 다른 마디들의 깊이는 대개 양수인데, 그 숫자는 목차에서 들여쓰기로 양을 나타낸다. 들여쓰기가 많이 되면 될 수록 그 마디는 하위 마디라는 뜻이다. 이렇게 마디에 깊이를 두어 목차에서 들여쓰기를 하면, 매우 긴 프로그램의 구조를 보다 명확하게 파악할 수 있다. 프로그램이 짧을 때는 모든 깊이를 0으로 하면 된다. 별표 마디는 언제나 새로운 페이지로 나타나는데, 별표 마디의 깊이를 일부러 1 부다 큰 수로 하면 새로운 페이지가 아닌 현재 진행중인 페이지에 나타난다.

각 마디의 중간부에는 여러개의 매크로 정의, 포멧 정의들이 순서 없이 올 수 있다. 매크로 정의는 @d로 시작하고 포멧 정의는 @f 또는 @s로 시작한다.

`@d` $[TM \to M]$ 매크로 정의는 `@d`로 시작하고 그 다음에 식별자가 오고 그 다음에 파라미터는 옵션이고 마지막으로 C 텍스트가 나온다.

`@f` $[TM \to M]$ 포멧 정의는 `@f`로 시작한다. 이 명령어는 식별자가 C 텍스트에서 사용될 때 특별한 모양으로 출력되도록 CWEAVE를 조종하는 일을 한다. 포멧 정의의 일반적인 형태는 '`@f` l r' 이고, 뒤이어 `/*` 와 `*/`의 주석문이 올 수 있다. 여기서 l 과 r 은 식별자이다. CWEAVE가 위와 같은 명령어를 만나면, l을 r 다루듯이 한다. 이 명령어는 CWEB 프로그래머가 자신만의 데이타 타입을 만들었거나 기존의 C 예약어들의 그 기능을 멈추게 하는 등 유용하게 사용될 수 있다. CWEB은 '**int**' 처럼 C의 예약어를 굵을 글꼴로 표시한다. 따라서 프로그래머는 자신이 만들어낸 데이타 타입도 C의 예약어 처럼 취급하라고 CWEAVE에게 알려줄 때, 바로 이 명령어를 사용할 수 있다. 또 'error'나 'line' 같은 단어는 C의 전처리기에서 특별한 의미를 가지고 있으므로, CWEAVE는 그 단어들을 C의 예약어와 동일한 방식으로 다룰텐데, *error*나 *line*을 변수로 사용하고 싶으면, 아래와 같은 방식으로 CWEAVE에게 알려줘야 한다.

```
@f error normal        @f line normal
```

만약에 r이 특별한 식별자 '*TeX*'이라면, 식별자 l 은 TEX의 명령어 처럼 취급된다. 예를 들면, CWEB 파일 내에 '`@f foo TeX`'가 쓰이면, 식별자 *foo*는 CWEAVE에 의해서 TEX 파일에 `\foo`로 출력된다. 따라서 프로그래머는 `\foo`가 TEX의 수학 모드에서 사용된다고 가정하고 그것의 의미를 정의해야 한다. (위의 명령어를 통해서 TEX 명령어를 만들 때, 밑줄은 `x`로 변환되고, 달러 기호는 `X`로 변환되므로 *foo_bar*는 `\fooxbar`로 변환된다. 숫자와 그 밖의 다른 문자들은 아무런 변환 없이 그대로 출력된다. 그래서 그 문자들을 명령어의 일부가 아니라 명령어의 파라미터들로 간주한다. 예를들어, 아래와 같은 문장은

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

`x1`과 `x2`를 *x1*과 *x2*가 아닌 x_1과 x_2로 만든다.)

만약에 r이 특별한 식별자 '*make_pair*'라면, 식별자 l은 C++ 함수 템플릿(function template)처럼 취급된다. 예를 들어, `@f convert make_pair` 다음에 '`convert<int>(2.5)`'라고 쓸 수 있는데, 이렇게하면, `<`와 `>`가 수식의 부등호로 오인하는 일 없이 사용할 수 있다.

한가지 다행인 것은 CWEAVE는 **typedef**로 정의된 식별자는 예약어로 간주하기 때문에 CWEB 프로그래밍하는데 있어서 포멧 정의를 쓸 일은 그다지 많지 않다.

`@s` $[TM \to M;\ L]$ 포멧 정의 명령어 `@f`와 동일한 기능을 하지만, 이 명령어를 사용하면 CWEAVE는 정의를 TEX 파일에 출력하지 않는다. 마찬가지로 뒤에 C의 주석문이 올 수 있다. 주로 `@i` 파일들 내에서 사용된다.

다음으로 마디의 C언어부에서 사용되는 명령어들을 설명한다.

`@c` `@p` $[TM \to C]$ 이름이 없는 마디의 C언어부는 `@c`로 시작한다. (또는 `@p`로 시작하는데, 이는 "프로그램"을 의미하고 이 둘은 같은 의미이다.) CTANGLE은 이 명령어 다음에 나오는 C 텍스트들로 3쪽에서 설명한 대로, 1차 패스 프로그램 텍스트를 만든다. CWEAVE는 TEX 파일에 '`@c`'를 출력하지 않는다. 그래서 TEX 기능을 위주로 하는 CWEB 문서를 기반으로 하는 CWEB 파일을 만들고자 한다면, 이름 없는 마디의 적당한 위치에 `@c`를 넣어야 하는 것을 잊지 말아야 한다.

`@<` $[TM \to C;\ C;\ c]$ $*$ 이 명령어 다음에는 마디 이름 (또는 앞에서 설명한 대로 다른 것과 구별할 수 있는 최소한의 접두어)이 온다. 마디 이름은 TEX 텍스트가 오고 `@>`로 끝을 맺는다. 전체 `@<...@>`는 개념적으로는 C 요소이다. 이 명령어가 하는 일은 주어진 환경에 따라 달라진다.

`@<`가 T와 M 환경에서 사용될 때, 이 명령어는 뒤따라 나오는 마디 이름을 현재 마디에 덧붙이고, 그 마디의 C 언어부를 시작하도록 한다. `@>` 다음에는 반드시 `=` 또는 `+=`이 나와야 한다.

C 환경에서, `@<`는 이름 있는 마디가 사용된다는 것을 가리킨다.—3쪽에 설명된 대로, CTANGLE에 의해서 그것의 C 코드가 삽입된다. 에러를 발견하기 위한 수단으로, 그러한 마디 이름 다음에 `=`가 나오면, CTANGLE과 CWEAVE는 에러 메시지를 낸다. 왜냐하면 `=`이 나온다는 것은 새로운 마디가 정의된다는 의미이기 때문이다. '`@␣@<...@>=`'는 가능하다. ⟨ foo ⟩ = *bar*와 같은 문장을 반드시 사용해야 한다면, `=` 전에서 한줄 띄워야 한다. 여기서 ⟨ foo ⟩는 정의되는 것이 아니라 사용되는 것이다.

마지막으로, 내부 C 환경 (즉, 마디의 TEX부에 있는 '|...|' 내의 환경이나, 주석문) 에서 @<...@>는 이름 가진 마디가 인용된다는 것을 의미한다. CTANGLE는 이와 같은 것들을 무시한다.

@($[TM \to C;\ C;\ c]$ * 마디 이름은 @(로도 시작할 수 있다. 이 명령어가 하는 일은 한가지만 빼고 @<가 하는 일과 동일하다. CTANGLE은 @(foo@>라는 이름을 가진 마디의 C 코드들를 foo라는 이름의 파일로 출력한다는 것이다. 이 방법을 통해서 하나의 CWEB에서 여러 개의 파일을 얻을 수 있다. (주의할 것은 @d 정의들은 그러한 파일로 출력되지 않고, 오로지 CWEB 파일이 CTANGLE에 의해서 변환되는 .c 파일에만 출력된다는 것이다.) 이 방법이 유용하게 사용되는 한가지 경우는 바로 현재 작성 중인 CWEB 파일과 함께 로드될 다른 프로그램 모듈들을 위한 헤더 파일들을 만드는 경우이다. 또다른 경우는 주어진 프로그램의 테스트 코드를 만들어내는 경우이다. 하나의 CWEB 파일로 주요 소스 파일을 물론 헤더 파일들과 테스트 코드를 관리할 수 있다는 것은 매우 유용하고 편리한 기능이다. @(bar1@>와 @(bar2@>가 동시에 @<foo@>를 사용할 수 있기 때문에, 이름을 가진 하나의 마디가 다른 여러개의 출력 파일들에 속할 수 있다는 것을 명심해야 한다.

@h $[Cc]$ 이 명령어를 사용하면 CTANGLE은 모든 마디의 중간부에서 정의된 매크로들을 이 명령어가 사용된 지점에 #define 문장들로 변환해서 C 파일의 맨 처음에 기록된다. 이 명령어는 매크로 정의들을 헤더 파일들 다음에 위치시키고 싶을 때 헤더 파일들 다음에 이 명령어를 사용하면된다

다음에 설명 할 명령어들은 "명령 텍스트(control text)"라는 것을 소개하는데, 명령 텍스트는 명령어와 '@>' 사이에 있는 텍스트를 말한다. 기억할 것은 '@>'은 반드시 명령어와 같은 줄에 있어야 한다는 것이다. 명령 텍스트의 환경은 제한된 (restricted) 환경이 된다.

@^ $[TMCc]$ * 이 명령어의 명령 텍스트는 C 프로그램의 식별자들 처럼 색인 리스트에 들어간다. 이 텍스트는 색인에서 로마체로 나타난다. 예를 들어, "system dependencies"를 색인에 넣으려면, '@^system dependencies@>' 라고 입력하면 된다.

@. $[TMCc]$ * 이 명령어의 명령 텍스트는 색인에 타자 글꼴로 나타난다.

@: $[TMCc]$ * 이 명령어의 명령 텍스트는 색인에 TEX 매크로 '\9'가 정하는 형태대로 나타나는데, 매크로 \9는 반드시 사전에 정의되어 있어야 한다.

@t $[MCc]$ * 이 명령어의 명령 텍스트는 TEX의 \hbox에 들어가서 그와 이웃하는 C 프로그램과 어울려 쓰여진다. CTANGLE은 이 명령 텍스트를 완전히 무시하지만, CWEAVE은 다양한 방법으로 이 명령어를 유용하게 사용한다. 예를 들어, 이 명령어를 이용해서 C 코드와 수식이 어우러진 '$size < 2^{15}$'와 같은 멋진 주석문을 만들 수 있는데, 이 주석문은 '|size < 2@t^{15}@>|'를 입력하면 얻을 수 있다.

@= $[MCc]$ * 이 명령어의 명령 텍스트는 C 프로그램에 글자 그대로 (verbatim) 복사된다.

@q $[LTMCc]$ 이 명령어의 명령 텍스트는 오로지 CWEB 파일을 읽는 사람에게만 해당되는 주석이어서 CTANGLE과 CWEAVE은 이 명령 텍스트를 완전히 무시한다. 이 명령어는 @i를 이용해서 in limbo에 포함될 파일들을 주로 @q를 이용해서 주석처리하는 경우에 사용된다. 다른 한가지 예는 C 문자열에서 균형이 맞지않는 괄호의 균형을 맞출 때 쓰인다. 이 경우 괄호를 매치시키는 기능을 제공하는 에디터가 괄호의 균형이 맞지 않을때 이상하게 동작하는 것을 막을 수 있다.

@! $[TMCc]$ * 식별자나 명령 텍스트 바로 앞에 '@!'를 붙이면, 그 식별자나 명령 텍스트가 색인에 들어갈 때, 그것들이 사용된 마디 번호에 밑줄이 그어진다. 이 명령어는 식별자가 정의된 마디인지, 사용된 마디인지를 구별할 경우에 사용된다. 한 글자 짜리 예약어나 식별자는 색인에 들어가지 않는데, 한 글자 짜리라고 하더라도 밑줄이 그어진 식별자는 색인에 들어간다. 어떤 식별자가 C 코드에서 정의되거나 선언될 때, CWEAVE는 자동적으로 '@!'를 붙여준다. 예를 들어, 다음과 같은 정의를 하면,

$$\textbf{int}\ array[max_dim],\ count = old_count;$$

변수명 *array*와 *count*는 자동적으로 밑줄이 그어져서 색인에 들어간다. 레이블, $main(\textbf{int}\ argc, \textbf{char}\ *argv[\,])$와 같은 함수 정의, **typedef** 정의들도 CWEAVE가 알아서 자동적으로 밑줄을 그어준다. 프로토타입이 없는 구식 함수 정의는 함수의 인자들을 정의하지 않지만 만약에 그 타입들이 함수의 몸체 전에 '$\textbf{int}\ argc;\ \textbf{char}\ *argv[\,];\ \{\ldots\}$'

처럼 보통의 방식으로 정의되었다면, 색인에 밑줄이 그어져서 들어간다. 따라서, 흔히 사용되는 식별자의 정의나,

```
enum boolean {@!false, @!true};
```

위와 같은 경우를 제외하면, `@!`를 자주 사용할 일을 없을 것이다. 위 경우 `@!`를 사용하는 것을 매우 권장하는데, 이는 열거형 자료형 정의는 색인에 들어갈 때 기본적으로 밑줄이 그어지지 않기 때문이다.

다음으로 CTANGLE만이 사용하는 명령어를 살펴보자.

`@'` $[MCc]$ 이 명령어는 CWEB과 원래의 WEB에서 전혀 다르게 동작하기 때문에 이 명령어 사용에는 주의를 기울여야 한다. CWEB에서는, 문자열의 길이가 1인 문자열의 ASCII 코드에 해당하는 10진수를 만들어낸다. (예, `@'a'`는 97로, `@'\t'`는 9로 CTANGLE된다.) 이 명령어는 비ASCII 컴퓨터에서 ASCII로 작동하는 코드를 만들고자 할때 사용될 것이다. 그러나 대부분의 경우, `<ctype.h>`에서 문자셋에 무관하게 프로그래밍 하도록 되어져 있어서 이 명령어를 사용할 일은 드물다.

`@&` $[MCc]$ 이 명령어는 이 명령어의 좌우에 있는 것을 붙여서 하나의 C 코드를 만들어낸다. 어떠한 공백들이나 줄바꿈들도 이 명령어로 결합된 것을 갈라놓지 못한다.

`@l` $[L]$ CWEB 프로그래머들은 TEX 텍스트 안에서 눈에 보이지 않는 범위인 125–255 중의 임의의 8-비트 문자 코드를 사용할 수 있는 옵션을 가지고 있다. 그러한 눈에 보이지 않는 코드를 문자열에서 사용할 수 있고, 심지어는 C 프로그램의 식별자로도 사용할 수 있다. 기본 ASCII 표준의 다양한 확장 하에서 위 범위의 8-비트 코드들은 엑센트 글자(accented letters), 비-라틴 알파벳 글자들 등과 대응된다. 그러한 문자들이 식별자로 쓰이면, 올바른 C 코드를 생성하기 위해서 CTANGLE은 그들의 표준 ASCII 숫자 혹은 알파벳 코드나 _로 변환해야한다. 이러한 변환은 변환표(transliteration table)를 통해서 이루어진다. 변환표는 기본적으로 문자열 Xab를 ASCII 코드 $^{\#}ab$에 결합시킨다. (여기서 a와 b는 16진수 숫자이고, $a \geq 8$ 이다.) in limbo에서 `@l␣ab␣newstring` 처럼 정의하면, CTANGLE에게 이 문자를 newstring으로 바꾸라고 명령을 하게되는 것이다. 예를 들어, 글자 'ü'의 ISO Latin-1 코드는 ^FC (또는 `'\374'`) 이고, CTANGLE은 이 코드가 식별자로 쓰였다면, 세글자짜리 XFC로 바꾼다. 만일 `@l fc ue`라고 하면, 그 코드는 대신에 ue로 변환된다.

CWEAVE는 8-비트 문자들을 아무런 변환없이 그대로 TEX에게 보낸다. 그러므로 TEX은 반드시 그것들을 받을 준비가 되어있어야 한다. 만일 당신의 모든 비표준 식별자를 자신 고유의 명령어로 다루고자 한다면, TEX으로 하여금 그 모든 비표준 문자를 글자로 인식하게 하여야 한다. 그렇지 않으면, 당신만의 8-비트 코드들을 TEX의 "active" 코드로 만들던가, 당신이 필요로 하는 특별 문자를 가지고 있는 글꼴을 읽어들여야 한다. (TEX 명령어 `\it`로 선택된 글꼴은 식별자를 나타내는 글꼴로 사용된다.) 당신이 사용하는 언어 작성된 CWEB 사용자를 위한 매크로 패키지가 있는지 확인해 보라. 만약, 그러한 매크로 패키지가 없고, 당신이 용감하다면 직접 만들어 보라.

다음에 설명할 명령어 여덟개(이름하여, '`@,`', '`@/`', '`@|`', '`@#`', '`@+`', '`@;`', '`@[`', '`@]`')는 CTANGLE이 만들어내는 C 프로그램에는 아무런 영향도 미치지 못하는 것들이다. 이 명령어들은 특수한 환경 하에서 단순히 CWEAVE가 만들어낸 C 코드 문서를 보기 좋게 하는데 쓰이는 명령어들이다. CWEAVE에 내장된 문서 포멧 도구는 매우 좋아서 문법적으로 올바른 C 텍스트를 잘 다룰 수 있다. 하지만, 모든 경우를 다 커버하는 것은 아니다. 왜냐하면 매크로나 마디 이름에 포함된 텍스트의 일부분도 다룰 수 있어야 하는데, 그 일부분은 C의 문법을 따르지 않을 수도 있기 때문이다. 비록 CWEB이 자동 포멧을 당신의 구미에 맞게 수정할 수 있는 도구를 제공하긴 하지만, 수정하지말고 일단 CWEAVE가 자동적으로 처리하는 것을 자세히 지켜보라. 그러면 대부분의 경우, CWEAVE가 놀랄 정도로 당신이 원하는 대로 올바르게 처리할 것이므로, 당신이 직접 수정해서 새로운 포멧 도구를 만들 일은 거의 없을 것이다.

`@,` $[MCc]$ 이 명령어는 CWEAVE가 만들어낸 파일 안에 작은 간격을 넣는다. 이 간격은 예를 들어, 두 식별자가 서로 붙어 있는 등 일반적이지 않은 방식으로 매크로들을 사용할 때 필요하다.

`@/` $[MC]$ 이 명령어는 CWEAVE가 만들어낸 C 코드 문서에서 강제로 줄바꿈을 할 경우에 사용한다. 줄바꿈은 99% TEX이 알아서 올바르게 자동적으로 해줄것이나, 당신은 때대로 프로그램의 논리상 줄바꿈을 하면 프로그램을 더욱 이해하기 쉬운 곳이 있을 것이다. 그때에 이 명령어를 사용하면 된다. 만약에 줄바꿈 하고자 하는 지점 바로 뒤에 주석문이 나온다면, 주석문 전에서 줄바꿈을 하기 위해서는 '`@/@,`'를 입력해야 한다.

`@|` $[MC]$ 이 명령어는 표현식 한가운데서 선택적으로 줄바꿈을 할 때 사용한다. 예를 들어, 대입문의 우변이 매우 긴 경우에 이 명령어 `@|`를 써서 줄바꿈을 하여 TEX이 규칙에 의해서 만들어내는 것보다 보기 좋고 프로그램의 논리 흐름상 더 좋은 문서를 만들 수 있다.

`@#` $[MC]$ 이 명령어는 `@/` 처럼 강제적으로 줄바꿈할 때 사용한다. 한 가지 다른점은 줄바꿈한 후에 그지점에 약간의 간격을 둔다는 것이다. 예를 들어, 프로그램 구조상 의미 있는 것들을 그룹으로 하여 하나의 코드 무리를 만들 수 있는데, 각 그룹들 사이에 약간의 간격이 있다면, 프로그램 소스를 보기도 좋고 이해하기도 쉬울 것이다. 관례상 전역 변수 선언부와 함수 선언부 사이에, 선언과 실행문 사이에 약간이 필요한데, 이때 이 명령어를 쓰면 된다.

`@+` $[MC]$ 이 명령어는 특별한 주문이 없는 한, CWEAVE가 알아서 자동적으로 줄바꿈하는 자리, 예를 들어, '**else**' 바로 앞 지점 같은 자리에 줄바꿈이 일어나지 못하도록 한다. if–else 구문을 한 줄에 사용하고 싶은 경우에 사용하면 된다. 만약에 함수의 몸체가 시작되는 부분에 '`{@+`'라고 입력하면, 함수 몸체의 첫번째 문장은 이 왼쪽 괄호와 같은 줄에 생기고, 들여쓰기 정도는 다음 줄의 문장과 같은 만큼 들여쓰기 된다.

`@;` $[MC]$ 이 명령어는 문서를 보기 좋게 하는 목적으로 세미 콜론과 같이 취급되는데, 다른 점은 세미 콜론이 눈에 보이지 않는다는 것이다. 예를 들어, 당신은 이 명령어를 C 코드 안에서 사용되는 마디 이름뒤나 매크로 다음에 사용할 수 있다. 이때 마디나 매크로는 여러 줄로 구성된 것이면 이 명령어를 사용하는 의미가 더욱 확실하겠다. 다음과 같은 문장을 생각해 보자.

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

위에서 *macro*는 괄호로 둘러싸인 여러 문장으로 된 매크로라고 하자. 여러 문장으로 된 매크로가 사용될 때, 세미콜론을 붙이면 C 문법에 맞지 않는데, 이 명령어를 이용하면 그것을 해결 할 수 있다.

`@[` $[MC]$ `@]`를 보라.

`@]` $[MC]$ CWEAVE가 표현식으로 사용할 프로그램 텍스트 주위에 `@[...@]`로 둘러싸라. (이 명령어는 경우에 따라서 비정상적인 매크로 인자에 적용되기도 한다.) 또는 아래와 같이, '`@[@]`'를 함수의 포인터를 선언하는 경우에 자료형 이름과 왼쪽 괄호 사이에 넣어라.

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

만약 그렇게 하지않으면, CWEAVE는 위 선언문을 C++ 표현식 '**int**$(*f)$'의 첫번째 부분과 혼동할 것이다. 또다른 예는, C 코드 중간에 저수준 명령 `#define`를 사용하고자 할 때, 그리고 정의가 캐스트로 시작할 때 이 명령어를 사용하라.

```
#define foo @[(int)(bar)@]
```

남은 명령어들은 CWEB 입력을 처리하는 명령어들이다.

`@x @y @z` $[$*change_file*$]$ CWEAVE와 CTANGLE은 *web_file*과 *change_file*이라고 불리는 두 종류의 입력 파일을 다룰 수 있도록 디자인 되었다. 여기서 *change_file*은 *web_file*에서 변경 할 부분을 대체할 자료들을 가지고 있다. 이 두 파일을 합하여 생긴 파일이 사실상 CWEB 파일이라고 불리는 것이다.

이 명령어가 동작하는 방식은 다음과 같다. 수정 파일(change file)은 0개 이상의 수정 항목을 가지고 있는데, 수정 항목은 '`@x`⟨old lines⟩`@y`⟨new lines⟩`@z`'와 같은 형태를 가지고 있다. 특수 명령어 `@x`, `@y`, `@z`는 오로지 수정 파일에서만 쓰일 수 있는데, 반드시 줄의 첫번째 컬럼부터 시작되어야 하고, 그 줄에서 위 형식이 갖추어지고난 나머지는 무시된다. ⟨old lines⟩은 *web_file* 내에 있는 줄들과 정확히 일치해야 하고, ⟨new lines⟩은 0개 이상의 줄로 구성되는데, 앞의 old를 대체할 줄들로 구성된다. 수정 항목의 "old line"의 첫번째 줄과 정확히 일치하는 *web_file*의 줄이 발견될 때마다, 수정 항목의 나머지 다른 줄들도 반드시 일치하여야 한다.

수정 파일은 수정 항목들 사이, 첫번째 수정 항목 앞, 마지막 수정 항목 뒤에, '`@x`', '`@y`', 또는 '`@z`'로 시작하지 않는 임의의 갯수의 줄을 가질 수 있다. 그러한 줄들은 설명이나 주석을 담고 있어서 그냥 무시되는 줄이지, 일치하는 목적으로 사용되는 줄들이 아니다.

이러한 두 종류의 파일을 입력으로 취하는 특징은 다음과 같은 경우에 사용하면 적당하다. 우선 `tangle.w` 또는 `weave.w` 또는 `tex.web`와 같은 마스터 CWEB 파일이 있고, 이 파일 자체를 수정하고 않고, 현재 로컬 시스템에서 CWEB 파일로 만들어진 실행 파일이 동작하도록 하기 위해서 CWEB 파일의 일부가 변경될 필요가 있을 때, 그 변경

부분만을 골라내어 수정 파일을 만드는 경우이다. 이렇게 하면, 로컬 시스템에 맞게 변경된 부분을 디버깅할 때, 마스터 웹파일을 건드리지 않고 작업할 수도 있다. 그래서 일단, 수정 파일이 제대로 동작한다는 것이 확인되면, 자주 새롭게 배포되는 마스터 웹파일에 간단하게 이 수정 파일을 접목 시킬 수 있다.

`@i` [*web_file*] 더구나 *web_file* 자체는 여러 파일들을 조합해서 구성할 수 있다. CWEAVE나 CTANGLE이 파일을 읽다가 어떤 줄의 처음에 명령어 `@i`를 만나면, 현재 읽고 있는 것을 잠깐 멈추고 `@i` 다음에 나오는 파일 이름의 파일을 읽어들인다. 이는 마치 C 프로그램에서 `#include`를 만났을 때 C 전처리기가 취하는 행동과 동일하다. 이 명령어로 포함하는 파일을 모두 읽어들이면, 프로그램은 원래 읽어들이던 파일의 그 장소로 옮겨가서 계속 파일 내용을 읽어들인다. `@i` 다음에 나오는 파일 이름은 " 문자로 둘러쌀 수 있는데, 반드시 그렇게 해야하는 것은 아니다. 포함되는 파일에 `@i`가 쓰여서 또 포함할 수 있다. 즉 중첩해서 파일을 포함하는 것이 가능하다.

수정 파일은 `@i`로 시작하는 줄들을 가질 수 있다. 이러한 방법으로 포함할 파일을 다른 파일로 바꿀 수 있다. 개념적으로 보면, 위에서 설명한 치환 체계는 우선 자기에게 주어진 일을 처리하고, 그 결과는 `@i`를 위해 검사된다. 만약 어떤 수정 파일의 `@y`와 `@z` 사이에 `@i foo`이 있다면, `foo` 파일의 개별적인 줄들과 그것이 포함하는 파일들은 변경할 수 없다. 그러나 파일의 줄들을 변경할 수 있다.

UNIX 시스템 상에서 (그리고 다른 환경 변수를 제공하는 다른 시스템 상에서), 환경 변수 CWEBINPUTS이 정해졌거나, 컴파일시에 컴파일 플래그(flag)로 CWEBINPUTS이 정의되었다고, 현재 디렉토리에 포함하려는 파일이 없다면, CWEB은 포함할 파일을 위 변수에 정의된 디렉토리에서 찾으려고 할 것이다.

추가적인 특징과 주의사항들

1. 확장 문자셋이 사용가능한 시스템에 CWEB이 설치된 경우에 다음과 같은 문자들은 '++', '--', '->', '!=', '<=', '>=', '==', '||', '&&', '<<', '>>' 각각 다음 문자들로 축약될 수 있다: '↑', '↓', '→', '≠', '≤', '≥', '≡', '∨', '∧', '⊂', and '⊃'.

2. 만약 확장 문자셋을 사용할 수 있다면, 앞서 설명한 `@l` 명령어 규칙대로 단지 최소의 제약조건들만로 지키고 그 확장 문자셋을 사용할 수 있다. 그러나 프로그램을 작성하고자 한다면, 확장 문자셋을 사용하지 못하는 불쌍한 영혼들을 위해서 기본 ASCII 문자들을 주로 사용하기 바란다.

3.CWEAVE가 만들어낸 TEX 파일은 한 줄에 최대 80개의 문자들이 들어있는 줄들로 나뉜다. TEX 텍스트를 복사하면, 기존의 줄 바꿈 표시도 복사된다. 파일을 작성할 때 너무 이상한 짓만 하지 않는다면, CWEAVE는 TEX 주석문이 두줄 이상에 걸쳐서 나눠질때 그것을 인식해서 새 줄이 시작할때마다 '%'를 입력할 것이다.

4. C 텍스트는 "상향식(bottom up)" 절차로 TEX 텍스트로 변환된다. 상향식 절차는 C의 각각의 토큰을 "구문(part of speech)"으로 여기고 CWEAVE의 문서에 설명된 특별 규칙에 따라서 그 구문들을 결합하여 점점더 포함하는 큰 절로 만든다. 간단한 식별자나 짧은 표현식을 위한 변환 규칙을 배우는 것은 그저 CWEAVE가 동작하는 몇가지 예제만 보는 것만 살펴보는 것으로 족하다. 그러나 CWEAVE가 단지 C 텍스트만을 다루는 것이 아니기 때문에 일반적인 규칙은 다소 복잡하다 더구나 출력물은 사용된 글꼴과 원하는 페이지 너비를 기반으로 해서, 필요에 따라 TEX이 들여쓰기를 하고 줄바꿈을 하도록 하는 코드를 포함한다. 최상의 출력물을 얻고자 한다면, 긴 C 텍스트를 `|...|`로 둘러싸는 일은 삼가해야겠다. 왜냐하면 `|...|`로 C 텍스트를 둘러싸면 그 텍스트는 TEX 텍스트로 변환되는데, 이때 C 텍스트 고유의 들여쓰기나 줄바꿈 같은 것들이 완전히 제거되기 때문이다. `|...|`로 둘러싸는 C 텍스트는 간단한 표현식이나 문장만을 고려하는 것이 좋다. 만일 C의 전처리기 명령을 `|...|`로 둘러싼다면, #는 줄의 맨 앞에 와야한다. 그렇지 않으면, CWEAVE가 제대로된 출력을 내보내지 않을 것이다.

5. 주석은 `|...|` 텍스트 안에서 사용할 수 없다. '|'가 TEX 텍스트를 C 텍스트로 바꾸하는 신호를 준 후에, 문자열이나 명령 텍스트나 마디 이름의 일부가 아닌 그 다음 나오는 '|'가 C 텍스트를 종료시킨다.

6. 주석문은 반드시 왼쪽과 오른쪽 괄호가 적절히 매치되어야 한다. 그렇지 않으면, CWEAVE가 에러 메시지를 뿜어낼 것이다. 그러나 CWEAVE는 TEX을 너무 심한 혼란에 빠뜨리지 않게 하기 위해서 괄호의 균형을 맞추려고 시도한다.

7. CWEB 프로그램을 디버깅하려고 하는데, C 코드 중 일부를 제외하고 하기로 했다고 하자. 이때 단순히 그 제외하고자 하는 코드를 절대로 주석 처리하면 안된다. 그러한 상황에서의 주석 처리는 CWEB이 문서를 만들어내는 정신에 위배되는 행동이다. 왜냐하면, 그러한 주석들은 마치 프로그램의 설명 중 일부인양, 프로그램을 읽는 사람들에게 보여지기 때문이다. 게다가, 프로그램의 주석문은 반드시 올바른 TEX 텍스트 이어야 한다. 따라서 만약 주석문 처리를 `/*|...|*/` 처럼 하지 않고, `/*...*/` 처럼 한다면, CWEAVE는 혼란에 빠질 것이다. 어떤 C 코드를 반드시 주석 처리해야 한다면, `#if 0==1`

and `#endif`와 같은 전처리기 명령을 이용해서 해야 한다. . 8. 명령어 `@f`는 한 식별자를 TEX이 다른 식별자 다루듯이 다루게 할 때 사용하는 명령어인데, 이러한 `@f`를 이용한 포멧 정의가 여러번 나올 때, 그 명령이 실행되는 순서는 포멧 정의가 정의된 대로 순차적으로 실행된다. 일반적으로, 주어진 한 식별자는 문서 전체를 통해서 한가지 모양으로 출력된다. 그리고 그 모양은 그 식별자에 `@f` 명령어가 적용되기 전까지 계속 유지된다. 그 이유는 CWEAVE는 주어진 파일을 두 번의 패스를 통해서 TEX 파일을 만들어내기 때문이다. 우선 CWEAVE는 첫번째 패스동안에 `@f`로된 포멧 정의들과 교차-참조들을 처리하고, 그 결과를 두번째 패스로 보낸다. (그러나, `typedef` 정의에 의해서 굵은 글꼴의 모양을 갖게된 식별자들에게는 이 규칙들이 적용되지 않는다. 그 식별자들은 `typedef` 정의 전과 후의 모양이 다르다. 이러한 상황이 그리 좋은 것만은 아니지만, 그렇다고 쉽게 고칠 수 있는 것도 아니다. 이러한 제약 상황을 해결하는 방법은 `typedef` 정의를 하기 전 또는 후에 '`@s foo int`'라고 입력하는 것이다.)

9. TEX으로 조판된 C 코드에 약간의 간격을 넣는 것이 '`@,`'가 만들어내는 좁은 간격보다 더 자연스러울 때가 종종 있다. `@t`의 특징이 이러한 목적으로 사용될 수 있다. 예를 들어, TEX 명령어 `\4`는 cwebmac에 들여쓰기 간격만큼 왼쪽으로 들여쓰라는 명령이므로, '`@t\hskip 1in@>`'는 1인치의 공간을 마련할 것이다. '`@t\4@>`'는 들여쓰기 간격만큼 왼쪽으로 들여쓰는 명령이다. (이 명령어는 예를 들어, 레이블이 있는 줄의 시작부에서 레이블과 줄의 시작부와 약간의 간격을 두기위해서 사용된다.) 그리고 또한 표현식의 중간에서 강제적으로 간격을 두기위해서 '`@t}\3{-5@>`'를 사용할 수도 있다.

10. CWEB에서 사용된 각각의 식별자는 그 고유의 표현 방식이 있다. 그러므로 자료형 이름과 `struct`의 일부분을 표시하기 위해서 비록 C가 그것을 허용한다고 하더라도, 같은 식별자를 사용해서는 안된다.

프로그램 실행하기

UNIX에서 CTANGLE은 다음과 같은 명령으로 실행할 수 있다.

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

CWEAVE도 위와 동일한 방법으로 실행할 수 있다. 만약 '`-`' 나 수정 파일(change file)을 정하지 않으면, 수정 파일은 프로그램 상에서 널(null)이 된다. 확장자 `.w`와 `.ch`는 주어진 파일 이름에 점이 없을 때에만, 자동으로 붙는데, 찾고자하는 웹 소스 파일을 찾을 수 없을때는 확장자 `.web`를 붙여서 다시 시도한다. 예를 들어, 명령어 '`cweave cob`'는 먼저 `cob.w` 파일을 읽어들이려 시도할 것이고, 그 시도가 실패하면, 포기하기 전에 `cob.web`를 읽어들이려 시도한다. 결과 파일 이름을 정하지 않으면, CTANGLE은 주어진 파일 이름에 `.c`를 붙이고, CWEAVE는 `.tex`을 붙여서 파일을 생성한다. CWEAVE가 부수적으로 만드는, 색인 관련 파일들은 확장자 `.idx`와 `.scn`가 붙어서 자동으로 생성된다.

간단 명료한 것을 좋아하는 프로그래머는 자신이 사용하는 OS의 쉘을 조정해서 '`wv`'를 '`cweave -bhp`'로 확장할 수도 있는데, 그 방법은 에러 메시지를 제외하고는 CWEAVE이 모니터로 내뿜는 모든 메시지를 무시한다.

옵션은 `-` 나 `+` 기호와 함께 나오는데, `-`는 옵션을 끄는 역할을 하고, `+`는 켜는 일을 한다. 예를 들어, 옵션 '`-fb`'는 `f`와 `b` 옵션의 역할을 못하도록 끄고, '`+s`'는 `s`옵션을 켠다. 옵션들은 파일 이름 앞에 올 수도 있고, 뒤에 올 수도 있고, 앞뒤에 올 수도 있다. 아래에 설명하는 옵션들은 현재 프로그램에서 구현된 것들이다.

`b` 프로그램이 실행될 때, 배너를 출력한다. (기본으로 실행되는 옵션)

`e` CWEAVE가 C 코드를 `\PB{...}`로 감싼다. 그래서 그 코드에 특별한 일을 할 수 있다. (기본으로 꺼져있는 옵션이고, CTANGLE에는 아무런 영향을 미치지 않는다.)

`f` CWEAVE가 C 문장들을 다룰 때, 그 문장들 다음에 강제적으로 줄바꿈을 한다. (기본 실행 옵션이다. `-f` 옵션은 줄바꿈을 하지 않으므로 종이를 절약할 수 있느나, C 코드 처럼 보이지 않을 수 있다. CTANGLE에는 전혀 영향을 미치지 않는 옵션이다.)

`h` 프로그램이 실행을 정상적으로 끝마쳤을때, 제대로 실행했다는 행복한 메시지를 뿌려준다. (기본 실행 옵션이다.)

`p` 프로그램이 실행할 때 진행상황을 보여준다. (기본 실행 옵션이다.)

`s` 프로그램이 실행을 정상적으로 끝마쳤을 때, 프로그램의 메모리 사용에 대한 통계를 보여준다. (기본 실행 옵션이 아니다.) 만약 프로그래밍하는 CWEB 파일이 매우 크거나 마디의 수가 많다면, CTANGLE 이나/또는 CWEAVE의 용량을 얼마나 초과하는지를 살펴볼 필요가 있을 것이다.

`x` CWEAVE가 만들어낸 TEX 파일이 색인이나 목차 정보를 포함하도록 한다. (기본 실행 옵션이고, CTANGLE에는 전혀 영향을 끼치지 않는다.)

문서를 만들기위해 더 알아야할 세부사항

CWEB 프로그래밍을 하다보면, CWEAVE가 처리하는 방식이 맘에 들지 않을 때가 있다. 그 상황을 참지 못하겠다면, CWEAVE가 동작하는 방식이 기록된 문법표를 수정해서 CWEAVE를 자신이 원하는대로 동작하도록 만들수 있다. 그렇게 하려면, CWEAVE 프로그램의 소스를 직접 수정해야 하는데, 이 일이 경우에 따라서는 즐거운 일이 될수도 있다. 어쨌든 CWEAVE의 동작 원리가 기록된 문법표는 CWEAVE 프로그램 소스 코드에 기록되어 있는데, 이를 별도의 파일로 만들어서 보고싶으면, CWEB의 소스 파일 중에서 prod.w라는 파일이 있는데, 'cweave -x prod', 곧 이어 'tex prod'라는 명령들을 통해서 만들 수 있다. CWEAVE가 어떻게 C 코드를 파싱해서 아름다운 문서를 만드는지를 자세히 알고싶으면, 그 C 코드 앞에 '@ @c @2'라고 입력하면 된다. (명령어 @2는 CWEAVE가 하는 일을 자세히 볼 수 있도록 스위치를 켜는 역할을 하고 그 반대로 스위치를 끄는 명령어는 '@0'이다.) 예를 들어, 아래와 같은 코드를 CWEAVE로 돌리면,

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

모니터에 다음과 같은 암호같은 코드들이 나온다.

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

첫번째 줄은 CWEAVE가 C 코드를 파싱하는 문법 중에 4번째 규칙을 적용한다는 것을 의미하고, 현재 이것을 처리하는 순간의 CWEAVE가 사용하는 메모리는 "스크랩(scraps)"이라고 불리는 일련의 TEX 코드들을 가지고 있는데, 그 일련의 TEX 코드들은 각각 *exp* (표현식), 여는 괄호, 다시 *exp*, 닫는 괄호, 그리고 나머지 아직 파서가 처리하지 않고 남아 있는 스크랩들이라는 것을 나타낸다. (+ 와 - 기호는 TEX이 스크랩으로 나누는 경계에 있을때 수학 모드 안에 또는 밖에 있어야 한다는 것을 분명히하는데 사용되고, *는 현재 파서의 위치를 나타낸다.) 그리고나서 11번째 규칙이 적용되고, (*exp*)는 *exp*이 되는 등, 연속적으로 적용이 되고있다는 것을 보여준다. 이러한 과정을 통해서 위의 코드는 결국은 함수(*function*) 형이라는 커다란 스크랩이 되는 것이다.

CWEAVE의 문법은, 입력된 C 코드를 완전히 이해하여 언제나 올바르게 적용되지는 않아서, 입력한 C 코드가 모두 한 줄에 다 나와 버리는 경우가 생긴다. 예를 들어, 위의 프로그램에서 '@<Argument definitions@>'가 'char **argv;' 대신에 나왔다고 해보자. 그러면 CWEAVE는 마디 이름을 그냥 단순히 *exp*라고 판단하기 때문에 다소 혼란에 빠진다. 그래서 TEX에게 '*main*(*argc*, *argv*)'와 같은 줄에 '⟨ Argument declarations 2 ⟩'라는 형태로 출력할 것이다. 이 경우 CWEAVE가 올바르게 동작할 수 있도록 'main(argc,argv)' 다음에 '@/'를 써주어야한다.

CWEAVE는 자동적으로 선언문들과 그 다음의 첫번째 실행문 사이에 약간의 공간을 두는데, CWEAVE 명령을 실행할 때, 옵션으로 -o를 주어서 전체적으로 공간을 두지 말라고할 수도 있다. 이렇게 전체적으로 공간을 없애는 것이 아니라 필요한 곳에서만 공간을 없앨수도 있는데, 그 방법은 아래와 같다.

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

'@#'는 '⟨ Other locals ⟩' 다음에 약간의 공간을 만든다.

하이퍼텍스트와 하이퍼도큐먼트화

CWEB이 월드와이드웹과 유사한 점이 많다는 것을 눈치 챈 사람이 많을 것이다. 사실 CWEB의 매크로들은 CWEAVE의 결과물인 TEX 파일을 PDF 파일 (Portable Document Format)로 쉽게 바꿀 수 있는 기능들을 가지고 있다. 그래서 Adobe의 Acrobat Reader를 이용하면 CWEB 문서 내에서 하이퍼링크를 클릭하여 여기 저기를 자유롭게 오고 갈 수 있다. 이 특징은 Mark A. Wicks가 개발한 dvipdfm이라는 오픈 소스 프로그램으로 가능하게 되었다. CWEAVE를 이용해서 cob.w를 cob.tex로 변환한 후에 보통의 방식으로 TEX 컴파일을 하는 것이 아니라, 아래와 같은 명령들을 이용하면 하이퍼텍스트 문서를 만들 수 있다.

```
tex "\let\pdf+ \input cob"
dvipdfm cob
acroread cob.pdf
```

(위의 명령들을 가능하도록 매크로를 만든 Hans Hagen, César Augusto Rorato Crusius, Julian Gilbey 에게 감사한다.) 위 방법 말고 더 쉬운 방법이 있는데, 단순히 'pdftex cob'라고 하면 곧바로 cob.pdf를 얻을 수 있다. 이에 대해서는 Hàn Thế Thành 과 Andreas Scherer 에게 감사한다.

CWEAVE와 비슷한 기능을 하지만, 그보다는 좀 더 공들인 CTWILL라 불리는 프로그램이 있는데, 이 프로그램은 CWEAVE의 일반적인 교차 참조 (cross reference) 기능을 좀 더 확장한 프로그램으로, 변수가 사용된 지점에 연결된 링크를 통해서 곧바로 그 변수가 정의된 곳으로 갈 수 있다. CTWILL은 책이나 문서 같은 하드 카피를 만들기 위해 작성된 프로그램이지만 이 프로그램도 하이퍼텍스트 기능을 갖출 수 있다. 도널드 커누스의 책, *Digital Typography* (1999) 11장을 살펴보라. 그리고 CTWILL 프로그램의 소스들을 ftp://ftp.cs.stanford.edu/pub/ctwill 에서 찾을 수 있다.

부록

부록 A는 CWEB 언어로 작성한 실제 프로그램의 예로써, CWEB 시스템의 그 자체의 소스에서 일부를 발췌하였다. 이 부록에 실린 내용을 주의 깊게 살펴보면, CWEB 프로그램에 대한 기본 개념과 감을 잡을 수 있다.

부록 B는 CWEAVE의 결과물인 TEX 파일을 처리하기 위해서 컴파일러 TEX을 설정하는 매크로 파일을 설명하고, 부록 C는 다양한 출력물을 얻기 위해서 그 매크로들을 사용하는 방법을 설명한다.

사실 이 사용자 설명서는 완벽한 버전이 아니다. CWEB 시스템의 사용자 설명서의 완벽한 버전은 UNIX 명령어 make fullmanual로 얻을 수 있고, 그 완벽한 버전에는 이 설명서에는 없는 부록 D, E, F가 더 있으며, 그 부록은 CWEB 시스템의 소스 코드를 담고 있는 프로그램 파일인 COMMON, CTANGLE, CWEAVE를 가지고 있다.

부록 A: CWEB 프로그램에서 발췌한 예제

부록 A는 네 부분으로 구성되어 있다. 첫번째는 CWEAVE와 CTANGLE이 공통적으로 사용하는 루틴들을 담고 있는 common.w 파일의 12–15 마디들을 만드는 소스 코드이다. 아래 소스 리스트를 보면, 사실은 필요없는 프로그램의 들여쓰기등이 되어 있는 것을 알 수 있다. 들여쓰기 같은 것은 사용자가 어떻게 하던 상관없이 CWEAVE와 CTANGLE 프로그램이 C 코드를 이해하고 있어서 알아서 자동으로 들여쓰기등 보기 좋은 모양으로 원래의 소스 리스트를 편집하여 문서로 만들어준다. 하지만, 지금처럼 CWEB 소스 파일을 직접 살펴 볼 경우도 있으므로 이와 같은 들여쓰기를 할 필요도 있다.

두번째와 세번째는 첫번째 부분에 해당하는 CTANGLE이 만들어낸 C 코드 출력물과 CWEAVE가 만들어낸 TEX 코드 출력물이다. 네번째는 최종 출력 문서이다.

```
@ 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);
}
```

이 페이지는 앞 페이지의 소스에 해당하는, CTANGLE이 만들어낸 C 코드를 담고 있다. 13, 14, 15번 마디가 12번 마디 안으로 들어간 것(tangled)을 확인하라.

```
/*: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:*/
```

common.tex 파일에서 앞의 페이지들에 해당하는 부분을 발췌한 것이다.

```
\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
```

앞 페이지에 해당하는 부분을 최종 문서로 만들어 낸 결과이다.

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* ≡ *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.

```
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.

부록 B: cwebmac.tex 파일

부록 B는 "플레인 TEX" 포멧을 확장하여 CWEAVE가 만들어 낸 출력물을 멋진 문서로 만들기 위한 매크로들을 담고 있는 포멧 파일을 보여준다.

```
% standard macros for CWEB listings (in addition to plain.tex)
% Version 3.64 --- February 2002
\ifx\documentstyle\undefined\else\endinput\fi % LaTeX will use other macros
\xdef\fmtversion{\fmtversion+CWEB3.64}
\chardef\cwebversion=3 \chardef\cwebrevision=64
\newif\ifpdf
\ifx\pdf+\pdftrue\fi
% Uncomment the following line if you want PDF goodies to be the default
\ifx\pdf-\else\pdftrue\fi
\def\pdflinkcolor{0 0 1} % the RGB values for hyperlink color
\newif\ifpdftex
\ifx\pdfoutput\undefined \pdftexfalse \else\ifnum\pdfoutput=0 \pdftexfalse
\else \pdftextrue \pdfoutput=1 \input pdfcolor \let\setcolor\pdfsetcolor \fi\fi
\newif\ifacro \ifpdf\acrotrue\fi \ifpdftex\acrotrue\fi

\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{}
 % with this definition of \9 you can say @:sort key}{TeX code@>
 % to alphabetize an index entry by the sort key but format with the TeX code
\font\eightrm=cmr8
\let\sc=\eightrm % for smallish 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\){{\tentex\kern-.05em}\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{{\acrofalse\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~\ifacro{\pdfnote#2.}\else#2\fi.\par}}

\newtoks\toksA \newtoks\toksB \newtoks\toksC \newtoks\toksD
\newtoks\toksE \newtoks\toksF \newtoks\usersanitizer
\newcount\countA \countA=0 \newcount\countB \countB=0
\newcount\countC \countC=0
\newif\iftokprocessed \newif\ifTnum \newif\ifinstr
{\def\\{\global\let\spacechar= }\\ }

\ifacro % The following are pdf macros
\def\thewidth{\the\wd0 \space}
\def\theheight{\the\ht\strutbox\space}
\def\thedepth{\the\dp\strutbox\space}
\ifpdftex
```

```
  \ifx\pdfannotlink\undefined\let\pdfannotlink\pdfstartlink\fi% for pdfTeX 0.14
  \def\pdflink#1#2{\hbox{\pdfannotlink height\ht\strutbox depth\dp\strutbox
    attr{/Border [0 0 0]} goto num #1 \BlueGreen #1\Black\pdfendlink}}
\else\def\pdflink#1#2{\setbox0=\hbox{\special{pdf: bc [ \pdflinkcolor ]}{#1}%
    \special{pdf: ec}}\special{pdf: ann width \thewidth height \theheight
      depth \thedepth << /Type /Annot /Subtype /Link
      /Border [0 0 0] /A << /S /GoTo /D (#2) >> >>}\box0\relax}\fi
\def\pdfnote#1.{\setbox0=\hbox{\toksA={#1.}\toksB={}\maketoks}\the\toksA}
\def\firstsecno#1.{\setbox0=\hbox{\toksA={#1.}\toksB={}%
    \def\makenote{\addtokens\toksB{\the\toksC}\def\makenote{\toksD={}
      \toksC={}\let\space\empty}\makenote}\maketoks}}
\def\addtokens#1#2{\edef\addtoks{\noexpand#1={\the#1#2}}\addtoks}
\def\poptoks#1#2|ENDTOKS|{\let\first=#1\toksD={#1}%
  \ifcat\noexpand\first0\countB=`#1\else\countB=0\fi\toksA={#2}}
\def\maketoks{\expandafter\poptoks\the\toksA|ENDTOKS|%
  \ifnum\countB>`9 \countB=0 \fi
  \ifnum\countB<`0
    \ifnum0=\countC\else\makenote\fi
    \ifx\first.\let\next=\maketoksdone\else
        \let\next=\maketoks
        \addtokens\toksB{\the\toksD}
        \ifx\first,\addtokens\toksB{\space}\fi
    \fi
  \else \addtokens\toksC{\the\toksD}\global\countC=1\let\next=\maketoks
  \fi
  \next
}
\def\makenote{\addtokens\toksB
    {\noexpand\pdflink{\the\toksC}{\romannumeral\the\toksC}}\toksC={}\global\countC=0}
\def\maketoksdone{\edef\st{\global\noexpand\toksA={\the\toksB}}\st}
\def\pdfURL#1#2{\ifpdftex\pdfannotlink height\ht\strutbox depth\dp\strutbox
  attr {/Border [0 0 0]} user { /Type /Action /Subtype /Link /A
      << /S /URI /URI (#2) >>}\BlueGreen #1\Black \pdfendlink
  \else \ifpdf{\setbox0=\hbox{\special{pdf: bc [ \pdflinkcolor ]}{#1}%
    \special{pdf: ec}}\special{pdf: ann width \thewidth\space height \theheight
      \space depth \thedepth\space << /Border [0 0 0]
      /Type /Action /Subtype /Link /A << /S /URI /URI (#2) >> >>}\box0\relax}%
  \else #1 ({\tt#2})\fi\fi}
{\catcode`\~=12 \gdef\TILDE/{~}} % ~ in a URL
{\catcode`\_=12 \gdef\UNDER/{_}} % _ in a URL
\def\sanitizecommand#1#2{\addtokens\usersanitizer
       {\noexpand\dosanitizecommand\noexpand#1{#2}}}
\def\dosanitizecommand#1#2{\ifx\nxt#1\addF{#2}\fi}

\catcode`\[=1 \catcode`\]=2 \catcode`\{=12 \catcode`\}=12
  \def\lbchar[{]  \def\rbchar[}]
\catcode`\[=12 \catcode`\]=12 \catcode`\{=1 \catcode`\}=2
\catcode`\~=12 \def\tildechar{~}  \catcode`\~=13
\catcode`\|=0 |catcode`|\=12 |def|bschar{\} |catcode`|\=0 \catcode`\|=12
\def\makeoutlinetoks{\Tnumfalse\afterassignment\makeolproctok\let\nxt= }
\def\makeolnexttok{\afterassignment\makeolproctok\let\nxt= }
\def\makeolgobbletok{\afterassignment\makeolnexttok\let\nxt= }
\def\addF#1{\addtokens\toksF{#1}\tokprocessedtrue}
% now comes a routine to "sanitize" section names, for pdf outlines
\def\makeolproctok{\tokprocessedfalse
  \let\next\makeolnexttok % default
  \ifx\nxt\outlinedone\let\next\outlinedone
  \else\ifx{\nxt \else\ifx}\nxt \Tnumfalse \instrfalse % skip braces
  \else\ifx$\nxt % or a $ sign
  \else\ifx^\nxt \addF^\else\ifx_\nxt \addF_% sanitize ^ and _
  \else\ifx\nxt\spacechar \addF\space
  \else\if\noexpand\nxt\relax % we have a control sequence; is it one we know?
    \ifx\nxt~\addF\space
    \else\ifx\nxt\onespace\addF\space
    \else\the\usersanitizer
    \iftokprocessed\else\makeolproctokctli
    \iftokprocessed\else\makeolproctokctlii
    \iftokprocessed\else\makeolproctokctliii % if not recognised, skip it
    \fi\fi\fi\fi\fi
   \else  % we don't have a control sequence, it's an ordinary char
    \ifx/\nxt \addF{\string\/}% quote chars special to PDF with backslash
```

```
    \else\ifx(\nxt \addF{\string\(}\else\ifx)\nxt \addF{\string\)}%
    \else\ifx[\nxt \addF{\string\[}\else\ifx]\nxt \addF{\string\]}%
    \else\expandafter\makeolproctokchar\meaning\nxt
   \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
  \next
}
\def\makeolproctokchar#1 #2 #3{\addF{#3}}
\def\makeolproctokctli{%
  \ifx\nxt\CEE\addF{C}\let\next\makeolgobbletok % \CEE/
  \else\ifx\nxt\UNIX\addF{UNIX}\let\next\makeolgobbletok % \UNIX/
  \else\ifx\nxt\TEX\addF{TeX}\let\next\makeolgobbletok % \TEX/
  \else\ifx\nxt\TeX\addF{TeX}\else\ifx\nxt\LaTeX\addF{LaTeX}%
  \else\ifx\nxt\CPLUSPLUS\addF{C++}\let\next\makeolgobbletok % \CPLUSPLUS/
  \else\ifx\nxt\Cee\addF{C}%
  \else\ifx\nxt\PB \let\next\makeolgobbletok \tokprocessedtrue % \PB{...}
  \else\ifx\nxt\.\tokprocessedtrue\instrtrue % \.{...}
      % skip \|
  \else\ifx\nxt\\\ifinstr\addF{\bschar\bschar}\else\tokprocessedtrue\fi
  \else\ifx\nxt\&\ifinstr\addF&\else\tokprocessedtrue\fi
  \else\ifx\nxt\~\ifTnum\addF{0}\else\addF\tildechar\fi % 077->\T{\~77}
  \else\ifx\nxt\_\ifTnum\addF{E}\else\addF_\fi  % 0.1E5->\T{0.1\_5}
  \else\ifx\nxt\^\ifTnum\addF{0x}\else\addF^\fi  % 0x77 -> \T{\^77}
  \else\ifx\nxt\$\ifTnum\tokprocessedtrue\else\addF$\fi % \T{77\$L}
  \else\ifx\nxt\{\addF\lbchar       \else\ifx\nxt\}\addF\rbchar
  \else\ifx\nxt\ \addF\space        \else\ifx\nxt\#\addF{\string\#}%
  \else\ifx\nxt\PP\addF{++}\else\ifx\nxt\MM\addF{--}%
  \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
}
\def\makeolproctokctlii{%
  \ifx\nxt\MG\addF{->}\else\ifx\nxt\GG\addF{>>}%
  \else\ifx\nxt\LL\addF{<<}\else\ifx\nxt\NULL\addF{NULL}%
  \else\ifx\nxt\AND\addF&\else\ifx\nxt\OR\addF|%
  \else\ifx\nxt\XOR\addF^\else\ifx\nxt\CM\addF\tildechar
  \else\ifx\nxt\MOD\addF{\string\%}\else\ifx\nxt\DC\addF{::}%
  \else\ifx\nxt\PA\addF{.*}\else\ifx\nxt\MGA\addF{->*}%
  \else\ifx\nxt\this\addF{this}\else\ifx\nxt\?\addF?%
  \else\ifx\nxt\E\addF{==}\else\ifx\nxt\G\addF{>=}%
  \else\ifx\nxt\I\addF{!=}\else\ifx\nxt\K\addF{=}%
  \else\ifx\nxt\l\addF{l}\else\ifx\nxt\L\addF{L}%
  \else\ifx\nxt\o\addF{o}\else\ifx\nxt\O\addF{O}%
  \else\ifx\nxt\R\addF!%
  \else\ifx\nxt\T \Tnumtrue \let\next\makeolgobbletok
    \tokprocessedtrue % \T{number}
  \else\ifx\nxt\AM\addF&\else\ifx\nxt\%\addF{\string\%}%
  \fi\fi\fi\fi\fi \fi\fi\fi\fi\fi
  \fi\fi\fi\fi\fi \fi\fi\fi\fi\fi
  \fi\fi\fi\fi\fi\fi
}
\def\makeolproctokctliii{%
  \ifx\nxt\V\addF{||}\else\ifx\nxt\W\addF{&&}\else\ifx\nxt\Z\addF{<=}%
  \else\ifx\nxt\*\addF*\else\ifx\nxt\Xand\addF{\space and\space}%
  \else\ifx\nxt\Xandxeq\addF{\space and_eq\space}%
  \else\ifx\nxt\Xbitand\addF{\space bitand\space}%
  \else\ifx\nxt\Xbitor\addF{\space bitor\space}%
  \else\ifx\nxt\Xcompl\addF{\space compl\space}%
  \else\ifx\nxt\Xnot\addF{\space not\space}%
  \else\ifx\nxt\Xnotxeq\addF{\space not_eq\space}%
  \else\ifx\nxt\Xor\addF{\space or\space}%
  \else\ifx\nxt\Xorxeq\addF{\space or_eq\space}%
  \else\ifx\nxt\Xxor\addF{\space xor\space}%
  \else\ifx\nxt\Xxorxeq\addF{\space xor_eq\space}%
  \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
  \fi\fi\fi\fi\fi
}
\def\outlinedone{\edef\outlinest{\global\noexpand\toksE={\the\toksF}}%
  \outlinest\let\outlinedone=\relax}
\fi % End of pdf macros

\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}%
  \ifpdftex\smash{\raise\baselineskip\hbox to0pt{%
     \let\*=\empty\pdfdest num \secstar fith}}
  \else\ifpdf\smash{\raise\baselineskip\hbox to0pt{%
     \let\*=\empty\special{%
       pdf: dest (\romannumeral\secstar) [ @thispage /FitH @ypos ]}}}\fi\fi}
\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.{% beginning of starred section
  \ifacro{\toksF={}\makeoutlinetoks#3\outlinedone\outlinedone}\fi
  \gdepth=#1\gtitle={#3}\MN{#2}%
  \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
  \def\stripprefix##1>{}\def\gtitletoks{#3}%
  \edef\gtitletoks{\expandafter\stripprefix\meaning\gtitletoks}%
  \edef\next{\write\cont{\ZZ{\gtitletoks}{#1}{\secno}% write to contents file
   {\noexpand\the\pageno}{\the\toksE}}}\next % \ZZ{title}{depth}{sec}{page}{ss}
  \ifpdftex\expandafter\xdef\csname curr#1\endcsname{\secno}
    \ifnum#1>0\countB=#1 \advance\countB by-1
      \advancenumber{chunk\the\countB.\expnumber{curr\the\countB}}\fi\fi
  \ifpdf\special{pdf: outline #1 << /Title (\the\toksE) /Dest
    [ @thispage /FitH @ypos ] >>}\fi
  \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$\langle\,${\let\I=\ne#2\eightrm\kern.5em
    \ifacro{\pdfnote#1.}\else#1\fi}$\,\rangle$\XX}
\def\Y{\par\yskip}
\let\Z=\le
\let\ZZ=\let % now you can \write the control sequence \ZZ
\let\*=*

\def\Xand{\W} \def\Xandxeq{\MRL{{\AND}{\K}}} \def\Xbitand{\AND}
\def\Xbitor{\OR} \def\Xcompl{\CM} \def\Xnot{\R} \def\Xnotxeq{\I} \def\Xor{\V}
\def\Xorxeq{\MRL{{\OR}{\K}}} \def\Xxor{\XOR} \def\Xxorxeq{\MRL{{\XOR}{\K}}}

%\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

\newif\ifheader
\def\lheader{\headertrue\mainfont\the\pageno\eightrm\qquad\grouptitle
  \hfill\title\qquad\mainfont\topsecno} % top line on left-hand pages
\def\rheader{\headertrue\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\startpdf{\ifpdftex\pdfcatalog{/PageMode /UseOutlines}\else
    \ifpdf{\special{pdf: docview << /PageMode /UseOutlines >>}}\fi\fi}
\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, ##2.{\par\hangindent2em\noindent##1:\kern1em
    \ifacro\pdfnote##2.\else##2\fi.} % 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~\ifacro{\pdfnote##2.}\else{##2}\fi.}}
  \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\*=*
  \ifacro \def\outsecname{Names of the sections} \let\Xpdf\X
  \ifpdftex \makebookmarks \pdfdest name {NOS} fitb
    \pdfoutline goto name {NOS} count -\secno {\outsecname}
    \def\X##1:##2\X{\Xpdf##1:##2\X \firstsecno##1.%
      {\toksF={}\makeoutlinetoks##2\outlinedone\outlinedone}%
      \pdfoutline goto num \the\toksA \expandafter{\the\toksE}}
  \else\ifpdf
    \special{pdf: outline -1 << /Title (\outsecname)
      /Dest [ @thispage /FitH @ypos ] >>}
    \def\X##1:##2\X{\Xpdf##1:##2\X \firstsecno##1.%
      {\toksF={}\makeoutlinetoks##2\outlinedone\outlinedone}%
      \special{pdf: outline 0 << /Title (\the\toksE)
        /A << /S /GoTo /D (\romannumeral\the\toksA) >> >>}}
  \fi\fi\fi
  \readsections}
\def\makebookmarks{\let\ZZ=\writebookmarkline \readcontents\relax}
\def\expnumber#1{\expandafter\ifx\csname#1\endcsname\relax 0%
  \else \csname#1\endcsname \fi} % Petr Olsak's macros from texinfo.tex
\def\advancenumber#1{\countA=\expnumber{#1}\relax \advance\countA by1
  \expandafter\xdef\csname#1\endcsname{\the\countA}}
\def\writebookmarkline#1#2#3#4#5{{%
```

```
  \let\(=\let \let\)=\let \let\[=\let \let\]=\let \let\/=\let
  \pdfoutline goto num #3 count -\expnumber{chunk#2.#3} {#5}}}
\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 \startpdf
  \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#5{\ifnum#2=0 \smallbreak\fi
    \line{\consetup{#2}#1
      \rm\leaders\hbox to .5em{.\hfil}\hfil
      \ \ifacro\pdflink{#3}{\romannumeral#3}\else#3\fi\hbox to3em{\hss#4}}}
\def\consetup#1{\ifcase#1 \bf % depth -1 (@**)
  \or % depth 0 (@*)
  \or \hskip2em % depth 1 (@*1)
  \or \hskip4em \or \hskip6em \or \hskip8em \or \hskip10em % depth 2,3,4,5
  \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
```

부록 C: CWEB 매크로 사용법

cwebmac 파일에 정의 되어있는 매크로들 이용하면 CWEAVE의 결과물인 TEX 코드들을 직접 수정하지 않고도 여러 다양한 형태의 문서를 만들 수 있다.부록 C 에서는 그러한 매크로들의 사용법을 설명한다.

1. CWEB 매크로는 PLAIN TEX이 사용하는 기본 글꼴 외에 네가지의 글꼴을 더 제공한다. 중간 크기의 알파벳 대문자체인 UNIX를 얻고자 한다면, '{\mc UNIX}' 라고 하면 되고, 작은 크기의 알파벳 대문자체인 STUFF를 얻고 싶으면 '{\sc STUFF}' 라고 하면 된다. 그리고 문서의 제목 같은 다소 큰 글꼴을 얻기 위해서는 \titlefont 나 \ttitlefont 를 이용하면 되는데, 후자는 타자 글꼴로 나타낼 때 사용한다. 그 외에 \UNIX/ 와 \CEE/ 같은 매크로가 있는데, 이들은 중간 크기의 대문자체인 UNIX 와 C를 나타낸다.

2. TEX 텍스트에서 C 프로그램에서 사용되는 식별자를 언급할 때는 앞에서 배운 바와 같이 '|identifier|'를 이용한다. 그런데 그 방법 말고 '\\{identifier}'로 하는 방법이 더 있다. 이 두 경우 출력은 동일하다. 그런데 후자의 경우는 *identifier*를 색인에 넣지 않는다. 왜냐하면 위와 같은 표기는 CWEAVE가 C 모드를 다룰 때 고려해야 할 대상에서 제외되기 때문이다. 또한 후자의 경우, 식별자에 밑줄(underline)이 사용될 때는 반드시 그 앞에 백슬래쉬를 넣어야 한다. 이는 전자와 같이 할때는 CWEAVE가 다 알아서 해주기 때문에 필요없는 작업이다.

3. 'CWEB'과 같은 타자 글꼴을 이용하고자 한다면, 매크로 '\.'를 이용하면 된다. (예를 들어, '\.{CWEB}'). 이 매크로의 인자로 CWEAVE이 '특수 문자열'이라고 알고 있는 리스트에 속하는 기호들 앞에는 백슬래시를 하나 더 넣어야 한다. 즉, 그 리스트에 속하는 기호로는 백슬래쉬나 달러 기호와 같은 기호이 있다. '\␣'는 눈에 보이는 공백 문자를 나타낸다. TEX 명령어 다음에 나오는 공백 문자를 앞의 경우와 달리 눈에 보이지 않게 하려면, '{␣}' 라고 해야한다. 만일 표현하고자 하는 문자열이 길다면, '\)' 를 이용해서 그 문자열을 둘로 나눌 수 있는데, 이때 TEX이 그 문자열에서 줄 바꿈을 하고자 할 경우, 후자는 discretionary 백슬래쉬를 나타낸다.

4. 출력 페이지의 크기를 변경하고자 한다면, 작성하는 CWEB 파일의 in limbo에 다음과 같은 명령어들을 재정의 하면 된다. \pagewidth, \pageheight, \fullpageheight. 기본 값은 다음과 같다. 즉, 지금 보고 있는 바로 이 문서의 규격에 해당하는 값이다.

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

\fullpageheight의 값은 \pageheight의 값에 각 페이지의 상단에 있는 헤드라인과 페이지 번호를 위한 공간을 더한 값이다. 위의 값들을 변경한 다음에는 곧바로 매크로 \setpage를 반드시 호출해야 한다.

5. 매크로 \pageshift는 오른쪽 페이지들(즉, 페이지 번호가 홀수인 페이지들)이 왼쪽 페이지들(즉, 페이지 번호가 짝수인 페이지들)에 비해서 오른쪽으로 얼마만큼 이동되었는지를 나타내는 값이다. 이 값을 조정하면 양면 인쇄를 할 때에 보기좋은 적절한 페이지들을 얻을 수 있다.

6. 매크로 \title은 이 문서의 제목을 나타내는데, 매 페이지 상단에 작은 대문자체로 나타난다. 그리고 이 매크로를 재정의 하지 않으면, 작성하고 있는 CWEB 파일의 이름이 나온다.

7. 대개의 경우 첫번째 페이지의 페이지 번호는 1이다. 그런데 첫번째 페이지의 번호를 16으로하고, 목차 페이지의 번호를 15로 하고 싶으면, 다음과 같이 하면 된다. '\def\contentspagenumber{15} \pageno=\contentspagenumber \advance\pageno by 1'.

8. 다른 문서를 보더라도 일반적으로, 타이틀 페이지에는 헤드라인이 없다. 그러한 목적으로 CWEB 문서에서 어떤 페이지를 타이틀 페이지라고 지정하는 방법은 매크로 \iftitle을 '\titletrue'로 정의하는 것이다. CWEB의 모든 페이지의 \iftitle의 값은 기본적으로 \titlefalse 이다. 당연히 페이지 번호가 붙지 않는 타이틀 페이지는 예외로 \titletrue 이다.

타이틀 페이지는 \topofcontents, \botofcontents라는 두 개의 매크로를 재정의해서 원하는 대로 수정할 수 있다. 매크로 \topofcontents는 목차 정보가 로드 되기 바로 직전에 호출되고, 매크로 \botofcontents는 바로 다음에

호출된다. 위 두 매크로의 전형적 예는 다음과 같다.

```
\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 3.64)} \vfill}
```

위와 같은 경우에 목차 페이지의 상단에 원하는 정보를 나타낼 때는 오른쪽 페이지들의 헤드라인을 결정하는 `\rheader` 매크로를 재정의 하면 된다.

9. 목차를 만들기 위한 자료는 색인들이 TEX으로 처리된 후에 읽혀지는 파일에 기록된다. 그 파일의 한줄 한줄은 각각 목차에 표시될 별표 마디의 정보를 가지고 있다. 파일 `common.toc`는 예를 들어 다음과 같은 것들을 담고 있다.

```
\ZZ {Introduction}{0}{1}{28}{}
\ZZ {The character set}{2}{5}{29}{}
```

매크로 `\topofcontents`는 `\ZZ`를 원하는 형태로 출력하기 위해서 재정의 할 수 있다. (아래의 항목 19를 살펴보라.)

10. `CWEAVE`의 결과 파일인 TEX 파일을, 개별적으로 처리 가능한 여러 개의 서브 파일로 나눠야 할 경우가 종종 있다. 예를 들어, TEX 파일 소스가 500 페이지가 넘어서, 컴퓨터의 메모리나 기타 시스템의 자원 제한 때문에 그것을 한 번에 컴파일 하기에는 역부족인 경우가 있다. 크기가 엄청나게 큼에도 불구하고, TEX 파일을 나누지 않고 그냥 작업하다가는 TEX이 더이상 도저히 작업을 진행할 수 없는 경우가 생긴다. 이러한 경우에 처음으로 되돌아가서 그 파일을 여러 개의 작은 파일로 나누고 작업을 다시 시작해야 하는데, 이와 같은 삽질을 막기 위해서라도 파일을 나누어도 그 결과물은 하나의 파일에서 얻은 결과물과 같게 하는 방법을 알고 있어야 한다.

그 방법을 설명하기 위해 예를 들어 세개의 파일 α, β, γ로 나눈다고 해보자. 여기서 각 파일은 별표 마디로 시작한다. 문서의 형태와 모양 결정하는 모든 매크로는 α의 in limbo에 정의되어 있어야 하고, 그러한 TEX 코드들은 β, γ 파일을 첫부분에 마찬가지로 복사되어 있어야 한다. 일단 각 파일을 개별적으로 처리하기 위해서 두 가지를 고려해야 한다. 우선은 β와 γ 파일의 시작 페이지 번호를 적절히 정해야 하고, 그다음에 이 세개의 파일로 최종적으로 만들어질 문서의 목차는 이 세개의 파일이 가지고 있는 정보를 누적해서 모두 담고 있어야 한다.

`cwebmac` 파일에 정의되어 있는 매크로들 중에는 `\contentsfile`과 `\readcontents`라는 두개의 매크로가 있는데, 이 매크로들은 위에서 방금전에 언급한 두 가지 고려사항을 해결하는데 필요한 매크로들이다. 우선 파일 α의 in limbo에 '`\def\contentsfile{cont1}`'라고 입력하고, 파일 β의 in limbo에는 '`\def\contentsfile{cont2}`'라고 입력한다. 이렇게 하면, TEX은 파일 α와 β의 목차에 관한 내용을 파일 `cont1.tex`과 `cont2.tex`에 기록한다. 이제 마지막 파일인 γ 파일에 다음과 같이 하면 그만이다.

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

위와 같이 하면, 세 파일의 모든 자료를 올바른 순서로 가져오게 된다.

하지만, 위와 같이 하더라도 아직도 페이지 번호 붙이기 문제는 해결되지 않는다. 이 문제를 해결하는 한가지 방법은 아래와 같은 내용을 β 파일의 in limbo에 입력하는 것이다.

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

그리고나서 TEX 컴파일을 할때, TEX이 물어보는 질문에 간단히 대답만 하면 된다. 이와 비슷한 방법을 파일 γ의 시작부에도 하면 된다.

물론, 위와 같은 방법을 이용하면 하나의 파일을 세개가 아닌 여러개의 임의의 갯수로 나눌 수 있다.

11. 때때로 색인에 들어가는 아이템을 일반적인 로마체나 타자글꼴로 표시하는 것이 아니라 특별한 방법으로 표시하는 것이 보기 좋을 때가 있다. 만약 색인에 넣고자하는 아이템이 식별자나 예약어가 아니라면, 앞에서 `CWEB`의 명령어에 대해서 알아볼 때 설명한 명령어를 이용하면 된다. '`@^`'는 색인에 들어갈 아이템을 로마체로 만들고, '`@.`'는 타자체로 만든다. 예를 들어, 색인에 'TEX'을 넣고자 한다고 하자. 'TEX'을 로마체로 색인에 넣고자 하여 '`@^\TeX@>`'라고 했다면, 백슬래쉬 문자가 방해가 되어 TEX은 색인 T에 나오지 않는다.

해결책은 다음과 같은 방법으로 단순히 정렬키를 제거하는 매크로 '@:'의 특징을 이용하는 것이다.

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

이제, CWEB 파일 내에 '@:TeX}{\TeX@>'라고 하면, CWEAVE는 이것을 정렬키에 기반한 알파벳 순으로 색인에 넣을 것이다. 그리고 정렬키는 출력되지 않는 매크로 호출 '\9{TeX}{\TeX}'을 실행할 것이다.

위와 비슷한 아이디어로 보이지 않는 객체를 마디 이름에 넣기 위해서 사용할 수 있다. 어떤 이들은 이러한 트릭을 "special refinements"라고 부르고, 다른 사람들은 모두 "kludges"라고 부른다.

12. TEX 명령어 \secno는 페이지 번호를 설정하는데 사용된다.

13. 변경된 마디들만을 색인과 함께 나열하고 싶으면, in limbo에 '\let\maybe=\iffalse'와 같은 명령을 입력하라. 이와 같은 작업을 수정 파일의 서두에서 하는 것이 대부분이다.

그러나 이 특징은 TEXnical한 한계를 가지고 있다. 위 명령을 \proclaim 또는 \+ 또는 \newcount와 같은 플레인 TEX에 '\outer'로 정의된 명령어와는 함께 사용할 수 없다. 왜냐하면, TEX은 그러한 명령어들을 조용히 넘어가려고 하지 않기 때문이다. 이러한 한계를 이겨내는 방법은 다음과 같이 하는 것이다.

```
\fi \let\proclaim\relax \def\proclaim{...} \ifon
```

여기서 \proclaim은 원래의 그것과 동일한 기능을 하지만 \outer를 띄어버린 것으로 재정의된 것이다. (\fi는 조건 분기를 멈추게 하는 것이고, \ifon는 다시 그것을 가능하게 하는 것이다.) 비슷하게, 다음 문장은

```
\fi \newcount\n \ifon
```

\newcount를 사용하는 안전한 방법이다. 플레인 TEX은 이미 비-outer 매크로 \tabalign를 제공하는데, 이 매크로는 \+와 같은 일을 하는 매크로이다. 그래서 아래와 같은 명령어를 이용하면,

```
\fi \let\+\tabalign \ifon
```

\tabalign보다 짧은 \+를 이용할 수 있다.

14. 영어가 아닌 다른 언어로 CWEB 프로그램을 작성할 때는 다음과 같은 매크로들을 재정의 하면된다. \A, \As, \ATH, \ET, \ETs, \Q, \Qs, \U, \Us, \ch, \fin, \con, \today, \datethis, \datecontentspage. CWEAVE를 직접 수정할 필요가 없는 것이다.

15. 매크로 \noatl, \noinx, \nosecs, \nocon들을 이용하면, 출력 문서에서 목차와 같은 일부분을 선택적으로 생략 할 수 있다. 즉 문서의 크기가 매우 작아서 목차가 필요없겠다고 판단되면 작성하는 CWEB 파일의 in limbo에 \nocon이라고 입력하면 된다.

16. 플레인 TEX 포멧에 정의된 모든 액센트(accents)와 특수 기호들은 *The TEXbook*의 제9장에 설명된대로 CWEB 문서에서도 그대로 동작한다. 단, 한가지 예외가 있다. 보통 TEX에서 \.로 표시하는 닷 엑센트(dot accent)는 CWEB에서는 반드시 \:로 입력해야 한다. CWEB은 이미 \.을 다른 용도로 사용하고 있다.

17. cwebmac.tex 파일을 잘 살펴보면 에는 몇몇 매크로나 명령어들이 주석 처리되어 있는 것을 발견할 수 있는데, 당신의 취향에 따라서 그 주석들을 풀어서 그 매크로의 기능을 이용할 수 있다. 예를 들어, 주석 처리된 것들 중 어느 하나를 풀면 빈 페이지를 만들 수 있는데, 이는 duplex 프린터를 사용하고 있다면 유용한 기능이 된다. 이 사용자 설명서의 완전한 번전은 부록 D, E, F가 있는데, 그 부록의 소스 리스트에서는 대입 연산자로 '=' 대신에 '←'를 사용한다. 그 부록들을 보고 취향에 맞는 어떤 주석을 풀어야 할지 말지를 결정해서 사용하면 된다.

18. Andreas Scherer는 \pdfURL라는 매크로를 작성해 주었는데, 이 매크로는 CWEB 파일의 TEX부 또는 C 주석문의 어느 곳에서도 다음과 같은 방법으로 사용이 가능하다.

```
다음의 주소 \pdfURL{the author}{mailto:andreas.scherer@@pobox.com}로 이메일을 보내거나,
웹사이트 \pdfURL{his home page}{http://www.pobox.com/\TILDE/scherer}를 방문하길 바란다.
```

PDF 문서에서, 이 매크로의 첫번째 인자는 클릭이 가능한 파란색 글씨로 나타난다. 이 매크로를 올바르게 사용했다면, Acrobat reader는 사용자의 웹브라우저나 이메일 클라이언트를 통해서 링크된 쪽으로 리다이렉트 시켜서 해당 페이지

를 연다. PDF가 아닌 하드카피에서 이 매크로를 사용하면, 문서에는 두 인자가 동시에 출력된다. (`the second in parentheses and typewriter type`). 인터넷 주소에 사용되는 일부 특별한 문자는 다소 이상한 방법으로 사용되어야 하는데, 이는 `CWEAVE` 와/또는 TEX의 명령어와 혼동하지 않기 위함이다. `@`를 사용할 자리에는 `@@`를 쓰고, `~` 대신에 `\TILDE/`를 써야하고, `_` 대신에 `\UNDER/`를 사용해야 한다.

19. PDF 문서는 목차에 사용된 모든 별표 마디의 제목에 대한 책갈피(bookmarks)를 가지고 있고, 명령어`@*`의 깊이 특성을 이용하면 앞에서 명령어를 공부할 때 살펴본 바와 같이, 어떤 것은 다른 것들의 소제목이 되기도 한다. 그러한 책갈피 요소들을 "outlines"라고 한다. 책갈피의 마지막 그룹 제목은 'Names of the sections'인데, 이 그룹은 클릭하면 모두 펼쳐져서 프로그램에 사용된 모든 마디 이름을 볼 수 있다. 따라서 Acrobat 사용자는 이 기능을 통해서 원하는 마디를 쉽게 찾아 볼 수 있다.

`cwebmac.tex`의 매크로들은 책갈피에 사용된 모든 이름들에서 PDF의 outline의 제한된 기능때문에 outline에 사용되기에는 부적합한 특수 문자나 명령어나 매크로같은 쓸데 없는 코드를 조심스럽게 순화하기 위해서 "제거한다(sanitize)." 예를 들어, `CWEAVE`의 어떤 마디 이름이 'Cases for *case_like*'인데, 이는 TEX 코드로 `cweave.tex` 파일에서 '`Cases for \PB{\\{case\_like}}`'로 나타나는데, 이 이름을 outline을 위해서 방해되는 것을 제거하여 순화하면 단순히 '`Cases for case_like`'가 된다. (`.pdf` 파일들이 만들어질때, `.toc` 파일 안에 있는 모든 `\ZZ` 매크로의 다섯번째 인자는 첫번째 파라미터의 순화된 버전이다. 앞서 나온 항목 9와 뒤에 나올 항목 20을 참고하라.)

일반적으로, outline을 위한 순화는 `CWEB`가 직접 정의한 명령어들을 제외한 TEX의 명령어와 괄호들을 제거한다. 대부분 경우 이러한 제거 과정은 순조롭게 이루어지지만, TEXnical한 속임수를 사용하면, 당신은 이러한 기본 제거 과정을 당신이 원하는 당신이 정한 규칙대로 제거할 수 있다. 예를 들어, 아래와 같은 문장이 적용되면,

```
\sanitizecommand\foo{bar}
```

명령어 `\foo`는 '`bar`'로 바뀐다. 그리고 다음과 같은 문장이 적용되면,

```
\def\kluj#1\\{foo}
```

TEX 코드 '`\kluj bar\\`'는 'foo'로 출력되지만 책갈피에는 '`bar`'로 출력된다. 왜냐하면 명령어 `\kluj`와 `\\`는 제거 과정에서 사라지기 때문이다.

20. 게다가, 그룹 제목들은 `\ifheader` 매크로를 사용해서 헤드라인에서도 바뀌어야 한다면 임의의 순화된 텍스트로 바뀔 수 있다. 예를 들어, 아래와 같은 두줄로 시작하는 `CWEB` 소스 파일을 생각해보자.

```
\def\klujj#1\\{\ifheader FOO\else foo\fi}
@*Chinese \klujj bar\.
```

이 코드는 헤드라인에는 'CHINESE FOO'로 나타나고 목차에는 'Chinese foo'로 나타나는 그룹 제목이 '**Chinese foo**'를 만든다. 그러나 이에 해당하는 책갈피는 '`Chinese bar`'가 된다. 그리고 해당하는 `.toc` 파일에 나오는 항목은 '`\ZZ {Chinese \klujj bar\\}{1}{1}{1}{Chinese bar}`'이다.