Ruweb, сын Nuweb

Н.Е. Известный

Много-много лун назад Великий Программист всех времен и народов Дональд Кнут сказал: "А не слабо ли нам, братья, писать программы так, что их потом читать можно было ?"

Так была рождена концепция "literate programming", которую, как ни крути, перевести не удается. Поэтому оставим эти попытки и изложим идею. Идей точнее две:

1. писать код как документ;
2. зафиксировать процесс построения кода сверху-вниз.

Вторая идея с позиций практикующего программиста мне даже больше нравится. Но и первая изрядно облегчает жизнь, позволяя в любой момент времени работы над программой или ее модификации взмахнуть волшебной палочкой и получить идеально синхронизированный с текущим состоянием программы документ, отпечатанный со всем шиком и стилем, присущим TEX, описывающий все "ins and outs" программы, идеи, лежащие в ее основе, результаты тестов, историю создания и много всего.

И сказали братья, что это хорошо и пошли создавать разные пилы и лопаты, чтобы все программы были литературны. И сам ДК написал `weave` и `tangle` — первая из которых "скручивает" все, что написано, в окончательный текст программы, который обычно написан на каком-либо языке типа C или Pascal. Этот текст фактически не предназначен для человеческого глаза, поэтому лишен обычной икебаны — комментариев, отступов и пр. Его предназначение — быть проглоченным компилятором и появиться на другом конце в виде исполняемого кода.

Вторая программа (`tangle`) "прядет" из начального документа TEX-овский текст, который, с помощью черной и белой магии TEX возникает перед вами, как белый лебедь, пластилиновая ворона или Василиса Премудрая, кого на что хватит, и в котором все наилучшим образом отформатировано, преобразовано, отиндексировано, разложено по разделам и подраздельчикам. А если уж там и написана какая глупость, так это ваша проблема.

Много братьев намастерили кому-какие нравятся пилы и лопаты для всякой литературной кнутовщины или кнутовской литературщины, но больше всего мне понравился скребок под именем `nuweb`.

У него следующие преимущества:

- работает с любым языком, не навязывает какой-либо стиль написания кода;
- содержит минимальный набор команд, который тем не менее делает почти все, что нужно;
- доступен в исходном коде и автор не возражал против того, чтобы в нем поковырялись. Автор даже не возражал против того, чтобы поменять ему имя на `ruweb`, чтобы отразить руссификацию.

Конечно, за каждым из достоинств скрывается свой Янь — `ruweb` понятия не имеет, с каким это языком вы имеете дело и не может поэтому оказать каких-либо языково-зависимых услуг. В частности, выделить идентификаторы и служебные слова, разукрасить комментарии или включить в них графику. За этим, если вошли во вкус, обращайтесь к другим, более мощным, `web`-средствам.

1 Как `ruweb`?

Чтобы `ruweb` что-то обработал, ему нужен исходный файл. Писать этот файл можно как обычный TEX-овский документ, используя в не-программной части все средства TEX.

Программу же `ruweb` описывает как состоящую из набора "блоков", которые представляют собой смеси операций, описанных на текущем языке программирования и указаний на включение других блоков.

Начинать можно с весьма общего описания структуры программы, например, программа построчной обработки зарплатной ведомости может быть на самом общем уровне описана как

```
----- Начало ruweb-файла --------------
\documentclass{article}
\usepackage[koi8-r]{inputenc}
\usepackage[russian]{babel}
\usepackage{latexsym}
\title{Система расчета зарплат}
\date{}
\author{Н.Е. Известный}
\begin{document}
\maketitle

Эта замечательная программа анализирует ведомость
на зарплату и проверяет, кто слишком много заработал.

@d Программа "Зарплатка" @{
int main(argc, argv)
int argc;
char argv[];
{
        @< Откуда-то получаем данные @>
        @< Как-то их обрабатываем @>
        @< Куда-то их посылаем @>
}
@}
\end{document}

----- Конец ruweb-файла --------------
```

Видно, что в основном это достаточно обычный LaTeX-файл с некоторыми вставками, ограниченными `@`. Конечно, после обработаки `ruweb`-ом чудес не произойдет и единственное, что вы в конце концов получите — это суррогат текста программы

```
int main(argc, argv)
int argc;
char argv[];
{
}
```

но `ruweb` по крайней мере вам ничего и не напортил.

Добавив в `ruweb`-файл описание блока обработки

```
Читаем файл-ведомость зарплаты и обрабатываем его
по-строчно.
@d Как-то их обрабатываем @{
for ( int i = 0, itogo = 0; i <= 10000; i++ ) {
        @< Читаем строку ведомости: ФИО, дата, сумма прописью @>
        @< Проверяем сумму S @>
        itogo += S;
@}
```

получаем слегка более содержательную программу, где уже виден результат автоматической сборки:

```
int main(argc, argv)
int argc;
char argv[];
{
        for ( int i = 0, itogo = 0; i <= 10000; i++ ) {
                itogo += S;
                }
}
```

Система расчета зарплат

Н.Е. Известный

Эта замечательная программа анализирует ведомость на зарплату и проверяет, кто слишком много заработал.

⟨Программа "Зарплатка"?⟩ ≡

```
int main(argc, argv)
int argc;
char argv[];
{
        ⟨Откуда-то получаем данные ?⟩
        ⟨Как-то их обрабатываем ?⟩
        ⟨Куда-то их посылаем ?⟩
}
◇
```

Ссылка на макрос в схеме ?.

Читаем файл-ведомость зарплаты и обрабатываем его по-строчно.

⟨Как-то их обрабатываем ?⟩ ≡

```
for ( int i = 0, itogo = 0; i <= 10000; i++ ) {
        ⟨Читаем строку ведомости: ФИО, дата, сумма прописью ?⟩
        ⟨Проверяем сумму S ?⟩
        itogo += S;
◇
```

Ссылка на макрос в схеме ?.

"prog.c" ? ≡
⟨Программа "Зарплатка"?⟩◇

Рис. 1: Результат обработки TEX

Продолжая подобным образом развивать программу, описывая блоки вплоть до уровня уже исключительно команд языка программирования, получим зафиксированный в `ruweb`-файле процесс построения программы сверху-вниз, где логика и детали реализации будут видны гораздо лучше, чем при традиционной технологии. Для того, чтобы получить документ на программу такж достаточно просто обработать `ruweb`-файл программой `ruweb`. Если специально не отказываться, то в результате обработки создасться и специальный TEX-файл, который не обязательно рассматривать, но после обработки LATEX-ом он создаст документ, показанный на рис. 1.

Выше мы продемонстрировали лишь пару комманд `ruweb` — настало время описать их полностью ! К счастью, их всего 4 :)

2 Основные команды

Все команды `ruweb` предваряются символом `@`, если необходимо поместить этот символ в выходной файл, создаваемый `ruweb`-ом, необходимо поместить в файл удвоенный символ `@@`.

2.1 Определение блока

Основной командой `ruweb` является `d` — определение блока.

`@d` ***Имя блока*** **Блок** Определяет Блок с именем *Имя блока*.

Тело Блока ограничивается символами `@{` и `@}`. Определение Блока может быть разделено между несколькими командами `d`, в этом случае различные части определения Блока сливаются вместе в том порядке, как они помещены в `ruweb`-файле.

Таким образом блок имеет общий общий вид `@{` *Все-что-угодно* `@}` где тело Блока содержит буквально каждый символ во *Все-что-угодно* — все пробелы, все символы табуляции, все символы перехода на новую строку.

Внутри блока может содержаться инструкции по включению других блоков.

`@<`*Имя-блока-2*`@>` — вставляет блок *Имя-блока-2* во внешний блок в тот момент когда код пишется в файл.

Имена блоков можно сокращать как в момент определения, так и в момент использования. Так, например, имя блока

```
@d Читаем строку ведомости: ФИО, дата, сумма прописью
```

можно сократить до

```
@d Читаем...
```

При этом достаточно привести минимум символов, необходимых для определенной идентификации блока.

2.2 Вывод под флаги

Команда

`@o` *Имя-файла* `@` *Блок* `@` — производит вывод блока *Блок* во внешний файл *Имя-файла*.

При этом осуществляются все вложенные подстановки и слияния частей блока описанных различными операторами `@d`. Команде `o` можно указать некоторые флаги, полезные при генерации файлов для языков со специальными требованиями к разметке исходного текста.

Этих флагов всего 3:

`-d` вставляет в генерируемый текст директивы `#line` привязывающие сообщения об ошибках при компиляции к строкам `ruweb`-файла, что упрощает отладку.

`-i` подавляет отступы блоков. Это означает, что при подстановке блока не будет осуществляться его отступ на размер отступа по месту включения.

`-t` подавляет замену символов табуляции на пробелы. Полезно, а точнее незаменимо, при генерации `makefile`-ов.

2.3 Вспомогательные команды

Существуют 2 низкоуровневые команды, которые можно использовать в любом месте:

`@@` помещает единственный символ `@` в выходной поток.

`@i` *имя-файла* включает по месту содержимое *имя-файла*. Строка с этой командой не должна содержать чего-либо другого. Включения могут быть вложены, но не слишком глубоко ($\leq$ 10).

Некоторую пользу приносят разного сорта индексы и `ruweb` способен создать индексы файлов, имен блоков и определенных идентификаторов, помеченных пользователем. Все это производится с помощью команд

`@f` Индекс имен файлов.

`@m` Индекс имен блоков.

`@u` Индекс идентификаторов.

Обычно эти индексы заполняют собой соответствующие разделы документа-описания программы.

Для того, чтобы отметить идентификатор с целью включения его в индекс `@u` необходимо перечислить его в конце описания блока. Например

```
@d Как-то их обрабатываем @{
@| itogo S}
```

пометит переменные `itogo` и `S` — все их вхождения будут далее обнаружены автоматически. Более того, отмечать можно не только идентификаторы программы, а любую последовательность непробельных символов типа 3.141592. Заметим, что как в приведенном примере `itogo` или `S` необязательно должны встречаться в том блоке, шде они метятся.

3 Ключ на старт!

Вызов `ruweb` производится следующей командой:

`ruweb` *опции имя-файла...*

Можно обрабатывать несколько файлов одним вызовом `ruweb`. Программа ожидает у файлов расширение `.w`, так что если оно отсутствует при вызове, то `.w` будет автоматически добавлено. Результирующие файлы создаются в текущей директории независимо от того, где расположен источник `ruweb`-файлов.

Довольно часто возникает ситуация, когда необходимо что-то одно: либо текст программы, либо документ. С целью небольшой экономии опции, управляющие `ruweb`, позволяют отказаться либо от генерации документа в первом случае, либо от генерации кода во втором.

`-t` подавляет генерацию (TEX)-документа.

`-o` подавляет генерацию выходных файлов.

Таким образом команда

```
nuweb -to foo
```

просто прочитает файл `foo.w` и не создаст никакого вывода, за исключением сообщений об `ruweb`-ошибках и предупреждениях.

Еще одна опция слегка ускоряет процесс обработки:

`-c` отказ от проверки выходных файлов на предмет сделанных изменений.

Есть еще дополнительные опции командной строки:

`-v` заставляет `ruweb` выводить по каналу `stderr` информацию р ходе обработки.

`-n` нумерует блоки последовательно, начиная от 1.

4 Еще одна причина для работы с `ruweb`

Поскольку `ruweb`-файл — это практически TEX, в этой ситуации оказываются применимы масса других TEXнических средств. В первую очередь стоит обратить внимание на пакет `rcs`, позволяющий включить в текст документации историю разработки программы, сохраняющейся в `rcs`-файле. В сочетании с различными `web`-ами это дает очень полезную комбинацию.