В.В. Текин

Обработка длинных строк

Жующая публика, как всегда, приняла показанные ей
элементы воздушного боя за фигуры высшего пилотажа,
выполняемые отчего-то на боевых машинах.

Один из малоизвестных эпизодов авиакосмического салона «МАКС-2007»

Каждый видит лишь то, что хочет видеть. Так же обстоят дела и с программным обеспечением — пресловутым софтом («мягкой рухлядью» преимущественно от Microsoft).

Текстовыми процессорами обычно называются инструментальные системы подготовки текстов. И казалось бы, результатом использования таких процессоров должны быть тексты, т.е. строки символов с кодами 00..FFh длиной от 0 до 255, завершающиеся комбинацией символов CR/LF возврата каретки CR (код 0Dh) и перевода строки LF (код 0Ah).

Не тут-то было. Отдельные строки изначально объединялись в абзацы, а сами абзацы выделялись абзацным отступом или/и разделяющей их пустой строкой. В результате получалось, что с момента своего появления текстовые процессоры работали с форматированными текстами, т.е. с документами. Другое дело, что формат таких документов был скрыт в их тексте.

В современных же текстовых процессорах формат документа, т.е. его шаблон, и составляющий документ текст разделены. Электронный образ (файл) такого документа состоит из описания его шаблона и собственно текста. Естественно, разбиения последнего на строки уже не требуется и за редактирующими спецификациями CR/LF сохраняется лишь функция указания границ абзацев.

Столь, казалось бы, незначительное изменение структуры документа коренным образом изменяет назначение процессора. Это уже не текстовый, а документный процессор, хотя последний термин еще не утвердился. И сегодня наиболее известным из таких процессоров безусловно является Word фирмы Microsoft.

Пока документные процессоры работают в символьном режиме, но активное использование графики обнаруживает задачу их перевода в графический режим. И если этот перевод состоится, то непонятно, во что выродится контекстный поиск графического объекта и как такой объект можно будет указать.

Но не будем забегать вперед и сосредоточимся лишь на форматах выгрузки документов популярной справочно-правовой системы (СПС) «КонсультантПлюс». Таких форматов разработчики последних версий СПС «КонсультантПлюс» предлагают три:

1) документ в формате RTF (*.rtf);

2) обычный текст без форматирования;

3) текст в формате UNICODE (*.txt).

Разработанный Microsoft формат RTF по существу является аналогом гораздо более удобного формата описания гипертекстов HTML. Однако вне программных продуктов Microsoft формат RTF не получил распространения, тогда как HTML прекрасно воспринимается и Word'ом. И остается только гадать, чем здесь руководствовались разработчики СПС «КонсультантПлюс», отдавая предпочтение чисто майкрософтовской экзотике RTF.

«Обычный текст без форматирования» представляет поток абзацев текста, разделенных символами CR/LF, в кодировке windows-1251. На размер абзаца в отличие от ранних версий СПС «КонсультантПлюс» ограничений здесь уже не накладывается. Этот формат успешно воспринимается текстовым процессором Word и может быть им преобразован в великое множество других форматов.

«Текст в формате UNICODE (*.txt)» отличается от «обычного текста без форматирования» использованием другого способа представления символов. Текст в этом случае состоит из заголовка-сигнатуры и последовательностей пар байтов, первый из которых (младший) указывает номер алфавита, а второй (старший) является порядковым номером символа в алфавите. В частности, английскому алфавиту присвоен номер 0, а русскому алфавиту соответствует номер 4. Какие еще алфавиты намерена поддерживать Microsoft, неизвестно. Кстати, в чистом виде «Текст в формате UNICODE (*.txt)» не воспринимается Word'ом, что опять же порождает вопросы к разработчикам СПС «КонсультантПлюс».

Проведенное сравнение показывает, что из возможных форматов выгрузки документов СПС «КонсультантПлюс» практически значимым является только «обычный текст без форматирования». С использованием Word'а этот (а также RTF) формат может быть легко преобразован в обычный «текст DOS с разбиением на строки», т.е. последовательность ASCII-строк в альтернативной кодировке. Однако пользоваться услугами всемогущей Microsoft здесь необязательно. Тем более что умение независимым от Microsoft образом работать с «обычным текстом без форматирования» пригодится и для многих других случаев.

Для решения поставленной задачи, во-первых, надо изменить используемую кодировку windows-1251 на альтернативную кодировку ASCII (или кодировку DOS, как ее предпочитает называть сама Microsoft). И во-вторых, абзацы надо разделить на строки. Разумеется, на каждом шаге ждут подвохи.

Так, при смене кодировки приходится учитывать различие алфавитов windows-1251 и ASCII. Если отдельные символы windows-1251 можно заменить последовательностями символов ASCII, то для других символов сделать это оказывается затруднительным. Например, знак авторского права (копирайт) можно заменить последовательностью символов «(c)». А вот чем заменить символы с умляутом (видом немецкой диакритики) или со знаками акцента (в виде кавычки сверху), далеко не ясно.

Возможно, что принятый в программе листинга 1 способ преобразования символов windows-1251 в альтернативную кодировку ASCII не является наилучшим, однако на текстах без экзотики, в частности на документах СПС «КонсультантПлюс», он вполне себя оправдывает. Соответствующие преобразования производятся здесь вызовом функции ConvertChar.

Данная и все последующие программы написаны на входном языке компилятора TurboPascal 3.01A.

Преобразованный текст сначала помещается в промежуточный файл. Этот файл переименовывается в выходной лишь при успешном завершении программы. При аварийном завершении возвращается код первой вызвавшей сбой ошибки, все открытые файлы закрываются, а промежуточный файл удаляется.

На время работы программы клавиатура блокируется.

Аналогичный образом производится обработка ошибок и во всех остальных программах.

Входной файл не может быть открыт, если он не существует либо имеет атрибуты R/O. Промежуточный файл не может быть создан, если он совпадает с входным. Ошибка переименования промежуточного файла в выходной возникает, когда выходной файл существует и имеет атрибут R/O. Способ разрешения возникающих конфликтов очевиден.

Проблема разбиения абзацев на строки связана с тем, что в выгружаемых из СПС «КонсультантПлюс» текстах встречаются таблицы, выровненные с помощью последовательностей обычных пробелов. А так как наибольшее значение ширины таких таблиц заранее неизвестно, максимальный размер строки целесообразно брать близким к предельно допустимому. Полученный таким образом текст даже простейшие текстовые процессоры под DOS вроде давно забытого Lexicon'а Евгения Веселова (удивительно напоминающего Word младших версий) позволяют без всяких проблем привести в удобочитаемый вид.

К сожалению, выбрать максимальную длину строки равной 255 символам тоже нельзя, ибо другие текстовые процессоры, как, например, по-своему замечательный LUX С.А. Назаренко, некорректно обрабатывают абзацы строк длиной более 230 символов.

Естественно, максимальная длина строки не может быть и меньше размера абзацного отступа. Причем во всех случаях перед ее записью имеет смысл удалять «хвостовые» пробелы (именно так и поступает все тот же LUX С.А. Назаренко).

Наконец, большой необходимости сохранять «связующие» («неразрывные») пробелы с кодом 00h нет, так как такие пробелы чаще всего являются плодом нездорового воображения авторов документов и почти всегда могут быть заменены обычными пробелами. Например, так поступает все тот же Word.

С учетом указанных ограничений подробный алгоритм разделения абзацев на строки можно представить следующим образом.

1. Начало. Открываем входной и выходной файлы.

2. При обнаружении конца входного файла уходим на 17.

3. Читаем входной файл, игнорируя символы 00h («связующий» пробел), 13h (возврат каретки CR), 20h (обычный пробел).

4. Если встретился символ, отличный от символа перевода строки LF (10h), уходим на 6.

5. Записываем в выходной файл пустую строку и уходим на 3.

6. Начало первой строки абзаца: Line := отступ + символ.

7. При исчерпании файла ввода записываем в выходной файл строку Line и уходим на 17.

8. Читаем очередной символ из файла ввода.

9. Если получен возврат каретки CR (13h), возвращаемся на 7.

10. Если встретился перевод строки LF (код 10h), пишем строку Line в выходной файл и возвращаемся к 2.

11. Если встретился «связующий» пробел (00h), заменяем его обычным (20h).

12. При длине строки Line меньше допустимой записываем символ в конец строки: Line := Line + символ и возвращаемся к 7.

13. Выделяем из строки Line максимальную подстроку SubLine, ограниченную первым пробелом в Line справа. Из строки Line удаляем подстроку SubLine, затем в подстроке SubLine удаляем «хвостовые» пробелы (возможно, что строка Line, или подстрока SubLine, или они обе после этого будут пусты).

14. Если пуста SubLine, переходим к 16.

15. Если пришли сюда, то длина строки Line строго меньше допустимой. Пишем в выходной файл SubLine, дописываем введенный символ в конец строки: Line:= Line + символ и возвращаемся к 7.

16. «Режем» слишком длинную строку Line, записывая ее текущее значение в выходной файл, после чего полагаем Line := символ и возвращаемся к 7.

17. Закрываем все файлы. Конец.

Предполагается, что операции открытия/закрытия файлов и ввода/вывода включают в себя и механизм обработки возможных состояний доступа.

Как видно из приведенного алгоритма, использование в качестве разделителей абзацев сдвоенных кодовых комбинаций CR/LF (0Dh/0Ah) совершенно излишне. Вполне достаточно и одного из этих символов, в качестве которого здесь выбран символ перевода строки LF (0Ah), что вполне соответствует концепции, принятой в UNIX.

И хотя приведенный алгоритм выглядит несколько запутанным, тем не менее он допускает простую программную реализацию. Причем в данном случае нет смысла отказываться от нелюбимых поклонниками высокого стиля программирования злосчастных GoTo — код получается короче и проще.

В листинге 2 приведен исходный текст программы для разделения абзацев на строки, в которой реализован только что описанный алгоритм. Способ обработки ошибок ввода/вывода здесь целиком заимствован из программы листинга 1.

Преобразование кодировки символов можно совместить и с форматированием абзацев без внесения радикальных изменений в строй программы листинга 2. Правда, при этом придется ограничиться лишь простой заменой: символ символ. Но именно так поступает и хваленый Word. Исключение составляют лишь знаки многоточия, торговой марки и копирайта, которые Word преобразует следующим образом:

1) многоточие заменяется последовательностью символов «...»;

2) знак торговой марки заменяется строкой «(tm)»;

3) знак авторского права (копирайт) записывается как «(c)».

Остальные не имеющие ASCII-аналогов символы Word заменяет знаком вопроса «?».

По необходимости знаком вопроса «?» программа листинга 3 заменяет и перечисленные три знака. Последнее, конечно, является ограничением, но зато позволяет избежать коренной ломки программы листинга 2, простой модификацией которой является программа листинга 3. Отличие только в том, что программа листинга 3 каждый извлекаемый из входного потока символ предварительно преобразует в другой символ с учетом указанных ограничений. Иными словами, разделение абзацев на строки и перекодировка символов выполняется здесь совместно.

Конечно, трудно предположить, до чего могут додуматься законодатели в своем законотворческом раже, но пока что в документах СПС «КонсультантПлюс» встречать многоточие, знак торговой марки и копирайт не приходилось. Еще раз отметим, что Word знак «связующего» пробела в windows-1251 с кодом A0h заменяет (назло врагам отечества) обычным пробелом с кодом 20h.

Между тем снять все ограничения на преобразование символов большой проблемы также не представляет. Достаточно считать, что символы будущих строк поступают не из входного файла, как предполагалось ранее, а из некоторого буфера. При этом очередной символ извлекается из буфера, если тот не пуст. В противном случае символ сначала считывается из входного файла, а затем заменяется помещаемой в буфер строкой из одного или более символов в соответствии с таблицей перекодировки.

Естественно, что таблица перекодировки состоит уже из строк переменной длины. В частности, входному символу многоточия с кодом 85h соответствует последовательность символов «...» из трех точек «.», а для символа копирайта с кодом A9h такая последовательность имеет вид «(c)». Прочие символы, не имеющие ASCII-представления, заменяются цепочками из одного или более символов ASCII, визуально воспроизводящими начертание символа windows-1251.

Программа однопроходного полнофункционального конвертера приведена в листинге 4. Таблица замен символов, не имеющих ASCII-представления, здесь в точности соответствует использованной в программе листинга 1.

Проверка предложенных программ на отдельных документах из СПС «КонсультантПлюс» с «обычным текстом без форматирования» (в том числе и содержащих таблицы) дала полностью совпадающие результаты.

Заметим, что для программы листинга 2 кодировка символов достаточно безразлична, поскольку роль «связующего» (00h) и разделяющего (20h) пробелов, а также символов возврата каретки CR (0Dh) и перевода строки LF (0Ah) давно устоялась и совпадает во всех известных кодировочных таблицах DOS.

Наверное, надо пояснить, почему во всех программах файл, используемый для ввода только отдельных символов, объявлен как текстовый вместо, казалось бы, более естественного символьного. Ответ прост. Во-первых, потому, что текстовый файл является более универсальным. Во-вторых, потому, что при текстовом вводе всегда используется буфер, размерами которого к тому же можно управлять. В-третьих, потому, что блокирование/разблокирование записей в текстовых файлах (по крайней мере Turbo Pascal 3.01A) действует чрезвычайно эффективно. И если первое из обстоятельств почти никакого значения не имеет, то два последних дают заметный выигрыш в скорости доступа к отдельным символам текстового файла по сравнению с файлом символьным.

В заключение хотелось бы высказать одно наблюдение. На мой взгляд, грамотно описанный алгоритм куда ценнее своей программной реализации. И все потому, что выразительность живого языка явно несравнима с выразительностью языка искусственного.

Как тут не вспомнить, что ровно четверть века назад один толковый преподаватель (и крупный специалист по базам данных) учил нас программированию на... русском языке. Вернее, не самому программированию, а полному и точному описанию методов решения задач. Его наставлениям я и старался следовать при написании 17 пунктов приведенного выше алгоритма.

Приложение 1. Листинги

Приложение 2. Программа для конвертации текстов из кодировки ASCII в windows-1251 и обратно.

Примечание. Программа запускается из командной строки с указанием входного и выходного файлов:

```
Win3.com <входной файл> <выходной файл>.
```

После этого необходимо выбрать нужное направление конвертации.