БОЧАРОВ Б.П. **ВОЕВОДИНА М.Ю.**

ФОРМИРОВАНИЕ ОТЧЕТОВ В ЭЛЕКТРОННЫХ КАТАЛОГАХ

В настоящей статье рассказывается, как в рамках библиотечного каталога, работающего под DOS, можно получать улучшенные выходные формы: страницы html и документы Word. Исходные данные для нашего примера мы взяли из программы «Библиотека 4», однако предлагаемая методика вполне приемлема для любого другого электронного каталога, способного экспортировать данные в формате MARC.

Исходные данные и тексты программ можно найти в электронном приложении к статье, которое размещено на сайте журнала.

В подкаталоге **part1** находится файл **src.mrc**. Это экспортированные из «Библиотеки 4» (в формате USMARC) описания новых поступлений. На основании информации из этого файла сформируем «Список новых поступлений» в формате **html** и **rtf**.

Дополнительную информацию можно получить в статьях [1,2].

Подготовка данных для отчета

Сначала нужно перевести информацию из формата USMARC (iso-файл) в текстовый вид. Исходные данные и тексты программ находятся в подкаталоге **part1** приложения.

Для перевода информации в текстовый вид нужно выполнить командный файл **i2t.bat.**

```
rem      i2t.bat
rem Перевод iso-файлов в текстовый вид
..\gawk --re-interval -f iso2text.awk src.mrc >src.txt
rem Перекодировка из DOS в Windows 1251
..\transdwk dw src.txt
```

В отличие от ISIS (см. [2]), «Библиотека 4» не разбивает выходные iso-файлы на строки, поэтому и удаление лишних концов строк не нужно.

Перекодировка информации из DOS в Windows осуществляется с помощью специальной программы:

transdwk.exe.

Эта программа очень простая, поэтому мы не будем останавливаться на ее работе. Если вызвать ее без параметров, программа выдаст на экран короткую инструкцию.

Программа перевода информации в текстовый вид (**iso2txt.awk**) подробно описана в [2]. Единственное отличие заключается в том, что в файле, полученном из «Библиотеки 4», содержатся другие разделители. Разделитель записей в формате USMARC – символ, шестнадцатеричный код которого равен **1E**, а код разделителя полей – **1D**.

Приводим текст программы без комментариев.

```
#       iso2text.awk
# Перевод iso-файлов в текстовый вид
BEGIN{
 FS = "$";
 RS = "#";
}
{
 tmp = $1;
```

```
        m1 = substr(tmp,6,7);
        m2 = substr(tmp,18,7);
        printf("xxxxx%sxxxxx%s\n",m1,m2);
        sub(/^.{24}/,"",tmp);
        i = 0;
        while(tmp != ""){
         nfld[i++] = substr(tmp,1,3);
         sub(/^.{12}/,"",tmp);
        }
        for(i=2; i<=NF; i++)
         if($i != "") print nfld[i-2] $i;
        print "***** End of record N " NR;
      }
```

Выходной файл – **src.txt**.

Примечание

В каталоге **part1** находится и программа перевода информации из текстового файла в формат USM*A*RC. Программа подробно описана в [2].

Перейдем теперь к выбору информации для списка новых поступлений. Для этого мы слегка упростим конвертор из формата MARC в автоматизированную картотеку книгообеспеченности, подробно описанный в [1]. Исходные данные и тексты программ находятся в подкаталоге **part2** приложения.

Мы предполагаем получить улучшенный библиографический список, поэтому разобьем сокращенное библиографическое описание книги на четыре части (потом каждую часть мы отформатируем отдельно):

1. автор;
2. заглавие;

3. продолжение заглавия и сведения об ответственности;
4. выходные данные книги.

Необходимо выбрать следующие подполя из записей формата USMARC:

100a - первый автор, ФИО;
245a – заглавие;
245b – продолжение заглавия;
245c – сведения об ответственности;
260a - место издания;
260b – издательство;
260c – дата издания.

Вызов программы выбора информации из файла **src.txt** осуществляется с помощью следующего командного файла:

```
rem      go.bat
rem Подготовка информации для отчета
..\gawk -f get_info.awk src.txt > info.txt
```

Выходной файл – **info.txt**.

Сама программа выбора информации не сильно отличается от конвертора, описанного в [1]. Естественно, поля имеют другую кодировку. Кроме того, разделитель подполей в формате USMARC – символ, шестнадцатеричный код которого равен **1F.**

Текст программы и комментарии к ней приведены ниже.

```
#      get_info.awk
# Выбор информации для списка поступлений
BEGIN{
# Устанавливаем разделитель подполей
```

```
FS = “\x1F”;
}
# Из поля 100 выбираем подполе a и записываем во
# временную переменную tmp_100a
/^100/{
 for(i=2; i<=NF; i++){
  val = $i;
  let = substr(val,1,1);
  sub(/^./,””,val);
  if(let != “a”) continue;
  tmp_100a = val;
  break;
 }
}
# Из поля 254 выбираем подполя abc и записываем
во
# временные переменные
# tmp_245a , tmp_245b , tmp_245c
/^245/{
 for(i=2; i<=NF; i++){
  val = $i;
  let = substr(val,1,1);
  sub(/^./,””,val);
  if(let == “a”){
   tmp_245a = val;
   continue;
  }
  if(let == “b”){
   tmp_245b = val;
   continue;
  }
```

```
  if(let == “c”){
   tmp_245c = val;
   continue;
  }
 }
}
# Из поля 260 выбираем подполя abc и записываем
во
# временные переменные
# tmp_260a , tmp_260b , tmp_260c
/^260/{
 for(i=2; i<=NF; i++){
  val = $i;
  let = substr(val,1,1);
  sub(/^./,””,val);
  if(let == “a”){
   tmp_260a = val;
   continue;
  }
  if(let == “b”){
   tmp_260b = val;
   continue;
  }
  if(let == “c”){
   tmp_260c = val;
   continue;
  }
 }
}
#################################
#   Конец записи в iso-файле
#################################
```

```
/^\*\*\*\*\*/{
# Автор
 s1 = tmp_100a;
# Заглавие
 s2 = tmp_245a;
# Продолжение заглавия и сведения об
ответственности
 s3 = “”
 if(tmp_245b != “”) s3 = s3 “: “ tmp_245b;
 if(tmp_245c != “”) s3 = s3 “ / “ tmp_245c;
# Место издания, издательство, год издания
 s4 = sprintf(“-%s: %s, %s”,
        tmp_260a,tmp_260b,tmp_260c);
# Вывод информации
 printf(“%s##%s##%s##%s\n”,s1,s2,s3,s4);
# Очистка временных переменных
 tmp_100a = “”;
 tmp_245a = “”;
 tmp_245b = “”;
 tmp_245c = “”;
 tmp_260a = “”;
 tmp_260b = “”;
 tmp_260c = “”;
}
```

В выходном файле информация разделена строкой “##”.

Отметим, что электронный каталог позволяет подготовить информацию своими внутренними средствами. Однако, на наш взгляд, предлагаемый подход более универсален и позволяет достаточно легко адаптировать наши программы для любых iso-файлов, экспортируемых любыми электронными каталогами. Сравнение текстов программ

показывает, насколько просто использовать программы, работающие с данными ISIS, для работы с информацией «Библиотеки 4».

После того, как подготовлена информация для отчета, приступаем к его формированию в форматах **html** и **rtf**. Алгоритм формирования отчета достаточно прост и одинаков для обоих форматов:

1. Разработка «шаблона» отчета. Место в шаблоне, куда будет помещена информация, кодируется специальным образом.. В примерах используется кодировка ***N** (N - цифра).
2. Отчет разбивается на части, каждая из которых может выводиться несколько раз. Для разделения частей используется специальная строка символов, которая наверняка не встречается в шаблоне. Мы используем в качестве такого разделителя строку, начинающуюся символами **<!**##########.
3. Специальная программа заполняет поля шаблона и выводит информацию в выходной файл.

Отчет в формате html.

Формат **html** значительно проще формата **rtf**. Поэтому начнем с формирования отчета в этом формате. Исходные данные и тексты программ находятся в подкаталоге **part3** приложения.

Шаблон отчета – файл **html_ptn.html**.

```
<html>
<head>
<title>Список новых поступлений</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
```

```
</head>
<body bgcolor="#FFFFFF" text="#000000">
<h3 align="right">***1</h3>
<h2 align="center">Список новых поступлений</h2>
<table width="100%" border="0">
<!########## - конец части 0>
<tr>
 <td width="5%" valign="top" align="right">***1</
td>
 <td width="95%" align="left">
  <b>***2 <i>***3 </i></b>***4
  <font face="Arial, Helvetica, sans-serif" size="2">
   ***5
  </font>
 </td>
</tr>
<!########## - конец части 1>
</table>
</body>
</html>
```

В браузере шаблон будет выглядеть примерно так:

```
                                        ***1
        Список новых поступлений
***1  ***2  ***3  ***4 ***5
```

Шаблон разбит на три части (части нумеруются, начиная с нуля).

Часть 0 – начало отчета. В этой части только одно переменное поле (*****1**) – текущая дата.

Часть 1 – строка таблицы.

Таких строк в отчете будет столько, сколько наименований книг в исходных данных.

В части 5 переменных полей мы перечислим их, сохраняя форматирование этих полей в шаблоне:

***1 – номер по порядку;

***__2__ – автор;

*****3****** – заглавие;

***4 – продолжение заглавия и сведения об ответственности;

***5 – выходные данные книги.

Покажем, как из шаблона и исходных данных получается список новых поступлений в формате **html**.

Вызов программы генерации отчета осуществляется с помощью следующего командного файла:

```
rem      go.bat
rem Формирование отчета в формате html.
..\gawk --re-interval -f rpt_html.awk info.txt >res.html
```

Выходной файл отчета – **res.html**.

Рассмотрим подробно программу формирования отчета. Она начинается с описания функции, кодирующей некоторые символы, которые нельзя использовать в файлах html непосредственно. Ниже перечислены эти символы:

Символ	Обычное имя символа	HTML запись
<	“меньше чем”, левая угловая скобка	**<**
>	“больше чем”, правая угловая скобка	**>**
&	амперсант	**&**
“	двойные кавычки	**"**

Функция, кодирующая специальные символы html имеет вид:

```
# str - исходная строка
# s - локальная переменная
function HTML_spec(str,  s){
 s = str;
 gsub(/\&/,"\\\\\\&amp;",s);
 gsub(/\"/,"\\\\\\&quot;",s);
 gsub(/</,"\\\\\\&lt;",s);
 gsub(/>/,"\\\\\\&gt;",s);
 return s;
}
```

Текст функции требует пояснения. Во-первых, AWK использует специфическое описание и вызов функций. Все переменные, записанные в круглых скобках, в описании функции могут быть и параметрами, и локальными переменными. Все зависит от вызова функции.

Пусть описана функция **f(x,y,z)**. Если мы вызовем **f(1)**, то переменная **x** станет параметром и ей будет присвоено значение **1**. Если же мы вызовем **f(1,2,3)**, то все три переменные станут параметрами функции, им будут присвоены значения **x=1, y=2,z=3**.

Принято отделять в тексте программы переменные, которые мы собираемся использовать как параметры, от локальных переменных несколькими пробелами.

Бросается в глаза слишком большое количество символов **"\"** (обратная косая черта) во втором параметре функции **gsub**. Дело в том, что для этой функции амперсант – специальный символ и должен экранироваться обратной косой чертой специальным образом. В этих условиях программа **AWK** требует для правильной работы специфической записи.

Функция чтения шаблона отчета имеет следующий вид.

```
# fn - имя файла шаблона отчета
# i - локальная переменная
function read_ptn(fn, i){
 ptn_count = 0;
 while (getline < fn){
  if($0 ~ /^<!##########/)
   ptn_count++;
  else
   ptn_src[ptn_count] = ptn_src[ptn_count] $0 "\
n";
 }
 close(fn);
 ptn_count++;
 for(i=0; i<ptn_count; i++)
  ptn_cur[i] = ptn_src[i];
}
```

Мы используем встроенную функцию getline, которая просто читает строку из указанного файла. Функция возвращает 1, если считала строку и 0, если дошла до конца файла. После чтения информации из файла нужно закрыть его с помощью функции **close**.

Для хранения частей отчета мы определили два массива: **ptn_src** и **ptn_cur**.

Количество частей отчета будет присвоено переменной **ptn_count.**

Эту же переменную используем в качестве номера считываемой части отчета. Каждую часть отчета мы можем выводить несколько раз, при этом снова заполняя переменные поля информацией из исходных данных. В этом случае нам нужна «эталонная» копия части отчета.

В массиве **ptn_src** мы будем хранить исходные тексты частей отчета, а изменять информацию будем в массиве **ptn_cur**.

Функция чтения шаблона очень проста. До тех пор, пока в файле не встретится строка, разделяющая части отчета, записываем считанные строки в элемент массива **ptn_src**, соответствующий текущей части отчета (для части 0 – **ptn_src[0]**, для части 1 – **ptn_src[1]**, и т.д.).

Если считана строка – разделитель частей (напомним, что это - строка, начинающаяся символами **<!**##########), то увеличиваем номер текущей считываемой части на 1.

В конце функции записываем все элементы массива **ptn_src** в массив **ptn_cur**.

После чтения шаблона **html_ptn.html** переменной **ptn_count** будет присвоено значение 3, а части отчета будут записаны в элементы 0, 1 и 2 массивов **ptn_src** и **ptn_cur**.

До чтения первой записи из исходных данных проделываем следующие действия:

```
BEGIN{
# Устанавливаем разделитель подполей
 FS = "##";
# Читаем шаблон
 read_ptn("html_ptn.html");
# Получаем текущую дату в виде ДД.ММ.ГГГГ
 s = strftime("%d.%m.%Y");
# Заменяем ***1 в части 0 отчета
# на текущую дату
 gsub(/\*\*\*1/,s,ptn_cur[0]);
# Выводим часть 0 отчета
 printf(ptn_cur[0]);
# n – номер книги в списке
 n = 0;
}
```

Для каждой строки исходных данных выполняется:

```
{
# Записываем исходный текст части 1 отчета
# в ptn_cur[1]
 ptn_cur[1] = ptn_src[1];
# Форматируем номер книги
 s = sprintf("%d.",++n);
# Меняем ***1 на номер книги
 gsub(/\*\*\*1/,s,ptn_cur[1]);
# Меняем ***2 на автора
 gsub(/\*\*\*2/,HTML_spec($1),ptn_cur[1]);
# Меняем ***3 на заглавие
 gsub(/\*\*\*3/,HTML_spec($2),ptn_cur[1]);
# Меняем ***4 на продолжение заглавия
# и сведения об ответственности;
 gsub(/\*\*\*4/,HTML_spec($3),ptn_cur[1]);
# Меняем ***5 на выходные данные книги
 gsub(/\*\*\*5/,HTML_spec($4),ptn_cur[1]);
# Выводим часть 1
 printf(ptn_cur[1]);
}
```

Отметим, что в дате и номере книги специальные символы **html** встретиться не могут. Во всех остальных полях исходных данных специальные символы кодируются с помощью функции **HTML_spec**.

После чтения последней строки выводим часть 2 отчета, переменной информации в этой части нет.

```
END{
 printf(ptn_cur[2]);
}
```

Выполним программу и получим файл **res.html**

25.08.2003

Список новых поступлений

1	**Лихачева О.Н.** ***Финансовое планирование на предприятии :*** Учебно-практическое пособие / О.Н.Лихачева -М.: Проспект, 2003
2	**Лагерь А.И.** ***Инженерная графика :*** Учебник для вузов / А.И.Лагерь -М.: Высш.школа, 2003
3	**Пирогов Е.Н.** ***Сопротивление материалов :*** Конспект лекций с примерами типичных расчетов/ Е.Н.Пирогов,В.Ю.Гольцев -М.: Айрис-Пресс, 2003
4	**Бердникова Т.Б.** ***Оценка и налогообложение имущества предприятий :*** Учеб.пособие / Т.Б.Бердникова -М.: ИНФРА-М, 2003

Отчет в формате rtf

Формат rtf (Rich Text Format) был определен фирмой Microsoft как стандартный формат для обмена текстовыми документами.

Формат **rtf** поддерживается программами WORD для Macintosh, начиная с версии 3.X, и WORD для PC, начиная с версии 4. X, и многочисленными текстовыми процессорами, работающими в UNIX-подобных операционных системах.

Изучение формат **rtf** – дело трудоемкое. Мы надеемся, что предложенный нами алгоритм создания шаблона позволит читателю создавать отчеты в формате **rtf** даже при том условии, если программист библиотеки вообще ничего не знает об этом формате.

Исходные данные и тексты программ находятся в подкаталоге **part4** приложения.

Для создания шаблона отчета воспользуемся программой Microsoft Word. Нужно создать новый документ и проделать следующие действия:

1.в начале страницы написать заголовок отчета.

***1

Список новых поступлений

2. вставить таблицу:

количество строк – 1,

столбцов – 2.

Сделать границы таблицы невидимыми.

Уменьшить ширину 1-го столбца (здесь будет находиться номер книги в списке).

Полученная таблица будет выглядеть примерно так:

***1
Список новых поступлений

3.Занести в таблицу переменные поля.

При записи в переменные поля русскиго текста, необходимо в описании поля написать что-нибудь по-русски. В противном случае *Word* может «запутаться» в шрифтах. Мы напишем в полях 2,3,4,5 между звездочками и цифрой слово «Поле”:

***1	*** Поле 2 *** Поле 3 *** Поле 4 *** Поле 5

4. Вставить колонтитулы. Данная операция выполняется для более удобного пользования документом и улучшения его вида. Естественно, она может быть исключена, если подобная задача не ставится пользователем.

5. Сохранить документ в формате **rtf**.

В подкаталоге **part4** приложения имеется полученный таким образом файл **rtf_tpt.rtf**. Превратим этот файл в шаблон отчета.

Шаблон отчета не будет «правильным» файлом формата **rtf**, поэтому скопируем его с новым именем **rtf_tpt.__r**) и приступим к редактированию. Рекомендуем пользоваться при этом встроенным редактором программы **FAR**.

Выделим ту часть файла **rtf**, которая является строкой таблицы. Эта часть файла имеет такой вид:

\trowd
текст, не содержащий **\trowd**
******* (начало описания переменных полей)
текст, не содержащий **\row**
\row }

Процедура получения шаблона:

1. Открыть файл **rtf_tpt.__r** для редактирования.

2. С помощью функции поиска найти строку символов *******. На экране можно будет увидеть примерно следующее (символ над курсором подчеркнут):

\cgrid\langnp1049\langfenp1049 {\b\fs28 ***1

3. Первое переменное поле не относится к таблице, Поэтому нужно найти следующее - это тоже поле *****1**. Переводим курсор за звездочки и ищем строку символов *******. На экране увидим: символ над курсором подчеркнут:

\faauto\adjustright\rin0\lin0 {*1\cell }**

4. Теперь найдем первую строку символов **\trowd**, которая находится в файле перед только что найденными звездочками.

Для этого в окне параметров поиска установим флажок **Reverse search** (“Поиск в обратном порядке”). Результат поиска будет выглядеть на экране примерно так:

\par }\trowd \trgaph108\trbrdrt

5. Переносим **\trowd** на две строки вниз и в пустую строку записываем <!########## - конец части 0>.
\par }

<!########## - конец части 0>
\trowd \trgaph108\trbrdrt

6. Теперь нужно найти конец строки таблицы. Для этого ищем сначала ***, а затем строку символов **\row** (флажок **Reverse search** нужно снять, мы ищем от позиции курсора до конца файла). На экране увидим:

\cellx9462\row }\pard \ql \li0

7. Переносим **\pard** на две строки вниз и в пустую строку записываем <!########## - конец части 1>.

\clwWidth8922 \cellx9462\row }
<!########## - конец части 1>
\pard \ql \li0\ri0\widctlpar\aspalpha

8. Для того, чтобы программа получения отчета в формате **rtf** не отличалась существенно от программы, генерирующей отчет в формате **html**, необходимо удалить из описания переменных полей слово «Поле». Word представляет русские буквы в следующем формате:

\’xx , где **xx** – шестнадцатеричный код символа.

Слово «Поле» записывается так: **\’cf\’ee\’eb\’e5**.
Для удаления из файла этой строки символов можно

воспользоваться функцией редактора **Replace** (“Заменить”). Поле **Replace with** (“Заменить на”) нужно оставить пустым.

Шаблон отчета готов (файл **rtf_tpt.__r** находится в подкаталоге **part4** приложения), Получим теперь список новых поступлений в формате **rtf**.

Вызов программы генерации отчета осуществляется с помощью следующего командного файла:

```
rem      go.bat
rem Формирование отчета в формате rtf.
..\gawk --re-interval -f rpt_rtf.awk info.txt >res.rtf
```

Рассмотрим подробно программу **rpt_rtf.awk**, которая формирует отчет.

Все символы, которые мы собираемся вставлять в отчет, нужно представить в шестнадцатеричном виде: **\’xx**.

Программа **AWK** не имеет стандартных функций работы с символами, поэтому поставленная задача потребует некоторых ухищрений. С помощью функции **RTF_hex_init** (функция вызывается один раз в начале программы) сформируем массив, индексы которого – символы, а значения элементов – шестнадцатеричное представление символов.

Всего символов - 256 (десятичные коды от 0 до 255).

```
# Локальные переменные : i,c,h
function RTF_hex_init(  i,c,h){
 for(i=0; i<=255; i++){
  c = sprintf(“%c”,i);
  h = sprintf(“\\’%2.2x”,i);
  char2rtfhex_array[c] = h;
 }
}
```

Для получения символа и его шестнадцатеричного представления из десятичного кода воспользуемся функцией **sprintf** (форматный вывод в строку). Вся информация записывается в массив подстановок **char2rtfhex_array**.

Функция **RTF_hex** осуществляет перевод символов исходной строки в шестнадцатеричный вид. С помощью функции **substr** из строки последовательно выбираются символы, а в выходную строку записываются значения элементов массива подстановок, соответствующие этим символам.

```
# Локальные переменные : i,c,s
function RTF_hex(str,  i,c,s){
 for(i=1; i<=length(str); i++){
  c = substr(str,i,1);
  s = s char2rtfhex_array[c];
 }
 return s;
}
```

Функция чтения шаблона read_ptn ничем не отличается от аналогичной функции, использованной для чтения шаблона в формате **html**.

Алгоритм вывода отчета в формате **rtf** практически не отличается от вывода отчета в формате **html**:

1. В начале программы вызывается функция **RTF_hex_init**, формирующая массив подстановок для представления выводимой информации в шестнадцатеричном виде.

2. Вместо функции **HTML_spec** используется функция **RTF_hex**.

```
BEGIN{
 FS = "##";
 RTF_hex_init();
```

```
read_ptn("rtf_ptn.__r");
  s = strftime("%d.%m.%Y");
  gsub(/\*\*\*1/,s,ptn_cur[0]);
  printf(ptn_cur[0]);
  n = 0;
}

{
  ptn_cur[1] = ptn_src[1];
  s = sprintf("%d.",++n);
  gsub(/\*\*\*1/,s,ptn_cur[1]);
  gsub(/\*\*\*2/,RTF_hex($1),ptn_cur[1]);
  gsub(/\*\*\*3/,RTF_hex($2),ptn_cur[1]);
  gsub(/\*\*\*4/,RTF_hex($3),ptn_cur[1]);
  gsub(/\*\*\*5/,RTF_hex($4),ptn_cur[1]);
  printf(ptn_cur[1]);
}

END{
  printf(ptn_cur[2]);
}
```

Выполним программу и получим файл **res.rtf.** Отчет по внешнему виду не отличается от отчета в формате **html**.

Список литературы

1. Бочаров Б.П., Воеводина М.Ю. AWK - универсальная программа работы с текстовыми файлами // Библиотеки учебных заведений. - 2002. - N 4

2. Бочаров Б.П., Воеводина М.Ю. Глобальная корректировка баз данных с использованием программы AWK // Библиотеки учебных заведений. - 2003. - N 7