Perl в Windows – обработка текстов и графический интерфейс

Дмитрий Ландэ
Информационный центр «ЭЛВИСТИ»
dwl@visti.net

Perl (Practical Extraction and Reporting Language) ассоциируется с UNIX. Это правильная ассоциация – изначально (с 1986 г.) язык Perl рассматривался как инструмент системного администрирования в UNIX, однако со временем превратился в один из самых популярных языков-интерпретаторов, идеально подходящий для обработки текстов на любых операционных платформах.

Популярность Perl в последнее время обуславливается несколькими причинами:
- актуальностью задач «пакетной» обработки текстов;
- бесплатностью и доступностью (данный язык является свободно распространяемым программным продуктом);
- многоплатформностью (поддерживаются практически все разновидности UNIX и Windows 9x/NT);
- возможностями использования в Internet в качестве языка CGI-сценариев и благодаря интеграции со средствами генерации динамических html-страниц. Эти возможности очень популярны, однако зачастую возникают проблемы перегрузки серверов, на которых одновременно запускаются сотни Perl-приложений;
- появлением действительно эффективных средств графического интерфейса пользователя (GUI).

Именно последнему аспекту уделена значительная часть данной статьи. До последнего времени Perl применялся в основном для «пакетной» обработки информационных массивов, визуализацией же занимались другие пакеты, особенно явно это проявлялось в самых популярных у пользователей операционных средах Windows 9x/NT. С появлением графических средств GUI Perl может стать реальным конкурентом таких средств, как например Visual Basic. Вместе с тем, уже не секрет, что в настоящее время ActiveState готовит плагин «графического» Perl для Visual Studio 7.

Хотя определенная часть информации, приведенная в статье, присущи любой пратформе, на которой установливается Perl, некоторые сведения (инсталляция и графический интерфейс пользователя) относятся исключительно к операционным системам Windows 9x/NT.

Возможности Perl

Прежде всего, следует заметить, что язык Perl являлся изначально интерпретируемым, поэтому для выполнения программ на Perl достаточно ввести команду:
perl <имя программы> [<параметры>]

где “имя программы” – исходный текст программы на Perl, а “параметры” – опциональные значения, которые могут быть использованы программой.

Мы не ставим задачи дать здесь полноценное руководство по Perl, однако для демонстрации того, что этот язык действительно простой (а точнее многоуровневый) в освоении, приведем описание некоторых его возможностей. Как и каждый полноценныый язык программирования высокого уровня, Perl, например, включает такие операторы, как:

- присвоение (“=”)
- сравнение (“<”,”>”,”<=”,”=>”,”lt”,”gt”,”le”,”ge”)
- равенство (“==”,”!=”,”< = >”,”eq”,”ne”,”cmp”)
- условные операторы (“if”, “else”, “elsif”, “unless”)
- операторы циклов (“for”, “foreach”, “while”, “until”)
- операторы управления циклом (“next”, “last”, “redo”)
- оператор перехода (“goto”)

и многие другие…

Perl, как язык ориентированный на текстовую обработку содержит большое количество функций для работы со строками, причем программисты разного уровня могут выбирать свои подмножества Perl для достижения одних и тех-же целей. Тех, кто пришел к программированию на Perl от Clipper или Pascal некоторое время будут привлекать такие функции, как index и substr. Для UNIX-специалистов будут близки средства шаблонов и регулярных выражений (regexp) – самая «сокоровенная» часть Perl.

Приведем некоторые функции работы со строками:

index – определение позиции подстроки в строке (первое вхождение)
rindex - определение позиции подстроки в строке (последнее вхождение)
length – длина строки
substr – подстрока текстовой строки
split – разбивка строки на набор подстрок по шаблону-разделителю
sprintf – форматирование выходной строки в стиле C
printf – печать форматированного текста в стиле C
chomp – удаление символа конца строки (обычно - \n) из строки
chr – преобразование числа в символ
ord – преобразование символа в код

Наряду с традиционными для языков высокого уровня средствами работы со строками, в Perl присутствуют «непривычные» для простого пользователя средства работы с регулярными выражениями и шаблонами, которые были перенесены из UNIX. Да, регулярные выражения требуют серьезных усилий при их освоении, однако, научившись работать и поработав с этими “оригинальными” средствами, ни один программист уже не захочет возвращаться к традиционным способам обработки строк.

При этом, по сути, регулярные выражения позволяют сопоставлять строки с указанными шаблонами и выполнять замену подстрок. В Perl имеется всего два основных оператора, связанных с регулярными выражениями:

- m/…/ - проверка совпадений;
- s/…/…/ - подстановка текста

Третий, родственный этим двум операторам – оператор замены текста – tr/…/…/ не использует регулярные выражения.

Приведем пример использования оператора подстановки текста s. Допустим во фразе “Hello, Basic” мы хотим заменть слово “Basic” на слово “Perl”.
Фрагмент программы будет выглядеть так:

```
$text=”Hello, Basic”;
$text=~s/Basic/Perl/;
```

При использовании традиционных средств программирования аналогичный фрагмент выглядел бы приблизительно следующим образом:

```
$text=”Hello, Basic”;
$i=index($text,”Basic”);
if ($i>-1) {
        $tmp=substr($text,0,$i);
        $text=$tmp.”Perl”;
}
```

Возможности оператора проверки совпадения m покажем на втором примере, заключающемся в необходимости выделения числового значения из произвольной фразы, например “Perl был создан в 1986 году”.

```
$text=”Perl был создан в 1986 году”;
if ($text=~m/.*\s(\d+)\s.*/) {
        $num=$1;
}
```

Следует отметить, что регулярные выражения - это одна из наиболее сложных (но и мощных) возможностей Perl, программы, написанные профессионалами с их использованием выглядят как произведения искусства, а элегантность кода поражает воображение. Однако настоящую мощь Perl обрел, начиная с 5 версии, когда появился механизм его расширения, приведший к фактической децентрализации языка. В результате было создано огромное количество «системных» приложений, в том числе и графический интерфейс пользователя GUI.

Графический интерфейс пользователя

Hello, world

Программисты, знакомящиеся к любым графическим интерфейсом, считают первым по счету достижением в освоении новой среды вывод сообщения “Hello, world”. Мы тоже начнем с этого. Для начала опишем в нашей Perl-программе загрузку графического интерфейса:

```
use Win32::GUI;
```

Затем определим Windows-окно (100x100) для вывода сообщения:

```
$main = Win32::GUI::Window->new(-width => 100, -height => 100, -
name=>'Main');
```

Известно несколько методов вывода сообщений, воспользуемся одним из них – AddLabel():

```
$main->AddLabel(-text => "Hello, world");
```

Активизируем новое окно:

```
$main -> Show();
```

Для обеспечения выполнения интерактивных функций и ожидания результатов их работы вызывается функция организации диалога:

```
Win32::GUI::Dialog();
```

После создания окна и вывода в него сообщения возникнет потребность в корректном завершении программы, например путем стандартного закрытия созданного окна. Закрытие окна вызовет ситуацию Terminate, корректную обработку которого обеспечит подпрограмма:

```
sub Main_Terminate {
      -1;
}
```

Таким образом, вывод сообщения "Hello, world" в графическое окно выполняется следующей простой программой:

```
use Win32::GUI;

$main = Win32::GUI::Window->new(-width => 100, -height => 100,
-name => 'Main');

$main->AddLabel(-text => "Hello, world");

$main -> Show();

Win32::GUI::Dialog();

sub Main_Terminate {
      -1;
}
```

Некоторые (далеко не все) графические элементы

Labels (метки) – метод AddLabel(). См. выше.

Buttons (кнопки) – метод AddButton().

Если в приведенном выше примере заменить строку с AddLabel следующей строкой:

```
$main->AddButton(-text => “Hello, world”, -name => “Camel”);
```

то в результате выполнения программы получим картинку:

При этом нажатие на кнопку вызывает состояние, обработка которого может быть организована в подпрограмме с именем: Camel_Click

Text Fields (текстовые поля) – метод AddTextfield().

Замена стоки с AddLabel в примере следующей строкой:

```
$main->AddTextfield(-text => "Hello, world", -left => 5, -top => 10,
-width => 90,-height => 20);
```

(здесь в явном виде указаны координаты и размеры), приводит к появлению текстового поля:

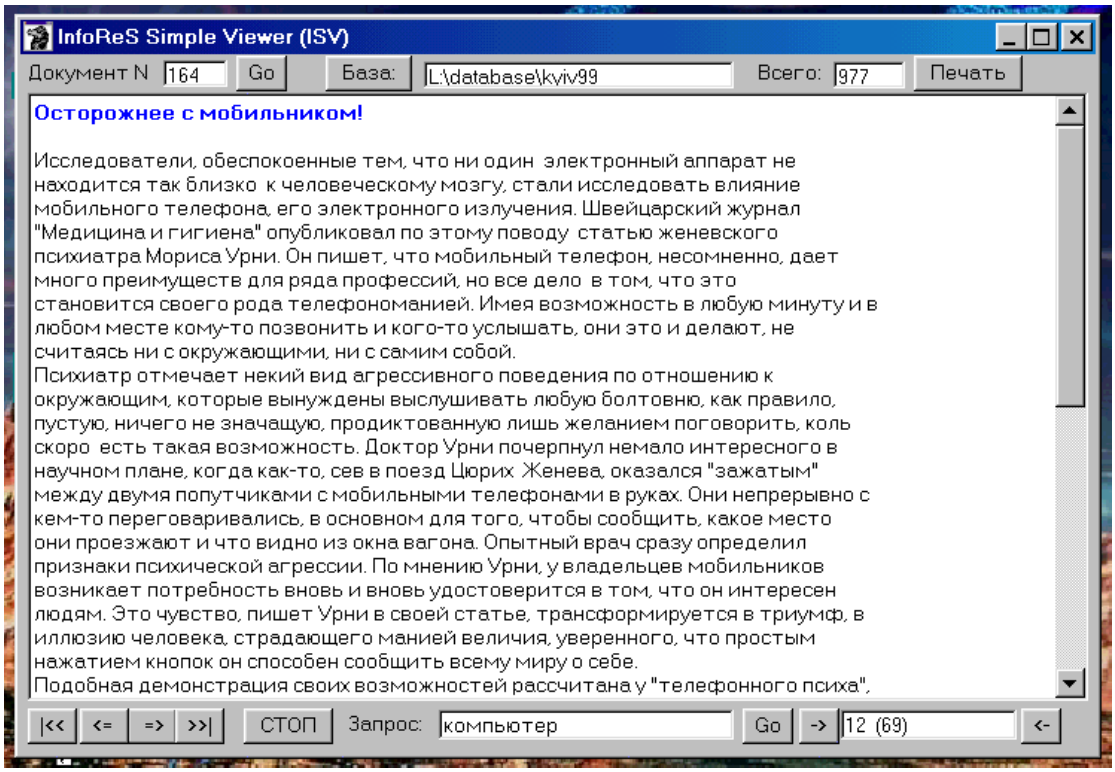
Text Edit (редактирование текста) – метод AddRichEdit().

Замена стоки с AddLabel в примере следующей конструкцией:

```
$Textbox = $main->AddRichEdit(
    -name    => "WText",
    -left    => 5,
    -top     => 5,
    -width   => 90,
    -height  => 50,
    -style   => WS_CHILD | WS_VISIBLE | WS_VSCROLL
                | ES_LEFT | ES_MULTILINE | ES_AUTOVSCROLL,
);
$Textbox->Text("Hello,\nworld");
```

приводит к появлению поля редактирования текста (здесь перенос слова world на вторую строку был обеспечен использованием спецсимвола “\n”):

Существуют и другие возможности графического интерфейса пользователя (GUI), - их подробное описание может занять целый учебник, отдельные разделы которого представлены в поставляемой с пакетом документацией. Для демонстнации некоторых «оконных» диалоговых возможностей Perl в качестве примера приведем интерфейс простейшего средства просмотра и поиска в базах данных информационно-поисковой системы InfoReS-XL. Данное средство просмотра, в отличие от основной ветви системы, разработанной на C, и адаптированной для работы в Internet, реализовано на ActivePerl с использованием GUI.



В этом примере тест документа выводится в поле редактирования (метод AddRichEdit()), диагностическая информация и запросы выводятся в текстовое поле (метод AddTextfield()), а все кнопки выводятся методом AddButton(). Некоторые особенности программированя на Perl в среде Windows 9x/NT, с которыми автор столкнулся при написании этой программы будут приведены ниже.

Инсталляция для Windows

Для установки последней версии Perl достаточно инсталлировать пакет ActivState Perl, который свободно доступен по адресу http://www.activstate.com или

на прилагаемом к журналу CD (файл /perl/active/APi515e.exe). Для установки графического интерфейса GUI следует после установки ActiveState Perl перенести модули с CD (каталог /perl/gui.exe.lib/blib/) в соответствующие библиотеки Perl :

/perl/gui.exe.lib/blib/arch/auto//Win32/GUI – в /Perl/site/lib/auto/Win32/GUI;
/perl/gui.exe.lib/blib/html/lib/Win32 – в /Perl/html/lib/Win32;
/perl/gui.exe.lib/blib/lib/Win32 – в /Perl/site/lib/Win32.

В Internet один из адресов, по которому можно «скачать» модуль графического интерфейса пользователя (и много других полезных программ) – http://dada.perl.it

После выполнения инсталляции бывает полезно выяснить версию установленного программного обеспечения. В случае Perl достаточно ввести команду с параметром: Perl –v . Если все было установлено корректно, должно появится сообщение с версией Perl, конкретно Perl 5.005_3.

Названный выше графический интерфейс пользователя не подразумевает наличие графической среды разработки, попытки создания которой сегодня не прекращаются. В настоящее время разработано большое количество редакторов для работы с исходными текстами Perl, как свободно распространяемых, так и коммерческих. Однако, в качестве самого простого редактора достаточно использовать, например, Notepad. Вместе с тем, мы рекомендуем использовать версию популярнейшего UNIX-редактора VI, адаптированного под Windows-платформу. Этот редактор можно использовать, ничего не зная о командах текстового редактора VI для UNIX, но его истинная мощь достигается использованием специальных команд и макросов, часть которых была опубликована в 6 номере журнала за 2000 год. Текстовый редактор типа VI находится в каталоге /perl, имя файла winvi32.exe. В Internet информацию об этом редакторе (и сам модуль) можно получить по адресу - http://home.snafu.de/ramo/WinViEn.htm . При работе с средах Windows 9x/NT для обеспечения переносимости программ, разработанных на языке Perl, путем их компиляции и превращения в exe-модули существует несколько программ, большинство из которых является коммерческими. С вполне работоспособной демо-версией одной из таких программ (perl2exe) можно ознакомиться в Internet по адресу http://www.perl2exe.com или на CD в каталоге /perl/perl2exe.

Некоторые особенности

При использовании Perl в среде Windows 9x/NT, да и еще вдобавок при работе с символами кириллицы, приходится мириться с некоторыми особенностями реализации интерпретатора (и компиляторов) и «обходить» их. Приведем некоторые примеры, понимая, что полноты такого рода особенностей достичь не удастся.

Путь к файлу

При использовании Perl под управлением Windows 9x/NT всегда следует указывать полный путь к файлу, над которым будут выполняться какие-либо действия, например:

```
Open (FILE,”d:/perl/database/comp.txt”);
```

Обработка символа «конец файла» (EOF) в двоичном файле

Если в UNIX симовл EOF (код 26) является обычным символом, который вполне корректно читается оператором read и входит в прочитанную этим оператором знаковую последовательность, то в Windows 9x/NT операция чтения файла оператором read безусловно завершается при достижении EOF. Это бывает достаточно неудобно, когда символ EOF является не признаком конца файла, а байтом двоичного числа, записанного в файл. Для обхода этой ситуации автор не может предложить ничего лучшего, чем использовать побайтовое чтение (используя оператор getc) с позиционированием в пределах файла с помощью оператора seek, указывая реальный базовый адрес и нулевое смещение. Приведем пример – чтение 4-байтового числа из файла FILE.

UNIX:

```
read (FILE,$x,4,0);

$nx=unpack("L",$x);

print $nx,"\n";
```

Windows 9x/NT:

```
$x="";
for ($jx=0; $jx<4; $jx++) {

     seek FILE,$jx,0;
     $c=getc(FILE);
     if (length($c)==0) { $c=chr(26); }
     $x=$x.$c;
}
$nx=unpack("L",$x);
print $nx,"\n";
```

Обработка символов «Ъ» в кодировке KOI8 и «я» в CP1251

ActivState Perl корректно обрабатывает этот символ (код 255), однако компилятор типа perl2exe воспринимает его как специальный и отображает некорректно. Рекомендуется вместо явного задания этого символа использовать значение chr(255).

Функция печати

Средства графического интерфейса пользователя не включают стандарных средств печати (например, из окна редактирования). Существует несколько путей решения этой задачи, например вызов стандартных средств операционной системы. Приведем пример ($doc – буфер с документом, который требуется распечатать):

```
sub Dprint_Click {
      if (-e "$ENV{'WINDIR'}/System32/mshtml.dll") {
```

```
        $dll="$ENV{'WINDIR'}/System32/mshtml.dll";
    }
    else {
        $dll="$ENV{'WINDIR'}/System/mshtml.dll";
    }
    open OU,">$ENV{'TEMP'}\\1.htm";
    print OU "<pre>\n$doc";
    $ou="$ENV{'TEMP'}\\1.htm";
    close OU;
    system("rundll32.exe $dll,PrintHTML $ou");
    return 0;
}
```