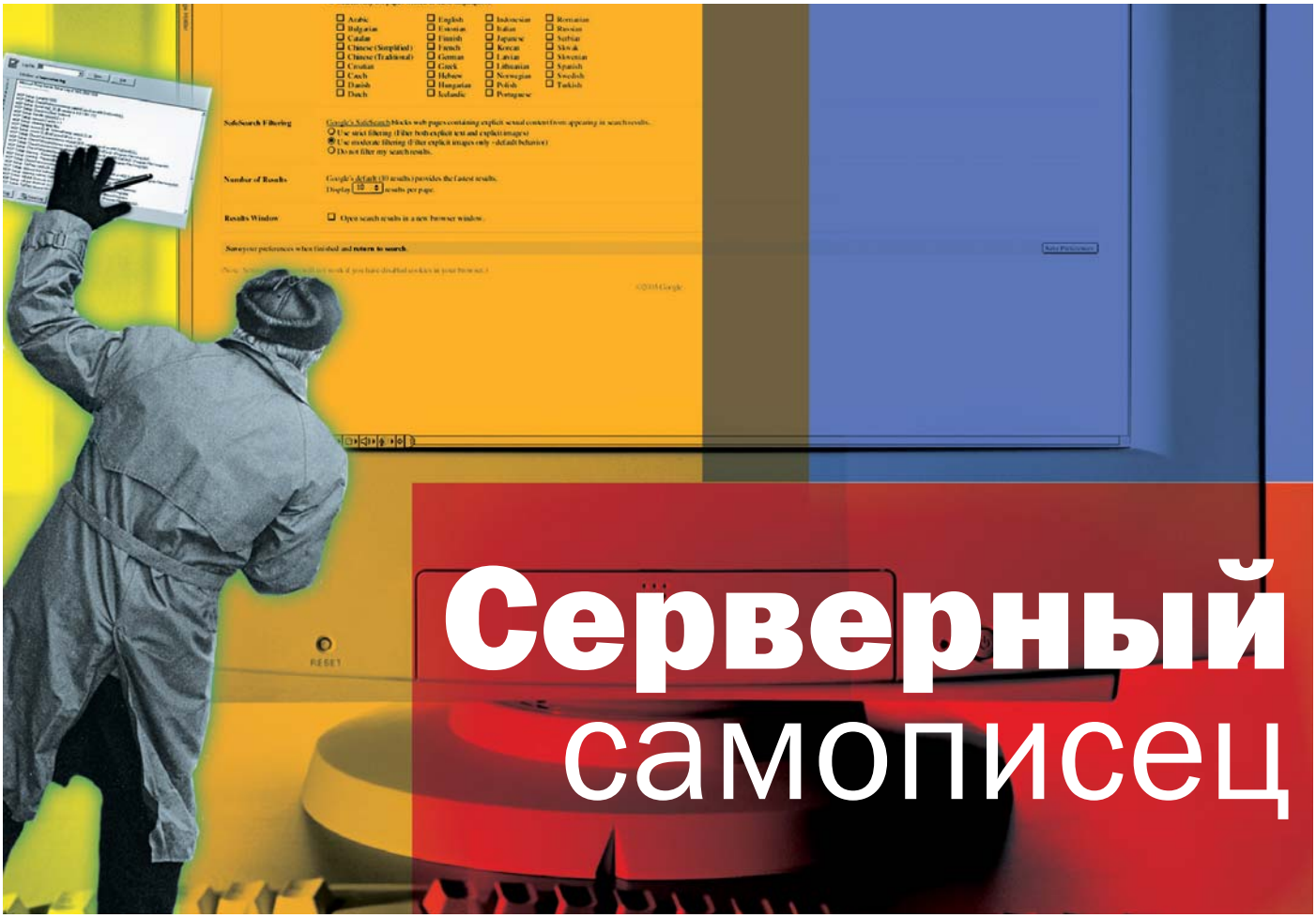


Серверный самописец

Web-разработчикам известны многие способы оценки посещаемости сайтов, но самым точным, надежным и исчерпывающим из всех средств является, несомненно, анализ лог-файлов сервера. При соответствующих настройках web-сервер детально протоколирует все клиентские запросы, а также ошибки и иные примечательные события, помещая отчеты в специально назначенные для этих целей файлы.

Лог-файлы

В большинстве случаев сервер ведет параллельно два файла: журнал доступа (access log) и журнал ошибок (error log). В первый из отчетов заносятся сведения обо всех корректных клиентских запросах, полученных сервером (информация по каждому запросу добавляется сразу же по окончании его обработки). Структуру журнала доступа можно варьировать в широких пределах, но на практике чаще всего используется один из общепринятых форматов, о нем пойдет речь чуть позже.

Второй журнал предназначен для регистрации различных ошибок, возникающих в ходе работы сервера. Настройка этого журнала по сравнению с предыдущим менее гибкая (если говорить о популярном сервере Apache). В частности, при помощи директивы LogLevel администратор может определить лишь «степень тяжести» ошибок, которые заносятся в лог-файл. Таким образом, в зависимости от предпочтений администратора журнал ошибок может содержать сообщения о самых серьезных сбоях, при которых сервер как таковой вообще не загружается, а может регистрировать отладочные сведения о событиях, ошибками вовсе не являющихся.

Возможность доступа к лог-файлам сервера — без сомнения, одно из ключе- »

вых отличий достойного коммерческого хостинга от бесплатных альтернатив. Анализ журнала доступа позволяет проследить любые, даже самые сложные тенденции, связанные с посещаемостью сайта, а также обнаружить попытки атак злоумышленников. Журнал ошибок незаменим при отладке серверных приложений. Как же воспользоваться потенциалом лог-файлов наиболее эффективно?

Журнал ошибок трудных

Вообще говоря, формат лог-файлов может быть совершенно произвольным. Консорциум W3C делает шаги в направлении стандартизации на этом поприще, но пока что документ, посвященный «расширенному формату» (Extended Log Format), находится в стадии рабочего проекта, и до статуса официальной рекомендации ему, по всей видимости, еще далеко. Ознакомиться с текущей версией проекта можно по адресу: www.w3.org/TR/WD-logfile.html. Что же касается реалий сегодняшнего дня, то наиболее распространенным представлением файла журнала доступа является так называемый «комбинированный формат» (combined format), реализованный впервые в web-сервере NCSA. Структура журнала ошибок подчинена лишь фантазии производителей программ-серверов.

Что представляет собой комбинированный формат файла журнала доступа? Фактически это текст с разделяемыми полями. Каждому обработанному клиентскому запросу соответствует отдельная строка. Всякая строка, в свою очередь, состоит из девяти полей, разделенных пробелами. Внутри содержимого полей также допускаются пробельные символы, но при этом оно должно быть обрамлено двойными кавычками (или квадратными скобками, как в случае с полем, отражающим дату и время обработки запроса).

Пример реальной записи файла журнала доступа:

```
ip107-166.dialup.wplus.net — — [27/Jul/2003:02:26:35 +0400] "GET / HTTP/1.1" 200 11819 "http://www.yandex.ru/yandsearch?text=%E6%F3%F0%ED%E0%EB+Chip" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)".
```

Разбор каждого из значений примера можно найти в блоке «Пример строки журнала доступа».

Анализируй это

Разумеется, изучение лог-файлов сервера в чистом виде — нетривиальная задача, потому что даже при средней посещаемости web-ресурса суточный отчет может занимать десятки мегабайт. Но никаких самоистязаний и не требуется: существует множество программ (таких, к примеру, как Webalizer, WebTrends Log Analyzer, WebLog Expert, WebSpy Analyzer, Analog), коммерческих и бесплатных, способных генерировать самые разнообразные статистические отчеты, касающиеся посещаемости сайта, на базе файлов журнала доступа web-сервера.

Большой популярностью в России (в том числе среди провайдеров хостинга) пользуется программа Webalizer, позиционируемая как дополнение для сервера Apache. Как и многое в мире Linux, она доступна для бесплатного скачивания (www.mrunix.net/webalizer). Написанная на языке C, программа работает исключительно быстро. Среди недостатков — не всегда корректная работа с данными на русском языке. Невелико и разнообразие генерируемых отчетов (по сравнению, скажем, с WebTrends Log Analyzer, но этот продукт распространяется на сугубо коммерческой основе). Для повседневного мониторинга статистики посещаемости сайта возможностей Webalizer вполне достаточно. Но порой возникают ситуации, когда требуются более специфичные отчеты. К тому же при помощи Webalizer бывает невозможно прочесть содержание русскоязычных поисковых запросов.

Штудируем логи

В связи с этим я написал простой CGI-скрипт — фильтр данных. Все, что он умеет, — находить в файле журнала доступа строки, содержащие определенный шаблон, и выводить их на экран. Казалось бы, что можно ожидать от этого примитивного средства? Однако не спешите торопиться с выводами: для более-менее опытного web-мастера даже такой незатейливый инструмент может стать хорошим подспорьем. Размер скрипта без учета комментариев — 1,5 Кбайт.

```
#!/usr/bin/perl
```

Формируем поле Content-type заголовка ответа:

```
print "Content-type: text/html\n\n";
```

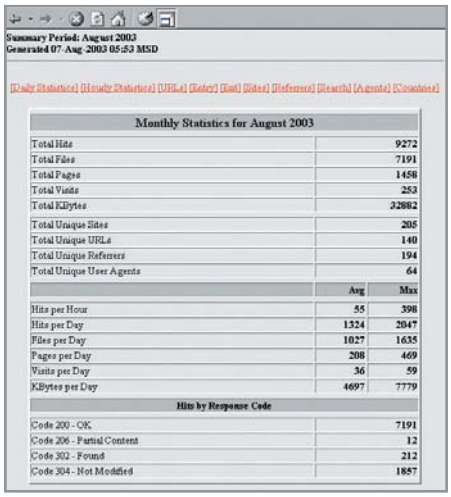
Формируем начало HTML-документа:

```
print "<html>\n\n<head>\n<title>Поиск в логе доступа</title>\n</head>\n\ n<body>\n<h1>Поиск в логе доступа</h1>\n";
```

Выводим HTML-форму с единственным текстовым полем с именем string. Предполагается, что файл данного скрипта называется filter.pl:

```
print "<form action=\"filter.pl\" method=\"GET\">\n";
print "<p>Что ищем: <input type=\"text\" size=\"20\" maxlength=\"100\" name=\"string\"> <input type=\"submit\" value=\"Обработать\"></p>\n";
print "</form>\n";
```

Строка запроса передается методом GET. Обрабатываем ее встроенными средствами Perl (подробно об этом можно прочесть в статье «Открываем шлюзы»). И хотя передается всего один параметр, мы не будем отказываться от сложной процедуры генерации хэша $userdata, потому что скрипт предполагает дальнейшее расширение функциональности: »


Summary Period: August 2003
Generated 07-Aug-2003 05:53 MSD

Monthly Statistics for August 2003

Total Hits	9272
Total Files	7191
Total Pages	1458
Total Visits	253
Total KBytes	32882
Total Unique Sites	205
Total Unique URLs	140
Total Unique Referrers	194
Total Unique User Agents	64

	Avg	Max
Hits per Hour	55	398
Hits per Day	1324	2047
Files per Day	1027	1635
Pages per Day	208	469
Visits per Day	36	59
KBytes per Day	4697	7779

Hits by Response Code

Code 200 - OK	7191
Code 206 - Partial Content	12
Code 302 - Found	212
Code 304 - Not Modified	1857

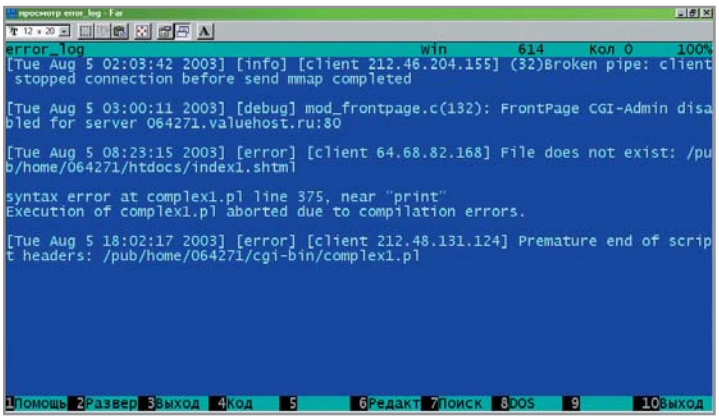
▲ Фрагмент отчета, сформированного программой Webalizer

```
$query = $ENV{QUERY_STRING};

if($query ne '')
{
 @query = split('&', $query);
 foreach $arg(@query)
 {
  @fields = split('=', $arg);
  $fields[0] =~ s/\+/ /g;
  $fields[0] =~ s/%([0-9A-H]{2})/pack('C',
hex($1))/eg;
  $fields[1] =~ s/\+/ /g;
  $fields[1] =~ s/%([0-9A-H]{2})/pack('C',
hex($1))/eg;
  $userdata{$fields[0]} = $fields[1];
 }
}

$userdata{'string'} = '' if(!(defined($userda-
ta{'string'})));
```

Примеры записей журнала ошибок



Если содержимое поля string, полученное из строки запроса, не пустое, и при этом существует файл ../logs/access_log, то читаем все строки файла, пока не достигнем его конца. Если какая-либо строка содержит в себе подстроку, соответствующую шаблону, заданному пользователем в HTML-форме, эта запись выводится в генерируемый отчет. Последовательности, закодированные со знаком % и с указанием шестнадцатеричного кода символа, дважды декодируются при помощи регулярного выражения, использованного выше для обработки строки запроса. Если лог-файл не существует или скрипт запущен без параметра, выводятся предусмотренные для этих случаев сообщения:

```
if($userdata{'string'} ne '')
{
 if(-e '../logs/access_log')
 {
  print "<p>Записи, соответствующие за-
просу:</p>\n";

  open(InFile, '../logs/access_log');

  $i = 0;
  while ($inline = <InFile>)
  {
   chomp $inline;
   if($inline =~ /$userdata{'string'}/)
   {
    $inline =~ s/%([0-9A-H]{2})/pack('C',
hex($1))/eg;
    $inline =~ s/%([0-9A-H]{2})/pack('C',
hex($1))/eg;
    print "<p>$inline</p>";
    $i++;
   }
  }

  close(InFile);

  print "<p>Найдено записей: $i.</p>\n";
 }
 else
 {
  print "<p>Файл лога не найден.\n</p>";
 }
}
else
{
 print "<p>Поиск не производил-
ся.</p>\n";
}
```

По косточкам

Пример строки журнала доступа

Значение	Комментарий
ip107-166.dialup.wplus.net	Адрес хоста клиента. В этом поле вместо доменного имени может фигурировать IP-адрес.
-	Дефис (прочерк) означает, что поле не содержит никаких данных. Вообще же, на этой позиции размещается информация от сервера идентификации identd при условии, что последний запущен на удаленной машине.
-	В этом поле помещается имя пользователя, примененное им при авторизации в случае, если запрошен документ, находящийся в закрытой зоне web-узла. В нашем примере запрошен общедоступный ресурс.
[27/Jul/2003:02:26:35 +0400]	Дата и время окончания обработки запроса. В данном случае — локальное время, проект W3C Extended Log Format будет предполагать указание времени по Гринвичу (GMT).
«GET / http/1.1»	Первая строка заголовка клиентского запроса. Содержит название метода (GET), адрес запрошенного ресурса (/ — корневой каталог), название и версию протокола, использованного при подключении (в нашем случае — HTTP 1.1).
200	Код ответа сервера. Число 200 свидетельствует об успешной обработке — клиентский запрос правилен, и в ответе сервера содержатся запрошенные данные. Полный перечень возможных кодов ответа можно без труда найти в Интернете и в любом специализированном справочнике.
11819	Число байт, переданных в ответе сервера (размер заголовка ответа не учитывается).
«http://www.yandex.ru/yandsearch?text=%E6%F3%F0%ED%E0%EB+Chip»	Реферер — URL страницы, по ссылке с которой пришел пользователь. В нашем случае посетитель нашел запрошенный документ через Yandex по словам «журнал Chip».
«Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)»	Информация о клиенте пользователя, то есть типе и версии браузера, установленного на клиентской машине.

▲ Результат обработки строки поиска 'yandex.ru' скриптом, описанным в статье. Один из поисковых запросов, заданных Yandex, выделен

▲ По запросу 'filter.pl' скрипт расскажет о том, сколько раз его запускали за историю наблюдений, доступную в лог-файле

» Заканчиваем формирование HTML-кода и завершаем программу:

```
print "</body>\n\n</html>";

exit(0);
```

Большие возможности маленького скрипта

Имя и местонахождение лог-файла в коде скрипта (../logs/access_log) нужно заменить реальными координатами этого файла. Разместите скрипт в директории cgi-bin своего сервера, наделив его правами доступа 755. Все готово к работе. Наберите в адресной строке браузера URL скрипта, и, если все сделано правильно, перед вашими глазами появится форма, состоящая из единственного текстового поля. Поэкспериментируем?

Введем первый запрос: ' 404 '. Обратите внимание на пробелы вокруг числа: нас интересует, как вы понимаете, код ответа 404, а не, скажем, размер переданных данных 140457 байт. Вот и первый отчет, не доступный программе Webalizer. Перед вами все сведения о запросах, повлекших ошибку «404 Not Found». Такой отчет весьма полезен — он позволяет отследить «битые» ссылки на сайте, а также страницы на других сайтах, неверно ссылающиеся на ваш ресурс.

Второй наш запрос будет таким: '.zip'. Получаем подробную статистику: кто, сколько раз и в какое время скачивал с сайта ZIP-архивы. Сопоставив объемы переданных сервером данных с фактическими размерами архивов, размещенных на сайте, можно определить, все ли пользователи скачивали файлы до конца, или же имели место факты неполной загрузки вследствие обрыва соединения или перегруженности канала. Предположим, в приватной зоне сервера зарегистрирован пользователь с именем vasya. Давайте попробуем задать его имя в форме поиска: ' vasya '. Чего мы добились? Исчерпывающего отчета о том, какие ресурсы этот пользователь запрашивал, в какое время и сколько раз.

Можно привести не один десяток примеров использования скрипта для сбора самой различной «шпионской» информации, но для начала работы может помочь даже такой маленький, но полезный скрипт.

■ ■ ■ **Артемий Ломов**

Развиваем идею

Дополнения к скрипту

У нашего скрипта, конечно же, есть существенные недостатки, обусловленные чрезвычайной простотой решения. Во-первых, как вы понимаете, поиск заданной последовательности производится во всех полях лог-файла без разбора, и по запросу '404' в отчет могут быть включены «лишние» записи, в которых это число соответствует, скажем, количеству переданных сервером байт, а не коду статуса. Во-вторых, наш скрипт не позволяет строить сложные отчеты, проверяющие содержимое сразу нескольких полей. Например, нет никакой возможности выделить только такие записи, которые соответствуют запросам, порожденным пользователем vasya и повлекшим ошибку «401 Authorization Required».

Очевидно, для реализации подобных возможностей наш скрипт нужно усовершенствовать. К сожалению, объем публикации в журнале не позволяет в деталях рассмотреть процесс (и, самое главное, результат) модернизации, поэтому ограничимся лишь ключевыми моментами.

Прибегнув к помощи регулярных выражений, необходимо заменить пробелы, разделяющие соседние поля каждой обрабатываемой строки лог-файла, какими-либо другими разделителями, но так, чтобы пробельные символы внутри содержимого полей не были подвержены замене. После этого можно применить функцию split для преобразования строки в массив, элементами которого будут являться разобранные поля записи журнала доступа.

В HTML-форме надо предусмотреть не одно, а целых девять текстовых окошек (по одному на каждое поле). Обеспечив сравнение значения каждого поля формы с соответствующим элементом массива, мы получим возможность генерации сложных отчетов, о которых говорилось выше.