Indexing and full text searching documentation

Perl Diving

A Perl installation featuring the latest CPAN modules can easily have a few thousand manual pages. High time to define an index to support full text searching to aid our quest in finding precious knowledge.

BY MICHAEL SCHILLI

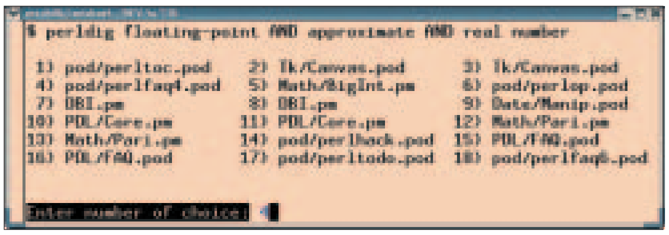
Now which Perl FAQ was it that explained why floating point operations sometimes produce horribly inaccurate results? *perldoc -q floating* will not be any help here as the question in the FAQs was

```
Why am I getting long ⮐
decimals (eg, 19.949999999999) ⮐
instead of the numbers I should⮐
 be getting (eg, 19.95)?
```

and it does not really contain any of the keywords you might expect. The *perldig* script that we will be looking at in this article makes things easier, as it churns its way through the manpages of your Perl distribution, including any additional modules you may have installed, and defines an index to facilitate command-line based full text searches that can handle things like

```
 perldig floating-point AND ⮐
 approximate AND real number
```

This search will return a numerical menu of all documents containing a combination of the above keywords (see Figure 1). After you make a selection, the *less* pager is launched to continue the search within that selection.



```
$ perldig floating-point AND approximate AND real number

  1) pod/perltoc.pod      2) Tk/Canvas.pod        3) Tk/Canvas.pod
  4) pod/perlfaq4.pod     5) Math/BigInt.pm       6) pod/perlop.pod
  7) DBI.pm               8) DBI.pm               9) Date/Manip.pod
 10) PDL/Core.pm         11) PDL/Core.pm         12) Math/Pari.pm
 13) Math/Pari.pm        14) pod/perlhack.pod    15) PDL/FAQ.pod
 16) PDL/FAQ.pod         17) pod/perltodo.pod    18) pod/perlfaq8.pod


Enter number of choice:
```

Figure 1: Old fashioned but practical: A simple numerical menu allows you to select documents that match your search

The indexing tool used is called *swish-e*, a successor to the original Swish search engine that was enhanced at Berkeley and released under the GPL. The original Swish tool was written way back in 1994 by Kevin Hughes. *swish-e* comprises an executable file called *swish-e* and a library with the Perl interface *SWISH::API*.

As explained at [3], you normally call *swish-e* from the command line to create an index that can be used for later full text searches:

```
swish-e -c mytree.conf
```

where *mytree.conf* is a configuration file with something like the following content:

```
# mytree.conf
IndexDir /documents/tree
IndexFile /path/mytree.index
UseStemming Yes
```

IndexDir sets the path to the root of the document tree to be indexed, *IndexFile* is the index file that *swish-e* creates (in fact this creates two files to be more precise, *mytree.index* and *mytree.index-prop*).

The *UseStemming* option tells *swish-e* to reduce the keywords to their stem form before adding them to the index, thus converting "floating" to "float", for example. A query for *floater* would also be reduced to *float*, and would find documents containing words like *floating*. Currently, this kind of function will only work reliably on English language documents. Instead of the path for the index, *IndexDir* will also accept an executable program or script:

```
# mytree.conf
IndexDir some_script
IndexFile /path/mytree.index
UseStemming Yes
```

If you call the indexer as follows:

```
swish-e -c mytree.conf -S prog
```

it will run *some_script* which is expected to split out all files to be indexed in this format:

```
Path-Name: file_name
Document-Type: TXT*
Content-Length: 12345

Text ...
```

Valid document types are: *TXT** (Text), *HTML**, and *XML**. *Content-Length* specifies the length of the text following after two carriage returns.

The *perldig* Perl script kills three birds with one stone: If you specify the *-u* (for "update") option, it creates a configuration file and sets the full path to the script itself as the *IndexDir*.

If you call the script without any options or arguments, it roots through all the directory trees containing core or module documentation in the current Perl installation, removes any POD markup, and sends the cleartext results and Swish headers to *STDOUT*.

If you specify one or multiple arguments, the script will assume that these arguments are search keys and launch a *swish-e* search operation to return a list of matching documents.

The Digger

The first module that *perldig* (see Listing 1) needs is the core module *Config.pm*, which uses the *installsitearch*, *installsitelib*, *installarchlib* and *installprivlib* $Config hash entries to return the paths to the modules installed in the current Perl installation.

The *arch* elements return processor dependent paths, *site* points to the *site_perl* branch, and *installprivlib* is usually something like */usr/lib/perl5/5.8.0*. The order of these entries is significant, as *perldig* will later try them in succession to compose the absolute module paths from relative ones. The longest path should come first, so a simple string substitution used later will work efficiently.

The *getopts()* function, which is exported from *Getopt::Std*, parses command line options. The only option it accepts is *-u*. The If clause starting at

Listing 1: perldig

```
001 #!/usr/bin/perl
002 ##############################
003 # perldig
004 # Mike Schilli, 2003
005 ############################'##
006 use warnings;
007 use strict;
008
009 use Config;
010 use SWISH::API;
011 use Shell::POSIX::Select;
012 use File::Basename;
013 use Getopt::Std;
014 use File::Path;
015 use File::Find::Rule;
016 use Pod::Simple::TextContent;
017
018 our $LESS  = "less";
019 our $SWISH = "swish-e";
020
021 our $IDX_FILE =
022  glob
    "~/.perldig/perldig.index";
023 our $CNF_FILE =
024  glob
    "~/.perldig/perldig.conf";
025
026 our @DIRS =
    ($Config{installsitearch},
027    $Config{installsitelib},
028    $Config{installarchlib},
029    $Config{installprivlib},
030    );
031
032 getopts("u", \my %opts);
033
034 if($opts{u}) {
035  update_index();
036 } elsif(@ARGV) {
037  search(@ARGV);
038 } else {
039  stream_files();
040 }
041
042 ##############################
043 sub search {
044 ##############################
045  my $term = join " ", @_;
046
047  my $swish = SWISH::API->
     new($IDX_FILE);
048
049  $swish->AbortLastError
050   if $swish->Error;
051
052  my $results = $swish->Query(
053    join ' ', @ARGV);
054
055  $swish->AbortLastError
056   if $swish->Error;
057
058  my $hits = $results->Hits;
059  if ( !$hits ) {
060   print "No Results\n";
061   return;
062  }
063
064  my @results = ();
065  my @select  = ();
066  my %map     = ();
067
068  while (my $r = $results
     ->NextResult) {
069  push @results, $r;
070  my $path = my $org_path =
071    $r->Property(
   "swishdocpath");
072  $path =~ s|^$_/|| for @DIRS;
073  push @select, $path;
074  $map{$path} = $org_path;
075  }
076
077  our($Eof, $Reply);
078
079  select my $file (@select) {
080   system "$LESS $map{$file}";
081   last;
082  }
083 }
084
085 ##############################
086 sub stream_files {
087 ##############################
088  my $rule = File::Find::Rule
089    ->file()
090    ->name('*.pod', '*.pm')
091    ->start(@DIRS);
092
093  while(my $file = $rule-
    >match()) {
094   print STDERR "Processing
    $file\n";
095   my $parser =
096    Pod::Simple::TextContent-
    >new();
097   my $output = "";
098   $parser-
    >output_string(\$output);
099   $parser->parse_file($file);
100   my $size = length($output);
101
102   print "Path-Name: $file\n",
103    "Document-Type: TXT*\n",
104    "Content-Length:
    $size\n\n";
105   print $output;
106  }
107 }
108
109 ##############################
110 sub update_index {
111 ##############################
112  if(! -e $CNF_FILE) {
113   print "Creating
    $CNF_FILE\n";
114   mkpath(dirname($CNF_FILE));
115   open FILE, ">$CNF_FILE" or
116    die "Can't open $CNF_FILE
    ($!)";
117   print FILE "IndexDir
    $0\n",
118     "IndexFile $IDX_FILE\n",
119     "UseStemming Yes\n";
120   close FILE;
121  }
122  system("$SWISH -c $CNF_FILE
    -S prog");
123 }
```

line 34 triggers one of the three actions described previously: Either updating the index (*update_index*), or performing a search (*search*), or parsing module files and outputting them with headers (*stream_files*).

glob "*~/...*" in lines 22 and 24 expands to the current user's home directory.

search() in line 43 first creates a new *SWISH::API* class object; its method, *AbortLastError*, terminates the program execution and issues an error message if the *Error()* method detects an error in the previous action.

In line 52 the *Query()* method passes a string containing a comma separated list of keywords. *Query()* returns an object as a result. Its *Hits()* method shows the number of *swish-e* hits.

The *while* loop starting at line 68 uses *NextResult()* to iterate through the results, which are subdivided by so-called properties. One of them, *swish-docpath*, specifies the path to the file matching the query.

Line 72 uses the following:

```
$path =~ s|^$_/|| for @DIRS;
```

to delete the absolute pathnames at the start of each file path, ultimately leaving something like *Log/Log4perl.pm*, although the complete path may have been something like:

```
/usr/lib/perl5/5.8.0/Log/➘
Log4perl.pm
```

Perldig uses the *%map* hash to store the associations between relative and absolute pathnames, for quickly retrieving a module file if the user selects it.

Line 79 contains a new construct dragged into Perl by Tim Maher's *Shell::POSIX::Select* module, including all the strings attached to it.

This is a port of the *select* Shell loop that is probably known only to UNIX veterans and which would otherwise wait for the user selecting a new item by number. The *our* statement squashed into line 77 declares a few variables thrown in by *Shell::POSIX::Select*, and makes the whole thing presentable in *use strict* mode.

The *system()* call in line 80 launches the *less* pager and outputs the matching file. *last* in line 81 terminates the *select* loop, which would otherwise expect a new numeric variable.

Streaming Files

The *stream_files()* function browses the file tree, starting with the directories specified in *@DIRS*.

The *File::Find::Rule* module, which behaves a bit differently than the ubiquitous *File::Find*, is used here. Instead of a callback function it defines a series of filters for the filesystem entries it finds, and only allows entries that match every filter to pass through. The *file()* rule in line 89 restricts the selection to files only (no directories or links). Line 90 drops anything that does not end with *.pod* or *.pm*, and line 91 tells the script to start doing this in the directories defined in *@DIRS*.

The *while()* loop starting in line 93 repeatedly calls the *match()* method of the *File::Find::Rule* object, and thus iterates over all module files found. As we will be indexing only the text content and not the POD markups, we also need a *Pod::Simple::TextContent* parser. The parser's *output_string()* method defines a string, which it uses to store its output. *parse_file()* in line 99 launches the parser, which then plows through the current module file hit.

The *print* statement in line 102 outputs the header required by *swish-e* for the ensuing document; the length of the document in bytes is ascer-tained in line 100 by a call to *length()*.

Updating the Index

The *update_index()* function, defined in line 110 and following, helpfully creates a configuration file for *swish-e* if the file does not exist. The *system()* call in line 122 then ensures that *swish-e* is launched and creates a new index based on the settings in the configuration file.

This can take a few minutes, depending on the number of documents that need indexing, and this is why the name of document currently being processed is output to *STDERR* in line 94.

If you run *perldig -u* as a cronjob, you will want to send this output straight to the trashcan, as follows:

```
0 4 * * * /path/perldig -u >➘
/dev/null 2>&1
```

perldig can also search for combinations of words linked by *AND* and *OR*, as in:

```
perldig local AND ➘
'"input record"'
```

If a query contains blank-separated word combinations, don't forget the double quotes – if you want your input to go to *swish-e* rather than falling victim to the Shell. You also need to protect parentheses:

```
perldig "(override OR overload)➘
 AND object-oriented"
```

returns all OO documents that contain either the *override* or *overload* keywords.

Installation

The *SWISH-E* distribution is available as a tarball from [2], and can be installed using the usual steps:

```
./configure
make install
```

The tarball also contains the *SWISH::API* Perl module. To install this just step down to the perl subdirectory:

```
cd perl
perl Makefile.PL
make install
```

You may need to modify the configuration variables in lines 18 through 24 of *perldig*. After that you need to create an index using *perldig -u* (*u* for update, remember?) before you start searching:

```
perldig obfuscated
```

returned an astonishing number of nine documents: Hidden pieces of Perl humour! ■

INFO

[1] Listings for this article: *ftp://www.linux-magazin.de/pub/listings/magazin/2003/10/Perl*

[2] Swish search engine Homepage: *http://swish-e.org*

[3] "How to Index Anything", Josh Rabinowitz, Linux-Journal 07/2003, *http://www.linuxjournal.com/article.php?sid=6652*