Hogyan indexeljünk?

Honlapunkon valószínűleg akad valamilyen keresőfelület – de mi a helyzet a súgóoldalak vagy a levelezőrendszer keresésével?

Számtalan oka lehet annak, ha indexeket szeretnénk rendelni a dokumentumokhoz. Az egyik ilyen gyakran emlegetett ok a honlapok keresőfelülete, de ugyanígy elképzelhető, hogy valaki a levelezését vagy a műszaki dokumentumait szeretné indexelni. Aki próbált már ilyen rendszert készíteni, az valószínűleg jól tudja, hogy egyáltalán nem olyan egyszerű dolog, mint amilyennek első látásra gondolnánk. Számos esemény jöhet közbe, ami megakadályozza a hatékony munkát.

A vénséges vén és nélkülözhetetlen `grep` és társai igen hatékonyan keresnek szöveges adatokban. A `grep`, az `egrep` és rokonaik azonban nem tudnak bármit megadni nekünk. Nem keresnek több soron keresztül, nem tudnak rangsorolt keresési eredményt megjeleníteni, és lineáris keresési algoritmusuk nem igazán teszi alkalmassá őket a nagyobb dokumentumokkal való munkára.

A HTML sem lendít sokat a dolgon. Megjelenítés-központú képességei, egyedi nyelvezete, a formázási és entitástagok serege igencsak megnehezíti a helyes értelmezést.

Az adattárolási skála másik végén az adatbázisokba szervezett adatok állnak. Mindenhol megtalálható példa erre az SQL adatbázis, amely viszonylag kifinomult keresési eszközökkel rendelkezik, de többnyire nem fut igazán gyorsan. Néhány adatbázismotor – kiemelten a MySQL 4 – a kérdést gyors és rendszerezett (ranked) kereséssel próbálja megoldani, viszont nem lehet a kívánt mértékben testreszabni.

Ebből a cikkből megtudhatjuk, miképpen készíthetjük el a saját indexeinket Linux alatt a SWISH-E, a Perl és az XML segítségével. A példákon keresztül bemutatjuk, hogyan használhatjuk a SWISH-E programot HTML-fájlok, PDF-fájlok és súgóoldalak indexének a felépítéséhez.

A SWISH-E (Simple Web Indexing System for Humans – Enhanced) az 1994-ben ***Kevin Hughes*** készítette SWISS utódja. A SWISH 1996-ban hibajavítás és új képességek hozzáadása céljából átkerült az UC Berkeley könyvtárba. Az eredményt GPL alatt adták ki, és SWISH-E-re nevezték át. A fejlesztések ***Bill Moseley***, a jelenlegi projektkarbantartó vezénylete mellett tovább folytatódnak, Billt fejlesztők egész csoportja segít a munkájában.

Mi a SkateboardDirectory.com-nál akkor akadtunk rá a SWISH-E-re, amikor indexelő eszközkészletet kerestünk. Rájöttünk, hogy egyedülálló képességekkel bír: a SWISH-E nemcsak gyors és erőteljes eszközkészletet kínál, amivel indexeket építhetünk, illetve kérdezhetünk le, de egyben kiválóan dokumentált, folyamatos fejlesztés és hibajavítás alatt áll, és Perl csatolófelülettel is rendelkezik. Az is nagyon tetszett, hogy a csomagfelelős Moseley és a többi tapasztalt SWISH-E-felhasználó és -fejlesztő általában igen gyors és pontos választ ad, ha a SWISH-E-levelezőlistán kérdések vagy hibák merülnek fel.

A SWISH-E telepítése

Példánkat egy gyári Red Hat 7.3 munkaállomáson futtattuk, amire a Software Development csomagcsoportot telepítettük fel. A példákat Red Hat 6.2-t futtató munkaállomáson, valamint Debian Woody alatt is kipróbáltuk.

Jelenleg Red Hat alatt csak forrásból rakhatjuk fel a SWISH-E rendszert, továbbá felépítéséhez előzőleg a ***zlib*** és ***libxml2*** könyvtárakat is fel kell telepítenünk. Ha úgy tűnik, hogy valamelyiket fel kell raknunk, a terjesztésünk csomagjai között valószínűleg megtaláljuk. Példáinkban az `xpdf` csomagot is használni fogjuk, így ezt is érdemes feltelepíteni, ha esetleg még nem tettük volna meg. Az általunk megjelölt Red Hat 7.3 munkaállomás az összes SWISH-E rendszerhez szükséges valamennyi előtelepítési feltételt teljesíti.

A következőkben bemutatjuk a SWISH-E 2.4 használatát, ami a fejlesztőcsapat szerint körülbelül a cikk megjelenésével egy időben lesz majd elérhető. A SWISH-E rendszert a következő parancsok sorozatával telepíthetjük (az x.x helyére az időszerű változatszám kerül):

```
% wget http://swish-e.org/Download/
➥swish-e-x.x.tar.gz
% tar zxf swish-e-x.x.tar.gz
% cd swish-e-x.x
% ./configure
% make
% make test
```

Ha a SWISH-E futtatható állományt, a C-könyvtárakat és a súgóoldalakat az alapértelmezett ***/usr/local*** könyvtárba szeretnénk helyezni, rendszergazdaként gépeljük be a `make install` parancsot. Így a SWISH-E végrehajtható állomány a ***/usr/local/bin*** könyvtárba kerül. Amennyiben ez a könyvtár még nincs a `PATH` útvonalunkban, két dolgot tehetünk: vagy módosítjuk a megfelelő ponttal kezdődő fájlt, hogy a ***/usr/local/bin*** könyvtárat is helyezze a `PATH` változóba, vagy teljes útvonalával együtt hívjuk meg a `swish-e` végrehajtható állományt: ***/usr/local/bin/swish-e***.

Most készítsük el és telepítsük a forrás Perl-könyvtárában található `SWISH::API` Perl-modult. Később még szükségünk lesz rá, amikor a súgóoldalaink indexeihez használt Perl-ügyfelet készítjük el. A `SWISH::API` modult a szokásos Perl modultelepítési folyamattal tehetjük fel:

```
% cd perl
% perl Makefile.PL
% make
% make test
```

Ezt követően a `make install` parancsot rendszergazdaként begépelve telepíthetjük a `SWISH-E` Perl-modult.

Miután a `SWISH-E` és a `SWISH::API` Perl-modult teljesen feltelepítettük, a SWISH-E kipróbálásához a HTML-fájlok nyilvántartására készítsünk egy egyszerű indexet. Ebben a példában az Linux Documentation Project (LDP) HOGYAN-

1. lista Az sman-index-prog.pl az indexeléshez súgóoldalakat alakít XML formátumúvá

```
#!/usr/bin/perl -w

use strict;
use File::Find;

my ($cnt, @files) = (0, get_man_files());
warn scalar @files, " man pages to
                    ➥index...\n";
for my $f (@files) {
   warn "processing $cnt\n" unless ++$cnt % 20;
   my ($hashref) = parse_man($f);
   my $xml = make_xml($hashref);
   my $size = length $xml;
   print "Path-Name: $f\n",
       "Document-Type: XML*\n",
       "Content-Length: $size\n\n", $xml;
}

sub get_man_files {  # angol kézikönyvoldalak
                     # keresése
    my @files;
    chomp(my $man_path = $ENV{MANPATH} ||
      `manpath` || ´/usr/share/man´);
    find( sub {
      my $n = $File::Find::name;
      push @files, $n
      if -f $n && $n =~ m!man/man.*\.!
   }, split /:/, $man_path );
   return @files;
}
sub make_xml { # a tömb XML-változat kiírása
   my ($metas) = @_;
   my $xml = join ("\n",
   map { "<$_>" . escape($metas->{$_}) .
         ➥"</$_>" }
   keys %$metas);
   my $pre = qq{<?xml version="1.0"?>\n};
   return qq{$pre<all>$xml</all>\n};
}
sub escape { # az átadott számokat módosítja!
   return "" unless defined($_[0]);
   s/&/&amp;/g, s/</&lt;/g, s/>/&gt;/g
   for $_[0];
   return $_[0];
}

sub parse_man {    # ez a lényeg
   my ($file) = @_;
   my ($manpage, $cur_content) = (´´, ´´);
   my ($cur_section,%h) = qw(NOSECTION);
   open FH, "man $file  | col -b |"
   or die "Failed to run man: $!";
   my ($line1, $lineM) =
      ➥(scalar(<FH>) || "", "");
   while ( <FH> ) {  # kézikönyvoldalak
                     # szakaszolása
      $line1 = $_ if $line1 =~ /^\s*$/;
      $manpage .= $lineM = $_ unless /^\s*$/;
      if (s/^(\w(\s|\w)+)//
         ➥|| s/^\s*(NAME)/$1/i){
         chomp( my $sec = $1 );
         $h{$cur_section} .= $cur_content;
         $cur_content = "";
         $cur_section = $sec; # új szakasz név
      }
      $cur_content .= $_ unless /^\s*$/;
   }
   $h{$cur_section} .= $cur_content;

   # megvizsgálandó a NAME, HEAD, FOOT,
   # (és esetleg a fájlnév is).
   close(FH) or die "Failed close on pipe
                    ➥to man";
   @h{qw(A_AHEAD A_BFOOT)} = ($line1, $lineM);
   my ($mn, $ms, $md) = ("","","","");
   # NAME mn, DESCRIPTION md, & SECTION ms
   for(sort keys(%h)) { # először A_AHEAD  #
                        & A_BFOOT
      my ($k, $v) = ($_, $h{$_}); # key&val
                                  # másolása
      if (/^A_(AHEAD|BFOOT)$/) {
          # look for the ´section´ in ()´s
         if ($v =~ /\(([^)]+)\)\s*$/)
            ➥{$ms||= $1;}
            } elsif($k =~
            ➥s/^\s*(NOSECTION|NAME)\s*//) {
         my $namestr = $v || $k;
         if ($namestr =
            ➥~ /(\S.*)\s+--?\s*(.*)/) {
            $mn ||= $1 || "";
            $md ||= $2 || "";
         } else { # ez a regex hibázhat.
            $md ||= $namestr || $v;
         }
      }
   }
   if (!$ms && $file =~ m!/man/man([^/]*)/!) {
      $ms = $1; # a  sec-et a path-ból
                # vesszük ha nem található
   }
    ($mn = $file) =~ s!(^.*/)|(\.gz$)!!
    ➥unless $mn;
    my %metas;
    @metas{qw(swishtitle sec desc page)} =
       ($mn, $ms, $md, $manpage);
    return ( \%metas ); # ref visszaadása 5-
                        # kulcsos tömbnek.
}
```

jainak egy oldal/fejezet alapú változatának HTML-oldalait fogjuk indexelni, ami a *~/HOWTO-htmls/* könyvtárba kerül. A cikkben felhasznált LDP-dokumentumok a ➲ http://www.tldp.org/docs.html oldalról származnak.

HTML indexelése a fájlrendszeren

A SWISH-E alapú index létrehozásának első lépése a beállításfájl elkészítése. Hozzunk létre egy *~/indices* nevű könyvtárat, majd lépjünk bele, és a következőket írjuk be egy

./howto-html.conf nevű állományba:

```
# howto-html.conf
IndexDir  ../HOWTO-htmls/

IndexOnly .html

IndexFile ./howto-html.index
```

Az `IndexDir` parancs adja meg azt a könyvtárat, ahol a `SWISH-` az indexelendő fájlokat keresi. Az `IndexOnly` utasítás jelzi, hogy csak a *.html* végződésű állományokat kell indexelni. Végül a létrehozandó index helyét a `IndexFile` utasítás adja meg.

Az első indexünk

A következő paranccsal készítsük el HTML-fájlindexünket:

```
% swish-e -c howto-html.conf
```

A `-c` kapcsoló adja meg, hogy melyik SWISH-E beállításfájlt kell felhasználni. Régebbi rendszereken az index elkészítése akár perceket is igénybe vett, egy mai gépen azonban egy perc alatt végezni kell. Az *1. ábra* a HTML-fájlok SWISH-E alapú indexelésének folyamatát mutatja be a fájlrendszeren.

Keresés az indexben

Próbáljuk ki az új indexünket, és végezzünk el egy egyszerű keresést, ami megadja az NFS-kifejezéssel kapcsolatos HTML-fájlokat. A SWISH-E indexeket a `swish-e` végrehajtható állomány segítségével gyorsan és könnyen tesztelhetjük, ha az indexet a `-f` kapcsolóval adjuk meg, majd a keresett szöveget a `-w` kapcsoló után írjuk. A SWISH-E indexek keresései kis- és nagybetűérzékenyek. Mivel igen sok oldalt (vagy találatot) várunk, ami tartalmazza az NFS szót, a `-m 3` kapcsolóval háromra korlátozzuk a keresések számát:

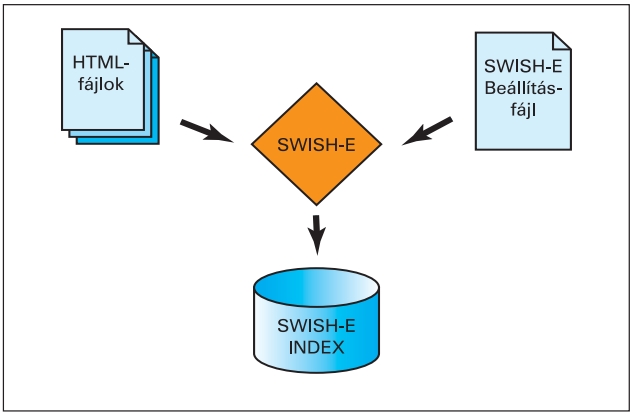
```
% swish-e -f howto-html.index -m 3 -w nfs
```

Az előző sor a következőket adja vissza (rövidítve és újraformázva):

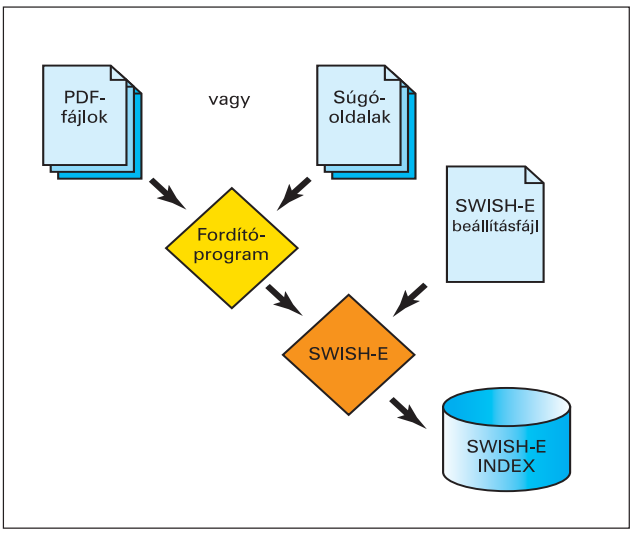
```
1000 ../HOWTO-htmls/NFS-HOWTO/performance.html
       "Optimizing NFS Performance" 33288
998 ../HOWTO-htmls/NFS-HOWTO/intro.html
       "Introduction" 10966
993 ../HOWTO-htmls/NFS-HOWTO/security.html
       "Security and NFS" 35968
```

Nem rossz – ezek az oldalak egyértelműen az NFS-ről szólnak, és a kimenet is intuitív. Az első oszlop a SWISH-E által adott helyezés. A leginkább ide tartozó találatok mindig 1000-es helyezést kapnak, míg a kevésbé ide tartozóak egyre kisebb rangot. A második oszlop a fájlnevet mutatja, a harmadik az oldal címét adja meg, míg a negyedik az indexelt adat bájtszámlálója. A SWISH-E egyik HTML-értelmező motorjának segítségével a HTML-tagokból minden egyes oldal címét meghatározza.

A beépített SWISH-E értelmező motorok neve TXT, HTML és XML. Valamennyi értelmező a nevének megfelelő adattípus kezelésére képes. A SWISH-E jelenlegi változatai már képesek a *libxml2* könyvtár HTML2 és XML2 értelmező hátterét is használni. A beépített változatokkal szemben inkább az XML2 és a HTML2 értelmező használatát javaslom, különösen igaz ez a HTML2 esetében. Ez az oka annak, hogy a *libxml2*, bár



1. ábra HTML indexelése a fájlrendszeren SWISH-E rendszerrel



2. ábra Tetszőleges adatok indexelése külső programmal és SWISH-E-vel

technikailag csak javasolt rész a SWISH-E létrehozásakor, majdhogynem előkövetelmények számít.

A SWISH-E keresési formátumának alapjai

A SWISH-E teljes értékű szövegkereső nyelvvel rendelkezik, amelynek formátumában megtalálható az `AND`, `OR`, `NOT` logikai szerkezet és a zárójelezés. Valamennyi az elvárásoknak megfelelően működik. Például a következő kereséseknek logikus formátumú a szerkezete:

```
% swish-e -f howto-html.index -w nfs AND tcp
% swish-e -f howto-html.index -w nfs OR tcp
% swish-e -f howto-html.index -w '(gandalf OR
frodo) OR (lord AND rings)'
```

A beállításfájl

A SWISH-E beállításfájljai egyszerű szöveges állományok, amelyekben minden egyes sor parancs vagy megjegyzés lehet. Azokat a sorokat, amelyekben az első nem üres karakter a # (kettős kereszt), a SWISH-E megjegyzésként figyelmen kívül hagyja. Minden más nem üres sornak a következő formátumúnak kell lennie:

```
Utasítás kapcsoló [érték] ...
```

Amennyiben üres karaktereket tartalmazó kapcsolót kell megadnunk, idézőjeleket használunk:

```
Utasítás "kapcsoló üres karakterekkel!"
```

Amennyiben a kapcsoló egyszeres idézőjelet (aposztrófot) tartalmaz, a kettős idézőjelet használjuk, és vice versa, például:

```
Utasítás "Fred´s Index Option"
Utasítás ´Josh "joshr" Rabinowitz alkotása´
```

(és ha a szöveg mindkét fajta idézőjelet tartalmazza, akkor sajnáljuk: – a ford.)

A SWISH-E beállításfájljaiban tucatnyi parancsot adhatunk meg, a SWISH-E leírásában kimerítő listát találunk róluk.

Az index

Minden SWISH-E index egy fájlpárban tárolódik. Az egyik fájl neve az `IndexFile` utasítás alapján készül, a másik neve mindig ***indexname.prop*** lesz. Ha SWISH-E indexről beszélünk, mindig erre a fájlpárra gondolunk.
Az indexek elég nagyra nőhetnek. Az iménti HTML-fájlindexelő példánkban használt index 11 MB-ot foglalt el – ez körülbelül az egynegyede az eredeti, beindexelt állományok méretének.

PDF-állományok indexelése

Egészen mostanáig csak a HTML-, XML- és szövegfájlok indexeléséről beszéltünk. Nézzünk meg egy haladóbb példát: indexeljünk be a Linux Documentation Project PDF-dokumentációit.
Ahhoz, hogy a SWISH-E tetszőleges (PDF- vagy egyéb) fájlokat indexelhessen, az állományokat előbb szöveges fájlokká kell alakítani, lehetőleg HTML vagy XML alakúra, majd a SWISH-E segítségével indexelnünk kell az eredményt.
A PDF-fájlokat tehát úgy indexelhetjük, ha a lemezen mindegyiket átalakítjuk, majd indexeljük őket. Ehelyett azonban inkább megragadjuk az alkalmat, és egy rugalmasabb indexelési megoldást mutatunk be: a SWISH-E programozott elérési módszerét (2. *ábra*).
A PDF-állományok indexelését a SWISH-E beállításfájl létrehozásával kezdjük. Nevezzük el ***howto-pdf.conf***-nak, majd a következő tartalommal töltsük fel:

```
# howto-pdf.conf
IndexDir       ./howto-pdf-prog.pl
                 # prog file to hand us XML
                 # docs
IndexFile      ./howto-pdf.index
                 # Index to create.
UseStemming    yes
MetaNames      swishtitle swishdocpath
```

Itt az `IndexDir` utasítás most azt jelenti, hogy a SWISH-E e külső program meghívásával kapja meg az indexelendő állományok adatait, és nem a könyvtárban található fájlokat használja. A `UseStemming yes` utasítás hatására a SWISH-E indexelés vagy keresés előtt kikeresi a szavak tövét. E nélkül a szolgáltatás nélkül a „runs” szó keresésekor a „running” szót tartalmazó dokumentumokat nem találná meg. A szótőkereséssel a SWISH-E felismeri hogy a „runs” és „running” szavak töve azonos, és megtalálja a megfelelő dokumentumokat.

Beállításfájlunk utolsó, de nem kevéssé fontos bejegyzése a `MetaNames` utasítás. Ez a sor különleges képességgel ruházza fel az indexünket: lehetőségünk nyílik csak a címekben vagy fájlnevekben keresni.
Írjuk meg az indexelendő PDF-fájlokról adatokat visszaadó külső programot! Hagyományosan a SWISH-E forrásával együtt egy ***pdf2xml.pm*** nevű példamodult is kapunk, ami az xpdf csomagot használva PDF-fájlokat alakít át XML formátumúvá, megjelölve őket a SWISH-E által használható előtagokkal. Ezt a modult (a ***~/indices*** könyvtárba másolva) a ***howto-pdf-prog.pl*** nevű külső programunkban felhasználhatjuk:

```
#!/usr/bin/perl -w
use pdf2xml;
my @files =
    `find ../HOWTO-pdfs/ -name ´*.pdf´
     ➥-print`;
for (@files) {
    chomp();
    my $xml_record_ref = pdf2xml($_);
    # Ez egy XML fájl
    # SWISH-E fejléc
    print $$xml_record_ref;
}
```

Miután a SWISH-E beállításfájllal és a fenti külső programmal felvérteztük magunkat, készítsük el az indexet:

```
% swish-e -c howto-pdf.conf -S prog
```

A `-S prog` kapcsoló mutatja meg a SWISH-E-nek, hogy az `IndexDir` valójában egy program, ami az indexelendő adatokról ad vissza információkat. Amennyiben a `-S prog` kapcsolót elfelejtjük megadni, miközben külső programot használunk a SWISH-E-hez, magát a külső programot fogjuk indexelni és nem az általa leírt dokumentumokat.
Miután a PDF-index elkészült, próbáljunk ki egy keresést:

```
% swish-e -f howto-pdf.index -m 2 -w boot
disk
```

Ilyesféle eredményt kell kapnunk:

```
1000 ../HOWTO-pdfs/Bootdisk-HOWTO.pdf
       "Bootdisk-HOWTO.pdf" 127194
983 ../HOWTO-pdfs/Large-Disk-HOWTO.pdf
      "Large-Disk-HOWTO.pdf" 85280
```

A `MetaNames` utasítás alkalmazása folytán címek és a PDF-fájlok útvonala szerint is kereshetünk:

```
% swish-e -f howto-pdf.index -w
swishtitle=apache
% swish-e -f howto-pdf.index -w
swishdocpath=linux
```

A keresések minden kombinációja támogatott, például:

```
% swish-e -f howto-pdf.index -w ´(larry and
wall)
    OR (swishdocpath=linux OR
swishtitle=kernel)´
```

Az előbbi példában azért szükséges idézőjeleket használnunk, hogy a zárójeleket megóvjuk a héjprogram értelmezésétől.

Súgóoldalak indexelése

Utolsó példánkban megmutatjuk, hogyan készíthetünk súgóoldalainkhoz hasznos és hatékony indexeket, és miként használhatjuk a `SWISH::API` Perl-modult az index-keresőfelület kialakításához. Ahogy eddig is, a munkát a beállításfájl elkészítésével kezdjük:

```
# sman-index.conf
IndexFile ./sman.index
  # elkészítendő index.
IndexDir  ./sman-index-prog.pl
IndexComments no
  # a megjegyzések szövegeit nem indexeljük
UseStemming yes
MetaNames      swishtitle desc sec
PropertyNames              desc sec
```

A legtöbb utasítás jelentését már korábban bemutattuk, most azonban néhány új `MetaNames` nevet is megadtunk, illetve egy új, `PropertyNames` nevű parancsot vezettünk be. Dióhéjban összefoglalva: a `MetaNames` határozza meg, hogy mi alapján keressen a SWISH-E. Az alapértelmezett `MetaName` a `swishdefault`, azaz ha a lekérdezésben nem adunk meg `MetaName` kapcsolót, akkor ez alapján fog keresni. A `PropertyNames` a visszaadott találatok leírására használható mező.
A SWISH-E által visszaadott eredmény általában az `Auto Properties` (önműködő tulajdonságok parancs) szerint jelenik meg, ilyen a `swishtitle`, `swishdesc`, `swishrank` és a `swishdocpath`. A beállításunkban olvasható `MetaNames` utasítás azt jelenti, hogy egymástól függetlenül nemcsak a teljes dokumentum, hanem csak a címek, a leírás vagy a szakasz alapján is keresni szeretnénk. A `PropertyNames` sor mutatja meg, hogy minden egyes találat visszaadásakor a `sec` és `desc` tulajdonságot szeretnénk látni, vagyis az oldal szakaszát (`sec`) és rövid ismertetőjét (`desc`).
A súgóoldalak XML formátumúvá alakítását és SWISH-E fejlécekké formázását az *1. listában* látható módon végezzük (***sman-index-prog.pl***).
Az 1. lista első ciklusa lesz a program fő váza. Itt pillantunk bele minden egyes súgóoldalba, szükség szerint értelmezzük, XML-é alakítjuk át, majd a SWISH-E igényeinek megfelelő fejlécekkel egészítjük ki:

- A `get_man_file()` a `File::Find` függvényt használja a súgóoldalak könyvtáraiban, hogy megtalálja a feldolgozandó súgófájlokat.
- A `make_xml()` és az `escape()` együtt a `parse_man()` által visszaadott `hashref`-ből készíti el az XML formátumot.
- A `parse_man()` végzi el a trükkös részt, kiemelve a szükséges mezőket a súgóoldal forrásából.

Most, hogy már megismertük a működését, használjuk is aprogramunkat:

```
% swish-e -c sman-index.conf -S prog
```

Miután ezzel megvagyunk, a `swish-e -w` kapcsolóját alkalmazva a korábbiakhoz hasonló módon kipróbálhatjuk a rendszert.

Végül nézzük meg, hogyan használhatunk `SWISH::API`-t alkalmazó Perl-parancsfájlokat az imént elkészített indexhez, létrehozva ezzel a unixos `apropos` parancs egy fejlettebb változatát. A kódot a 2. listában (51. CD Magazin/SWISH-E könyvtárában) találjuk (sman). Lássuk röviden a szerkezetet: az 1–14. sor az alapértékeket állítja be és a parancssori kapcsolókat értelmezi, a 15–23. sor kezeli a lekérdezést, illetve felületes hibakezelést végez, végül a 24–39. sor jeleníti meg a lekérdezés eredményét a `SWISH::API`-n keresztül visszakapott `Properties` alapján.
A Perl-ügyfél ilyen egyszerű. Próbáljunk meg súgóoldalainkból kikeresni egy témát:

```
% ./sman -m 1 boot disk
```

Az alábbit kell visszakapnunk:

```
ootparam (7) Introduction to boot time para...
```

Ugyanakkor már a következőképpen is kereshetünk:

```
% ./sman sec=3 perl
```

Ezáltal a keresést a 3. szakaszra korlátozzuk. Továbbá az `sman` program elfogadja a `--max=#` parancssori kapcsolót, ami korlátozza a visszaadott találatok számát, a `--file` kapcsolót, ami megmutatja a keresett szavakat tartalmazó fájl nevét, és a `--rank` kapcsolót, ami az adott lekérdezésen belül kapott rangot mutatja meg:

```
% ./sman --max=1 --file --rank boot
```

A fenti parancs a következő eredményt adja:

```
1000 lilo.conf (5) configuration file for lilo
     /usr/man/man5/lilo.conf.5
```

Figyeljük meg, hogy a rang az első, míg a forrásfájl az utolsó oszlopba kerül.
A sman csomag fejlettebb változatát a
➲ http://joshr.com/src/sman/ címen érhetjük el.

Összefoglalás

A SWISH-E rendszerének van két említésre érdemes árnyoldala is. Először is: csak a 8 bites ASCII adatokat kezeli. Másodszor: a SWISH-E indexből nem lehet elemeket törölni, a törléshez a teljes indexet újra létre kell hozni. A mérleg másik serpenyőjében a SWISH-E számos olyan képességét találjuk, amit itt még csak megemlíteni sem tudtunk. A részleteket a SWISH-E honlapján, a ➲ http://www.swish-e.org címen találhatjuk meg.

A cikkhez tartozó listák megtalálhatóak az 51. CD Magazin/SWISH-E könyvtárában.

Linux Journal 2003. július, 111. szám

Josh Rabinowitz
13 éve a programipar veteránja, aki tudását a NASA Ames-i kutatóközpontjában, illetve a CNET.com-nál és egyéb webcégeknél csiszolgatta. Jelenleg független tanácsadó és a SkateboardDirectory.com kiadója.