Felületfüggetlen CD-tárgymutató

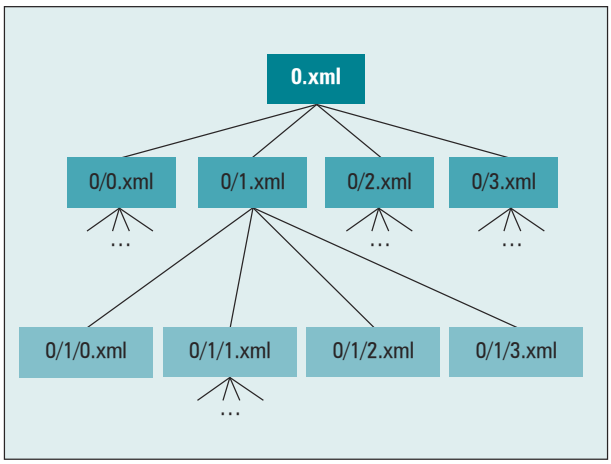
Jó lenne a CD-ROM-ok tartalmát egy olyan keresővel kereshetővé tenni, amelyik állandó indexfájlok használatával bármelyik böngészőben képes futni. A JavaScript és az XML ezt lehetővé teszi.

Nemrég az egyik ügyfelem számára CD-katalógust készítettem. Az volt a feladatom, hogy kulcsszavakra lehessen rákeresni. Már meglévő megoldások után kutattam, de sajnos csak olyanokat találtam, amelyek egy adott zárt kódú operációs rendszer alatt futottak, telepíteni kellett őket a felhasználó számítógépére, és minden példány után használati díjat kellett fizetni. Az ilyen telepítési feltételek korlátozóak és hosszú távon nagyon sokba kerülnek. Továbbá nem minden CD-felhasználó futtatja azt a bizonyos zárt kódú operációs rendszert, így a lehetséges felhasználók táborát mesterségesen csökkentenénk. Miközben a feladat megoldásán rágódtam, felfigyeltem a Linux Journal Archive CD-re. Rájöttem, hogyha valaki már megoldotta ezt a feladatot, az biztosan szerepel az LJ Archive CD-n. Képzelhetitek a csalódásomat, amikor kiderült, hogy ugyan az LJ Archive CD-nek van tárgymutatója, de nem kereshető. Nekem kell megoldani a feladatot, magamnak kell rábukkannom a megoldásra is. Szerencsére rátaláltam a jsFindre.

Felhasználási feltételek

A jsFind terjesztésre és felhasználási feltételeire már viszonylag korán gondoltam. A korai változatokat megmutattam munkatársaimnak, akik azt javasolták, hogy kövessem a szokásos módszert: előbb védessem le a kódot és utána próbáljam eladni. Ez azt jelentette volna, hogy a jsFind ugyanazt a gyakorlatot követi, mint versenytársai. Szívesebben foglalkozom kódolással, mint eladással, és amúgy sem hiszem, hogy a terméknek túl nagy piaca lenne. Inkább legyenek olyan CD-ROM-ok, amelyeknek a tartalmában az összes böngészővel bármelyik operációs rendszer alatt lehet keresni.

A GPL (GNU Public License) sokkal inkább megfelelt a céljaimnak. Ha a jsFind szabadon terjeszthető, akkor a saját érdemei miatt fog elterjedni, és a közösség további fejlesztésekkel fog hozzájárulni a kódhoz. A zárt kódú rendszerek egyik célja, hogy a felhasználót bezárják az adott program börtönébe. Ha például egy CD-ROM-hoz mellékelt keresőt csak egyetlen böngészővel és egy adott operációs rendszer alatt lehet használni, akkor a felhasználó kénytelen azt a bizonyos operációs rendszert futtatni. A CD-ROM előállítójának muszáj lesz mindig megvennie az adott operációs rendszerhez való fejlesztőeszközöket, hogy naprakész maradjon. A folyamat oda vezet, hogy a gyártók és a felhasználók nem tudnak mást használni, csak azt a bizonyos zárt kódú operációs rendszert. A GPL feltételei szerint kiadott jsFind megtöri ezt az ördögi kört.



A jsFind B-fát hoz létre, amelyben minden csomópont egy XML-fájl

A megvalósítás

A jsFind kulcsszókereső-motor egy kis, ötszáz soros JavaScript-program. Minden böngésző, amelyik támogatja a DOM Level 3 JavaScript-bővítményeket, képes betölteni az XML-fájlokat. A Mozilla, a Netscape és a Microsoft Internet Explorer jelenlegi változatai mind támogatják ezeket a kiterjesztéseket, és a Konqueror következő változata is valószínűleg fogja. A tárgymutató XML-fájlokban tárolódik, a JavaScript pedig hatékonyan keres bennük, és előállítja az eredményt. A keresés eredményeit az őt lekérő weboldalra szintén JavaScript postázza vissza.

A jsFind tekintetbe veszi, hogy a CD-ROM tartalma állandó – a webbel vagy más változó adathalmazzal ellentétben a CD-ROM tartalma a legyártása után nem fog megváltozni. A SWISH-E jobban használható dinamikus indexelésre, különösen, ha valaki olyan szerencsés, hogy külön

A keresési példa eredménye

kiszolgálót állít be a kulcsszókeresésre. A jsFind ezért csak azt feltételezi, hogy adott egy JavaScriptet ismerő webböngésző és egy pár böngészhető fájl – ez jelentős megszorítást jelent a lehetséges megoldásokra nézvést.
A legtöbb indexelő algoritmus a beszúrás, a frissítés, a törlés és kiválasztás ideje közötti egyensúly megteremtésén fáradozik. Mivel a CD-ROM megváltoztathatatlan, soha nem kerül sor törlésre vagy frissítésre. A beszúrásra a CD megírása előtt kerül sor és az időigénye érdektelen. A felhasználó szemszögéből a kiválasztás ideje lesz a legfontosabb. A kis tárhelyigény is követelmény, hiszen egy átlagos CD-re 700 MB-nál több anyag nem fér fel.
A feltételek figyelembevételével az indexelő algoritmusokat ismét megvizsgálva érdekes eredményre jutunk: a leggyakrabban B-fákat és hasítótáblákat (hash) használnak. Azért választottam a B-fákat, mert a fájlrendszer a fájlokat faszerkezetbe rendezve tárolja, és ezt felhasználhatjuk a B-fa szerkezetének a tárolására is, helyet takarítva meg ezáltal. Ezenkívül elemezni lehet a kulcs–hivatkozás párokat és kiegyensúlyozott B-fát lehet létrehozni. Az XML-fájlok szerkezete a lehető legegyszerűbb, a helyfoglalás további csökkentésének céljából egybetűs elemeket használtam.

A B-fák leírása

A B-fa elnevezésű adatszerkezetet gyakran használják adatbázisok indexeléséhez és tárolóeljárásokhoz. Hatékonyan, kis időigénnyel kereshető, és a tárolást, illetve a visszakeresést tömbösen oldották meg, ez pedig jól együttműködik a mostani vasakkal. A B-fa csúcsokból áll, amelyeknek rendezett kulcslistájuk van. Minden kulcs egy hozzárendelt adathalmazra hivatkozik. Ha a kért kulcs a rendezésben két kulcs közé esik, akkor hivatkozást kapunk vissza egy másik kulcsokból álló csúcsra. A kiegyensúlyozott B-fa olyan, amelynél a keresésnél betöltendő csúcsok legnagyobb száma kicsi.
A jsFind B-fát hoz létre, amelyben az XML-fájlok jelentik a fa csúcsait és a fájlrendszer könyvtárai a hivatkozások egy másik csúcshalmazra. Ez lehetővé teszi, hogy a B-fa szerkezetének egy részét a fájlrendszer valósítsa meg.

1. lista Exportálás XML-be

```
$ swish-e -f mystuff.index -T INDEX_XML >
➥mystuff.xml
```

Ennek az XML-nek a szerkezete ilyen:

```
<index>
 <word>
  <name>itt_egy_kulcsszó</name>
  <path freq="11" title="Valami okosság">
    /cdrom/blah.html
  </path>
  <path freq="10" title="További menő dolgok">
    /cdrom/blah2.html
 </path>
 </word>
 <word>
 ...

</index>
```

2. lista A Mystuff.keys fájl kulcsszavai

```
$ tail mystuff.keys
you 134910
for 138811
i 149471
in 168657
is 179815
of 252424
and 273283
a 299319
to 349069
the 646262
```

Ha minden XML-fájl ugyanabban a könyvtárban helyezkedik el, a fájlmegnyitási idő hosszú lenne, ezért a hatékonyság érdekében a fájlrendszerben szükség van az alkönyvtárak használatára.

Rövid kezelési leírás

A végfelhasználóknak nem kell ezzel törődniük. Nekik elég a keresett szót beírniuk a weblapon, és a jsFind visszaadja a kulcsszavakat tartalmazó oldalakra mutató hivatkozásokat. Nem kell telepíteni, nem kell vacakolni a beállításával, minden megy, mint a karikacsapás.
Ha tartalomfejlesztő vagy, a te életed egy kicsit nehezebb. A jsFind eszközkészlet megkísérli megkönnyíteni a munkádat. A kezdéshez elég a Perl és sok gépidő, hogy elkészítsd a tárgymutatót. Valószínűleg az összes böngészőt érdemes beszerezned, hogy összevethesd az eredményeket. Egyetlen példa `Makefile` található a jsFind-terjesztésben, de több lépcsőben kell az egyéni igényeknek megfelelően testreszabni.
Első lépésben meg kell szerezni a kulcsszavak és a hivatko-

3. lista Tárgymutató B-fába rendezett XML-fájllá alakítása

```
$ mkindex.pl mystuff-filtered.xml 25
blocksize: 20
keycount: 101958
depth: 4
blockcount: 5098
maximum keys: 194480
fill ratio: 0.524259563965446
bottom fill: 92698
working: 11%
```

zások adathalmazát, méghozzá XML formátumban.
A SWISH-E program saját magam foltozta változatát használom a tárgymutató kinyeréséhez és létrehozásához, majd az eredményeket a jsFind Perl-programjai számára emészthető XML formátumban exportálom. Feltéve, hogy a SWISH-E által gyártott tárgymutató a ***mystuff.index*** fájlban van, az *1. listában* látható parancs végzi el az XML-be történő exportálást.
Az XML-fájl a kulcsszó alapján van rendezve. A kapott adathalmaz valószínűleg még túl nagy, mert a SWISH-E nem foglalkozik a gyakori szavak kiszűrésével. Az eredményt két Perl-programmal szűrhetjük: az ***occurrences.pl*** és a ***filter.pl*** segítségével. Az ***occurrences.pl*** elkészíti a kulcsszavak listáját, és meghatározza, hogy hányszor fordulnak elő a tárgymutatóban:

```
$ occurrences.pl mystuff.xml | sort -n -k 2 >
➥mystuff.keys
```

A ***mystuff.keys*** fájl minden sorában egy kulcsszó található, amit előfordulásainak a száma követ (*2. lista*).
Ezen a ponton véget is ér a kizárandó kulcsszavak listájának hallatlanul bonyolult elkészítése. A kulcsfájlban szerepeljenek azok a szavak, amelyeket a végső tárgymutatóból ki szeretnél hagyni. A saját fájl szerkesztésénél még jobb ötlet, ha a háromszáz leggyakoribb angol szó listáját megszerzed a ➲ http://www.zingman.com/commonWords.html címről.
Ezután futtasd a szűrőt. A csomagban mellékelt ***filter.pl*** nevű Perl-program elkészíti az eredményhalmazt. Jelenleg arra lett beállítva, hogy minden egybetűs tárgymutatókulcsot kiszűrjön (kivéve a C betűt). A két számjeggyel kezdődő kulcsokat (azaz a 3com és társai rendben vannak), valamint mindent mást, ami a kivételfájlban meg van adva, ugyancsak kiszűr:

```
$ filter.pl mystuff.xml mystuff.keys
➥>mystuff-filtered.xml
```

Ez a lépés meglehetősen hosszú ideig tart. Ellenőrizd, hogy a végső fájl elfér-e a rendelkezésre álló tárhelyen. A végső tárgymutató mérete nagyjából 75 százaléka lesz a szűrt tárgymutatónak. Ha ez túl nagy, a méretét egy nagyobb kulcsszó-kivétellista megadásával leszoríthatod.
A következő nagy lépés magának a tárgymutatónak a létrehozása. Létezik egy olyan parancsfájl, amelyik a tárgymutatót B-fába rendezett XML-fájlokká alakítja (*3. lista*).
A következő megfontolandó feladat a paraméterezés. A `blockcount` értéke azt jelenti, hogy hány csúcsot kell létrehozni a B-fában. Minden csúcs egy kulcsfájlt, egy adatfájlt és egy könyvtárat jelent. Ha a fájlok és könyvtárak száma túl nagy, a `blocksize` növelésével be lehet szabályozni. Ha a `blocksize` túl nagy, a keresés lelassul. A `bottom fill` értékkel egyensúlyban lehet tartani. Ha ennyi kulcsot teszünk az alsó sorba, az alsó sor lezáródik és nem lehet több csúcsot létrehozni. Így egy kiegyensúlyozott fát kapunk.
Ha minden jól megy, a pillanatnyi könyvtárban három fájl jön létre: a ***0.xml***, a ***_0.xml*** és a ***0*** könyvtár – ezek a tárgymutató fájljai. A következő lépésben az eredményt a mellékelt példa segítségével illeszd be a saját HTML/JavaScript-oldaladba. Az eredményeket a mellékelt programrész kapja meg, és ezt kell visszaküldeni a pillanatnyi weboldalra. A példa JavaScriptet használ dinamikus HTML előállítására.

Összegzés

A jsFind sokféleképpen fejleszthető tovább, és bizonyára ezek a fejlesztések meg is fognak valósulni, amikor a nyílt forrás hívei elkezdik használni a programot. Ilyen lehetőség lenne például a képarchívumok bélyegképekkel való keresése, a többlapos eredménykijelzés vagy a böngészőmegfelelőség-ellenőrzés. Ezek a mostani kódból kiindulva mind megvalósíthatók. A Linux Journal 2002-es Archive CD-jén már a jsFind keresőmotor működik. Ha CD-ROM-okat állítasz össze, a zárt kódú operációs rendszerekre elérhető megoldások helyett használd inkább a jsFindot. Olcsóbb lesz és növeli a lehetséges felhasználók számát. A végfelhasználók jobban örülnek, ha semmit nem kell telepíteniük a gépükre, és nem kell egy másik operációs rendszert elindítaniuk, ha keresni akarnak egy CD tartalmában. Ezt a programot valószínűleg másra is fel lehet használni, ezért hát tekintsük nyílt forrású szerszámosládánk újabb elemének.

Linux Journal 2003. december, 116. szám

Shawn P. Garbett
Tanácsadó több mint 15 év tapasztalattal a háta mögött a Unix-rendszerek és azok orvosi alkalmazásainak a területén. Kedveli a dzsesszt és a fordulatos történeteket.

KAPCSOLÓDÓ CÍMEK

GNU ➲ http://www.gnu.org

Josh Rabinowitz „Hogyan indexeljünk?”
Linuxvilág 2003. szeptember
➲ http://www.linuxvilag.hu

A jsFind honlapja
➲ http://www.elucidsoft.net/projects/jsfind