Fájlrendszerek indexelése a libferris segítségével

A teljes értékű szöveges és metaadat alapú keresés többé nem csupán álom. A libferris könyvtárral, mely a fájlok keresését tartalom és számos egyéb jellemző szerint teszi lehetővé, már ma is bárki számára elérhető.

A *libferris Project* 2001 elején indult, célja egy megosztott könyvtárként működő virtuális fájlrendszer létrehozása volt. A *libferris* egyetlen fájlrendszer felületen keresztül több fa jellegű szerkezethez biztosít hozzáférést. Mivel a *libferris* a rendszermag tér helyett a felhasználói térben fut, nagyszámú fa jellegű szerkezet elérhetővé tételére képes. Ezeket a forrásokat a *Linux* rendszermagból nehéz lenne elérni. A *libferris* használatakor minden fájlrendszer a `root://` *URI*-n keresztül érhető el, amely felöleli a rendszermag `file://` *URL*-jeit; a relációs adatbázisokat; az *XML* fájlokat és adatbázisokat; a hálózati, például *HTTP*/*FTP* alapú kiszolgálókat; az egyéb összetett fájlokat – *db4*, .tar és *RDF* –; továbbá a szabványos fájlrendszereket, mint az *ext3* és az *XFS*.

A fájl és a könyvtár fogalma *libferris* alatt egyetlen elvont fogalommá egyesül. Ezzel lehetővé válik, hogy a *libferris* fájlrendszerként fűzzön be például .tar állományokat. A .tar állomány például fájl és könyvtár is egyszerre.

A ***kiterjesztett jellemző (extended attribute, EA)*** felület különböző forrásokból származó adatokat tesz elérhetővé, ide értve a rendszermag *listxattr(2)* felületét, az *RDF*/*bdb* gyűjteményeket és a dinamikusan kibontott értékeket is. Dinamikus *EA* például egy kép szélessége. A képszélesség *EA* olvasásakor a *libferris* egy beépülő modul közreműködésével határozza meg az adott képfájlban lévő kép szélességét. Egy másik példa a hangfájlok mintavételezési gyakorisága. További *EA*-kat és ezek leírását a *libferris* webhelyén lehet találni. (Lásd az internetes forrásokat.)

A *libferris* kétféle típusú indexet képes létrehozni és lekérdezni, teljes szövegeset és *EA* típusút. A teljes szöveges indexek segítségével a fájlok között tartalmuk alapján tudunk keresni, míg az *EA* indexek a metaadatok alapján végzett kutakodást teszik lehetővé. A teljes szöveges vagy *EA* típusú lekérdezések feldolgozásához szükséges indexelő szerkezetek jelentős mértékben eltérők. A teljes szöveges indexek például az egyes szavakat tartalmazható dokumentumok listáját (fordított fájl) tárolva tudják feldolgozni az olyan kéréseket, mint a „keress meg minden a *libferris* szót tartalmazó dokumentumot”. Az *EA* indexeknek ezzel szemben a tartományokra vonatkozó lekérdezéseket is képeseknek kell lenniük kezelni, mint például „keresd meg a múlt hónapban módosított fájlokat”.

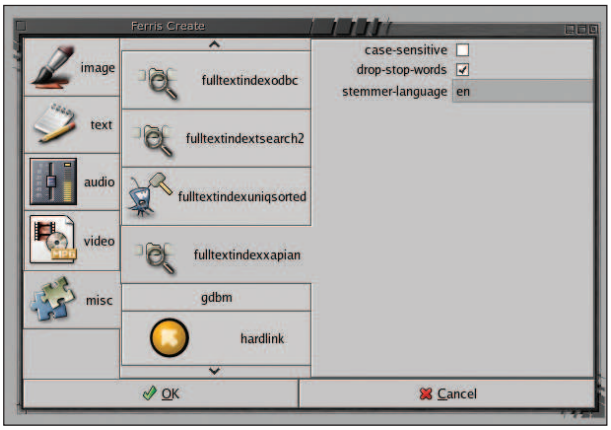
Miért épp a libferris?

Néhány érv amellett, miért érdemes a *libferrist* választani a fájlrendszerek indexelésére és a lekérdezések elvégzésére:

- Beépülő modulok segítségével, teljes szövegindexelés céljából képes a fájlokba ágyazott szövegek kibontására.
- Egyesített felületet ad minden az indexelés bemeneteként szolgáló forráshoz. Például a következő források egységesen indexelhetők a *libferrisszel*: *SleepyCat dbxml* fájlokban található szövegek, .tar állományok tartalma, *mbox* fájlokban vagy relációs adatbázisok által tárolt anyagok.
- A fájlokban található metaadatokat szintén lehet indexelni és keresni. Például a zenei fájlokban található *ID3* címkék alapján előadó szerinti keresésre nyílik lehetőség.
- Az alapszintű hozzáadó/lekérdező parancsok minden beépülő modulnál azonosak, az indexelő megoldások között tehát könnyedén lehet váltani.
- Kombinációs keresések szövegekre és a fájlrendszer kiterjesztett tulajdonságaira. A *ferris-search* keresőeszköz segítségével több keresést is egyetlen eredménykészletbe lehet egyesíteni.
- Fájlok keresése olyan metaadatok alapján, amelyekkel korábban rendelkeztek.
- Fájlkeresés *felügyelt gépi tanulás (Supervised Machine Learning, SML)* alapú szűréssel – gyakorlatilag szemétszűrés a fájlrendszeren belül. Az *SML* ismertetése sajnos túlmutatna írásom keretein.

A *libferris* beépülő modulokkal kezeli mindezeknek az indexeknek a megvalósításait. A teljes szöveges indexek esetében a következő megvalósítások bármelyikét, sőt, akár mindegyikét is használhatjuk: belső, fordított fájlokra épülő formátum; *gcj*-vel lefordított *Apache Lucene*, relációs adatbázis hátterű *ODBC*; *Xapian* vagy *TSearch2* modul *PostgreSQL*-lel. Az *EA* indexek esetében ugyancsak belső, fordított fájlokra alapuló megoldást; *LDAP*-ot; *gcj*-vel fordí-

tott *Apache Lucene*-t; relációs adatbázis hátterű *ODBC*-t vagy natív, némi *PGSQL*-lel kiegészített *PostgreSQL*-t választhatunk. Általános használatra a teljes szöveges indexekhez a *Xapian* vagy a *TSearch2* az ajánlott, *EA* indexekhez pedig a *PostgreSQL* vagy az *ODBC*.
A *PostgreSQL* modulok hasonlóak az *ODBC* alapúakhoz, ám ezek *PGSQL*-t és egyéb, kifejezetten a *PostgreSQL*-re jellemző szolgáltatásokat használnak. Ha a teljes szöveges indexeléshez a *PostgreSQL TSearch2* beépülő modulját használjuk, akkor *PostgreSQL* kiszolgálónkon egy sablon adatbázist kell létrehoznunk. A részletekről bővebben az internetes források közt lehet olvasni.
A *libferris* minden indexet a saját könyvtárába helyez. Az alapértelmezett teljes szöveges és az *EA* index rendre a *~/.ferris* könyvtár *full-text-index* és *ea-index* könyvtárába kerül. Az indexek létrehozása a *ferriscreate* csomag `fcreate` vagy `gfcreate` parancsával történik. A *libferris* más eszkö-



1. ábra Teljes szöveges Xapian index létrehozása a gfcreate segítségével

zeihez hasonlóan itt is igaz, hogy a *gf* előtaggal ellátott program ugyanazt teszi, mint az *f* előtagú, ám előbbi *GTK+2* felülettel is rendelkezik. A továbbiakban mindkét típusú index létrehozásáról, feltöltéséről és lekérdezéséről lesz szó.

Teljes szöveges indexelés

Vágjunk a közepébe: először létrehozunk egy teljes szöveges indexet a */tmp* könyvtárban, hozzáadunk néhány fájlt, majd az index használatával elvégzünk egy lekérdezést. Hozzuk létre az új index könyvtárát, majd a `gfcreate` segítségével magát az indexet is:

```
$ mkdir /tmp/text-index
$ gfcreate /tmp/text-index
```

A `gfcreate` grafikus felületének bal oldali lapján jelennek meg a fő *MIME* típusok, a *misc (speciális)* lapon pedig a létrehozható, de a *MIME* típusoktól különböző dolgokat találjuk. A *misc* kiválasztása után egy második lapszinten az összes elérhető indexformátum megjelenik. Az *1. ábrán* követhető, hogy *Xapian* teljes szöveges indexet választottam, a szótőképzéshez angol nyelvet jelöltem meg, illetve nem akarom megkülönböztetni a kis- és nagybetűket.

Amikor hozzáadjuk a fájlokat a teljes szöveges indexhez, a *libferris* az *as-text EA* használatával próbálja előállítani a fájl szöveges formáját. Az *as-text EA* támogatására számos beépülő modul készült; a *PDF* és a *HTML* fájlok, a *man* oldalak és a *djvu* képek egyaránt támogatják az *as-textet*. A `findexadd` és a `findexquery` eszközöknek a `-P` parancssori kapcsolóval adhatjuk meg, hogy melyik indexet használják. Az alábbi példában a *Samba 3.0.3* csomag egy *man* oldalát és egy *PDF* fájlját használjuk bemenetként. Mivel a tényleges elérési útvonalak az egyes *Linux* terjesztések esetében eltérők lehetnek, a fájlok előtti előtagokat a `/.../` karakterlánccal helyettesítettem:

```
$ findexadd -P /tmp/text-index \
/.../samba-3.0.3/docs/Samba-HOWTO-Collection.pdf
$ findexquery -P /tmp/text-index samba
ID 1 99% [file://.../Samba-HOWTO-Collection.pdf]
Found 1 matches at the following locations:
file://.../Samba-HOWTO-Collection.pdf
$ findexadd -P /tmp/text-index /.../samba.7.gz
$ findexquery -P /tmp/text-index smbstatus
ID 1 100% [file://.../Samba-HOWTO-Collection.pdf]
ID 5 93% [file://.../samba.7.gz]
Found 2 matches at the following locations:
file://.../Samba-HOWTO-Collection.pdf
file://.../samba.7.gz
```

A `findexquery` legfontosabb kapcsolói a `-P`, amely az indexfájl elérési útját adja meg; a `--ranked`, amely rangsorolt teljes szöveges lekérdezések indítására utasít; valamint a `--xapian`; amellyel nyers *Xapian* formátumú lekérdezéseket adhatunk át a háttérszolgáltatásoknak. (Lásd a forrásokat.) Az alapértelmezett lekérdezésformátum a `Boolean`. Ennél a keresés az indexben az összes alfanumerikus szóra történik, és négy belső jelölésű (infix) logikai műveleti jelet lehet használni: & (és), | (vagy), ! (tagadás) és - (mínusz). A rangsorolt mód minden kifejezést egyesít, majd visszaadja a lekérdezés alapján legérdekesebbeknek minősülő dokumentumokat. A *Xapian* formátumnál a *libferris* közvetlenül adja át további feldolgozásra a háttérszolgáltatásnak a lekérdezést. Jelenleg kizárólag a *Xapian* háttérrendszer képes az ilyen lekérdezések kezelésére.

EA indexelés

Az *EA* indexek hozzáadásának és lekérdezésének folyamata rendkívül hasonló a teljes szövegesekéhez. Az *EA* indexek a `feaindexadd` és a `feaindexquery` paranccsal kezelhetők, mindkettő fogadja a már megismert `-P /index_elérési_útja` kapcsolót.
Az *EA* indexek hangolására három átadott érték használható, ezeket az indexek létrehozásakor kell megadni. Azt szabják meg, hogy a fájlok indexelésekor mely *EA*-k jussanak szerephez. Létrehozhatunk például egy kisebb *EA* indexet, amely csak fájlneveket, -méreteket és néhány képjellemzőt tartalmaz, segítségével jól tudunk képekre keresni. Dönthetünk úgy is, hogy bizonyos *EA*-kat figyelmen kívül hagyunk, például azért, mert kiszámításuk időigényes lenne, vagy egyszerűen érdektelenek számunkra. Ha az indexet nem akarjuk a fájlok sértetlenségének ellenőrzésére használni, akkor az *MD5-* és az *SHA-1-*

kivonatok és az egyéb ellenőrző összegek elhagyásával számottevő mennyiségű időt takaríthatunk meg, ugyanis az ellenőrző összegek számításához végig be kell olvasni az összes fájlt.

Az *EA* indexekre vonatkozó általános átadott értékek közül az első a `max-values-size-to-index`, amely bájtokban mérve azt adja meg, hogy legfeljebb mekkora lehet a valamely jellemző vizsgálatakor az indexhez hozzáadott érték. A legtöbb *EA* értéke röviden, kevesebb mint 100 bájton ábrázolható. Az alapbeállítás meglehetősen bőkezű, és bármely *EA* értéknél 1024 bájtos maximális méretet engedélyez. A másik két átadott érték az `attributes-not-to-index` és az `attributes-not-to-index-regex`. Ezek a fájlok indexhez való hozzáadásakor figyelmen kívül hagyandó *EA*-k nevét adják meg. Természetesen magunknak kell eldöntenünk, hogy minden *EA*-t indexelünk, ekkor a lekérdezésekhez minden adat rendelkezésünkre fog állni, viszont a fájlok hozzáadása lassabb lesz, vagy csak az *EA*-k egy kisebb részét vesszük figyelembe, amivel felgyorsul a hozzáadás, ám a lekérdezések egy része eredménytelen lesz.

A `not-to-index` átadott értékek alapbeállításainak lehetővé kell tenniük a fájlok gyors hozzáadását, ugyanakkor biztosítaniuk kell a figyelemre érdemes *EA*-k indexelését. A három átadott érték alapbeállítását a `cc/capplets/index/ferris-capplet-index` segédprogram futtatásával lehet felülbírálni. A program megadja az új indexek létrehozásakor figyelembe vett alapbeállításokat, és a további fájlindexelésekhez módosítja a *~/.ferris* alatt található indexeket.

Az *EA* index tárolására *PostgreSQL* adatbázist használunk. Az *EA* indexek zökkenőmentes létrehozásához az `fcreate` eszközök futtatása előtt némi telepítési-beállítási jellegű munkát is kell végeznünk. A `PGSQL` nyelvet alapértelmezés szerint engedélyezni kell az új adatbázisokhoz. Rootként futtatva az alábbi parancs pontosan ilyen hatással jár:

```
# createlang -d sablon1 plpgsql
```

Ha nem akarjuk megváltoztatni a `sablon1` adatbázist, akkor a *PostgreSQL* adatbázist kézzel is létrehozhatjuk. Engedélyezzük hozzá a `plpgsql`-t, majd az alábbi `fcreate` parancshoz fűzzük hozzá a `db-exists=1` karakterláncot.

A *PostgreSQL EA* indexek esetében a *PostgreSQL* adatbázis által használt felhasználónevet, jelszót, állomásnevet, kaput és adatbázisnevet is megadhatjuk. Alapesetben (`db-exists=0`) a megadott nevű adatbázisnak még nem szabad léteznie, hiszen éppen most hozzuk létre az új *EA* indexhez.

A relációs adatbázisra alapuló *EA* indexekhez tartozik még egy érdekes beállítás, ennek segítségével megváltoztathatjuk, hogy bizonyos *EA*-k hogyan kerülnek normalizálásra az adatbázisban. Az alapértékek nagy valószínűséggel ebben az esetben is megfelelők. Röviden összefoglalom a kérdés lényegét.

Az `extra-columns-to-inline-in-docmap` megadja azon *EA*-k listáját, amelyek olyan fontosak a keresések szempontjából, hogy denormalizálni kellene őket a `docmap` táblába. A szinte minden fájl esetében egyedi értéket mutató EA-k tárolását hatékonyabban lehet megoldani a `docmap` táblában, belsőleg. Ha egy *EA*-t ilyen módon szeretnénk denormalizálni, akkor az *EA SQL*-típusát is meg kell adnunk.

1. táblázat *Példák EA-kra*

name-extension	A fájl kiterjesztése, például tar
treeicon	Egy a fájlhoz illeszkedő kép URL-je
is-audio-object	A fájl MIME főtípusa audio
is-source-object	A fájl forráskódot tárol
is-remote	A fájl a gép számára távoli.
language-human	A fájl emberi nyelve
a52-channels	Hangcsatornák száma
év	A zeneszámot tartalmazó album, lemez kiadásának éve

EA normalizálása relációs adatbázisban

A normalizált *EA*-kra vonatkozó lekérdezések teljesítésében négy tábla jut szerephez. A `docmap` tábla a fájl *URL*-jét és mesterségesen előállított belső kulcsát, a `docid`-t tárolja. Az `attrmap` tábla egy *EA* nevét tartalmazza, és szintén mesterséges belső kulcsot rendel hozzá, ez az `attrid`. Emellett az érték típusától függően a számos – egyébként rendkívül hasonló – `valuemap` tábla valamelyikét használjuk.

Az `strlookup` például a `varchar` értékekhez egy mesterséges belső kulcsot rendel, ez a `vid`. Végül, egy egyesítő (join) tábla, a `docattrs` egyesít egy `docid`-t egy `attrid`-vel és egy `vid`-vel, ezzel rögzítve, hogy adott fájl adott értékű jellemzővel rendelkezik. Ha tehát a (szélesség<=800) lekérdezést akarjuk feldolgozni, és a szélesség *EA* normalizálva van, akkor kell végeznünk egy keresést az `attrid`-re és a `vid`-re, egyesítenünk kell őket a `docattrs` táblába, ezzel előáll a lekérdezés feltételeinek megfelelő szélességgel rendelkező `docid`-k listája.

A normalizált *EA*-k tárolása egy oszlop formájában közvetlenül a `docmap` táblában történik. A fenti szélességre keresés végrehajtása során az egyező `docid`-k közvetlen keresése a `docmap.szélesség` oszlopról készített relációs adatbázis index alapján történik. A normalizálással időt takaríthatunk meg, miközben tárhelyet használunk fel. Általában elmondható, hogy a keresésekben gyakran szereplő *EA*-kat érdemes belsőleg tárolni. A *stat(2)* hívásokban nem szereplő vagy érdekesnek ítélt *EA*-k indexelését az `attrmap`, a `valuemap` és a `docattrs` táblára érdemes hagyni.

Én például saját felhasználónevemet használom, az adatbázis neve (`dbname`) pedig `lj`. Az alábbiakban a második paranccsal, a nem interaktív fcreate segédeszközzel létrehozom az *EA* indexet. A harmadik paranccsal a megosztott képkönyvtáramból az összes *JPEG* fájlt hozzáadom az indexhez. A `feaindexadd` parancsot is használhatjuk a `-d` kapcsolóval, ekkor pontosan felsorolhatjuk a parancssorban a kívánt elérési útvonalakat. A `-d` kapcsoló nélkül futtatva a `feaindexadd` rekurzívan megvizsgálja a megadott elérési utat:

```
$ mkdir /tmp/ea-index
$ fcreate --create-type=eaindexpostgresql \
--target-path=/tmp/ea-index dbname=lj user=ben
```

```
# ha már van új adatbázisunk, fűzzük hozzá
# a db-exists=1 kapcsolót
$ find /usr/share/backgrounds/images \
-name "*.jpg" \
| feaindexadd -P /tmp/ea-index --filelist-stdin
```

A képkönyvtáramban 42 JPEG fájl található. Kérdezzük le az indexet:

```
$ feaindexquery -P /tmp/ea-index '(width>=640)'
Found 34 matches at the following locations:
file:///usr/share/backgrounds/images/
➥dewdop_leaf.jpg
...
$ feaindexquery -P /tmp/ea-index '(size>=100k)'
Found 42 matches at the following locations:
file:///usr/share/backgrounds/images/
➥dewdop_leaf.jpg
...
$ feaindexquery -P /tmp/ea-index \
'(&(width<=800)(size>=100k))'
Found 19 matches at the following locations:
file:///usr/.../images/space/
➥apollo08_earthrise.jpg
...
```

Az *EA* index lekérdezésének írásmódja az *LDAP* keresési szűrők karakterlánc formátumú megadására épül (*„The String Representation of LDAP Search Filters", RFC 2254*). Meglehetősen egyszerű írásmódról van szó, a bal és a jobb oldalon szereplő érték összevetésére alkalmas kifejezéseket állíthatunk össze, ezeket kombinálhatjuk, illetve logikai ÉS (&), VAGY (|) és NEM (!) műveletekkel egészíthetjük ki. Minden kifejezés zárójelek közé kerül, a műveleti jelek argumentumaik előtt szerepelnek. A műveleti jelek fölöttébb egyszerűek: == az egyenlőség, <= és >= az értéktartomány és =~ a reguláris kifejezés egyeztetése.

Keresés a múltból

Az *ODBC* (kiegészítő szolgáltatásként) és a *PostgreSQL* (minden esetben) *EA* indexelő beépülő modulok lehetővé teszik, hogy egy fájl *EA*-inak több változatát is tároljuk. Ha egy fájl metaadataiból több változattal is rendelkezünk, akkor annak valamikor korábban érvényes jellemzői alapján is végezhetünk kereséseket.
A szolgáltatás használatához egy különleges *EA* segítségével meg kell adnunk a bennünket érdeklő időszakot. Az időkorlátozó *EA*-k az `atime`, a `ferris-current-time`, a `multiversion-mtime` és a `multiversion-atime`. A két utóbbi *EA* összehasonlítása a keresett fájl `mtime` és `atime` értékével történik. Az adott fájl indexadatainak egy változatához tartozó `ferris-current-time` *EA* a fájl indexelésének időpontját adja meg. Ha időtartományt nem választunk ki, akkor a lekérdezés csak az egyes fájlok metaadatainak legújabb változatát veszi figyelembe.
Az időkorlátozásokat karakterlánc formájában adhatjuk meg, a *libferris* megpróbálja a lehető legpontosabban meghatározni a kapott karakterlánc formátumát. A *libferris* csomag *tests/timeparsing* könyvtárában található egy időfeldolgozó segédeszköz, ennek időértékeket tudunk átadni, és így ki tudjuk deríteni, hogy a *libferris* mit hoz ki az egyes karakterláncokból. A használható időkarakterláncokról a *libferris GYK* vonatkozó részében találunk bővebb ismertetőt. (Lásd a forrásokat.)
Az alábbi példa az időkorlátozott lekérdezést szemlélteti; ebben az esetben az egy évvel ezelőtt indexelt, meghatározott szélességtartományba eső képeket keressük:

```
$ feaindexquery -P /tmp/ea-index \
 '(&(width>=1600)(ferris-current-time<=1
 ➥year ago))'
```

Ha egy nagyobb fájlt két éve indexeltünk, majd később lecseréltük a kicsinyített változatára és újraindexeltük, akkor szerepelni fog a fenti lekérdezés eredményében. Ennek oka az, hogy metaadatainak egyik változata egyezést mutat a keresési feltételekkel.
Mivel az *EA* lekérdezésekre vonatkozó időkorlátozások kezelése ugyanazon a felületen keresztül történik, mint az *EA* értékekre vonatkozóké, a kívánt időtartományt a megszokott lekérdezési módszerekkel tudjuk megadni. Kiválaszthatjuk például azokat a dokumentumokat, amelyek indexelése 2003-ban és megadott szélességgel történt, vagy a megadott személy tulajdonában lévő, az elmúlt hónapban módosítottakat:

```
## megjegyzés: egy sorban az egész
$ feaindexquery -P /tmp/ea-index '
(|
  (&
    (width>=1600)(ferris-current-time>=begin
    ➥2003)
    (ferris-current-time<=end 2003)
  )
  (&
    (owner-name==sarusama)
    (multiversion-mtime>=end last month)
  )
)
```

Összefoglalás

Őszintén remélem, hogy sikerült érthetően felvázolnom, mire is képes a *libferris* jelenlegi megvalósítása az *EA* alapú és a teljes szöveges indexelés terén, és minél több olvasó figyelmét sikerült felkeltenem. A jelenlegi változat elkészítése szükségszerű lépés volt egy sokkal nagyobb mértékben formalizált szemantikus fájlrendszerlekérdező és böngésző felület megvalósításának végső célja felé.

Linux Journal 2005. február, 130. szám

A cikkhez tartozó források elérhetősége:
➲ *www.linuxjournal.com/article/7928*

Ben Martin több mint tíz évig fájlkezelőkön dolgozott. Jelenleg doktoranduszként a jelentéstani fájlrendszerek és a formális fogalomanalízis egyesítésén dolgozik, amitől az ember-fájlrendszer kapcsolattartás javulását várja.