Ügyféloldali parancsfájlok készítése

Marco megmutatja, hogy miként lehetséges böngészés közben az oldalaknak csak azt a részét letölteni, amely valóban érdekel bennünket.

Linuxhoz rengetegféle böngészésre és FTP-re alkalmas eszköz létezik, amelyek szolgáltatásban gazdagok és a felhasználóik minden igényét képesek kielégíteni: kezdve a parancssor megszállottaitól a 3D többmonitoros képernyőfüggőkig. De az ilyen eszközöknek létezik egy nagy hibája: elvárják a felhasználótól, hogy ott üljenek a billentyűzet előtt. Természetesen akadnak olyan eszközök, amelyek egész webhelyet képesek tükrözni, amíg te alszol, ilyen például a `wget` is, azonban ezeknek az eszközöknek először meg kell találnod a megfelelő URL-t, ha pedig az anyagot letöltötték, azt bitről bitre végig kell olvasnod. Kicsi, statikus oldalaknál ez nem gond, de mi történik olyankor, ha egy oldalt minden nap le kell töltened egy véletlenszerű címről? Vagy ha le akarsz tölteni egy 100 K-s dokumentumot, és csak néhány címszó érdekel belőle? Ismerkedj meg az ügyféloldali parancsfájlok készítésével, és minden olyan módszerrel, amellyel lehetővé válik, hogy csak olyan oldalakat vagy oldalrészeket tölts le, amelyek valóban érdekelnek, és a számítógép már előkereste őket. Ilyen parancsfájlokkal csak azokat a közlekedési és időjárási adatokat kell elolvasnod, amelyek a te környezeteddel kapcsolatosak, nem kell feleslegesen érdektelen képeket nézegetned, és még a továbbhaladáshoz szükséges hivatkozásokat is kézhez kapod.

Lényeges adatok a szerzői jogról és a sávszélességről

Amellett, hogy az ügyféloldali parancsfájlok használatával időt takaríthatsz meg, még sok egyebet is meg tanulhatsz, többek között önfegyelemre nevel. Ha az itt leírt módszereket válogatás nélkül alkalmazzuk, az esetleg szerzői jogsértésnek minősülhet, vagy a teljes sávszélességedet felemészthetik, olyannyira, hogy végül akár az internethozzáférésedet is elveszted. Másrészről ez a szabadság csak addig áll fenn, míg a weboldalak magántulajdonnak nem minősülő nyelveken készülnek (HTML, illetve XML), és szabadon hozzáférhető ASCII-ban írják őket. Rengeteg jó weboldal marad életben mindenféle további költség nélkül, ha megfelelő mennyiségű hirdetést töltenek le tőlük, tehát a leírtakat csak meggondoltan szabad alkalmazni.

A lehetőségek

Mielőtt nekikezdenénk megalkotni programunkat, szokás szerint alaposan nézzünk körül, hátha valaki már elkészítette azt, és esetleg az ő munkáját is felhasználhatjuk. Ha a Freshmeat.net keresőjébe beírjuk a „news ticker” szavakat, azonnal 18 találatot kapunk, olyanokat, mint a Kticker, a K.R.S.S – egészen a GkrellM Newstickerig. Ezek mindegyike nagyon jól használható eszköz, de csak híreket töltenek le, és nem működnek anélkül, hogy bizonyos dolgokat meg ne változtatnál bennük. Ezen túlmenően a felsorolt projektek mindegyike valamilyen grafikus eszköz, tehát `cron`-feladatként nem képesek futni, és az sem biztos, hogy a kimenetüket egy másik programnak továbbítani tudod.
Ezen a területen, ha a saját elképzeléseidet szeretnéd megvalósítani, csaknem mindig magadnak kell megalkotnod azt.
Ez okból most mi sem kísérletezünk semmilyen hiánytalan megoldás közreadásával, helyette az általános megközelítést tanulmányozzuk.

Az alapadatok összegyűjtése

```
#! /usr/bin/perl
#20011210

use strict;

use LWP::UserAgent;
use LWP::Simple;
use HTML::Parse;
use HTML::Element;
use URI::URL;
use Image::Grab;

my $HTML_FILE = get($ARGV[0]);
my @HEADER    = head($ARGV[0]);

my @ALL_URLS;
my $PARSED_FILE =
➥HTML::Parse::parse_html($HTML_FILE);
for (@{$PARSED_FILE->extract_links()}) {
        my $LINK = $_->[0];
        my $URL  = new URI::URL $LINK;
        my $FULL_URL = $URL->abs($ARGV[0]);
        push @ALL_URLS, $FULL_URL;
}
```

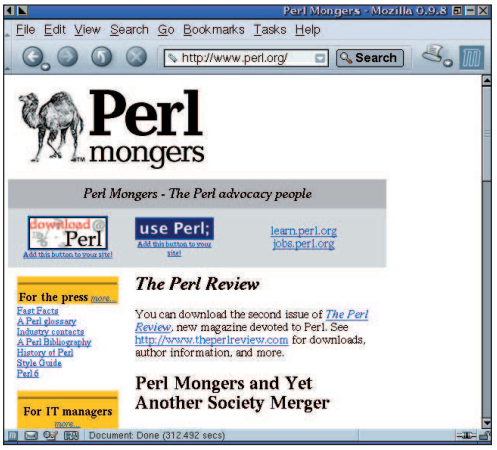
Mire van szükség?

Hogy az e cikkben leírtakból előnyt kovácsolhass, mindössze néhány eszközre lesz szükséged, valamint valamelyest ismerned kell a Perlt, össze kell tudnod állítani egy-két szabályos kifejezést, illetve a következő Perl-modulok használata is elkél: `LWP::UserAgent`, `LWP::Simple`, `HTML::Parse`, `HTML::Element`, `URI::URL` és `Image::Grab`.
A CPAN-ról mindegyiküket ➲ http://www.cpan.org letölthető. Ahhoz, hogy ezeket a modulokat feltelepíthesd, még akkor is, ha nem rendelkezel rendszergazdai jogosultsággal (ez az irodai gépek esetében általános), nem kell mást tenned, mint bemásolni őket egy általad kiválasztott könyvtárba – ahogyan az a Perl leírásában és a kapcsolódó README fájlokban le van írva.
A cikkben leírtakat egy Red Hat Linux 7.2-n próbáltam ki, és amennyiben a kódban szereplő abszolút útvonalakat megváltoztatod, bármilyen Unixon működniük kell, amelyeken a Perl és a szükséges alkalmazások elérhetők.

A szükséges adatok összegyűjtése

A lejjebb leírt feladatok mindegyikéhez és általánosságban véve az ügyféloldali parancsfájlok készítéséhez is szükséges,

hogy kezdésképpen képes legyél néhány weboldalt letölteni és tárolni, amelyeket később elemezhetsz: mikor módosították utoljára, milyen URL-ekre található hivatkozás bennük vagy ezek kombinációja.
Ezek az adatok a webügyfelek elején elhelyezkedő néhány soros programocskákkal mind összegyűjthetők, mint ahogyan az *listában* láthatjuk.
A Perl-parancsfájlok a kötelező `use strict` meghatározással kezdődnek, majd betöltik a szükséges modulokat. Amint ez megtörtént, a webhely teljes tartalmát a `get()` tagfüggvényen keresztül a `$HTML_FILE` változóba mentjük.



A következő műveletekkel a HTTP-fejléc minden sorát egyenként a `@HEADER` tömbbe mentjük. Végül pedig egy tömböt `(@ALL_URLS)` hozunk létre, és egy `for` ciklussal az összes hivatkozást mentjük az oldalról – egyúttal a relatív hivatkozásokat az `abs()` tagfüggvénnyel abszolúttá alakítjuk. Így a ciklus végén az `@ALL_URLS` tömb az összes hivatkozást, amelyet az oldalon talált, tartalmazni fogja.
Egyebek mellett e tagfüggvények teljes leírását megtalálod a ***Web Client Programming*** című könyvben (lásd a Hivatkozások részt).
Miután ezt az anyagot összeszedtük, elkezdhetjük használni. Ha a weboldal tartalmát menteni szeretnéd, az eredeti kódhoz egy `print` utasítást kell hozzáadnod:

```
print $HTML_FILE;
```

Majd futtasd le a héjprogramból:

```
./webscript.pl http://www.fsf.org > fsf.html
```

Ez a parancssor letölti neked a ➲ http://www.fsf.org kezdőoldalát és menti az ***fsf.html*** fájlba. Ne felejtsd el, ha csak ennyire van szükséged, erre a célra a `wget` sokkal jobb eszköz (Lásd a Hivatkozások részt: ***Letöltés böngésző nélkül***).

Képek mentése weboldalról

Ha az abszolút hivatkozások már az `@ALL_URLS` tömbben vannak, a következő `for` ciklussal az összes képet is letölthetjük:

```
foreach my $GRAPHIC_URL (grep
➥/(gif|jpg|png)$/, @ALL_URLS) {
  $GRAPHIC_URL =~ m/([^\/]+)$/;
  my $BASENAME = $1;
  print STDERR "$GRAPHIC_URL elmentØse ide:
➥$BASENAME....\n";
  my $IMG = get ($GRAPHIC_URL);
  open (IMG_FILE, "> $BASENAME") || die
➥"Nem tudtam megnyitni: $BASENAME\n";
  print IMG_FILE $IMG;
  close IMG;
}
```

A ciklus a dokumentumból minden olyan hivatkozást ment, melynek ***.gif***, ***.jpg*** vagy ***.png*** a kiterjesztése (ezeket az eredeti tömb `grep`-elésével kapja meg). A szabályos kifejezés először megtalálja a valódi fájlnevet, melyet az utolsó törtvonal és a hivatkozás vége közül vág ki – ezt a kifejezést lehetne általánosítani úgy, hogy olyan rendszereken is működjön, ahol a könyvtárelválasztó egy fordított perjel.
Az illesztés eredménye a `$BASENAME` változóba kerül, magát a képet pedig a már ismert `get()` tagfüggvénnyel mentjük az `$IMG` változóba. Ezután a fájlt helyileg megnyitjuk, és a változó tartalmát egészében beleírjuk.
Természetesen a legtöbb esetben nincs minden képre szükséged, hiszen a többségük hirdetés, vagy éppen a pillanatnyi oldal logója. Ilyen esetekben ha vetsz egy pillantást az oldal forrására, szinte azonnal meg tudod mondani, hogy melyik képet választod. Tegyük fel, hogy egy mindig változó nevű képre van szükséged, amely azonban három hivatkozásként szerepel. Ebben az esetben a kódot a következőképpen módosítsuk:

```
my $IMG_COUNT  = 0;
my $WANTED_IMG = 3;
foreach my $GRAPHIC_URL (grep
➥/(gif|jpg|png)$/, @ALL_URLS) {
        $IMG_COUNT++;
        next unless ($IMG_COUNT ==
        ➥$WANTED_IMG);
        # a ciklus tovÆbbi rØsze vÆltozatlan
        last if ($IMG_COUNT == $WANTED_IMG);
}
print "A mai napon nincs ilyen fÆjl\n" if
➥($IMG_COUNT != $WANTED_IMG);*
```

Az első utasítás a ciklusban mindig növeli a képszámláló változónkat, míg következő ***next*** utasítás újraindítja a ciklust, amennyiben még nem értünk a megfelelő képhez. A ***last*** utasítással megelőzhetők a felesleges ciklusok, ha a keresett képet már megtaláltuk. A legutolsó sor azt ellenőrzi, hogy találtunk-e megfelelő számú képet – amennyiben nem, hibát jelez.
Ha a kép neve nem teljesen véletlenszerű, még egyszerűbb a dolgunk, mivel a nevére külön is kereshetünk:

```
foreach my $GRAPHIC_URL
➥(grep /(^daily(\d+).jpg)$/, @ALL_URLS) {
```

Ez a ciklus csak olyan képeket keres, melyek a „daily” szóval kezdődnek, tetszőleges számú szám követi őket, és ***.jpg*** a kiterjesztésük.
A két módszer kombinálható, és természetesen egyéb módszerek is elképzelhetők. Ha tudod, hogy a kép neve megegyezik az oldal címével, majd ezt követi a dátum ÉÉÉÉHHNN formátumban, akkor először keressük ki az oldal címét:

```
$HTML_FILE =~ m/<TITLE>([^<]+)<\/TITLE>/;
my $TITLE = $1;
```

Majd számoljuk ki a dátumot:

```
my ($sec, $min, $hour, $day, $month, $year,
➥@dummy) = localtime(time);
$month++;       # a h napok nullÆnÆl kezdıdnek
$year += 1900; # felkØsz lt nk aY2K-ra ;-)))
$TODAY = $year.$month.$day;
```

Végül pedig ez alapján szűrjünk:

```
foreach my $GRAPHIC_URL (grep
➥/(^$TITLE$TODAY.jpg)$/, ALL_URLS) {
```

Jelenítsük meg a szöveg bizonyos részét!

Most kezd csak igazán érdekes lenni a dolog. A legtöbb erőfeszítést és időt igénylő művelet az, ha azt szeretnénk elérni, hogy az oldalnak csak bizonyos része jelenjen meg, mivel ebben az esetben az összes oldal teljes oldalfelépítését elemeznünk kell, és ha valamelyikük szerkezete megváltozik, annak az oldalnak az elemzését elölről kezdhetjük.

Ha lassú az internetkapcsolatod, vagy éppen gyors, de nem akarod lelassítani az MP3-ak és játékok letöltését, bizonyosan megtérül a parancsfájlok írására szánt időd. Mindamellett ha a hozzáférésedet perc alapján számlázzák (mint nekem), még pénzt is megtakaríthatsz!

Az elemezni kívánt HTML-fájlt meg kell nyitnod, és ki kell találnod, hogy milyen szabványos kifejezéssel ollózhatod ki belőle a szükséges elemeket. A Perl ***LWP*** könyvtára alaphelyzetben függvényeket kínál, amelyek segítségével egy HTML-fájlból akár a teljes szövegrészt kiemelheted. Ha a dokumentumnak csak az ASCII-változatára van szükséged, akkor máris elindulhatsz.

Ilyen esetekben rendkívül csábító az ***LWP*** használata, mivel az összes szöveget kivágja a dokumentumból, és ezzel már könnyen dolgozhatsz. Ez a szolgáltatás akkor is jól jöhet, ha az oldalból csupán néhány sorra van szükséged, mivel sokkal egyszerűbb a kivágott szövegen dolgozni, mint a teljes HTML-fájlon. Ez a módszer azonban sok esetben mégis jóval bonyolultabb feldolgozást eredményez. Természetesen a tiszta ASCII-szöveg sokkal könnyebben olvasható, ám a dokumentumból a HTML-jelölések elvesznek, pedig ezekkel könnyebben meghatározható lenne, hol kezdődik az érdekes rész.

Kezdjük mindjárt a legegyszerűbb példával: tegyük fel, hogy csak a hírekre van szükségünk, melyek `<H1>` és `</H1>` tagok között helyezkednek el. Ezeket a tagokat egy szabályos kifejezéssel könnyedén megtalálhatod, nélkülük viszont elég nehezen lehetne rávenni a programot, hogy felismerje a híreket tartalmazó részt.

Hogy valós helyzetben mutassuk be a példánkat, próbáljuk meg az FSF híroldalán ➲ http://www.fsf.org/news/news.html található címeket közvetlenül a saját terminálunkon kinyomtatni. Ha a programunkat elküldjük erre a címre, az oldal teljes tartalmát menteni fogja a `$HTML_FILE` változóba. Most pedig alkalmazzuk a következő szabványos kifejezéseket (javaslom, hogy előtte nézd meg a kérdéses oldalt és a forráskódját, hogy megérthesd, miről is van szó):

```
$HTML_FILE =~ s/.*>Press Releases<//gsmi;
$HTML_FILE =~ s/.*<DL>//gsmi;
$HTML_FILE =~ s/<\/DL>.*$//gsmi;

$HTML_FILE =~ s/<dt>([^<]*)<\/dt>/->
➥$1: /gi;

$HTML_FILE =~ s/<dd><a href=[^>]*>([^<]*)<\/a>/
➥$1 /gsmi;
$HTML_FILE =~
➥s/\.\s+\([^\)]*\.\)<\/dd>/<DD>/gsmi;

$HTML_FILE =~ s/\s+/ /gsmi;
$HTML_FILE =~ s/<DD>/\n/gsmi;
```

Az első három sor minden lényegtelen dolgot levág, ami nem tartozik a hírek közé. A negyedik sor megkeresi a dátumot, és levágja róla a HTML-címkéket. A következő két sor ugyanazt teszi hírek címével. Az utolsó két sor eltávolítja a felesleges szóközöket, és a szükséges helyeken a szöveget új sorokra tördeli. Ezen a 2001. december 14-i napon a héjamban a következő látható (a kimeneten az olvashatóság végett kicsit változtattam):

```
-> 3 December 2001: Stallman Receives
    ➥Prestigious...
-> 22 October 2001: FSF Announces Version 21
   ➥of the...
-> 12 October 2001: Free Software Foundation
   ➥Announces...
-> 24 September 2001: Richard Stallman and
   ➥Eben Moglen...
-> 18 September 2001: FSF and FSMLabs come
   ➥to agreement...
```

A fenti kifejezéslista nem teljes, például a hírek frissítését nem kezeli. A kifejezéseket a kisebb HTML-módosításoktól (például a színek cseréjétől, betűtípusok méretétől stb.) lehetőség szerint függetleníteni kellene. A következő szabályos kifejezés minden betűtípusokkal kapcsolatos jelölést eltávolít:

```
$HTML_FILES =~ s/<font face="Verdana"
➥size="3">([^<]+)<\/font>/$1/g;
```

Ez ugyanazt teszi, de bármilyen fajtájú és (pozitív) méretű betűtípus esetén működik:

```
$HTML_FILES =~ s/<font face="[^"]+"
➥size="\d+">([^<]+)<\/font>/$1/g;
```

Az itt bemutatott példák szemléltetik a módszer alapelveit, és mint már szó volt róla, egyszeri befektetéssel a későbbiekben sok-sok időt nyerhetünk.

Hírek megjelenítése a saját képernyődön

Ha már sikerült az értékes szöveget valamilyen oldalról kinyerned, természetesen nem vagy arra korlátozva, hogy csak a saját konzolodon, egyénileg használd fel. Amennyiben valami mást is szeretnél tenni, például mindig értesülni arról, ha mondjuk ***Stallman***-tól jelenik meg valami, csak három lépés szükséges hozzá. Első lépésben vedd fel a parancsfájlt a `cron`-bejegyzéseid közé (ezzel kapcsolatban a `man cron` parancs mindent elmond), ezután pedig a programodhoz add a következő ellenőrzést:

```
if ($HTML_FILE =~ m/Stallman/) {
        #  RTES˝T S K LD SE
}
```

Így a parancsfájl a teendőit csak akkor végzi el teljesen, ha a megadott feltétel teljesül, és a hírekben szerepel Stallman neve (vagy természetesen bárki másé, akiét beállítjuk). A következő lépésben ezt a pár sort írjuk be a kapcsos zárójelek közé:

```
open (XMSG, "|/usr/bin/X11/xmessage
➥-title \"NEWS!\" -file -") || die;
print XMSG $HTML_FILE;
close XMSG;
```

Ez a néhány sor megnyit az X alatt egy ablakot a `-title` után

megadott címmel, és a `-file` tulajdonság után megadott fájl tartalmával. Esetünkben a `-tulajdonsÆg` azt jelenti, hogy a program a szöveget az állandó bemenetről olvassa be. Ezekután a Perl-parancsfájl addig vár, amíg az `xmessage` ablakot be nem zárod. Talán pont erre van szükséged. Nem szabad azonban arról sem megfeledkezni, hogy a `cron`-ból futunk, tehát jobb megoldás, ha az `xmessage`-t a háttérben futtatjuk egy ideiglenes fájlon, majd pedig kilépünk:

```
open (XMSG, "> /tmp/gee") || die;
print XMSG $HTML_FILE;
close XMSG;
exec "/usr/bin/X11/xmessage
➥-title \"NEWS!\" -file /tmp/gee&";
```

Ellenőrizzük, hogy bizonyos idő elteltével megváltozott-e az oldal!

Amennyiben az oldalt csak akkor szeretnéd feldolgozni, ha az utóbbi látogatásod óta vagy az elmúlt két órában megváltozott a tartalma, a ***Last-Modified*** HTTP-fejlécre lesz szükséged. Ez már `@HEADER` tömbünk 3. elemeként rendelkezésre áll. A hozzá tartozó érték 1970. január 1-je óta a másodperceket számolja. Ezért ha csak olyan oldalakra van szükséged, amelyek az utóbbi két órában módosultak, számold ki az időt, amit ez a számláló két órája mutatott (mindig az „eltelt másodpercek” egységben):

```
$NOW = time;
$TWO_HOURS_AGO = $NOW - (3600 2);
```

Majd ezt az időt hasonlítsd össze a weblap módosítási idejével:

```
if ($HEADER[2] > $TWO_HOURS_AGO) {
        # a sz ksØges feladat elvØgzØse
}
```

Dinamikus könyvjelzők hozzáadása az ablakkezelő menüjéhez

Ez a művelet azon kevés esetek egyike, amikor a csináld magad szabály alól kivételt kell tennünk: töltsd le a WMHeadLines nevű programot (lásd a Hivatkozások részt), majd telepítsd fel, és az ízlésednek megfelelően állítsd be. A program 120 különböző webhely híreihez nyújt hozzáférést, amelyeket aztán elhelyez a BlackBox, a WindowMaker, az Englightenment és a Gnome menüiben, oly módon, hogyha kattintasz rajtuk, a böngészőt elindítva a kért oldalakra jutsz.

A böngésző vezérlése parancsfájlból

A Netscape-nek például többféle parancsot kiadhatunk a héjból vagy akár egy parancsfájlból. Egy ilyen paranccsal a Netscape arra utasítható, hogyha eddig még nem futott, induljon el és jelenítse meg a kért oldalt, illetve amennyiben már futott, az oldal a pillanatnyi vagy pedig új ablakban jelenjen meg. Hogy milyen parancsot kell használnunk, attól is függ, hogy a böngésző éppen fut-e. Vess a WMHeadLines csomagban egy pillantást az ***nslaunch.pl*** parancsfájlra, és meglátod, miként kérdezhető le, hogy a Netscape fut-e már.
A Netscape parancsfájljaidból egyéb feladatok elvégzésére is utasítható, például miután behozta a választott oldalt, a böngésző segítségével nyomtathatunk is:

```
exec($NETSCAPE, ·-noraise·, ·-remote·,
➥"openURL($URL,new-window)");
```

Az oldal PostScriptbe mentése:

```
exec($NETSCAPE, ·-noraise·, ·-remote·,
➥"saveAs(/tmp/netscape.ps, PostScript)");
```

Végül pedig nyomtatása:

```
exec("mpage -PYOURPRINTER -1 /tmp/netscape.ps");
```

Akár a könyvjelzők közé is felvehetjük:

```
exec($NETSCAPE, ·-noraise·, ·-remote·,
➥"addBookmark($SOME_URL, $ITS_TITLE)");
```

A Konquerort, a KDE webböngészőjét egyszerűen a következő módon indíthatjuk el:

```
system("/usr/bin/konqueror $URL");
```

A Konqueror nagyon jól kezelhető parancsfájlokból áll, és nemcsak a Webre jellemző feladatokat végezhetünk vele, hanem akár fájlokat is másoltathatunk, vagy eszközöket fűzhetünk be. Működésének megismeréséhez írd be a következőt:

```
kfmclient -commands
```

A Galeon is ugyanígy indítható:

```
system("/usr/bin/galeon $URL");
```

Mint ahogyan az „A User’s Guide to Galeon” cikkben látható (Hivatkozások rész), még azt is eldöntheted, hogy az új hely egy új fül alatt nyíljon-e meg:

```
system("/usr/bin/galeon -n $URL");
```

Esetleg egy teljesen új ablakban:

```
system("/usr/bin/galeon -w $URL");
```

Vagy akár egy ideiglenes könyvjelzőt is készíthetünk:

```
system("/usr/bin/galeon -t $URL");
```

Okos böngészés

Egy ellentétes megközelítés, mint például az általános tükrözés, vagy a képek letöltése elvégezhető a böngészőből is, például a Konquerorból vagy a KMailből. Ha a jobb egérgombbal kattintasz egy hivatkozáson, a megjelenő menüből kattints a „Megnyitás ezzel…” menüpontra, és ha beírod a programod elérési útját, a következőkben már önmagától fel fogja ajánlani. Ez annyit jelent, hogy a `fetch_images` parancsfájllal néhány kattintással elkészítheted az oldalakról a saját másolatodat, amennyiben követed az utasításokat és a programodat a háttérben futtatod.

Okos tükrözés és FTP

Az `@ALL_URLS` tömbben található URL-ekkel FTP-oldalakról is készíthetsz tükrözést. Ez a Perlből is teljességgel elvégezhető, a sok-sok FTP-ző és tükröző modulok valamelyikének felhasználásával, vagy pedig olyan módon, hogy a szükséges címeket összegyűjtjük, és a dolog lényegi részét a `wget`-re vagy a `curl`-re bízzuk, mint az ***A. J. Chung*** „Downloading without a browser” című cikkében látható (Hivatkozások rész).
Ha a kedvenc portálod napról napra változtatja a kinézetét, és te mégis le szeretnéd tölteni magadnak, csak mentsd az oldal

címét, ahogyan képek esetén tennéd, majd a programodban add ki a következő parancsot:

```
exec "wget -m -L -t 5 $COMPLETE_URL";
```

Az URL-eket a Perl programnak tulajdonságként továbbadva az összes szükséges parancs végrehajtható, amely a párhuzamos FTP-zéshez, vagy a tükrözéshez szükséges.

Építsd fel a saját portálodat!

Sokunknak több kedvelt oldala is akad, és az összes kedvencünket egyetlen ablakban szeretnénk látni. Általános megoldás, ha a HTML-törzset minden oldalból a következő módon vágod ki:

```
$HTML_FILE = s/^.*<body[^>]*>//i;    # levÆg
➥mindent a HTML t rzs elıtt
$HTML_FILE = s/<\/body[^>]*>.*$//i; #  Øs utÆn
```

Ekkor kinyomtat egy HTML-táblát, amelynek minden egyes négyzetében az oldalak egyenként találhatók meg:

```
print<<END_TABLE;
 {{Ide j n minden HTML HEAD Øs BODY dolog}}
<TABLE>
<TR><TD>$HTML_FILE_1</TD></TR>
<TR><TD>$HTML_FILE_2</TD></TR>
.........
</TABLE></BODY></HTML>
END_TABLE
```

A programot ***myportal.html*** néven mentsük a saját könyvtárunkba, a böngésződben erre az oldalra állítsd be a kezdőoldalt, és gyönyörködj benne! A teljes programnak szüksége lehet arra, hogy az oldalak CSS- és karakterkészlet-beállításain csiszoljon, de mostanra már te is képes vagy erre, tudod?

Összegzés

Éppen csak érintettük az ügyféloldali héjprogramozás felszínét. Ezeken kívül még sok egyéb bonyolult megoldás is elképzelhető, például a sütik és a jelszóval védett oldalak kezelése, önműködő űrlapkitöltés, keresés a weben bármilyen jellemzőkkel, weboldalak elemzése, és a tíz legtöbbet kiválasztott hivatkozás megjelenítése, vagy akár a webes levélellenőrzés. Mindehhez csak egy kis türelem szükségeltetik, egy csipetnyi Perl-gyakorlat, és a rendelkezésre álló modulok ismerete.
Jó böngészést kívánok!

Marco Fioretti
alkatrészrendszerek fejlesztésével foglalkozik hálózati kábel nélküli ATM-eszközökhöz az Ericsson Labnél Olaszországban. Érdekli még a természetjárás, a hegymászás, a Perl, a webdesign, az etika és az új módszerek közötti kapcsolat, továbbá a Linux-asztal hatékonnyá varázsolása. Marco Rómában él családjával, és a linuxdesk@inwind.it címen érhető el.