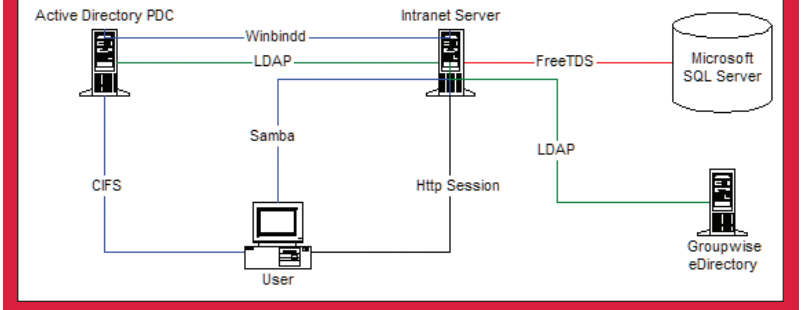
Irodai intranet – kicsiben

Ebben a cikkben értékes tanácsokkal kívánok szolgálni vállalati szolgáltatások felhasználóbarát intranetes integrálásához.

■ Az ***intranet*** már elég régóta meghatározza az életünket. Egyike volt a ***World Wide Web*** alternatív felhasználásának a 90-es évek elején. Noha a web házon belüli alkalmazása vonzó volt, nehézséget jelentett a már meglévő szoftvereszközök integrálása. Emiatt sok ***intranet*** csupán egyszerű hirdetőtáblaként funkcionált. A helyzet mostanra eléggé megváltozott, hiszen a nyílt forrásnak köszönhetően egyszerűbbé és költséghatékonyabbá vált az ***intranet***ek beállítása. Az úgynevezett ***LAMP*** kiváló felületet nyújt számos alkalmazás közös felhasználói felület alá történő integrálásához. Ezt próbáltuk mi is megvalósítani. 1999-ben egy webalapú hirdetőtáblával és egy céges naptárral indult ***intranet***ünk, mely ***Apache*** alól, egy ***Red Hat 6.0*** szerveren kapott helyet. Statikus ***HTML*** oldal volt, amelyet a marketing igazgatónk tervezett és tartott karban. Miután 2002-ben kilépett a cégtől, olyan ***intranet***et szerettünk volna, amely nem egy személytől függ. Ahogy az lenni szokott, újabb és újabb funkciókkal ruháztuk fel a rendszert az évek során. Jelenlegi állapotában nagyon hasznos és felhasználóbarát ***intranet***ünk van mindenféle felesleges vagy statikus tartalom nélkül, amelyek megnehezítenék a karbantartást. Ebben a cikkben saját példákon keresztül fogom bemutatni, hogyan oldottuk meg a négy leggyakoribb integrációs problémát ***LAMP*** környezetben.

Technikai áttekintés

Jelenlegi – 70 alkalmazottat kiszolgáló – ***intranet***ünk egy ***IBM x335***-ös szerveren fut ***Fedora Core 4*** operációs rendszer alatt. A hagyományos ***LAMP*** eszközöket (***Linux, Apache 1.3x, MySQL, Perl***)használjuk ***mod_perl*** modullal kiegészítve a megfelelő teljesítmény érdekében. A szerveren az ***Apache*** mellett e-mail ellenőrző, belső ***DNS szerver***, ***Jabber***, ***Samba*** és pár egyéb szolgáltatás is fut. A közös szervernek köszönhetően egyszerűbb a hálózati meghajtók kezelése és a hálózati forgalom is kisebb. Néhány cégnek ez bizonyára szűk keresztmetszetet nyújtana, de az elveket figyelembe véve hasonlóképp kivitelezhető többszerveres környezetben is. Minden felhasználó ***Windows XP***-t használ és az ***Active Directory*** azonosítja őket. ***GroupWise***-t használunk email fogadásra egy ***NetWare 6***-ot és ***Novell eDirectory***-t futtató szerveren. Van továbbá egy ***Windows NT 4.0*** szerver ***Microsoft SQL*** adatbázissal, amely a munkaidő nyilvántartásért és a számlázásért felel. Az *1. ábrán* látható, hogy függnek össze a szolgáltatások.



■ ***1. ábra*** Céges szolgáltatások kapcsolata

Szerveroldali hitelesítés

A felhasználóknak ne kelljen mindig azonosítaniuk magukat az ***intranet*** felé, ezt már az elején kikötöttük. A szerver automatikusan „tudja” az IP cím és a munkaállomás bejelentkezési adatai alapján, hogy ki használja éppen arról a gépről az ***intranet***et. Mi ezt szerveroldali hitelesítésnek (***SSC – Server-Side Credentialing***) hívjuk. Eredetileg egy saját készítésű kliensoldali alkalmazással oldottuk meg, amely a szerveroldali ***CGI*** szkript kéréseire válaszolt, valahányszor szükség volt azonosításra. Noha működik, túl sok bizalmat vet a kliensoldalba. Egy lehallgatóprogram és egy ***Perl*** szkript segítségével például bármely kliensről be lehet csapni a szervert. Most ***Samba***-t és ***winbindd***-t használunk ugyanerre.
Minthogy az ***intranet*** szerverünk a megbízható belső hálózatra csatlakozik, így minden hálózati ténykedés – ki honnan lépett be – titokban marad. Minden irodai számítógép bejelentkezéskor hozzárendel egy meghajtó-betűjelet a ***Samba*** szerverhez. Ezután a kapcsolódások listája alapján egyértelműen azonosítható, hogy ki honnan lépett be. A meghajtó betűjel csupán a szerveroldali hitelesítés miatt szükséges. Úgy vélem, ez egy

■ *2. ábra* Egy példa a felhasználói profilra az Intraneten

lényeges tulajdonság, hiszen egyszerűbbé teszi a regisztráció és belépés nélküli böngészést. A felhasználóknak már így is elég sok felhasználónevet és jelszót kell észben tartaniuk.
A szerveroldali hitelesítés beállítása nagyban függ a felhasználók azonosítási módjától. Mi az ***Active Directory***-t használjuk, így azzal fogom bemutatni. Az ***Active Directory*** bosszantó egy kicsit(nem túl meglepő), hiszen nem tárol információkat a kapcsolatokra vonatkozóan. Hagyományos ***RPC hívásokat*** kell használnunk a ***Samba*** net parancsával, hogy megbízható eredményeket kapjunk. A szerveroldali hitelesítő szkriptünket ***smbconn.sh***-nak hívjuk, és valahogy így néz ki:

```
#!/bin/sh
net status sessions parseable  \
| grep -i "\\\\$1\\\\"         \
| sed 's/^.*\\\(.*\)\\.*\\.
➥*\\.*$/\1/g'   \
| sed 's/DOMAIN+//g' | tr -d ' '
```

Elég egyszerű, nem? Csupán a `DOMAIN`-t kell kicserélni az általunk használt ***Active Directory*** domainjére. A szkript az általunk megadott IP-ről belépett felhasználói objektumot adja válaszként. Ez az objektum az ***Active Directory sAMAccountName*** mezőjét takarja. Ezzel az információval felvértezve már indíthatunk ***LDAP*** keresést, amely visszaadja a felhasználó teljes nevét, de akár bármi mást, amire szükségünk van. Ezt a szkriptet az *1. Lista* tartalmazza.
Első paraméterként a felhasználó ***sAMAccountName*** értékét várja, második paraméterként pedig egyéb opcionális attribútumot adhatunk meg. Ha nem adunk meg opcionális attribútumot, akkor a felhasználó teljes nevével tér vissza. Ehhez használhatunk egyéni ***mod_perl*** kiegészítést, így az információ mindig elérhető lenne, de úgy tűnik, hogy ez a legtöbb esetben feleslegesen bonyolítana a dolgokon. A mi esetünkben csupán pár részre korlátozódik a dolog, így minden ***CGI*** programot igény szerint futtatunk. A szerveroldali azonosítás kérése így néz ki az egyik ***CGI*** programunkban:

```
##: Get this connection's user
➥credentials
my $remoteip=$ENV
➥{'REMOTE_ADDR'};
open(SMBCONN,"smbconn.sh
➥$remoteip |");
my $cn=<SMBCONN>;
$cn=~s/\s+//g;   ##: Strip
➥whitespace
close(SMBCONN);
open(GETEMPINFO,"getempinfo.pl
➥$cn |");
my $username=<GETEMPINFO>;
close(GETEMPINFO);
if($username eq "") {
  $username="Guest";
}
```

Ez a kódrészlet a felhasználó `sAMAccountName` értékét adja vissza a `$cn` változóban és a teljes nevét a `$username` változóban. Ha a `$username` tartalma `Guest`, akkor sikertelen volt a keresés vagy a kliens számítógépen nincs belépett felhasználó. Most már minden létfontosságú információnk megvan, hogy eldöntsük, jogosult-e a felhasználó megtekinteni az oldalt vagy sem. Ezzel az is lehetővé válik, hogy a felhasználó számára egyedi oldalt jelenítsünk meg. Ezt az ***intranet*** oldalunk ***index.cgi*** fájljának egy részével demonstrálnám:

```
##: My Intranet section
my $mint="";
if(($username eq "Guest") ||
➥($username eq "")) {
  open(EMPSNAP,"./random-
  ➥employee.pl 2>&1 |");
  my @snap=<EMPSNAP>;
  close(EMPSNAP);
  $mint.=join("\n",@snap);
  chop($mint);
} else {
  $mint.=&get_emp_card($cn);
  $mint.="<b>E-Mail
➥Controls:</b><br>\n";
  $mint.="<a href=
  ➥'selfserv.cgi'>My Mail
  ➥</a>\n";
  ...
}
...
print STDOUT $mint;
```

Látható, hogy ellenőrizzük, vajon az oldalt megtekintő személy azonosított felhasználó-e. Ha nem, akkor véletlenszerűen megjelenítjük véletlenszerűen egy munkatársunk fotóját és adatlapját az oldal adott részén. Ha azonosított, akkor az *LDAP*-ból lekérjük a megfelelő információkat és összeállítjuk a *My Intranet* (az *Intranet*em) oldalt. Ezen az oldalon a felhasználó módosíthatja a profilját, megnézheti a levelezését, stb. A `get_emp_card($cn)` eljárás csupán kikeresi a felhasználó adatait az *Active Directory*-ból és egy *HTML* formátumú részt jelenít meg (2. *ábra*).

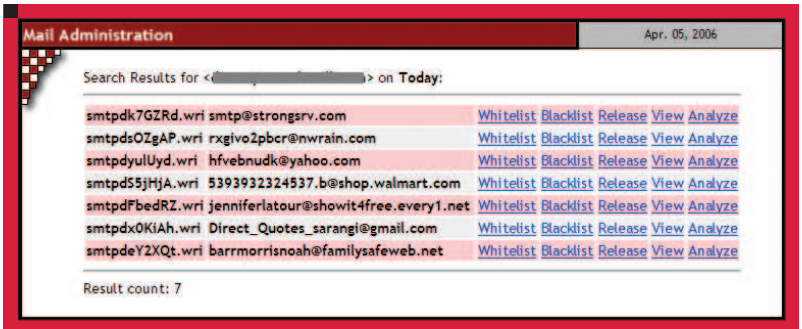
Az Active Directory integrálása

Az *intranet*ünk másik hasznos bővítése: *Active Directory* felhasználói adatbázis elérése *LDAP*-on keresztül. Ezt használjuk a cégnél dolgozó összes munkatárs listázásakor. Az adatbázist valós időben építjük fel, valahányszor szükség van rá, ezáltal tehermentesítve a rendszergazdákat. Valahányszor új felhasználó kerül az *Active Directory*-ba, azonnal látszik az *intranet*es listában is. Felhasználóink maguk szerkeszthetik adataikat, melyek aztán egy *CGI szkript* segítségével kerülnek be az *Active Directory*-ba. Az eljárás magától értetődő, azonban néhány dolgot figyelembe kell venni. Bemutatom lépésről-lépésre, mi hogyan csináltuk.
Először is létrehoztuk az *Active Directory*-ban a *proxyuser* felhasználót. Ezzel a felhasználónévvel azonosítja magát szkriptünk az *LDAP* felé. A *proxyuser* írhatja és olvashatja azokat a felhasználói objektumokat, melyek ou=Domain Users konténerében vannak. Ezt az *Active Directory*-n belül kell végre hajtanunk. A *CGI* programjainkhoz *Perl*-t használunk, így az *LDAP* elérése a *Net::LDAP* könyvtáron keresztük történik. *CGI* szkriptünkkel így kapcsolódhatunk az *Active Directory*-hoz:

```
##: Active Directory connection
use Net::LDAP;
my $ldap=Net::LDAP->new
➥('adserver.domain.com');
my $mesg=$ldap->bind
➥('proxyuser@domain.com',
            password=>
             ➥'proxyuser'  );
```

1. Lista getempinfo.pl

```
#!/usr/bin/perl -w
use Net::LDAP;
use strict;
my $cn=$ARGV[0] || "none";
my $attr=$ARGV[1] || "none";
##: If nothing was given on command line then return
if($cn eq "none") {
  print STDERR "ERROR: No LDAP cn given\n";
  exit(1);
}
##: Bind anonymously to the ldap database
my $ldap=Net::LDAP->new('directory.domain.com',timeout=>5)
  or die "Couldn't connect to directory server.\n";
my $mesg=$ldap-
>bind('proxyuser@domain.com',password=>'proxyuser')
  or die "Couldn't connect to directory server.\n";
##: Query LDAP to get a list of employees
if($attr ne "none") {
  $mesg=$ldap->search( base=> "ou=Domain
Users,dc=domain,dc=com",
                       filter=> "(sAMAccountName=$cn)",
                       attrs=> ['givenName','sn',"$attr"] );
} else {
  $mesg=$ldap->search( base=> "ou=Domain
  ➥Users,dc=domain,dc=com",
                       filter=> "(sAMAccountName=$cn)",
                       attrs=> ['givenName','sn'] );
}
my $count=$mesg->count();
($count==1) or die "Error: LDAP enumeration error.";
my $entry=$mesg->entry();
my $value;
my @values;
if($attr ne "none") {
  $value="";
  @values=$entry->get_value("$attr");
  my $i=1;
  for(@values) {
    if($i>1) {
      $value.="/$_";
    } else {
      $value.=$_;
    }
    $i++;
  }
} else {
  $value=($entry->get_value('givenName')." ";
  $value.=$entry->get_value('sn'));
}
##: See if that attribute was defined for the given cn
if(!(defined($value))) {
  print STDERR "ERROR: That attribute was not defined.\n";
  exit(1);
}
$mesg=$ldap->unbind;
print("$value\n");
```



■ *3. ábra* Műveletek az elfogott spamekkel

Az ***Active Directory*** ilyen formátumban várja a felhasználónevet. Ez egyike azoknak, amelyet az ***Active Directory LDAP*** felülete igényel. Kapcsolódás után már lekérhetjük a `ou=Domain Users` konténerben található felhasználók listáját:

```
##: Query LDAP to get a list of
➥employees
my $basedn="ou=Domain
➥Users,dc=domain,dc=com";
my $filter="(objectClass=user)";
$mesg=$ldap->search(
  base=> $basedn,
  filter=> $filter,
  attrs=> ['givenName','sn',
   ➥'mail',
        'telephoneNumber',
         ➥'streetAddress',
        'l','st','department',
         ➥'postalCode',
        'employeeNumber',
         ➥'homePhone',
        'title',
         ➥'sAMAccountName' ]
  );
```

Megkapjuk a listát az összes felhasználóról az összes kapcsolódó attribútummal egyetemben. Tovább finomíthatjuk a keresést azzal, hogy csupán azokat a dolgozókat listázzuk, akiknek a vezetékneve a ***CGI szkript*** által kapott betűvel kezdődik. Így címjegyzékszerű végeredményt kapunk és nem kell mind a 70 felhasználót egyszerre megjeleníteni. Ha a ***CGI szkript*** nem kapott kezdőbetűt, akkor az *„a”* betűs neveket jeleníti meg:

```
##: Get letter requested in the
➥URL
my $letter;
$letter=param('letter') || "a";
...
my $filter="(&(objectClass=user)
➥(sn=$letter*))";
```

Ha az Olvasó kevéssé járatos az ***LDAP*** keresésekben, akkor mindenképp érdemes átolvasni az ***RFC-2254***-es dokumentumot. Ez az a pont, ahol végignézzük a kapott listát és igény szerint csinosítjuk. Lekérdezhetjük a munkatársakhoz tartozó ***sAMAccountName*** értéket is. Ha a találat megegyezik az azonosított személlyel, akkor kitehetünk egy linket, és arra kattintva módosítani tudja az adatait. Ez valahogy így néz ki:

```
##: Display the directory
foreach my $entry ($mesg->
➥sorted('sn')) {
  my $san=$entry->
   ➥get_value('sAMAccountName');
  $empdir.="<div class=
   ➥'empcard'>";
  if(lc($cn) eq lc($san)) {
    ##: This is our man.
➥ Add a button.
    $empdir.="<a href=
     ➥'empedit.cgi'>Edit</a>";
  }
  $empdir.="<span id='name'>";
  $empdir.=$entry->get_value
   ➥('givenName')." ";
  $empdir.=$entry->get_value
   ➥('sn');
  $empdir.="</span><br>";
  $empdir.="<span id='title'>";
  $empdir.=$entry->get_value
   ➥('title').";
  $empdir.="</span><br>";
  ...
  $empdir.="</div>";
}
print STDOUT $empdir;
$mesg=$ldap->unbind();
```

SpamAssassin és E-mail integrációja

2001-ben állítottuk üzembe a cég email átjáróját, és azóta is azt használjuk. A ***Linux Journal 2001*** decemberi számában írtam róla egy cikket. Rengeteget változott azóta, de az alapok ugyanazok maradtak. Egyszerűen tárol, ellenőriz és továbbít. Minthogy mindez a ***Linux***os szerveren belül bonyolódik, így a ***Windows*** felhasználók nem férnek hozzá a személyes ***SpamAssassin*** konfigurációjukhoz. Ezt néhány ***CGI szkript***tel oldottuk meg, melyekkel a felhasználók testre szabhatják ***SpamAssassin*** beállításaikat. A felhasználók beállításaikat a ***My Intranet*** megfelelő részén érik el (*2. ábra*). A lenyíló menüből kiválaszthatják, melyik napra kíváncsiak. A gombra kattintva elindul a ***selfserv.cgi*** szkript. A szkript nem vár semmiféle azonosítást, hiszen azt már korábban a szerveroldali hitelesítéssel megoldottuk. A kezdeti lekérdezés után újra meghívjuk a ***getempinfo.pl***-t az alábbi módon, hogy kiderítsük az email címét:

```
##: Get this user's email address
open(GETEMPINFO,"-|",
➥"getempinfo.pl",$cn,"mail");
my $searchstring=<GETEMPINFO>;
close(GETEMPINFO);
```

Ezután a `$searchstring` változót – mint reguláris kifejezésünk alapját – fogjuk a */spam* könyvtárban történő kereséskor felhasználni. Minthogy az ***Active Directory*** email mezője szabadon módosítható, így az esetleges elgépelések ellen védekeznünk kell:

```
##: Make sure this email address
is valid
unless($searchstring=~/^[a-z]*\
➥@domain\.com$/) {
  print STDOUT "Content-Type:
➥text/plain\n\n";
  print STDOUT "Access Denied:
➥Your identity on \
    the network can't be
➥verified.\n";
  return(0);
}
```

Ha sikeresek voltak az ellenőrzések, akkor a szkript megjeleníti a kért nap spamforgalmát. Minden elem mellett számos link található. Ezek lehetővé

teszik a spam átengedését, a feladó engedélyezését vagy tiltását, *SpamAssassin* jelentés elkészítését vagy csupán az email tartalmának megjelenítését. Hogy a felhasználó mit tehet és mit nem tehet, azt jelen esetben is a szerveroldali hitelesítés segít eldönteni. Részletesebben nem tárgyalom a dolgot, hiszen a szkriptek csupán a fájlok megfelelő áthelyezését végzik. Szeretnék viszont a (engedélyező és tiltó) listákról bővebben beszélni.
A *SpamAssassin* felhasználói beállításai a felhasználó könyvtárában, a *.spamassassin/user_prefs.cf* fájlban találhatóak. Hétköznapi esetben, ha linuxos a levélszerver, akkor ez megfelelő. A mi esetünkben azonban nem. A *Linux* szerverünk csupán a be és kimenő levélforgalmat vizsgálja, nem rendelkezik információval a felhasználókról és email címeikről. A megoldás érdekében cselhez folyamodtunk. A *SpamAssassin* rendszerszintű konfigurációja a */etc/mail/spamassassin/local.cf* állományban található, amelyet minden induláskor beolvas. Valójában az összes *.cf* végződésű állományt végignézi a */etc/mail/spamassassin* könyvtárban. Ezt a javunkra fordíthatjuk és *CGI szkript*ünkkel itt hozzuk létre a felhasználói listákat `$cn_prefs.cf` formátumban. A spamd-t cron-nal óránként újraindítjuk memória-felszabadítás végett, így ezzel nem lesz gond. Amennyiben ezt a megoldást használjuk, figyeljünk oda, nehogy a felhasználók `*@hotmail.com` vagy ehhez hasonló *SpamAssassin* feltételeket hozzanak létre. Ezek ugyan személyes állományok, mégis kihatással vannak a *SpamAssassin* globális működésére, hiszen a fő konfigurációs könyvtárban találhatóak.

Microsoft SQL Server integrálása

Cégünk a *CPAS* munkaidő nyilvántartó és számlázó rendszert használja. Ez a szoftvercsomag tartalmazza az összes partnerünk és számlánk információit, valamint a marketing igazgatónk által a reklámkampányoknál felhasznált adatokat is. Szerettük volna biztosítani munkatársaink számára a hozzáférést néhány alapvető adathoz. Mindezt az ügyfélszolgálat megkerülésével, hogy időt takarítsunk meg. A *CPAS Microsoft SQL*-t használ, így az integráláshoz segítségül kellett hívnunk a *FreeTDS*-t és a *DBD::Sybase Perl* modult.
A beállítás pár egyszerű lépés. Először le kellett töltenünk a legfrissebb *FreeTDS* csomagot az *Internet*ről, és ki kellett csomagolnunk.
A könyvtárba belépve kiadtuk az alábbi parancsokat:

```
> ./configure -prefix=/usr/
➥local/freetds
> make
> su -c 'make install'
```

Ez a megfelelő helyre telepítette a *FreeTDS*, így egyszerűbb dolgunk lesz a *Sybase* modulnál. Következő lépésként letöltöttük a *CPAN*-ról a *DBD:Sybase* csomagot. Rendszergazdaként az alábbi parancsokat adtuk ki:

```
> perl -MCPAN -e shell
> install DBD::Sybase
```

Ha néhány teszt hibát jelez, nyugodtan folytassuk a telepítést, a csomag szerzője szerint ez elég gyakori.
A program ezután már a gépen van, de meg kell ejtenünk még a *FreeTDS* konfigurálását is. A konfigurációs fájlban adjuk meg az adatbázisok adatait, amelyekhez kapcsolódni szeretnénk. Az állomány jól dokumentált és a logikáját is elég könnyű átlátni. Így néz ki egy bejegyzés:

```
[JACKSON5]
   host = jackson5.domain.com
   port = 1433
   tds version = 4.2
```

Ha a *FreeTDS*-t beállítottuk, akkor már elérjük *Perl*-ből az adatbázist a klasszikus *DBI* interfészen keresztül. Íme egy példa, amelyben a *JACKSON5* nevű *Windows* szerverünkön lévő *concerts* adatbázishoz kapcsolódunk:

```
#!/usr/bin/perl -w
use DBI; $ENV{'SYBASE'} = '/
➥usr/local/freetds';
$dbh = DBI->connect
➥('dbi:Sybase:server=JACKSON5',
➥ 'username', 'password')
   or die 'connect';
$dbh->do("use concerts");
```

Vegyük észre: környezeti változóként meg kell adnunk a *FreeTDS* könyvtárát is mielőtt csatlakoznánk. A változó közli a *DBD::Sybase* modullal, hol találja a *FreeTDS* modulokat. Ezután már a megszokott módon használhatjuk a *DBI* felületet.
Ha eddig *MySQL*-lel dolgozott a kedves Olvasó, mindenképp nézze át alaposabban a *Microsoft SQL Server* használatára vonatkozó információkat. Néhány dolog igencsak más a megszokottakhoz képest.

Zárszó

Reményeim szerint sikerült pár hasznos ötletet adnom, hogyan lehet hatékonyan integrálni *intranet*té a legalapvetőbb szolgáltatásokat, amelyek egy irodában előfordulhatnak. Az *intranet* sokkal több egy elektronikus hirdetőtáblánál. Megfelelő *intranet* időt takarít meg a rendszergazdák számára, de munkatársak is hatékonyabban végezhetik munkájukat. A felhasználók számára ugyanis sokkal egyszerűbb webböngészővel dolgozni, mint parancssorból. Kovácsoljunk ebből előnyt és az *intranet* cégünk értékes befektetésévé válik.

Linux Journal 2006., 151. szám

Dave Jones a Pearce, Bevill, Leesburg & Moore cég informatikai igazgatója Birmingham-ban, Alabama államban. Nyolcéves hálózatadminisztrátori múlttal rendelkezik. Szabadidejében webes naplót ír és programokat fejleszt
➲ www.sector62.com

KAPCSOLÓDÓ CÍMEK

FreeTDS
➲ www.freetds.org

RFC-2254
➲ ftp.rfc-editor.org/in-notes/rfc2254.txt

E-mailes vírusok szűréséről szóló cikkem
➲ www.linuxjournal.com/article/4882

CPAS
➲ www.cpasoftware.com

Pearce, Bevill, Leesburg & Moore P.C.
➲ www.pearcebevill.com