Mikor a Linux a Windows megmentője...

„A mentést a /dev/null-ra irányítom – így sokkal gazdaságosabb minden szempontból: nem kell szalagokat cserélgetni ötpercenként, ráadásul még gyorsítja is a mentési folyamatot, tehát csak jó lehet.” – A pokoli operátor naplója, első nap.

A *Pokoli Operátor* naplója egyike az informatikus humor legnépszerűbb gyöngyszemeinek. Egy végtelen történet, amelyben a főhős a felhasználók kínzásának újabbnál újabb módszerei kifejlesztésével próbál meg úrrá lenni a lehangoló semmittevésen. A kollégáival folytatott töretlen harcát figyelve apránként megismerhetjük személyiségét is, és kiderül, hogy nem egy átlagos szakiról van szó, hanem egy igazi zseniről.
Ne felejtsük el azonban, hogy a történet a 90-es évek elején született.
A háttértárak árának mélyrepülése már egy ideje elért arra a pontra, ahol adott esetben a szalagos egységnél kifizetődőbb megoldás a hálózati mentés. Jelen cikk ennek egy megvalósítását ismerteti. Természetesen ez nem azt jelenti, hogy a mai kor pokoli operátorainak lelke végleg megnyugodhat, mindenesetre a */dev/null* használatára úgy tűnik, egy ideig nem lesz szükség.
Tegyünk félre egy pillanat erejéig minden bennünk dübörgő vallási érzést, és képzeljünk el egy heterogén hálózatot, amelyben egy *Windows NT-* és egy *Linux*-alapú belső hálózati kiszolgáló békésen megfér egymás mellett. Mi több, feladatunk a két számítógép házasítása úgy, hogy bizonyos érzékeny adatokat tartalmazó könyvtárakról az éj leple alatt mentés készüljön. Az érzékeny adatokat az *NT*-n egy *Sybase* adatbázisszerver által felügyelt fájlokat jelentik, a *Linux* pedig egy *FTP* kiszolgálóval áll rendelkezésünkre. A frigyhez *Perlt* fogunk alkalmazni az *NT*-n, az alábbi hat felvonásban:

```
stop_service;
sleep 60;
create_archive;
start_service;
ftp_upload;
unlink $zip_name;
```

A szolgáltatás leállítása

Noha szkriptünk éjszaka fog futni, amikor feltehetőleg minden felhasználó alszik, és így nem jelentenek veszélyt, az adatbázisszervert jobb leállítani a mentés megkezdése előtt. Ehhez a `Win32::Service` modult használjuk. Íme:

```
sub stop_service {
    print "Stopping service.\n";
    my %status;
Win32::Service::GetStatus ('',
➥$service_key, \%status);
    {
        last if ($status
➥{CurrentState} == 1);
        die "Cannot stop service
➥'".$service_key."'.\n" unless

Win32::Service::StopService ('',
➥$service_key);
    }
    print "Service stopped.\n";
}
```

A kód a `$service_key` globális változó meglétét feltételezi, amely a *Windows* szolgáltatás egyértelmű azonosítója. Ha ezt tudjuk, egyszerűen vegyük fel a szkript legelejére. Ha nem, akkor szintén a fenti modulra támaszkodva kinyerhetjük a szolgáltatás leírásából, amire később láthatunk is példát.

Az eljárás ellenőrzi, hogy fut-e a szolgáltatás, és ha igen, leállítja azt. Először a `Win32::Service` modul `GetStatus` metódusát használjuk. Ez három paramétert vár. Az első karakterlánc annak a számítógépnek a *NetBIOS* neve, amelyen a kérdéses szolgáltatás fut. Ha ez üres, akkor vizsgálódásunk tárgya a helyi számítógép. A második a szolgáltatás azonosítója, a harmadik asszociatív tömb pedig a státuszinformációt tároló változó referenciája, vagyis az eredmény.
A `GetStatus` hívás után egy blokkba lépünk, amit az első sorral azonnal el is hagyunk, ha a szolgáltatás nem fut (ezt jelzi az 1-es státusz). Ha a blokkot nem hagytuk el, akkor a `StopService` metódussal leállítjuk a szolgáltatást.

Egy kis pihenés

A szolgáltatás leállítása a legtöbb esetben minden mellékhatás nélkül megtörténik, viszont a metódus visszatérése után is időbe telhet egyes takarító folyamatok lefutása. A `sleep 60` segítségével 60 másodperc, azaz egy perc erejéig elaltatjuk a szkriptet. Ezalatt az adatbázis-kezelő biztosan leáll.
Természetesen egy jól irányzott ciklussal felkészülhetnénk arra az esetre is, ha ez nem történne meg, most viszont az egyszerűségre, és az ebből adódó kisebb hibalehetőségre hivatkozva rábízzuk magunkat az *NT* szolgáltatás-kezelőjére.
Vegyük észre, hogy a `StopService` meghiúsulása a szkript azonnali leállítását eredményezi, amiről, napi

mentésről lévén szó, a rendszergazda legkésőbb másnap értesül. Mindig az adott feladat határozza meg, hogy mennyire szigorúan követeljük meg a mentés sikerét. Ne feledjük, hogy a biztos mentés záloga egy időtúllépés esetén kikényszerített leállás lenne, ez viszont lehet, hogy több kárt okozna, mint hasznot.

Zip, zip, hurrá

Miután feltehetjük, hogy a szolgáltatás nem fut, az adatbázis-kezelő fájljait egyszerűen, az `Archive::Zip` modul segítségével tömörítjük.

```
sub create_archive {
    print "Creating archive.\n";
    my $zip_archive =
➥Archive::Zip -> new;
    foreach my $dir
➥(@directories) {
     die "'".$dir."' is not a
➥directory.\n" unless (-d $dir);
      print "Adding directory
➥tree '".$dir."' to
➥archive.\n";
   my $status = $zip_archive ->
➥addTree ($dir, basename($dir));
  die "Cannot add directory tree
➥'".$dir."' to archive.\n"
      unless ($status == AZ_OK);
    }
    print "Adding timestamp to
➥archive.\n";
    $zip_archive -> addString
➥(scalar localtime,
➥"timestamp.txt");
    print "Writing to file.\n";
    my $status = $zip_archive ->
➥writeToFileNamed ($zip_name);
    die "Cannot write to file '"
➥.$zip_name."'.\n"
    unless ($status == AZ_OK);
    print "Created archive.\n";
}
```

Ez a szubrutin elsőként létrehoz egy `Archive::Zip` objektumot, ami egy üres archívumot jelent. Ezután egyenként végiglépked a `@directories` globális tömb elemein, és minden elemről eldönti, hogy könyvtár-e. Ha nem, a szkript meghiúsul. Normális esetben rekurzív módon az elem teljes részfáját hozzáadja az archívumhoz. Természetesen ennél a lépésnél is történik hibaellenőrzés. Ebben a ciklusban a hangsúly az `addTree` metóduson van. Ennek első paramétere a könyvtár elérési útja, a második az a név, amelyen tárolni szeretnénk. Ennél segítségül hívjuk a `File::Basename` modul használatával elérhető `basename` függvényt, ami az azonos nevű ***UNIX*** paranccsal azonos viselkedésű.
A ciklus végeztével minden könyvtárat hozzáadtunk az archívumhoz. A teljesség kedvéért még egy sokszor meglepően hasznosnak bizonyuló metódussal időbélyeget helyezünk el az archívumban. Az `addString` metódus az átadott karakterláncot a hivatkozott fájlban helyezi el. Ehhez egyetlen függvényhívásra van szükségünk, és még csak feleslleges átmeneti állományokat sem igényel a művelet. Végezetül a `writeToFileNamed` metódussal lemezre írjuk a mentést. Az itt használt `$zip_name` globális változó hatékony meghatározására később visszatérünk. Fontos, hogy az eddig összegyűjtött könyvtárak tartalma valójában csak most kerül feldolgozásra, tehát nem kell tartanunk attól, hogy néhány óriási fájl felemésztené a memóriát.

A szolgáltatás elindítása

Mivel a tömörített mentést már csak fel kell töltenünk, ami várhatóan időigényes feladat, a szolgáltatást nyugodt szívvel elindíthatjuk.

```
sub start_service {
    print "Starting service.\n";
    my %status; Win32::Service:
➥:GetStatus ('', $service_key,
➥\%status);
    {
        last if ($status
➥{CurrentState} == 4);
       die "Cannot stop service
➥'".$service_key."'.\n" unless
            Win32:
➥:Service::StartService ('',
➥$service_key);
    }
    print "Service started.\n";
}
```

Ez az eljárás nem rejteget sok újdonságot a legelsőhöz képest. Az egyetlen nem magától értetődő pont az, ahol a szolgáltatás futásának ellenőrzése történik. A 4-es státusz jelenti azt, hogy a szolgáltatás már fut, ekkor nem indítjuk el.

Ez például akkor történhet meg, ha az archívum készítése alatt valaki elindította az adatbázis-kezelőt. Nagy valószínűséggel a szkript ez esetben már kilépett volna fájlzárolási problémák miatt, de azért nem árt az óvatosság.

FTP feltöltés

A szkript talán egyik legszebb része következik, az ***FTP***-n történő feltöltés. Ehhez a `Net::FTP` modult használjuk.

```
sub ftp_upload {
    print "Uploading to ftp
➥site.\n";
    die "Cannot connect to '"
➥.$ftp_site."': ".$@."\n"
➥unless
        my $ftp_session =
➥Net::FTP -> new ($ftp_site,
➥Debug => 0);
    die "Cannot login:
➥".($ftp_session -> message)
➥unless
        $ftp_session -> login
➥($ftp_user, $ftp_pass);
    die "Cannot set binary
➥mode: ".($ftp_session ->
➥message) unless
        $ftp_session -> binary;
    die "Cannot cwd to '"
➥.$ftp_dir."': ".($ftp_session
➥-> message) unless
        $ftp_session -> cwd
➥($ftp_dir);
    die "Cannot put file
➥'".$zip_name."':
➥".($ftp_session -> message)
➥unless
        $ftp_session -> put
➥($zip_name);
    $ftp_session -> quit;
    print "Uploading done.\n";
}
```

Ha valaki használt már parancssoros ***FTP***-klienst, a fenti kód sorról sorra történő magyarázata inkább fárasztó lenne, mint hasznos. Egyetlen lényeges, elsőre mégsem feltétlenül szembeötlő lépés a kapcsolat binárisra állítása. Mivel egy ***zip*** állománynak nem tesz jót a sorvége jel átalakítása, ez a sor elengedhetetlen.
A szubrutin az `$ftp_site`, `$ftp_user`, `$ftp_pass`, `$ftp_dir` globális változók meglétét feltételezi.

Takarítás

Ez az a része a mentési folyamatnak, ami általában annyira sok nyűggel jár, hogy sokan inkább a */dev/null*-t választják. Szerencsére ez jelen esetben egyetlen sor:

```
unlink $zip_name;
```

A nagy kép

Mindez úgy áll össze egy szkriptté, hogy a mindenkinek kötelező `use strict` után felsoroljuk a használt modulokat, kitöltjük a globális változókat, beszúrjuk a fenti rutinokat, végül pedig leírjuk az elején említett forgatókönyvet.
Lustaságunk mihamarabbi elhatalmasodása érdekében lássuk azt a két globális változót, melyet nem egyszerű értékadással határozunk meg:

```
my $service_key;
{
    my %services; Win32:
➥:Service::GetServices ('',
➥\%services);
    foreach my $key (sort keys
➥%services) {
        if ($key =~
➥$service_name) {
            $service_key =
➥$services{$key}; last;
        }
    }
}
my $zip_name = $zip_dir."/
➥backup-".(strftime "%Y%m%d",
➥localtime).".zip";
```

Szó volt róla, hogy a szolgáltatás leállítása és elindítása csak a szolgáltatás egyértelmű azonosítója ismeretében lehetséges. Ezt megállapíthatjuk a szolgáltatáslista kézi böngészésével, vagy ha ennél rugalmasabb megoldásra vágyunk, ki is találhatjuk. A szolgáltatás leírására történő mintaillesztéssel Csak nagyjából kell, hogy emlékezzünk a szolgáltatás nevére, ami nagyságrendekkel növeli a félálomban, mégis hasznos munkával töltött órák számát. Erre példa a `$service_key` beállítása.
A `$zip_name` egy *backup-20060228.zip* alakhoz hasonló nevet határoz meg. Ahhoz, hogy a dátum valóban a jelenlegi dátum legyen, igénybe vesszük a *POSIX* modul `strftime` függvényét. Ez a parancssori `date`-hez hasonló formátumot vár el, amivel könnyen elérhetjük a kívánt fájlnevet.

Kész?

A helyes válasz az, hogy attól függ. Ha valóban elviselhető, hogy szélsőséges körülmények között egy esti mentés kimarad a nem egy helyen elképzelhető kilépés miatt, akkor mindenképpen. Megjegyzem, erre az egy hónapos teszt üzem alatt nem volt példa. Ami ennél fontosabb, az a *Linux* kiszolgálón keletkezett mentések kezelése. Feltétlenül meg kell oldani, hogy ezek ne egyék fel a merevlemezt. Erre egy pár soros, `crontab`-ból futtatott szkript megfelel. A *.zip* fájlok elnevezése kiválóan alkalmas annak eldöntésére, hogy a mentés törölhető-e vagy sem, ezért ezzel már nem untatnék senkit.

Fülöp Balázs
(fulop.balazs@initon.hu)

Az Initon Kft. informatikai munkatársa.