Adatmentési (backup) megoldások Linux alatt

Szabad szoftver keretrendszer

Készítette a Közigazgatási és Igazságügyi minisztérium E-közigazgatási
Szabad Szoftver Kompetencia Központja
Budapest, 2013

Kódszám: EKOP–1.2.15

A dokumentum legfrissebb változata letölthető a honlapunkról:
http://szabadszoftver.kormany.hu/szabad-szoftver-keretrendszer/

Tartalomjegyzék

Adatmentési (backup) megoldások Linux alatt

Az talán senki számára nem vitatható tény, hogy adatmentésnek, azaz biztonsági mentésnek minden rendszer alatt lennie kell. Az is fontos, hogy a mentés automatikusan, beavatkozásmentesen történjen, ugyanakkor szintén fontos, hogy a mentések rendszeresen ellenőrizve is legyenek, hiszen ha nyugodtan bízunk egy mentést végző 5-6 éves héjprogramban (shell szkript), amit sosem ellenőriztünk le, akkor kellemetlen meglepetések érhetnek minket, amikor elő kell venni a régi, megbízhatónak gondolt mentést, és kiderül, hogy egy adatbázist vagy egy könyvtárat mégsem követtünk le. A Linux-alapú rendszerekre számos kiváló mentési keretrendszer és héjprogram érhető el. Gyakorlatilag a legtöbb a unixos alapokkal rendelkező cp, tar[1], cpio[2] programokat használja, amelyek minden Linux rendszer standard tartozékai. Mások az rsync[3]-et és az SSH-t kombinálják, ezzel teszik lehetővé a biztonságos távoli menthetőséget is.

A főbb mentési módszerek

– *Teljes mentés:* ilyenkor a rendszer minden adata mentésre kerül, válogatás nélkül. Előnyre, hogy például egy lemezkép készíthető a szerver aktuális állapotáról, amely egyszerűen visszaállítható (gyakran csak a rendszertöltőt (boot loader) vagy még azt sem kell helyrerakni). Hátránya: sérülékeny és időigényes.

[1] http://www.gnu.org/software/tar/
[2] http://www.gnu.org/software/cpio/
[3] http://en.wikipedia.org/wiki/Rsync

MAGYARORSZÁG MEGÚJUL
A projekt az Európai Unió támogatásával, az Európai Regionális Fejlesztési Alap társfinanszírozásával valósul meg.

– *Inkrementális mentés:* Használata során csak a változások mentődnek, azaz leképezünk egy nullás mentést és a megváltozott állapotokat mentjük. Ennek a mentésnek 2 alfajtája van: a kumulatív és a differenciális mentés.
– *Kumulatív mentés:* alkalmazása esetén mindig az utolsó teljes mentés óta megváltozott adatok kerülnek mentésre. Adatváltozás esetén csak a változást menti, viszont azt minden esetben. Előnye, hogy jóval gyorsabb a teljes mentésnél, és kevesebb a területigénye is.
– *Differenciális mentés:* ebben az esetben csak az utolsó inkrementális mentés óta megváltozott adatokat mentjük. Előnye, hogy a leggyorsabb stratégia. Hátránya, hogy az első mentés és az összes követő mentés szükséges hozzá, azaz a leginkább sérülékeny az adatstruktúrát illetően.

A mentési eljárások két irányból hajthatók végre. A szerver végezhet adat „tolást” a mentő egység (szerver, NAS, szalag) felé, illetve a mentő egység/szerver is kezdeményezheti a mentési folyamatot, ilyenkor maga fele húzza az adatokat. A két módszer között főként biztonsági különbség van. Ha a backup szerver húzza maga felé az adatot, akkor azt a szervert kell úgy kialakítani, hogy biztonságos legyen. Ilyen esetben az éles szerverre való betörés esetén a backup szerver kevésbé sérülékeny, mint fordítva.

A legfontosabb teendők a mentés kialakításában

– *Felmérés:* összeírni a mentendő szoftverkörnyezet kialakítását. Eldönteni, hogy az adott mentő szoftver alkalmas-e valós időben menteni az adatbázisunk (szükséges-e ez egyáltalán), illetve pontosan mit és mikor célszerű mentenünk. Például: érdemes-e minden nap menteni a teljes fájlrendszert, mindenféle ideiglenes állománnyal együtt, vagy elég csak aktualizálni. Fontos gondolni a Disaster recovery-re[4] (katasztrófa-helyreállítás) is, amikor akár az egész gépterem elpusztulhat. Ismert ilyen példa, amikor a WTC[5] egyik épületében volt az éles alkalmazás szerver és a 2. toronyban pedig a backup. Természetesen ez nem minden nap előforduló lehetőség, de számításba kell venni a felmérés folyamatában. Fel kell tenni magunknak a kérdést, hogy az adataim hiánya, mekkora kárt okoz, illetve mekkora összegbe kerül egy RAID tömb visszaállítása az erre szakosodott szervizben.
– *Tervezés*: a felmérésre alapozva vázlatosan leírjuk a támasztott igényeinket. Betervezzük, milyen gyakran, milyen idősávban lesz lehetőség futtatni a mentést. Szétválasztjuk a távoli, a szalagos és a helyi mentéseket. Ütemezzük azokat, hogy ne fussanak egymásra. A mentési házirend figyelembe vétele, ha már van ilyen, ha nincs akkor írjunk.
– *Kiválasztás*: Az 1-2 pontok alapján kiválasztjuk a megfelelő szoftvert, amely a leírása alapján alkalmas lesz a munkára.
– *Megvalósítás*: A kiválasztott szoftverek segítségével felépítjük a mentési rendszert.
– *Tesztfuttatás:* Leteszteljük a már elkészült mentést. Megnézzük a lehetséges eseteket és azok befolyását a mentésünkre. Elég szalag áll-e rendelkezésre, működik-e az értesítési rendszer, amely tudatja velünk, ha elfogyott a szalag, vagy nincs elég tárterület. Egy híres mentési baki, amikor az egyik ismert banki kártyarendszer teljes hálózata azért állt meg karácsony előtt, mivel az automata leejtette a cserélendő szalagot és a mentés nagyobb prioritást élvezett, mint maga az éles üzem. Gyakran hiába állítjuk össze a legjobb mentő eszközt, és állítjuk be jól, ha egy hibásan beállított e-mail cím, vagy egy blokkolt SMTP szolgáltatás lehetetlenné teszi, hogy az értesítés el-

[4] http://en.wikipedia.org/wiki/Disaster_recovery
[5] http://hu.wikipedia.org/wiki/Vil%C3%A1gkereskedelmi_K%C3%B6zpont

jusson hozzánk, akkor hiába volt minden. Figyeljünk arra, hogy a mentés lehetőleg olyankor fusson amikor szükséges, de a mentés ne zavarja az éles üzemeltetést.

– *Ellenőrzés*: A mentés elkészültével fontos, hogy egy próba rendszeren végezzünk egy teljes visszaállítást, és nézzük meg, 100%-ban visszaállítható-e minden csak a backupra támaszkodva.
– *Ellenőrzés* ütemezése: legyen a havi/negyedéves/féléves/éves rutin része, hogy teljes rendszer-visszaállítást végzünk, csakúgy mint ahogy akár naponta-hetente megnézzük, hogy a mentés rendben ment-e. Érdemes a mentés által készített összegző naplót monitorozni.

Szoftverek a mentés megvalósítására

DD[6]

Az egyik legrégebbi Unix eszköz például a teljes lemezduplikációra. Felhasználási területe szélesebb a mentéseknél, azonban egy egyszerű klónozás esetében megoldást jelenthet, fenntartásokkal.

Tar, CP, CPIO

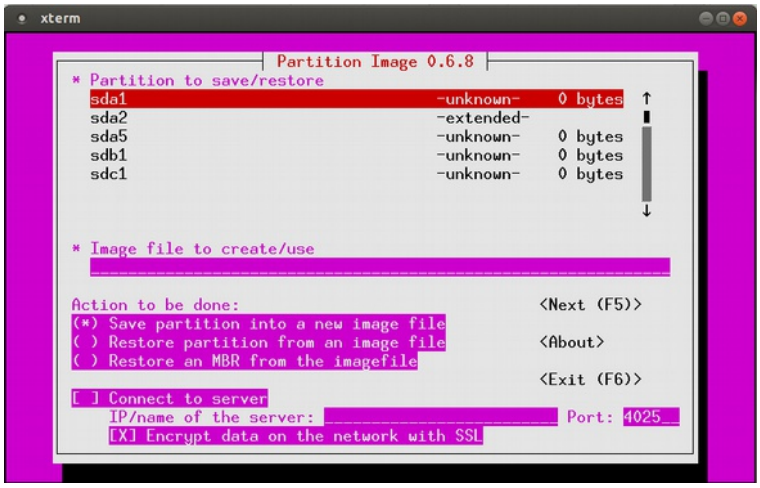
Az operációs rendszerek alapvető kelléktára. Héjprogramok segítségével használható mentésre, akár kazettás egység vezérlésére is. Önmagukban nagyon hasznosak, de egy modern rendszerben a rájuk épülő programokat használjuk inkább.

Partimage[7]

Egy konzol alapú lemezklónozó eszköz. A dd-hez nagyon hasonló, azonban képességei alapján nem mondható egy sima dd-nek. Kezelése egyszerű, egy párbeszédablakos felületen lehet kiválasztani a mentendő területet és azt, hogy hova mentsen. Tud titkosítani és darabolni is, valamint hálózati mentést csinálni. Hátránya, hogy csak offline rendszert lehet vele menteni, célszerűen egy live CD-vel. Szerver lemezének költöztetése esetén, vagy hardvercsere esetében jó megoldás lehet, illetve munkaállomások klónozásra is. Rendelkezik állapotfigyelővel, és számos lehetőséget nyújt a tömörítésre, darabolásra is. Ugyanakkor alap szintű lemez duplázásra remek megoldás, azaz ha elkészítettünk egy szervert (például egy tűzfalat, amelyen jó esély van rá, hogy hosszú távon főként az iptables konfigurációs állományának a változását kell lekövetni, vagy például egy telefonközpont – Asterisk) és mindenképpen szükséges egy hideg backup belőle, akkor egy tökéletes eszköz a partimage. Akár egy Ubuntu Live pendrive vagy CD-vel beindítjuk a gépet, úgy hogy a másolandó (forrás) és a cél merevlemez is benne van a gépben, majd a Partimage párbeszédablakában kiválasztjuk, mit szeretnénk csinálni, és készen is vagyunk. Futó (éles rendszert) értelemszerűen így nem tudunk menteni, de akár arra is használható, ha kliens parkot kell létrehoznunk, azaz adott 200 db ugyanolyan PC, és mindegyikre fel kell telepíteni ugyanazt a terjesztést. Telepítése rendkívül egyszerű:

```
apt-get install partimage
```

[6] http://en.wikipedia.org/wiki/Dd_(Unix)
[7] http://www.partimage.org/Main_Page



Rsync[8]

Az rsync ideális eszköz gépen belüli szinkronizációra, természetesen lehetőség van távoli elérésre is, akár ssh-n, akár rsyncd-n keresztül. A legtöbb backup megoldás alapeszközként használja, mások kiegészítőként, illetve pár megoldás kifejezetten az rsyncre alapozva igazából csak egy kezelőfelület. Paraméterezhetősége nagyon széles körű, talán pont ezért készült annyi eszköz, amely leegyszerűsíti a vezérlését (például: grsync). Előnye a többi hozzá képest sokkal egyszerűbb (cp, tar stb.) programmal szemben, hogy a számtalan lehetőség miatt rugalmasan kezeli például, ha a forrás nagyobb mint a cél lemez területe, akkor nem írja tele a célt stb. Általában héjprogram formájában vagy beépített eszközként használjuk. Az rsync segítségével nem csak gyorsabban tudunk menteni, hanem a speciális delta kódolásnak köszönhetően csak a változásokat fogja átvinni egyik helyről a másikra. Ennek különösen szűk hálózati keresztmetszetek között van nagy jelentősége, vagy pedig például, ha egy 1 GB-os lemezképben csak 2 MB tartalom változik meg, akkor az rsync delta technológia segítségével nem a teljes 1 GB-ot fogja lementeni, hanem egy összehasonlító vizsgálat után csak a változást – azaz jelen esetben a 2 MB-ot – fogja aktualizálni. A legtöbb terjesztés tartalmazza alap szerver kiépítésben, de telepíthetjük az

```
apt-get install rsync
```

parancs kiadásával. A legtipikusabban használt paraméterezése a következő:

```
rsync -av /forrás/ /cél/
```

ahol a -a az archív készítése, a -v pedig a részletes nézet. Ezt aztán kiegészíthetjük számtalan hasznos paraméterrel is, pl:

```
rsync -r -t -p -o -g -v --progress --delete -l -H /cél/ /forrás/
```

Az Rsync lokális használta esetén előfordulhat, hogy például egy igen gyors SSD lemezről másolunk egy USB1 vagy USB2 szabványon csatlakozó akár Flash memóriára, vagy merevlemezre. Ez persze főként otthoni használatban fordulhat elő, de szerveren belül is megeshet, hogy a különböző lemez alrendszerek más-más sebességgel működnek. Ilyen esetben és főként a tömegesen előforduló kis állományok szinkronizálása esetén előfordulhat, hogy feltorlódnak az I/O műveltek, és ezért emelkedni kezd a rendszerterhelés (load), fogyni kezd a memória, szélsőséges esetben már tárcserét (swap) kezd a rendszer. Ez pedig biztos út az összeomlás felé. Az ilyen összeomlások elkerülésére két eszköz használata javasolható, az egyik egy monitorozó eszköz, amely a top programhoz

[8] http://rsync.samba.org/

hasonlóan képes megjeleníteni és rendezni azokat a futó folyamatokat, amelyek az I/O terhelést okozzák. A program neve iotop,[9] telepítése és használata is igen egyszerű:

```
apt-get install iotop
```

és akár csak a top program esetén, csak indítsuk el. A másik segédeszköz, amely hasznos lehet egy esetlegesen rsync mentés kivitelezéséhez, az ionice[10]. Telepítése:

```
apt-get install ionice
```

Az ionice és iotop programok részletes ismertetésével a tuningolás fejezet foglalkozik.

Rsnapshot[11]

Szintén rsync és SSH alapokon nyugvó mentési eszköz. Jóval fejlettebb lehetőségeket tartogat, mint az eddigi eszközök. Alkalmas teljes fájlrendszer mentésére megszorításokkal (boot loadert nem tud menteni, illetve futási időben például adatbázist sem). Egyszerű paraméterezhetőség és könnyen átlátható konfiguráció jellemzi. Rugalmas eszköz, mivel képes lokális mentést végezni (alapesetben a differenciális mentést támogatja), az indulása cronból ütemezhető, vagy szkriptelhető. Létrehoz egy zároló állományt, amely jelzi az esetlegesen ráfutó következő rsnapshotnak, hogy még az előző mentés fut, ezzel elkerülve az esetleges feltorlódást és ütközést. Együtt tud működni az SSH-val is, így lehetőség van arra, hogy egy távoli backup szerver (amelynek csak ez a feladata és kellően biztosított) kezdeményezze a mentést SSH-val. De természetesen megoldható valamilyen hálózati fájlrendszeren keresztüli mentés, illetve külső illesztésű merevlemezre is. Nagyon hasznos funkciója, hogy a konfigurációs állomány az Apache-hoz hasonló szintaxis-ellenőrzéssel rendelkezik, vagyis az rsnapshot configtest parancs segítségével tesztelhetjük az esetlegesen a szintaxissal ütköző parancsokat, elütéseket. Így nem az éles mentés lefutása közben fog kiderülni a probléma. Külön naplóállományt készít, amelyben jól követhető a tevékenysége. A mentést elkülönítetten egy erre meghatározott területen kezeli, ahol a megadott intervallumtól függően készíti el a könyvtárstruktúrát. Az első alap mentés elkészítése jóval lassabb, a további inkrementális mentések már gyorsabbak, melyeket linkek segítségével köt össze. Az aktuális állapotok pedig hourly, daily, weekly, monthly címkével kerülnek tárolásra. Természetesen egy konfigurációs állományban felsorolhatjuk az óránkénti, napi, heti, havi mentések struktúráját is. Kivételkezelése könnyen paraméterezhető (az rsync használatának köszönhetően). Külön csak erre a feladatra fenntartott mentő szerver esetén (megfelelően védett backup szerver) jól automatizálható a mentés teljes folyamata az SSH kulcs alapú megoldásával. Ebben az esetben a backup szerver kezdeményezi a kapcsolatot SSH-n keresztül. Annak érdekében, hogy mindent le tudjon menteni, root hozzáférés szükséges (a legtöbb esetben), amely távoli elérésre normál esetben nem lenne biztonságos. Ezért az SSH szerver következő konfigurációváltoztatása javasolt: használjuk a

```
PermitRootLogin=forced-commands-only
```

opciót a root távoli hozzáféréséhez. Majd készítsük el a root SSH-kulcspárját jelszó nélkül, hogy a héjprogram ne kérjen majd jelszót. Válasszuk megfelelően nagy (4096 bájt vagy e feletti) kulcs készítését. Az éles mentendő szerveren a /root/.ssh/authorized_keys állományba másoljuk be a publikus kulcsot a következő kiegészítéssel:

```
from="172.20.1.10",command="/etc/validate-rsync"
```

ezt közvetlenül a legelső karakter helyére illesszük be, utána kezdődjön a kulcs. Az IP-cím helyén a backup szerver IP-címe legyen. Az /etc/validate-rsync héjprogramba pedig rakjuk be azokat a korlátozásokat, amelyek segítségével minimalizálni tudjuk a root SSH használatát. Pontos lista is elérhe-

[9] http://guichaz.free.fr/iotop/
[10] http://manpages.ubuntu.com/manpages/precise/man1/ionice.1.html
[11] http://www.rsnapshot.org/

tő az egész mentési módszer részletes leírásával együtt.[12] Mint látható, az rsnapshot egy komplex héjprogram-keretrendszer, amely egyszerűvé teszi a strukturált mentést. Azonban teljes visszaállítás csak az alap operációs rendszer újratelepítése után lehetséges egy esetleges teljes adatvesztés után. Ideális eszköz tehát nem HA rendszerek mentéséhez. Valamint gondoskodni kell az adatbázis külön mentéséről, hiszen futási időben erre az rsnapshot konzisztensen nem képes.

Esetünkben az /etc/validate-rsync állomány tartalma a következő volt:

```
#!/bin/sh
case "$SSH_ORIGINAL_COMMAND" in
*\&*)
echo "Rejected"
;;
*\(*)
echo "Rejected"
;;
*\{*)
echo "Rejected"
;;
*\;*)
echo "Rejected"
;;
*\<*)
echo "Rejected"
;;
*\`*)
echo "Rejected"
;;
*\|*)
echo "Rejected"
;;
rsync\ --server*)
$SSH_ORIGINAL_COMMAND
;;
*)
echo "Rejected"
;;
esac
```

Amanda Backup[13]

Kliens-szerver felépítésű mentőeszköz, képes Windows rendszereket is menteni. Windows esetén a Cygwin csomag, vagy Samba megosztás vagy pedig saját kliens segítségével végezhető el a mentés. Teljesen szerver alapú működés, előre definiálható adattárolók, amelyek lehetnek lemezről lemezre, lemezről kazettára, illetve lemezről felhő alapú tárterületre való mentések is. Eredetileg kezdetben kazettás mentések kialakítására tervezték, ezért amikor merevlemez alapú mentést definiálunk, akkor virtuális szalagokkal operálva a merevlemezen hoz létre alkönyvtárakat. A virtuális szalagok definiált méretét és darabszámát nem lépi át így sem. Igazi előnye, hogy nem kell statikus ter-

[12] http://troy.jdmz.net/rsync/#ref2
[13] http://www.amanda.org/

vet készíteni, meg kell neki adni, hogy hány darab teljes mentést illetve mennyi inkrementálist fogunk készíteni. Rögzítjük a felhasználható szalagok számát. Az Amanda mentés elején kiszámolja, hogy a mentendő könyvtárak vagy inkrementális mentéseihez aktuálisan mennyi helyre van szükség és önállóan eldönti, hogy az adott mentésbe milyen könyvtár milyen szinten kerüljön bele. Ha rendelkezésre áll hely, akkor előre hozza a teljes mentéseket, ha pedig kifut a helyből, akkor késlelteti és a naplóba figyelmeztetéseket ír. Az Amanda telepítésének fontos része az openbsd-inetd csomag is. Mivel titkosítást alapesetben nem használ a távoli gépre való mentés esetén, ezért érdemes VPN-t vagy a külön konfigurálható amcrypt-et[14] használni a nyílt hálózatokon. Belső gépek mentésére titkosítás nélkül is jól használható. Mivel az Amanda több TCP és UDP portot használ és nyit, ezért érdemes a tűzfalunkat felkészíteni a fogadására mind a két gépen, amennyiben a kialakításunk ezt szükségessé teszi.[15]

Bacula[16]

Szintén keresztplatformos biztonsági mentést lehet megvalósítani vele. Létezik hozzá Linux, Windows és OS X alá kliens program, amelyeknek konzol alapú, GTK+ felületű és wxWidgets grafikus variánsai léteznek. Legfőbb erőssége, hogy elosztott működésre képes, azaz különböző részekből áll, amelyeket külön gépekre telepíthetünk. Fő komponensei:

– Director daemon (director): végzi a rendszer irányítását, kapcsolatot tart a többi démonnal, kommunikál a különböző grafikus felületekkel, időzíti a mentéseket, aktualizálja a katalógust.
– Storage daemon (sd): közvetlenül a mentő egységekkel (szalagos meghajtó, HDD, DVD) tartja a kapcsolatot, kezeli a director olvasási/írási kéréseit, illetve fogadja az FD-től (lásd alább) érkező mentett adatokat.
– File Daemon (FD): feladata a mentendő szerverek és asztali gépek mentendő adatainak begyűjtése, tömörítése, titkosítása, majd az adatok továbbítása a kijelölt SD-nek. Valamint a visszaállításnál mindezt visszafelé irányban is ellátja.

A Bacula egy központi katalógusban tárolja a mentett állományokat és a fájlt tartalmazó kötet nevét, a mentés időpontját, valamint egyéb adatokat is, segítve és gyorsítva ezzel az esetleges visszaállítást. Nagy előnye, hogy szinte mindenre kiterjedően konfigurálható, így megoldható az is, hogy a mentés indulása egy tetszőleges egyedi héjprogram futtatásával kezdődjön, például egy adatbázismentést (MariaDB, MySQL, PostgreSQL dump) kezdeményezve. Egy igazi komplex megoldás, amely a titkosítása és az elosztott működése miatt a nagyvállalati felhasználásra is alkalmassá teszi[17].

Dirvish[18]

Szintén szerver alapú mentőeszköz, amely támaszkodik az rsync és a Perl képességeire. Tudása nagyban hasonló az Amanda, vagy a Bacula lehetőségeihez, konfigurálása népszerűsége és a fellelhető dokumentáció miatt talán könnyebb[19]. Ugyanúgy kezeli a legtöbb nagyvállalati szalagos eszközt és lehetőség van egyéb például merevlemez mentésére is. Az adatbázis mentéseket egy mentés indítását megelőző Perl vagy héjprogrammal lehet megoldani akár Oracle, PostgreSQL, MySQL vagy egyéb adatbázis-kezelő esetén is. A kliens-szerver alapú mentés természetesen futhat SSH

[14] http://wiki.zmanda.com/index.php/How_To:Set_up_data_encryption
[15] http://wiki.zmanda.com/index.php/How_To:Set_Up_iptables_for_Amanda
[16] http://www.bacula.org/en/
[17] http://sugo.ubuntu.hu/10.10/html/serverguide/hu/bacula.html.
[18] http://www.dirvish.org/
[19] http://www.googlux.com/dirvishconfig.html

megoldással[20], így biztosítva a távoli gépek között az adatbiztonságot mentés esetén. Telepítését a megszokott módon az apt-get segítségével kezdjük meg:

```
apt-get install dirvish ssh
```

A telepítés menete a Pull vagy Push (utolsó bekezdések) alapján általunk követendőnek tekintett felépítést elemzi, azaz a központilag védett és elkülönített backup szerver tud csatlakozni az összes klienshez, de a kliensek nem tudnak csatlakozni közvetlenül hozzá.

Az alap konfigurációs állomány mintákat a /usr/share/doc/dirvish/examples könyvtárban találjuk. Célszerűen innen elsőnek a master.conf-ot másoljuk át a /etc/dirvish/master.conf helyre. Akkor valami ehhez hasonlót fogunk látni:

```
bank:
/backup
exclude:
/etc/mtab
        /var/lib/nfs/*tab
        /var/cache/apt/archives/*.deb
        .cache/*
        .firefox/default/*/Cache/*
        /usr/src/**/*.o
        lost+found/
Runall:
eles1.szerver   21:00
eles2.szerver   23:00

expire-default: +15 days
expire-rule:
#      MIN HR    DOM MON       DOW  STRFTIME_FMT
*  *    *  *       1    +3 months
#      *  *    1-7 *        1    +1 year
#      *  *    1-7 1,4,7,10  1
*  10-20 *  *       *    +4 days
#      *  *    *  *       2-7  +15 days
```

Nézzük meg pontosan, mi mire való:

A master.conf-ban azokat a változókat fogjuk beállítani, amelyek az összes mentendő gépre vonatkozni fognak, valamint itt definiáljuk azokat a gépeket is, amelyeket menteni akarunk. Értelemszerűen vagy hozzuk létre a /backup könyvtárat 700-as jogokkal root:root felhasználóként, vagy írjuk át arra a könyvtárra, ahova menteni akarunk. A bank változóval határozhatjuk meg, hogy a program hol tárolja a tényleges mentéseket.

Az exclude paraméterrel azokat a könyvtárakat határozhatjuk meg, amelyeket nem akarunk menteni, azaz ki akarjuk zárni őket a teljes mentésből. Mint az látható, megadhatjuk abszolút és relatív módon is. Így vonatkozhat az összes .firefox/ alatt lévő Cache könyvtárra is, de explicit kizárhatjuk a /lost+found/-ot is. Ezek a paraméterek központilag fognak vonatkozni az összes később definiált hostra, amelyet menteni fogunk, így ott már nem kell külön ezeket felsorolni.

A Runall paraméterrel azt mondjuk meg a programnak, hogy az egyes gépek mentését mikor futtassa, ide tudjuk felsorolásszerűen befűzni a gépeinket, amelyeket a bank könyvtár alá létre is kell hozni. Azaz, létre kell hozni a /backup/eles1.szerver és a /backup/eles2.szerver könyvtárakat is.

[20] http://apt-get.dk/howto/backup/

Az expire-default és expire-rule paraméterekkel az elévülési időt határozhatjuk meg, azaz az egyes mentéseket meddig tartsa meg, és mikortól kezdje felülírni.

Hogy tovább tudjunk lépni, a /backup/eles1.szerver alá létre kell hoznunk egy dirvish könyvtárat és abban elhelyezni egy default.conf fájlt. Az elérési útja így nézzen ki: /backup/eles1.szerver/dirvish/default.conf
a tartalma pedig a példa kedvéért legyen a következő:

```
client: eles1.szerver.kft
tree: /
index: gzip
image-default: %Y-%m-%d
xdev: 1
exclude:
/var/spool/squid/
```

Értelemszerűen a client paraméternél megadjuk a mentendő éles szerver hostnevét vagy IP-címét, ahova a dirvish root felhasználóval fog csatlakozni. A tree paraméter pedig megmondja, hogy honnan indítjuk a mentés. Jelen pillanatban az egész fájlrendszert szeretnénk menteni, így a / került oda. Az image-default paraméterrel meghatározzuk a bank megfelelő alkönyvtárában, milyen struktúrát építsen, így az aktuális dátum lesz az IMAGE neve, azaz például 2013-10-17. Az xdev opció 1-re való állításával megmondjuk az rsync-nek, hogy a távoli gépen csatolt összes lemezt mentse le.

Ha ezzel megvagyunk, akkor – mivel ez egy időzített mentés – biztosítani kell, hogy a root felhasználó jelszó nélkül tudjon csatlakozni. A legjobb és egyben biztonsági szempontból is elfogadható kompromisszumos megoldás erre, ha a mentendő szerver root felhasználója számára létrehozunk egy megfelelő SSH kulcsot, amely nem kér jelszót, a következőképpen:

```
ssh-keygen -t rsa -b 8192 -C elesszerver
```

majd megadjuk neki, hogy mi legyen a kulcsállomány neve:

```
Enter file in which to save the key (/root/.ssh/id_rsa): /root/elesszerver1
```

A következő kérdésre, azaz mi legyen a kulcs jelszava, csak egy Entert nyomunk:

```
Enter passphrase (empty for no passphrase):
```

Az így elkészült kulcs használata esetén nem fog jelszót kérni az SSH kliens, így a dirvish sem, de nézzük mit kell még tennünk, annak érdekében, hogy ez működjön:
a /root/.ssh/config állományt szerkesszük (a backup gépen), és oda vegyük fel a következő sorokat:

```
Host *
ServerAliveInterval 60
ServerAliveCountMax 10

Host eles1.szerver
  IdentityFile /root/elesszerver1
  Port 22
  Protocol 2
  User root
  HostName 172.27.1.1
  PasswordAuthentication no
```

Ha ezzel megvagyunk, akkor az éles szerverre fel kell juttatni az elkészített SSH-kulcs publikus részét, tehát például scp-vel másoljuk azt oda:

```
scp /root/elesszerver1.pub root@eles1.szerver.kft:.ssh/authorized_keys
```

Teszteljük is le a backup gépről indított ssh parancs segítségével, hogy minden rendben van-e, azaz beengedtük a tűzfalon, az SSH engedi a root hozzáférést, stb.:

```
ssh -i /root/elesszerver1 root@eles1.szerver.kft
```

Ha sikerült bejutnunk, akkor már csak annyi a dolgunk, hogy a dirvish-t is inicializáljuk:

```
dirvish --vault eles1.szerver --init
```

Az elkészített konfiguráció alapján a dirvish az SSH-n keresztül lementi – a kivételek kezelése mellett – az egész gépet. Majdnem készen is vagyunk, most már csak az olyan dolgokat kell még mentenünk, amelyeket futási időben nem célszerű. Ilyen tipikus példa az SQL szerverek (MySQL, MariaDB, PostgreSQL stb.), ezeket nem csak nem célszerű futási időben egy egyszerű cp vagy rsync segítségével másolni vagy szinkronizálni, hanem felesleges is, mivel tipikusan egy nyitott adatbázisba a mentési idő alatt is írnak, azaz a tartalma változik. Az aktuális állapotokat az ilyen esetben érdemes a Dirvish segítségével vagy Pre, vagy Post szkriptek írásával[21], vagy pedig előre ütemezett SQL mentés segítségével a saját segédeszközeikkel dumpolni, majd a konzisztens és mentés értékű dump állományokat akár fájlrendszer szinten már menteni és verziókövetni. De nézzünk konkrét példákat:

Jelen esetben mivel a MySQL és a MariaDB teljesen kompatibilisek (lásd bővebben az Adatbázis-kezelők[22] fejezetben) a mentési parancsok és tervezés sem fog eltérni, ezért a továbbiakban csak SQL-nek fogjuk hívni a MariaDB-t és MySQL-t. Ez a mentési megoldás feltételezi, hogy az adatbázis-szerver külön szerveren szeparálva található, így ütemezetten a cron segítségével és egyedi szkript írásával készítjük el az SQL mentését, nem a dirvish adottságaira támaszkodunk.

Az SQL adatbázisok mentésére több megoldás is adott. Egyrészt van a hagyományos út, amikor a mysqldump-ot használjuk, vagy rendelkezésre áll egy hotcopy nevű kiegészítés, amelyet inkább csak a mysqldump mellé, biztosítékként, vagy más cél mentés készítésére találtak ki.

mysqldump[23]

Kifejezetten a konzisztens adatbázis-mentésre kihegyezett eszköz, a mysql csomag telepítésével a birtokosai leszünk az eszköznek. Természetesen ugyanúgy szabályozható az /etc/mysql/my.conf fájlban a dump által használt erőforrások egy része is, mint a sima MySQL csatlakozások esetén. Lokális mentés esetében a bevett módszer, hogy a mysqldump parancsot egy szkript részeként crontab-ból hívva egy olyan időintervallumban futtatjuk, amikor a szerver úgynevezett idle (üresjárati, csúcsterhelésen kívüli) periódusban van. Az ilyen időszakok tervezésében nagy segítségünkre lehet a munin, amelyről a Naplózás, monitoring[24] fejezetben olvashatunk bővebben. Egy egyszerű mentő szkript tehát valahogy így nézzen ki (az adatbázis-szerveren)

```
a szkript neve és elérési útja: /backup/ment.sh
#! /bin/sh
/usr/bin/mysqldump --all-databases -pjelszo --events | /bin/gzip > /backup/SQL/mysql-ALLDATABASE-$1.tgz
/usr/bin/mysqldump -p --skip-lock-tables information_schema | /bin/gzip > /backup/SQL/openmail-mysql-information_schema-$1.tgz
/usr/bin/mysqldump -p --events mysql | /bin/gzip > /backup/SQL/openmail-mysql-mysql-$1.tgz
```

[21] http://wiki.dirvish.org/ClientScriptsOnServer

[22] http://szabadszoftver.kormany.hu/szabad-szoftver-keretrendszer/

[23] http://dev.mysql.com/doc/refman/5.1/en/mysqldump.html

[24] http://szabadszoftver.kormany.hu/szabad-szoftver-keretrendszer/

```
/usr/bin/mysqldump -p website | /bin/gzip > /backup/SQL/openmail-mysql-website-$1.tgz
```

Mint az látható, az első sorban az összes adatbázist kimentjük egy nagy állományba, majd mindent még egyszer kimentünk egyenként külön tömörített állományokba is. Miért szükséges ez? A gyakorlati tapasztalat azt mutatja, hogy ha csak mindent egy nagy állományba mentenénk ki, akkor viszonylag nehezen férhetnénk hozzá, ha csak egy adott rekordot töröl le pl egy programozó a weboldalunk adatbázis-állományából és csak arra van szükség, akkor elég csak azt az egy adatbázist visszadumpolni egy teszt adatbázisba. Viszont ha például egy korrupt fájlrendszer miatt kell teljes visszaállítást végeznünk, akkor célszerű az egész komplett mentést egyszerre visszatölteni (--all-databases), hiszen ilyenkor minden reláció (amelyek az information_schema, vagy mysql nevű adatbázisban vannak tárolva) visszatöltődik. Éppen ezért, ha van rá lehetőségünk (nyilvánvalóan egy sok 100 GB vagy TB-os adatbázis mentését ne így végezzük) és lemezterületünk, akkor mind a két módszerrel mentsünk. Az első inicializáló mentést végezhetjük a time parancs a mentő szkript elé helyezésével, hogy legyen fogalmunk arról, az SQL motor mentés körülbelül mennyi időt vesz igénybe:

```
time /backup/ment.sh
```

Figyeljünk rá, hogy az SQL mentés néha képes hibákat visszaadni, így a cron kimenete legyen mindig egy olyan e-mail címre irányítva, amelyet olvasunk is. Ugyanis annál, hogy nincs mentés, csak az rosszabb, ha azt hisszük, tökéletes mentést készítünk, de már évek óta a /dev/null-ba irányított hibaüzenetekkel fut a mentés, amelynek esetlegesen az az eredménye, hogy üres vagy éppen hibás állományt mentünk. Éppen ezért célszerű néha kicsomagolni az SQL mentés állományait vagy zless segítségével belenézni (persze csak ha nem sok GB nagyságúak). Illetve célszerű betervezni olyan napokat esetleg havonta vagy fél évente, amikor a teljes SQL motort egy teszt környezetben újra felépítjük a mentésből.

Felmerülhet a kérdés, hogy mi van akkor, ha én az összes adatbázist is menteni szeretném egyben és külön-külön az adatbázisokat, de nem érek rá lekövetni, hogy ki mikor hoz létre új adatbázist, de mégis szeretném, hogy azok is benne legyenek a szeparált mentésben?

Erre egy remek lehetőséget ad a következő szkript[25], amely a mysql show és a mysqldump tudását használja fel arra, hogy ha valaki létrehozott a tudomásunkon kívül egy új adatbázist, akkor az is biztosan le legyen mentve:

```
#! /bin/bash

TIMESTAMP=$(date +"%F")
BACKUP_DIR="/backup/$TIMESTAMP"
MYSQL_USER="backup"
MYSQL=/usr/bin/mysql
MYSQL_PASSWORD="jelszo"
MYSQLDUMP=/usr/bin/mysqldump

mkdir -p "$BACKUP_DIR/mysql"

databases=`$MYSQL --user=$MYSQL_USER -p$MYSQL_PASSWORD -e "SHOW DATABASES;" | grep -Ev
"(Database|information_schema)"`

for db in $databases; do
```

[25] http://dev.mensfeld.pl/2013/04/backup-mysql-dump-all-your-mysql-databases-in-separate-files/

```
 $MYSQLDUMP --force --opt --user=$MYSQL_USER -p$MYSQL_PASSWORD --databases $db | gzip >
"$BACKUP_DIR/mysql/$db.gz"
done
```

Mysqlhotcopy[26]

A hotcopy egy remek megoldást kínál olyan esetekre, amikor csak bizonyos rekordokat kell visszaállítanunk, vagy éppen muszáj futási időben az éles szerveren menteni, de sok konkurens írás/olvasás miatt a mysqldump zárolásos mentési módszere nem várható ki, vagy olyan lassulást eredményezne, amely nem vállalható futási időben. Azaz ezt a módszert főként napközbeni duplikálásra, vagy közbenső mentésre ajánljuk. Használata csak az InnoDB[27] típusú adatbázis esetén engedélyezett, viszont mivel általában ez az alapbeállítás, így általános esetben működőképes. A kézikönyve szerint online mentésnek hívják, ugyanis nem zárolja az egész adatbázist, hanem táblánként használja a lock, a flush és az unlock parancsokat, miközben kimásolja állományba *.frm, *.MYD, *.MYI állományokat. Használhatjuk ugyanúgy szkriptben és cronból meghívva:

```
mysqlhotcopy -u root -p jelszo dbneve /backups/SQL/HOTCOPY/ --allowold --keepold
```

Jellemzően a rendszergazdák az SQL mentő szkripteket a /backup könyvtárban tárolják, és onnan is hívják meg. Két fontos dolog amire érdemes figyelni: a /backup könyvtár mindenestül csak a root számára legyen olvasható, írható és bejárást (+x) is csak számára biztosítsunk. A második fontos tanács, hogy a backup szkriptet is mentsük. Egy esetleges átállás vagy adatvesztés esetén egy rendes backup szkript elkészítése is sok munkaórát takarhat. Ezért érdemes vagy átlinkelni az /etc alá, amit mentünk, vagy eleve ott tárolni.

Melyiket válasszam: a Dirvish pre szkript rendszerét, vagy az ütemezett mentést az SQL gépen?

A válasz viszonylag egyszerű: ha nem akarunk vagy nem tudunk törődni azzal, hogy kézzel ütemezzük be az SQL mentéseket, akkor célszerű post szkriptet készíteni, hiszen így a mentési folyamat fogja megszólítani és utasítani az SQL gépet a mentésre, majd ha az készen van, akkor fogja átemelni a backup gépre a biztosan elkészült mentést. Ha viszont az SQL gépen már eleve kénytelenek voltunk más megfontolásból lefuttatni a mentést például hajnali 2-kor, mivel csak 2-3 között vállalható az, hogy egy ekkora kaliberű terhelés érje a gépet – azaz ez a szempont fontosabb – akkor ehhez kell igazítanunk a dirvish-t, és esetlegesen beépíteni úgynevezett jelzőket a rendszerbe. Azaz, ha valamiért (nem várt lassulás, például a terhelés megugrása miatt) az SQL mentés bár elindult hajnali 2-kor, de nem záródik le a megszokott 45 perc alatt, akkor a mentő szkript elhelyezhet egy LOCK állományt a /backup/SQL alatt (echo /backup/LOCK), amelyet egy dirvish post szkript segítségével figyelhetünk, és kaphatunk róla értesítést, vagy felfüggeszthetjük a mentést a LOCK állomány meglétéig. Felfüggesztés esetén ütemezzük úgy, hogy a dirvish utolsó dolga az SQL állományok áthozatala legyen, hogy egy esetleges SQL szerver fennakadás és ezzel együtt a mentés megakadása miatt a többi rész mentése ne akadhasson el.

Hogyan mentsünk, melyiket használjuk és mikor?

A leggyakoribb mentőeszközöket soroltuk fel, amelyek segítségével össze lehet állítani egy ideális mentést a dokumentum elején felsorolt szempontok figyelembe vétele mellett. Dönteni egyik vagy másik mentőrendszer mellett a mentendő szerverek és kliensek ismerete nélkül nem célszerű.

[26] http://dev.mysql.com/doc/refman/5.0/en/mysqlhotcopy.html

[27] http://dev.mysql.com/doc/refman/5.0/en/innodb-storage-engine.html

Ajánlásunk az, hogy kombinálva használjuk a fenti rendszereket szükség szerint. Azaz a klienseket sok esetben nem szabad kifelejteni a mentésből, még akkor sem ha belső szabályzat kimondja, hogy a gépen tárolt dokumentumok nincsenek mentve, hiszen az adatvesztés ilyenkor is jelentős lehet, ha egy felhasználó megszegve az előírásokat a saját gépén tárolta a fontos dokumentumokat.

Kisvállalati, kis irodai közegben éppen ezért a szerverek mentésének Bacula, Dirvish használatával vagy egyéb komplex módon történő megoldása esetén is érdemes gondolni a kliensgépekre, és ha esetleg nincs lehetőség azok befűzésére a központi mentő megoldások közé, akkor is megoldás lehet, ha a kliens gépre telepítünk (értelemszerűen asztali Linux esetén) egy rsnapshotot, amely például a halt parancs meghívása előtt, vagy akár ütemezetten cronból a lokális merevlemezre végez egy mentést. Így ha a felhasználó nem is tartotta be a játékszabályokat, nem éri adatvesztés egy véletlen törlés esetén. Természetesen fizikai hiba ellen ez sem véd, hiszen ugyanazon az eszközön tároltuk a mentést és az éles adatokat. Az ilyen esetek ellen csak a független mentő szerver kialakítása védhet, ahol minden alaposan mentve és tesztelve is van.

Szerverek esetében pedig érdemes kombinálni a mentéseket. Ha van rá lehetőségünk, akkor egy teljes gép tönkremenetele esetén nagyban gyorsítja a visszaállítást, ha van egy teljes lemezkép, amelyet esetleg csak a napi mentéshez kell aktualizálni. Ilyenkor egy arra betanított ügyeletes is képes lehet az alap lemezkép visszaállítására, amely órákkal vagy akár napokkal is gyorsabb lehet, mint egy alaprendszer telepítése, majd arra a rendszer visszaszinkronizálása mentésből. Érdemes felkészülni, és akár papíralapú dokumentáció formájában a szerver mellett tartani egy boot loader dokumentációt. Hasznos lehet, mivel tipikusan a boot loader az, amelyet nem használunk nap mint nap. Nagyon ritkán, de előfordulhat, hogy az adataink nem sérülnek, de a partíciós tábla igen. Az üzemeltetők jelentős része ezt soha sem menti, hiszen sok éves tapasztalat alapján nincs rá szükség, ugyanakkor csak egy egyszerű parancs, és ha gond adódik a táblával, akkor van kiindulási alap. Akár így is:

```
dd if=/dev/sda of=/mentes/gepemneve-lemezem bs=512 count=1
```

Helyreállítás a mentés lemezkép alapján:

```
dd if=/mnt/gepemneve-lemezem of=/dev/sda bs=1 count=64 skip=446 seek=446
```

Adatmegsemmisítés

A mentéshez kapcsolható feladat a nem használt **eszközök biztonságos megsemmisítése**, amelyre oly gyakran nem fordítunk egyáltalán figyelmet. Egy már használaton kívüli szerver merevlemeze, vagy bármilyen adattárolója veszélyforrást jelenthet, ha csak úgy kidobjuk. Főként a szalagos egységek jelentik a mentéshez kapcsolódóan a legnagyobb veszélyt, mivel a régi használaton kívüli szalagokat ritkán szokták biztonságosan megsemmisíteni, egyszerűen csak kidobják a szemétbe. Rendszeresen olvasni a hírekben, hogy különböző adatbiztonsággal foglalkozó cégek véletlenszerűen vásárolnak használt komplett PC-ket, vagy csak adathordozókat, és próba visszaállításokat végeznek. A legtöbb esetben már eleve csak gyors formázást, vagy még azt sem végeznek az emberek, így az adatok könnyedén visszaállíthatóak. A szerverek illetve szalagok esetében valamivel bonyolultabb a helyzet, mert komolyabb szaktudás szükséges a visszaállításhoz például egy RAID eszköz esetén. Érdemes tehát számításba venni, hogy a kidobásra ítélt eszközök adathordozói veszélyforrást jelenthetnek. Egy remek eszköz a **Wipe**[28], amely Linux alatt az adatok biztonságos felülírását végzi.

[28] http://wipe.sourceforge.net/

Pull vagy Push?

A felsorolt mentő eszközök jelentős része támogatja lehetőséget ad arra, hogy egy központi mentő szerverről indított távoli általában SSH-n keresztüli bejelentkezésen keresztül a mentő szerver mentsen, illetve azt is, hogy fordítva a kliensek kezdeményezzék a mentést a távoli backup szerverre. Mind a két megoldás lehet jó és rossz is egyszerre.

Ha képesek vagyunk a mentő szervert fizikailag, hálózatilag és szoftveresen teljesen szeparálni, azaz jól megvédeni, akkor adott esetben jó megoldás lehet, ha a mentő szerver rendelkezik a távoli szerverekre root vagy azzal egyenértékű jogokkal és így a mentő szerver fogja ütemezetten kezdeményezni a mentéseket. Ebben az esetben viszont mivel az összes éles szerver SSH kulcsát feltettük a mentő szerverre, a mentő szervert ténylegesen extrém módon meg kell védenünk és bármilyen hozzáférést tiltani.

Ha a másik módszert választjuk, amikor is az éles szervereink jelentkeznek be a mentő szerverre, akkor nem koncentrálódik egy kézben az összes hozzáférés, viszont ebben az esetben az Achilles-sarka a mentésnek az lesz, hogy az éles szerver esetleges kompromittálódása esetén a támadó el fog „látni” a backup szerver irányába. Ez ellen védekeznünk kell mindenképpen. Erre több jó megoldás is van. Az egyik ezek közül, ha a mentő szerveren minden éles szervernek elkülönített chrootolt (SFTP vagy scponly shell) hozzáférése van. Ekkor a támadó „csak” az éles szervert és annak mentését tudja megsemmisíteni. Ennek kikerülésére, ha ezt a módot választjuk, érdemes a chroot környezetből egy időzített szkript segítségével minden mentés végeztével a backup anyagokat elmozgatni egy olyan területre, ahol a chrootolt backup felhasználó – amely az éles szerver felől jön – nem láthatja már. Sajnos számtalan esettanulmány bizonyítja internet szerte, hogy ha egyszer ténylegesen betörtek az éles szerverünkre, akkor a legtöbb esetben – célzott támadás esetén mindenképpen– végigmennek az elérhető összes lehetőségen (mentett gépek továbbtámadása) is, így a backup mint egy könnyen felderíthető (cron szkripteket használó) rendszer, adja magát. Ebből a szemszögből nézve adott esetben jobb megoldás lehet az első változat, amikor a backup szerver kívülről az éles szerverek felől elérhetetlen, csak ő tud kezdeményezni.

Visszaállítási tesztek: Ahogy már említettük mindenképpen legyen egy ütemezett tervünk, mikor melyik mentést teszteljük és állítjuk belőle vissza az adatokat. Ha nincs rá külön fenntartott gépünk, akkor használjunk virtualizációs környezetet.

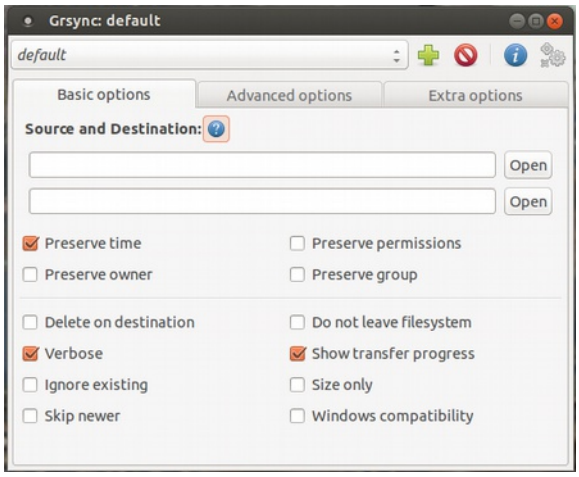
Kliensek mentése

Manapság inkább az jellemző, hogy az asztali számítógépek mint egyfajta vékonykliens viselkednek, azaz tárolunk ugyan adatokat rajta, de a munkával összefüggő dolgok vagy egy böngészőben, vagy egy SSH terminálban, vagy egy távoli SMB megosztásra történnek. Ebben az esetben az alapvető hozzáállás, hogy a klienst 1-2 órán belül ugyanolyanra tudjuk varázsolni, ha tényleg mindent a szerverre mentettünk. Ez így is van, azonban még így is célszerű menteni a kliensek /etc/ alatti konfigurációs állományait és legalább a /home környezetet. Ha a menteni kívánt kliensek Linux vagy BSD, vagy bármilyen Unix-klón alapúak, akkor szerencsések vagyunk, mert a már megismert módszerek egyikét alkalmazhatjuk. Azaz ha az asztali számítógép tud SSH kapcsolatot indítani vagy fogadni, akkor ugyanúgy menthetjük Rsnapshot-tal vagy akár Dirvish-sel is központilag. Ha pedig szeretnénk a felhasználóra bízni (ez általában nem jó ötlet), hogy a saját kliensén mit ment és mit nem (hiszen egy központilag mentett meghajtóra dolgozik amúgy is) akkor például a mentendő kliensen futtathatunk Grsyncet.

Grync[29]

Ezt a grafikus felületet az rsync meglehetősen bonyolult opcióinak átláthatóbbá tételére hozták létre. A fejlesztők a leginkább használt funkciókat grafikus formába öntötték. Akár a teljes funkcionalitást könnyen el lehet érni a kiterjesztett nézetben, azonban az igazán hasznos funkciója az, hogy az 1-2 perc alatt egérrel összeállított beállításokat egy nézet menüben parancs formájában is lementhetjük. Kezdő rendszergazdáknak ideális a kapcsolók nem teljes ismerete esetén, ugyanis a tévedést lehet vele minimalizálni. Működtetését szerverek között érdemes kombinálni az ssh-agent[30] és az ~/.ssh/config állományba felvett rövid gépnevek felsorolásaival. Például:

```
Host szervergep
  IdentityFile /home/admin/identity-szerver-titkoskulcs
  Port 22
  Protocol 2
  User root
  HostName szerver.belsohalo.hu
  PasswordAuthentication no
```



[29] http://www.opbyte.it/grsync/
[30] http://en.wikipedia.org/wiki/Ssh-agent