keny részese, egyre otthonosabban mozog a zenei folyamatban, műveli valamely alkotóelemét (legyen az ritmus, dallam, összhangzás, hangszín), amely számára megoldhatónak tűnik, amelyre érzi, hogy megvan a kompetenciája. A gyermek ilyen módon önmagát vezeti be a minőségileg magasabb szintű, zeneileg többrétű feladatok cizellált világába, megfelelő és tapintatos tanári vezetés mellett.

Irodalom

Dimény Judit (1981): *Hangjáték*. Zeneműkiadó, Budapest.

L. Nagy Katalin (1997): Alter-NAT-íva – örömmel zenére vagy zenével örömre nevelés heti egy órában is. *Parlando*, 6. 42–46.

Laczó Zoltán (1997): Egy zenehallgatás-központú tantervről. *Iskolakultúra*, 9. 92–98.

Dunun, Robert, E. (1999): Fall, Stop, Look, Listen and Move. Children's Perceptual Meditation and Music Listening. *Bulletin of the Council for Research in Music Education*, 142, 80.

Gordon, Cox (1999): *The Development of Creative Music in Schools.* Some Perspetcives from the History of Music in Education of Under-Twelves (MEUT), 1943–1983, No. 141. 32.

Wiggins, Jacqueline, H. (1999/2000): The nature of shared musical understanding and its role in music thinking. *Bulletin of the Council for Research in Music Education*, 1–143. 65.

Sáry László (2000): *Kreatív zenei gyakorlatok*. Ars Longa, Jelenkor Kiadó, Pécs.

Raymond, A. R., – Donald, Mac – Mill, Dorothy (2000): Creativity and Music Education. The Impact of Social Variables. *International Journal of Music Education*, 36. 58.

Burián Miklós
Tanítóképző Főiskolai Kar,
Kecskeméti Főiskola

Számítógépterem hatékony karbantartása ssh-val

Géptermünkben (a Budapesti Műszaki Főiskola Keleti Károly Gazdasági Karán) húsz darab egyforma PC van (szerencsés helyzet!), ugyanolyan „vason" (hardver) ugyanolyan operációs rendszer (Linux) és ugyanolyan alkalmazói programok. Néha szükséges elvégezni különféle műveleteket mindegyik gépen. Körbejárni a gépteremben, minden gép elé leülni, begépelni ugyanazon parancsokat... brr!

Kezdetben mondák néhányan, igazi programozók: „...és legyen hálózat!" (számítógépek összehangolása, együttes működése) – és lett hálózat, a mai internet őse. Nem lévén rosszindulatú ember, sem rosszindulatú államhatalom a hálózat körül, föl sem merült a megfejthetetlenül komoly titkolózás igénye. Ezen „boldog békeidők" máig élő emlékképe, hogy a leggyakrabban használt, legfontosabb hálózati műveletek és az azokhoz kapcsolódó jelszavak kis igyekezettel bárki által lehallgathatók, kimazsolázhatók az adatforgalomból. Az ftp, a pop3, a telnet mind ilyen.

Később, ráébredvén a titkolózás szükségességére, létrehoztak ehhez szükséges különféle eszközöket, majd eljutottunk a csúcsra, az RSA algoritmus számítógépes körülmények közé történt átültetésére: gondoljunk *Phil Zimmermann* PGP-jére vagy a telnet kapcsolatot (távoli gépre történő bejelentkezés) kiváltó ssh-ra (secure shell).

Az ssh (OpenSSH) azonban nemcsak arra jó, hogy a helyi terminál és a távoli gép közötti párbeszédet titkosítja (beleértve a bejelentkezéshez esetleg szükséges jelszót is), de kötegelt módban is használható. Végtelen rugalmassága rendkívüli lehetőségeket nyújt arra is, hogy a távoli gépeken végzendő műveleteket, karbantartást kényelmesebben, egyszerűbben és gyorsabban csinálhassuk meg.

Az OpenSSH feltehetően része az alaptelepítésnek bármely disztribúció esetén. Különféle szolgáltatásait és működési módjait illetően lásd: man ssh (valamint ennek a végén a „see also” – „lásd még” – pontban felsoroltakat).

A továbbiakhoz a következőket kell tudni. Az RSA algoritmus sajátossága, hogy a titkosítás és a visszaalakítás két kulccsal történik, melyek összetartozó párokat alkotnak. A kulcspár egyik fele az úgynevezett titkos kulcs, erre úgy vigyázunk, mint a szemünk fényére: ha a gépünkről kikerül, vége a titkosságnak. A kulcspár másik felét, az úgynevezett nyilvános kulcsot viszont közöljük az egész világgal, weblapon, telefonkönyvben és europlakátokon stb.

A kulcspár egyik felével titkosított szöveget csak ezen kulcs párjával lehet visszaalakítani. Ha tehát valaki a saját titkos kulcsával, majd a címzett nyilvános kulcsával kódolja üzenetét, a címzett azt saját titkos és a feladó nyilvános kulcsával tudja visszalakítani. Ekkor – feltéve a titkos kulcsok titkosságát és a nyilvános kulcsok hitelességét – a címzett biztos lehet benne, hogy valóban a feladótól származik az üzenet, és hogy útközben sem változott meg annak tartalma. RSA-kulcspárja van a számítógépnek és van a felhasználóknak. Az előbbit az ssh első indításakor generálja, az utóbbit a felhasználó az ssh-keygen-t rsa paranccsal hozhatja létre.

Az ssh egyik nagy előnye, hogy a távoli gépre való bejelentkezésnél a távoli gép és a felhasználó azonosítását a szokványos jelszó helyett RSA-kulcsok alapján is elvégzi, megtakarítva evvel a jelszavak beírogatását, ami a kötegelt feldolgozásnál alapvető követelmény. Ehhez azonban szükséges, hogy saját nyilvános kulcsunkat elhelyezzük a távoli gépeken.

Vegyük példának az én helyzetemet: saját gépemen (valami.valahol.hu) valaki felhasználóként jelentkezem be. A 20 db géptermi gép IP címei 10.66.24.75-től 10.66.24.94-ig terjednek, a saját gépemen az /etc/hosts tartalma a következő:

```
10.66.24.75 pc1
10.66.24.76 pc2
... ... ...
10.66.24.94 pc20
```

Ahhoz, hogy a saját nyilvános kulcsomat eljuttassam a húsz géptermi gépre, még mindenképpen jelszóval kell bejelentkeznem. A biztonsági kockázatok alapos elemzése után nem járom körbe a géptermet, hogy lemezről bemásoljam a nyilvános kulcsomat mindegyikre, hanem már ezt is az ssh-val végzem el. Feltéve, hogy a géptermi gépeken a rendszergazda (root) még semmilyen ssh-kulcsműveletet nem végzett, a következőkre van szükség:

```
ssh root@pc1 „mkdir /root/.ssh; chmod 0700 /root/.ssh“
...
ssh root@pc20 „mkdir /root/.ssh; chmod 0700 /root/.ssh“
```

Ezt a húsz parancsot egybefoghatjuk (a még mindig parancssori!)

```
i=1 ; while (($i<21)) ; do ssh root@pc$i „mkdir /root/.ssh; chmod 0700 /root/.ssh“; ((i=i+1)) ; done
```

utasítással.

Ez létrehozza az összes géptermi gépen a /root/.ssh könyvtárat, és beállítja annak jogosultságát: „rwx——“, de előbb még meg kell adnom a rendszergazdai jelszót mindegyik gép esetében. Ezután jön a nyilvános kulcs másolása (Ha az egyes gépeken a root felhasználónak már van értékes adatokat tartalmazó /root/.ssh/authorized_keys állománya, úgy azt nem felülírni kell az id_rsa.pub állomány tartalmával, hanem a végéhez hozzáfűzni azt):

```
scp /home/valaki/.ssh/id_rsa.pub pc1:/root/.ssh/authorized_keys
```

illetve:

```
cat /home/valaki/.ssh/id_rsa.pub | ssh root@pc1 „cat >> /root/.ssh/authorized_keys
```

értelemszerűen mind a húsz gépre, pc1-től pc20-ig.

Sajnos még egyszer meg kell adnom a jelszót. Ezek után viszont az ssh root@pc1 parancs azt eredményezi, hogy a parancssori prompt valaki@valahol helyett root@pc1 lesz, azaz az első géptermi gép rendszergazdai parancsértelmezőjében vagyok (visszajönni az exit kiadásával lehet).

Ezután a valaki@valahol felhasználó jelszavak megadása nélkül tud a pc1, pc2,

... pc20 gépeken rendszergazdai jogosultsággal parancsot végrehajtani, valamint fájlokat másolni rájuk. Másra pedig nincs is szükség, mert ami nem hajtható végre egy (esetleg összetett) paranccsal, azt le kell írni egy scriptbe, azt eljuttatni minden érintett gépre, majd lehet azokat futtatni...

Azért lehet még kényelmesebbé és gyorsabbá tenni a karbantartási műveleteket. Ehhez az alábbi keret-scriptet használom, és éppen emiatt kapták az egyes gépek a nem különösebben fantáziadús neveket:

```
#!/bin/bash
#
for i in 1 2 3 4 5 6 7 8 9 10 11 12 13 14
15 16 17 18 19 20 ; do
   ssh root@pc$i „df“
done
```

Természetesen az ssh root@pc$i „df” helyére kerül a tényleges ssh vagy scp utasítás. A FOR ciklus oka az, hogy ha valamely gép átmenetileg nem elérhető, vagy az adott feladatot nem kell rajta elvégezni, akkor úgy a legkönnyebb kihagyni a sorból, hogy a megfelelő számot egyszerűen kitörlöm a sorból, később pedig szükség esetén visszaírom. Ha előbb a fentebbi első ssh-s parancsot írom a ciklusmagba, majd a másodikként szereplő scp-t, nincs más dolgom, mint gyorsan egymás után, összesen negyvenszer megadni a rendszergazdai jelszavakat. Saját gépemről utoljára.

A gépek pillanatnyi elérhetőségének vizsgálata az alábbi script futtatásával történik, az eredmény a ping_van (az eredménytelenség a ping_nincs) állományokban olvasható.

```
#!/bin/bash
#
van=ping_van.txt
nincs=ping_nincs.txt

if [ -f $van ]; then
   rm $van          # Ha van korábbról, le-
töröljük...
fi
if [ -f $nincs ]; then
   rm $nincs        # ...ezt is
fi

let i=1
while ((i<21)) ; do
  if ping -c 1 pc$i 2>&1 > /dev/null; then
     echo pc$i >> cping_van.txt
  else
        echo  pc$i  >>  cping_nincs.txt
  fi
  let i=i+1
done
```

A továbbiakban, ha bármilyen fájlt kell a géptermi gépekre eljuttatnom, a keret-script ciklusmagjába a fentebbihez hasonló, megfelelő scp utasítást teszem, ha pedig parancsot kell végrehajtani, akkor a megfelelő ssh root@pc$i „parancs” sort használom. Tegyük fel, hogy szeretném összegyűjteni, hogy a géptermi gépek /mnt/hda2/hely könyvtáraiban (ilyen nevű könyvtár létezik mindegyik gépen) mi minden található. A ciklusmag a következő két utasítás lesz:

```
   echo
„=========pc$i:/mnt/hda2/hely==  ===”
>> lista.txt
   ssh root@pc$i „ls -laR /mnt/hda2” >>
lista.txt
```

A lista.txt természetesen(?) a saját gépen keletkezik.

Azért van szépséghibája az ssh-nak is: a nohup ... & utasítással nem jutunk dűlőre, ugyanis az ssh nem lép ki addig, amíg az elindított folyamatok (process) be nem fejeződnek. Ha például a gépeket szeretném távolról leállítani, akkor azt ilymódon nem tudom megtenni, mert az ssh nem lép ki, a távoli gép leállását követően viszont az ssh-kapcsolat „befagy”. Azért erre is van megoldás: előbb minden gépre odamásolom az init 0, vagy a /sbin/shutdown -h now parancsot tartalmazó scriptecskét mondjuk /root/stop néven, majd a következő parancsot írom a keret-script ciklusmagjába: ssh root@pc$i „at -f /root/stop now +1 min”.

Ennek eredményeképpen a ciklus gyorsan és zavartalanul végigmegy az összes gépen, a tényleges leállásuk ugyan egyperces késéssel indul el, viszont az időpont megadását illetően, az at rugalmas lehetőségeinek köszönhetően, az egyes gépek pillanatnyi rendszeridejéhez képest viszonylagosan. Természetesen a módszer alkalmazható időnyerés céljából is, ha az

egyes gépeken néhány másodpercet meghaladó időigényű műveleteket kell végezni.

Persze arra is szükség van, hogy az egyes gépekre egyedileg jelentkezzek be. Ennek további könnyítésére szolgál a következő megoldás *Flickenger* ötlete nyomán (*Flickenger*: ‚Linux bevetés közben', Kiskapu, 2003). Az alábbi scriptet a saját gépemen a /home/valaki/bin könyvtárba helyezem el (keresési útvonalban benne van) mondjuk ssh-oda néven:

```
#!/bin/bash
#
if [ $# -eq 0 ]; then
  user="root"
else
  user=$1
fi
host=`basename $0`
ssh $user@$host
```

Majd csinálok húsz linket a /home/valaki/bin/ könyvtárban:

```
ln -s ssh-oda pc1
ln -s ssh-oda pc2
... ... ...
ln -s ssh-oda pc20
```

Ezután, ha mondjuk az ötödik gépre kell bejelentkeznem rendszergazdaként, akkor a pc5 parancsot elég kiadnom. Ha ugyanerre a gépre mint „diak" felhasználó szeretnék bejelentkezni, akkor a pc5 diak parancsot adom ki... és ennél egyszerűbb már csak úgy lehetne, hogy „Hyppolit, legyen szíves..." :)

Fölmerülhet valakiben a kérdés, hogy milyen mértékű biztonsági kockázatot jelent az itt vázolt megoldás. Ha ugyanis valaki jogosulatlanul bejut a „valami.valahol.hu" gépre (valaki vagy rendszergazda jogosultsággal), akkor hozzáfér a titkos RSA-kulcsomhoz, egyben az összes géptermi géphez is. A probléma általános: az RSA algoritmus „gyönge pontjára" világít rá: a titkosság a kulcsok biztonságos megőrzésén áll vagy bukik. Nos, a /home/valaki/.ssh könyvtár lehet szimbolikus link egy usb-kulcs felé (pár kilobájtról van szó), amikor is a bejutás a gépemre a távollétemben semmiben sem segíti elő a hozzáférést a további gépekhez... Ha én dolgozom, akkor a kulcsnak benne kell lennie a gépben, és ha ilyenkor sikerül valahonnan a világ végéről betörni... – hát a biztonság valahol ott kezdődik, hogy nemcsak az ethernet kártyából húzzuk ki a lengőkábelt, hanem a tápkábelt is a 220V-os dugaljból.

Keszthelyi László
Keleti Károly Gazdasági Kar,
Budapesti Műszaki Főiskola

Az analízis hérosza

Találkozás Robert Musillal

Robert Musil nemcsak kiemelkedő, hanem a szó minden értelmében reprezentatívvá is vált alakja a közép-európai kulturális univerzumnak. Annak a klasszikus modernségnek pedig, amely a mi korszakunk magasabb szellemiségét jelenti, a közvetlen kiindulópontja.

Jellemző módon Musil 1914 előtt alig foglalkozik Ausztria-Magyarország másik felével. Annál érdekesebbek azok a feljegyzései, melyeket 1938-ban, egy rövid budapesti látogatás során készített, s melyek a rendkívüli analitikus képességekkel megáldott (vagy megvert) író éleslátásával már majdhogynem elhihetetlen képet adnak a világháború előtti Budapestről: „Dollárélet, és eközben (van) idő arra, hogy élvezzék a levegőt, a fényt, az asszonyt, a férfit, egy jó lószerszámot és mindent, ami feltűnő. (...) A kendőknek olyan erős a színük, hogy még párizsi festők sem találnak ki olyat. (...) Nem tudom százalékokban kifejezni, mennyi magyar