Linux rendszerfelügyelet: felhasználói kézikönyv

Kivonat a mi francia szakácsunk hamarosan megjelenő könyvéből.

Jövőre jelenteti meg új könyvemet az Addison Wesley Longman kiadó, címe Linux rendszerfelügyelet: felhasználói kézikönyv. Felkértek, hogy adjak egy kis ízelítőt a hamarosan megjelenő könyvből.
Először hadd mutassam be az alaphelyzetet. Sötét, viharos éjszaka van (ezt mindig is le akartam írni), a magányos rendszergazda már régen szerette volna befejezni a munkáját és hazamenni.
Ez a részlet a 15. fejezetből való, amit én egyszerűen csak „alkotó lustaság”-ra kereszteltem. Néhány évvel ezelőtt kisebb-nagyobb társaságban szívesen emlegettem, hogy én úgy gondolom, hogy az alkotó lustaság egy csodálatos dolog. Mindig is csodálatra méltónak találtam a lusta embereket, mert mindig megtalálják a feladat megoldásának legegyszerűbb és legelegánsabb módját. Idézném a tudományos-fantasztikus irodalom nagyját, *Robert A. Heinleint,* aki egyszer a következő gondolatot vetette papírra: „A lusta ember a dolgokat mindig a legegyszerűbben intézi el.”
Ebben a fejezetben az automatizálás eszközeit fogom bemutatni. Már elmúlt este 11 óra. A pizza már a múlté, a kávéházban a kávé is elfogyott. Fények. Kamera. Csapó, tessék...

Az interaktív folyamatok automatizálása: expect

Először is úgy tűnik, igencsak fogytán vagy a szerencsédnek, ha az emberi beavatkozást szeretnéd automatizálni. Ezek azok a dolgok, amihez egy emberre is szükség van: választani egy menüből, beírni a jelszót vagy dönteni a megjelenő adatok alapján. Az interaktív alkalmazásoknak szükségük van a felhasználó beavatkozására, nem igaz? A lusta rendszergazda szerint nem mindig, köszönhető ez egy expect nevű kis programnak.
Bár én már régebben is hallottam az expectről, csak néhány éve fedeztem fel, mennyire hasznos. A munkatársammal egy olyan webalapú rendszert fejlesztettünk, ahol szükség volt egy egyszerű frissítésre a központi számítógép adatbázisából, ahol nem volt engedélyezett a parancsfájlok használata. A végrehajtáshoz szükséges adatok, az SQL utasítás szerint, csak a felhasználói felületen vihetők be. Ez az SQL utasítás fogja létrehozni a webes felülethez szükséges adatfájlokat. Az egész folyamat mozgatórugója a felhasználó, aki a terminál előtt ülve különböző helyekre különféle adatokat ír be. Később az expect lehetővé tette a weboldal további igények szerinti fejlesztését.
Vajon tényleg szükségünk van erre? Emlékszünk még a fejezet elején található elmélkedésre a lustaságról? Nos, tegyük fel, késő éjszakáig dolgozol, és indulás előtt az utolsó teendőd, hogy bejelentkezz egy távoli weboldalra és megbizonyosodj arról, hogy az alkalmazás elvégezte a munkát (hajnali háromra mindig kész is lesz), majd letöltöd az alkalmazás által létrehozott fájlt a helyi oldalra.
Este tíz órakor már inkább a hazamenésen jár az eszed, mint a fájl elkészülésének varázslatos pillanatán. Csak el kellene indítanod a programot, amely elindítja az ncftp-t a letöltéshez, de nem tudod a fájl nevét, mivel az minden futtatáskor megváltozik. A nevet megtalálod, ha bejelentkezel a rendszerbe és megvizsgálod a befejezési naplót (completion log).
Az egész aprócska folyamatot csupán a szemléltetésség kedvéért mutattam be, mert ez is jól példázza, hogy egy egyszerű kis parancsállomány milyen nagy segítséget jelenthet.
Az expect parancsfájl alapformája a következő:

```
#!/usr/local/bin/expect
# Megjegyzések (név, mit csinál stb.,
# opcionális)
spawn valamilyen_parancs
set response énválaszom
expect "Valami válasz . . . ."
send $response\r
close
```

Nézzük, mi is történt: a spawn kulcsszó megadja az expectnek, hogy indítsa el a programot. Ez lehet egy parancsállomány (spawn /bin/bash), vagy a folyamat bármilyen parancsa. A set kulcsszóval beállítottam egy választ az előre meghatározottak közül. A nyelv érdekessége, hogy az expect kulcsszó itt pontosan azt teszi, amit jelent. Megvizsgálja a spawn-ban meghatározott parancs kimenetét, és megkeresi az egyező szöveget. Azután a send paranccsal válaszol a keresett szövegre az első változóval, `$response`. Most nézzünk egy valódi példát.
Egy Apache webkiszolgálót futtattam a rendszeremen OpenSSL-kiegészítéssel, a biztonságos ügyletek céljából. Az Apache, az OpenSSL kiegészítéssel, az indulás során egy biztonsági jelszót (pass phrase) kér, mivel a szerveren az egyedi kulcsfájlok kódolva vannak. Addig minden rendben van, amíg én ott vagyok és beírom a jelszót, de mi történik akkor, ha a kiszolgáló éppen akkor áll le, amikor én nem vagyok ott? Ez hónapokig nem fordul elő, csak akkor számíthatok rá, ha éppen szabadságra megyek. Egy őrült pillanatomban egy SCSI kártyát építettem be az új szalagos meghajtómhoz. Csakhogy, elfelejtettem (megtörtént) újraindítani a kiszolgálót, ami az OpenSSL-t is futtatta. Most mi lesz? A feledékenység kiküszöbölésére írtam az alábbi egyszerű expect parancsfájlt:

```
#!/usr/bin/expect
# Routine: startapachessl
# Purpose: A web szerver elindítása az OpenSSL \
       kiegészítéssel
#
log_file -a /tmp/expectlog
#log_user 0
spawn /bin/bash

sleep .2
send "usr/local/apache/bin/apachectl startssl\r"
expect "Enter pass phrase*"
sleep .2
send "azéntitkosválaszom\r"
sleep .2
close
```

Ha a rendszer újraindul, akár ott vagyok, akár nem, ez a parancsfájl újraindítja az Apache kiszolgálót is az OpenSSL kiegészítéssel.
A programot vizsgálva, néhány érdekes dologra lehetünk figyelmesek. Például a log_file értéke új. Meg kell határozni egy log fájlt a parancsállomány végrehajtásához. Azt, hogy a fájlba írjunk-e, vagy sem, a log_user értékében határozhatjuk meg. Ha ez „1”, a naplózás meg fog történni. Én a parancsfájl ellenőrzése során szívesen hasz-

náltam a log_user értéket; mindenki el tudja dönteni, hogy az állandó naplózásra szüksége van-e vagy sem. Megjegyzem, én a bash héjból indítottam a kiszolgáló újraindítására szolgáló parancsfájlt, ami azután alvó állapotba került. Minden esetben, a parancsállományban egyötöd másodpercet vártam a folytatással. Végül, a záró lépés jelzi az elindított folyamatnak, hogy a továbbiakban nincs rá szükség. Ezzel az expect parancs kilépett, és visszatért az őt elindító folyamathoz.
A továbbiakban nincs szükség arra, hogy más nyelveken programozzuk ezeket a funkciókat, mert az expect használatával sokkal egyszerűbb. Amint azt az előbbiekből is láthattuk, nem minden eszköz felel meg minden feladathoz. Az interaktív alkalmazások gyors és aljas indokból elkövetett automatizálásához, az expect parancsnál jobbat nem igen találunk. Az expect teljes megismeréséhez egy egész könyvre szükség lenne (nekem például van egy). Hogy is tudnám jobban megismertetni a használatát, mint példákkal bemutatni a nyelv minden egyes aspektusát. Mielőtt elszaladnál kísérletezni, nézzük meg az alábbi példákat.

1. kép Űrlap jelszóváltoztatáshoz

Mindannyian tudjuk, a jelszóváltoztatás alapvető dolog, ahogy egy jó jelszó kiválasztása is. Ez nem olyan nagy dolog, ha a felhasználó éppen bejelentkezett; de egy kicsit bonyolultabb, ha egyáltalán nincs felhasználói fiókja. Értem ezalatt azokat az e-mail felhasználókat, akik POP3 (vagy webes) eléréssel használják a kiszolgálót, de nincs parancssori hozzáférésük. Számos irodában, ahol e-mail kiszolgálóként és Internet-átjáróként Linux-alapú rendszert használnak, pontosan ezzel a gonddal küzdenek. Hogyan engedélyezzük a felhasználóknak jelszavaik megváltoztatását, ha nincs engedélyük a belépésre? Mindenek előtt a jelszó-változtatási eljárás az alábbiak szerint történik:

```
[root@myhost] # passwd
New UNIX password:
```

Kicsit bonyolultabb a helyzet, ha a felhasználó nem rendszergazda, és mégis meg szeretné változtatni a jelszavát. Ehhez először meg kell adnia a régit, így a párbeszéd még bonyolultabb. Most mi lesz? Létre kell hozni egy webalapú űrlapot, ahová a felhasználó minden szükséges adatot beírhat (*lásd 1. kép*). Az űrlap mögött dolgozó Perl parancsfájl kiemeli a változókat és beilleszti az expect parancsállományt, ami már elvégzi a további feladatokat. Ha szeretnéd alaposan megvizsgálni vagy használni ezt az ügyes kis webes alkalmazást, nyugodtan letöltheted a weboldalamról. Addig is nézzük meg ezt a részt az alkalmazás oldaláról.

```
#!/usr/bin/expect
# Routine: psdcmd
# Purpose: A felhasználói jelszó megváltoztatása \
       az expect használatával
log_user 1
set uservar [lindex $argv 0]
set currpassword [lindex $argv 1]
set newpassword [lindex $argv 2]
set renewpassword [lindex $argv 3]
#
# log_file -a /tmp/expectlog
# send_user "Spawning passwd command with \
       uservar.\n"
spawn su -l -c "passwd" $uservar
expect "Password:"
sleep .1
send "$currpassword\r"
sleep .1
#
expect {
   "(current) UNIX password:" {send \
       "$currpassword\r"}
   "su: incorrect password" {exit 0}
   }
sleep .1
expect {
   "su: incorrect password" {exit 0}
      "New UNIX password:" {send \
       "$newpassword\r"}
}
sleep .1
expect {
   "BAD PASSWORD:" {exit 0}
      "Retype new UNIX password:"
      å {send "$renewpassword\r"}
}
sleep .1
expect {
   "su: incorrect password" {exit 0}
      "New UNIX password:" {exit 0}
}
#End of password change routine
```

A set varname [lindex $argv num] felépítés az expectnek megfelelő tartományt mutatja. Megjegyzem, a „spawn” paraméter hívja a su parancsot a felhasználó jelszavának megváltoztatásához. Alaphelyzetben a webkiszolgálón található CGI parancsfájlok hajtják végre az átlag felhasználók (mint „nobody”, „www”) jelszavainak megváltoztatását, de a kiemelt felhasználók jelszavait nekünk kell megváltoztatni. Mellesleg, az Apache-kiszolgáló alapértelmezett felhasználóját meg kell tartani nem rendszergazda jogosultságúnak, különben borzalmas eseményeknek leszünk szem- és fültanúi.
Ebben a parancsfájlban egy másik új elemet is találhatunk, a send_user értéket. Ez tulajdonképpen egy állapotnyomtatási sor. Szándékosan benne hagytam a parancsfájlban, mert szerettem volna, ha tisztán látszik a hibakeresés (debugging) működése az expect parancsfájlban. Minden programozó beilleszt egy hibakeresési sort, hogy láthassa mi, miért és hogyan történik. Ez ebben az esetben is így van. A send_user parancsot használhatjuk a parancsállomány külvilággal társalgására, beleértve a végrehajtás folyamatát is. Mivel a Perl parancsfájl segítségével látom az expect kimenetét, jól láthatom az alábbi üzeneteket is. Egyúttal a Perl meghívja az alábbi parancssort:

```
$return_code =3D './psdcmd "$username"
"$currpassword" "$newpassword" "$renewpassword" ';
```

Amint látszik, az expect parancsfájl hívásra kerül a felhasználói névvel, a jelenlegi jelszóval, az új jelszóval és az új jelszó ismétlésével. Egyszerűen beírhatjuk az új jelszót kétszer, de az ellenőrzési eljárást

a parancssori megoldáshoz hasonlóan kell megoldani. Azonkívül feltétlenül jó ötlet a felhasználói jelszóváltoztatás megerősítésének kikényszerítése.

Az interaktív automatika automatizálása

Most, hogy túl vagyunk az expect parancsállomány bevezetőjén, hihetetlenül könnyen tudunk különböző eljárásokat automatizálni. A parancsállomány létrehozására nem lehetne egy programot készíteni? Az expect telepítésekor felkerül a gépre egy autoexpect nevű apró kis program. Egyszerűen szólva, az autoexpect figyeli az interaktív folyamatban való ténykedésünket, és elkészíti számunkra az expect parancsállományt.
A parancsot az alábbi alakban kell használni:

```
autoexpect -f script_outputfile command_string
```

Tegyük fel, hogy be szeretnénk jelentkezni egy tűzfal mögött lévő távoli gépre, kétszeres bejelentkezéssel. Miután bejelentkeztünk a tűzfalra, bejelentkezünk a belső hálózatba is (telnet, ssh stb.), majd elindítjuk a szokásos menüt. Ezt az egész bejelentkezési és programindítási eljárást szeretnénk automatizálni. A parancssorba a következőket kell begépelni:

```
autoexpect -f superlogin.script telnet \
       firewall.mycompany.com
```

Adatok

Szerző: Marcel Gagné
Cím: Linux rendszerfelügyelet: Felhasználói kézikönyv
A kiadás tervezett ideje: 2001
Azonosítója: ISBN 0-201-71934-7

Amikor végeztünk a bejelentkezéssel, ki kell lépni a menüből, és ki kell jelentkezni. Az autoexpect szépen rögzítette az egész folyamatot. Mielőtt az új parancsállományt futtatnánk, akad némi szerkesztési munkánk, mivel az autoexpect egy kicsit bőbeszédűbb, mint azt mi szeretnénk. Továbbá el kell távolítani a kilépést a menüből és a rendszerből, de az alap-parancsállomány teljes testi épségében elérhető. Már csak indítható állományt kell elkészíteni a parancsállományból, és már végeztünk is.
Még valamit azonban meg kell tennünk, az új expect parancsállomány végére be kell illeszteni a következőt: interact.
Ez a parancs utasítja az expectet, hogy adja vissza a vezérlést a rendszernek, miután elvégezte a feladatát. Enélkül elfoglalná az egész gépet, és csupán be- és kijelentkezhetnénk, de azt nagyon gyorsan. Véleményem szerint, az expect használatáról nem lehet teljes képet adni, de remélem, hogy e rövid bevezetővel kellemes étvágyat keltettem, és megfelelő képzelőerő segítségével, az alkotó lustaság tökéletességig való fejlesztéséhez lökést adtam. Ezután más elfoglaltságot kereshetünk magunknak.
Mi ez az izé a varázsköpenyről a képernyődön?

Marcel Gagné (mggagne@salmar.com) Mississaugában, Ontarióban él. A Salmar Consulting Inc. (rendszerintegrációs és hálózati tanácsadó vállalat) elnöke. Továbbá pilóta, tudományos fantasztikus regények szerzője, és a TransVersions tudományos fantasztikus, fantasy- és horrorválogatás szerkesztője. A Linux és Unix-változatok nagy kedvelője, és ezt nyilvánosan is hangoztatja. Honlapján rengeteg más érdekes dolog is található ➲ http://www.salmar.com/marcel/.