Robert Meolic
meolic@uni-mb.si
zadnja sprememba: 20. 02. 2009

Zapiski predavanj 2

interno gradivo za predmet VSO, 2008/09

9. Naloge sistemskega administratorja

Naloga sistemskega administratorja računalniškega sistema je poskrbeti, da bo računalniški sistem čimbolj optimalno deloval. Bolj podrobno lahko njegove naloge opredelimo takole:

- dodajanje/brisanje uporabniških računov,
- nadzor na sistemski viri,
- nameščanje popravkov in posodobitev operacijskega sistema ter programske opreme,
- dodajanje novih naprav v sistem,
- povezovanje naprav med seboj,
- varnostno shranjevanje podatkov,
- odkrivanje in popravljanje varnostnih lukenj v sistemu,
- in številne druge, med katerimi ni najmanj pomembna ta, da mora sistemski administrator poskrbeti, da njegovemu šefu deluje elektronska pošta :-)

Za večino od nalog sistemskega administratorja lahko rečemo, da gre za **vzdrževanje sistemske programske opreme**, bodisi neposredno (npr. posodabljanje operacijskega sistema), bodisi posredno (skrb za programsko podporo raznim napravam).

Vse naštete praktične naloge so pomembne, je pa najtežje pri sistemski administraciji to, da pravilno reagiramo v kritičnih situacijah, ko pride do nenamerne ali celo namerne napačne uporabe / zlorabe našega sistema. Zato so za vsakega sistemskega administratorja poleg znanja pomembne tudi izkušnje.

Poleg skrbi za računalniški sistem je naloga sistemskega administratorja tudi podpora in pomoč uporabnikom tega sistema. Zaradi tega mora sistemski administrator tudi:

- vedeti, kaj in kako uporabniki uporabljajo računalniški sistem,
- seznaniti uporabnike z novimi programi oz. storitvami in z njihovo pravilno uporabo,
- znati od uporabnikov pridobiti ustrezne informacije o problemih.

Med naštetimi je eno najbolj "tečnih" opravil prav gotovo izdelovanje varnostnih kopij in restavriranje izgubljenih podatkov. Uporabniki pogosto "po nesreči" zbrišejo uporabne podatke in jih potem želijo imeti takoj nazaj. Hkrati pa so tudi okvare diskov bolj ali manj pričakovan pojav. Zato bo dober sistemski administrator temu delu posvetil še posebno skrb in če ga restavriranje zbrisanih podatkov ne bo preveč obremenjevalo, potem bo tudi vse ostalo lažje.

10. Upravljanje operacijskega sistema MS Windows iz lupine Windows Command

10.1 Nadzor na uporabniki

Za dodajanje, spreminjanje in brisanje uporabnikov je v sistemu MS Windows na voljo ukaz `NET USER`. Za spreminjanje parametrov v zvezi z gesli posameznih uporabnikov, pa imamo ukaz `NET ACCOUNTS`. Tukaj je nekaj primerov:

Dodajanje uporabnika:
```
NET USER username [password] /ADD
```

Brisanje uporabnika:
```
NET USER username [password] /DELETE
```

Spreminjanje gesla uporabnika:
```
NET USER username password
```

Nastavitev, da morajo gesla vsebovati vsaj šest znakov in da potečejo vsakih 7 dni:
```
NET ACCOUNTS /MINPWLEN:6 /MAXPWAGE:7
```

Nastavitev, da so lahko gesla poljubno kratka in da nikoli ne pretečejo:
```
NET ACCOUNTS /MINPWLEN:0 /MAXPWAGE:UNLIMITED
```

Ukaza `NET USER` in `NET ACCOUNTS` brez parametrov lahko izvedejo vsi uporabniki, pri čemer se le izpišejo podatki o tem, kateri uporabniki so na sistemu in kakšna je politika v zvezi z gesli.

V sistemu MS Windows je na voljo tudi grafično okolje, v katerem lahko podrobno nastavimo zaščite kot je npr. politika v zvezi z gesli. To okolje enostavno poženemo z ukazom `GPEDIT.MSC`

Pravilno je, da ima administrator računalnika na sistemu tudi uporabniško ime brez administratorskih pravic in da opravila kot je npr. brskanje po spletu opravlja brez administratorskih pravic. Nekoliko nadležno je le, če je potrebno na hitro izvesti nek ukaz ali program kot administrator. Pomagamo si lahko z ukazom `RUNAS`, s katerim določen program ali ukaz izvedemo na tak način, kot bi ga zagnal nek drug uporabnik. Ukaz `RUNAS` je zelo koristen tudi takrat, če želimo preizkusiti, ali bo neka skripta pravilno delovala za določenega uporabnika, ne želimo pa se prijaviti v njegov račun. Tukaj je nekaj primerov (seveda bodo ukazi delovali le, če bomo na poziv vnesli pravilno geslo za administratorja):

Nastavimo geslo za uporabnika uporabnik na novo_geslo:
```
RUNAS /USER:Admin "NET USER uporabnik novo_geslo"
```

Zbrišemo geslo za uporabnika uporabnik:
```
RUNAS /USER:Admin "NET USER uporabnik \"\""
```

Sprožimo postopek spreminjanja gesla za uporabnik, računalnik bo vprašal za novo geslo:
```
RUNAS /USER:Admin "NET USER uporabnik *"
```

10.2. Nadzor nad procesi

Operacijski sistemi za vsak proces vodijo številne podatke, med katerimi je najpomembnejši njegova enolična oznaka PID.

Procese v operacijskem sistemu MS Windows nadzorujemo z ukazom TASKLIST. Če ne podamo nobenih kretnic se izpiše seznam vseh procesov. Uporabne so npr. naslednje kombinacije kretnic:

Izpiše podrobnosti o procesih:
```
TASKLIST /V
```

Izpiše vse procese določenega uporabnika:
```
TASKLIST /FI "USERNAME eq uporabnik"
```

Izpiše vse procese in za vsakega vse storitve:
```
TASKLIST /SVC
```

Izpiše vse procese in za vsakega vse knjižnice DLL, ki jih ta proces uporablja (dolg izpis!):
```
TASKLIST /M
```

Izpiše vse procese, ki uporabljajo knjižnice DLL, katerih ime ustreza vzorcu GDI*, npr. GDI32.DLL:
```
TASKLIST /M GDI*
```

S kretnico /FI podamo filter. Poleg zgoraj prikazanega filtra, ki izpiše procese določenega uporabnika, lahko uporabimo še številne druge. Tukaj je nekaj zanimivih (v oklepaju so možne vrednosti):

STATUS („Running“, „Not Responding“) - poiščemo procese, ki se ne odzivajo,
PID (številka) – poiščemo procese glede na številko PID,
IMAGENAME (ime datoteke EXE) – poiščemo procese glede na ime programa,
MODULES (ime datoteke DLL) – poiščemo procesem, ki uporabljajo določen DLL

S kretnico /FO LIST lahko namesto izpisa v obiki tabele dobimo izpis v obliki seznama, s kretnico /FO CSV pa v obliki CSV, ki je zelo primerna za obdelavo v skriptah. Pri tem pride prav tudi kretnica /NH, s katero se odpovemo prvi vrstici, v kateri je glava tabele.

V nujnih primerih lahko določene procese tudi na hitro končamo (ubijemo). Uporabimo ukaz TASKKILL. Procese lahko končamo na prijazen način (enako ukazu exit v programu) ali pa nasilno. Običajno s kretnico /T zahtevamo, da se končajo tudi vsi njegovi procesi otroki.

Naenkrat lahko končamo več procesov, ki jih določimo po imenu (kretnica /IM) ali po številki PID (kretnica /PID). Uporabimo pa lahko tudi vse filtre naštete pri ukazu TASKLIST.

Končaj vse procese, v katerih teče notepad.exe
```
TASKKILL /IM notepad.exe
```

Končaj vse procese, v katerih teče `notepad.exe`, na nasilen način:
`TASKKILL /F /IM notepad.exe`

Končaj procesa s številko PID 1001 in 1003 ter vse njune procese otroke:
`TASKKILL /T /PID 1001 /PID 1003`

Opravila v računalniku lahko razporedimo tako, da se zaženejo ob določenem času in datumu. V sistemu MS Windows sta na voljo dva mehanizma. Starejši in preprostejši je z uporabo ukaza AT.

Tukaj je nekaj primerov:

```
AT 24:00 Defrager.bat
AT 6:00 /EVERY:T DiskClean.bat
AT 7:00 /EVERY:M,W,F MapNtwkDrive.bat
```

Modernejši pristop je uporaba ukaza `SCHTASKS`, ki ima naslednje možnosti:
`/CREATE` – tvori novo opravilo, ne smemo uporabiti za spreminjanje obstoječih opravil
`/CHANGE` – spremeni obstoječe opravilo
`/DELETE` – zbriši opravilo iz seznama
`/QUERY` – izpiše podatke o opravila v seznamu
`/RUN` – požene opravilo, tudi če še ni napočil zahtevan čas/datum
`/END` – prekine izvajanje opravila (paziti je potrebno, da ne zgubimo kakšne podatke)
Pri tvorjenju novega opravila so uporabne naslednje kretnice:
`/RU` – podamo ime uporabnika, s katerimi pravicami se bo opravilo izvedlo
`/RP` – podamo geslo (potrebno le, če smo uporabili /U)
`/SC` – frekvenca izvajanja, navedemo lahko `MINUTE, HOURLY, DAILY, WEEKLY, MONTHLY, ONCE, ONSTART, ONLOGON, ONIDLE`
`/D` – dan v tednu, navedemo lahko `MON, TUE, WED, THU, FRI, SAT, SUN`
`/M` – mesec v letu, navedemo lahko `JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC`
`/ST` – čas, ko se naj opravilo požene
`/TN` – ime oz. komentar opravila
`/TR` – program oz. skripta, ki se naj požene

Primer, kako določimo, da se skripta `DiskClean.bat` požene vsak petek ob 18.00:

```
SchTasks /Create /RU SYSTEM /SC WEEKLY /D FRI /ST 18:00:00 /TN "Disk Cleaning" /TR "C:\DiskClean.bat"
```

10.3. Nadzor nad sistemom

Ena od osnovnih nastavitev na vsakem računalniškem sistemu sta datum in čas. Lupina Windows command ima v ta namen vgrajena ukaza `DATE` in `TIME`. Ko ju izvedemo, se izpiše trenutni datum oz. čas, računalnik pa nas pozove, da vnesemo morebitno spremembo. Če želimo samo izpis, uporabimo kretnico `/T`. Datum in čas lahko spremeni le administrator.

Prav tako osnovna operacija je ugašanje računalnika. V lupini Windows command to storimo z ukazom SHUTDOWN, ki ga lahko izvede le administrator in ki ima lahko nekaj kretnic:

Uporabnik se odjavi (to lahko izvedemo brez administratorskih pravic)
```
SHUTDOWN /L
```

Ugasnemo računalnik:
```
SHUTDOWN /S
```

Zaustavitev in ponovni zagon računalnika (kretnica /R) čez 5 minut (300 sekund), odprte aplikacije bodo prekinjene brez opozorila (kretnica /F), vsi uporabniki bodo dobili podano obvestilo:
```
SHUTDOWN /R /C "Zaustavitev sistema čez 5 minut!" -F -T 300
```

Osnovne podatke o sistemu dobimo z ukazom SYSTEMINFO, bolj podrobne podatke pa z ukazom MSINFO. Oba ukaza lahko izvede navadni uporabnik. Program MSINFO se nahaja v posebni mapi, zato moramo za njegovo izvajanje napisati celotno pot in sicer:

```
C:\Program Files\Common Files\Microsoft Shared\MSInfo\msinfo32
```

Če pri ukazu MSINFO podamo parameter, ki mora biti ime datoteke, potem se ne odpre grafični vmesnik ampak se podatki zapišejo v tekstovno datoteko. Tvorjenje te datoteke traja precej dolgo (10 minut), dobimo pa res veliko podatkov. S kretnico /CATEGORY se omejimo le na podatke o določeni kategoriji npr. ComponentsPrinting, če nas zanimajo podatki o tiskalnikih. Imena kategorij izvemo, če grafični vmesnik zaženemo s kretnico /SHOWCATEGORY.

Podatke o odprtih datotekah vseh uporabnikov dobimo z ukazom OPENFILES ali pa z grafičnim orodjem Shared Folders iz paketa Microsoft Management Console, ki ga enostavno poženemo z ukazom FSMGMT.MSC. Da bi ukaz OPENFILES deloval, mora biti vključeno sledenje odprtim datotekam, kar naredimo z ukazom OPENFILES /LOCAL ON in ponovnim zagonom računalnika.
Podatke o naloženih gonilnikih dobimo z ukazom DRIVERQUERY. Zanimiva kretnica je /SI, ki za vsak uradno podprt gonilnik pokaže proizvajalca.

Podatke o sistemu dobimo tudi z ukazom NET.

NET CONFIG – ime računalnika, domena in drugi osnovni podatki
NET STATISTICS – statistika o količini poslanih in prejetih podatkov
NET VIEW – seznam mrežnih pogonov, na katere se lahko priključimo

Operacijski sistem Microsoft Windows precejšnjo pozornost namenja končnici v imenu datoteke. Končnica je niz znakov za zadnjo piko v imenu. Z ukazoma ASSOC in FTYPE izpišemo oz. spremenimo podatek o programu, ki je prirejen določeni končnici. Ukaz ASSOC za podano končnico izpiše oz. spremeni ime določila, ki določa program, ukaz FTYPE pa za vsako določilo izpiše ukazno vrstico, ki se bo izvedla.

Na primer:

```
C:\>ASSOC .bat
.bat=batfile

C:\>FTYPE batfile
batfile="%1" %*
```

Sistem MS Windows veliko nastavitev (tudi tiste o končnicah) hrani v posebni datoteki, ki ji rečemo **register**. Register je hierarhična zbirka podatkov in ima podatke razvrščene v 5 razredov imenovanih ključi in sicer:

- HKEY_CLASSES_ROOT
  To je podključ ključa HKEY_LOCAL_MACHINE\Software. Informacije, shranjene v tem ključu, zagotavljajo, da se odpre pravilen program, ko odprete datoteko z Raziskovalcem. V novejših sistemih so te informacije shranjene v ključih HKEY_LOCAL_MACHINE (privzete nastavitve, ki lahko veljajo za vse uporabnike v lokalnem računalniku) in HKEY_CURRENT_USER (nastavitve, ki preglasijo privzete nastavitve in veljajo samo za interaktivne uporabnike). Ključ HKEY_CLASSES_ROOT tako ponuja le pogled na združene informacije iz teh dveh virov in je namenjen predvsem programom, ki so oblikovani za starejše različice sistema pred Windows Server 2000. Vse spremembe je potrebno opraviti v HKEY_LOCAL_MACHINE in HKEY_CURRENT_USER. Kratica za ta ključ je HKCR.
- HKEY_CURRENT_USER
  Vsebuje koren informacij o konfiguraciji za trenutno prijavljenega uporabnika. V tem ključu so shranjene uporabnikove mape, barve zaslona in nastavitve nadzorne plošče. Informacije so povezane z uporabnikovim profilom. Kratica za ta ključ je HKCU.
- HKEY_LOCAL_MACHINE
  Vsebuje informacije o konfiguraciji, ki veljajo za računalnik (za vse uporabnike). Kratica za ta ključ je HKLM.
- HKEY_USERS
  Vsebuje vse uporabniške profile v računalniku. HKEY_CURRENT_USER je podključ ključa HKEY_USERS. Kratica za ta ključ je HKU.
- HKEY_CURRENT_CONFIG
  Vsebuje informacije o profilu strojne opreme, ki ga uporablja lokalni računalnik ob zagone sistema. Kratica za ta ključ je HKCC.

V registru so informacije, na katere se operacijski sistem Windows med delovanjem nenehno sklicuje, na primer profili za posamezne uporabnike, programi, nameščeni v računalniku, vrste dokumentov, ki jih vsak lahko ustvari, nastavitve lista z lastnostmi ta mape in ikone programov, katera strojna oprema je nameščena v sistemu, ter vrata, uporabljena v sistemu.

Za iskanje po registru in spreminjanje nastavitev imamo na voljo grafično orodje REGEDIT.EXE. Za brskanje po registru pa lahko uporabimo tudi ukaz REG QUERY. Podati pa moramo tudi odsek, v katerem iščemo podatke, vsako ime odseka se začne s kratico ključa. Na sistemu Windows Server 2003 lahko ukazu REG QUERY kot parameter podamo iskani niz znakov. Ta opcija iskanje raznih vrednosti v registru precej poenostavi.

11. Upravljanje operacijskega sistema Linux iz lupine Bash

11.1 Nadzor na uporabniki

Novega uporabnika lahko doda samo `root` tako, da uporabi ukaz `useradd`, ki se nahaja v imeniku `/usr/sbin`. Osnovna oblika ukaza je naslednja:

```
/usr/sbin/useradd -c "Ime Priimek" uporabnisko_ime
```

Ukaz `useradd` ima več parametrov, ki pa jih lahko izpustimo in se nastavijo privzete vrednosti. Privzete vrednosti izpišemo z ukazom `/usr/sbin/useradd -D`.

Ko tvorimo uporabnika, njegov račun še ni pripravljen za delo. Nastaviti mu moramo tudi geslo. To storimo z ukazom `passwd`, ki ga lahko pozneje izvedejo tudi običajni uporabniki, če si želijo spremeniti geslo. Ker začetno geslo nastavimo kot `root`, moramo povedati, kateremu uporabniku nastavljamo geslo.:

```
passwd uporabnisko_ime
```

Ko izvedemo ukaz `passwd` kot `root` moramo dvakrat vnesti novo geslo in potem je nov uporabniški račun pripravljen za delo. Podatke o uporabniku spremenimo z ukazom `usermod`, zbrišemo pa ga z ukazom `userdel`. Novo skupino ustvarimo z ukazom `groupadd`, podatke o njej spremenimo z `groupmod`, zbrišemo pa jo z ukazom `groupdel`. Vsi ti ukazi se nahajajo v imeniku `/usr/sbin`.

Če želimo določen ukaz ali program pognati z administratorskimi pravicami, pa nismo prijavljeni kot `root`, uporabimo ukaz `sudo`. Primer:

```
sudo ukaz
```

Ukaz `sudo` uporabimo tudi takrat, če želimo ukaz ali program izvesti kot drug uporabnik, ki ni `root`. V tem primeru moramo navesti uporabniško ime.

```
sudo -u uporabnik ukaz
```

11.2. Nadzor nad procesi

Tudi Linux za vsak proces vodi številne podatke, med katerimi je najpomembnejši njegova enolična oznaka PID.

Podatki o procesih so na Linuxu na voljo v obliki tekstovnih datotek v navideznem datotečnem sistemu `/proc`. Na voljo pa imamo tudi veliko ukazov. Ukaz `top` izpiše statistiko o trenutnem stanju sistema in našteje procese, ki trenutno najbolj obremenjujejo procesor. Če nas zanima le poraba pomnilnika, uporabimo ukaz `free`. Preprosto grafično predstavitev obremenjenosti sistema dobimo z ukazom `tload`.

Z ukazom `ps` izpišemo seznam procesov. Če ne podamo nobenih kretnic, se izpišejo samo podatki o procesih pognanih iz tiste lupine. Več procesov dobimo z uporabo kretnic.

Izpiši vse procese na sistemu:
```
ps -e
```

Izpiši procese določenega uporabnika:
```
ps -u uporabnik
```

Izpiše podrobne podatke o procesih določenega uporabnika:
```
ps -flu uporabnik
```

Vsak proces na sistemu ima določeno prioriteto, ki je številka med -20 in +19. Manjša številka pomeni, da je proces bolj pomemben. Prioriteto lahko procesu podamo že ob zagonu z ukazom `nice` ali pa mu jo spremenimo naknadno z ukazom `renice`.

Če je potrebno, lahko procese ubijemo z ukazom `killall` in `kill`. Razlika med njima je, da pri `killall` podamo ime procesa, pri `kill` pa njegovo številko PID. Proces ubijemo tako, da mu pošljemo ustrezen signal. Najmočnejši signal je SIGKILL, ki ima številko 9 in ki se mu procesi ne morejo izogniti. Tukaj sta dva primera:

```
killall firefox-bin
kill -9 10344
```

Za razporejanje ponavljajočih se opravil je na Linuxu na voljo več mehanizmov. Eden od razširjenih načinov je uporaba ukaza `crontab`. Opravila, ki jih želimo razporediti, opišemo v tekstovni datoteki v posebnem formatu in potem uporabimo ukaz:

```
$crontab ime-datoteke
```

Tabelo razporejenih opravil pogledamo z ukazom:

```
$crontab -l
```

V datoteki za crontab je lahko več opravil, vsako opravilo opišemo v eni vrstici.

Format vrstice je naslednji:

```
# Znak lojtra označuje vrstico s komentarjem
# +---------------- minute (0 - 59)
# |  +------------- ure (0 - 23)
# |  |  +---------- dan v mesecu (1 - 31)
# |  |  |  +------- mesec (1 - 12)
# |  |  |  |  +---- dan v tednu (0 - 7) (Sunday=0 or 7)
# |  |  |  |  |
# *  *  *  *  *  program oz. ukaz, ki se naj izvši
```

Če želimo našeti več vrednosti za posamezno polje, uporabimo vejico. Polja so med seboj ločena s presledki ali pa s tabulatorji.

Primer datoteke za crontab:

```
#==========================================================
#      SYSTEM ACTIVITY REPORTS
#  8am-5pm activity reports every 20 mins during weekdays.
#  activity reports every hour on Saturday and Sunday.
#  6pm-7am activity reports every hour during weekdays.
#  summary prepared at 18:05 every weekday.
#==========================================================
0,20,40 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3 &
0 * * * 0,6 /usr/lib/sa/sa1 &
0 18-7 * * 1-5 /usr/lib/sa/sa1 &
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 3600 -ubcwyaqvm &
```

11.3. Nadzor nad sistemom

Na Linuxu datum in čas na računalniku izpišemo ter spremimo z ukazom `date`. Na voljo imamo veliko število različnih formatov. Tukaj je nekaj najbolj uporabnih primerov:

`date` (izpiše: "tor feb 28 12:00:00 CET 2006")
`date +"%c"` (izpiše "tor 28 feb 2006 12:00:00 CET")
`date +"%d.%m.%Y"` (izpiše "28.02.2006")
`date +"%d %b %Y"` (izpiše "28 feb 2006")
`date +"%A, %d. %B %Y"` (izpiše "torek, 28. februar 2006")
`date +"%T"` (izpiše "12:00:00")
`date +"%Y-%m-%d %H:%M:%S %z"` (izpiše: "2006-02-28 12:00:00 +0100")

Ukaz, s katerim ugasnemo računalnik, se tudi v Linuxu imenuje `shutdown`. Vendar pa se nahaja v imeniku `/sbin` in ga zato lahko izvede le `root`. Ukaza `shutdown` ne moremo uporabiti za odjavo.

Takoj ugasni računalnik:
```
/sbin/shutdown -h now
```

Ugasni in ponovno zaženi računalnik čez 5 minut:
```
/sbin/shutdown -r -t 300 now "Ponovni zagon racunalnika cez 5 minut"
```

Za poizvedovanje o sistemu in uporabnikih so na voljo številni ukazi, na primer:

- `uname` - izpiše osnovne podatke o sistemu, celoten izpis dobimo s kretnico `-a`
- `uptime` - izpiše, koliko časa je računalnik že prižgan,
- `hostname` - izpiše ime računalnika,
- `whoami` - izpiše naše uporabniško ime,
- `id` - izpiše nekaj podatkov o našem računu, med njimi naše uporabniško ime in ime skupine,
- `w` - izpiše podatke o prijavljenih uporabnikih računalnika (kaj delajo),
- `last` – izpiše, kdo vse je bil na računalnik prijavljen v preteklosti (`/var/log/wtmp`)

12. Nameščanje in posodabljanje programske opreme

Programsko opremo lahko na računalnik namestimo na različne načine:

- prenesemo izvorno kodo novega programa, ga prevedemo in namestimo,
- prenesemo izvršilno kodo novega programa in jo namestimo na računalnik,
- povežemo se z bazo podatkov, ki vsebuje programe za naš sistem, in s pomočjo posebnega orodja za nameščanje prenesemo ter namestimo željen program.

Posodabljanje programov je na videz podobno nameščanju novih programov, a je precej bolj kočljiva zadeva. Pomisliti moramo vsaj na dvoje:

- ali ni kateri drugi program odvisen od trenutno nameščene različice programa in
- ali bo nova različica programa znala pravilno obdelati podatke shranjene s trenutno različico programa.

Za posodabljanje programov imamo na voljo enake možnosti kot pri nameščanju novih, pri čemer je najbolj zanesljiva zadnja možnost. Orodja za nameščanje namreč znajo preveriti odvisnosti med programi in tako preprečijo, da bi na računalnik namestili nezdružljive različice.

Nameščanje in posodabljanje programov s pomočjo sistema za nameščanje je običajno na voljo le administratorju sistema oz. le tistim, ki imajo ustrezne pravice. Posebnost operacijskega sistema MS Windows je to, da se orodje za nameščanje sproži vedno, ko poženemo datoteko SETUP.EXE. Zato take datoteke navadni uporabnik ne more pognati, ne glede na to, kaj piše v njej. Če je program takšen, da pri namestitvi ne potrebuje dostopa do sistema (program se v celoti namesti na uporabniško področje diska), se lahko znajdemo tako, da datoteko SETUP.EXE preimenujemo v nekaj drugega in potem jo lahko izvedemo.

Za posodabljanje programov, ki jih dobimo skupaj z operacijskim sistemom MS Windows uporabljamo spletni strani:

- **Windows Update** (http://update.microsoft.com/windowsupdate/) in
- **Microsoft Update** (http://update.microsoft.com/microsoftupdate/) - ta vključuje tudi posodobitve za druge Microsoftove izdelke npr. MS Office.

Za lažje posodabljanje programov imamo na voljo orodje Automatic Updates, ki samodejno preveri, ali je na voljo nova različica oz. popravek katerega od programov.

Pri posodabljanju večih računalnikov si delo lahko olajšamo z uporabo strežnika **WSUS (Windows Server Update Services)**. To je namenski strežnik, ki vodi lokalno bazo novih različic in popravkov in jih potem posreduje drugim računalnikom v našem omrežju.

V operacijskih sistemih Windows Vista in Windows Server 2008 namesto spletnega programa uporabljamo dodatne možnosti programa Automatic Updates, ki zna naložiti popravke iz skladišča Windows Update. Če želimo, lahko to spremenimo tako, da bo uporabljal Microsoft Update.

Pri Microsoftu se v zadnjem času držijo prakse, da redne popravke izdajo vsak drugi torek v mesecu. Nujne varnostne popravke izdajo takoj, ko se pojavi potreba. Od časa do časa izdajo večji komplet popravkov, ki vključuje bistvene izboljšave sistema in se imenuje Service Pack. Popravke lahko namestimo le na legalno pridobljene sisteme.

V svetu obstajajo razne institucije, ki se trudijo sproti odkrivati in raziskovati varnostne probleme računalniških sistemov. Za operacijski sistem Windows bomo zelo na tekočem, če bomo npr. redno sledili obvestilom US-CERT, ki je ameriška državna institucija (http://www.us-cert.gov/). Neposredna povezava do obvestil je http://www.us-cert.gov/cas/alerts/ .

Pri operacijskem sistemu Linux nimamo enega centralnega mesta, kjer bi bili shranjeni novi in posodobljeni programi, ampak se le-ti nahajajo v različnih zbirkah vsepovsod po svetu. Poznamo tudi različne formate teh zbirk. Najbolj znana sta formata **rpm** in **deb**. Da bi prenesli programe iz teh zbirk imamo na voljo posebna orodja:

- za obdelovanje datotek rpm imamo na voljo orodje z istim imenom **rpm,** za iskanje in prenašanje iz zbirk formata rpm pa sta na voljo orodji **apt** in **yum,**
- za obdelovanje datotek deb imamo na voljo orodje **dpkg,** za iskanje in prenašanje iz zbirk formata deb pa je na voljo orodje **apt.**

Z orodjem **apt** lahko torej delamo z obema formatoma. Njegova omejitev je, da omogoča le nameščanje programov iz oddaljenih zbirk. Če imamo datoteko rpm že na svojem disku, uporabimo neposredno orodje **rpm** s kretnico -i (install) ali -U (upgrade). Če imamo datoteko deb na svojem disku, uporabimo neposredno orodje **dpkg** s kretnico -i, še rajši pa uporabimo preprosto orodje **gdebi** oz. nekoliko bolj kompleksno orodje **dselect**.

Vsi našteti programi imajo tekstovni vmesnik. Oglejmo si najprej nekaj zanimivih kretnic orodja rpm.

Izpiši podatke o paketu `bash`, če je ta nameščen:
```
rpm -q bash
```

Izpiši podatke o paketu `bash` v obliki ime-verzija-izdaja, na koncu dodaj znak za novo vrstico:
```
rpm -q --queryformat "%{NAME}-%{VERSION}-%{RELEASE}\n" bash
```
Izpiši podatke o vseh nameščenih paketih:
```
rpm -qa
```

Izpiši podatke o vseh nameščenih paketih, ki se začenjo s črko A:
```
rpm -qa a*
```

Izpiši podatke o paketih, ki za svoje delovanje potrebujejo paket `bash`
```
rpm -q --whatrequires bash
```

Paket bash odstranimo iz sistema z ukazom:
```
rpm -e bash
```

Če želimo testirati brisanje paketa bash (v resnici se ne bo nič zbrisalo), uporabimo ukaz:
```
rpm -e --test bash
```

Brisanje paketov zahteva dovoljenje za pisanje v sistemske imenike, zato je dovoljeno le administratorju. Pri testiranju brisanja se nič ne spremeni na sistemu, zato to lahko izvedejo tudi navadni uporabniki.

Testiranje brisanja je zelo uporabna možnost. S kretnico `--whatrequires` dobimo spisek vseh paketov, ki imajo navedeno, da določen paket potrebujejo. Vendar pa nekateri paketi potrebujejo kakšno knjižnico `*.so`, ki sicer je del nekega paketa (vse kar je nameščeno, je del nekega paketa!), vendar pa ni navedeno, v katerem paketu je ta knjižnica (to bi lahko šteli kot napako pri tvorjenju paketov, ki pa je žal zelo pogosta). Pri brisanju se za vse datoteke določenega paketa preveri, ali se smejo zbrisati in šele takrat računalnik preveri, ali ni slučajno kateri drugi paket odvisen od njegovih knjižnic `*.so`. Da bi dobili realno sliko, katere pakete res lahko zbrišemo iz sistema, moramo zato namesto kretnice `--whatrequires` uporabiti kar poskus brisanja.

Oglejmo si še osnovno uporabo apt. Paket k sebi prenesemo in namestimo z ukazom apt-get:

```
# apt-get install mozilla
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  libnss3 mozilla-mailnews mozilla-psm
Suggested packages:
  mozilla-chatzilla
Recommended packages:
  myspell-en-us myspell-dictionary
The following NEW packages will be installed:
  libnss3 mozilla mozilla-mailnews mozilla-psm
0 upgraded, 4 newly installed, 0 to remove and 3 not upgraded.
Need to get 2658kB of archives.
After unpacking 7991kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Orodji apt in yum imata tudi številne tekstovne in grafične vmesnike, za apt npr. obstaja dober tekstovni vmesnik **aptitude,** uporabimo lahko tudi že prej omenjeni **dselect,** še najbolj pa sta prijazna grafična vmesnika **Synaptic** in **KPackage**. Synaptic ima popolnoma enako funkcionalnost kot ukaz apt-get, le da ponuja lep grafični vmesnik. KPackage je bolj splošen, saj zna delati tudi neposredno s programoma rpm in dpkg, torej ne potrebuje nujno apt.

Za yum je več dobrih grafičnih vmesnikov, npr. **Yum Extender**, **KYum, Pup** in **Pirut**.

13. Arhiviranje podatkov

Pri izdelavi varnostnih kopij ločimo tri korake:

- množico datotek, za katere želimo narediti varnostno kopijo, združimo skupaj in hkrati stisnemo, da v podatkovnem skladišču zasedejo manj prostora,
- pripravljene podatke shranimo v podatkovnem skladišču,
- ob morebitni potrebi podatke obnovimo.

Ločimo več vrst varnostnih kopij:

- polna (full backup)
- diferenčna (differential backup)
- inkrementalna (incremental backup)

V operacijskem sistemu Linux imamo za združevanje datotek in za stiskanje datotek na voljo različna orodja, med njimi so najbolj pogosto uporabljena orodja `tar` (za združevanje) ter `bzip` in `gzip` (za stiskanje).

Tukaj je primer ukaza, ki podatke združi in hkrati stisne z orodjem `bzip`:
```
tar -cjpv -f arhiv.tar -X nobackup.txt imenik
```

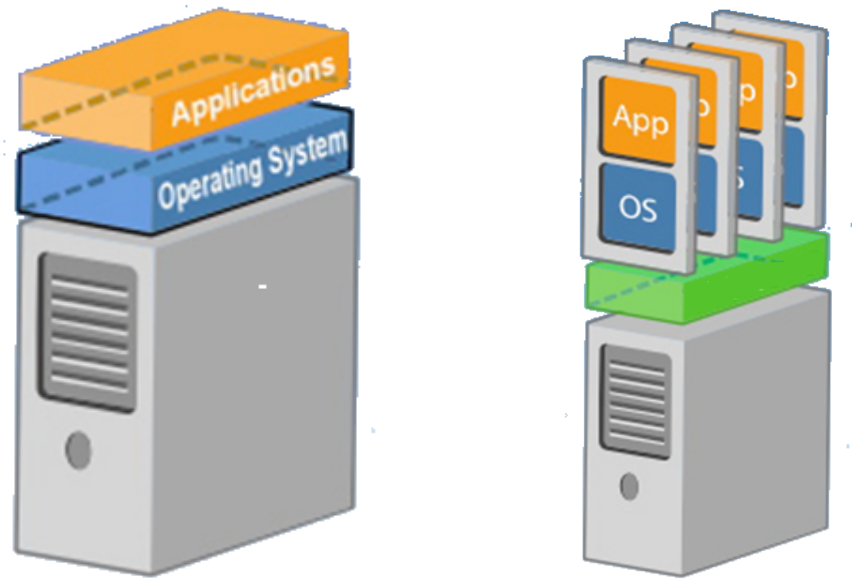
Če uporabimo kretnico `-g`, potem ukaz `tar` poleg arhiva tvori tudi evidenco o arhivu. Če evidenca s podanim imenom že obstaja, potem `tar` s kretnico `-g` tvori diferenčni arhiv (v izhodno datoteko se shranijo le spremenjeni podatki).

Primer:
```
tar -cjpv -g evidenca -f arhiv.tar imenik
```

Za obnovitev datotek in imenikov uporabimo ukaz `tar` z naslednjimi kretnicami:
```
tar -xjpv -f arhiv.tar datoteke_in_imeniki
```

14. Virtualizacija računalniških sistemov

Virtualizacija je sposobnost enega fizičnega sistema, da se obnaša kot multipliciran fizični sistem. Med drugim nam omogoča sočasno izvajanje več operacijskih sistemov na gostiteljskem računalniku. Z večanjem zmogljivosti posameznih sistemov postaja virtualizacija vedno bolj popularna, saj omogoča zelo učinkovito izrabo strojnih virov, kar znižuje stroške. Hkrati zagotavlja večjo varnost, ki je posledica delovanja ključnih aplikacij v ločenih računalniških sistemih.



Zeleno obarvana komponenta, ki jo vstavimo v sistem, se imenuje VMM (V*irtual Machine Monitor)* ali pa tudi *hypervisor.* VMM je lahko:

- programski modul, ki teče neposredno na strojni opremi gostitelja (ta rešitev je prikazana na sliki)
- programski modul, ki teče v okolju določenega operacijskega sistema (to rešitev prikažemo tako, da na sliki pod zeleno komponento dodamo še eno modro).

Obstaja veliko virtualizacijskih tehnologij, zelo znane so naslednje:

- Xen (Citrix Systems, GPL, http://www.xen.org/), teče na Linuxu in nekaterih drugih sistemih, ne pa na MS Windows,
- WMware Server (WMware Inc., komericalni produkt, http://www.vmware.com/products/ server/), teče na Linuxu in MS Windows,
- VirtualPC (Microsoft, prosto dostopno, http://www.microsoft.com/windows/products/ winfamily/virtualpc/), teče na MS Windows,
- QEMU (odprtokodni projekt, GPL, http://bellard.org/qemu/), teče na Linuxu in MS Windows,
- VirtualBox (Sun Microsystems, prosto dostopno, http://www.virtualbox.org/), teče na Linuxu in MS Windows,
- Oracle VM (Oracle Corporation, prosto dostopno, http://www.oracle.com/technologies/ virtualization/), teče neposredno na strojni opremi, uporablja podobne rešitve kot Xen,
- WMware ESX (WMware Inc., komericalni produkt, http://www.vmware.com/products/vi/ esx/), teče neposredno na strojni opremi.

Ideja virtualizacije pravzaprav ni nova. Že leta 1974 sta Gerald J. Popek in Robert P. Goldberg v članku z naslovom "Formal Requirements for Virtualizable Third Generation Architectures" opisala osnovne zahteve, ki jih mora izpolnjevati nek sistem, da lahko na njem uspešno emuliramo druge sisteme. Virtualizacijo sta definirala kot uspešno, če:

- se emulirani sistem obnaša popolnoma enako kot takrat, ko teče neposredno na strojni opremi,
- VMM lahko popolnoma nadzira vse vire (strojno opremo), ki jo uporablja emulirani sistem,
- večina strojnih instrukcij emuliranega sistema se mora izvesti brez posredovanja VMM (drugače rešitev ni učinkovita).

Danes vsi VMM izpolnjujejo prva dva pogoja, glede učinkovitosti pa se med seboj precej razlikujejo. V članku sta tudi pokazala, da je sposobnost uspešne virtualizacije odvisna od strojnega jezika procesorja. Problem je, če obstajajo občutljive instrukcije, ki niso privilegirane instrukcije. Pri tem so kot občutljive označene tiste, ki vplivajo na pravilnost delovanja VMM, privilegirane pa so tiste, ki jih noben program ne more uporabiti mimo omejitev, ki jih definira nek poseben sistemski program (bolj preprosto povedano za tiste, ki poznate strojne jezike, privilegirane instrukcije so tiste, ki jih lahko prestrežejo s pastmi – traps). Zanimivo je, da nekateri stari računalniki, kot npr. IBM System/370 izpolnjujejo ta pogoj, Intelovi procesorji i386 pa imajo v svoji zasnovi številne napake.

Intel v nekatere novejše procesorje (a ne vse!) vgrajuje tehnologijo Intel VT-x, ki zakrpa vse luknje glede virtualizacije. Procesorji, ki imajo VT-x so npr. Pentium 4 model 662 in 672, Pentium D vsi

modeli 920 – 960 razen 915, 925, 935 in 945, nekateri Core 2 procesorji itd., te tehnologije pa nimajo procesorji Celeron, Pentium M, pa tudi ne procesor Atom.

Tudi AMD v svoje novejše procesorje vgrajuje podobno tehnologijo pod imenom AMD-V. Ta tehnologija je vgrajena le v 64-bitne procesorje, npr. v Athlon 64, Athlon 64 X2, Athlon 64 FX, Turion 64 X2, Opteron od 2. generacije naprej, Phenom itd. Te tehnologije pa nimajo procesorji Sempron.

Intel VT-x in AMD-V sta pomembni tehnologiji zato, ker brez njiju npr. operacijskega sistema MS Windows ne moremo emulirati v okolju Xen.

15. Izvajalna okolja

Izvajalna okolja so posebna vrsta virtualizacije, pri katerih je cilj le emulirati izvajanje določenega programa. V osnovi je to lažje od splošne virtualizacje, a ker so zahteve izvajalnih okolij drugačne (tukaj je učinkovitost zelo pomembna), so rešitve ravno tako kompleksne.

Izvajalna okolja imenujemo tudi „process virtual machines“ ali „application virtual machines“. Za razliko od tega, splošne tehnologije za virtualizacijo imenujemo „system virtual machines“ ali „hardware virtual machines“.

Tradicionalno računalništvo temelji na programski opremi, katere življenjski cikel sestavljajo:

- **izvorna koda** – napisana v višjem programskem jeziku, razumljiva za človeka, v grobem neodvisna od procesorja in operacijskega sistema,
- **objektna koda** – to je strojni jezik, ponazorimo ga z zbirnim jezikom (assembler-jem), namenjeno točno določeni vrsti procesorjev, v grobem neodvisna od operacijskega sistema,
- **izvršilna koda** – tudi to je strojni jezik, dobimo jo tako, da združimo objektno kodo programa in objektno kodo knjižnic operacijskega sistema, izvajamo jo lahko samo na točno določeni vrsti procesorjev in točno določeni vrsti operacijskega sistema.

Izvršilna koda (strojni jezik) neposredno izvaja procesor tako, da zaporedoma:

- naloži eno inštrukcijo izvršilne kode,
- dekodira inštrukcijo in po potrebi naloži potrebne podatke iz pomnilnika,
- izvede inštrukcijo in po potrebi napiše rezultat nazaj v pomnilnik.

Največji problem tradicionalnih programov je nadzor na pomnilnikom. Program lahko pomotoma ali nalašč piše v tisti del pomnilnika, ki je rezerviran za sistem oz. ki je prirejen drugemu uporabniku. Moderni operacijski sistemi pisanja v tuj pomnilnik ne dovoljujejo in ob vsakem poskusu nasilno prekinejo izvajanje programa – to je napaka med izvajanjem in program se zruši!

S pojavom jave se je v računalništvu pojavil drugačni model razvoja in izvajanja programov. Pri razvoju programske opreme imamo naslednji, krajši cikel:

- **izvorna koda** - napisano v višjem programskem jeziku, razumljivo za človeka,
- **vmesna koda** – to je strojni jezik, podobno objektni kodi.

Vmesna koda se od navadne objektne kode precej razlikuje. Najpomembnejši razliki sta:

- pri vmesni kodi je uporabljen strojni jezik nekega navideznega procesorja in ne ciljnega sistema,
- vmesne kode ni potrebno povezovati v izvršilno kodo, ker je dovolj strukturirana, da se navzkrižna sklicevanja rešijo kar med izvajanjem.

Da bi izvedli program v javi, moramo na ciljni računalnik namestiti **javino izvajalno okolje** (JRE - JAVA Runtime Environment). Izvajanje programa poteka na naslednji način:

- JRE naloži eno inštrukcijo vmesne kode
- JRE dekodiraj inštrukcijo in naloži potrebne podatke iz pomnilnika
- JRE preveri ali je ta inštrukcija pravilna (ne bo povzročila napake med izvajanjem)
- JRE preveri ali se ta inštrukcija sme izvesti na ciljnem sistemu
- JRE pretvori eno inštrukcijo vmesne kode v eno oz. več instrukcij strojnega jezika za ciljni procesor
- procesor izvede dobljene instrukcije, kar vključuje tudi morebitno pisanje v pomnilnik

Kot vidimo iz zgornjega opisa, izvajanje programa ves čas nadzoruje JRE, pravimo, da teče program v nadzorovanem okolju. Če JRE dobro opravi svojo nalogo, potem:

- se program ne bo nikoli zrušil,
- program ne bo nikoli naredil nobene škode na računalniku,
- isti program, ki je že v prevedeni obliki, lahko izvajamo na različnih procesorjih in različnih operacijskih sistemih.

To so velike prednosti v primerjavi s tradicionalnimi programi. Slabost pa je v tem, da se programi izvajajo počasneje in da med izvajanjem potrebujemo kar nekaj pomnilnika za delovanje JRE. Ker je danes razvoj strojne opreme zelo hiter (procesorji so vedno hitrejši, pomnilnika je vedno več), naveden slabosti niso preveč hud problem.

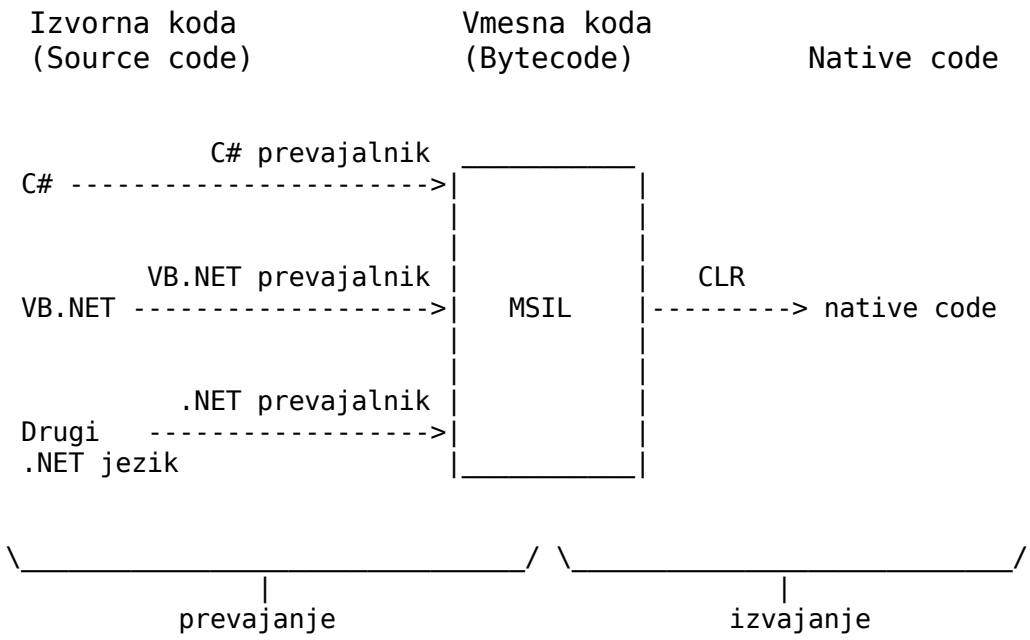
JRE je tipičen primer sistemske programske opreme, za katero skrbi administrator sistema.

JAVA je uvedla tudi novo vrsto programov, tako imenovane spletne programčke (JAVA Applets). Omejeno različico javinega izvajalnega okolja namreč lahko vgradimo kar v spletne brskalnike. Razvoj in izvajanje spletnih programčkov poteka na naslednji način:

- ponudnik napiše program v javi in ga z javinim prevajalnikom prevede v vmesno kodo,
- ponudnik naloži vmesno kodo programa na spletni strežnik,
- uporabnik prenese vmesno kodo na svoj računalnik in jo izvede s pomočjo omejenega JRE, ki je vgrajen (oz. dodan - plugin) v spletni brskalnik.

JRE je lahko torej omejen na neko podmnožico javinih razredov. Prednost majhnih JRE-jev, ki poznajo le osnovne javine razrede, je v tem, da so primerni za izvajanje v napravah z omejenimi viri (manj zmogljiv procesor, malo pomnilnika). Primer takih naprav so npr. mobilni telefoni, ki torej kljub omejenim virom lahko izvajajo tiste javine programe, ki uporabljajo le osnovne knjižnice.

Microsoft je na osnovi ideje jave ustvaril tako imenovano .NET tehnologijo. Prevajalniki, ki imajo končnico .NET, prevedejo programe v vmesno kodo imenovano MSIL (Microsoft Intermediate Language) in ne v objektno kodo za ciljni procesor. Programi se nato izvajajo v nadzorovanem okolju imenovanem CLR (Common Language Runtime), ki je vgrajeno v operacijski sistem oz. si ga namestimo posebej.



JRE in CLR sta v osnovi tolmača (interpreterja), saj program med izvajanjem sproti pretvarjata iz enega strojnega jezika v drugi strojni jezik. Ker pa v primerjavi s klasičnimi tolmači (npr. tistimi za skriptne jezike) vsebujeta številne napredne funkcije značilne za prevajalnike, jima damo oznako **prevajalnik tipa JIT** (just-in-time compiler).