Robert Meolic
meolic@uni-mb.si
zadnja sprememba: 14. 02. 2007

Zapiski predavanj 3

interno gradivo za predmet VSO, 2006/07

14. Oddaljena prijava preko interneta

V preteklosti je bil najbolj pogost način oddaljene prijave preko interneta storitev telnet. Ker pri telnetu ni poskrbljeno za ustrezno zaščito podatkov, je uporaba telneta že nekaj časa odsvetovana.

Na sistemih Unix se je v preteklosti uporabljala (in se še uporablja) storitev rsh. Tudi pri rsh ni ustrezno poskrbljeno za varnost, zato se je moramo izogibati.

Varen način oddaljene prijave je storitev ssh (Secure Shell). V začetku je bil na voljo le kot komercialni izdelek. Danes je za Linux je na voljo paket OpenSSH, za Microsoft Windows pa program putty, ki sta oba brezplačna in omogočata uporabo storitve ssh. Oglejmo si, kako ssh uporabljamo na Linuxu.

Na oddaljen računalnik `vso.ms.si` se prijavimo na naslednji način:

```
ssh vso.ms.si
```

Če se želimo prijaviti z drugačnim uporabniškim imenom, kot ga imamo na lokalnem računalniku, uporabimo kretnico `-l`, ki ji (običajno brez presledka) sledi uporabniško ime:

```
ssh -lUporabnik vso.ms.si
```

15. Prenos datotek preko interneta

V preteklosti je bil najbolj pogost način prenašanja datotek preko interneta storitev ftp. Ker pri ftp-ju ni poskrbljeno za ustrezno zaščito podatkov, se danes ftp uporablja le za t.i anonimni prenos datotek, ki ne vsebujejo zaupnih podatkov.

Na sistemih Unix se je v preteklosti uporabljala (in se še uporablja) storitev rcp. Tudi pri rcp ni ustrezno poskrbljeno za varnost, zato se je moramo izogibati.

Varen način prenosa datotek preko interneta sta storitvi sftp (Secure FTP) in scp (Secure Copy). Na Linuxu sta oba del paket OpenSSH, na operacijskem sistemu Microsoft Windows pa je na voljo program WinSCP, ki prav tako pozna obe storitvi.

Uporaba sftp-ja je podobna uporabi ftp-ja. Uporabljamo ukaze dir, cd, put, get, mput, mget itd.

Uporaba scp-ja je podobna ukazu cp. Datoteko Vaje.pdf prenesemo iz računalnika vss.ms.si k sebi z naslednjim ukazom:

```
scp vso.ms.si:imenik/Vaje.pdf .
```

Obratno, torej prenos na oddaljeni računalnik, pa izvedemo na naslednji način:

```
scp Vaje.pdf vso.ms.si:imenik/podimenik
```

Če se želimo datoteko prenesti iz uporabniškega računa z drugačnim uporabniškim imenom, kot ga imamo na lokalnem računalniku, potem uporabimo ukaz:

```
scp Uporabnik@vso.ms.si:imenik/Vaje.pdf .
```

16. Deljenje datotek v omrežju

Pri deljenju datotek omogočimo drugim uporabnikom v omrežju (lokalno ali pa kar preko interneta), da dostopajo do datotek shranjenih na lokalnem disku. Oddaljeni uporabniki lahko datoteke tudi spreminjajo.

Pogoj za deljenje datotek je vzpostavljena **omrežna povezava**. Omrežno povezavo lahko vzpostavimo na več načinov, a se danes skoraj izključno uporabljajo le **internetne povezave,** ki jih vzpostavimo s pomočjo protokolov **TCP/IP** ali **UDP/IP** (IP je omrežni protokol, TCP in UDP pa sta transportna protokola, za vzpostavitev omrežne povezave je potreben IP in eden od transportnih protokolov). Naj omenimo še dve drugi možnosti. V omrežjih **Netware** (ta omrežja so danes skoraj izumrla, bila so podobna današnjim intranetom, razvijalo jih je podjetje Novell) se je omrežna povezava tvorila z uporabo protokolov **IPX/SPX** (IPX je omrežni protokol podoben IP-ju, SPX pa transportni protokol podoben TCP-ju). V omrežjih **Microsoft Windows Network** se povezava tvori s pomočjo protokola **NetBIOS**. Za razliko od prej omenjenih pa to ni omrežni ali transportni protokol, ampak sejni protokol (peti sloj OSI modela). NetBIOS je na nižjih slojih originalno uporabljal protokol NetBEUI, ki je hkrati omrežni in transportni protokol, a precej manj zmogljiv od npr. kombinacije TCP/IP. Microsoft je pozneje NetBEUI kombiniral skupaj s TCP/IP in to kombinacijo imenoval NBT.

Ko imamo omrežno povezavo, potrebujemo še aplikacijski protokol, s pomočjo katerega se bodo računalniki dogovorili o tem, katere datoteke so deljene in kakšna je njihova zaščita. Poznamo več takih protokolov:

- **NFS** (Network File System) so naredili v podjetju Sun, danes pa skrbi za njega organizacija IETF. Deluje preko internetne povezave, uporabljamo pa ga lahko na sistemih MS Windows, Linux in drugih.

- **SMB** (Server Message Block) je Microsoftov protocol, ki se primarno uporablja na sistemih MS Windows. Deluje preko protokola NetBIOS. Ker pri deljenju datotek kompleksni sejni sloj, kot je NetBIOS, pravzaprav ni nujno potreben, so pri Microsoftu v sistem Windows 2000 in novejše vgradili različico protokola SMB, ki deluje neposredno nad internetno povezavo. Danes je protokol SMB dobro podprt tudi na sistemih Linux, orodje za delo z njim se imenuje Samba. Microsoft je konec 90-ih let želel protokol SMB nadgraditi v zmogljivejši protokol z imenom CIFS (Common Internet File System), vendar pa tega projekta niso dokončali, tako da je bil rezultat le novo ime in nekaj dodatne funkcionalnosti. Samba podpira tudi vso novo funkcionalnost iz projekta CIFS.

- **NCP** (NetWare Core Protocol) so naredili v podjetju Novell. Deluje s povezavami TCP/IP in tudi IPX/SPX. Uporabljamo ga lahko na sistemih MS Windows, Linux in drugih.

- **OpenAFS** je različica protokola AFS (Andrew file system), ki so si ga izmislili na univerzi CMU v ZDA, nato pa ga je razvijalo podjetje IBM. Od leta 2000 je to odprto-kodni projekt. Deluje preko internetne povezave in je podoben NFS. Tudi OpenAFS lahko uporabljamo na različnih sistemih, vključno z MS Windows in Linuxom.

- **AFP** (Apple Filing Protocol) so naredili v podjetju Apple. Uporablja internetne povezave in deluje v sistemu Mac OS.

- **WebDAV** (Web-based Distributed Authoring and Versioning) je zanimiv protokol, ki omogoča deljenje datotek preko vzpostavljene HTTP povezave. Razvija ga združenje IETF.

Aplikacijski protokoli za deljenje datotek (pravzaprav spadajo v šesti sloj OSI modela, to je predstavitveni sloj) so tipa **strežnik – odjemalec**. Torej moramo na tistem računalniku, na katerem se nahajajo datoteke, namestiti ustrezni strežnik. Za dostop do datotek ne potrebujemo strežnika, dovolj je le odjemalec.

16. 1. Deljenje datotek preko interneta po protokolu NFS (Network File System)

Deljenje datotek po protokolu NFS pogosto najdemo na komercialnih različicah Unix-a, npr. na Solarisu. Na računalniku, na katerem želimo dati datoteke v skupno rabo preko interneta, moramo namestiti strežnik za NFS.

Strežniki in odjemalci za NFS so na sistemih MS Windows manj razširjeni. S strani Microsofta lahko dobimo paket **Windows Services for UNIX**, ki vključuje NFS strežnik. Ta paket ne podpira sistema Windows XP Home Edition, je pa vključen v sistem MS Windows Vista. Na internetu lahko izbrskamo tudi nekatere komercialne produkte, npr. Allegro NFS Server, Omni NFS Server itd. Vsi našteti paketi vsebujejo tudi odjemalce. Med odjemalci je zanimiv tudi komercialni izdelek Reflection NFS Client.

Na sistemih Linux je protokol NFS dobro podprt in dokaj razširjen. Ko namestimo strežnik, imenike, ki so deljeni, zapišemo v datoteko `/etc/exports`. Seveda lahko to stori le administrator sistema.

Tukaj je preprost primer:

```
$ cat /etc/exports
/home/meolic/test              meolic.uni-mb-si(rw,sync)
/home/meolic/test1             meolic.uni-mb-si(ro,sync)
/home/meolic/test2             meolic(rw,sync) altair.uni-mb.si(ro,sync)
```

V vsaki vrstici podamo podatke o enem imeniku. Najprej napišemo celotno ime imenika, nato pa naštejemo računalnike, ki jim dovolimo dostop. Če naštejemo več računalnikov, jih ločimo s presledki. Za vsakega v oklepaju podamo ustrezne parametre. Nastavimo lahko številne parametre, najlažje je, če si pomagamo s kakšnim grafičnim vmesnikom.

Če smo datoteko `/etc/exports` tvorili brez grafičnega orodja, moramo uporabiti tudi ukaz `/usr/sbin/exportfs`, ki prebere datoteko `/etc/exports` in omogoči dostop navedenim računalnikom do navedenih imenikov.

Na računalniku, kjer je strežnik NFS, lahko spremljamo podatke o tem, kdo vse je trenutno priklopljen na deljene imenike. Podatki so zapisani v datoteki /var/lib/nfs/rmtab. Na primer:

```
$ cat /var/lib/nfs/rmtab
164.8.22.109:altair.uni-mb.si:0x00000002
altair.uni-mb.si:/home/meolic/test2:0x00000003
```

Če želimo do deljenih datotek dostopati iz Linuxa, uporabimo ukaz mount. Priklop je torej podoben priklopu lokalnih particij, le da pri kretnici -t, ki določa tip datotečnega sistema, napišemo `nfs`. Oblika ukaza je naslednja:

```
mount -t nfs racunalnik:/imenik/podimenik oddaljen/imenik
```

Primer:

```
mount -t nfs altair.uni-mb.si:/home/meolic/test2 mojtest
```

Podobno, kot pri priklopu lokalnih particij, lahko v datoteko `/etc/fstab` napišemo ustrezno vrstico in potem se bo oddaljeni imenik samodejno priklopil.

16.2. Deljenje datotek preko interneta po protokolu SMB (Server Message Block)

SMB je protokol, ki ga za deljenje datotek uporablja operacijski sistem MS Windows. Strežnik in odjemalec za SMB sta vgrajena v vse sisteme MS Windows. Imenike damo v skupno rabo s pomočjo raziskovalca ali pa iz lupine Windows command z ukazom NET SHARE. Imenik priklopimo prav tako s pomočjo raziskovalca ali pa iz lupine Windows command z ukazom `NET USE`.

Na Linuxu je protokol SMB podprt s paketom Samba. Uporaba Sambe je podobna uporabi NFS, le da ima več funkcionalnosti.

Imenike, ki so deljeni, zapišemo v datoteko `/etc/samba/smb.conf` (na nekaterih sistemih se uporablja `/etc/smb.conf`). Seveda lahko to stori le administrator sistema. Vsakemu imeniku moramo damo oznako. Ker lahko nastavimo številne parametre, je tudi tukaj najbolje, če si pomagamo s kakšnim grafičnim orodjem. Tukaj je primer datoteke, ki jo je tvorilo grafično orodje, ko smo imenika `/home/meolic/test` in `/home/meolic/test1` dali v skupno rabo pod oznako `test` oz. `test1`.

```
$ cat /etc/samba/smb.conf

... najprej je dosti komentarjev, ki razlagajo posamezne opcije ...

[test]
        comment = to je test
        path = /home/meolic/test
;       writeable = no
        browseable = yes
        guest ok = yes

[test1]
        comment = to je test1
        path = /home/meolic/test1
        writeable = yes
        browseable = yes
        guest ok = yes
```

Na računalniku z operacijskim sistemom Linux, na katerem želimo dostopati do deljenih datotek, uporabimo ukaz mount. Priklop je torej podoben priklopu lokalnih particij, le da pri kretnici `-t`, ki določa tip datotečnega sistema, napišemo smbfs. Oblika ukaza je naslednja:

```
mount -t smbfs //racunalnik/oznaka oddaljen/imenik
```

Primer:

```
mount -t smbfs //meolic.uni-mb.si/test mojtest
```

Tudi v tem primeru, lahko vrstico shranimo v datoteko `/etc/fstab`.

Na Linuxu imamo običajno nameščen tudi pripomoček smbmount, pri katerem ni potrebno podati kretnice -t, ampak napišemo kar:

```
smbmount //racunalnik/oznaka oddaljen/imenik
```

Če ne podamo uporabniškega imena, se uporabi uporabniško ime na lokalnem računalniku (glede na spremenljivki `USER` in `LOGNAME`) oz. uporabniško ime `GUEST`. Uporabniško ime in geslo lahko podamo kot opcija pri ukazu mount v obliki "`username=...,password=...`".

Še boljše pa je, da uporabniško ime in geslo shranimo v datoteko (npr. `.smbpasswd`), ki jo zaščitimo pred branjem tretjih oseb in v katero napišemo naslednji dve vrstici:

```
username= ...
password=...
```

V tem primeru uporabimo opcijo "`credentials=datoteka`".

V povezavi s protokolom SMB imamo na voljo tudi ukaz `smbclient`, s katerim lahko do deljenega imenika po protokolu SMB dostopamo na podoben način kot pri ftp-ju. Poženemo ga lahko na naslednje načine:

```
smbclient //racunalnik/oznaka -U username
smbclient //racunalnik/oznaka -U username%password
smbclient //racunalnik/oznaka -A .smbpasswd
```

17. Izvajalna okolja

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
- dekodiraj inštrukcijo in naloži potrebne podatke iz pomnilnika
- izvedi inštrukcijo
- po potrebi napiši 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.

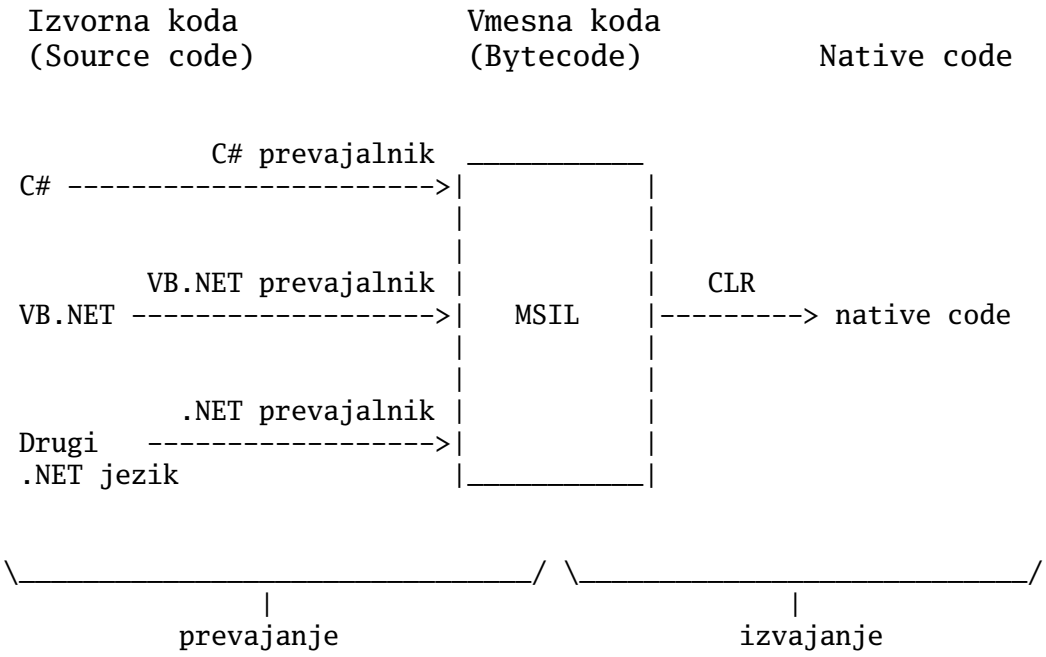
JRE je tipičen primer sistemske programske opreme, za katero skrbi administrator sistema.

JAVA je uvedla tudi novo vrsto programov, tako imenovane spletne programčke (JAVA Applets). Javino izvajalno okolje namreč v osnovi ni kompleksen program, zato ga 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 s pomočjo JRE vgrajenega (oz. dodanega) v spletni brskalnik izvede.

Obstaja več vrst JRE, ki se med sabo razlikujejo v kompleksnosti. Bolj kompleksni JRE poznajo več javinih knjižnic. Prednost majhnih JRE-jev, ki poznajo le osnovne javine knjižnice, je v tem, da so primerni za izvajanje v napravah z omejenimi viri (manj zmogljiv procesor, malo pomnilnika). Dober 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).