UL FRI - Sistemska programska oprema

Upravljalec paketov

Nejc Kišek

9.1.2016

Seminarska naloga pri predmetu Sistemska programska oprema

1 Uvod

Programi za svoje delovanje potrebujejo veliko komponent: izvršilne in konfiguracijske datoteke, knjižnice, dokumentacijo, zunanje programe (odvisnosti), ... Ko želi uporabnik uporabljati nek program, lahko prevede izvorno kodo, dobljene datoteke skopira na pravilno mesto, jim nastavi ustrezne pravice, napiše konfiguracijske datoteke, poskrbi za odvisnosti, itd. Ko je program nameščen ga je treba znati tudi odstraniti ali posodobiti, zato si moramo zapomniti katere datoteke mu pripadajo, katera verzija je nameščena, ...

Na računalnikih, ki jih uporabljamo vsak dan, imamo na tisoče programov, zato je praktično nemogoče, da bi na ta način namestili vsakega izmed njih - nameščanje (ali vsaj del nameščanja) je potrebno avtomatizirati.

2 Načini nameščanja programov

2.1 Nameščanje na nivoju aplikacij

Če operacijski sitem nima sistema za nameščanje programske opreme, potrebujemo zraven vsakega programa tudi namestitveni program - installer. Vsak installer ima lahko povsem svoj način nameščanja in lahko program namesti sam od sebe, vendar pa sistem o tem ne bo nujno obveščen. Brez nekega dodatnega sistema za spremljanje nameščenih programov je razreševanje odvisnosti, spremljanje verzij in upravlanje z nameščenimi programi zelo težko.

Tak pristok k nameščanju programske opreme uporabljajo Microsoftov DOS in Windows operacijski sistemi, novejše verzije Windows-ov pa za spremljanje nameščenih programov uporabljajo register - s tem se olajša odstranjevanje/posodabljanje programov.

2.2 Nameščanje na sistemskem nivoju

Na Unix/Linux sistemih za nameščanje programov skrbi del sistema, ki jih namešča v obliki paketov. Paket je zbirka izvajalnih, konfiguracijskih datotek, knjižnic in drugih datotek, ki skupaj sestavljajo neko celoto - običajno nek program. Pri tem gre lahko za uporabniški ali sistemski program, za knjižnico ali pa le dodatek k drugemu programu. Na današnjih GNU/Linux sistemih je praktično vsa programska oprema nameščena v obliki paketov, tako da isti package manager lahko namesšča, upravlja in posodablja sistemske in uporabniške programe.
Pri upravljanju s paketi nadalje ločimo dva pristopa:

2.2.1 Nadziranje namestitve

Ta način se je razvil iz klasičnih Unix orodij za prevajanje (`make`) in se zato osredotoča na sam proces prevajanja kode - običajno niti ne podpira nameščanja že prevedenih datotek. Uporabljajo ga namestitveni sistemi na BSD Unix-ih (FreeBSD, NetBDS, ...), ki se imenujejo "Ports".

Tak sistem deluje kot make z razširitvami - najprej paket prilagodi našemu sistemu (konfiguracijske datoteke, patchi), ga prevede in namesti, nato pa si ga zapiše še v podatkovno bazo. Omogoči nam tudi, da tik pred namestitvijo program spremenimo ali da nastavimo zastavice v prevajalniku - tako je lahko vsak program prilagojen našemu sistemu in maksimalno optimiziran. Moderne verzije vsebujejo tudi napredne funkcionalnosti, kot je razreševanje odvisnosti ali možnosti avtomatskega pridobivanja kode iz spletnih virov.

Ports in podobni sistemi ne omogočajo nameščana vnaprej prevedenih datotek, kar je za velike programe zelo nepraktično, zato danes lahko na BSD Unix-ih danes hkrati uporabljamo tudi sisteme z drugim pristopom k nameščanju (`pkg`).

2.2.2 Sledenje datotekam

Ta način podpira samo nameščanje že prevedenih datotek in se uporablja na večini komercialnih Unix sistemov (npr. Image packaging system na Solarisu).

Paket pri takem sistemu je arhiv ki vsebuje vse potrebne datoteke, informacijo o pravicah in lokacijo, kamor naj se namestijo. Sistem paket odpre, skopira datoteke na ustrezno mesto in si za vsako posebej v bazo podatkov doda vnos. Na ta način lahko kasneje enostavno preverimo, katera datoteka pripada kateremu programu, izvemo njeno verzijo,... To nam močno olajša odstranjevanje in posodabljanje datotek.

Zaradi uporabe vnaprej prevedenih datotek ta pristop ni tako fleksibilen in prenosljiv, je pa hitrejši.

3 Upravljalec paketov - Package manager

Na modernih GNU/Linux operacijskih sistemih je programska oprema nameščena s pomočjo upravljalca paketov, ki uporablja kombinacijo obeh pristopov k nameščanju. Najpogosteje imamo na voljo pakete z že prevedeno kodo (hitrost), ali pakete z izvorno kodo (prenosljivost, prilagodljivost), uporabnik pa se sam odloči, kakšen paket bo uporabil. Večina upravljalcev paketov podpira tudi naprednejše funkcije, kot je razreševanje odvisnosti, pridobivanje paketov s spletnih virov, različni grafični ali CLI vmesniki,...

S kombiniranjem teh funkcij je nameščanje programov lahko zelo enostavno: npr. uporabnik v grafičnem vmesniku izbere program, ki ga želi namestiti in ta je čez nekaj sekund nameščen. V ozadju package manager pridobi paket za ta program s spletnega repozitorija, na enak način pridobi njegove odvisnosti, vse pakete namesti, poskrbi za dokumentacijo, konfiguracijske datoteke in bljižnice v sistemskih menijih.

Na voljo pa imamo tudi zelo močna orodja: npr. uporabnik zahteva namestitev paketa z izvorno kodo, package manager mu ponudi, da uredi Makefile in nastavi zastavice prevajalnika, da doseže najboljšo prilagoditev svojim potrebam.

V nadaljevanju si bomo podrobneje pogledali RPM in dpkg package managerja.

3.1 RPM Package Manager

RPM package manager (originalno RedHat package manager) je bil razvit za RedHat Linux distribucijo, danes pa ga uporabljajo RedHat Enterprise Linux, Fedora, CentOS, openSUSE, SUSE Linux Enterprise, Mandriva, Mageia,... Zanj je na voljo več vmesnikov, npr. dnf, yum, zypper, ki jih lahko uporabljmo preko različnih grafičnih vmesnikov, npr. yumex, Gnome software, ...

Uporablja dve vrsti paketov: izvorne (source) pakete in prevedene (binary) pakete, ki so obicajno poimenovani kot

```
ime-verzija-izdaja.arhitektura.rpm
```

Primer: `gcc-5.3.1-2.fc23.x86_64.rpm` je paket, ki namesti gcc verzije `5.3.1`, izdaja paketa je `2.fc23` (2. izdaja za Fedoro 23), program je preveden za `x86_64` arhitekturo.

Izvorni paketi vsebujejo arhiv (običajno `.tar.gz`) z izvorno kodo. Vse spremembe originalne izvorne kode so zapisane v `.patch` datotekah. Poleg kode vsebuje tudi `.spec` datoteko, ki nam pove vse o paketu.
Spec datoteka vsebuje:

- informacije o imenu, verziji in izdaji: `Name`, `Version`, `Release`, ...
- opis in kategorijo programa: `Summary`, `%description`, `Group`, ...
- spletno stran programa in povezavo do izvorne kode: `URL`, `Source`
- informacije o odvisnostih: `Requires`/`BuildRequires`, `Provides`, `Conflicts`, ...
- navodila za prevajane in namestitev: `%prep`, `%build`, `%install`, ...
- dodatne informacije, npr. `%changelog`
- za vsak podpaket obstaja sekcija `%package`, ki ima lahko drugačen opis, odvisnosti in podobno.

Izvedba se nekoliko razlikuje od distribucije do distribucije.

Za razreševanje paketov imamo dve glavni polji - Requires in Provides.
`Provides` nam pove, katere funkcionalnosti namesti naš paket - med njimi so lahko ime našega paketa, podpaketov in knjiznic, pa tudi poljubne funkcionalnosti, ki niso nujno imena programov - npr. paket `java-1.8.0-openjdk` ima med provides `java-1.8.0`, `java`, `jre`, `java-fonts`,...
`Requires` nam pove, katere funkcionalnosti program potrebuje za delovanje, `BuildRequires` pa, katere funkcionalnosti potrebujemo med prevajanjem iz izvornega paketa.
Prevedeni (binary) paketi se generirajo iz izvornih rpm datotek in vsebujejo arhiv z vsemi potrebnimi datotekami. Informacije o imenu, verziji, odvisnostih, itd. se zapišejo v glavo .rpm datoteke, ker prevedeni paketi ne vsebujejo `.spec` datotek.

Uporaba RPM

Pakiranje poteka v `~/rpmbuild/` direktoriju, ki vsebuje poddirektorije `BUILD`, `BUILDROOT`, `RPMS`, `SRPMS`, `SOURCES` in `SPECS`. Izvorno kodo, `.patch` datoteke in morebitne druge komponente paketa postavimo v `SOURCES/`, `.spec` datoteko pa v `SPECS/`. S klicem rpmbuild nad `.spec` datoteko nato ustvarimo `.rpm` pakete, z pa zastavicami dolocimo, kaksen tip paketa želimo(`-bb` = binarne, `-bs` = izvorne, `-ba` = vse). Če imamo `.src.rpm` paket, lahko binarnega najhitreje dobimo z ukazom `rpmbild --rebuild paket.src.rpm`.

Pogosti ukazi:

- `rpm -U paket-1.0.0-1.x86_64.rpm`
  namesti paket, oz. posodobi verzijo, če je startejša verzija že nameščena.
  Opcija `-i` je za strogo nameščanje (lahko imamo dve različni verziji paketa naenkrat), opcija `-F` je za strogo posodabljanje (se ne namesti, če ni starejše verzije). Če ukaz poženemo na izvornem paketu, nam ga RPM samo odpakira v `~/rpmbuild/`.

- `rpm -e paket`
  odstrani paket. Odstranjevanje ni dovoljeno, če je nek drug paket od njega odvisen.

- `rpm -q paket`
  v bazi preveri, če je paket s tem imenom nameščen. Če dodamo še opcijo -a , nam izpiše vse nameščene pakete.

- `rpm -q --whatprovides ime` in `rpm -q --provides paket`
  prvi ukaz izpiše, v okviru katerih paketov imamo nameščeno neko funkcionalnost, drugi ukaz pa nam pove, katere funkcionalnosti nam namesti nek paket.

Pri običajni uporabi RPM-ja ne kličemo neposredno z ukazom `rpm`, ampak uporabimo kakšnega od vmesnikov - na RHEL, Fedora in CentOS distribucijah je to tipično `yum` oziroma njegov naslednik `dnf`. Glavna prednost uporabe vmesnika je možnost avtomatskega pridobivanja paketov različnih virov, npr. iz spletnih repozitorijev. Uporaba dnf vmesnika:

- `dnf install paket`, `dnf remove paket`
  doda oz. odstrani paket in vse njegove odvisnosti. Paket lahko podamo kot lokalno .rpm datoteko, ali pa dnf paket in odvisnosti poišče v repozitorijih, ki so definirani v `/etc/yum.repos.d/` direktoriju.

- `dnf update`
  za vse nameščene pakete v repozitorijih poišče novejše verzije in jih posodobi.

- `dnf search niz`
  poišče paket ki v imenu ali opisu vsebuje `niz`.

- `dnf provides ime`
  poišče paket, ki nam namesti funkcionalnost 'ime'. To je uporabno, če potrebujemo knjižnico ali program, ki je del nekega paketa, imena paketa pa ne poznamo.

3.2 Dpkg

Package manager dpkg (debian package) je bil razvit za Debian GNU/Linux distribucijo in ga danes poleg Debiana uporabljajo še njegove mnoge izpeljanke, npr. Ubuntu. Najpogosteje se ga uporablja v kombinaciji z APT (advanced package tool), aptitude ali synaptic vmesnikom.

Običajni (prevedeni) paketi so v obliki `.deb` datoteke formata

```
ime_verzija-izdaja_arhitektura.deb
```

izvorni paketi pa so sestavljni iz treh datotek

- originalna izvorna koda v `.tar.gz` obliki
- spremembe kode v `.tar.gz` ali `.diff.gz` obliki
- control datoteka `.dsc` z informacijami o paketu

`.dsc` datoteka vsebuje podobna polja kot `.spec` pri RPM-ju: ime, verzija, izvorne datoteke, odvisnosti (Depends, Build-Depends, Conflicts, ...), seznam podpaketov.

Prevedeni paketi so prav tako sestavljeni iz treh datotek

- `control.tar.gz`, ki obicajno vsebuje control datoteko, checksum in skripte, ki naj se izvedejo pred ali po namestitvi
- `data.tar.gz`, ki vsebuje vse prevedene datoteke
- datoteko s stevilko verzije formata (`debian-binary`)

Uporaba dpkg

Uporaba naposredno preko `dpkg` ukaza:

- `dpkg -i paket.deb`
  namesti binarni paket
- `dpkg -r paket`
  odstrani paket. Opcija `--purge` odstrani tudi njegove konfiguracijske datoteke.
- `dpkg -I paket.deb`
  izpiše vsebino control
  item `dpkg -l`
  izpiše vse nameščene pakete

Uporaba dpkg preko APT vmesnika:

- `apt-get install paket`
  namesti `.deb` paket ali pa ga poisce v repozitorijih, ki so navedeni v `/etc/apt/sources.list.d/` ali `/etc/apt/sources.list`. Namesti tudi odvisnosti.

- `apt-get remove paket`
  odstrani paket. Z dodano zastavico --purge odstrani tudi njegove konfiguracijske datoteke.

- `apt-get update`
  posodobi informacije o repozitorijih

- `apt-get upgrade`
  posodobi nameščene pakete

Viri

- Edward C. Bailey; Maximum RPM; 2000, Red Hat inc.
  http://www.rpm.org/max-rpm/
- RPM Guide; Eric Foster-Johnson, Stuart Ellis, Ben Cotton; 2005/2011, Fedora Project Contributors
  https://docs.fedoraproject.org/en-US/Fedora_Draft_Documentation/0.1/html/RPM_Guide/
- Debian Reference; Osamu Aoki; 2013, Osamu Aoki
  https://www.debian.org/doc/manuals/debian-reference/
- FreeBSD Handbook; 1995-2015, The FreeBSD Project
  https://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/index.html
- https://fedoraproject.org/wiki/Category:Package_Maintainers
- https://wiki.debian.org/PackageManagement