Tampereen yliopisto

Topi-Matti Ritala

JAKELURIIPPUMATON PAKETOINTI FLATPAKILLA

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintyö
Syyskuu 2019

TIIVISTELMÄ

Topi-Matti Ritala: Jakeluriippumaton paketointi Flatpakilla
Kandidaatintyö
Tampereen yliopisto
Tietotekniikan kandidaatin tutkinto-ohjelma
Syyskuu 2019

Aktiivisessa kehityksessä olevia Linux-jakeluita on lähes 500, joten ohjelmien julkaiseminen mahdollisimman laajasti on hankalaa. Jakelut sisältävät eri järjestelmäkirjastojen eri versioita ja yhteensopivuusongelmia voi tulla vastaan paljon. Flatpak on jakeluriippumaton paketinhallinta-järjestelmä, mikä pyrkii helpottamaan ohjelmien kehittämistä ja levittämistä jakeluiden pirstoutumisesta huolimatta.

Flatpak tarjoaa kehittäjille vakaan alustan ja helpon tavan julkaista ohjelmia laajalle yleisölle. Teknologian nuori ikä ja siitä johtuva epävarmuus sekä kilpailevan Snap-teknologian painostus ovat kuitenkin haasteita, mitkä Flatpakin täytyy vielä ylittää.

Avainsanat: Flatpak, Linux, paketinhallinta

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck –ohjelmalla.

SISÄLLYSLUETTELO

1. JOHDANTO....1
2. LÄHTÖKOHDAT....2
2.1 Linux-ydin....2
2.2 GNU C-standardikirjasto....3
2.3 Muut jaetut kirjastot....4
2.4 Paketinhallinta ja pakettivarasto....5
3. FLATPAK....6
3.1 Ajonaikainen ympäristö....7
3.2 Hiekkalaatikko ja portaalit....8
3.3 Pakettivarasto....9
3.4 Pakettien nimeäminen....9
3.5 Ohjelmiston pakkaaminen....10
4. FLATPAKIN HAASTEET....13
5. YHTEENVETO....14
LÄHTEET....15

LYHENTEET JA MERKINNÄT

git	versionhallintajärjestelmä
glibc	GNU projektin toteutus C-ohjelmointikielen standardikirjastosta
GNOME	GTK-pohjainen työpöytäympäristö
GNU	Unixin kaltainen käyttöjärjestelmä, jonka ohjelmia käytetään usein Linux-ytimen kanssa
GTK	alustariippumaton käyttöliittymäkirjasto
JSON	JavaScript Object Notation, tiedon siirtämiseen ja konfigurointiin käytetty ihmisluettava tiedostomuoto
KDE	Qt-pohjainen työpöytäympäristö
Linux	avoimen lähdekoodin käyttöjärjestelmäydin
Linux-jakelu	Linux-ytimen sisältävä kokoelma ohjelmia ja kirjastoja, mikä yhdessä muodostaa käyttöjärjestelmän
POSIX	Portable Operating System Interface käyttöjärjestelmästandardi
Qt	alustariippumaton käyttöliittymäkirjasto
SDK	Software Development Kit, kehitysympäristö
URI	uniform resource identifier, tiedon paikkaa tai nimeä kuvaava merkkijono
X11	ikkunointipalvelin
XDG	X Desktop Group avoimien työpöytäympäristöjen standardointiin keskittyvä projekti, nykyään nimellä freedesktop.org
YAML	Yet Another Markup Language, ihmisluettava usein konfigurointiin käytetty tiedostomuoto

.

1. JOHDANTO

Aktiivisessa kehityksessä olevia Linux-jakeluita on lähes 500 [1] ja ne saattavat sisältää toistensa kanssa yhteensopimattomia versioita eri ohjelmakirjastoista. Tämä pirstoutuminen vaikeuttaa ohjelmien levittämistä Linuxilla, sillä kehittäjä ei voi tehdä paljoa oletuksia ympäristöstä, missä käyttäjä tulee ohjelmaa ajamaan. Ohjelmistokehittäjien vaihtoehtoina on historiallisesti ollut jakeluriippuvaisten pakettien julkaiseminen muutamalle suositulle jakelulle tai tarkasti valitut niputetut kirjastot.

Flatpak on jakeluriippumaton paketinhallintajärjestelmä, joka tarjoaa kehittäjille mahdollisuuden julkaista ohjelmistonsa kerralla kaikille Flatpakia tukeville jakeluille. Se tarjoaa vakaan kehitysalustan ja takaa eteenpäin yhteensopivan ympäristön ohjelmille. Se tarjoaa myös tietoturvaa käyttäjille, sillä Flatpakilla pakatut ohjelmistot suoritetaan tiukaan eristetyssä hiekkalaatikossa ja niiden pääsyä isäntäjärjestelmän tiedostoihin ja prosesseihin kontrolloidaan tarkasti

Tässä työssä tutustutaan Flatpak-paketinhallintajärjestelmän toimintaan kirjallisuuskatsauksen muodossa. Luvussa 2 käydään läpi tyypillisen Linux-jakelun komponentteja ja niiden vaikutusta ohjelmien yhteensopivuudelle. Tässä luvussa kerrotaan myös yleisesti paketinhallintajärjestelmien tehtävistä ja toiminnasta. Luvussa 3 tutustutaan itse Flatpak-paketinhallintajärjestelmään. Ensiksi kerrotaan hieman Flatpakin taustoista. Tämän jäkeen käydään läpi Flatpakilla pakatun ohjelman peruskomponentit: ajonaikainen ympäristö, ohjelman hiekkalaatikko ja pakettivarasto. Lopuksi tutustutaan itse pakkaamiseen yleisluontoisesti. Luvussa 4 pohditaan Flatpakin ongelmia ja niiden mahdollista vaikutusta sen käytön yleistymiselle. Työ päättyy luvun 5 yhteenvetoon.

2. LÄHTÖKOHDAT

Linux-pohjaisia käyttöjärjestelmiä levitetään tavallisesti Linux-jakeluina. Jakelu on jonkin yhteisön tai yrityksen koostama kokoelma ohjelmia ja kirjastoja, jotka yhdessä muodostavat käyttöjärjestelmän. Tämä kokoelma sisältää yleensä vähintään Linux-ytimen, ohjelmistokirjastoja kuten C-standardikirjasto sekä nipun perusohjelmia kuten GNU Core Utilities tai BusyBox. Jakelu voi olla yleiskäyttöinen, eli se sopii käytettäväksi niin työpöydillä kuin palvelimilla, tai se voi olla suunniteltu erityiskäyttöön, kuten sulautettuihin järjestelmiin. Tässä työssä tullaan keskittymään jakeluriippumattomaan paketoimiseen nimenomaan yleiskäyttöisille työpöytäjakeluille.

Koska Linux-jakeluiden komponentit tulevat monesta lähteestä monilta eri kehittäjiltä, jakelut voivat sisältää huvinkin erilaisia kokoonpanoja eri järjestelmäkirjastojen ja -ohjelmistojen eri versioista. Moni jakelu noudattaa enemmän tai vähemmän POSIX-käyttöjärjestelmästandardia, mikä helpottaa jossain määrin siirrettävän koodin kirjoittamista. POSIX ei kuitenkaan määritä standardeja graafisten käyttöliittymien toiminnalle ja toteutukselle.

Ohjelmistojen yhteensopivuuden kannalta tärkeimmät komponentit jakeluissa ovat itse Linux-käyttöjärjestelmäydin, C-standardikirjasto, sekä jakelun ylläpitäjien pakkaamat muut jaetut ohjelmakirjastot.

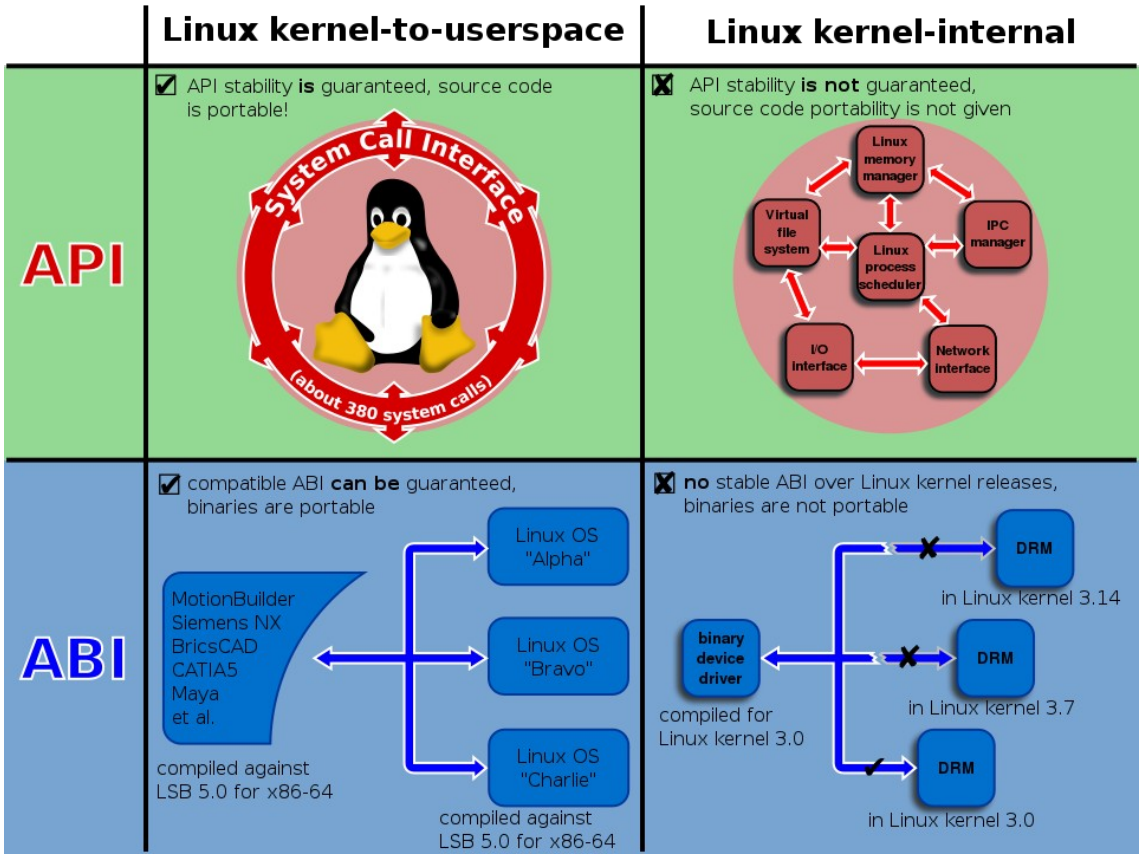
2.1 Linux-ydin

Linux-käyttöjärjestelmäydin on jokaisen Linux-jakelun perusta. Ydin on vastuussa keskeytysten käsittelystä, prosessien hallinnasta, sekä oheislaitteiden ohjauksesta. [2, s. 49] Ydin sijaitsee muistissa omassa osoiteavaruudessa, ydintilassa (kernel space), kun taas muut ohjelmat sijaitsevat sen ulkopuolella käyttäjätilassa (user space).

Linux-ydin tarjoaa käyttäjätilan ohjelmille palveluita järjestelmäkutsuilla. Järjestelmäkutsujen käyttö kuitenkin vaatii alustariippuvaisen assembly-koodin käyttämistä, joten niitä ei tavallisesti käytetä suoraan. C-standardikirjasto sisältää järjestelmäkutsuille alustariippumattomat käärefunktiot. [3] Linux-ytimen kehittäjillä on tiukka politiikka ytimen ja käyttäjätilan välisen rajapinnan vakaudesta. Ytimen muutoksista johtuvat käyttäjätilan regressiot otetaan vakavasti ja pyritään korjaamaan mahdollisimman nopeasti. [11]

Ytimeen on myös mahdollista kehittää ydinmoduuleja. Ydinmoduulit ladataan ydintilaan ja niiden avulla on mahdollista laajentaa ytimen toimintaa. Laiteajurit ovat tavallisesti ydinmoduuleja. Linux-ytimen sisäinen rajapinta on kuitenkin tarkoituksella epävakaa,

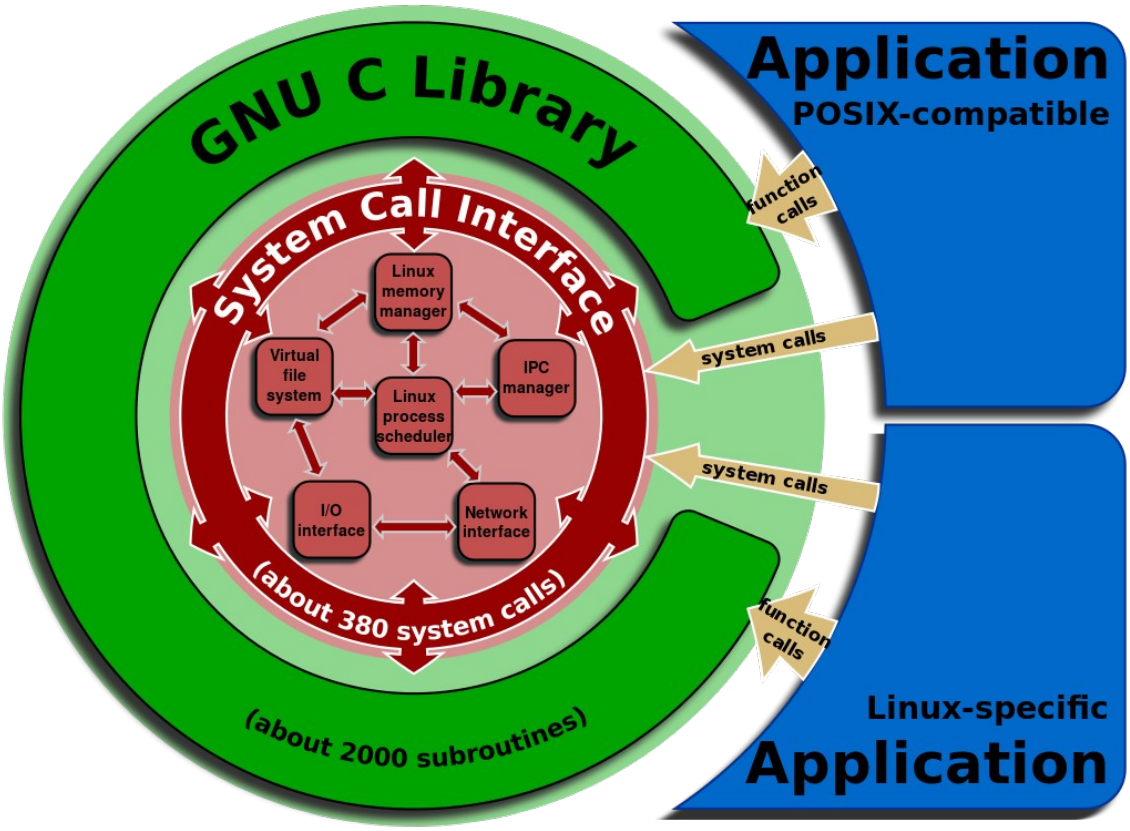
joten moduulit tulee kääntää uudelleen ytimen version muuttuessa. [19] Kuva 1 kuvaa ytimen sisäisen ja ulkoisen rajapinnan vakautta.



Kuva 1. *Kuvaus Linux-ytimen ulkoisesta ja sisäisestä rajapinnasta*. [32]

2.2 GNU C-standardikirjasto

GNU C-standardikirjasto eli ”glibc” on keskeinen osa Linux-jakeluita. Se toteuttaa rajapinnat ISO, POSIX sekä BSD C-kirjastoille ja käärefunktiot ytimen järjestelmäkutsuille. Linux-jakelut voivat käyttää myös vaihtoehtoisia toteutuksia C-standardikirjastosta, mutta glibc on selkeästi eniten käytetty ja tuettu. [4] Koska C-standardikirjasto on määritelty Unix- ja POSIX-käyttöjärjestelmästandardeissa ja se toteuttaa alustariippumattomat käärefunktiot ytimen järjestelmäkutsuille, riippuvat myös monella muullakin kielellä kirjoitetut ohjelmat joko suoraan tai epäsuoraan libc:stä. Kuva 2 esittää libc:n toiminnan järjestelmäkutsujen kääreenä ja POSIX-yhteensopivien ohjelmien pohjana.



Kuva 2. *Kuvaus GNU C-kirjaston toiminnasta ytimen järjestelmäkutsujen kanssa*. [33]

Glibc on pysynyt pääosin taaksepäin yhteensopivana vuonna 1997 julkaistusta glibc 2.0:sta lähtien. Taaksepäin yhteensopivuus on toteutettu käyttäen versioituja symboleja, jotka mahdollistavat vanhaa glibc:tä vasten käännettyjen ohjelmien toimimisen myös uuden glibc:n kanssa, vaikka joidenkin funktiokutsujen syntaksi olisikin muuttunut kahden vuosikymmenen aikana. [12] Tämä kuitenkin tarkoittaa sitä, ettei ajettava ohjelma voi olla käännetty järjestelmässä olevaa glibc:tä uudempaa versiota vasten.

2.3 Muut jaetut kirjastot

Linux-ytimen ja C-standardikirjaston tarjoamat rajapinnat ovat jokaisen Linux-jakelun vakaa perusta, mutta vain niitä käyttäen on käytännöllistä toteuttaa lähinnä tekstipohjaisia C tai C++ ohjelmia ja kirjastoja. Etenkin graafisia käyttöliittymiä kehittäessä tulee helposti tarve käyttää muita kirjastoja.

GTK ja Qt ovat suurimmat ja eniten käytetyt käyttöliittymäkirjastot Linuxilla. Valtaosa merkittävistä työpöytäympäristöistä perustuu näihin kirjastoihin. [13] Molemmat kirjastot pyrkivät säilyttämään taaksepäin yhteensopivuutta noin vuosikymmenen tukea saavilla pääversioilla. [14] [15] Tämä ei kuitenkaan aina toteudu ja etenkin GTK3 on saanut paljon kritiikkiä yhteensopivuuden rikkomisesta pääversioiden sisällä. Tämä ja Qt:n pa-

rempi tuki Linuxin ulkopuolisille alustoille on johtanut monen projektin siirtymiseen GTK:sta Qt:hen. [16] [17]

Jotkin kirjastot eivät lainkaan takaa taaksepäin yhteensopivuutta eri versioiden välillä. Esimerkiksi suositut Boost-kirjastot eivät takaa vakautta ja kirjaston päivittäminen vaatii koko ohjelman uudelleen kääntämistä. [18]

2.4 Paketinhallinta ja pakettivarasto

Paketinhallintajärjestelmä on ohjelma tai nippu ohjelmia, jonka tehtävä on hallita käyttöjärjestelmän paketteja. Se on vastuussa pakettien päivityksestä, asentamisesta, poistamisesta ja riippuvuuksien hallinnasta. Paketti on arkistotiedosto, mikä sisältää itse ohjelman lisäksi asennukseen ja hallintaan käytettävää metatietoa, kuten sen version, kehittäjän, riippuvuudet ja mahdollisesti tarkistussummia. Paketinhallinnan tarkoitus on helpottaa järjestelmän ylläpitoa ja eliminoida tarve yksittäisten ohjelmien manuaaliselle päivittämiselle tai asentamiselle.

Paketinhallintajärjestelmä etsii ja lataa paketteja pakettivarastoista. Linux-jakeluilla on yleensä virallinen keskitetty pakettivarasto, jota jakelun kehittäjät ylläpitävät, mutta myös kolmannen osapuolen pakettivarastoja on mahdollista käyttää.

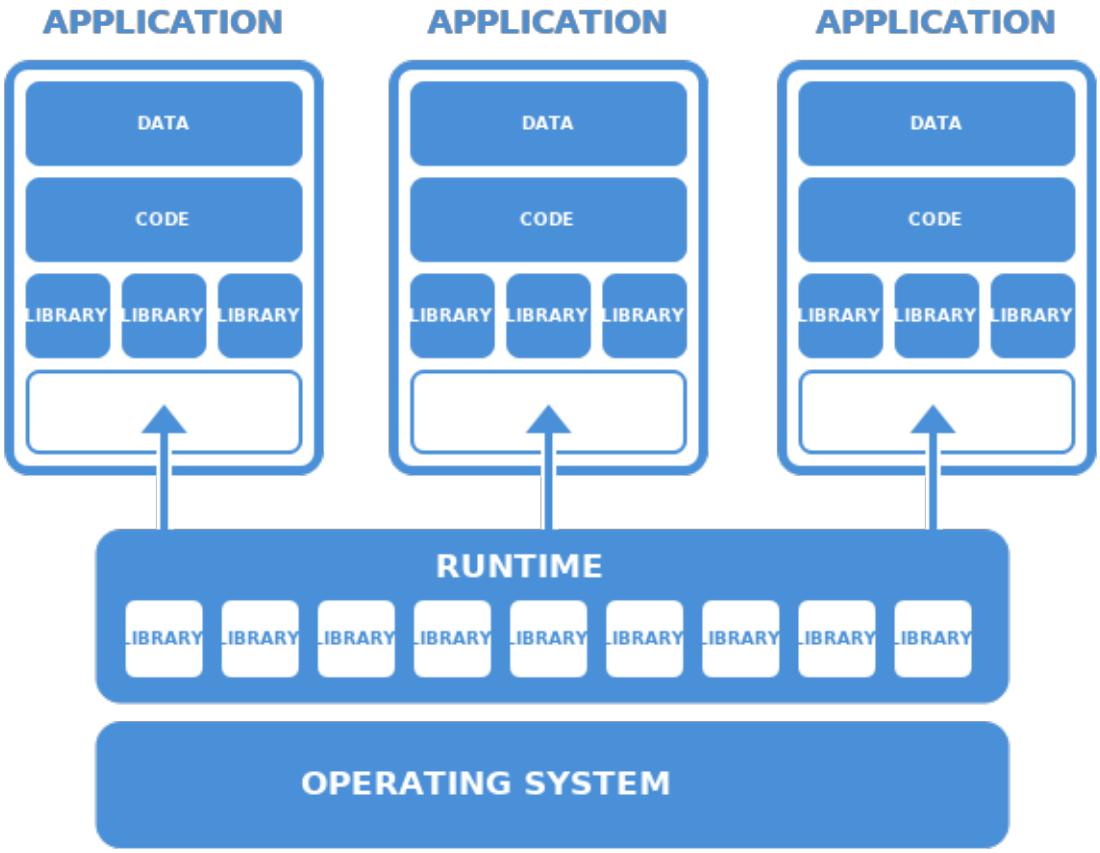
Vaikka paketinhallinnan tehtävä on hallita järjestelmään asennettuja ohjelmia ja niiden riippuvuuksia, ei sen käyttö kuitenkaan ole pakollista joka tilanteessa. Esimerkiksi jos jotakin ohjelmaa ei ole saatavilla pakettivarastoista tai pakettivarastoissa oleva versio ei ole juuri se mitä käyttäjä on hakemassa, voi sen ladata itse kehittäjän sivulta binäärimuodossa tai kääntää lähdekoodista. Käsin asennettujen ja käännettyjen ohjelmien kuitenkin tulisi kuitenkin sijaita erillään paketinhallinnan ylläpitämistä ohjelmista yhteensopivuusongelmien välttämiseksi.

3. FLATPAK

Flatpak on jakeluriippumaton paketinhallintajärjestelmä, jonka tavoitteena on tarjota vakaa eteenpäin yhteensopiva alusta ohjelmistoille ja helpottaa niiden levittämistä eri Linux-jakeluille. Se sai alkunsa Red Hatilla työskennelleen Alexander Larssonin xdg-app -projektista, jonka ensimmäinen julkinen versio julkaistiin vuonna 2015. Xdg-app otti inspiraatiota Larssonin aikaisemmista jakelumenetelmäkokeiluista ja vuonna 2013 julkaistusta Docker-sovelluksesta. Flatpak 1.0 julkaistiin elokuussa 2018 ja se on saatavilla kaikkien merkittävien Linux-jakeluiden pakettivarastosta. [5]

Flatpak perustuu OSTree- ja Bubblewrap-tekniikoihin. OSTree on Gitin tapainen versionhallintajärjestelmä. Gitistä poiketen se on suunniteltu toimimaan binääritiedostojen ja muun ei-tekstipohjaisen datan hallintaan. Bubblewrap on työkalu, mikä mahdollistaa ytimen eristysmenetelmien hallinnan ja käytön ilman pääkäyttäjän oikeuksia. [21]

Flatpakilla pakatun sovelluksen voidaan katsoa koostuvan muutamasta peruskomponentista. Nämä peruskomponentit ovat sovelluksen hiekkalaatikko (en. sandbox), niputetut ohjelmistokirjastot ja sovelluksen käyttämä jaettu ajonaikainen ympäristö (en. runtime). Kuva 3 esittää tämän Flatpakilla pakattujen sovellusten perusrakenteen visuaalisesti. Ajonaikainen ympäristö on nippu yleisiä järjestelmäsovelluksia ja -kirjastoja, joka muistuttaa rakenteeltaan minimaalista Linux-jakelua. Jokainen pakattu sovellus riippuu yhdestä ajonaikaisesta ympäristöstä. Sovellukset on pakattu eristettyihin hiekkalaatikoihin, jotka sisältävät itse sovelluksen lisäksi mahdollisia lisäkirjastoja, joita ajonaikainen ympäristö ei sisältänyt. [6]



Kuva 3. *Flatpakilla pakattujen sovellusten rakenne*. [6]

Ajonaikaisia ympäristöjä ja pakattuja sovelluksia jaetaan muiden paketinhallintajärjestelmien tavoin pakettivarastoista (en. repositories). Omia pakettivarastoja voi luoda itse tai sovelluksen voi julkaista virallisessa Flathub-pakettivarastossa.

3.1 Ajonaikainen ympäristö

Jokainen Flatpakilla pakattu sovellus riippuu yhdestä ajonaikaisesta ympäristöstä. Ajonaikainen ympäristö tarjoaa sovellukselle minimaalisen Linux-jakelun tapaisen nipun tärkeimpiä järjestelmäsovelluksia ja -kirjastoja. Ajonaikaiset ympäristöt tarjoavat myös kehitysympäristön (en. SDK) ja laajennuksia kuten lokalisointitiedostoja. Ajonaikaisia ympäristöjä on olemassa kolme: Freedesktop, GNOME ja KDE. [7]

Freedesktop tarjoaa sovelluksille tärkeimmät peruskirjastot ja -palvelut kuten D-Bus prosessien välisen kommunikoinnin rajapinta, GTK3-käyttöliittymäkirjasto, PulseAudio-äänipalvelin sekä X11- ja Wayland-ikkunointipalvelimet. Se toimii Flatpakin vakioympäristönä. Freedesktop-ympäristö on myös saamassa Debianin multiarch-toteutukseen perustuvan tuen vieraille suoritinarkkitehtuureille ristiinkääntämiselle ja vieraiden arkkitehtuurien kirjastojen asentamiselle. [8]

GNOME- ja KDE-ympäristöt perustuvat Freedesktop-ympäristöön ja lisäävät siihen tuen omille käyttöliittymäkirjastoillensa. Freedesktop itsessään tukee jo pieniä GTK-pohjaisia sovelluksi, mutta GNOME laajentaa GTK-tukea tarjoamalla lisäkirjastoja kuten GStreamer, Libnotify, Vala ja WebKitGTK. KDE tarjoaa tuen Qt-käyttöliittymäkirjastolle sekä sitä laajentavalle KDE Frameworks kirjastokokoelmalle.

3.2 Hiekkalaatikko ja portaalit

Kaikki Flatpakilla pakatut sovellukset suoritetaan oletuksena eristetyssä hiekkalaatikossa. Kehittäjä voi määrittää pakatessaan mitä käyttöoikeuksia sovellus vaatii toimiakseen oikein. Näihin oikeuksiin kuuluu Flatpakin hallitsema rajoitettu pääsy mm. PulseAudio-äänipalvelimelle ja X11-ikkunointipalvelimelle. Pakattu ohjelma voi pyytää pääsyä isäntäjärjestelmän tiedostojärjestelmään portaalien avulla. Ohjelmalle voidaan paljastaa yksittäisiä kansioita, kuten käyttäjän Kuvat-kansion tai koko tiedostojärjestelmän. Ohjelma voi myös avata URI-linkkejä portaalien kautta.

Hiekkalaatikon sisäinen tiedostojärjestelmä muistuttaa hierarkialtaan tavallista pientä Linux-jakelua. Se muodostuu ohjelmalle luodusta yksityisestä väliaikaisesta tiedostojärjeselmästä jossa:

- / on yksityinen tempfs, joka ei näy muualle.
- /usr on liitetty ajonaikaiseen ympäristöön.
- /app on liitetty itse Flatpakilla pakattuun ohjelmaan.
- Ohjelman luomat käyttäjäkohtaiset asetustiedostot ohjataan isännän $HOME/.var/app/$APPID -kansioon.
- /sys on liitos isännän /sys -kansioon ilman kirjoitusoikeuksia.
- /proc näyttää vain hiekkalaatikossa suoritettavan ohjelman ja sen aliohjelmien prosessit.
- /dev sisältää tärkeimmät näennäislaite-erikoistiedostot /dev/full, /dev/null, /dev/random, /dev/urandom, /dev/tty ja /dev/zero.
- /etc sisältää passwd, group ja resol.conf tiedostot.

Hiekkalaatikon sisällä on asetettu myös ympärisömuuttujat:

- PATH=/app/bin:/usr/bin
- LD_LIBRARY_PATH=/app/lib
- XDG_CONFIG_DIRS=/app/etc/xdg:/etc/xdg
- XDG_DATA_DIRS=/app/share:/usr/share
- XDG_RUNTIME_DIR=/run/user/$pid [9]

3.3 Pakettivarasto

Tavallisten paketinhallintajärjestelmien tavoin Flatpak käyttää pakettivarastoja. Flatpakilla on virallinen oletuspakettivarasto Flathub, jonne kehittäjät voivat julkaista ohjelmansa. Flathubia ylläpidetään GitHubissa ja ohjelmien lisääminen on toteutettu käyttäen tavallisia pull requesteja. Yksityisen pakettivaraston ylläpitäminen on myös mahdollista.

Pakettivarastot toimivat git-versionhallintaohjelman tavoin. Ne sisältävät ohjelmien kaikki version ja tiedot niiden välillä tapahtuneista muutoksista. Tämä mahdollistaa latauksien kokoa pienentävien ja niitä nopeuttavien delta-päivityksien käytön. [20]

3.4 Pakettien nimeäminen

Jokaisella Flatpakilla pakatulla ohjelmalla tai ajonaikaisella ympäristöllä on kolmiosainen nimi, joka on muotoa *com.company.App*. Nimen kaksi ensimmäistä osaa identifioivat ohjelman kehittäjän ja viimeinen osa yksilöi sovelluksen. Tällä tavoin kaksi eri kehittäjää voivat julkaista saman nimiset sovellukset *com.company1.App* ja *org.company2.App*, tai yksi kehittäjä voi julkaista useita ohjelmia *com.company.App1* ja *com.company.App2*.

Kolmiosaisen nimen lisäksi jokainen versio ohjelmasta identifioidaan täydellisen tunnistekolmikon avulla. Tunnistekolmikko on muotoa *nimi/arkkitehtuuri/haara* esimerkiksi *com.company.App/x86_64/stable*. Haara voi kuvata kehityshaaraa, kuten *stable* tai *testing* tai se voi olla jokin tietty versionumero. Haaran tai arkkitehtuurin voi ohittaa jättämällä kyseisen kentän tyhjäksi. Flatpak olettaa haaraksi *stable* ja arkkitehtuuriksi nykyisen arkkitehtuurin.

3.5 Ohjelmiston pakkaaminen

Flatpak käyttää omaa *flatpak-builder* -kääntämisjärjestelmää (en. build system). Se pystyy ajamaan muita kääntämisjärjestelmiä ja tukee myös riippuvuuksien lataamista ja kääntämistä. *Flatpak-builder* kääntää ohjelmat valitun ajonaikaisen ympäristön kehitysympäristöä vastaan isäntäjärjestelmän sijasta. Kehitysympäristö vastaa valittua ajonaikaista ympäristöä, mutta se sisältää lisäksi vielä kääntämiseen tarvittavia komponentteja, kuten kääntäjiä, virheenjäljittäjiä ja kirjastojen otsikkotiedostoja.

Flatpak mahdollistaa ulkopuolisten kirjastojen niputuksen ohjelman mukana. Tämä voi olla tarpeellista esimerkiksi jos valitussa ajonaikaisessa ympäristössä ei ole kaikkia haluttuja kirjastoja, kirjastojen version on väärä tai kirjastot vaativat muutoksia.

Flatpak olettaa että pakattu ohjelma noudattaa muutamia standardeja, jotta ne voidaan integroida isäntäjärjestelmään hiekkalaatikosta huolimatta.

- Ohjelmalla tulee olla käänteis-DNS notaation mukainen ID.
- Ohjelmalla tulee olla AppData-metadatatiedosto, jolla graafiset paketinhallinta työkalut voivat listata ohjelman.
- Freedesktop-standardin mukainen kuvake PNG- tai SVG-formaatissa.
- Freedesktop-standardin mukainen työpöytätiedosto, jolla ohjelma voidaan käynnistää graafisesta käyttöliittymästä.
- D-Bus:n käyttö prosessien väliseen kommunikointiin, mikäli tarpeellista.
- XDG base directory -standardin noudattaminen käyttäjäkohtaisten asetustiedostojen tallentamiseen.

Tavallinen Linux-ohjelma noudattaa jo todennäköisesti edellä mainittuja standardeja käänteis-DNS notaatiota lukuun ottamatta. [22]

Flatpak-builder ottaa syötteeksi manifesti-tiedoston. Se on JSON- tai YAML-formaattinen tekstitiedosto, joka kuvaa ohjelman riippuvuudet ja käännösohjeet. Kuvassa 4 on Flathubista saatavan GNOME Dictionary -ohjelman manifestitiedosto, jota käytetään myös Flatpakin virallisessa dokumentaatiossa esimerkkinä. Manifesti alkaa ohjelmaa ja sen käyttämää ajonaikaista ympäristöä kuvaavalla metadatalla. Esimerkkimanifestissa on määritetty ohjelman ID *app-id*, haara *branch* ja ohjelman ajava komento *command*. Siinä on määritetty myös käytettävä ajonaikainen ympäristö ja sen versio *runtime* ja *runtime-version* sekä ajonaikaista ympäristöä vastaava kehitysympäristö *sdk*.

```
{
    "app-id": "org.gnome.Dictionary",
    "runtime": "org.gnome.Platform",
    "runtime-version": "3.24",
    "branch": "stable",
    "sdk": "org.gnome.Sdk",
    "command": "gnome-dictionary",
    "tags": [],
    "finish-args": [
        /* X11 + XShm access */
        "--share=ipc", "--socket=x11",
        /* Wayland access */
        "--socket=wayland",
        /* Needs to talk to the network: */
        "--share=network",
        /* Needed for dconf to work */
        "--filesystem=xdg-run/dconf", "--filesystem=~/.config/dconf:ro",
        "--talk-name=ca.desrt.dconf", "--env=DCONF_USER_CONFIG_DIR
                                        =.config/dconf"
    ],
    "build-options" : {
        "cflags": "-O2 -g",
        "cxxflags": "-O2 -g"
    },
    "cleanup": ["/include", "/lib/pkgconfig",
                "/share/pkgconfig", "/share/aclocal",
                "/man", "/share/man", "/share/gtk-doc",
                "*.la", "*.a",
                "/lib/girepository-1.0",
                "/share/doc", "/share/gir-1.0"
    ],
    "modules": [
        {
            "name": "gnome-dictionary",
            "buildsystem": "meson",
            "builddir": true,
            "config-opts": [ "--libdir=/app/lib", "-Dbuild_man=false" ],
            "sources": [
                {
                    "type": "archive",
                    "url": "https://download.gnome.org/sources/gnome-
                            dictionary/3.24/gnome-dictionary-3.24.0.tar.xz",
                    "sha256": "41e7064a0cfab18e881a95ce9f1712ee
                               5c9f426904b16f3bc04c35ebd1bbd9f2"
                }
            ]
        }
    ]
}
```

Kuva 4. *GNOME Dictionary -ohjelman manifestitiedosto Flathubissa*. [23]

Osio *finish-args* kuvaa pakatun ohjelman hiekkalaatikon oikeuksia. Ilman argumentteja ohjelma ajetaan käytännössä täysin erillään isäntäjärjestelmästä. Kuvan 4 esimerkki-manifestissa ohjelmalle annetaan pääsy käyttämään isännän verkkoa ja Wayland- tai X11-ikkunointipalvelimia. Pääsy annetaan myös isännän dconf-asetustiedostoon. Käyttämällä *filesystem* argumenttia ohjelmalle voidaan antaa pääsy yksittäisille kansiolle tai esimerkiksi käyttäjän kotikansioon ilman pääsyä piilotiedostoihin. Se tukee myös vain luku rajoitteen asettamista käyttämällä kansion polussa *:ro*-päätettä. Ohjelmalle on

myös mahdollista antaa pääsy isäntäjärjestelmän näytönohjaimelle ja PulseAudio-ääni-palvelimelle. [24]

Manifestin *cleanup*-osio kuvaa tiedostoja, jotka kehittäjä tahtoo poistaa ennen pakkausvaihetta. Tämä yleensä sisältää käännöksen aikana luotuja väliaikaistiedostoja sekä mahdollisia ylimääräisiä kehittäjille suunnattuja dokumentaatiotiedostoja.

Modules-osio kuvaa itse ohjelman ja sen riippuvuuksien lähdekoodin tai binääritiedostojen sijaintia ja niiden käännösohjeita. Moduulin *sources*-kenttä kertoo lähdekoodin tai binääritiedoston tyypin ja sijainnin. Se voi sijaita lokaalisti levyllä tai se voidaan ladata internetistä suoraan arkistotiedostona. Myös git- tai svn-versionhallintajärjestelmän tietovaraston kloonaaminen on tuettua. Lähdekoodille voidaan määrittää käytetty kääntämisjärjestelmä. Flatpak tukee automaattisesti yleisimpiä kääntämisjärjestelmiä kuten autotools, cmake ja meson. Kääntämisjärjestelmän tilalle voidaan määrittää myös sarja komentorivikomentoja. Tämä mahdollistaa esimerkiksi Pythonin oman pip-paketinhallintajärjestelmän käyttämisen ilman suoraa tukea. [25] [26]

4. FLATPAKIN HAASTEET

Ubuntua kehittävä Canonical on myös kehittänyt oman jakeluriippumattoman paketinhallintajärjestelmän Snap. Flatpakista poiketen Snap käyttää vain yhtä keskitettyä pakettivarastoa Snap Store, jonne ohjelmien lisääminen on verrattavissa esimerkiksi Androidin Play Storeen tai iOS:n App Storeen julkaisemiseen. [27] Flatpakin Flathub puolestaan vaatii pull requestin lähettämistä julkiseen git-tietovarastoon ja ohjelman ylläpidon tulee tapahtua julkisesti GitHubin kautta. [28] Etenkin suljetun lähdekoodin ohjelmistoille Snap Storen käyttö voi olla huomattavasti helpompaa. Snap Store tukee myös suoraan maksullisia ohjelmistoja, mitä Flatpak ei tue lainkaan.

Joidenkin monimutkaisten ohjelmien paketoiminen on Flatpakilla mahdotonta, sillä hiekkalaatikon eristyksestä johtuen Flatpak ei anna ohjelmien asentaa omia ydinmoduuleja. Tämä on ongelmallista esimerkiksi VirtualBox-virtualisointiohjelmalle, mikä käyttää ydinmoduuleja virtualisoinnin apuna. [29]

Flatpak on myös hyvin nuori teknologia. Vaikka Flatpakin dokumentaatio väittääkin: ”Forward-compatibility: the same Flatpak can be run on different versions of the same distribution, including versions that haven’t been released yet. This doesn’t require any changes or management by application developers.” [30], ei Flatpakilla ole vielä näyttöä yhteensopivuuden ylläpidosta pitkällä aikavälillä. Edellä mainittuun lainaukseen tuo vielä epävarmuutta se, ettei Flatpakilla ole virallista suunnitelmaa elinkaarensa loppuun päässeiden ajonaikaisten ympäristöjen arkistoimiseen. [31]

5. YHTEENVETO

Flatpak on nuori mutta lupaava teknologia Linux-jakeluiden aiheuttaman pirstoutumisen helpottamiseen. Se on jo saatavissa kaikissa suosituimmissa Linux-jakeluissa. Se myös helpottaa ohjelmien julkaisemista ja riippuvuuksien hallintaa ja (ainakin teoriassa) lupaa vakaan alustan, millä ohjelma voidaan ajaa vaikka isäntäjärjestelmää päivitetään tai vaihdetaan täysin toiseen jakeluun.

Flatpakilla on kuitenkin vielä haasteita ylitettävänä. Käytännön näytön puute yhteensopivuuden ylläpidosta tuo alustalle epävarmuutta. Myös kilpaileva Snap-teknologia vaikuttaa Flatpakilla pakattujen ohjelmien lukumäärään. Etenkin suljetuille ja kaupallisille ohjelmille Snap tarjoaa tällä hetkellä huomattavasti paremman alustan.

LÄHTEET

[1] The LWN.net Linux Distribution List. Viitattu: 12.7.2019. Saatavissa: https://lwn.net/Distributions/

[2] Haikala, Ilkka & Järvinen, Hannu-Matti: Käyttöjärjestelmät, Helsinki: Talentum, 2003, ISBN 951-762-837-4

[3] Linux Programmer's Manual – syscalls. Viitattu: 15.7.2019. Saatavissa: http://man7.org/linux/man-pages/man2/syscalls.2.html

[4] Linux Programmer's Manual – libc. Viitattu: 15.7.2019. Saatavissa: http://man7.org/linux/man-pages/man7/libc.7.html

[5] Alexander Larsson: Flatpak – a history. Viitattu 9.8.2019. Saatavissa: https://blogs.gnome.org/alexl/2018/06/20/flatpak-a-history/

[6] Flatpak Documentation – Basic concepts. Viitattu 9.8.2019. Saatavissa: http://docs.flatpak.org/en/latest/basic-concepts.html

[7] Flatpak Documentation – Available runtimes. Viitattu 12.8.2019. Saatavissa: http://docs.flatpak.org/en/latest/available-runtimes.html

[8] Valentin David: Freedesktop SDK cross compilers. Viitattu 20.8.2019. Saatavissa: https://valentindavid.com/posts/2019-06-25-freedesktop-sdk-cross-compiler/

[9] Flatpak GitHub wiki – Sandbox. Viitattu 22.8.2019. Saatavissa: https://github.com/flatpak/flatpak/wiki/Sandbox

[10] Flatpak GitHub wiki – Filesystem. Viitattu 24.8.2019. Saatavissa: https://github.com/flatpak/flatpak/wiki/Filesystem

[11] Linus Torvalds: Re: [Regression w/ patch] Media commit causes user space to misbahave – LKML. Viitattu: 28.9.2019. Saatavissa: https://lkml.org/lkml/2012/12/23/75

[12] FAQ glibc wiki - What is symbol versioning good for? Do I need it? Viitattu: 28.9.2019. Saatavissa: https://sourceware.org/glibc/wiki/FAQ#What_is_symbol_versioning_good_for.3F__Do_I_need_it.3F

[13] ArchWiki – Comparison of desktop environments. Viitattu: 28.9.2019. Saatavissa: https://wiki.archlinux.org/index.php/Comparison_of_desktop_environments

[14] Wikipedia – Qt version history. Viitattu: 28.9.2019. Saatavissa: https://en.wikipedia.org/wiki/Qt_version_history

[15] Wikipedia – GTK releases. Viitattu: 28.9.2019. Saatavissa: https://en.wikipedia.org/wiki/GTK#Releases

[16] Dirk Hohndel: GTK to Qt – a strange journey, Linux.conf.au 2014. Viitattu: 29.9.2019. Saatavissa: http://mirror.linux.org.au/pub/linux.conf.au/2014/Thursday/83-Gtk_to_Qt_-_a_strange_journey_-_Dirk_Hohndel.mp4

[17] Wireshark blog – We’re switching to Qt. Viitattu: 29.9.2019. Saatavissa: https://blog.wireshark.org/2013/10/switching-to-qt/

[18] Boost FAQ – How can the Boost libraries be used successfully for important projects? Viitattu: 4.10.2019. Saatavissa: https://www.boost.org/users/faq.html

[19] Linux kernel documentation - _stable_api_nonsense. Viitattu: 4.10.2019. Saatavissa: https://www.kernel.org/doc/Documentation/process/stable-api-nonsense.rst

[20] Flatpak Documentation – Repositories. Viitattu: 4.10.2019. Saatavissa: http://docs.flatpak.org/en/latest/repositories.html

[21] Flatpak Documentation – Under the Hood. Viitattu: 4.10.2019. Saatavissa: http://docs.flatpak.org/en/latest/under-the-hood.html

[22] Flatpak Documentation – Requirements & Conventions. Viitattu: 5.10.2019. Saatavissa: http://docs.flatpak.org/en/latest/conventions.html

[23] Flathub – Gnome Dictionary manifest. Viitattu: 5.10.2019. Saatavissa: https://github.com/flathub/org.gnome.Dictionary/blob/d9cd95cf16870c05c3153ca4dc423ba8f36c64ad/org.gnome.Dictionary.json

[24] Flatpak Documentation – Sandbox Permissions. Viitattu: 6.10.2019. Saatavissa: http://docs.flatpak.org/en/latest/sandbox-permissions.html

[25] Flatpak Documentation – Manifests. Viitattu: 6.10.2019. Saatavissa: http://docs.flatpak.org/en/latest/manifests.html

[26] Flatpak Documentation – Python. Viitattu: 6.10.2019. Saatavissa: http://docs.flatpak.org/en/latest/python.html

[27] Snapcraft documentation – Releasing to the Snap Store. Viitattu: 6.10.2019. Saatavissa: https://snapcraft.io/docs/releasing-to-the-snap-store

[28] Flathub GitHub wiki – App Submission. Viitattu: 6.10.2019. Saatavissa: https://github.com/flathub/flathub/wiki/App-Submission

[29] VirtualBox manual – Installing on Linux Hosts. Viitattu: 6.10.2019. Saatavissa: https://www.virtualbox.org/manual/ch02.html#externalkernelmodules

[30] Flatpak Documentation – Reasons to use Flatpak. Viitattu: 6.10.2019. Saatavissa: http://docs.flatpak.org/en/latest/introduction.html#reasons-to-use-flatpak

[31] Flatpak GitHub – Proposal: Document clear plan for EOL of runtimes. Viitattu: 6.10.2019. Saatavissa: https://github.com/flathub/flathub/issues/364

[32] ScotXW: Illustration of Linux kernel interfaces. Viitattu: 8.10.2019. Saatavissa: https://commons.wikimedia.org/wiki/File:Linux_kernel_interfaces.svg CC BY-SA 3.0 -lisennsin alaisuudessa.

[33] ScotXW: The GNU C Library is a wrapper around the system calls of the Linux kernel. Viitattu 8.10.2019. Saatavissa: https://commons.wikimedia.org/wiki/File:Linux_kernel_System_Call_Interface_and_glibc.svg CC BY-SA 3.0 -lisennsin alaisuudessa.