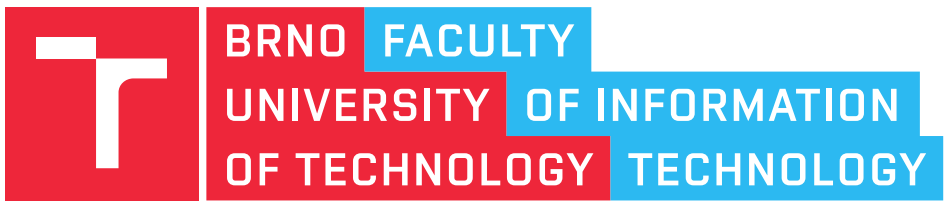
# Nasazení programů

Jaroslav Dytrych

Fakulta informačních technologií Vysokého učení technického v Brně
Božetěchova 1/2. 612 66 Brno - Královo Pole
dytrych@fit.vutbr.cz




7. dubna 2021

# Obsah přednášky

- Nasazení aplikací
- Instalátory
- Součásti instalace
- Tvorba instalátorů
  - Balík pro Linux
  - Instalátor pro Windows

# Nasazení aplikací u zákazníka

- Nasazení výrobcem SW
  - Programátor fyzicky přijde k zákazníkovi a SW nainstaluje
  - Typicky u složitého nasazení specializovaných aplikací
- OEM verze (Original Equipment Manufacturer)
  - SW je předinstalován „výrobcem" počítače
  - Vazba na HW (problém při výměně MB, CPU, HDD, . . . )
- Krabicová verze
  - Zákazník si koupí krabici s instalačním médiem a příručkou
  - Dříve běžné, dnes méně časté (drahé řešení)
- Stažení z webu a instalace
  - Zákazník si SW sám stáhne a nainstaluje
  - Platba za stažení nebo za licenční klíč
- Online instalace
  - Zákazník si SW instaluje přímo ze sítě
  - Google Play, Microsoft Store, Apple Store
- Stažení z webu bez instalace (přenositelná verze)
  - Zákazník si stáhne spustitelnou verzi aplikace, která se neinstaluje (přenositelné binární soubory – např. Java)

# Instalátor

- Automatizuje úkony instalace
  - Rozbalení na cílové umístění
  - Symlink/spustitelný soubor v `PATH`
  - Umístění do menu
  - Ikonka na ploše
  - Výchozí konfigurace
  - ...
- Zjednodušuje instalaci (může ji zvládnout i laik)
- Měl by řešit závislosti:
  - upozorněním uživatele (např. „Tento program vyžaduje běhové prostředí jazyka Java, které si můžete zdarma stáhnout ...“)
  - automatizovanou instalací (vhodné potvrzení uživatelem)
- Může být součástí:
  - instalačního balíku (např. samorozbalovací archiv)
  - operačního systému či jeho nástrojů – např. debianní balík či balík pro MSI (Microsoft Installer, nyní Windows Installer)
    - Instalátorem programu pak často (nepřesně) nazýváme onen balík

# Proč instalátor?

- Manuální kopírování souborů na cílové umístění je pracné
  - Vysoká cena práce
  - Problém s odinstalací (co bylo kam nainstalováno?)
- Zákazník typicky nezvládne složité úkony instalace a není ochoten kvůli malému programu zvát „drahého“ specialistu
- Přenositelné verze
  - Problémy se závislostmi – vše musí být přibaleno, nebo to musí manuálně instalovat zákazník
  - Menší uživatelská přívětivost (program není v nabídkách apod.)

# Skript vs. instalátor

- Instalační skript (`./configure; make; make install`)
  - Potřebuje interpret skriptovacího jazyka, často i další nástroje
  - Jednorázově provede sadu úkonů, typicky nedochází k žádné registraci do DB ani k instalaci odinstalátoru
  - Veškeré závislosti a konflikty řeší sám – ignoruje-li konflikt, může poškodit jiný SW
- Instalátor
  - Nepotřebuje interpret skriptovacího jazyka a pomocné nástroje – jedná se o samostatný spustitelný soubor, nebo pro spuštění stačí správce balíků
  - Typicky zaregistruje SW v DB operačního systému
    - Odinstalace přes správce aplikací
    - Dotazem do DB lze zjistit závislosti a konflikty
  - Typicky nainstaluje odinstalátor
    - Odinstalátor by měl odinstalovat pouze to, co instalátor nainstaloval, a to beze zbytku (s výjimkou uživatelských dat)

# Součásti instalace

- Binární soubory programu (nikoliv zdrojové)
- Ikony, výchozí konfigurace a další pomocné soubory
- Spouštěcí skript
- Knihovny
- Zásuvné moduly
- Odinstalátor
- Seznam akcí instalátoru
- Licenční ujednání
- ...

# Obsah instalace – příklady

- Windows

```
MyApp
|- MyApp.exe
|- *.dll
|- *.png
```

- Mac

```
MyApp.app
|- Contents
    |- Resources
    |   |- *.png
    |- Frameworks
    |   |- *.dylib
    |- MacOS
    |   |- MyApp
    |- Info.plist
    |- PkgInfo
```

- Liší se pro různé operační systémy
- Nutno zahrnout zásuvné moduly (plugins)
- Různé cesty
  - Spouštěcí skript pro nastavení
- Liší se dle programovacího jazyka – např. pro C++
  - Knihovna jazyka C++
  - Knihovna GNU C (pokud není v základu OS)
  - Dynamické knihovny zvoleného frameworku
  - Zásuvné moduly v podadresářích
  - ...

# Zásuvné moduly

- Windows
  - Podadresář se zásuvnými moduly

```
imageformats/
|- qico4.dll
|- qjpeg4.dll
```

- Mac
  - Všechno v jednom spouštěcím adresáři
  - Zásuvné moduly v podadresáři Frameworks
  - Např. pro Qt nástroj macdeployqt vytvoří „disk image“ (obdoba balíku), včetně přibalení zásuvných modulů

# Spouštěcí skript

- Činnost
  - Nastavit proměnné prostředí
    - `Path` (Windows)
    - `PATH` a `LD_LIBRARY_PATH` (Linux)
  - Spustit binární soubor programu
- Windows
  - Skript je vhodné konvertovat na spustitelný soubor (Vbs2Exe)
  - Zásuvné moduly lze přidat do cesty např. takto (Visual Basic):

```
Set objShell = CreateObject("WScript.Shell")
Set objEnv = objShell.Environment("System")
Set envProcess = objShell.Environment("Process")
oldSystemPath = objEnv("PATH")
pathToAdd = ";" & objShell.CurrentDirectory & "\lib"
newSystemPath = oldSystemPath & pathToAdd
envProcess("Path") = newSystemPath
objShell.Run(".\bin\app.exe")
```

- Linux

```
#!/bin/bash
LD_LIBRARY_PATH="~/myprogdir/lib/:$LD_LIBRARY_PATH"
export LD_LIBRARY_PATH
~/myprogdir/myprogram "$@"
```

- Java
  - Přenositelné binární soubory
  - Bajtkód interpretovaný virtuálním strojem (JVM)
  - Archiv `jar`
    - Zkompilovaná aplikace
    - Závislosti
    - Manifest
  - Manifest
    - `Class-Path`
    - `Main-Class`

```
Manifest-Version: 1.0
Ant-Version: Apache Ant 1.9.6
Created-By: 1.8.0_111-b14 (Oracle Corporation)
Class-Path: lib/swing-layout-1.0.4.jar
X-COMMENT: Main-Class will be added automatically ...
Main-Class: appname.main
```

- Python
  - Skripty
    - Nutno mít nainstalovaný interpret
    - Uživatel si může prohlížet (a měnit) zdrojový kód
  - Zapouzdření skriptu do binárního souboru
    - Skryje zdrojové soubory
    - Přibalí závislosti a interpret
    - Binární soubor už není přenositelný – nutno vygenerovat pro různé platformy
  - PyInstaller (`sudo pip install pyinstaller`)

    ```
    pyi-makespec main.py
    pyinstaller main.spec
    ```

    - Vytvoří spustitelný binární soubor pro danou platformu
    - Není to instalátor!

# Možnosti vytvoření instalátoru

- Linux
  - Debianní balíček (Debian package) – přípona `.deb`
  - RPM
  - Balíček se závislostmi (Snap apod.)
  - Spustitelný binární soubor
  - Skripty
- Windows
  - Balíček pro MSI (Microsoft Installer, nyní Windows Installer)
  - Balíček Universal Windows Platform (UWP)
    - `.msix`/`.appx` pro 1 platformu
    - `.msixbundle`/`.appxbundle` pro více platforem
  - Instalační program
  - Spustitelný samorozbalující se archiv
- Mac
  - Obraz disku `.dmg` (Apple Disk Image)
  - Balíky pro OS X Installer `.pkg` (Apple software package)
- Android
  - Android Application Package (APK)
- iOS
  - iOS App Store Package (`.ipa`)

# Volba instalátoru pro Linux

- Typicky distribuční balíček
  - Obvykle pro konkrétní verzi konkrétní distribuce
  - Pohodlná instalace, snadná odinstalace
  - Řeší závislosti (sdílené v rámci systému)
- Balíček se závislostmi (Snap apod.)
  - Oproti distribučnímu balíčku jednodušší a spolehlivější instalace
  - Závislosti pro každý program zvlášť (místo na disku, RAM, ...)
- Časté je využití GNU Autoconf

  (`./configure; make; make install`)
  - Nezávislé na distribuci
  - Nutno doinstalovat nezbytné závislosti
  - Problematická odinstalace
  - „Nepořádek" v systému
- Spustitelný binární soubor či samorozbalující se archiv jsou neobvyklé
  - Důvěryhodnost
  - Problematické řešení závislostí
  - Často problematická automatizace instalace

# Nástroje pro vytvoření instalátoru

- Linux
  - Balíčkovací systémy (dh_make, dpkg, . . . )
- Windows
  - MS Visual Studio
  - InnoSetup
  - NSIS (Nullsoft Scriptable Install System)
  - WiX toolset
- Mac
  - Installer for Mac OS X
  - VISE X
- Multiplatformní
  - BitRock InstallBuilder
  - InstallAnywhere
  - IzPack
  - InstallJammer (2011)
  - **Multiplatformní nástroj neznamená, že vždy vytvoří multiplatformní instalátor!**
    - Lze připravit instalátor pro Linux s binárními soubory pro Windows, ale nebude to fungovat

- Typy balíků
  - Debian
  - Red Hat (RPM)
- Převody balíků
  - Alien

```
fakeroot alien myPackage.rpm
fakeroot alien -c myPackage.rpm
fakeroot alien --to-rpm myPackage.deb
fakeroot alien --to-rpm -c myPackage.deb
```

    - `-c` se pokusí převést i skripty po instalaci, před odebráním apod. Ve výchozím nastavení se zahodí, protože na jiném systému mohou mít nečekané efekty
    - fakeroot umožňuje změny vlastníků a práv pro soubory v balíčku

# Vytvoření debianího balíku

- Stažení potřebných nástrojů

```
sudo apt-get install build-essential autoconf
automake autotools-dev dh-make debhelper
devscripts fakeroot cdbs lintian pbuilder
```

- Příprava programu k balíčkování
- Vytvoření podadresáře debian
- Konfigurace balíku
- Vytvoření balíku

# Příprava programu k balíčkování

- Složka musí mít název ve formátu `nazevProgramu-verze`
- Makefile by měl být napsaný podle konvencí GNU
  `https://www.gnu.org/prep/standards/html_node/Makefile-Conventions.html`
- Makefile musí mít cíl `install` využívající `DESTDIR`

```
.PHONY: all clean install
...
install: meancalc
    mkdir -p $(DESTDIR)/usr/bin
    install $(INSFLAGS) meancalc $(DESTDIR)/usr/bin/
```

- Je rozumné připravit ikonku (`.xpm` 32 x 32 px) a spouštěč na plochu/do menu (`mujProgram.desktop`):

```
[Desktop Entry]
Categories=Qt;Game;
Exec=kufr
GenericName=Simple Game
Icon=/usr/share/pixmaps/kufr.xpm
Name=Kufr
Type=Application
X-KDE-StartupNotify=false
```

- Vše zabalit do nadřazeného adresáře (`.tar.gz`)

- Vytvoření podadresáře debian:

```
dh_make -e <email> -n -s -c gpl -f ../mujProgram-01.tar.gz
```

`-c license` (gpl, gpl2, gpl3, lgpl, lgpl2 lgpl3, artistic, apache, bsd, mit)

`-n` (nativní – bez přidání čísla revize pro Debian a generování `.orig`)

`-s` (1 binární soubor) nebo `-m` (více)

# Konfigurace balíku

- Důležité soubory podadresáře debian

  `control` – metadata (název, správce, závislosti, . . . )

  `copyright` – licence, povinně musí být uvedena

  `changelog` – seznam změn v jednotlivých verzích (povinný)

  `install` – co kam nakopírovat

  `rules` – pravidla pro vytvoření balíčku (v podstatě speciální Makefile)

- Soubor `control`
  - Nutně musíme nastavit závislosti (pro build a pro běh)
  - Je vhodné nastavit popis

```
Source: kufr
Section: Games
Priority: extra
Maintainer: Jaroslav Dytrych <dytrych@fit.vutbr.cz>
Build-Depends: debhelper (>= 8.0.0),
               qtdeclarative5-dev, qt5-qmake
Standards-Version: 3.9.4

Package: kufr
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: Kufr
 Simple game
```

- Soubor `install`
  - Soubory pro nakopírování do systému
  - Např. pro přidání položky do menu:

```
install/kufr.xpm /usr/share/pixmaps
install/kufr.desktop /usr/share/applications
```

- Soubor `rules` – často stačí vygenerovaný obsah, např.:

```
#!/usr/bin/make -f
include /usr/share/cdbs/1/rules/debhelper.mk
include /usr/share/cdbs/1/class/qmake.mk
```

nebo např.:

```
#!/usr/bin/make -f
# -*- makefile -*-
%:
        dh $@
```

# Trigger skripty

- Po vygenerování mají příponu `.ex` (example) – před použitím odmazat
- Vhodné pro vytvoření chybějících složek, ldconfig (přidání cesty do `/etc/ld.so.conf` a spuštění `ldconfig`) apod.
- Před/po odebrání úklid, ale opatrně (**nikdy nesmíte odebrat to, co jste nepřidali!**)

- `preinst` – před instalací
- `postinst` – po instalaci
- `init.d` – po startu OS (démon)
- `prerm` – před odebráním (odinstalováním)
- `postrm` – po odebrání

# Postup vytvoření balíku – rekapitulace

- Připravit program do adresáře s názvem ve formátu `nazevBaliku-verze`
- Zabalit do nadřazeného adresáře (`.tar.gz`)
- Vytvořit adresář `debian`

  `$dh_make -e example@example.com -f ../yourpackage.tar.gz`

  `-c license` (gpl, gpl2, gpl3, lgpl, lgpl2 lgpl3, artistic, apache, bsd, mit)

  `-n` (nativní – bez přidání čísla revize pro Debian a generování `.orig`)

  `-s` (1 binární soubor) nebo `-m` (více)
- Upravit soubory `control` a `install` (případně i další)
- Případně upravit trigger skripty pro instalaci/odinstalaci
- Vytvořit balík

  `dpkg-buildpackage -rfakeroot`

  `-uc` (nepodepisovat `.changes`)

  `-b` (binární balík – bez zdrojových souborů)

- QMake může vygenerovat `Makefile` s příslušným cílem:

```
...
TARGET = kufr
...
# install
target.path =    $$PREFIX/bin
INSTALLS += target
```

- Obdobně lze využít CMake – ukázka za chvíli . . .

# Postup vytvoření balíku – Qt FAQ

- Nesnažte se upravit `Makefile` vygenerovaný pomocí qmake – při generování balíku se automaticky smaže a vygeneruje znovu.
- Jak vybrat verzi Qt?
  - Vždy je automaticky spuštěn `qmake` bez parametrů – je potřeba nastavovat prostředí.
  - `QT_SELECT=qt5 dpkg-buildpackage ...`
- Qt – závislosti
  - Pro build min. `libqt4-dev` a `qt4-qmake` nebo `qtdeclarative5-dev` a `qt5-qmake`
  - Pro běh lze nechat vygenerovat automaticky

# Postup vytvoření balíku – Java

- 2 možnosti:
  - dh_make – jako předchozí příklady
  - mh_make – speciálně pro Javu
- **dh_make**
  - `Makefile` spustí Maven a v cíli `install` nainstaluje
  - Kromě samotného balíku v Javě (`.jar`) je potřeba nainstalovat ještě spouštěcí skript
    - `exec java -jar "./kufr-0.0.1-SNAPSHOT.jar"`
  - Maven využívá repozitář `~/.m2/repository` – s fakeroot `/root/.m2/repository` – ale tam nelze zapisovat!
    - `-Dmaven.repo.local=../repo/`
- **mh_make**
  - `DEBFULLNAME="Jaroslav Dytrych" DEBEMAIL="dytrych@fit.vut.cz" DEBLICENSE="gpl" mh_make`
  - Interaktivní skript, který požádá o spoustu informací (většinou lze zvolit doporučenou výchozí volbu).
  - Balík se instaluje jako knihovna v Javě – s ikonkou a spouštěčem na plochu je potřeba nainstalovat i upravený spouštěcí skript do `/usr/bin`

- Vytvoření sktruktury

```
yum install @development-tools
yum install fedora-packager
/usr/sbin/useradd makerpm
usermod -a -G mock makerpm
rpmdev-setuptree #when logged as makerpm user
```

- Struktura

```
|- BUILD
|- BUILDROOT
|- RPMS
|- SPECS
|- SOURCES
|- SRPMS
```

# Postup vytvoření balíku

- Zabalit vše do adresáře SOURCES
- Zahrnout i všechna data, obrázky, ...
- Napsat skript `PACKAGENAME.spec`, kde je vše potřebné
- Zkompilování balíku

```
rpmbuild -ba program.spec
```

# Spec soubor

- Vytvoření šablony (rpmdev-newspec)
- Vyplnění sekcí instalátoru

```
%prep
#rozbalí vše z adresáře SOURCES do adresáře BUILD
%build
#kompilace adresáře BUILD do BUILDROOT
%pre
#co se má stát před instalací
%install
#kam se má co překopírovat při instalaci
%post
#co se má udělat po instalaci
%preun
#co se stane před odinstalací balíku
%postun
#po odinstalaci balíku
```

Děkuji za pozornost!