# Free & Open Source software

Lukáš Zapletal

Liberix o.p.s.

14. 9. 2006

# Obsah prezentace

# Jazyky C, C++

- jazyk C jako základ UNIXu a Linuxu, jazyk C++ se prosazuje na UNIXech pomaleji
- tyto jazyky jsou standardizovány institucemi ANSI (American National Standards Institute) a ISO (International Organization for Standardization)
- moderní aplikace pro X Window jsou často psány v C++
- dostupné překladače C/C++ pro MS Windows: MS Visual C++, Borland C++, GCC (GNU Compiler Collection) – Cygwin, Mingw, LLC, ICC (Intel C Compiler) a jiné
- dostupné překladače C/C++pro Linux: GCC (GNU Compiler Collection), ICC (Intel C Compiler) a jiné
- překladače jsou konzolové aplikace, patří k nim další nástroje jako je například GNU Makefile

# Jazyk C v Linuxu

- GNU C Compiler ze sady GCC je odborníky označován za jeden z nejlepších kompilátorů jazyka C vůbec
- implementuje normy ISO (C89/C90, C99) a GNU rozšíření, lze libovolně přepínat
- GCC neobsahuje základní knihovnu jazyka C, je tedy potřeba knihovna GNU C (glibc) – součástí systému
- pozor na rozšíření jazyka a knihoven firmy Microsoft, programátoři z Windows si musejí dát pozor na nekompatibility
- Objective-C – alternativa k C++, vyšší flexibilita, nižší nároky na překladač, podpora ze strany firmy Apple (platforma Cocoa)
- literatura: Pavel Herout. Jazyk C. Nakl. Kopp. (2 díly)

# Jazyk C++ v Linuxu

- GNU C++ Compiler ze sady GCC
- implementuje normu ISO/IEC 14882 z roku 1998 včetně úprav z roku 2003 a pozdějšího TR
- GCC neobsahuje základní knihovnu jazyka C++, je tedy potřeba knihovna The Standard GNU C++ Library (libstdc++) – součástí systému
- velmi úspěšná je knihovna Boost (www.boost.org)
- Objective-C++ – pouze syntaxe C++ k Objective-C (firma Apple)
- literantura: Bjarne Stroustrup. C++ Programming Language.

# GNU Makefile

- slouží k sestavování libovolných projektů, ke kterým existuje nějaký překladač či program, který přijímá vstupní soubory jako své parametry
- lze ho využít nejen k programování, ale sestavování libovolných věcí (diplomová práce psaná v Latexu aj.)
- Makefile – popisuje, jakým způsobem (jakými příkazy) se sestaví daný projekt, obsahuje závislosti mezi soubory, cesty ke knihovnám a zdrojům
- make – příkaz, který provádí dané akce

# Autoconf, Automake

- GNU make neřeší přenositelnost – například funkce v jazyce C mají na různých systémech jiné názvy, jiné parametry, mohou být definovány v jiných hlavičkových souborech nebo linkovány do jiných knihoven
- make neřeší jednotný přístup k automatické konfiguraci softwaru (debug, release, další definice)
- programátoři z GNU projektů začali psát shell skripty, které se snažily uhodnout, jaké verze knihoven, hlavičkových souborů, funkcí je na daném systému
- to byl základ skriptů configure

# Autoconf, Automake

- configure – skript napsaný v shellu, který prohlédne systém a vytvoří config.h a soubor(y) Makefile pro celý projekt
- poté může uživatel spustit make, program sestavit a nainstalovat
- typický proces instalace balíčku s programem využívajícím configure skript: rozbalení, spuštění configure, make, make install
- skript configure vytváří obvykle následující cíle: all, install, uninstall, clean, distclean, check, dist
- instalace se provádí pomocí tzv prefixu (obvykle /usr/local)

# Autoconf, Automake

- skript configure detekuje mnoho nastavení, ale dá se ovlivňovat
- parametry: prefix, CC, CCFLAGS, CXX, CXXFLAGS, LDFLAGS, CPPFLAGS a jiné (blíže pak configure –help)
- configure také usnadňuje cross-compilation (parametry build, host, target), balíčkování, vnořené podprojekty
- configure skript nezačíná generovat config.h a Makefile od píky, ale předkládáme mu šablony (config.h.in, Makefile.in)

# Autoconf, Automake

- configure je velice komplikovaný skript, tudíž se nevytváří přímo, ale je generován programem Autoconf
- Autoconf vytváří ze šablon configure.ac, Makefile.am šablony Makefile.in, config.h.in a samotný skript configure
- v souboru configure.ac definujeme, co má vygenerovaný soubor configure testovat a zjišťovat (jaké knihovny, jaké má ověřit názvy funkcí)
- pro tyto účely se používá makrojazyk m4
- autoreconf –install – vygeneruje configure a in šablony, nakopíruje několik podpůrných souborů (aclocal.m4, depcomp, missing, install-sh)

# Autoconf – nástroje

- autoconf – vytváří configure z configure.ac
- autoheader – vytváří config.h z configure.ac
- autoreconf – spouští tyto nástroje ve správném pořadí (aclocal, autoconf, autoheader, automake)
- autoscan – kontroluje projekt na chyby v přenositelnosti
- autoupdate – aktualizuje makra v configure.ac
- ifnames – sbírá makra #if/#ifdef
- autom4te – srdce celého procesu, napsáno v m4

# Automake - nástroje

- automake – vytváří Makefile.in z Makefile.am a configure.ac
- aclocal – vytváří aclocal.m4 pro configure.ac

# Autoconf, Automake

- obvyklý postup: vytvoříme soubory configure.ac a Makefile.am, spustíme autoreconf –install, při změně nějakého konfiguračního souboru pak spustíme příslušný nástroj, který přegeneruje potřebné
- ano, je to komplikované, ale takhle se to prostě dělá – usnadňujete pak práci nejen vývojářům či uživatelům, ale také balíčkářům
- některé IDE prostředí jsou schopny s Autotools spolupracovat
- v Linuxu je na tom výborně například KDevelop (bohužel bývá někdy nestabilní a padá)

# Autoconf, Automake

- možné alternativy: qmake, cmake (různé ”nadstavby” make)
- SCons – modulární sestavovací systém napsaný v Pythonu
- Apache Ant, Apache Maven – napsané v Javě (ale široké použití)
- a mnoho dalších

# Autotools v praxi

- stáhneme balíček se zdrojovým kódem, ten je již ”předpřipravený” a je v něm vytvořen skript configure
- tar -xvzf program-1.0.0.tar.gz && cd program-1.0.0
- ./confgure [parametry] (například –prefix=/opt/program)
- make
- make install
- využijeme v případě, že chceme program upravovat, nebo když není dostupný binární balíček pro naši distribuci (vyšla například nová verze)

# Bash

- Bourne Again Shell
- drtivá většina linuxových distribucí používá právě Bash, většina má startovací (a jiné) skripty napsány právě v Bashi
- Bash musí znát každý linuxový geek (administrátor), velice mocný nástroj zejména za pomoci filtrů (sed, head, tail...)
- jednoduchá ale velmi mocná syntaxe (popsána v manuálové stránce)
- funguje také na MacOS, Windows (Cygwin), DOSu (DJGPP)
- příklad: for f in ‘ls *mp3‘ do oggenc *f*; *rm*f; done

# Perl

- Practical Extraction and Report Language (název vznikl jinak, ale toto je praktické)
- netypový jazyk, interpretovaný
- autor: Larry Wall (programátor, lingvista, autor programu patch)
- Perl najdete na každém UNIXu, současná verze Perl 5.8
- přednosti: bohatá syntaxe, dostupnost, vestavěná (rychlá) implementace pro regulární výrazy, velký počet modulů (CPAN), vhodný na zpracování textu (logů atd), kvalitní dokumentace
- zápory: bohatá syntaxe, pomalejší interpret
- Perl 6 – vyvíjí se od roku 2000, zatím se pracuje na VM (Parrot)

# Python

- velmi úspěšný typový, objektově orientovaný jazyk, autor: Guido van Rossum
- nachází uplatnění při psaní webových aplikací (Google, Seznam.cz), desktop aplikací (BitTorrent) i krátkých skriptů či nástrojů (portage)
- podobně jako Java probíhá překlad do bytecode, existuje více intepreterů (CPython, JPython)
- klady: dobrá podpora OOP, přehlednost kódu, rychlá křivka učení, výborná knihovna, dobrá rozšiřitelnost, možnost imperativního přístupu
- zápory: nekonzistentní API
- Python 3 (neboli 3000) – nebude zpětně kompatibilní, zatím vzdálená budoucnost (sběr požadavků)

# PHP

- primitivní imperativní jazyk
- vznikl pro psaní skriptů vsazených do HTML (PHP: Hypertext Preprocessor)
- po jazycích C a Java zřejmě třetí nejoblíbenější jazyk (spolu s C++ a Visual Basicem)
- postupně se vypracoval z jazyku pro jednoduché dynamické stránky v jazyk s OOP podporou pro psaní větších webových aplikací (Zend, CakePHP...)
- výhody: velké rozšíření, snadnost nasazení, jednoduchá syntaxe, dobrá dokumentace a podpora
- nevýhody: beztypovost, nemá jmenné prostory, časté změny v jazyku, nekompatibility mezi verzemi

# Další jazyky

- Java, C# – budeme se věnovat podrobněji
- Ruby – objektově orientovaný jazyk inspirovaný Perlem, Smalltalkem, Pythonem, úspěch jako jazyk pro psaní webů (Ruby On Rails)
- TCL – vyslovujeme ”tikl”, jednoduchý skriptovací jazyk, původně se používal pro CGI a pomocí toolkitu TK také pro psaní GUI aplikací (odtud TCL/TK),je velmi snadno rozšiřitelný a stále se používá jako například ”vsazený” jazyk

# Další jazyky

- AWK – vyslovujeme ”ók”, jakyk pro manipulaci s textem, možnost proudového zpracování, často se používá také s nástrojem SED (stream editor)
- LISP – druhý nejstarší stále se používající jazyk (po Fortranu), funkcionální (ale také procedurální a objektově orientovaný), derivátem je například Scheme, využití jako ”vsazený” jazyk, ve výzkumných ústavech, na školách ale také prakticky (Emacs)
- další jazyky dostupné na Linuxu: Smalltalk, Fortran, Cobol, Algol, Pascal, BASIC nebo i assembler (gas, nasm)

# Toolkity

- GTK+ – hlavní knihovna nad kterou je postaveno GNOME, jazyky C, C++, .NET, Java, skriptovací jazyky, používá jej například Mozilla
- QT – hlavní knihovna nad kterou je postaveno KDE, dříve komerční, poté komerční jen pro Windows, dnes svodobná (Trolltech prodává EmbeddedQT – pro ”malá” zařízení)
- TK – ”jednodušší” toolkit, není tak hezký
- FLTK/FOX – odlehčené toolkity, také nevypadají příliš hezky, ale nečerpají příliš systémových prostředků
- Motif – historická záležitost (Solaris), open-source implementace Lesstif

# Toolkity

- wxWidgets – vysokoúrovňový toolkit, multiplatormní, na Windows používá MFC, na Unixech GTK+ nebo Motif, na MacOS zase Cocoa
- XUL – pro produkty Mozilly, opět vysokoúrovňové API, postaveno nad GTK+, MVC přístup
- AJAX – hit dnešních dnů – toolkity pro vytváření webových aplikací jako desktopových aplikací: Google Web Toolkit, DoJo Toolkit, Tibco General Interface
- pokud neznáte, zkuste například stránky projektu http://dojotoolkit.org (jsou tam dema)
- nebo vynikající Jabber/ICQ/MSN webový klient: http://www.meebo.com

# Platforma Mono

- poskytuje nezbytný software pro vývoj a spouštění .NET aplikací na Linuxu, Solarisu, MacOSu, Windows a UNIXu, projekt sponzorován Novellem, samozřejmě Open Source
- Mono je multiplatfomní implementací podle ECMA/ISO standardů, podporuje .NET 1.2 (verze 2.0 se dokončuje), je možné kompilovat jazyky C#, Java, Python, Boo (jakýsi fork Pythonu pro .NET)
- mono – VM, mcs – kompilátor, MonoDevelop – vývojové prostředí, zatím nestabilní (betaverze)

# Platforma Mono

- drtivá většina jmenného prostoru System je již implementována, nicméně zatím chybí hodně dokumentace (u mnoha tříd najdete na stránce http://www.go-mono.com/docs/ pouze ”To be added.”)
- k dispozici jsou kromě standardních tříd také: GTK# (Cocoa#, Windows.Forms – zatím se implementuje, musí se zčásti emulovat), Mono.TAO (OpenGL), Mono.Directory (LDAP a spol), Mono.Data (databáze), Mono.Cairo (2D rendering), Mono.Posix, Mono.Remoting (CORBA, sockety, unixové sockety), Mono.Http, Mono.XML...
- v současné době je asi nejlepší psát desktopové aplikace pomocí toolkitu GTK#, existuje také poměrně stabilní podpora pro psaní ASP.NET aplikací pomocí Apache modulu mod_mono (1.1, verze 2.0 je ve vývoji)
- aplikace psaná pro Mono.GTK pojede i na Windows (a jinde)

# Platforma Mono

- mono hodně sponzoruje Novell (staví na něm některé své aplikace)
- platformu .NET se snaží prosazovat také pracovní prostředí GNOME (do nových verzí už se dostalo několik aplikací psaných v .NETu)
- ačkoli MonoDevelop ještě stabilní není, virtuální mašina je dostatečně stabilní (jede v ní například Java prostředí Eclipse)
- namátkou aplikace: F-Spot, Tomboy, Blam, Beagle, Muine, iFolder
- také běží na Nokii 770

# Platforma Java

- vynikající platforma pro vývoj multiplatformních aplikací
- Java se usídlila na serverech (kde dominuje)
- postupně proniká i na pracovní plochy uživatelů
- v Linuxu je stejně dobrá nabídka vývojových nástrojů, knihoven a aplikací v Javě, jako na Windows
- praktické ukázky: Eclipse IDE, Netbeans IDE, IntelliJ IDEA
- představení Rich Client Platforms: Eclipse, Netbeans

# VCS

- systémy pro správu verzí
- dělíme na nedistribuované (centrální repozitář – RCS/CVS, Subversion, Rational/IBM ClearCase, Preforce, MS SourceSafe) a distribuované (GIT, GNU Arch, Bazaar, Darcs, Monotone)
- základní pojmy: repozitář, checkout, checkin (commit), update, conflict, resolve, revision, tak, branch, version
- budete mít na zvláštní prezentaci (Mgr. Jan Outrata)
- nástroje diff a patch – zasloužily se o úspěch Open Source

- C/C++ – standardní jazyky v Linuxu
- Autotools machinery
- skriptovací jazyky (Perl, Python, PHP)
- Mono – .NET pro Linux a jiné systémy (vč. Windows)
- Java – také skvělá volba