

Open Source Programování

<http://rtime.felk.cvut.cz/osp/>

Pavel Píša

<pisa@fel.cvut.cz>

<http://cmp.felk.cvut.cz/~pisa>

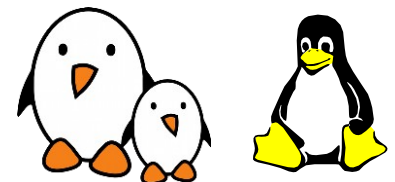
Michal Sojka

František Vacek

DCE FEL ČVUT



© Copyright 2004-2010, Pavel Píša, Michal Sojka, František Vacek,
Andrew Tridgell, Free-Electrons.com, GNU.org,
kernel.org, Wikipedia.org
Creative Commons BY-SA 3.0 license Latest update: 31. III 2010



- ▶ Následující doporučení a diskuze vychází z přednášky autora projektu Samba (Andrew Tridgell)
- ▶ Samba je FOSS implementace souborového, tiskového a autentizačního serveru kompatibilního s protokoly použitými Microsoftem pro tyto služby v MS Windows
- ▶ V současné době je to projekt využívaný po celém světě od firem, přes vládní instituce, univerzity po domácnosti
- ▶ Často je součástí síťových úložišť /NAS devices
- ▶ Projekt začal jako pokus o Unixový sever pro DOS v roce 1991
- ▶ V současné době na něm aktivně pracuje 15 členů v posledním roce okolo 40 dalších přispěvatelů, 25 změn/den
- ▶ Nepřímo zaměstnává/platí (podpora, integrace atd.) množství lidí

- ▶ Zápál nadšení
 - ▶ Mnoho projektů vzniklo v důsledku nadšení a zápalu jednoho vývojáře
 - ▶ Málo kdy zakladatel přemýšlí o všech souvislostech a náležitostech vedení FOSS projektu
- ▶ Je potřeba znát doporučení a recepty?
 - ▶ Obvykle to zjednodušuje život, ale dobrý kuchař improvizuje a hledá nové cesty.
- ▶ Je potřeba pomoc nebo motivace?
 - ▶ Je víc potřeba pomoc s napsáním první verze nebo pocit užitečnosti/motivace od uživatelů?
 - ▶ Pokud je potřeba pomoc, tak pomoc s volbou organizace a infrastruktury je velmi podstatná. Je dobře se tedy učit od jiných projektů a poučit se dobrými i špatnými volbami a zkušenostmi

- ▶ Co je cílem projektu
 - ▶ Není dobré na začátku přehánět (Unix × Multics)
 - ▶ Být světovou špičkou stojí čas – nejdřív je potřeba začít od malých věcí
- ▶ Jaká má být struktura projektu
 - ▶ Na začátku stačí velmi jednoduchá struktura
 - ▶ Je srozumitelnější a s ní méně potíží
 - ▶ Pokud je jen jeden správce, tak je začleňování změn jednoduché
- ▶ Jaká má být licence
 - ▶ Nemá smysl vymýšlet novou licenci, budou v ní chyby a nebude kompatibilní pro integraci kódu nebo jeho znovuvyužití
 - ▶ **Pozor**, rozhodnutí o licenci je velmi závažné, po integraci cizí práce ji lze většinou jen velmi těžko změnit a taková snaha může být důvodem ke přím, rozdělení a až zániku projektu
- ▶ Správa zdrojového kódu (repositář)
- ▶ Vlastní nebo veřejný předpřipravený projektový hosting (SF.net,)
- ▶ Bude potřeba e-mailová konference (mailing-list)? IRC? Web site?
- ▶ Bude potřeba systém pro zprávu chyb?

- ▶ Důležité je začít s něčím, co alespoň trochu chodí
 - ▶ Již před prvním zveřejněním je potřeba, aby bylo alespoň něco, co lze ukázat (výjimky – obtížná výzva, např ReactOS)
 - ▶ Když je již co nabídnout, tak je naděje na pozitivní odezvu od potenciálních uživatelů a přispěvatelů
 - ▶ Funkční kód neznamena perfektní kód
- ▶ Dodržování obvyklých postupů a konvencí pomůže
 - ▶ Učit se, jak podobné projekty řeší vývoj, překlad a úpravu kódu
 - linux-devel/Documentation/CodingStyle
 - GNU Coding Standards
 - Code Conventions for the Java Programming Language
 - ▶ Když to jde tak využít funkční postupy a i kód
- ▶ Na prvním dojmu záleží
 - ▶ Projekt musí být uživatelsky přístupný, snadná první instalace/překlad

- ▶ Počáteční oznámení
 - ▶ Projekt by měl být vložen do nějakého katalogu
obecně pro Unix a Linux je nejvhodnější **freshmeat.net**
důležité je i správné zařazení, kategorie/tagy
 - ▶ Oznámení by mělo být posláno do e-mailových konferencí, které se danou problematikou nebo podobnými projekty zabývají
 - ▶ Pozor, aby oznámení/styl nebyl považován za spam
myslete z pohledu druhých, co jejich projektům může váš projekt nabídnout a nebo se rozhodněte svět přesvědčit, že jste lepší
 - ▶ Co by vždy oznámení mělo obsahovat
 - ▶ K čemu je/může sloužit
 - ▶ V jakém jazyce je napsaný, na čem závisí
 - ▶ Na jakých cílových platformách by měl běžet
 - ▶ Jaká byla zvolena licence
 - ▶ Kde je možné se dozvědět více
 - ▶ Buďte poctiví, přiznejte, že je to třeba jen hra nebo naopak součást/podpora nějakého firemního řešení, jak předpokládáte, že bude vypadat další vývoj atd.

- ▶ Kladná odezva
 - ▶ Klíčovým faktorem je pozitivní komunikace a zpětná vazba k těm, co přispějí
 - ▶ Odpovězte na každý příspěvek, snažte se je povzbudit (Linus – a lazy bastard)
 - ▶ Snažte se odpovídat rychle, využijte IRC
- ▶ Vydání (Releases)
 - ▶ Vydávejte balíčky rychle a často
Eric S. Raymond: The Cathedral and the Bazaar
 - ▶ Použijte snapshot vydání, když to má cenu
dnes je nakonec výhodnější trvale zkompilovatelný kód z repositáře
 - ▶ Údržba seznamu změn ke každému vydání (Changelog)
 - ▶ Vždy uveďte autora každé změny (i nápadu)
Ingo Molnar: credits Con Kolivas, for pioneering the fair-scheduling approach
- ▶ Podporujte diskusi
 - ▶ Ptejte se druhých na jejich názory
 - ▶ Poslouchejte a analyzujte všechny odezvy

- ▶ Je potřeba začít uvažovat o dalších záležitostech
 - ▶ Vytváření balíčků pro hlavní distribuce a platformy
 - ▶ Zvážit přípravu binárních balíčků
 - ▶ Má smysl napsat článek(y) do časopisů/na webové portály
 - ▶ Má smysl vést k vývoji blog
- ▶ Struktura projektu
 - ▶ Je potřeba průběžně testovat funkčnost projektu
noční můra regrese
 - ▶ Má smysl nějak projekt formalizovat
 - ▶ Určitě je to potřeba nejdříve prodiskutovat
 - ▶ Vytvořit zájmovou skupinu/konsorcium
 - ▶ V některých případech je to jasné již při založení
 - ▶ Je potřeba začít definovat role (developer, release manager, atd.)

- ▶ FOSS projekty mohou růst velmi rychle
 - ▶ Nepřerůstá množství práce možnosti jednoho/daného člena projektu
 - ▶ Je možné projekt rozdělit na funkční celky/samostatné projekty
 - ▶ Má smysl rozdělit konference (vývojáři, uživatelé, jednotlivé celky)
 - ▶ Předávání pravomocí a úkolů, nalezení lidí, kterým lze věřit a rozdělení úkolů
- ▶ Co může pomoci
 - ▶ Pravidelné posílání informací se shrnutím aktuálního a plánovaného vývoje
 - ▶ Organizace projektových konferencí a srazů
 - ▶ Opět učit se z toho, jak svůj růst zvládají a organizují jiné projekty. Vybírat si to, co funguje.

- ▶ Jednoduché nástroje
 - ▶ diff, patch a tar
 - ▶ Změny (patch-e) posílané přímo v e-mailech základní pravidla, žádné HTML, přímo v těle, pozor na spatné e-mailové programy – tabelátory, lámání řádků atd.
 - ▶ Každý si udržuje svůj vlastní zdrojový strom
 - ▶ Distribuce přes FTP a usenet

- ▶ Základní způsob výměny změn v kódu
 - ▶ existuje množství formátů – v dnešní době je unidiff standardem
 - ▶ ke změně je přidán minimální („dostatečný“) nezměněný kontext
 - ▶ hlavní nástroje: diff, patch, diffstat

```
$diff -u -N -p -r prj-ver.orig prj-ver >prj-ver.diff
```

```
--- prj-ver/source3/rpc_server/srv_svcctl_nt.c
+++ prj-ver.orig/source3/rpc_server/srv_svcctl_nt.c
@@ -466,9 +466,7 @@ WERROR _svcctl_EnumServicesStatusW(pipes_struct *p,
     }

     blob = ndr_push_blob(ndr);
-     if (blob.length >= r->in.offered) {
-         memcpy(r->out.service, blob.data, r->in.offered);
-     }
+     memcpy(r->out.service, blob.data, r->in.offered);
 }
```

```
cd prj-ver-other
patch -p1 <../prj-ver.diff
```

- ▶ Vždy použít diff -up (-r -N), vynechat generované soubory (-x, -X)
- ▶ Vložit statistiku změn – diffstat
- ▶ Přidat popis změny a uvést původního autora
 - ▶ U většiny systémů zprávy verzí se první řádka komentáře (v e-mailu předmět zprávy) objeví jako short log, měla by tedy být výstižná, další popisné.
- ▶ U většiny projektů je k odlišení v e-mailové konferenci zvykem vkládat na začátek předmětu značku [PATCH]
- ▶ Patch vždy jako plain text a přímo v těle zprávy
- ▶ Ještě jednou, nikdy ne jako HTML, přílohu jde někdy možná obhájit
- ▶ Větší změny vždy rozdělit do logických kroků, pak jako patchseries
- ▶ Vždy zkontrolujte dodržování stylu zápisu kódu
linux-2.6.x/scripts/checkpatch.pl
- ▶ Ještě jednou zkontrolujte, že posíláte patch na správnou adresu
- ▶ Přidejte Signed-off-by řádku, je li to u projektu zvykem
- ▶ Obrňte se trpělivostí, čekejte, na připomínky odpovídejte

- ▶ První SCM (source code management) byly RCS a SCCS
RCS 1982, Walter F. Tichy, Purdue University
 - ▶ Pracují pouze s jednotlivými soubory přímo na disku *,v
 - ▶ Pouze jeden uživatel může editovat soubor v daném čase
 - ▶ Žádná možnost slučování nezávislých změn (merge)
 - ▶ Dokumentují historii vývoje
 - ▶ Klíčové údaje jsou kdo, co a kdy

- ▶ Concurrent Versions System – paralelní správa verzí
 - ▶ Založený na základech a formátu RCS
 - ▶ Dovoluje paralelní vývoj (na jednom počítači i distribuovaně)
 - ▶ Obsahuje základní nástroje na řešení slučování (merge) a řešení konfliktů – jsou založené na nástrojích diff a patch
- ▶ Velmi rozšířené ve světě FOSS projektů
 - ▶ Převládající nástroj v letech 1991 až 2005
 - ▶ Stále široce užívaný, ale rok od roku méně
- ▶ Množství nedostatků
 - ▶ V podstatě žádná podpora pro přejmenování souborů a minimum pro práci s adresáři – v podstatě jen recursive
 - ▶ Téměř všechny operace vyžadují komunikaci se serverem
 - ▶ Slabá podpora slučování větví

- ▶ Kde je projekt hostovaný
 - ▶ CVS server je centrální prvek
 - ▶ Vývojář má pouze svůj (aktuální) checkout/sandbox/pískoviště
 - ▶ Většina/všechna metadata (historie atd.) jsou uloženy pouze na centrálním serveru
- ▶ Distribuovaná správa verzí
 - ▶ Každý vývojář má vlastní kopii celé historie projektu
 - ▶ Většina takových systémů nabízí podporu pro snadné zakládání větví a jejich slučování

```
export CVS_RSH=ssh
CVSROOT=":ext:ppisa@ulan.cvs.sourceforge.net:/cvsroot/ulan"
CVSMODULE="ulan"
```

```
git cvsimport -v -d $CVSROOT -C ulan-devel -i -k -a -r ulan-sf $CVSMODULE
```

- ▶ Další pokus implementovat CVS tentokrát již správně
 - ▶ Snaha o znovuvytvoření systému s centrální správou verzí
 - ▶ Řeší mnoho omezení CVS
 - ▶ Revize jsou zaznamenávané přes celý projekt
 - ▶ Často nově nasazované i přechod z CVS od roku 2001 a dále
 - ▶ Stále často užívané
- ▶ Centralizovaný návrh
 - ▶ Kritika chybějícího distribuovaného návrhu
 - ▶ Existuje nadstavba pro distribuované použití (svk), ale není často používaná
 - ▶ Obecně lze v době Git, Mercurial (Hg), Darcs označit za minulost

```
git svn clone https://sdcc.svn.sourceforge.net/svnroot/sdcc/trunk sdcc
```

```
cd sdcc ; git svn rebase ; git gc
```


- ▶ Na počátku
 - ▶ Code Co-Op (pro Windows) 1997
 - ▶ GNU Arch (nazývaný TLA – Tom Lord's Arch) 2001
- ▶ Bitkeeper
 - ▶ Použitý na Linuxové jádro od roku 2002
 - ▶ Problematický licenční model (viz „Bydlení pro jádro“)
 - ▶ Přesto nesmírně přispěl k rychlosti vývoje jádra
- ▶ Novější systémy
 - ▶ Darcs – David Roundy úvaha o novém formátu patchů pro GNU Arch, po několika měsících v roce 2002 nakonec vlastní systém postavený na teorii patchů v C++, od roku 2003 v Haskellu
 - ▶ Mnoho dalších systémů se objevilo v a po roce 2003 bazaar, mercurial, monotone
 - ▶ Git – Linus Torvalds 2005

- ▶ Příkazová řádka převažuje
 - ▶ Nejvíce FOSS uživatelů používá příkazovou řádku
 - ▶ Nástroje jsou zacílené na rychlou práci a umožňují skriptování
 - ▶ Většina SCM systémů nabízí nějaké GUI a nebo integraci do editorů
- ▶ Webová rozhraní
 - ▶ Většina SCM systémů nabízí tyto nadstavby/nástroje/integraci
 - ▶ Především má význam k procházení historie vývoje
 - ▶ cvsweb, svnweb (trac?) a gitweb či cgit
 - ▶ Často jsou upravovány či tvořeny na míru
- ▶ Rozhraní pro propojení s jinými systémy
 - ▶ Nástroje pro propojení se sledováním chyb – trac
 - ▶ Integrace s kompilačními systémy a farmami

- ▶ Integrace SCM s kompilační farmou
 - ▶ Automatické testování pomáhá rychle odhalit chyby
 - ▶ Důležité pro zajištění/zachování přenositelnosti
- ▶ Co to taková kompilační farma je
 - ▶ Rozsáhlá škála strojů (často i virtualizace), různé HW platformy a operační systémy
 - ▶ Automaticky spouští regresní testy po každém commitu (změně)
 - ▶ Chyby při kompilaci nebo testu mohou být poslané na e-mail a jsou k dispozici v logu (přes web)
- ▶ Příklady
 - ▶ Tinderbox
 - ▶ Samba build farm
 - ▶ Build-bot

- ▶ Mnoho kompletních nabídek služeb
 - ▶ sourceforge.net, berlios.de, savannah.gnu.org (kernel.org)
 - ▶ Mnoho/většina FOSS projektů používá právě tyto veřejné služby
 - ▶ Velmi zjednodušuje spuštění a správu projektu
 - ▶ Na jednu stranu méně flexibilní než vlastní řešení, na druhou stranu pod profesionální správou, integrací a vývojem služeb
<http://sourceforge.net/projects/sourceforge>
- ▶ Distribuovaná zpráva verzí DVCS
 - ▶ Práce vyžaduje personální větve a repositáře s jednoduchou správou a třeba i bez nutnosti velkého zázemí
 - ▶ Vzniká mnoho takových serverů pro tyto služby
 - ▶ Git: repo.oz.cz, github.com
 - ▶ Hg: bitbucket.org, freehg.org
 - ▶ bzt: launchpad.net

- ▶ Když si vyberete projekt na práci nebo potřebujete nějaký upravit
 - ▶ Jak naleznu více informací?
 - ▶ Co by bylo dobré vědět?
- ▶ Obvyklé zdroje informací
 - ▶ Manuálové stránky/dokumentace
man abc, info/pinfo abc, cd /usr/share/doc
 - ▶ Popis binárního balíčku, zpráva balíčku v distribuci
<http://packages.debian.org/>
 - ▶ Hledání na Internetu/Webu
 - ▶ <http://freshmeat.net/> dříve LSM (Linux Software Map)

- ▶ Kdo je zapojený do vývoje?
- ▶ Jak je vývoj/projekt organizovaný?
- ▶ Jak je licencovaný?
- ▶ Jak je spravovaný zdrojový kód?
- ▶ Jak jsou připravovaná/publikovaná stabilní vydání (release)?
- ▶ Jaké jsou komunikační nástroje?
- ▶ Jak jsou ukládané informace o chybách a případně i patchích, přáních atd.?
- ▶ Jak je projekt propojený s dalšími projekty?

- ▶ Jakou strukturu projekt má
 - ▶ Existuje projektový tým?
 - ▶ Je projekt součástí většího celku/projektu?
 - ▶ Je spojený/je do něj zapojena nějaká organizace nebo firma?
 - ▶ Existují zde nějaké formální požadavky (na příspěvky atd.)?
 - ▶ Kdo je správcem a kdo rozhoduje?
- ▶ Katedrála nebo bazar ('Cathedral' or 'Bazaar')?
 - ▶ Kazatel/bůh na věži nebo zmatek na tržišti

- ▶ Nejdřív si udělej domácí úkoly!
 - ▶ Neptej se na otázky, které jsou zodpovězené na stránkách projektu
 - ▶ Přečti si “Asking smart questions” FAQ (Eric Steven Raymond)
<http://www.catb.org/~esr/faqs/smart-questions.html>
 - ▶ Hledej odpovědi v e-mailové konferenci a určitou dobu ji čti
<http://search.gmane.org/> často pomůže lépe než Google
- ▶ Pokud dojde na pokládání otázek
 - ▶ Ještě jednou zkontroluj, že se na ní dříve někdo neptal
 - ▶ Dodej dostatečné množství informací aby šlo odpovědět (verzi systému, zdrojových kódů projektu, architekturu, knihovny, popis případné chyby a testcase)
 - ▶ Pokud to není v rámci placené podpory, ptej se slušně a nevyžaduj
 - ▶ Ukaž, že jsi již investoval do nalezení odpovědi, případně nabídni co jsi zjistil

- ▶ Nejdřív hledej a analyzuj
 - ▶ V jaké formě má patch být?
 - ▶ Proti jaké verzi zdrojových kódů?
není to již v aktuální verzi opravené, není použitý základ již tak starý, že vývojáře nezajímá
 - ▶ Jak moc podrobný popis je požadovaný?
 - ▶ Má projekt vývojářskou příručku (developer guide)?
 - ▶ Jak je nakládáno s cizími patchi?
- ▶ Testovat, testovat, testovat!
 - ▶ Buď si jistý, že je úprava funkční
 - ▶ Zkontroluj, že nepokazí něco jiného
 - ▶ Je úprava přenositelná (endianning/32/64-bit/alignment)
- ▶ Klid a trpělivost
 - ▶ Může to trvat dlouho a vyžadovat mnoho další práce, než bude patch integrován