



Co když zmáčknu to červené tlačítko?

aneb Jak neodpálit jadernou bombu.

Lukáš Růžička
(lruzicka@redhat.com)

6. května 2019, FIT ČVUT Praha



Žádná dokumentace, dobrá dokumentace?



Dokumentace není potřebná, když je každému hned jasné, co a jak má s danou věcí dělat.

Tedy danou věc lze použít

- ▶ pouze jedním způsobem
- ▶ s pevně daným výstupem
- ▶ a nemožností selhání

Tak raději přece?



O dokumentaci byste měli uvažovat vždy, když:

- ▶ existuje více možností, jak danou věc použít
- ▶ věc nabízí více než jednu funkci
- ▶ je možné, že výstup se může lišit v závislosti na podmínkách použití
- ▶ věc nemusí vždy fungovat správně

Úkol pro vás



Na základě zmíněných kritérií jmenujte jednu věc, pro kterou nepotřebujete dokumentaci.



Bohové musí být šílení (1980)

K přemýšlení



- ▶ Když lidé nebudou o vašem projektu vědět, nebudou ho používat.
- ▶ Když lidé nebudou umět nainstalovat váš program, nebudou ho používat.
- ▶ Když lidé nepoznají, jak použít váš kód, nepoužijí ho.
- ▶ Když váš program nebude mít očekávané výsledky, lidé ho nebudou používat.

Jaký máme výběr?



Mezi dokumentaci můžeme zahrnout několik různých forem:

- ▶ konceptuální
- ▶ procedurální
- ▶ referenční
- ▶ výukovou

Konceptuální dokumentace



Konceptuální dokumentace se snaží vysvětlit uživateli danou problematiku (**koncept**) tak, aby tito pochopili princip fungování a potřebné souvislosti. Například:

- ▶ Co je počítačová síť a jak se po ní posílají datagramy?
- ▶ Co jsou optické vlastnosti objektivu a jak ovlivňují výslednou fotografii?
- ▶ Jak funguje útok *hrubou silou* a proč je důležité volit si komplikovaná přístupová hesla?
- ▶ Jak působí bakterie zubního plaku a jak zvolit správný kartáček?

Procedurální dokumentace



Procedurální dokumentace poskytuje jasný návod, většinou rozdělený na jednotlivé kroky (**procedura**), ke splnění uživatelského záměru (**user case**). Mimo jiné:

- ▶ Jak nainstalovat aplikaci XY.
- ▶ Jak napsat a odeslat email pro úplné začátečníky.
- ▶ Jak vytvořit nového uživatele a přidělit mu patřičná systémová práva.
- ▶ Jak provést předletovou kontrolu osobního letadla Boeing 747.

Referenční dokumentace



Referenční dokumentace nabízí ucelený přehled (**referenci**) vlastností, voleb, nastavení a způsobů použití, aby si uživatel mohl sám objevit vlastní přístup a sestavit si své vlastní postupy:

- ▶ manuálové stránky v Linuxu
- ▶ přehled typů žárovek pro osvětlení vozidla
- ▶ přehled ingrediencí potřebných pro upečení dortu
- ▶ přehled voleb příkazu `dnf` ve Fedoře.

Výuková dokumentace



Výuková dokumentace, tzv. **tutoriály**, je spojení konceptuální a procedurální stránky dokumentace, takže výsledkem není jenom splněný uživatelský záměr, ale také částečné pochopení problematiky a kontextu.

Je velmi důležité, abychom z konceptuálního hlediska vysvětlili pouze tolik, kolik je nezbytně nutné. Chceme-li pomocí tutoriálů vysvětlit širší problematiku, pak volíme několik na sebe navazujících.

Úkol pro vás



Navrhněte dokumentaci pro nějakou věc denní spotřeby. Kterou formu zvolíte? Proč?

Správný styl dokumentace



V dokumentaci bychom měli zachovávat několik stylistických pravidel:

- ▶ jednoznačné výrazy
- ▶ jednoduché a krátké formulace
- ▶ neutrální výrazy
- ▶ genderově neutrální prvky
- ▶ v případě pochybností je lepší více informací než méně
- ▶ rozkazovací způsob (procedury)

Nesprávný styl dokumentace



V dokumentaci bychom se raději měli vyhnout:

- ▶ estetickým formám jazyka (metafory, přirovnání, nadsázka)
- ▶ ironii a sarkasmu
- ▶ humoru
- ▶ zlehčování problémů
- ▶ utěšování uživatelů

Porušování pravidel



Někdy můžeme uznat za vhodné pravidla porušit a získáme tak jinou formu dokumentace, jež je

- ▶ zajímavá
- ▶ neotřelá
- ▶ vtipná
- ▶ parodická

Je však nutné si uvědomit, kdo budou čtenáři naší dokumentace a jsou-li tito ochotni takový přístup snášet.

Úkol pro vás



Jakou věc by bylo možné dokumentovat s porušením předchozích pravidel? Jakou věc byste nikdy nechtěli takto dokumentovat?

Technické zpracování



Dokumentaci můžeme psát ve spoustě různých formátů. Záleží, co přesně od dokumentace očekáváme:

- ▶ obtížnosti tvorby dokumentace (WYSIWYG, markdown, markup, ...)
- ▶ metod zobrazení (web, čtečka, tištěné médium, audiovizuální metody)
- ▶ možností spolupráce (žádná, cvs, git, Google)
- ▶ možností publikace (ručně, automaticky (CI))
- ▶ dostupnosti metadat textu (sémantický markup)

Docs As Code



S dokumentací zacházíme jako s kódem a používáme:

1. popisovací jazyk (Markdown, AsciiDoc, \LaTeX , Docbook)
2. sdílený repozitář (Gitlab, Github)
3. správu verzí (Git, SVN)
4. kontinuální integraci (CI)
5. automatické publikování (web)

Formáty



Pokud se v souvislosti s dokumentací mluví o formátech, pak máme na mysli v podstatě dva typy a to:

- ▶ **zdrojové** formáty, tedy ty, ze kterých se dokumentace překládá
- ▶ **cílové** formáty, tedy ty, do kterých se překládá

Některé souborové formáty mohou být jak zdrojové, tak cílové (HTML)

Stručný přehled zdrojových formátů



- ▶ čistý text
- ▶ markdown (Markdown, AsciiDoc)
- ▶ markup (HTML, XML, DocBook, Mallard, reST, \LaTeX)
- ▶ nativní formáty WYSIWYG aplikací (doc, docx, fm, odt, indd)

Stručný přehled cílových formátů



- ▶ HTML (web)
- ▶ epub, mobi a další (ebooky)
- ▶ pdf (tiskárna)

Nejčastěji používané formáty



- ▶ Markdown (Github)
- ▶ AsciiDoc (Red Hat)
- ▶ reST (Python)
- ▶ Mallard (Gnome)
- ▶ T_EX, L^AT_EX
- ▶ DocBook (SuSE, IBM)

Markdown



Výhody:

- ▶ jednoduchý
- ▶ čitelný

Nevýhody:

- ▶ omezené typografické možnosti
- ▶ minimální sémantika
- ▶ nejasné hranice mezi prvky
- ▶ významy některých značek kolidují s reálnými znaky (*)

Markdown je vhodný především pro malé projekty, například **github.io**.

AsciiDoc



Výhody:

- ▶ jednoduchý a čitelný
- ▶ částečně programovatelný
- ▶ lze psát sémantický kód

Nevýhody:

- ▶ nejasné hranice mezi prvky
- ▶ vlastní úpravy porušují standard (kompatibilita)
- ▶ sémantika zvyšuje složitost a omezuje čtivost
- ▶ kolidující značky

AsciiDoc je vhodný pro malé i větší projekty, pokud se spokojíme s omezenými možnostmi.

reST a Sphinx



Výhody:

- ▶ jednoduchý a čitelný
- ▶ programovatelný
- ▶ sémantický
- ▶ jasné hranice mezi prvky

Nevýhody:

- ▶ nutnost přesně dodržovat strukturu kódu

reST, a jeho rozšíření Sphinx, jsou velmi vhodné pro veškeré typy projektů. Jsou oblíbené především mezi programátory Pythonu, protože struktura formátu připomíná Python.

Mallard



Výhody:

- ▶ sémantický
- ▶ jasné hranice mezi prvky
- ▶ modulární

Nevýhody:

- ▶ méně přehledný a čtivý
- ▶ poměrně neznámý
- ▶ modulární

Mallard byl původně určen pro psaní **Gnome** help. Je navržen pro *topic based authoring* (tématicky zaměřené psaní), takže zdrojový kód se spíše skládá z fragmentovaných prvků než velkých celků.

T_EX, L^AT_EX

Výhody:

- ▶ sémantický
- ▶ jasné hranice mezi prvky
- ▶ poměrně přehledný a čtivý
- ▶ dlouhodobě vyvíjený

Nevýhody:

- ▶ nehezky výstup do HTML
- ▶ mnoho odnoží s různými výstupy a funkcemi

Sázecí systém T_EX a sada maker zjednodušujících práci s ním, L^AT_EX, byl vyvinut pro sazbu matematických publikací. Nejlepší výsledky poskytuje pro vytváření tištěné dokumentace.



Docbook



Výhody:

- ▶ sémantický
- ▶ jasné hranice mezi prvky
- ▶ dlouhodobě vyvíjený, tedy robustní
- ▶ systém validace nedovolí chyby

Nevýhody:

- ▶ nepřehledný
- ▶ mnoho různých prvků a vztahů mezi nimi
- ▶ vyžaduje čas na naučení
- ▶ neodpouští chyby

Docbook je pravděpodobně nejkomplexnější volbou pro psaní dokumentace.

Výběr vhodného formátu



Před výběrem formátu pořádně zamyslet, co od něj přesně chceme, neboť špatný výběr formátu může celou tvorbu dokumentace zkomplikovat.

Ptejme se na:

- ▶ formu spolupráce
- ▶ obtížnost psaní
- ▶ automatizované testování a publikování
- ▶ formy výstupu
- ▶ přenositelnost do jiných formátů
- ▶ budoucnost projektu

Obecně platí, že **čím složitější** formát, tím **více možností použití**.

Obvyklý postup při vytváření dokumentace



Dokumentace obvykle vzniká následujícím postupem:

1. zjišťování potřeb uživatelů
2. plánování a alokace zdrojů
3. stanovení formy spolupráce a formátu
4. tvorba zdrojových dokumentů
5. překlad do cílových dokumentů
6. publikování
7. vyhledání a opravení chyb
8. znovu přeložení a publikování (nová subverze)

Dokumentace v Red Hatu obsahově



Red Hat v současné době udržuje mnoho dokumentačních projektů. Organizační struktura dokumentačního oddělení má několik pozic:

- ▶ obsahový stratég (content strategist)
- ▶ technický manažer (pillar lead)
- ▶ manager pro lidské zdroje (people manager)
- ▶ programový manažer (document program manager)
- ▶ dokumentátor (technical writer)

Každý z nich je zodpovědný za jinou oblast vývoje dokumentace.

Dokumentace v Red Hatu technicky



V současné době se pro většinu dokumentačních projektů v Red Hatu používá

- ▶ modulární přístup
- ▶ AsciiDoc

.

Pracovní postup

Při vytváření dokumentace psané v AsciiDocu se postupuje takto:

- ▶ klonování Git repozitáře
- ▶ ruční úprava dokumentů v textových editorech
- ▶ sledování změn v paralelní Git větvi (branch)
- ▶ validace a překlad na lokálním stroji (asciidoctor, ccutil)
- ▶ **peer review** → kontrola kvality
- ▶ **merge** do hlavní větve
- ▶ překlad hlavní větve (ccutil → publican) a vystavení na portále



Probíhající diskuse o dokumentaci

V dokumentačním týmu v minulosti proběhlo několik diskusí a iniciativ o tom, jak dělat věci lépe

- ▶ spolupráce upstream \longleftrightarrow downstream (upstream first)
- ▶ konzistentní terminologie (IBM Style Guide)
- ▶ Docbook versus AsciiDoc (AsciiDoc)
- ▶ správný postup u peer review (definice postupu)
- ▶ jednotný způsob zápisu značek u AsciiDocu (doporučený způsob)
- ▶ sémantický zápis značek v AsciiDocu (zatím se neujalo)
- ▶ modulární dokumentace (objevují se problémy)



Dokumentace pro Fedoru



Fedora je komunitní distribuce Linuxu, na jejímž vývoji se může podílet kdokoliv. Tak i na její dokumentaci.

- ▶ repozitáře na pagure.io (Git)
- ▶ využívá projekt Antora (antora.org)
- ▶ vydána na **docs.fedoraproject.org**

Přispět můžete po přečtení

<https://docs.fedoraproject.org/en-US/fedora-docs/contributing/>

Otázky



Kdo se moc ptá, moc se doví.

Takže?



Děkuji za pozornost.