Posudek bakalářské práce

Matematicko-fyzikální fakulta Univerzity Karlovy

Autor práce	Arkadiusz Martin Antoniewicz
Název práce	Podpora tokenizace pro Diff a Patch
Rok odevzdání	2021
Studijní program	Informatika
Studijní obor	Programování a softwarové systémy

Autor posudku	Vojtěch Horký	Oponent
Pracoviště	Katedra distribuovaných a spolehlivých systémů	

K celé práci

	lepší	OK	horší	nevyhovuje
Obtížnost zadání		X		
Splnění zadání		X	X	
Rozsah práce *... textová i implementační část, zohlednění náročnosti*		X	X	

Cílem práce bylo naimplementovat funkce programů `diff` a `patch`, které nejsou omezeny na konec řádků jako oddělovače. Tj. napsat programy pro porovnávání a záplatování souborů, kde uživatel může zvolit porovnávání například po jednotlivých slovech.[1]

Text práce popisuje jednotlivé algoritmy pro původní programy `diff` a `patch`, poté navazuje jejich rozšířením o uživatelem definovaná pravidla pro rozdělení textu na tokeny; závěr práce pak patří návrhu a implementaci vč. stručného zhodnocení výkonnosti.

Téma práce je zajímavé i velmi užitečné, popisované řešení – především uživatelsky definované tokeny a speciální ošetření *whitespace* znaků – je rozumné a dodaná implementace funguje. Nedostatky práce jsou v chybějících citacích a spíše minimalistické implementaci.

Ve srovnání s předchozí verzí práce autor opravil nejzávažnější chyby v implementaci a lehce vylepšil textovou část práce. I přes tyto změny považuji práci za hraniční: zlepšení známky reflektuje spíše změnu než samotný stav celé práce.

Textová část práce

	lepší	OK	horší	nevyhovuje
Formální úprava *... jazyková úroveň, typografická úroveň, citace*			X	
Struktura textu *... kontext, cíle, analýza, návrh, vyhodnocení, úroveň detailu*		X		
Analýza		X		
Vývojová dokumentace		X	X	
Uživatelská dokumentace		X	X	

Textová část práce je poměrně rozumně strukturována a text se dobře čte.

První kapitola popisuje jednotlivé algoritmy pro hledání podobností v textu a funkci „záplatových“ programů (včetně 3-cestného *merge*). Vzhledem k tomu, že většina definic není v textu dále používání/referencována, mohla by být tato část kratší a důraz přenesen na popis rozšíření těchto algoritmů. Naopak by bylo záhodno více do detailů popsat vytváření vlastních definic a rozhodnutí ohledně zvoleného výstupního formátu.

Vývojová dokumentace je minimální. Část textu o sdílené knihovně je spíše zavádějící (vizte, prosím, komentář k implementaci). Popis výstupního formátu je spíše minimalistický.

(pokračování na další straně)

[1] Posudek je částečně kopií minulé verze, protože téma práce zůstalo stejné.

Uživatelská dokumentace je omezená na popis přepínačů, citelně chybí příklady uživatelsky definovaných gramatik (především „uživatelské“ vysvětlení jejich principu). Gramatika po slovech by navíc mohla být výchozí, pokud uživatel nezadá ani `-a` či `-f`. Další definice jsou sice součástí testové sady, ale bez dalších komentářů je poměrně složité je pochopit.

Jako závažnou chybu je nutné zmínit absenci citací při doslovné kopii několika úryvků (např. z Wikipedie nebo manuálové stránky).

Mezi drobnosti pak patří zdrojový kód vložený jako obrázek (Fig 3.2) nebo různé překlepy.

Implementační část práce

	lepší	OK	horší	nevyhovuje
Kvalita návrhu *... architektura, struktury a algoritmy, použité technologie*			X	
Kvalita zpracování *... jmenné konvence, formátování, komentáře, testování*			X	
Stabilita implementace		X		

Programátorsky jde o relativně o malé dílo (cca 2 000 řádků kódu), jednotlivé programy jsou jednoúčelové, takže nevyžadují žádnou komplikovanou architekturu. Autor odevzdal dílo výhradně jako přílohu do SISu, odkaz na on-line repozitář na GitLabu byl uveden až na výslovný dotaz oponenta.

Odevzdaná práce obsahuje i několik automatizovaných testů, jediná výtka je, že jsou poměrně křehké kvůli tomu, že porovnávají výstup včetně data poslední změny souboru.

Autor v textu píše o sdílené knihovně, nicméně jde pouze o několik zdrojových kódů, které jsou kompilovány do více programů. Z tohoto úhlu pohledu má sice program přibalený `makefile`, ale jeho pravidla žádnou knihovnu nevytváří a navíc ani neumožňuje inkrementální sestavení (chybí jakékoliv závislosti). Takže jsou sdílené soubory rekompilovány při každém sestavení a pro každý program zvlášť. (Plus další drobnosti, které ukazují určité nepochopení zvyklostí při práci s `make`m jako nepoužití `$(LDFLAGS)` nebo (ne)využívání `.PHONY` cílů).

Zdrojové kódy jsou v podstatě bez jakýchkoliv *zajímavých* komentářů, hloubka zanoření řídících struktur je na několik místech za hranicí rozumné čitelnosti. Bezdůvodné použití globální proměnné čitelnosti také nepřispívá.

Implementace často postrádá rozumné rozdělení podle úrovně abstrakce – např. existuje funkce `parse_one_line_automata` (která sice řádek ve skutečnosti neparsuje), ale načtení ze souboru je řešeno přímo v `main`u.

Rozhraní mezi C++ a C (resp. `char *` vs `std::string`) není jednoznačné a nerespektuje logické bloky programu (např. volání `mmap` by spíše mělo vracet `string_view` než `char *`).

Z hlediska použití by bylo vhodné, aby program více kopíroval funkce standardních nástrojů, tj. např. `patch` čte záplatu ze standardního vstupu. Přepínač `--debug` by neměl ovlivnit funkci programu kromě extra výpisů o průběhu činnosti.

Obecně jsou chybové hlášky minimalistické, vestavěná nápověda je prakticky nepoužitelná.

Vyhodnocení výkonnosti přes virtuální stroj je poněkud nešťastné, protože se de-facto měří i výkon virtualizovaného disku apod. než jen výkon vlastní aplikace.

Usuzovat na linearitu ze dvou bodů je poněkud ošemetné (byť autor posudku může potvrdit, že na velikost vstupu od cca 6 MB do 30 MB se jak tokenizace tak vytváření záplaty chová lineárně).

Celkové hodnocení Dobře

Práci navrhuji na zvláštní ocenění Ne

V Praze 25. ledna Podpis