Posudek bakalářské práce

předložené na Matematicko-fyzikální fakultě
Univerzity Karlovy

Autor práce:	Peter Gebauer
Název práce:	Log analyser
Rok odevzdání:	2021
Studijní program a obor:	Informatika, Obecná informatika
Autor posudku:	RNDr. Radek Hušek, oponent
Pracoviště:	Informatický ústav Univerzity Karlovy

K celé práci	lepší	OK	horší	nevyh.
Obtížnost zadání	X			
Splnění zadání	X			
Rozsah práce		X		

Práce je zaměřena na implementaci nástroje pro zpracování logů, který usnadní jejich analýzu se zaměřením na rychlost zpracování. Jde tedy o potenciálně velmi užitečný program, ačkoliv zatím některé vlastnosti chybí – např. nativní export dat ve strukturovaném formátu jako `json`. Jelikož ale program umožňuje definování akcí nad záznamy v jazyce Python, lze tato omezení typicky obejít, i když za cenu mírného snížení výkonu.

Textová část práce odpovídá známce výborně, implementace by si ale zasloužila mírně dočistit a je na hranici mezi výborně a velmi dobře. Celkově si tedy práce zaslouží hodnocení výborně.

Textová část práce	lepší	OK	horší	nevyh.
Formální úprava	X			
Struktura textu	X			
Analýza		X		

Text práce začíná podrobným představením dvou RFC popisujícím logování, následuje popis cílů implementace a zvoleného přístupu. Střed práce tvoří uživatelská a programátorská dokumentace a práce je ukončena benchmarky, porovnáním s ostatními obdobnými nástroji a poznámkami k implementaci. Je psána čtivou angličtinou bez velkého množství chyb.

Na začátku práce bych u popisu RFC ocenil, kdyby alespoň výčtem byly zmíněny důležité programy, které se podle nich skutečně řídí, a naopak které důležité programy řeší logování jinak. Taktéž bych uvítal, kdyby stručný popis již existujících řešení bych již na začátku práce a ne až v poslední kapitole. Jinak je text práce v pořádku.

Implementační část práce	lepší	OK	horší	nevyh.
Kvalita návrhu		X		
Kvalita zpracování		X		
Stabilita implementace	X			
Dokumentace	X			

Implementace je zpracována v jazyce C++ a částečně Python (pro uživatelské akce). Její pozitiva bezpochyby jsou kvalitní dokumentace, stabilita programu a použití profileru k odhalení příliš pomalých částí kódu. Na druhou stranu za kontroverzní lze považovat zvolení nástroje Grok pro implementaci regulárních výrazů – z něj je využívána jen relativně malá část a i v té je z důvodu výkonu nutné obcházet standardní rozhraní a více méně záviset na implementačních detailech.

Vzhledem k tomu, že Grok není příliš aktivně vyvíjen, by bylo lepším řešením přímo vzít odpovídající část zdrojového kódu Groku a upravit ho dle svých potřeb. Toto by mělo být možné, jelikož Grok požívá BSD licenci. Tím by se jednak odstranila nutnost vymýšlet rovnák na ohejbák, ale zároveň by se odstranila závislost na Groku, který chybí v repozitářích mnoha linuxových distribucí.

Další poznámky k implementaci jen stručně:

- Kód obsahuje typ `parsing::Result`, který ale není používán.
- Signal handler dělá netriviální činnost a pravděpodobně může dojít k race condition.
- Není jasné proč kód používá vlastní implementaci string view místo standardní.
- Kód modifikuje načtený string vkládáním znaku `` `\0` ``, aby mohl používat funkce očekávající null-terminated string, i když existují alternativy, které berou délku stringu jako argument.
- Proč je úprava funkce pro výpis v Pythonu implementována jako funkce `print` v modulu `b`, a ne vlastní implementací `sys.stdout`, což by umožnilo používat `print` vestavěný?
- Kvůli nevhodně navrženému rozhraní je na mnoha místech nutné přetypování z `const char*` na `char*`.
- Proč `Context` obsahuje samostatné struktury pro final a intermediate atributy?

Celkové hodnocení: výborně

Práci navrhuji na zvláštní ocenění: ne

V Praze dne 21. června 2021

Radek Hušek