Posudek bakalářské práce

předložené na Matematicko-fyzikální fakultě
Univerzity Karlovy v Praze

❑ posudek vedoucího ☒ posudek oponenta

Autor/ka: Petr Sušil

Název práce: Anotátor programů v jazyce C

Studijní program a obor: Informatika, obecná informatika

Rok odevzdání: 2006

Jméno a tituly vedoucího/oponenta: Mgr. Mikuláš Patočka

Pracoviště: Katedra softwarového inženýrství

	excelentní	**odpovídající**	**slabší**	**nevyhovující**
Náročnost zadaného tématu		X		
Míra splnění zadání		X		
Struktura textové části práce		X		
Jazyková a typografická úroveň		X		
Analýza				X
Vývojová dokumentace		X		
Uživatelská dokumentace		X		
Kvalita zpracování softwarové části			X	
Stabilita aplikace		X		

Nejvýznamnější klady:

Autor prokázal schopnost programovat v jazyce C a vyvinout středně velký program. Program cspot je použitelný pro vyhledávání v menších projektech, ve kterých nedochází k podmíněné kompilaci jednotlivých částí.

Nejzávažnější nedostatky:

Program cspot používá preprocesor a parser jazyka C, čímž se odlišuje od jiných podobných programů. Považuji tohle za zásadní nevýhodu programu. Větší projekty většinou mají části, které jsou kompilovány pouze na určitém operačním systému, na určitém procesoru, pouze pokud je přítomna určitá knihovna apod. Vezměme si například zdrojový kód obsahující následující řádky:

```
#if defined(unix)
        udelej_neco();
#elif defined(WIN32)
        udelej_neco_jineho();
#elif defined(OS2)
        udelej_neco();
        a_jeste_neco();
#endif
```

Pokud na takovémto zdrojovém kódu autorův program necháme vyhledat volání funkce "`udelej_neco_jineho()`", a program momentálně neběží na Windows, tak nám výsledek nenajde. Pokud program pustíme na Windows, tak zase nebude prohledávat větve příslušející Unixu nebo OS/2. Autorův program tak dává horší výsledky než `grep` a než všechny konkurenční programy pracující bez znalosti syntaxe jazyka pouze s tokeny.

Nejčastěji vznikne potřeba pro vyhledání odkazů na nějakou funkci, pokud programátor změní její sémantiku a chce se ujistit, že všechna její volání jsou v souladu s novou sémantikou – pokud použije `grep`, dostane všechna volání funkce a některé řádky, které nejsou voláními, navíc – pokud pužije autorův program, nedostane žádný řádek navíc, ale některá volání budou chybět (ta, která jsou ve větvi, která se momentálně nekompiluje). Chybějící výpisy považuji za mnohem závažnější vadu než výpisy navíc – a proto nedoporučuji tento program na žádném větším projektu používat.

Tento nedostatek bohužel nelze opravit – autor si to při analýze problému a návrhu řešení neuvědomil.

V určitém případě může být pouštění preprocesoru výhoda – pokud v cizím programu hledáme definici funkce, přičemž název funkce v její definici je generován spojováním tokenů pomocí preprocesoru (takový případ nelze nalézt pomocí programu `grep`) – ovšem sám jsem se s podobným problémem setkal jen pákrát.

Další poznámky:

Další nepříjemné chyby (opravitelné):

* nevypisuje chyby při špatné syntaxi příkazů, při neexistujícím souboru (příkazy fl, ip, def) nebo při špatném formátu souboru (příkaz fl). Tváří se, jako by příkaz provedl, avšak neprovede nic.
* při syntaktické chybě ve zdrojovém kódu přestane tento soubor prohledávat, avšak nevypíše žádnou chybu.
* příkaz ls spadne na segmentation fault, pokud je databáze prázdná.
* nenalezl jsem způsob, jak zadávat příkazy rovnou z argumentů (nápověda říká „cspot -příkaz" – pokud zadám „cspot -ls", tak dostanu chybu, že databáze není otevřená. Pokud zadám „cspot -db-ro spot.db" nebo „cspot '-db-ro spot.db' ", dostanu chybu, že argument db-ro je neznámý, ačkoli je to příkaz pro otevření databáze.

	výborně	**velmi dobře**	**dobře**	**neprospěl/a**
Návrh známky		X		

Datum:

Podpis: