

CHA: C Header Analysis

Zodpovědný cvičící: Petr Zemek (izemek@fit.vutbr.cz)

1 Detailní zadání úlohy

Vytvořte skript pro analýzu hlavičkových souborů jazyka C (přípona `.h`) podle standardu ISO C99, který vytvoří databázi nalezených funkcí v těchto souborech.

Tento skript bude pracovat s těmito parametry:

- `--help` Viz společné zadání všech úloh.
- `--input=fileordir` Zadaný vstupní soubor nebo adresář se zdrojovým kódem v jazyce C. Předpokládejte, že soubory budou v kódování UTF-8. Je-li zadána cesta k adresáři, tak jsou postupně analyzovány všechny soubory s příponou `.h` v tomto adresáři a jeho podadresářích. Pokud je zadána přímo cesta k souboru (nikoliv k adresáři), tak příponu souboru nekontrolujte. Pokud nebude tento parametr zadán, tak se analyzují všechny hlavičkové soubory (opět pouze s příponou `.h`) z aktuálního adresáře a všech jeho podadresářů.
- `--output=filename` Zadaný výstupní soubor ve formátu XML v kódování UTF-8 (přesný formát viz níže). Pokud tento parametr není zadán, tak dojde k vypsání výsledku na standardní výstup.
- `--pretty-xml=k` Skript zformátuje výsledný XML dokument tak, že (1) každé nové zanoření bude odsazeno o k mezer oproti předchozímu a (2) XML hlavička bude od kořenového elementu oddělena znakem nového řádku. Pokud k není zadáno, tak se použije hodnota 4. Pokud tento parametr nebyl zadán, tak se neodsazuje (ani XML hlavička od kořenového elementu).
- `--no-inline` Skript přeskočí funkce deklarované se specifikátorem `inline`.
- `--max-par=n` Skript bude brát v úvahu pouze funkce, které mají n či méně parametrů (n musí být vždy zadáno). U funkcí, které mají proměnný počet parametrů, počítejte pouze s fixními parametry.
- `--no-duplicates` Pokud se v souboru vyskytne více funkcí se stejným jménem (např. deklarace funkce a později její definice), tak se do výsledného XML souboru uloží pouze první z nich (uvažujte průchod souborem shora dolů). Pokud tento parametr není zadán, tak se do výsledného XML souboru uloží všechny výskyty funkce se stejným jménem.
- `--remove-whitespace` Při použití tohoto parametru skript v obsahu atributů `rettype` a `type` (viz níže) nahradí všechny výskyty jiných bílých znaků, než je mezera (tabelátor, nový řádek atd.), mezerou a odstraní z nich všechny přebytké mezery. Např. pro funkci „`int * func(const char arg)`“ bude hodnota parametru `rettype` „`int*`“ a hodnota parametru `type` pro parametr bude „`const char`“.

Všechny parametry jsou nepovinné a na jejich pořadí nezáleží. Soubory či adresáře mohou být zadány jak relativní tak absolutní cestou. Skript bude prohledávat všechny hlavičkové soubory od místa uložení hlouběji (myšleno k podadresářům). POZOR! Testovací adresářová struktura s hlavičkovými soubory, ale případně i jinými soubory, které je třeba ignorovat, bude ke skriptu nahrávána námi automaticky. Pokud dojde k chybě, tak skript vypíše chybové hlášení na standardní chybový výstup a skončí s předepsanou návratovou hodnotou.

Výstupem bude XML dokument v následujícím formátu:

```
<?xml version="1.0" encoding="utf-8"?>
<functions dir="">
  <function file="" name="" varargs="" rettype="">
    <param number="" type="" />
    ...
  </function>
  ...
</functions>
```

Element `functions` je kořenový. Na pořadí XML elementů v rámci stejné úrovně nezáleží. Všechny atributy XML elementů jsou povinné. Jejich význam a obsah je následující:

- **dir** pokud byl zadán parametr `--input` a jedná se o adresář, pak se zde použije hodnota tohoto parametru (ta může být buď relativní, nebo absolutní). Pokud tento parametr nebyl zadán, použije se „./“. Pokud byl zadán, ale jedná se o soubor, pak bude hodnota tohoto atributu prázdná.
- **file** soubor, ve kterém byla funkce nalezena (včetně podadresářů). Pokud je hodnota atributu `dir` kořenového elementu neprázdná, bude cesta k souboru relativní k této hodnotě. Příklad: při spuštění s parametrem `--input=/usr/include/` a nalezením funkce v souboru `alisp.h` v podadresáři `alsa` bude hodnota tohoto atributu „`alsa/alisp.h`“. Pokud je hodnota atributu `dir` kořenového elementu prázdná (to znamená, že `--input` obsahuje cestu k souboru), pak bude obsahem tohoto atributu hodnota parametru `--input`.
- **name** název funkce.
- **varargs** pokud se jedná o funkci s proměnným počtem parametrů, tak bude hodnota atributu „yes“, v opačném případě „no“.
- **rettype** návratový typ funkce. Pro zjednodušení zde uvažujte i případné specifikátory, které nejsou součástí návratového typu. Příklad: pro funkci „`extern int *f(volatile const int *arg);`“ bude hodnota tohoto atributu „`extern int *`“.
- **number** pořadové číslo parametru (čísluje se od 1).
- **type** typ parametru. Příklad: pro funkci uvedenou výše bude hodnota tohoto atributu „`volatile const int *`“.

Závěrečné poznámky:

- Element `function` se vytvoří pouze tehdy, pokud prohledávaný soubor nějakou deklaraci funkce obsahuje.
- Pokud není zadán parametr `--remove-whitespace`, tak u typu funkce a typu parametrů zachovejte množství bílých znaků mezi jednotlivými symboly (kromě začátku a konce, odkud přebytečné bílé znaky odstraňte vždy, nezávisle na tom, zda byl použit parametr `--remove-whitespace`). Příklad: pro funkci

```
„ int* f ( int    * arg ) ;“
```

bude její typ „`int*`“ a typ parametru „`int *`“.

- *Zanoření* (viz parametr `--pretty-xml`) se týká vztahů mezi XML elementy a podadresáři na něj nemají žádný vliv. *Novým zanořením* je zde myšlena situace, kdy rodičovský element má syna, čili když element `functions` má podelement `function` a ten má podelement `param`.
- U funkce „`void f();`“ uvažujte, že se jedná o funkci bez parametrů.
- Ignorujte deklarace funkcí v makrech, komentářích či řetězcích.
- Nezapomeňte, že každá definice je zároveň deklarací.
- Můžete předpokládat, že všechny soubory budou vyhovovat standardu ISO C99. Taktéž v rámci základního zadání (tj. bez rozšíření) můžete předpokládat, že každý parametr bude mít kromě typu i název.

Reference:

- Standard ISO/IEC 9899:TCC Committee Draft — September 7, 2007. Dostupné na <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1256.pdf> [citováno 1.2.2011]

2 Bonusová rozšíření

FUN: Zpracování složitějších funkcí a jejich parametrů. Příklad: pro funkci

```
int * (* func(int * arg[]))();
```

bude její návratový typ „`int * (*)()`“ a typ parametru „`int * []`“. Implementací tohoto rozšíření lze získat až 1 bod.

PAR: Zpracování deklarací funkcí, u kterých chybí název některých (nula až všech) parametrů. Příklad:

```
int func(int *, const float, char[])
```

Implementací tohoto rozšíření lze získat až 1 bod.

3 Poznámky k hodnocení

Výstupy budou zpracovávány nástrojem na porovnání XML souborů s referenčními XML soubory (nástroj se umí vypořádat s prohozeným pořadím podelementů v rámci jednoho elementu apod.).