Dokumentace úlohy CHA: C Header Analysis v PHP 5 do IPP 2015/2016
Jméno a příjmení: František Šumšal
Login: xsumsa01

1 Úvod

Cílem tohoto projektu byla implementace skriptu v jazyce PHP 5, který analyzuje a zpracovává hlavičkové soubory jazyka C. Výstupem skriptu je XML dokument, ve kterém se vyskytují všechny funkce a jejich argumenty vyhovující standardu C99 (výjimkou jsou pokročilejší formáty funkcí) a parametrům specifikovanými na příkazové řádce.

2 Implementace

Vlastní implementace je rozdělena do několika částí: parsování parametrů příkazové řádky, zpracování vstupu, upravení formátu vstupního souboru, zpracování vstupního souboru a výsledné generování výstupního XML dokumentu.

2.1 Parsování parametrů

Vlastní parsování probíhá ve funkci check_opts. Zde jsou nadefinované očekávané parametry a ty jsou dále předány funkci getopt. Bohužel v PHP se funkce getopt nechová tak, jak v ostatních jazycích, takže vrací jen pole parametrů, které vyhovují zadaným podmínkám. Jelikož zadání vyžaduje detekci nevalidních parametrů, je nutný „workaround“ ve formě funkce filter_opts, který z globální proměnné $argv odstraní již zpracované parametry a název souboru skriptu. Pokud se po zavolání této funkce vyskytují nějaké položky v $argv, tak se téměř jistě jedná o neplatné parametry. Všechny validní parametry jsou poté uloženy do globální proměnné $options.

2.2 Zpracování vstupu

Následně je třeba zjistit, zda-li pracujeme s adresářem či se souborem. Tuto činnost obstarává funkce process_input. V případě souboru se rovnou volá funkce parse_file, kdežto v případě adresáře je nutné jej nejprve rekurzivně zpracovat a vybrat jen soubory s příponou .h. K tomu slouží funkce scan_dir, která pro každý nalezený hlavičkový soubor zavolá výše zmíněnou funkci parse_file.

2.3 Upravení formátu vstupního souboru

Jelikož zadání vyžaduje, aby byly ignorovány všechny deklarace a definice funkcí v řetězcových literálech, makrech a blokových/řádkových komentářích, je třeba obsah souboru nejdřív patřičně upravit. Jednou z možností by byly regulární výrazy, ale ty by ve výsledku byly velmi komplexní a je u nich velké riziko chyb. Jednodušším a spolehlivějším řešením je použití konečného automatu. Ten se vyskytuje ve funkci process_content a odstraní ze souboru všechny výše zmíněné konstrukce, takže se obsah dá dále jednoduše zpracovat.

2.4 Zpracování vstupního souboru

Každý vstupní soubor je po patřičné úpravě formátu zpracován funkcí parse_file. Jádrem celého zpracování jsou dva regulární výrazy:

```
$fnc_pattern = "/(?'dtype'[[:alnum:]_][[:alnum:]_\s\*]+?(?!\s*\())"
             . "(?'name'[[:alnum:]_]+)\s*\(\s*(?'args'[^)]*)\s*\);?/";
$arg_pattern = "/(((?'dtype'[[:alnum:]_\s\*]+)\s*?(?'name'(?<=(\s|\*))"
             . "[[:alnum:]_]+))|\s*(?'va'[.]{3})\s*)\,?/";
```

První regulární výraz rozdělí deklaraci funkce na její datový typ, identifikátor a řetězec s argumenty. Tento řetězec poté zpracuje druhý výraz, který každý argument rozdělí na datový typ a identifikátor (v případě funkcí s proměnným počtem parametrů jen nastaví patřičný index výsledného pole). Z těchto informací je poté vytvořeno globální pole $input_data.

2.5 Vygenerování XML

Závěrečným krokem je vygenerování výsledného XML dokumentu, což obstarává funkce generate_XML. Ta využívá třídy XMLWriter, pomocí které vygeneruje pro každou funkci element function, který může obsahovat několik subelementů param popisující argumenty dané funkce. Výsledný dokument se poté ukládá do specifikovaného výstupního souboru nebo je vypsán na standardní výstup.