Dokumentace úlohy CHA: Header Analysis v PHP 5 do IPP 2016/2017
Jméno a příjmení: Martin Bažík
Login: xbazik00

1 Task

The task was to create a script in PHP 5 that would analyze header file written in C programming language and parse individual function declarations into the file of XML format.

2 Parameters

The scanning of script parameters is done by sequential verification and parsing individual parameters. The names of the parameters are saved as keys and its values are saved in values for the given key in an array. Every parameter has a restriction for its values and these limits are checked in this section. In a case of an unexpected combination parameters and its values, the scripts end with an error code and a message on standard error output. Scanning and checking of parameters is taken care of by series of functions.

3 Opening input file

The opening of an input file differs depending on the fact whether the given path is the path to a directory or the path to a file. Script processes this in a function `check_input()`. If the path corresponds to the file. Assuming the path leads to a directory, this directory and subdirectories are searched for files with `".h"` extension. This is done by recursive function `search_dir()` and function `parse_h()`. Occasional errors of input file path are solved in the function `check_input()`. In a case, the script is given a directory, permissions of each C header file are verified by `check_pemr()` function. Incidents concerning permission problems and wrong input path result in corresponding error code and error message. Individual files are then processed one by one in a loop.

4 Reading input file

The reading of the input file is performed by the function `process_input()`. This function takes the file, which was converted to string by `file_get_contents()` function and reduced by `reduce()` function, and processes it. The process uses regular expressions to parse the file into useful arrays. These array are used later to generate final XML file.

Before the parsing, the text is reduced by `reduce()` function using Finite State Machine.

5 Finite State Machine

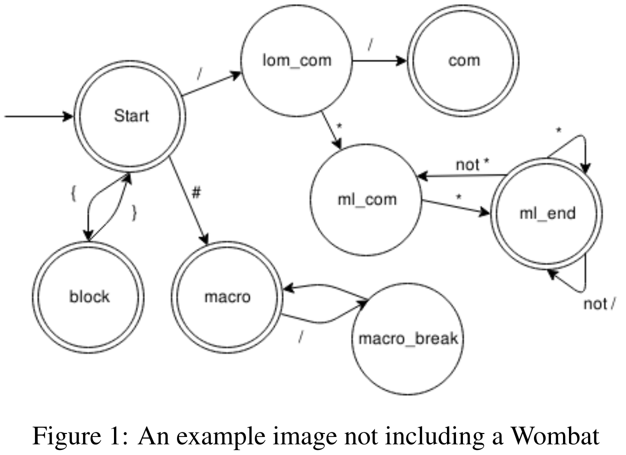

Figure 1: An example image not including a Wombat

The Finite State Machine consists of 5 accepting states:

- Start - Starting state
- com - single line comment
- ml_com - multiline comment
- macro - macro
- block - block of code

In all these state the text is deleted, except block. Each block is replaced with ”;”.

6 Scanning input file

The scanning uses regular expressions to parse individual parts of function declaration. These parts consist of return type, function name and parameters. After the first parsing, the second regular expression is used to parse individual parameters. During this process, potential ”...” for variable number of arguments is found.

7 Output

Output file is either given by ”–output” parameter or is standard output. The given output file is tested for permission to write. The XML file itself is produced by `generate_start_xml()` and `generate_xml_body`. These functions take into consideration all the script parameters, files and directories to generate correct form of XML file.