

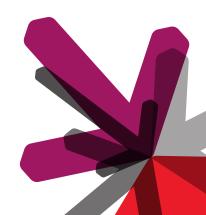
C STYLE CHECKER

Flex and Bison Implementation.

November 30, 2013

Bruce Dearing

Acadia University



OVERVIEW

Overview

- 1. Introduction
- 2. Implementation Description
- 3. Possible Extensions
- 4. Summary and Conclusions
- 5. Questions?



COMP 2103 students are required to format their programs according to a set of style guidelines.

SOLUTION DESCRIPTION

- → Initial Approach
 - → Flex and Bison
- → Second approach
 - → Flex and Bison
 - → GNU Indent
- → Third approach
 - → Flex and Bison
 - → GNU Indent
 - → Vim 'cindent'

Flex and Bison are a set of tools originally designed for constructing compilers. They have proven to be very useful in building programs which handle structured input.

```
%{ /* Declarations and optiions */
int chars = 0:
int words = 0;
int lines = 0;
%}
%% /* Patterns and actions. */
[a-zA-Z]+ { words++; chars += strlen(yytext); }
\n
         { chars++; lines++; }
         { chars++; }
/* C code that is copied to the generated scanner. */
main(int argc, char **argv)
   vvlex();
   printf("%8d%8d%8d\n", lines, words, chars);
```

```
%{
                   /* declare options */
%}
%token ONE TWO EOL /* declare tokens */
%start start
%%
start
    : exp EOL /* grammar rules */
exp
    : ONE
     TWO
%%
                  /* C code that is copied to parser */
main(int argc, char **argv)
   yyparse();
yyerror(char *s)
   fprintf(stderr, "error: %s\n", s);
```



The C style checker is composed of four main components

- → Comments:
- → Indentation;
- → Common errors, and Style;

which perform the following checks:

→ Format, and White space.

The components are tied together with a shell script which is accessed through a web based interface.

COMMENTS

The comment component of the style checker attempts to:

- 1. verify that the program starts with a header comment similar to the following, and
- 2. that functions preceded by a short comment describing the function are correctly formatted.

```
HEADER COMMENT
                                     FUNCTION COMMENT
/*
                                      /*
        A2P1.c
* File:
                                       * Name:
                                                    my_func
* Author: My Name 100123456
                                       * Purpose:
 * Date: 2011/09/12
                                       * Arguments:
 * Version: 1.0
                                       * Output:
                                       * Modifies:
 * Purpose:
                                       * Returns:
                                       * Assumptions:
                                       * Bugs:
                                       * Notes:
                                                      . . .
                                       */
```

The check_comments program is a combined scanner parser.

```
%x COMMENT /* Exclusive start state. */
%%
"/*"
              { BEGIN(COMMENT); }
<COMMENT>"*/" { BEGIN(INITIAL); return(END_COMMENT);}
```

The parser is responsible for determining if the stream of tokens provided by the scanner conforms to the grammar specification.

```
%token IDENTIFIER FILE LBL AUTHOR START COMMENT END COMMENT VERSION DATE
%token NAME ARGUMENTS OUTPUT MODIFIES RETURNS ASSUMPTIONS BUGS NOTES
%token PURPOSE LAST VAL
%start program body
program_body
    : program start program comments
program comments
    : comment start
     program_comments comment_start
program_start
    : START COMMENT header comment END COMMENT {check header():}
comment start
    : START_COMMENT comment_body END_COMMENT
 . . .
```

INDENTATION

The indentation portion of the style checker attempts to verify the file has been indented correctly.

```
vim -e -s $temp_in < indent/vim_commands.scr</pre>
```

indent/vim commands.scr

```
: set shiftwidth=4
: set cinoptions=e0,n0,f0,{0,}0,:2,=2,11,b0,t0,+4,c4,C1,(0,w1
: normal gg=G
: wq
```

The Common errors and style portion of the style checker consists of two components.

- 1. common_errors program which initiates checks for:
 - → Common white space errors. and
 - → Code block bracket location.
- composite_check program which combines the ANSI C grammar specification with a combination of additional grammar productions to produce style error checks.

The format portion of the style checker utilizes the GNU indent program to verify the file has been correctly formatted.



C Preprocessor

Construct the C style checker to first run the files through the C preprocessor.

Continue Productions

Continue constructing grammar productions to produce checks for additional style errors.

Eliminate or extend indent

Extend or modify indent's source code to enhance its flexibility.



AVAILABLE CHECKS

The C style checker provides the following checks.

- Array Trailing Comma
- Type Name
- 3. Default Comes Last
- 4. Empty Block
- Empty Statement
- 6. Fall Through
- 7. File Length
- 8. Indentation
- 9. Left Curly
- 10. Line Length
- 11. Local Variable Name
- 12. Magic Number
- 13. Method Name
- 14 Method Param Pad

- 15. Missing Switch Default
- 16. Modified Control Variable
- 17. One Statement Per Line
- 18. Parameter Name
- 19. Paren Pad
- 20. Right Curly
- 21. Type Name
- 22. Type cast Paren Pad
- 23. Header comment
- 24. White space After
- 25. White space Around
- 26. Multiple variable declarations with initializations

Required Software:

- → VIM Vi IMproved version 7.3.547
- \rightarrow flex 2.5.35
- → bison (GNU Bison) 2.7.12-4996 (any version above 2.5).
- → GNU indent 2.2.11
- \rightarrow gcc 4.7.2

DEMONSTRATION

Demonstration

