Makefile CodeCount™ Counting Standard

University of Southern California

Center for Systems and Software Engineering

December , 2016

Revision Sheet

Date	Version	Revision Description	Author
6/13/2016	1.0	Original Release	Matthew Swartz
7/12/2016	1.1	Added complexity section	Matthew Swartz

Table of Contents

1. Definitions

1.1. **SLOC –** Source Lines of Code is a unit used to measure the size of software program. SLOC counts the program source code based on a certain set of rules. SLOC is a key input for estimating project effort and is also used to calculate productivity and other measurements.

1.2. **Physical SLOC –** One physical SLOC is corresponding to one line starting with the first character and ending by a carriage return or an end-of-file marker of the same line, and which excludes the blank and comment line.

1.3. **Logical SLOC –** Lines of code intended to measure "statements", which normally terminate by a semicolon (C/C++, Java, C#) or a carriage return (VB, Assembly), etc. Logical SLOC are not sensitive to format and style conventions, but they are language-dependent.

1.4. **Data declaration line or data line –** A line that contains declaration of data and used by an assembler or compiler to interpret other elements of the program. There are no explicit data declaration statements in Makefiles.

1.5. **Compiler Directives –** A statement that tells the compiler how to compile a program, but not what to compile.

The following table lists the Ada keywords that denote data declaration lines:

`include`	`-include`	`sinclude`

Table 2 Compiler Directives

1.6. **Blank Line –** A physical line of code, which contains any number of white space characters (spaces, tabs, form feed, carriage return, line feed, or their derivatives).

1.7. **Comment Line –** A comment is defined as a string of zero or more characters that follow language specific comment delimiter. Makefile comment delimiter is "#". A whole comment line may span one line and does not contain any compilable source code. An embedded comment can co-exist with compilable source code on the same physical line. Banners and empty comments are treated as types of comments.

1.8. **Executable Line of code –** A line that contains software instruction executed during runtime and on which a breakpoint can be set in a debugging tool. An instruction can be stated in a simple or compound form.

- An executable line of code may contain the following program control statements:
  - Selection statements (if, ? operator, switch)
  - Iteration statements (for, while, do-while)
  - Empty statements (one or more ";")
  - Jump statements (return, goto, break, continue, exit function)
  - Expression statements (function calls, assignment statements, operations, etc.)

  - Block statements
- An executable line of code may not contain the following statements:
  - Compiler directives
  - Data declaration (data) lines
  - Whole line comments, including empty comments and banners
  - Blank lines

2. Checklist for source statement counts

PHYSICAL SLOC COUNTING RULES			
MEASUREMENT UNIT	ORDER OF PRECEDENCE	PHYSICAL SLOC	COMMENTS
Executable Lines	1	One per line	Defined in 1.8
Non-executable Lines			
Declaration (Data) Lines	2	One per line	Defined in 1.4
Compiler Directives	3	One per line	Defined in 1.5
Comments			Defined in 1.7
On their own lines	4	Not Included	
Embedded	5	Not Included	
Banners	6	Not included	
Empty comments	7	Not included	
Blank lines	8	Not Included	Defined in 1.6

LOGICAL SLOC COUNTING RULES				
NO.	STRUCTURE	ORDER OF PRECEDENCE	LOGICAL SLOC RULES	COMMENTS
R01	Variable assignment	1	Count once	Independent statement
R02	Variable definition	2	Count once each line	Independent statement
R03	Target	3	Count once each line	Independent statement
R04	Clean	4	Count once each line	Independent statement
R05	Secondary Expansion	5	Count once each line	Independent statement
R06	Compiler Directive	6	Count once each directive	Independent statement

3. Examples

EXECUTABLE LINES

SELECTION Statements

ESS1 – variable assignment

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<name> = <value>	OBJECTS = file1.o file2.o file3.o	1
<name> ?= <value>	immediate ?= deferred	1
<name> := <value>	immediate := deferred	1
<name> += <value>	immediate += deferred	1

ESS2 – clean statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
clean: <statements>	clean: rm -f file1.txt file2.txt	1 1

ESS3 – target prerequisite statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
target: <prerequisite recipe>	file1.o: file1.c $(CC) -g -c file1.c edit: main.o kbd.o command.o \ display.o insert.o	1 1 1 0

ITERATION Statements

EIS1 – secondary expansion

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
.secondexpansion: <statements>	.SECONDEXPANSION: AVAR = top onefile: $(AVAR) twofile: $$(AVAR)	1 1 1 1 1 1

COMPILER DIRECTIVES

CDL1 – directive types

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
include <file_name>	include makefile1	1

4. Complexity

Complexity measures the occurrences of different keywords in code baseline. Below table identifies the categories and their respective keywords that are counted as part of the complexity metrics.

Math Functions	Trig	Log	Calculations	Conditionals	Logic	Pre-processor	Assignment
		LOG	+	IFNDEF		-INCLUDE	?=
			-	IFNEQ		SINCLUDE	:=
			*	IFDEF		INCLUDE	=
			/	WHILE			
				IFEQ			
				ELSE			
				FOR			