Dos-Batch CodeCount™

Counting Standard

University of Southern California

Center for Systems and Software Engineering

May, 2014

Revision Sheet

Date	Version	Revision Description	Author
10/30/2007	1.0	Original Release	Jie Zheng

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 (Dos-batch, 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. None exists in dos batch file.

1.5. **Compiler Directives –** A statement that tells the compiler how to compile a program, but not what to compile. None exists in dos batch file.

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.

Batch adopts five ways of commenting. We count two of them. One is using REM command, this is the only documented way to insert a comment. REM this line should be counted as one SLOC line. The other way is using :: label. This is a non-documented comment line. A whole comment line may span one line and does not contain any compliable source code. An embedded comment can co-exist with compliable source code on the same physical line.

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, CHOICE)
  - Iteration statements (FOR)
  - Jump statements (GOTO, EXIT)
  - Expression statements (function calls (CALL), assignment statements (SET), operations, etc.)
  - Block statements (())
- An executable line of code may not contain the following statements:

- 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	N/A	Defined in 1.4
Compiler Directives	3	N/A	Defined in 1.5
Comments			Defined in 1.7
On their own lines	4	Not Included (NI)	
Embedded	5	NI	
Banners	6	NI	
Empty Comments	7	NI	
Blank Lines	8	NI	Defined in 1.6

LOGICAL SLOC COUNTING RULES				
NO.	STRUCTURE	ORDER OF PRECEDENCE	LOGICAL SLOC RULES	COMMENTS
R01	“FOR”, “IF” or “CHOICE” statement	1	Count Once	“FOR” is an independent statement.
R02	Statements ending by a newline	2	Count once per statement, including empty statement	A newline used with R01 is not counted.
R03	Block delimiters, parenthesis (…)	3	Count once per pair of parenthesis (..), except an left parenthesis comes after a keyword “else”.	Parenthesis used with R01 and R02 are not counted.

3. Examples

EXECUTABLE LINES

SELECTION Statement

ESS1 – IF, ELSE and nested IF statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
IF [NOT] ERRORLEVEL number command	IF NOT ERRORLEVEL 1 GOTO END	1 1
	IF ERRORLEVEL 1 SET ERRORLEV=1	2
IF [NOT] string1==string2 command	IF %ERR100%==2 GOTO 200	2
IF [NOT] EXIST filename command	IF EXIST c:\mydir\nul GOTO process	2
	IF NOT EXIST product.dat (ECHO Can't find data file)	1 0 1 0
IF [NOT] <…> (Command) ELSE (Command)	IF EXIST filename. (del filename.) ELSE (echo filename. missing.)	1 1 0 1
	IF EXIST filename. (del filename.) ELSE echo filename.missing	3

ESS2 – CHOICE statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
choice [/C[:]choices] [/N] [/S] [/T[:]c,nn] [text]	choice /c:ync Yes, No, or Continue	1

ITERATION Statement

EIS1 – FOR statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
FOR %%A IN (list) DO command	FOR %? in (1 2 3)	1

[parameters]	do echo %?	1

JUMP Statement

EJS1 – GOTO label

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
GOTO label	if not errorlevel 1 goto end :end Echo done	2 0 1

EJS2 – EXIT

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
EXIT [/b] [ExitCode]	EXIT	1

EXPRESSION Statement

EES1 – CALL another batch program/ CALL a label

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
CALL [drive:][path]filename [batch-parameters]	call checknew %1 %2 CALL end :end	1 1 0

EES2 – SET statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
SET variable=[string]	set include=c:\inc set path=%1;%path%	1 1

BLOCK Statement

EBS1 – block=related statements treated as a unit

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
:: start of block (<definitions>	:: start of block (SET i=2	0 0 1

<statement>) :: end of block	ECHO i) :: end of block	1 0 0

4. Notes on Special Character Processing

1) Quotes:

Start of Quotes:	"\""
End of Quotes:	"\""
Escape Front Quotes:	'\\'

2) Line Continue: "^"

3) Extension for Batch: ".bat"