C Shell Script Code Count™

Counting Standard

University of Southern California

Center for Systems and Software Engineering

December , 2016

Revision Sheet

Date	Version	Revision Description	Author
6/22/2016	1.0	Original release	Matthew Swartz

Table of Contents

No.	Contents	Page No.
1.0	Definitions	4
1.1	SLOC	4
1.2	Physical SLOC	4
1.3	Logical SLOC	4
1.4	Data declaration line	4
1.5	Compiler directive	4
1.6	Blank line	4
1.7	Comment line	4
1.8	Executable line of code	4
2.0	Checklist for source statement counts	6
3.0	Examples of logical SLOC counting	7
3.1	Executable Lines	7
3.1.1	Selection Statements	7
3.1.2	Iteration Statements	8
3.1.3	Jump Statements	8
3.1.4	Expression Statements	9
4.0	Complexity	10

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.

1.5. **Compiler Directives –** A statement that tells the compiler how to compile a program, but not what to compile. C Shell Script does not contain any 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.

C Shell Script comment delimiters are "#". 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, switch, switch)
  - Iteration statements (foreach, while)
  - Empty statements (one or more ";")
  - Jump statements (goto, break, continue, exit)
  - Expression statements (function calls, assignment statements, 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	NA	Defined in 1.4
Compiler directives	3	NA	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	"*foreach*", "*while*" or "*if*" statement	1	Count once.	"*while*" is an independent statement.
R02	Statements ending by a semicolon or newline	2	Count once per statement, including empty statement.	

3. Examples

EXECUTABLE LINES

SELECTION Statement

ESS1 - if, else if, else and nested if statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

if (<*Boolean expression*>) then
  <*statements*>
endif
	if ($x != 0) then
	  echo “non-zero”
	endif
		1
		1
		0

if (<*Boolean expression*>) then
  <*statements*>
else
  <*statements*>
endif
	if ($x > 0) then
	  echo “positive”
	else
	  echo “negative”
	endif
		1
		1
		0
		1
		0

if (<*Boolean expression*>) then
  <*statements*>
else if (<*Boolean expression*>)
  <*statements*>
else
  <*statements*>
endif
	if ($x == 0) then
	  echo “zero”
	else if ($x > 0) then
	  echo “positive”
	else
	  echo “negative”
	endif
		1
		1
		1
		1
		0
		1
		0

if (<*Boolean expression*>) <*statement*> NOTE: complexity is not considered, i.e. multiple “&&” or “||” as part of the expression.	if ($x != 0) echo “non-zero”	2

ESS2 - switch and nested switch statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

switch (<*string*>)
  case <*string 1*>:
  <*statements*>
  breaksw
  case <*string 2*>:
  <*statements*>
	switch ($str)
	  case “1”:
	  echo “one”
	  breaksw
	  case “2”:
	  echo “two”
		1
		0
		1
		1
		0
		1

breaksw case <*string 3*>: <*statements*> break default: <*statements*> endsw	breaksw case "3": echo "three" breaksw default: echo "invalid case" endsw	1 0 1 1 0 1 0

ITERATION Statement

EIS1 - foreach

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
foreach <*name*> (<*wordlist*>) <*statements*> End	foreach i ($d) echo $i end	1 1 0

EIS2 - while

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
while <*expression*> <*statements*> End	while ($j <= 10) echo '2 **' $j = $i @ i *= 2 @ j++ end	1 1 1 1 0

JUMP Statement

EJS1 – goto, label

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
goto *label* . . label:	loop1: $x++ if ($x < $y) goto loop1	0 1 2

EJS2 – break

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
break	if ($i > 10) break	2

EJS3 – exit function

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
exit *return_code*	if ($x < 0) exit 1	2

EJS4 – continue

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

continue	while ($i < 10)
	  $i++
	  if ($i == 5) then
	  continue
	  else
	  $j++
	  endif
	end
		1
		1
		1
		1
		0
		1
		0
		0

EXPRESSION Statement

EES1 – function call

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<*function_name*> (<*parameters*>)	read_file (name)	1

EES2 – assignment statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
$<name> = <value>	$a = “value”	1

EES3 – empty statement(is counted as it is considered to be a placeholder for something to call attention)

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
one or more “;” in succession	;	1 each

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	Pointer
			>>	else if	>=		=	
			<<	foreach	<=			
			++	switch	!=			
			--	while	==			
			+	case	&&			
			-	else	||			
			/	if	=~			
			*		!~			
			%		!			
					~			
					>			
					<			
					&			
					^			
					|			