2022

Programming 2

SHORT CIRCUIT EVALUATION

MANUEL HOLGUIN

09-30-2022 |

Contents

Problem Description

For this programming project we were tasked with evaluating each of the following programming languages: ADA, C-Shell (borne or Kshell), PHP, and PERL for short circuit evaluations. To explain we run a conditional function in any of the languages if statement1 is false AND statement 2 is true does the language immediately terminate the condition statement or does it still evaluate the second statement?

Program Findings Summary

Below is a summary of the results for my experiments.

Program Language	IF (TRUE and FALSE)	IF(FALSE and TRUE)	CONCLUSION
ADA	TEST EVALUATED TRUE AND FALSE	TEST CASE EVALUATED FALSE AND TRUE	Supports short circuit by the use of "AND then" Otherwise using AND only does not support short circuit. Image 1
BSHELL	TEST EVALUATED TRUE AND FALSE	TEST CASE EVALUATED FALSE ONLY	Supports short circuit operators test case for "and" operator. Image 2
PHP	TEST EVALUATED TRUE AND FALSE	TEST CASE EVALUATED FALSE ONLY	Supports short circuit operators test case for "and" operator. Image 3
PERL	TEST EVALUATED TRUE AND FALSE	TEST CASE EVALUATED FALSE ONYL	Supports short circuit operators test case for "and" operator. Image 4

ADA program results and output

Ran the program using a function f which prints an evaluation statement and returns the Boolean True. If statement is F and T if I see for T it also printed the evaluation statement, I was able to tell that ADA did not support short circuiting. As it evaluated both statements. Although ADA can have short circuit implemented using the “and then” command in the if statement.

```
gnatbind -x ADACode.ali
gnatlink ADACode.ali
nholguin@bach:~/CS471/projects/program2> ./ADACode
---------Test case 1 T and F---------
I have been evaluated False
Test 1 passed T and F
---------Test case 2 F and T ---------
I have been evaluated True
Test 2 passed  F and T if it prints I have been evaluated then ADA does not supp
ort short circuit.
---------Test case 2.1 F and T ---------
Test 2 passed F and T in this scenario utilizing AND THEN implements short circu
it.  Since function f did not print out it's Put_line It proves the hypothesis.
---------Test case 3 T and T ---------
I have been evaluated True
I have been evaluated True
Test 3 statement A is T statement B Is T this should  print
---------Test case 4 F and F ---------
I have been evaluated False
Test 4 passed F and F if the eval statement prints then short circuit is not sup
ported since statement 1 was already false no need to check second statement.
nholguin@bach:~/CS471/projects/program2>
```

Figure 1 Output of ADA code

BShell program results and output

Bash was a little harder to evaluate but I was able to conclude that bash does indeed have short circuit evaluation as proven in Test #3 when F and T the program only evaluated False and exited the if statement.

```
m2018@DESKTOP-GVCGB19 MINGW64 ~/OneDrive/Documents/NMSU/Fall 2022/C
ents/programming2/projects/Bshell
$ ./cshellcode.sh
./cshellcode.sh: line 21: $b: ambiguous redirect
This true statement has been evaluated!
Test one was a success T and T
./cshellcode.sh: line 29: ==: command not found
This false statement has been evaluated!
Test 2 was a Success T and F = F
This false statement has been evaluated!
Test 3 was a success F and T = F
This false statement has been evaluated!
This false statement has been evaluated!
Test 4 was a success F and F = F do not print
```

Figure 2 Screen Shot of Bourne Shell output.

PHP program results and output

In PHP I used the same approach as ADA and found that is does indeed have short circuit operators and fails to execute the second statement of the and condition if the first one has already been proven false. Image below if the language did not support short circuit the same message as in test 1 would have appeared for test 3 and test 4.

```
[Running] php "c:\Users\m2018\OneDrive\Documents\NMSU\Fall 2022\CS
471\assignments\programming2\projects\PHP\index.php"
Hello Human!
evaluating Statement 1 and Statement 2
Test 1 was passed! T and T = T
evaluating Statement 1 and Statement 2
Test 2 passed T and F = F did not print if statement.
Test 3 passed F and T = F did not print if statement.Short circuit.
Test 4 passed F and F = F did not print if statement.Short circuit.

[Done] exited with code=0 in 0.073 seconds
```

PERL program results and output

PERL evaluates programs using short circuiting following the same steps as the previous languages I used multiple if statements to check for each individual possibility. I used function calls to evaluate if PERL was running the second statement after the first one had tested F.

```
[Running] perl "c:\Users\m2018\OneDrive\Documents\NMSU\Fall 2022\CS
471\assignments\programming2\projects\Perl\program2.pl"
evaluating and T statement.
A T and T = T and works for this test works
evaluating and F statement.
Test 2 showed that Perl evaluated both statements correctly
evaluating and F statement.
Test 3 showed that Perl evaluated 1st statement correctly and ended conjunction
evaluating and F statement.
Test 4 showed that Perl evaluated 1st statement correctly and ended conjunction

[Done] exited with code=0 in 0.053 seconds
```

Appendix

Code Base

ADA Code Base

```
--Title:Ada Short Circuit program.

--Author: Manuel Holguin

--Date 09-29-2022 Re-submission

--Purpose: Is to evaluate the AND short circuit Boolean in ADA

--      Each individual case tests for a posibilities in which the function f and fal's

--      Put_Line statements are printed. If for any of the test cases if the functions' lines

--      are printed then this means that ADA doesn't support short circuit evaluation.

--      However an addition to the code in test case 2.1 it was realized that using the follow

--      syntax if(<expression1> AND the <expression2>) then will actually implement short circuit evaluation.

with Ada.Text_IO;

use Ada.Text_IO;

with Ada.Integer_Text_IO;

use Ada.Integer_Text_IO;


procedure ADACode is

A : Integer := 0;

B : Integer := 50;


--If this function's Put_Line prints when statement 1 is False and Statement 2 is T then

--ADA does not support short circuiting

--However if this print statement doesnt print when statement 1 is false then short circuiting is supported.

function f return Boolean is

begin

   Put_Line("I have been evaluated True");

    return True;
```

```
end f;
-- This function evaluates to false by default.  The purpose is to check in the scenario where
-- If<expr1> == false And <expr2> this function will be replace <expr2> if the Put_line prints
-- Then it means short circuit evaluation is not supported in ADA all Statements get checked.
function fal return Boolean is
begin
   Put_Line("I have been evaluated False");
    return False;
end fal;

begin
--THE FOLLOWING TEST SHORT CIRCUIT IN THE AND OPERATOR--
   --Test case 1 T and F
   Put_Line("---------Test case 1 T and F---------");
   if B > A and fal then
      Put_Line("Test 1 Stament 1 is True but statement 2 is False this should not print.");

   else
      Put_line("Test 1 passed T and F");
   end if;

   --Test case 2 F and T
   Put_Line("---------Test case 2 F and T ---------");
   if B < A and f then
      Put_Line("Test 2 statement A is False but statement 2 Is True this should not print");

   else
     Put_Line("Test 2 passed  F and T if it prints I have been evaluated then ADA does not support short
circuit. ");
```

```
   end if;
   --Test case 2.1 F and T using AND Then
   Put_Line("---------Test case 2.1 F and T ---------");
   if B < A and then f then
      Put_Line("Test 2 statement A is False but statement 2 Is True this should not print");

   else
     Put_Line("Test 2 passed F and T in this scenario utilizing AND THEN implements short circuit.  Since function f did not print out it's Put_line It proves the hypothesis.");
   end if;

   --Test case 3 T and T
   Put_Line("---------Test case 3 T and T ---------");
   if f and f then
      Put_Line("Test 3 statement A is T statement B Is T this should  print");

   else
     Put_Line("Test 3 Failed!");
   end if;

   --Test case 4 F and F
   Put_Line("---------Test case 4 F and F ---------");
   if B < A and fal then
      Put_Line("Test 4 statement A is False but statement 2 Is True this should not print");
   else
     Put_Line("Test 4 passed F and F if the eval statement prints then short circuit is not supported since statement 1 was already false no need to check second statement.");
   end if;
end ADACode;
```

Bourne-Shell Code Base

```
#!/bin/bash

#Title: ProgrammingAssignment 2 B-Shell short Circuit program

#Author: Manuel Holguin

#Date: 09-29-2022 Re-submission

#Description: Runs AND operator evaluations to check for short circuit in B-Shell

#       Tests are run with using function calls within the functions is an

#       echo statement which lets me know it was evaluated and

#       a return value depending on the usage of the function it will return

#       true or false.


set ${a}=10

set ${b}=20


#functions used to return T or F to the test cases

function isTrue(){

echo "This true statement has been evaluated!"

return 0

}

#This function evaluates to false by default.  The purpose is to check in the scenario where

#If<expr1> == false And <expr2> this function will be replace <expr2> if echo prints

#Then it means short circuit evaluation is not supported in B-Shell all Statements get checked.

function isFalse(){

echo "This false statement has been evaluated!"

return 1

}


#Case 1 testing T and T
```

```
if ( ($a < $b)  &  (isTrue) );
then
   echo "Test one was a success T and T"
else
   echo "Test one failed T and T = T this should not print."
fi

#Case 2 testing T and F
if ( ($a == "10" )  &  ( isFalse) );
then
   echo "Test 2 was a Failure T and F  this should not print."
else
   echo "Test 2 was a Success T and F = F"
fi

#Case 3 testing F and T
if ((isFalse ) &&  (isTrue));
then
   echo "Test 3 was a failure F and T should not print"
else
   echo "Test 3 was a success F and T = F"
fi

#Case 4 testing F and F
if ((isFalse) &  (isFalse)); then
   echo "Test 4 was a failure F and F = F should not print"
else
   echo "Test 4 was a success F and F = F do not print"
fi
```

PHP Code Base

```
<?php
//Title: Programming Assignment 2 PHP Short Circuit Operator
//Author: Manuel Holguin
//Date: 09-29-22 Re-submission
//Description: Tests for short circuiting AND operator in PHP.
//          The implementation of the program utilizes two
//          functions ev and evfals. That print out a statement
//          in order to evaluate if short circuiting is used
//          inside the PHP language.
echo "Hello Human!\n";

//If either of these two functions arent printed during testing then
//This means the PHP is a short circuiting programming language.
function ev(){
    echo "evaluating Statement 1 and Statement 2 \n";
    return True;

}
function evfals(){
    echo "evaluating Statement 1 and Statement 2 \n";
    return False;

}
    $number1 = 1;
    $number2 = 2;

    //Test case 1 for T and T
    if($number1 < $number2 and ev() )
```

```
        $case1 = "Test 1 was passed! T and T = T\n";
    else {
        $case1 = "Test was failure\n";
    }

    echo $case1;
    //Test Case 2 for T and F
    if($number1 < $number2 and evfals() )
    $case1 = "Test 2 failed T and F = F this should not print\n";
    else {
    $case1 = "Test 2 passed T and F = F did not print if statement.\n";
    }

    echo $case1;

    //Test Case 3 for F and T
    if($number1 > $number2  and ev() )
    $case1 = "Test 3 failed F and T = F this should not print\n";
    else {
    $case1 = "Test 3 passed F and T = F did not print if statement.Short circuit.\n";
    }

    echo $case1;

    //Test Case 4 for F and F
    if($number1 > $number2 and evfals() )
    $case1 = "Test 4 failed F and F = F this should not print\n";
    else {
    $case1 = "Test 4 passed F and F = F did not print if statement.Short circuit.\n";
```

```
    }


    echo $case1;
?>
```

PERL Code Base

```
#Title: PERL Programming Assignment 2 Short Circuit evaluation
#Author: Manuel Holguin
#Date: 09-29-2022 Re-submission
#Description: Program evaluates the and conjunction
#       in order to see if the language implementes short circuiting.
#       The utilization of fucntion calls allowed me to evaluate if the
#       operator indeed short circuited or tested both statements after
#       the value of the first one was established.

sub falsy_func{
   print "evaluating and F statement.\n";
   return 0;

}
sub truthsly{
   print "evaluating and T statement.\n";
   return 1;

}
sub short_c{

   #test1 T and T
   if(1 == 1 && truthsly ){
   print "A T and T = T and works for this test works\n";
   }
   else{
   print "Test 1 failed\n"
```

```
    }
    #Test 2 T and F will implement a function to make sure the statement is evaluated
    if(1 == 1 && falsy_func ){
    print "A T and F = F  if this prints short circuit did not eval second Statement\n";
    }
    else{
    print "Test 2 showed that Perl evaluated both statements correctly \n"
    }
    #Test 3 F and T if short circuit should exit after evaluating statement 1
    # to make sure statement 2 is check function truthsly will be called returning T
    if( falsy_func && truthsly ){
    print "A F and T = F  Only the Eval for falsy should run \n";
    }
    else{
    print "Test 3 showed that Perl evaluated 1st statement correctly and ended conjunction \n"
    }
    #Test 4 tests for F and F = F should only eval first statement
    if( falsy_func && falsy_func ){
    print "A F and F = F  This statement should not run. \n";
    }
    else{
    print "Test 4 showed that Perl evaluated 1st statement correctly and ended conjunction \n"
    }
}
short_c();
```