

EEP702-Software Lab  
Assignment 2 : String Occurrence  
Count Using Shell Script and  
C++ program

Harshit Kumar Gupta  
2013EET2369



Computer Technology  
Department Of Electrical Engineering  
IIT DELHI

January 20, 2014

# Contents

1	<a href="#"><u>PROBLEM STATEMENT</u></a>	2
2	<a href="#"><u>ABSTRACT</u></a>	3
3	<a href="#"><u>INTRODUCTION</u></a>	4
4	<a href="#"><u>SPECIFICATIONS AND ASSUMPTIONS</u></a>	5
5	<a href="#"><u>LOGIC USED/METHODOLOGY</u></a>	6
6	<a href="#"><u>EXECUTION INSTRUCTIONS</u></a>	7
7	<a href="#"><u>FLOWCHART</u></a>	8
8	<a href="#"><u>OUTPUT</u></a>	11

# Chapter 1

## PROBLEM STATEMENT

String manipulation using shell scripts - writing a program (called NumOfOccurrences) to find how many times a substring s2 occurs in a bigger string s1

Using csh, sed and awk (either separately or together - see pipes and |tee connectors to send the output of one program into another)

### 1. Part (a)

Find the total number of occurrences of a substring s2 to be entered by the user. Logically divide the string s1 into 'm' number of even partitions and parallelly search to count the total number of occurrences of s2 parallelly using threads, where 'm' is the number of threads to be entered by the user.

### 2. Part (b)

Let the length of string s1 [denoted as  $\text{len}(s1)$ ] be  $n1$  and number of threads be  $m$ , then the length of each partition would be  $(n1/m)$  and  $n2 = \text{len}(s2)$  ( $< (n1/m)$ ). During runtime, print using an external subroutine StringPrint(\*char s) the 'id' of thread in which the substring s2 is found. Each thread will find the occurrence in its local string partition. IPC and message queues can be used for communication among threads, and finally print the total number of occurrences of s2 in s1.

### 3. Part (c)

Repeat the above using any one of the languages C / C++ / Java

## Chapter 2

### ABSTRACT

This report includes description on two codes designed for above two problems. The first code deals with string manipulation in bash scripts and the second program which is in C++, deals with string manipulation using the concept of threading. Threading gives us an upper edge while processing concurrently the string matching algorithm to provide us the output in a very efficient and fast manner. The string s1 contains the text and string s2 contains the search texts, the string s1 is divided into many parts as per the no of threads is provided and concurrently starting the process of matching the desired string s2 in s1.

# Chapter 3

## INTRODUCTION

The first program which is a bash script computes the search and prints no. of occurrence of string s2 in s1.

Bash is a Unix shell written by Brian Fox for the GNU Project as a free software replacement for the Bourne shell (sh). Released in 1989, it has been distributed widely as the shell for the GNU operating system and as a default shell on Linux and Mac OS X. Bash is a command processor, typically run in a text window, allowing the user to type commands which cause actions. Bash can also read commands from a file, called a script. Like all Unix shells, it supports filename wildcarding, piping, here documents, command substitution, variables and control structures for condition-testing and iteration.

The second program is the same as the first but the only difference is that the second program is in C++ code and concepts of threads are used in it to execute the search in a very efficient time. Multi-threading is a widespread programming and execution model that allows multiple threads to exist within the context of a single process. These threads share the process' resources, but are able to execute independently. The threaded programming model provides developers with a useful abstraction of concurrent execution. Multi-threading can also be applied to a single process to enable parallel execution on a multiprocessing system. This advantage of a multithreaded program allows it to operate faster on computer systems that have multiple CPUs, CPUs with multiple cores, or across a cluster of machines, because the threads of the program naturally lend themselves to truly concurrent execution.

# Chapter 4

## SPECIFICATIONS AND ASSUMPTIONS

### Specifications

1. A substring s2 is input by the user in either of the following ways (a) either the user types s2 on the terminal or (b) from the shell prompt in two ways (again!)
  - (a) INVOCATION 1 : NumOfOccurrences s2 (only one string is given and it is understood as s2,s1 is hardcoded in the program)
  - (b) INVOCATION 2 : NumOfOccurrences s1 s2 (both strings are specified on the command line and the sequence determines which s s1 and which is s2)
2. The output of StringPrint(.) looks moreover like this: <current UTC time> <id of thread where s2 is found> [n times]  
Total embeddings of s2 in s1 = no of occurrences.

### Assumptions

1. The first string can be a text file containing string s1 of alphanumeric characters.
2. The second string s2 can be given as argument or can be taken as input from the user.
3. The last line of output only the words s2 and s1 are bold (emphasized) when the sentence is printed on the terminal.

# Chapter 5

## LOGIC USED/METHODOLOGY

The methodology that is used for developing the program is defined below:

1. First the string s1 and s2 is fed as input to the program as command line argumants.
2. we check if size of substring is grater than bigger string tehn we exit.
3. We start from first character of main string and match with first character of substring .if match then increment postion by 1 in both string
4. if no mismatch is found and substring is completed then we increment no of occurrences.
5. if mismatch is found then starting postion for matching in main string is incremented by one.
6. We repeat the above procedure till we reach at end of first string.
7. To achieve parallism in matching procedure we can use threading in c++ using pthread library.
8. To achieve parallism in bash program we can run matching procedure as backgroud process .OS schedules background process in parallel.

# Chapter 6

## EXECUTION INSTRUCTIONS

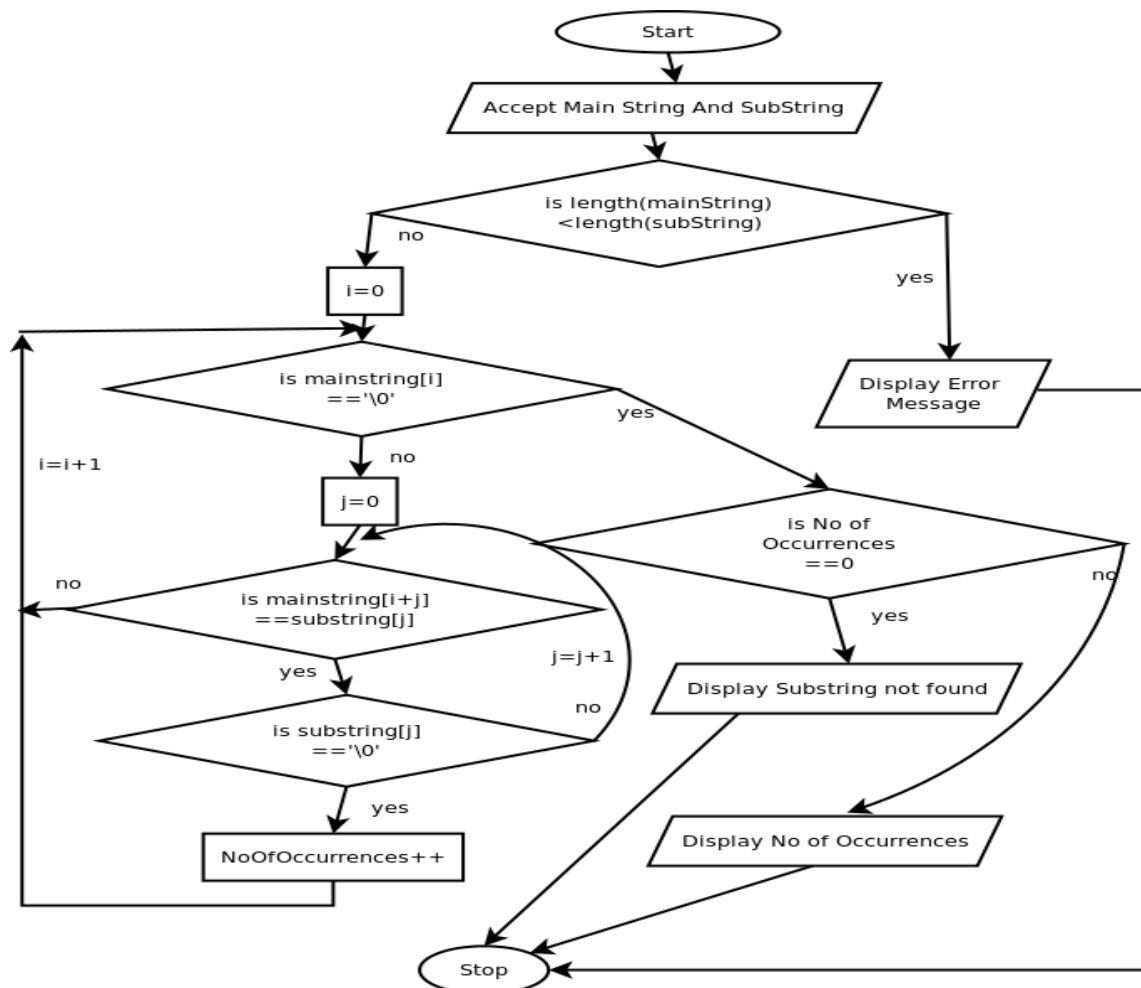
1. For first program in bash following instructions are used.
  - (a) `chmod 777 parallel-count.sh`
  - (b) `parallel-count.sh <s1> <s2>`
2. For second program in C++ following instructions are used.
  - (a) `g++ -Wall NoOfOccurrences.cpp -lpthread`
  - (b) We can also use makefile by — `make -f Makefile1`
  - (c) to excute the program use—  
`NoOfOccurrences <s1> <s2>`

Repeat the above instructions for different input strings of s1 and s2.



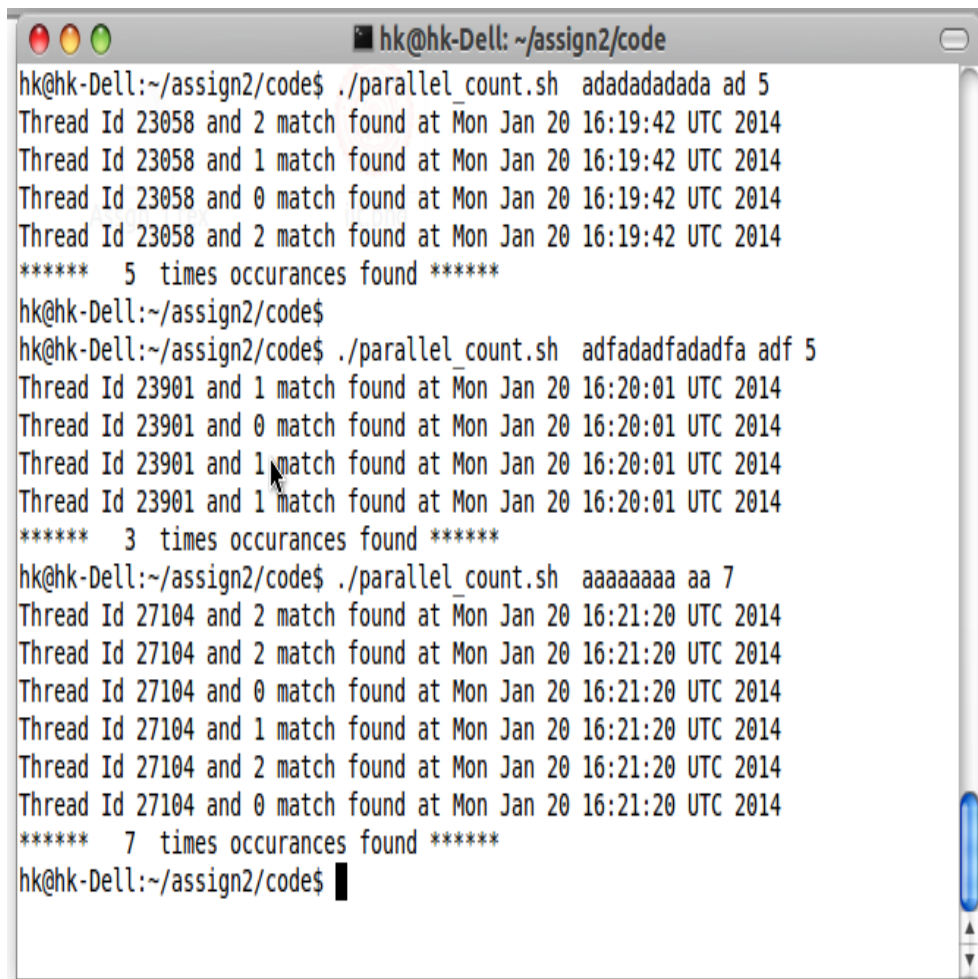
# Chapter 7

## FLOWCHART



# Chapter 8

## OUTPUT

A terminal window titled 'hk@hk-Dell: ~/assign2/code' showing the execution of a script named 'parallel\_count.sh'. The script is called three times with different inputs. Each call prints the number of matches found for each of five threads and a summary line. The first call uses 'adadadadada ad 5' and finds 5 occurrences. The second call uses 'adfadadfadadfa adf 5' and finds 3 occurrences. The third call uses 'aaaaaaaa aa 7' and finds 7 occurrences.

```
hk@hk-Dell:~/assign2/code$ ./parallel_count.sh adadadadada ad 5
Thread Id 23058 and 2 match found at Mon Jan 20 16:19:42 UTC 2014
Thread Id 23058 and 1 match found at Mon Jan 20 16:19:42 UTC 2014
Thread Id 23058 and 0 match found at Mon Jan 20 16:19:42 UTC 2014
Thread Id 23058 and 2 match found at Mon Jan 20 16:19:42 UTC 2014
***** 5 times occurrences found *****
hk@hk-Dell:~/assign2/code$
hk@hk-Dell:~/assign2/code$ ./parallel_count.sh adfadadfadadfa adf 5
Thread Id 23901 and 1 match found at Mon Jan 20 16:20:01 UTC 2014
Thread Id 23901 and 0 match found at Mon Jan 20 16:20:01 UTC 2014
Thread Id 23901 and 1 match found at Mon Jan 20 16:20:01 UTC 2014
Thread Id 23901 and 1 match found at Mon Jan 20 16:20:01 UTC 2014
***** 3 times occurrences found *****
hk@hk-Dell:~/assign2/code$ ./parallel_count.sh aaaaaaaaa aa 7
Thread Id 27104 and 2 match found at Mon Jan 20 16:21:20 UTC 2014
Thread Id 27104 and 2 match found at Mon Jan 20 16:21:20 UTC 2014
Thread Id 27104 and 0 match found at Mon Jan 20 16:21:20 UTC 2014
Thread Id 27104 and 1 match found at Mon Jan 20 16:21:20 UTC 2014
Thread Id 27104 and 2 match found at Mon Jan 20 16:21:20 UTC 2014
Thread Id 27104 and 0 match found at Mon Jan 20 16:21:20 UTC 2014
***** 7 times occurrences found *****
hk@hk-Dell:~/assign2/code$
```

Output of Bash Script

```
hk@hk-Dell: ~/assign2/code
hk@hk-Dell:~/assign2/code$ ./NoOfOccurrences adadadadadada ada 8
usage: Occurrences <BigString> <SubString>
hk@hk-Dell:~/assign2/code$ ./NoOfOccurrences adadadadadada ada
Enter Number of threads to count occurrences 6
thread Id 3069823856 1 match found UTC time:Mon Jan 20 16:23:08 2014
thread Id 3061431152 2 match found UTC time:Mon Jan 20 16:23:08 2014
thread Id 3053038448 1 match found UTC time:Mon Jan 20 16:23:08 2014
thread Id 3044645744 1 match found UTC time:Mon Jan 20 16:23:08 2014
thread Id 3078216560 2 match found UTC time:Mon Jan 20 16:23:08 2014
*****7times occurrences found*****
hk@hk-Dell:~/assign2/code$ ./NoOfOccurrences
usage: Occurrences <BigString> <SubString>
hk@hk-Dell:~/assign2/code$ ./NoOfOccurrences aaaaaaaaaaaaa aa 5
usage: Occurrences <BigString> <SubString>
hk@hk-Dell:~/assign2/code$ ./NoOfOccurrences aaaaaaaaaaaaa aa
Enter Number of threads to count occurrences 5
thread Id 3077917552 3 match found UTC time:Mon Jan 20 16:23:50 2014
thread Id 3061132144 3 match found UTC time:Mon Jan 20 16:23:50 2014
thread Id 3052739440 3 match found UTC time: thread Id 3069524848 3 match f
ound UTC time:Mon Jan 20 16:23:50 2014
*****12times occurrences found*****
hk@hk-Dell:~/assign2/code$
```

Output of C++ program

## Chapter 9

# RESULTS AND CONCLUSIONS

In both the cases the value output of the Bash Script and `c++` code is displayed on the user screen using the `echo` or `cout` command for the strings. In both the cases the output comes out to be as expected and is verified for all the conditions.