Ismail Tirmizi # 1111 56

This assignment is about implementing important functionalities in string using smart pointers. But the catch is that the code for smart pointers is to be written as well.

Smart String

Advance Programming Assignment 2

# Introduction

Our task is to provide implementation of Strings using the following types of smart pointers:

1. Copied Pointers
2. Owned Pointers
3. COW with Reference Counting
4. COW with Reference Linking

Task 1. And 3. Are implemented by Sir (Thank God). We have to just implement the 2. Owned pointers and 4. COW with reference linking.

# GitHub Link

https://github.com/herozero777/Assignment-2-AP

# How to run the application

Simply go to main and uncomment the implementation you want to try. Comment the implementation you are not using. Only one implementation can be run at a time.

# **Problem**

Our first task is to understand the code given to us by Sir Fahad Satti. Luckily we got many errors, so woohoo more work.

* I have spotted a memory leak in StringBuffer::StringBuffer(char\* newString, int length) file: StringBuffer.cpp. There is a memory leak when sir allocates a memory and then start to point at the user’s passed character array, while leaving the allocated memory alone.
* Next we tried run the code and immediately got error.
* Next we got error from the destructor in String & StringBuffer.

Next step is to implement the point 2. Ownership pointers.

* Need a way to keep track of ownership
* Next there was a need to implement the remaining packages of string.

So there was a problem of passing a string initialized in the stack memory. So when destructor is called onto this piece of memory it throw an exception. So I had to do something.

1. So it turns out that the smartCopy is not so smart after all. It uses what might seem a nifty syntax for character by character string copying but it is actually incorrect.
2. The next task it to implement reference linking.
3. There are 2 case of doubly linked list which needs to be catered.
   1. The first is the case when first object is created for the first time.
   2. The next case is that there are more than one objects in the linked list.

# **Solution**

To do this download the smart string sample code from LMS and read it thoroughly. To diagnose the code we commented everything in main and only left ss2 object uncommented. From there we would uncomment code incrementally when the previous lines would not produce any errors.

* But we are not sure … like not as in how to handle this but whether we should handle it or not.
* The problem was that code was trying to delete \_strbuf without first pointing it to any valid location. File: StringBuffer.cpp. func: StringBuffer::StringBuffer(char\* newString, int length)
* To correct this we changed delete[] this->\_str; to delete this->\_str; in file: String.cpp

For this I have made a new class String2.

* Create a variable bool itsOwn to keep track of ownership.
* I was implementing the functions but soon I realized about a problem.

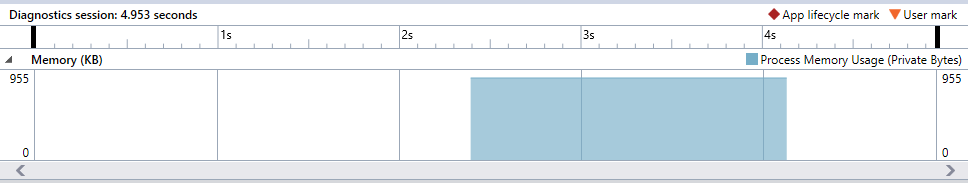
My solution is to make a copy of this string in heap so that we can call onto it easily. File: String.cpp & String2.cpp. func: String::String(char\* newString, int length).

1. File: StringBuffer.cpp. Change: \*\_strbuf++ = \*(newString->\_strbuf)++; to \_strbuf[it] = newString->\_strbuf[it];
2. For that I have created a new file: StringLink.cpp
   1. For this I used the logic that StringLink object points to itself.
   2. For this I used doubly linked list concepts I learned in the Object Oriented Programming course.

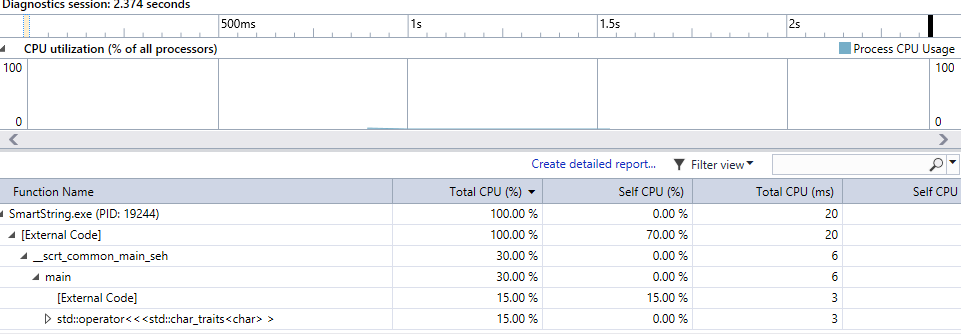
# Profiling

## Copied Pointers

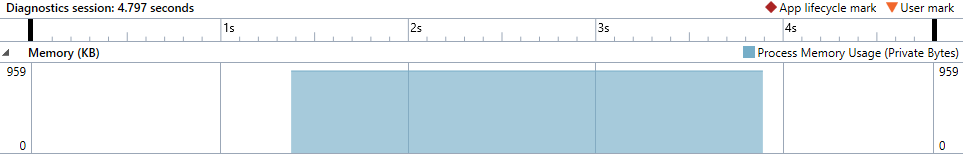
Memory utilization (small string)



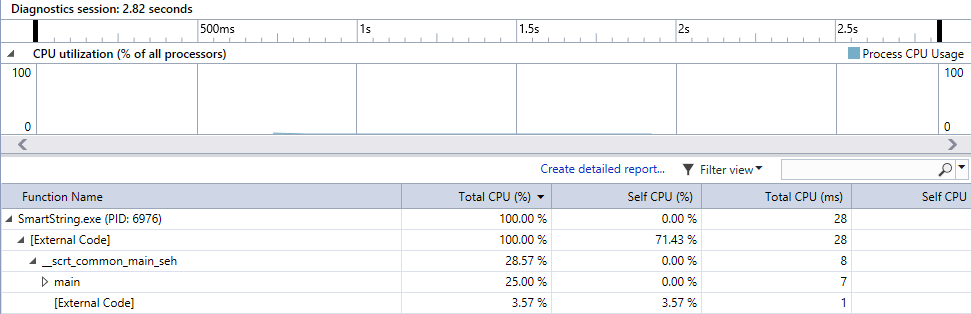
CPU (small string)



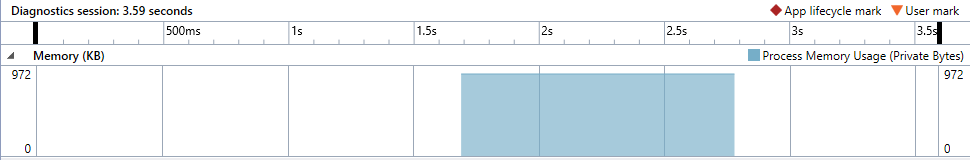
Memory utilization (medium string)



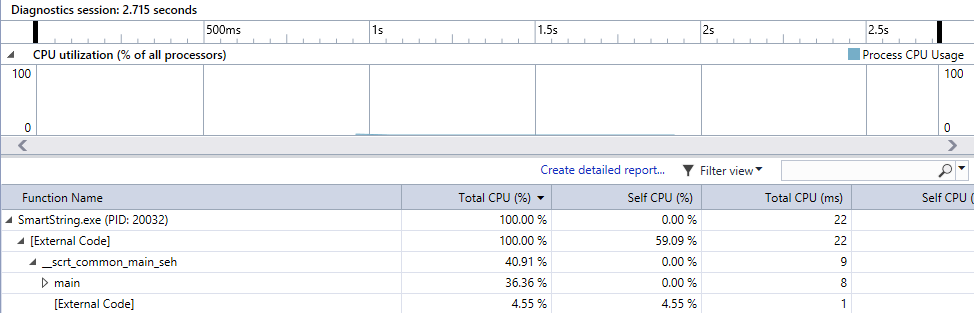
CPU (medium string)



Memory utilization (Large string)

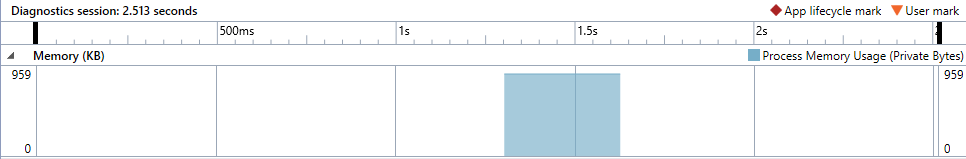


CPU (Large string)

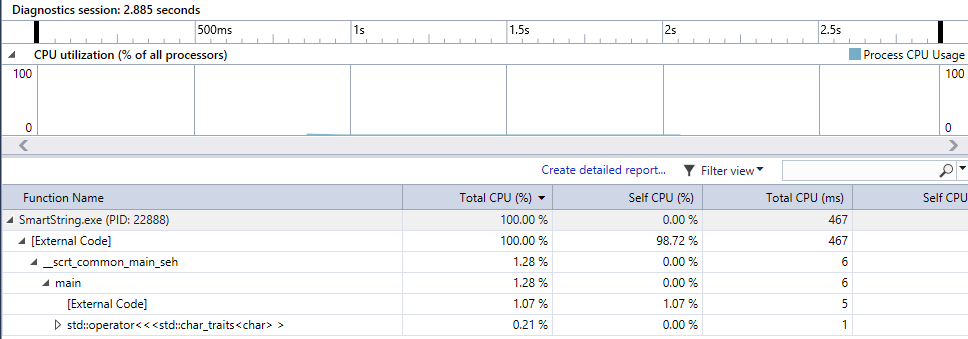


## Owned Pointers

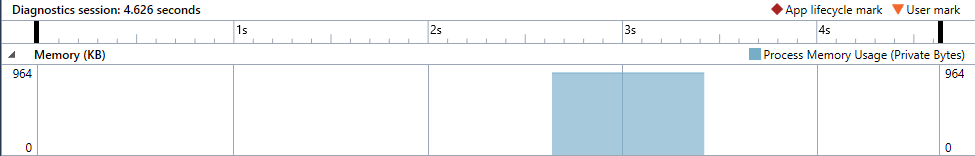
Memory utilization (small string)



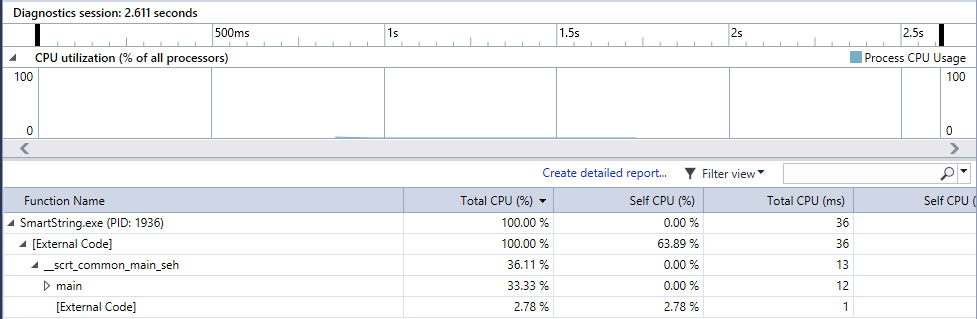
CPU (small string)



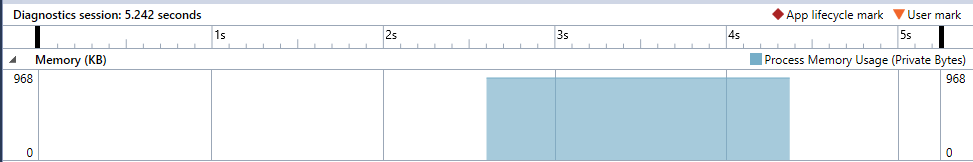
Memory utilization (medium string)



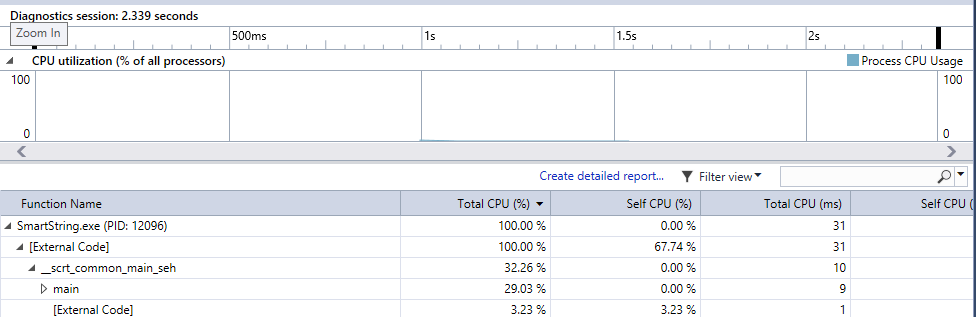
CPU (medium string)



Memory utilization (large string)

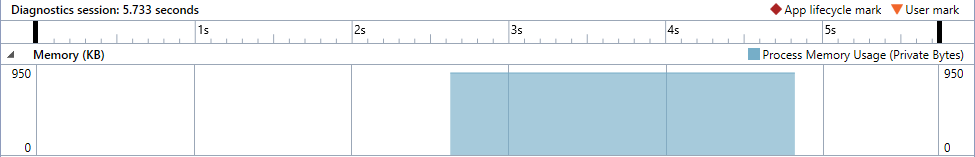


CPU (large string)

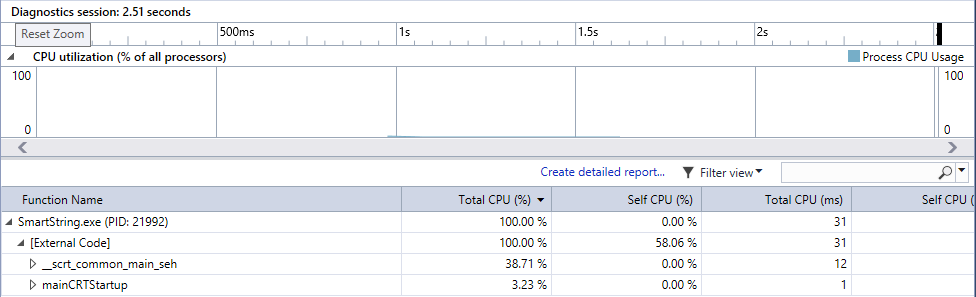


## Copy on write with reference counting

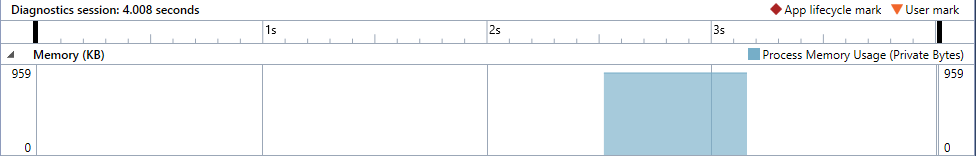
Memory utilization (small string)



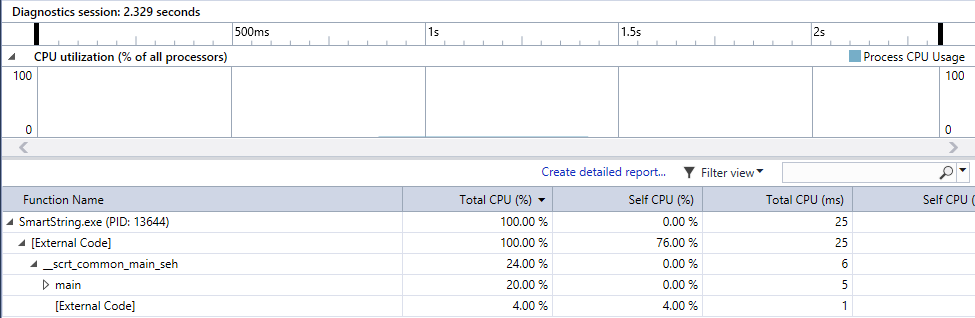
CPU (small)



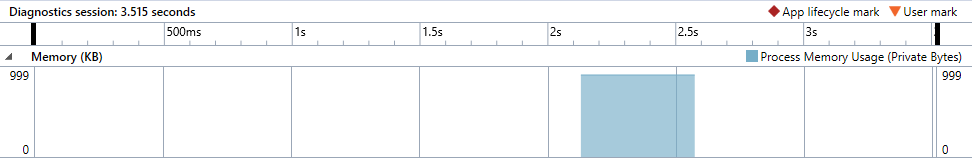
Memory utilization (medium string)



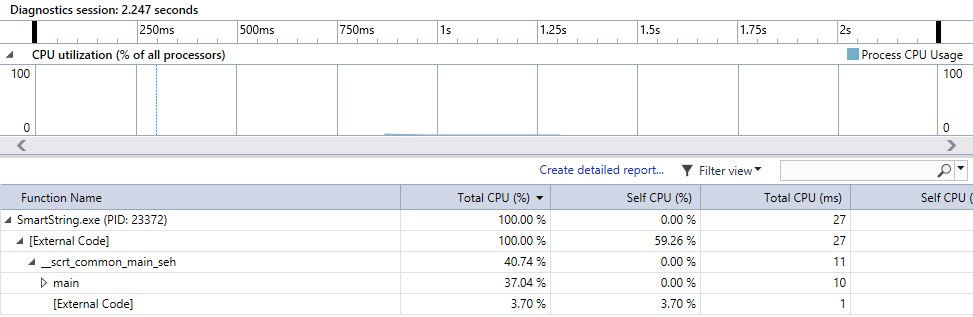
CPU (medium string)



Memory utilization (large string)

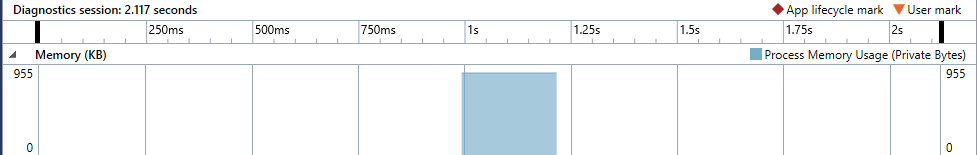


CPU (large string)

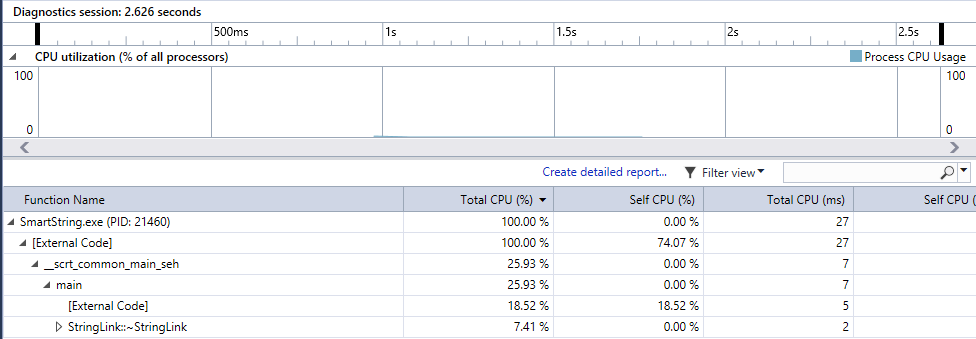


## Copy on write with reference linking

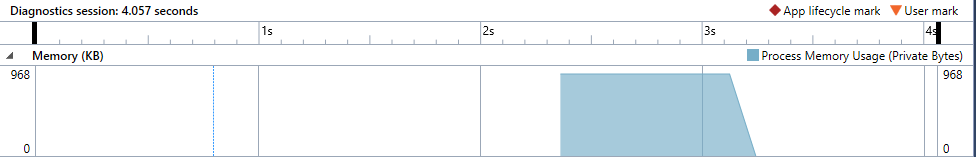
Memory utilization (small string)



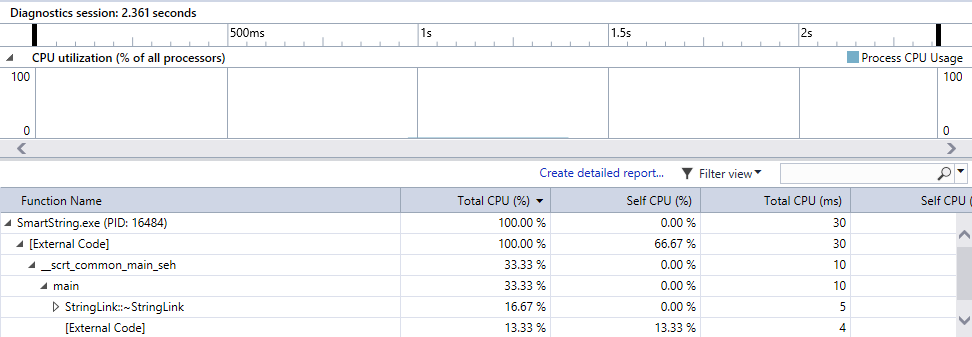
CPU (small string)



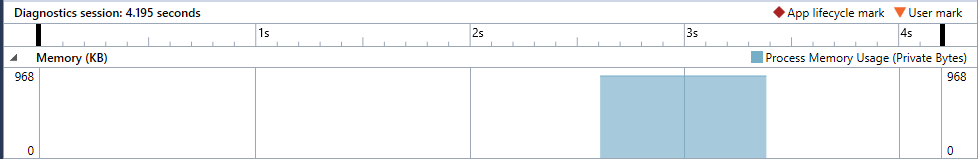
Memory utilization (medium string)



CPU (medium string)



Memory utilization (large string)



CPU (large string)

