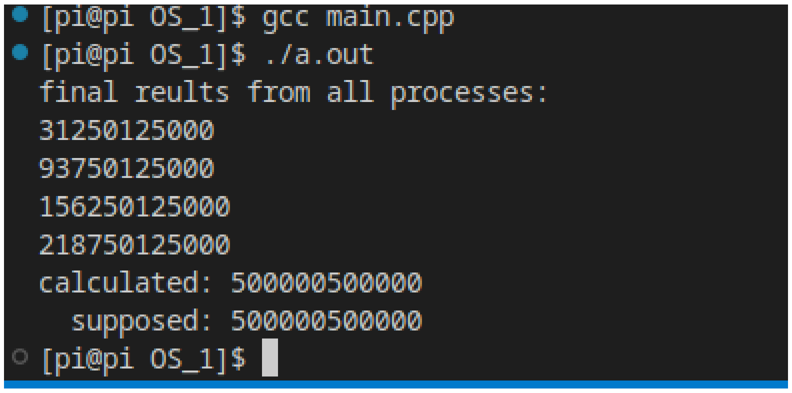
Ιωάννης Τσάμπρας 1066584 5οετης
1η εργασία Λειτουργηκά Συστήματα

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/mman.h>
int main(){
    //set and initialize various variables for the following code
    int numbers=1000000;
    int n=4;
    int i,j;
    pid_t pid1,pid2;
    int start;
    int end;
    int id;
    long long int *sum;


    sum = (long long int*)mmap ( NULL, n*sizeof(long long int),PROT_READ |
PROT_WRITE,MAP_SHARED | MAP_ANONYMOUS, 0, 0 ); //allocate shared memory with mmap

    if(sum == MAP_FAILED){ //check if shared mem allocation failed
    printf("Mapping Failed\n");
    return 1;
    }
        //double fork to spawn 4 processes
        pid1 = fork();
        pid2 = fork();
        //check what process id we have
        if(!pid1 && !pid2){
            id=0;
            //parent parent 0
        }
        if(!pid1 && pid2){
            id=1;
            //parent child 1
        }
        if(pid1 && !pid2){
            id=2;
            //child parent 2
        }
        if(pid1 && pid2){
            id=3;
            //child  child 3
        }
        //sum[id]=0;
        long long int temp=0; //temporary counter
        start=id*(numbers/n); //summing start
        end=(id+1)*(numbers/n); //summing end
        for (j=start+1; j<=end;j++){ //sum from start to end
            temp=temp+j;
        }
        //printf("im the process %d and i calculated :%lli\n",id,temp);//debugging
        sum[id]=temp; //finally add sum to shared memory


        if (!pid1 && !pid2){ //if this is the original process
            wait(NULL); //wait for others to end

            printf("final reults from all
processes:\n%lli\n%lli\n%lli\n%lli\n",sum[0],sum[1],sum[2],sum[3]); //print results
from other processes from shared memory
            long long int final_sum=0; //sum the individual results to one final_sum
            for(i=0;i<=n;i++){
                final_sum=final_sum+sum[i];
            }
            long long int test =500000500000; //using n*(n+1)/2
            printf("calculated: %lli\n  supposed: %lli\n",final_sum,test); //print
both supposed correct answer and calculated one
        }

}
```

Also on github: GitHub

Results:



```
[pi@pi OS_1]$ gcc main.cpp
[pi@pi OS_1]$ ./a.out
final reults from all processes:
31250125000
93750125000
156250125000
218750125000
calculated: 500000500000
  supposed: 500000500000
[pi@pi OS_1]$
```

Method for interprocess communication:
Shared Memory
https://man7.org/linux/man-pages/man2/mmap.2.html