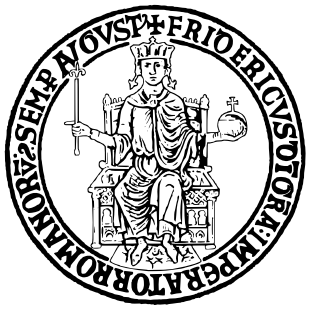
Università degli Studi di Napoli Federico II



Scuola Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione

Corso di Laurea Triennale in Informatica

Corso di Parallel and Distributed Computing

Calcolo della somma di N numeri

Relatore
Prof. Giuliano Laccetti
Prof.ssa Valeria Mele

Candidato
Fabrizio Vitale N97/0000
Giovanni Falcone N97/1111
Luigi Mangiacapra N97/2222

Anno Accademico 2023-2024

Indice

Capitolo 1

Descrizione del problema

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis

cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Capitolo 2

Descrizione algoritmo

Capitolo 3

Descrizione routine

Capitolo 4

Testing

Capitolo 5

Analisi performance

Capitolo 6

Source code

```c
# include <stdio.h>
# include <stdlib.h>
# include <unistd.h>
# include <fcntl.h>
# include <sys/types.h>
# include <string.h>

# define BUFFER_SIZE 4096

static void err_sys(char *msg);

/**
* @brief Create a file object
*
* @param buffer
* @param file_name
*/
static void create_file(char *buffer, char *file_name);

int main(int argc, char **argv){
    char first_buffer[BUFFER_SIZE];
    char second_buffer[BUFFER_SIZE];
    int fd;
    ssize_t n_read;

    if(argc != 3){
        fprintf(stderr, "Usage %s: <file_name> <n>\n", argv[0]);
        return EXIT_FAILURE;
    }

    char *FILENAME = argv[1];
    int N = atoi(argv[2]);

    if((fd = open(FILENAME, O_RDONLY | O_RDONLY)) < 0)
        err_sys("open error");

    if((n_read = read(fd, first_buffer, N)) < 0)
        err_sys ("read error");
    first_buffer[n_read] = '\0';
```

```
40      printf("first buffer:\n%s\n\n", first_buffer);
41
42      // in questo modo evito di fare le altre chiamare ed esco ↩
        direttamente
43      if(N >= lseek(fd, 0, SEEK_END)){
44          printf("The file has been read in its entirety\n");
45          // creo il primo file
46          create_file(first_buffer, "part1");
47          return EXIT_SUCCESS;
48      }
49
50      // se non ho raggiunto la fine del file allora setto il cursore per↩
         leggere la seconda parte
51      if(lseek(fd, N, SEEK_SET) < 0)
52          err_sys("lseek error");
53
54      // leggo finche' e' possibile
55      while((n_read = read(fd, second_buffer, BUFFER_SIZE)) == 0);
56
57      if(n_read < 0)
58          err_sys ("read error");
59
60      printf("second buffer:\n%s\n\n", second_buffer);
61
62      close(fd);
63
64      // creo il primo file
65      create_file(first_buffer, "part1");
66      // creo il secondo file (nessun controllo perche' non sono uscito ↩
        dal programma nel primo if)
67      create_file(second_buffer, "part2");
68
69      return 0;
70  }
71
72
73  void create_file(char *buffer, char *file_name){
74      int fd;
75
76      if((fd = open(file_name, O_RDWR | O_CREAT | O_TRUNC, S_IRWXU)) < 0)
77          err_sys("open error");
78
79      if(write(fd, buffer, strlen(buffer)) != strlen(buffer))
80          err_sys("write error");
81
82      fprintf(stdout, "File %s created!\n", file_name);
83
84      close(fd);
85  }
86
87
88  void err_sys(char *error) {
89      perror(error);
90      exit(1);
```

```
}

```