CENG 305 – Operating Systems – Fall 2022

Project – 1

Group Id: 13
Group Members:
1. Nihal UZUNYAYLA 19050111018
2. Edanur SARIKAYA 19050111004
3. Melih ULUHAN 19050111066
4. Hilal KABANLI 20050111070
5. Ceyda ALTIN 19050111061
6. Yaren ÇETİNKAYA 20050151001

Project Summary & Implementation Details:

Aim of the project is using and creating processes and threads for communication of data. Ordinary pipes were used as IPC mechanism. The hardest part of the project was the implementing inter-process communication for process part. The hardest part of the project was the understanding the which part of thread we should use for the thread part.

For process,
The new process created by the fork () system call. The data was taken from the file and saved in an array. Then, the data in the array was divided into two for two separate processes. Operations such as mad calculation and range calculation by child and parent processes were carried out using 4 ordinary pipes.
To elaborate a little more, the child process calculates the sum of half of all the data and sends it to the parent process. The parent process calculates the average using this sum. The calculated average is sent to the child process by the parent process and the deviations calculation is made in both processes. Then the child process calculates the deviation sum which it was calculated. It sends it to the parent process and the total deviation is calculated by the parent. Finally, the parent process calculates the MAD using the total deviation.
To calculate the range, the maximum and minimum values of the array in both the child and parent processes are calculated. Then the parent process sends its max and min values to the child process in an array. The child process finds the real max and min values by comparing both max and min values. Then using these, it calculates the range.

For basic program,
Range and mad calculations were made with basic c code without using thread and process. Since no communication way is used, the execution time will take longer.

For threads,
In mad_thread.c file we also saved numbers in array. After saving numbers in array, numbers size in array divided into given thread number. With this way we create tread parts with using mutex lock and unlock. We used thread parts for MeanAbsoluteDeviation function and Range function. The advantage of the thread is to make the function faster and make it parallel. In Range part, range calculated with given numbers with thread parts. Each thread part searched for certain part of code. Then searching max and min numbers are subtracted from each other. In main part, the given arguments will be declared. The file will be open. The size of file will be declared. Then, we create 2 separate threads for MeanAbsoluteDeviation and Range functions.

Results & Screenshots:

```
nihal@LAPTOP-2QI8IJF8:~$ touch mad_sequential.c
nihal@LAPTOP-2QI8IJF8:~$ nano mad_sequential.c
nihal@LAPTOP-2QI8IJF8:~$ gcc mad_sequential.c -o mad_sequential
nihal@LAPTOP-2QI8IJF8:~$ ./mad_sequential tiny.txt
Program is reading tiny.txt
Mad is 187.55
Range is 1023.28
Execution time for Range and MAD algorithm is 0.015525 seconds.
nihal@LAPTOP-2QI8IJF8:~$ ./mad_sequential large.txt
Program is reading large.txt
Mad is 27.13
Range is 10480.00
Execution time for Range and MAD algorithm is 0.078025 seconds.
```

```
nihal@LAPTOP-2QI8IJF8:~$ touch mad_process.c
nihal@LAPTOP-2QI8IJF8:~$ nano mad_process.c
nihal@LAPTOP-2QI8IJF8:~$ gcc mad_process.c -o mad_process
nihal@LAPTOP-2QI8IJF8:~$ ./mad_process tiny.txt
Program is reading tiny.txt
The child process is created
Range is 1023.3
MAD is 187.55
Execution time for Range and MAD algorithm is 0.000000 seconds.
nihal@LAPTOP-2QI8IJF8:~$ ./mad_process large.txt
Program is reading large.txt
The child process is created
Range is 10480.0
MAD is 27.13
Execution time for Range and MAD algorithm is 0.078125 seconds.
```

```
nihal@LAPTOP-2QI8IJF8:~$ nano mad_thread.c
nihal@LAPTOP-2QI8IJF8:~$ gcc mad_thread.c -o mad_thread
nihal@LAPTOP-2QI8IJF8:~$ ./mad_thread tiny.txt 3
Program is reading tiny.txt
3 threads are created
MAD is 187.55
Range is 1023.3
Execution time for Range and MAD algorithm is 0.015625 seconds
nihal@LAPTOP-2QI8IJF8:~$ ./mad_thread large.txt 3
Program is reading large.txt
3 threads are created
MAD is 27.13
Range is 10480.0
Execution time for Range and MAD algorithm is 0.046875 seconds
```