Laborator 4

Numar procesoare calculator =6

factor scalare hyperthreading= 1.3

M=6\*1.3=7.8

|  |  |  |  |
| --- | --- | --- | --- |
| Processes | Erxecution time [miliseconds] | [Relative] Speedup | [Relative] Efficency |
| 1 | 15.501887 | 1 | 1 |
| 2 | 9.813445 | 1.57965801000566 | 0.202520257693033 |
| 3 | 7.701599 | 2.0128140922424 | 0.258053088749026 |
| 4 | 6.864879 | 2.25814424405732 | 0.289505672315042 |
| 5 | 4.659513 | 3.32693287903693 | 0.426529856286786 |
| 6 | 5.058457 | 3.06454853723181 | 0.392890838106642 |
| 7 | 4.9414 | 3.13714473630955 | 0.402198043116609 |
| 8 | 3.873778 | 4.00174893863303 | 0.513044735722184 |
| 9 | 5.895386 | 2.62949482866771 | 0.337114721624065 |
| 10 | 5.18599 | 2.98918567139543 | 0.383228932230184 |
| 11 | 3.371955 | 4.59729948946531 | 0.589397370444271 |
| 12 | 5.583537 | 2.77635609829397 | 0.355943089524867 |
| 13 | 4.037979 | 3.83902120342874 | 0.492182205567788 |
| 14 | 6.061474 | 2.55744510328676 | 0.327877577344456 |
| 15 | 3.828519 | 4.04905578371167 | 0.51910971586047 |
| 16 | 3.903042 | 3.97174485952239 | 0.509198058913127 |

Cod sursa:

#define \_GNU\_SOURCE

#define \_POSIX\_C\_SOURCE 1999309L

#include <stdio.h>

#include<stdlib.h>

#include <unistd.h>

#include<math.h>

#include<time.h>/\* for simplicity the no.of processesis defined here,But a better idea is to read it from the command line \*/

#define PROCESSES 4

int isPrime(int n){

if (n <= 1)

return 0;

if (n <= 3)

return 1;

// This is checked so that we can skip

// middle five numbers in below loop

if (n % 2 == 0 || n % 3 == 0)

return 0;

for (int i = 5; i \* i <= n; i++)

if (n % i == 0)

return 0;

return 1;

}

// Function to check if the number is circular

// prime or not.

int checkCircular(int N)

{

// Count digits.

int count = 0, temp = N;

while (temp) {

count++;

temp /= 10;

}

int num = N;

while (isPrime(num)) {

// Following three lines generate the next

// circular permutation of a number. We move

// last digit to first position.

int rem = num % 10;

int div = num / 10;

num = (pow(10, count - 1)) \* rem + div;

// If all the permutations are checked and

// we obtain original number exit from loop.

if (num == N)

return 1;

}

return 0;

}

void do\_work(int id,int start,int end,int nr\_processes)

{

//printf("Hello there, from process %d! search from %d to %d with step:%d\n", id,start,end,nr\_threads);

for(int i=start;i<=end;i+=2\*nr\_processes)

if(checkCircular(i))

{

printf("thread %d : %d ,",id,i);

}

}

struct timespec start, finish, delta;

double sub\_timespec(const struct timespec \*time1, const struct timespec \*time0) {

return (time1->tv\_sec - time0->tv\_sec)

+ (time1->tv\_nsec - time0->tv\_nsec) / 1000000000.0;

}

int main(){

int i, pid;

/\* Take initial time here \*//\* Use clock\_gettime(). Do NOT use clock() \*/

int B=200000;;

int A=11;

FILE\* f=fopen("times.csv","w+");

for(int p=1;p<=16;p++)

{

clock\_gettime(CLOCK\_REALTIME, &start);

for(i=0;i<p;i++){

pid = fork();

if(pid < 0)/\* some error occurred –fork failed \*/

{

printf("Error");

exit(-1)

;}

if(pid == 0) /\* child process code \*/

{

do\_work(i,A+2\*i,B,p);

exit(0)

;}/\* do not place anywait()call here\*/}/\* wait for all processes to finish their execution \*/

for(i=0; i<p;i++)

wait(NULL);

clock\_gettime(CLOCK\_REALTIME, &finish);

double diff=sub\_timespec(&finish, &start)\*1000;

printf(" \nfor p=%d time execution:%lf ms\n",p,diff);

if(f)

fprintf(f,"%d , %lf\n",p,diff);

}

fclose(f);

/\* Take final time here \*//\* Use clock\_gettime(). Do NOT use clock() \*//\* Compute the execution time\*/

}/\* this function is executed by each process \*