**ASSIGNMENT -2**

**3.** Write a suitable code to speedup finding all prime numbers in a given range [1, N], creating M child processes by the parent process.

First try to run it for two processes then generalize it for M child processes .

**CODE :**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <math.h>

#define MAX\_PROCESS 100

int is\_prime(int num)

{

if (num <= 1)

return 0;

for (int i = 2; i <= sqrt(num); i++)

{

if (num % i == 0)

return 0;

}

return 1;

}

void find\_primes(int start, int end)

{

for (int num = start; num <= end; num++)

{

if (is\_prime(num))

{

printf("%d ", num);

}

}

}

int main()

{

int N, M;

printf("Enter the value of N (upper limit): ");

scanf("%d", &N);

printf("Enter the number of child processes (M): ");

scanf("%d", &M);

M = (M > MAX\_PROCESS) ? MAX\_PROCESS : M; // Limit M to MAX\_PROCESS

int range\_size = N / M;

int remaining = N % M;

for (int i = 0; i < M; i++)

{

int start = i \* range\_size + 1;

int end = start + range\_size - 1 + (i == M - 1 ? remaining : 0);

pid\_t pid = fork();

if (pid == 0)

{

// Child process

printf("Child %d: Primes in range [%d, %d]: ", i + 1, start, end);

find\_primes(start, end);

printf("\n");

exit(0);

}

else if (pid < 0)

{

// Fork failed

perror("Fork failed");

exit(1);

}

}

for (int i = 0; i < M; i++)

{

wait(NULL);

}

return 0;

}