با نام خدا

گزارش کار ازمایشگاه سیستم عامل

گزارش شماره ۵

Multi process programming

امیر حسین متقیان ۴۰۱۳۱۰۴۳

کیان پور اذر ۴۰۱۳۱۴۰۳

در این ازمایش میخواهیم تعداد زیادی کار را با یک process و با چند process انجام دهیم و زمان انجام را به دست اورده و مقایسه کنیم

کد task مربوطه این است که ۱۲ عدد رندوم بین ۰ تا ۱۰۰ تولید کرده و چک میکنیم اگر بیشتر یا برابر ۴۹ بود یک متغیر counter با مقدار اولیه ۰ را یکی زیاد کرده و در غیر این صورت یکی کم میکنیم و در نهایت خانه در ارایه hist که index ان برابر count است را یکی زیاد میکنیم.

کد مربوط به این قسمت:

```
for (int i = 0; i < ITERATION_NUMBER; i++) {
  int count = 0;
  for (int i = 0; i < 12; ++i) {
    int random_number = rand() % 100;
    if (random_number < 49)
      count--;
    else
      count++;
  }
  hist[count + 12]++;
}
```

ازمایش دو قسمت دارد که در قسمت اول این کار را در تعداد های متفاوت با یک process و در قسمت دوم با چند process باید انجام دهیم و زمان انجام انها را مقایسه کنیم.

قسمت اول

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include <sys/time.h>

#define HIST_SIZE 25
#define ITERATION_NUMBER 500000

void printArray(int *hist) {
  printf("[");
  for (int i = 0; i < HIST_SIZE; i++) {
    printf(" %d:%d ", i -12, hist[i]);
    if (i  != 24)
```

```
        printf(",");

    }
    printf("]\n");
}

void printHistogram(int* hist) {
    for (int i = 0; i < HIST_SIZE; i++) {
        printf("%2d: ", i - 12);
        for (int j = 0; j < hist[i]; j++) {
            printf("*");
        }
        printf("\n");
    }
}

int main(void) {
    // initial
    struct timeval start, end;
    int hist[HIST_SIZE] = {0};

    // main logic
    srand(time(NULL));
    gettimeofday(&start, NULL);

    for (int i = 0; i < ITERATION_NUMBER; i++) {
        int count = 0;
        for (int i = 0; i < 12; ++i) {
            int random_number = rand() % 100;
            if (random_number < 49)
                count--;
            else
                count++;
        }
        hist[count + 12]++;
    }

    gettimeofday(&end, NULL);

    // print duration
    double duration = (end.tv_sec - start.tv_sec) * 1000.0 + (end.tv_usec - start.tv_usec) / 1000.0;
    printf("Operation Number: %d| Execution Time: %.6f ms \n", ITERATION_NUMBER, duration);

    // print hist arr
    printArray(hist);

    // print histogram
```

```
    printHistogram(hist);
    return 0;
}
```

دو متغیر گلوبال HIST_SIZE که نشان دهنده سایز ارایه hist و ITERATION_NUM که نشان دهنده تعداد انجام کار است را نشان میدهد. اگر کد را با ITERATION_NUM = ۵۰۰۰ اجرا کنیم در خروجی مشاهده میشود:

```
/home/amir/CLionProjects/untitled2/cmake-build-debug/untitled2
Operation Number: 5000| Execution Time: 1.722000 ms
[ -12:0 , -11:0 , -10:8 , -9:0 , -8:75 , -7:0 , -6:237 , -5:0 , -4:570 , -3:0 , -2:972 , -1:0 , 0:1133 , 1:0 , 2:971 , 3:0 , 4:625 , 5:0 , 6:279 , 7:0 , 8:112 , 9:0 , 10:17 , 11:0 , 12:1 ]
-12:
-11:
-10: ********
-9:
-8: ************************************************************************
-7:
-6: ********************************************************************************************************************************************************************************
-5:
-4: ********************************************************************************************************************************************************************************
-3:
-2: ********************************************************************************************************************************************************************************
-1:
 0: ********************************************************************************************************************************************************************************
 1:
 2: ********************************************************************************************************************************************************************************
 3:
 4: ********************************************************************************************************************************************************************************
 5:
 6: ********************************************************************************************************************************************************************************
 7:
 8: ****************************************************************************************************************
 9:
10: ****************
11:
12: *

Process finished with exit code 0
```

که همان طور که میبینیم در 1.7ms اجرا مشود

اگر کد را برای ITERATION_NUM = 50000 اجرا کنیم در خروجی داریم

```
/home/amir/CLionProjects/untitled2/cmake-build-debug/untitled2
Operation Number: 50000| Execution Time: 19.452000 ms
[ -12:13 , -11:0 , -10:126 , -9:0 , -8:728 , -7:0 , -6:2390 , -5:0 , -4:5635 , -3:0 , -2:9209 , -1:0 , 0:11273 , 1:0 , 2:9888 , 3:0 , 4:6537 , 5:0 , 6:3037 , 7:0 , 8:965 , 9:0 , 10:181 , 11:0 , 12:18 ]
-12: ************
-11:
-10: ****************************************************************************************************************
-9:
-8: ********************************************************************************************************************************************************************************
-7:
-6: ********************************************************************************************************************************************************************************
-5:
-4: ********************************************************************************************************************************************************************************
-3:
-2: ********************************************************************************************************************************************************************************
-1:
 0: ********************************************************************************************************************************************************************************
 1:
 2: ********************************************************************************************************************************************************************************
 3:
 4: ********************************************************************************************************************************************************************************
 5:
 6: ********************************************************************************************************************************************************************************
 7:
 8: ********************************************************************************************************************************************************************************
 9:
10: ****************************************************************************************************************************************************************************************
11:
12: ******************

Process finished with exit code 0
```

برای 50000 بار در 19.4ms اجرا مشیود همچنین برای 500000 در 123.1ms اجرا میشود

بنابر این داریم:

تعداد نمونه	۵۰۰۰	۵۰۰۰۰	۵۰۰۰۰۰
زمان اجرا	1.7ms	19.4ms	123.1ms

به طور تقریبی میتوان گفت با 10 برابر شدن سایز مسئله زمان اجرا ان نیز 10 برابر شده و به صورت خطی افزایش یافته است.

قسمت دوم

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/wait.h>
#include <stdatomic.h>

#define HIST_SIZE 25
#define NUM_PROCESSES 10
#define ITERATION_NUMBER 5000

void printArray(int *hist) {
   printf("[");
   for (int i = 0; i < 25; i++) {
      printf(" %d:%d ", i -12, hist[i]);
      if (i  != 24)
         printf(",");
   }
   printf("]\n");
}

void printHistogram(int* hist) {
   for (int i = 0; i < HIST_SIZE; i++) {
```

```
        printf("%2d: ", i - 12);
        for (int j = 0; j < hist[i]; j++) {
            printf("*");
        }
        printf("\n");
    }
}

void performTask(_Atomic int* shared_hist, int iterations) {
    for (int i = 0; i < iterations; i++) {
        int count = 0;
        for (int j = 0; j < 12; ++j) {
            int random_number = rand() % 100;
            if (random_number < 49)
                count--;
            else
                count++;
        }

        atomic_fetch_add(&shared_hist[count + 12], 1);
    }
}

int main(void) {
    // create shared memory ,make it atomic, and initialize to 0
    struct timeval start, end;
    int shm_id = shmget(IPC_PRIVATE, HIST_SIZE * sizeof(_Atomic int), IPC_CREAT | 0666);
    if (shm_id < 0) {
        perror("shmget failed");
        exit(EXIT_FAILURE);
    }

    _Atomic int* shared_hist = (_Atomic int*)shmat(shm_id, NULL, 0);
    if (shared_hist == (_Atomic int*)-1) {
        perror("shmat failed");
        exit(EXIT_FAILURE);
    }

    for (int i = 0; i < HIST_SIZE; i++) {
        shared_hist[i] = 0;
    }

    // start main process
    srand(time(NULL));
    gettimeofday(&start, NULL);

    for (int process_id = 0; process_id < NUM_PROCESSES; process_id++) {
```

```
    pid_t pid = fork();

    if (pid < 0) {
        perror("Fork failed");
        exit(EXIT_FAILURE);
    } else if (pid == 0) {
        performTask(shared_hist, ITERATION_NUMBER / NUM_PROCESSES);
        exit(EXIT_SUCCESS);
    }
  }

  // wait for finish child process
  for (int i = 0; i < NUM_PROCESSES; i++) {
    wait(NULL);
  }

  gettimeofday(&end, NULL);

  // print duration
  double duration = (end.tv_sec - start.tv_sec) * 1000.0 + (end.tv_usec - start.tv_usec) / 1000.0;
  printf("Operation Number: %d | Process Number: %d | Execution Time: %.6f ms \n", ITERATION_NUMBER,
NUM_PROCESSES, duration);

  // print hist arr
  printArray((int*)shared_hist);



  // print histogram
  printHistogram((int*)shared_hist);

  // free memory
  shmdt(shared_hist);
  shmctl(shm_id, IPC_RMID, NULL);

  return 0;
}
```

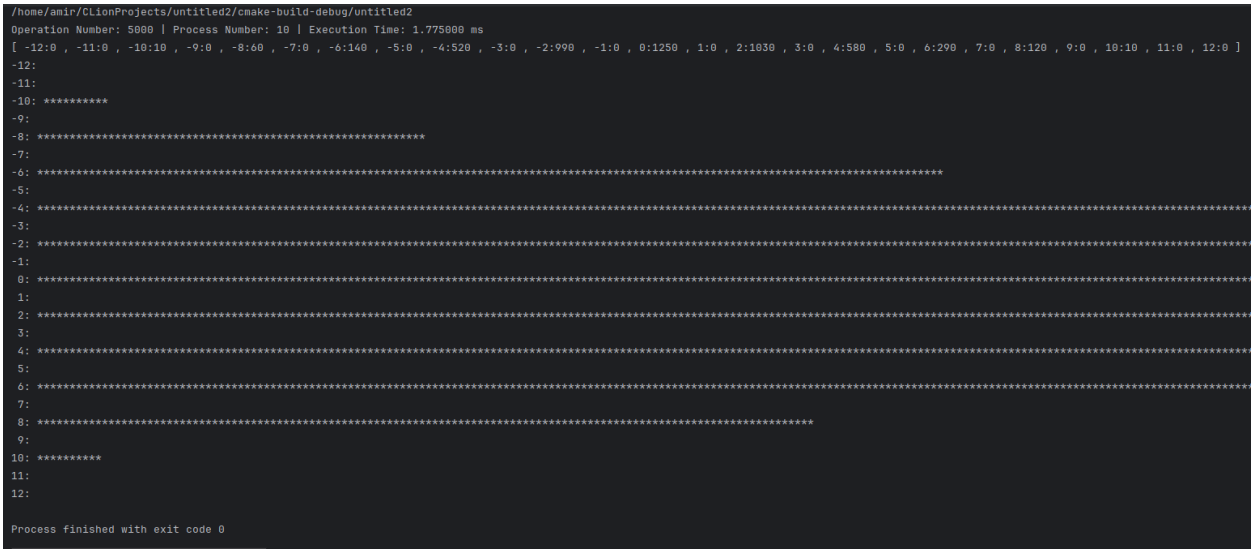
در قسمت دوم یک متغیر جدید به نام PROCESS_NUM اضافه شده که برنامه را به n فرایند جدا تقسیم کرده و سایز ورودی مسئله را به n مسئله کوچک تر میشکند.

برای انجام ان ارایه hist را در shared_memory تعریف کرده اما نکته جالب توجه این است که race condition پیش می اید چون که امکان دارد چندین فرایند به طور همزمان در خانه از ارایه شروع به نوشتن بکنند.

برای رفع این مشکل راه های مختلفی وجود دارد مثل mutex,semaphore,... در این مسئله از

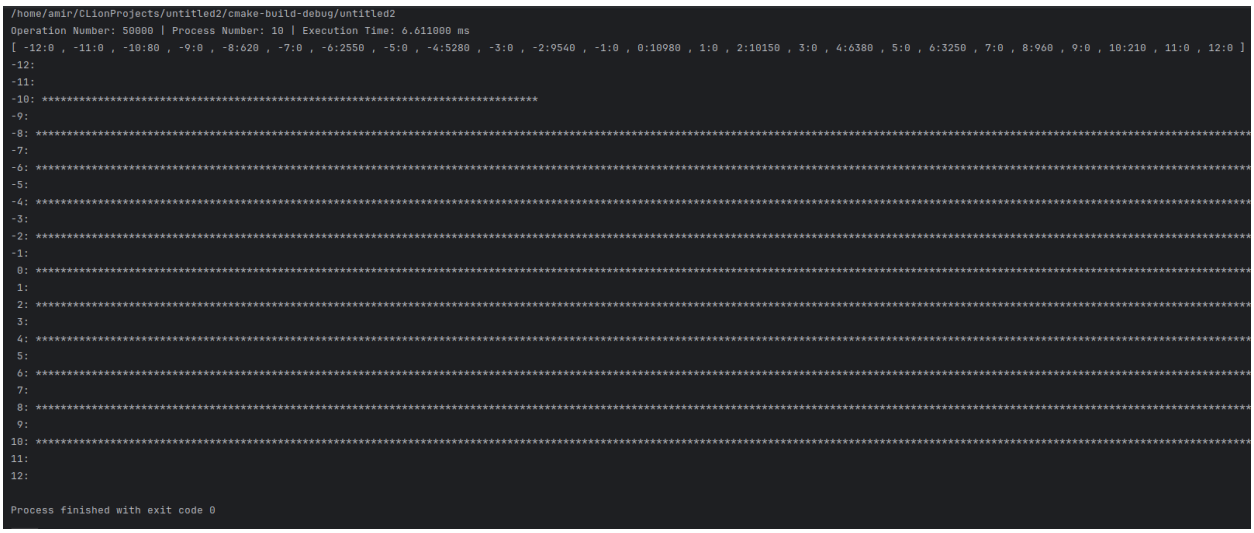
Atomic function ها استفاده کردم که از race condition جلوگیری میکند

اجرا با ITERATION_NUM = 5000 & PROCESS_NUM = 10



زمان اجرا برابر 1.7ms است

اجرا با ITERATION_NUM = 50000 & PROCESS_NUM = 10



زمان اجرا برابر 6.6ms است

همچنین اگر برای اجرا با ITERATION_NUM = 50000 & PROCESS_NUM = 10 اجرا کنیم زمانی برابر 43ms خواهد بود

بنابراین داریم:

تعداد نمونه	۵۰۰۰	۵۰۰۰۰	۵۰۰۰۰۰
زمان اجرا	1.7ms	6.6ms	43ms

که نشان دهنده بهبود بسیار زیاده در زمان خواهد بود

مقیاسه

تعداد نمونه	۵۰۰۰	۵۰۰۰۰	۵۰۰۰۰۰
افزایش سرعت	ناچیز 0.4ms	12.8ms	80.8ms

واضح است هرچه قدر سایز مسئله بزرگ تر میشود gap بیشتری بین زمان اجرا می افتد و در حال چند فرایندی بسیار سریع تر اجرا میشود.