با نام خدا

گزارش کار ازمایشگاه سیستم عامل

گزارش شماره 6

Multi thread programming

امیر حسین متقیان ۴۰۱۳۱۰۴۳

کیان پور اذر ۴۰۱۳۱۴۰۳

سوال یک)

```c
#include <stdio.h>
#include <stdlib.h>

#define SIZE 10

int* cal(int input[], int start,int end){
   int* count = (int*)malloc(sizeof(int)*end-start);
   count[start] = 1;

   for(int i=start+1;i<end;i++){
      count[i]=count[i-1]+input[i];
   }

   return count;
}

int main(){
   int input[SIZE]= {1,2,3,4,5,6,7,8,9,10};
   int* res = cal(input ,0,SIZE);

   for(int i=0;i<SIZE;i++){
      printf("%d ",res[i]);
   }
   printf("\n");
   return 0;
}
```

برای اجرا به صورت سریالی کافی است یک بار ارایه را پیمایش کرده و مقدار هر عنصر را با عنصر قبلی جمع کنیم و از پیچیدگی زمانی O(n) است

خروجی قسمت اول

```
/home/amir/CLionProjects/untitled2/cmake-build-debug/untitled2
1 3 6 10 15 21 28 36 45 55

Process finished with exit code 0
```

سوال دو)

میتوان ان را به صورت موازی انجام داد اما مشکلی که با ان مواجه هستم این است که:

۱ـاز کدام قسمت ارایه را بشکنیم و کار ها را تقسیم کنیم.

۲ـ این که تکه های i+1 به اخرین عدد (اخرین خانه) تکه i نیاز دارد تا جوابشان درست شود پس باید منتظر بماند که انها تمام شوند.

سوال سوم)

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

#define SIZE 10

typedef struct {
   int *input;
   int *output;
   int start;
   int end;
} SummationThreadArgs;



void* cal(void* args) {
   SummationThreadArgs* threadArgs = args;
   threadArgs->output = malloc(sizeof(int) * threadArgs->end -  threadArgs->start);

   threadArgs->output[0] = threadArgs->input[threadArgs->start];
   for (int i = threadArgs->start + 1; i < threadArgs->end; i++) {
      threadArgs->output[i - threadArgs->start] = threadArgs->output[i - threadArgs->start - 1] + threadArgs-
>input[i];
   }

  pthread_exit(NULL);
}

int main() {
   int input[SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
   int mid = SIZE / 2;

   SummationThreadArgs args1 = {input, NULL, 0, mid};
   SummationThreadArgs args2 = {input, NULL, mid, SIZE};

   pthread_t thread1, thread2;

   if (pthread_create(&thread1, NULL, cal, &args1) != 0) {
      perror("Failed to create thread1");
```

```
    return 1;
  }

  if (pthread_create(&thread2, NULL, cal, &args2) != 0) {
    perror("Failed to create thread2");
    return 1;
  }

  if (pthread_join(thread1, NULL) != 0) {
    perror("Failed to join thread1");
    return 1;
  }
  if (pthread_join(thread2, NULL) != 0) {
    perror("Failed to join thread2");
    return 1;
  }

  printf("Thread 1 results: ");
  for (int i = 0; i < mid; i++) {
    printf("%d ", args1.output[i]);
  }

  printf("\n");

  printf("Thread 2 results: ");
  for (int i = 0; i < SIZE - mid; i++) {
    printf("%d ", args2.output[i]);
  }

  return 0;
}
```

خروجی :

```
/home/amir/CLionProjects/untitled2/cmake-build-debug/untitled2
Thread 1 results: 1 3 6 10 15
Thread 2 results: 6 13 21 30 40
Process finished with exit code 0
```

تکه ابتدایی مشکلی ندارد اما تکه دوم مشکل دارد و عدد کمتری از چیز مورد نظر را نشان میدهد زیرا از خانه قبلی که در تکه اول است چیزی به ان اضافه نشده است.

سوال چهارم

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

#define SIZE 10

typedef struct {
   int *input;
   int *output;
   int start;
   int end;
   int offset;
} SummationThreadArgs;

void* cal(void* args) {
   SummationThreadArgs* threadArgs = (SummationThreadArgs*)args;

   threadArgs->output = malloc(sizeof(int) * (threadArgs->end - threadArgs->start));
   if (!threadArgs->output) {
      perror("malloc failed");
      pthread_exit(NULL);
   }

   threadArgs->output[0] = threadArgs->input[threadArgs->start] + threadArgs->offset;
   for (int i = threadArgs->start + 1; i < threadArgs->end; i++) {
      threadArgs->output[i - threadArgs->start] = threadArgs->output[i - threadArgs->start - 1] + threadArgs-
>input[i];
   }

   pthread_exit(NULL);
}

int main() {
   int input[SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
   int mid = SIZE / 2;

   SummationThreadArgs args1 = {input, NULL, 0, mid, 0};
   SummationThreadArgs args2 = {input, NULL, mid, SIZE, 0};
   pthread_t thread1, thread2;

   if (pthread_create(&thread1, NULL, cal, &args1) != 0) {
      perror("Failed to create thread1");
      return 1;
   }
   if (pthread_join(thread1, NULL) != 0) {
```

```
        perror("Failed to join thread1");
        return 1;
    }

    args2.offset = args1.output[mid - 1];

    if (pthread_create(&thread2, NULL, cal, &args2) != 0) {
        perror("Failed to create thread2");
        return 1;
    }
    if (pthread_join(thread2, NULL) != 0) {
        perror("Failed to join thread2");
        return 1;
    }

    printf("Thread 1 results: ");
    for (int i = 0; i < mid; i++) {
        printf("%d ", args1.output[i]);
    }
    printf("\n");

pv    printf("Thread 2 results: ");
    for (int i = 0; i < SIZE - mid; i++) {
        printf("%d ", args2.output[i]);
    }

    free(args1.output);
    free(args2.output);
    return 0;
}
```

خروجی :

```
/home/amir/CLionProjects/untitled2/cmake-build-debug/untitled2
Thread 1 results: 1 3 6 10 15
Thread 2 results: 21 28 36 45 55
Process finished with exit code 0
```

سوال پنجم:

ابتدا تکه کدی که زمانبندی را داشته باشد را داریم

اجرای سریالی

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



#define SIZE 5000



int* cal(int input[], int start,int end){
   int* count = (int*)malloc(sizeof(int)*end-start);
   count[start] = 1;

   for(int i=start+1;i<end;i++){
      count[i]=count[i-1]+input[i];
   }

   return count;
}

int main(){
   struct timeval start, end;
   srand(time(NULL));
   int input[SIZE];

   for (int i = 0; i < SIZE; i++){
      input[i] = rand() % 10;
   }



   gettimeofday(&start, NULL);

   int* res = cal(input ,0,SIZE);

   gettimeofday(&end, NULL);

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

   return 0;
}
```

اجرای پارالل

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

#define SIZE 10

typedef struct {
   int *input;
   int *output;
   int start;
   int end;
   int offset;
} SummationThreadArgs;

void* cal(void* args) {
   SummationThreadArgs* threadArgs = (SummationThreadArgs*)args;

   threadArgs->output = malloc(sizeof(int) * (threadArgs->end - threadArgs->start));
   if (!threadArgs->output) {
      perror("malloc failed");
      pthread_exit(NULL);
   }

   threadArgs->output[0] = threadArgs->input[threadArgs->start] + threadArgs->offset;
   for (int i = threadArgs->start + 1; i < threadArgs->end; i++) {
      threadArgs->output[i - threadArgs->start] = threadArgs->output[i - threadArgs->start - 1] + threadArgs-
>input[i];
   }

   pthread_exit(NULL);
}

int main() {
   srand(time(NULL));
   struct timeval start, end;
   int input[SIZE];
   int mid = SIZE / 2;

   for (int i = 0; i < SIZE; i++){
      input[i] = rand() % 10;
   }

   gettimeofday(&start, NULL);
```

```
  SummationThreadArgs args1 = {input, NULL, 0, mid, 0};
  SummationThreadArgs args2 = {input, NULL, mid, SIZE, 0};
  pthread_t thread1, thread2;

  if (pthread_create(&thread1, NULL, cal, &args1) != 0) {
     perror("Failed to create thread1");
     return 1;
  }
  if (pthread_join(thread1, NULL) != 0) {
     perror("Failed to join thread1");
     return 1;
  }

  args2.offset = args1.output[mid - 1];

  if (pthread_create(&thread2, NULL, cal, &args2) != 0) {
     perror("Failed to create thread2");
     return 1;
  }
  if (pthread_join(thread2, NULL) != 0) {
     perror("Failed to join thread2");
     return 1;
  }



  free(args1.output);
  free(args2.output);

  gettimeofday(&end, NULL);

  double duration = (end.tv_sec - start.tv_sec) * 1000.0 + (end.tv_usec - start.tv_usec) / 1000.0;
  printf("Execution Time: %.6f ms \n",duration);
  return 0;
}
```

500000	50000	5000	تعداد نمونه
2.188000ms	0.201000ms	0.032000ms	سریال
1.01000ms	0.123000ms	0.019000ms	دو نخه