ASSIGNMENT 4
12/09/23

NAME : SHRESTH SONKAR
REGNO : 20214272
GROUP : CS5D
TOPIC : OS LAB
CODE : CS-15203

```c
/*
 * Write a C program to calculate sum of array using
pthreads
 */

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

#define SIZE 1000
#define THRD 4

int arr[SIZE];
int sum = 0;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

void *calcSum(void *thrdID) {
    int tid = *((int *) thrdID);
    int ept = SIZE / THRD;
    int start = tid * ept;
    int end = (tid == THRD - 1) ? SIZE : (tid + 1) *
ept;

    int i, loc_sum = 0;

    for (i = start; i < end; i++) {
        loc_sum += arr[i];
    }

    pthread_mutex_lock(&mutex);
    sum += loc_sum;
    pthread_mutex_unlock(&mutex);
    pthread_exit(NULL);
}

int main() {
    int i;
    for (i = 0; i < SIZE; i++)
        arr[i] = i + 1;
    pthread_t threads[THRD];
    int thrdID[THRD];

    for (i = 0; i < THRD; i++) {
        thrdID[i] = i;
```

```
        pthread_create(&threads[i], NULL, calcSum,
&thrdID[i]);
    }
    for (i = 0; i < THRD; i++)
        pthread_join(threads[i], NULL);

    printf("Sum = %d\n", sum);
    return 0;
}
```

```c
/*
 * Write a C program to search element in an array
using pthreads
 */

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

#define SIZE 1000
#define THRD 4
#define KEY 42

int arr[SIZE];
int key = KEY;
int found = 0;
int ind = -1;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

void *searchEle(void *thrdID) {
    int tid = *((int *) thrdID);
    int ept = SIZE / THRD;
    int start = tid * ept;
    int end = (tid == THRD - 1) ? SIZE : (tid + 1) *
ept;
    int i;

    for (i = start; i < end; i++) {
        if (arr[i] == key) {
            pthread_mutex_lock(&mutex);
            found = 1;
            ind = i;
            pthread_mutex_unlock(&mutex);
            break;
        }
    }
    pthread_exit(NULL);
}

int main() {
    int i;
    for (i = 0; i < SIZE; i++)
        arr[i] = i + 1;
    pthread_t threads[THRD];
    int thrdID[THRD];
```

```
    for (i = 0; i < THRD; i++) {
        thrdID[i] = i;
        pthread_create(&threads[i], NULL, searchEle,
&thrdID[i]);
    }
    for (i = 0; i < THRD; i++)
        pthread_join(threads[i], NULL);

    if (found)
        printf("Element %d found at index : %d\n", key,
ind);
    else
        printf("Element not found!");
    return 0;
}
```

```c
/*
 * Write a C program to find max element in an array
using pthreads
 */

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

#define SIZE 1000
#define THRD 4

int arr[SIZE];
int max = INT_MIN;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

void *findMax(void *thrdID) {
    int tid = *((int *) thrdID);
    int ept = SIZE / THRD;
    int start = tid * ept;
    int end = (tid == THRD - 1) ? SIZE : (tid + 1) *
ept;

    int i, loc_max = INT_MIN;

    for (i = start; i < end; i++) {
        if (loc_max < arr[i])
            loc_max = arr[i];
    }

    pthread_mutex_lock(&mutex);
    if (loc_max > max)
        max = loc_max;
    pthread_mutex_unlock(&mutex);
    pthread_exit(NULL);
}

int main() {
    int i;
    for (i = 0; i < SIZE; i++)
        arr[i] = i + 1;
    pthread_t threads[THRD];
    int thrdID[THRD];
```

```c
    for (i = 0; i < THRD; i++) {
        thrdID[i] = i;
        pthread_create(&threads[i], NULL, findMax,
&thrdID[i]);
    }
    for (i = 0; i < THRD; i++)
        pthread_join(threads[i], NULL);

    printf("Max element in array = %d\n", max);
    return 0;
}
```

```c
/*
 * Write a C program to add and subtract matrix using
pthreads
 */

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

#define r 3
#define c 3
#define THRD 2

int A[r][c];
int B[r][c];
int add[r][c];
int sub[r][c];
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

void *matAdd(void *arg) {
    int tid = *((int *) arg);
    int ept = r / THRD;
    int start_row = tid * ept;
    int end_row = (tid == THRD - 1) ? r : (tid + 1) *
ept;
    int i, j;

    for (i = start_row; i < end_row; i++) {
        for (j = 0; j < c; j++) {
            add[i][j] = A[i][j] + B[i][j];
        }
    }
    pthread_exit(NULL);
}

void *matSub(void *arg) {
    int tid = *((int *) arg);
    int ept = r / THRD;
    int start_row = tid * ept;
    int end_row = (tid == THRD - 1) ? r : (tid + 1) *
ept;
    int i, j;

    for (i = start_row; i < end_row; i++) {
        for (j = 0; j < c; j++) {
```

```
            sub[i][j] = A[i][j] - B[i][j];
        }
    }
    pthread_exit(NULL);
}

int main() {
    int i, j;
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            A[i][j] = i + j;
            B[i][j] = i - j;
        }
    }
    pthread_t threads[THRD];
    int thrdID[THRD];

    for (i = 0; i < THRD; i++) {
        thrdID[i] = i;
        pthread_create(&threads[i], NULL, (i == 0) ?
matAdd : matSub, &thrdID[i]);
    }
    for (i = 0; i < THRD; i++)
        pthread_join(threads[i], NULL);

    printf("Matrix A :\n");
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    printf("Matrix B :\n");
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            printf("%d ", B[i][j]);
        }
        printf("\n");
    }

    printf("A+B :\n");
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            printf("%d ", add[i][j]);
```

```
        }
        printf("\n");
    }

    printf("A-B :\n");
    for (i = 0; i < r; i++) {
        for (j = 0; j < c; j++) {
            printf("%d ", sub[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

```
.../sem5/os/2023-09-12
 ➜ ~/desktop/cse/ASSGN/sem5/os/2023-09-12 $ clang q1.c -o q1
 ➜ ~/desktop/cse/ASSGN/sem5/os/2023-09-12 $ ./q1
Sum = 500500
 ➜ ~/desktop/cse/ASSGN/sem5/os/2023-09-12 $ clang q2.c -o q2
 ➜ ~/desktop/cse/ASSGN/sem5/os/2023-09-12 $ ./q2
Element 42 found at index : 41
 ➜ ~/desktop/cse/ASSGN/sem5/os/2023-09-12 $ clang q3.c -o q3
 ➜ ~/desktop/cse/ASSGN/sem5/os/2023-09-12 $ ./q3
Max element in array = 1000
 ➜ ~/desktop/cse/ASSGN/sem5/os/2023-09-12 $ clang q4.c -o q4
 ➜ ~/desktop/cse/ASSGN/sem5/os/2023-09-12 $ ./q4
Matrix A :
0 1 2
1 2 3
2 3 4
Matrix B :
0 -1 -2
1 0 -1
2 1 0
A+B :
0 0 0
0 0 0
0 0 0
A-B :
0 0 0
0 2 4
0 2 4
 ➜ ~/desktop/cse/ASSGN/sem5/os/2023-09-12 $
```