SYNCHRONIZATION PROBLEM

SPRING WORKER PROBLEM

ASIYA FATIMA TABASSUM (CS-22064)

SUBMITTED TO: DR. UROOJ AINUDDIN

BATCH:2022

1. PROBLEM STATEMENT:

Simulate a fruit-picking scenario where three picker threads collect fruits from a tree (represented as an array of integers) and place them in a shared crate with a capacity of 12. A loader thread waits for the crate to become full, then transfers its contents to a truck and resets the crate. Synchronization is required to ensure that:

- Fruits are picked only once.
- The crate is filled correctly.
- Mutual exclusion is enforced on shared resources.
- Partial crates at the end are also loaded.

2. OBJECTIVE:

The objective was to simulate a tree laden with fruits where three picker threads pick fruits and fill a shared crate of fixed capacity (12 slots). A loader thread is notified once the crate is full, who then loads it and resets the crate for reuse. Synchronization primitives ensure safe concurrent access.

3. TOOLS & CONCEPTS USED:

1. Programming Language: C
2. Concurrency: pthread, semaphore.h
3. Synchronization:
   - Counting Semaphores: availableSlots, readyToLoad
   - Binary Mutexes: fruitAccessMutex, crateAccessMutex

4. CODE:

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <semaphore.h>
#include <stdbool.h>

#define CRATE_CAPACITY 12
#define MAX_FRUITS 1000
#define NUM_PICKERS 3

int *fruitTree;               // Array representing fruits on the tree
int totalFruits;              // Total number of fruits
int nextFruitIndex = 0;       // Index of the next fruit to be picked
```

```c
typedef struct {
    int count;                // Number of fruits in crate
    int contents[CRATE_CAPACITY];  // Crate content
} FruitCrate;

FruitCrate currentCrate;
int crateCounter = 1;         // To count crates
bool allPicked = false;       // Signal that pickers are done

// Counting semaphores
sem_t availableSlots;         // To count empty slots in crate
sem_t readyToLoad;            // To notify loader when crate is full
```

```c
// Mutexes (mutual exclusion)
pthread_mutex_t fruitAccessMutex;  // Protects access to fruit array
pthread_mutex_t crateAccessMutex;  // Protects access to crate

void* fruitPicker(void* arg) {
    int pickerId = *((int*)arg);

    while (1) {
        // Lock fruit access
        pthread_mutex_lock(&fruitAccessMutex);
        if (nextFruitIndex >= totalFruits) {
            pthread_mutex_unlock(&fruitAccessMutex);
            break;
        }
        int fruit = fruitTree[nextFruitIndex++];
        pthread_mutex_unlock(&fruitAccessMutex);

        printf("Picker %d: grabbed fruit #%d from the tree.\n", pickerId, fruit);

        // Wait for space in the crate
        sem_wait(&availableSlots);
        pthread_mutex_lock(&crateAccessMutex);

        currentCrate.contents[currentCrate.count++] = fruit;
        printf("Picker %d: dropped fruit #%d into the crate (%d of %d filled).\n",
               pickerId, fruit, currentCrate.count, CRATE_CAPACITY);

        if (currentCrate.count == CRATE_CAPACITY) {
            sem_post(&readyToLoad);
        }
```

```c

        pthread_mutex_unlock(&crateAccessMutex);
        usleep(100000); // Simulate delay
    }

    free(arg);
    return NULL;
}
```

```
68  void* crateLoader(void* arg) {
69      while (1) {
70          sem_wait(&readyToLoad);
71          pthread_mutex_lock(&crateAccessMutex);
72
73          if (allPicked && currentCrate.count == 0) {
74              pthread_mutex_unlock(&crateAccessMutex);
75              break;
76          }
77
78          printf("Loader: Crate #%d loaded to truck contain [%d fruits]: [", crateCounter++, currentCrate.count);
79          for (int i = 0; i < currentCrate.count; i++) {
80              printf("%d%s", currentCrate.contents[i], (i < currentCrate.count - 1) ? ", " : "]\n");
81          }
82
83          currentCrate.count = 0;
84          pthread_mutex_unlock(&crateAccessMutex);
85
86          // Reset crate for pickers
87          for (int i = 0; i < CRATE_CAPACITY; i++) {
88              sem_post(&availableSlots);
89          }
90
91          if (allPicked && nextFruitIndex >= totalFruits) {
92              break;
93          }
94      }
95
96      printf("Loader: All crates loaded to truck!.\n");
97      return NULL;
98  }
```

```
100 // Initialization of semaphores
101 void initialize_semaphores() {
102     sem_init(&availableSlots, 0, CRATE_CAPACITY);
103     sem_init(&readyToLoad, 0, 0);
104     pthread_mutex_init(&fruitAccessMutex, NULL);
105     pthread_mutex_init(&crateAccessMutex, NULL);
106 }
107
108 int main() {
109     printf("Enter number of fruits on tree: ");
110     scanf("%d", &totalFruits);
111
112     if (totalFruits <= 0 || totalFruits > MAX_FRUITS) {
113         printf("Invalid input. Please enter between 1 and %d fruits.\n", MAX_FRUITS);
114         return 1;
115     }
116
117     fruitTree = malloc(sizeof(int) * totalFruits);
118     for (int i = 0; i < totalFruits; i++) {
119         fruitTree[i] = i + 1;
120     }
121
122     pthread_t pickers[NUM_PICKERS], loaderThread;
123     currentCrate.count = 0;
124
```

```
124
125     initialize_semaphores();
126
127     pthread_create(&loaderThread, NULL, crateLoader, NULL);
128
129     for (int i = 0; i < NUM_PICKERS; i++) {
130         int* id = malloc(sizeof(int));
131         *id = i + 1;
132         pthread_create(&pickers[i], NULL, fruitPicker, id);
133     }
134
```

```
134
135     for (int i = 0; i < NUM_PICKERS; i++) {
136         pthread_join(pickers[i], NULL);
137     }
138
139     allPicked = true;
140     sem_post(&readyToLoad);
141
142     pthread_join(loaderThread, NULL);
143     printf("WORK DONE! All fruit has been picked and loaded for transportation.\n");
144
145
146     return 0;
147 }
148
```

5. EXPLANATION:

Global Variables

Global variables like fruitTree, totalFruits, nextFruitIndex, currentCrate, and semaphores/mutexes serve as shared data accessible by all threads. They enable communication and synchronization among pickers and the loader, ensuring thread-safe and orderly execution of tasks.

Main Function

The main() function serves as the control center of the program. It begins by taking input from the user to determine how many fruits are present on the tree. It allocates memory for the fruit array, initializes the shared crate, and starts all picker and loader threads. Once all pickers complete their job, it signals the loader to finish any remaining work and waits for all threads to join before terminating.

Initialization

The initialize_semaphores() function sets up the synchronization tools used in the program. It initializes two semaphores—availableSlots (to track empty spaces in the crate) and readyToLoad (to notify the loader when the crate is full). Additionally, two mutexes (fruitAccessMutex and crateAccessMutex) are initialized to protect shared resources like the fruit array and crate from race conditions.

Picker Function

The picker function (fruitPicker) represents workers picking fruits from a tree. Each picker grabs the next available fruit, ensuring no two threads pick the same fruit by locking the shared index with a mutex. After picking a fruit, the picker waits for an empty slot in the crate, places the fruit inside, and if the crate becomes full, signals the loader. The picker then sleeps briefly to simulate real-life delay and repeats the process until all fruits are picked.

Loader Function

The loader function (crateLoader) waits to be signaled when the crate is full. Upon receiving the signal, it locks the crate, prints its contents, resets it to empty, and releases the crate back to pickers by resetting the availableSlots. At the end, the loader also checks if a partially filled crate needs to be processed before exiting.

6. CONCLUSION:

The "Spring Workers" synchronization problem was successfully implemented using multithreading in C, semaphores and mutexes to simulate a realistic fruit-picking and loading process. The final solution ensures thread safety, proper resource sharing, and correct sequencing of operations between multiple picker threads and a single loader thread.

Each picker independently picks fruits from a shared tree and places them into a shared crate with a fixed capacity. When the crate becomes full, it is handed over to the loader, who logs and clears the crate before returning it for reuse. All synchronization constraints were met, including:

- Mutual exclusion on shared resources (fruit tree and crate),
- Proper signaling when the crate is full or partially filled at the end,
- Guaranteed progress and termination of all threads once the tree is bare.

This project not only demonstrates the understanding of classical synchronization concepts but also highlights practical implementation skills involving concurrent programming, resource sharing, and inter-thread communication. The problem closely models real-world scenarios where multiple agents must collaborate on a limited resource under concurrency constraints.

7. OUTPUTS:

TEST CASE #	*INPUT FRUITS*	*EXPECTED OUTPUT AND SUMMARY*
1.	**0**	**INVALID INPUT**
2.	**24**	**2 FULL CRATES**
3.	**35**	**2 FULL AND 1 PARTIALLY FILLED CRATES**

TEST CASE #1

```
hp@DESKTOP-4PPU8MQ UCRT64 /c/Users/hp/Desktop/SYNCHRONIZATION PROBLEM
$ ./SpringWorkerProblem.exe
Enter number of fruits on tree: 0
Invalid input. Please enter between 1 and 1000 fruits.
```

TEST CASE # 2

```
hp@DESKTOP-4PPU8MQ UCRT64 /c/Users/hp/Desktop/SYNCHRONIZATION PROBLEM
$ ./SpringWorkerProblem.exe
Enter number of fruits on tree: 24
Picker 2: grabbed fruit #1 from the tree.
Picker 2: dropped fruit #1 into the crate (1 of 12 filled).
Picker 3: grabbed fruit #3 from the tree.
Picker 3: dropped fruit #3 into the crate (2 of 12 filled).
Picker 1: grabbed fruit #2 from the tree.
Picker 1: dropped fruit #2 into the crate (3 of 12 filled).
Picker 1: grabbed fruit #4 from the tree.
Picker 1: dropped fruit #4 into the crate (4 of 12 filled).
Picker 3: grabbed fruit #6 from the tree.
Picker 3: dropped fruit #6 into the crate (5 of 12 filled).
Picker 2: grabbed fruit #5 from the tree.
Picker 2: dropped fruit #5 into the crate (6 of 12 filled).
Picker 1: grabbed fruit #7 from the tree.
Picker 1: dropped fruit #7 into the crate (7 of 12 filled).
Picker 2: grabbed fruit #9 from the tree.
Picker 2: dropped fruit #9 into the crate (8 of 12 filled).
Picker 3: grabbed fruit #8 from the tree.
Picker 3: dropped fruit #8 into the crate (9 of 12 filled).
Picker 1: grabbed fruit #10 from the tree.
Picker 1: dropped fruit #10 into the crate (10 of 12 filled).
Picker 3: grabbed fruit #12 from the tree.
Picker 3: dropped fruit #12 into the crate (11 of 12 filled).
Picker 2: grabbed fruit #11 from the tree.
Picker 2: dropped fruit #11 into the crate (12 of 12 filled).
Loader: Crate #1 loaded to truck contain [12 fruits]: [1, 3, 2, 4, 6, 5, 7, 9, 8, 10, 12, 11]
Picker 2: grabbed fruit #13 from the tree.
Picker 2: dropped fruit #13 into the crate (1 of 12 filled).
Picker 3: grabbed fruit #15 from the tree.
Picker 3: dropped fruit #15 into the crate (2 of 12 filled).
Picker 1: grabbed fruit #14 from the tree.
Picker 1: dropped fruit #14 into the crate (3 of 12 filled).
Picker 3: grabbed fruit #16 from the tree.
Picker 3: dropped fruit #16 into the crate (4 of 12 filled).
Picker 1: grabbed fruit #18 from the tree.
Picker 1: dropped fruit #18 into the crate (5 of 12 filled).
Picker 2: grabbed fruit #17 from the tree.
Picker 2: dropped fruit #17 into the crate (6 of 12 filled).
Picker 3: grabbed fruit #19 from the tree.
Picker 3: dropped fruit #19 into the crate (7 of 12 filled).
Picker 1: grabbed fruit #21 from the tree.
Picker 1: dropped fruit #21 into the crate (8 of 12 filled).
Picker 2: grabbed fruit #20 from the tree.
Picker 2: dropped fruit #20 into the crate (9 of 12 filled).
Picker 3: grabbed fruit #22 from the tree.
Picker 3: dropped fruit #22 into the crate (10 of 12 filled).
Picker 2: grabbed fruit #24 from the tree.
Picker 2: dropped fruit #24 into the crate (11 of 12 filled).
Picker 1: grabbed fruit #23 from the tree.
Picker 1: dropped fruit #23 into the crate (12 of 12 filled).
Loader: Crate #2 loaded to truck contain [12 fruits]: [13, 15, 14, 16, 18, 17, 19, 21, 20, 22, 24, 2
3]
Loader: All crates loaded to truck!.
WORK DONE! All fruit has been picked and loaded for transportation.
```

TEST CASE #3

```
hp@DESKTOP-4PPU8MQ UCRT64 /c/Users/hp/Desktop/SYNCHRONIZATION PROBLEM
$ ./SpringWorkerProblem.exe
Enter number of fruits on tree: 35
Picker 1: grabbed fruit #1 from the tree.
Picker 1: dropped fruit #1 into the crate (1 of 12 filled).
Picker 2: grabbed fruit #2 from the tree.
Picker 2: dropped fruit #2 into the crate (2 of 12 filled).
Picker 3: grabbed fruit #3 from the tree.
Picker 3: dropped fruit #3 into the crate (3 of 12 filled).
Picker 2: grabbed fruit #4 from the tree.
Picker 2: dropped fruit #4 into the crate (4 of 12 filled).
Picker 1: grabbed fruit #6 from the tree.
Picker 1: dropped fruit #6 into the crate (5 of 12 filled).
Picker 3: grabbed fruit #5 from the tree.
Picker 3: dropped fruit #5 into the crate (6 of 12 filled).
Picker 1: grabbed fruit #7 from the tree.
Picker 1: dropped fruit #7 into the crate (7 of 12 filled).
Picker 3: grabbed fruit #9 from the tree.
Picker 3: dropped fruit #9 into the crate (8 of 12 filled).
Picker 2: grabbed fruit #8 from the tree.
Picker 2: dropped fruit #8 into the crate (9 of 12 filled).
Picker 1: grabbed fruit #10 from the tree.
Picker 1: dropped fruit #10 into the crate (10 of 12 filled).
Picker 3: grabbed fruit #11 from the tree.
Picker 3: dropped fruit #11 into the crate (11 of 12 filled).
Picker 2: grabbed fruit #12 from the tree.
Picker 2: dropped fruit #12 into the crate (12 of 12 filled).
Loader: Crate #1 loaded to truck contain [12 fruits]: [1, 2, 3, 4, 6, 5, 7, 9, 8, 10, 11, 12]
Picker 1: grabbed fruit #13 from the tree.
Picker 1: dropped fruit #13 into the crate (1 of 12 filled).
Picker 3: grabbed fruit #14 from the tree.
Picker 3: dropped fruit #14 into the crate (2 of 12 filled).
Picker 2: grabbed fruit #15 from the tree.
Picker 2: dropped fruit #15 into the crate (3 of 12 filled).
Picker 3: grabbed fruit #16 from the tree.
Picker 3: dropped fruit #16 into the crate (4 of 12 filled).
Picker 1: grabbed fruit #17 from the tree.
Picker 1: dropped fruit #17 into the crate (5 of 12 filled).
Picker 2: grabbed fruit #18 from the tree.
Picker 2: dropped fruit #18 into the crate (6 of 12 filled).
Picker 1: grabbed fruit #19 from the tree.
Picker 1: dropped fruit #19 into the crate (7 of 12 filled).
Picker 3: grabbed fruit #20 from the tree.
Picker 3: dropped fruit #20 into the crate (8 of 12 filled).
Picker 2: grabbed fruit #21 from the tree.
Picker 2: dropped fruit #21 into the crate (9 of 12 filled).
Picker 1: grabbed fruit #22 from the tree.
Picker 1: dropped fruit #22 into the crate (10 of 12 filled).
Picker 3: grabbed fruit #23 from the tree.
Picker 3: dropped fruit #23 into the crate (11 of 12 filled).
Picker 2: grabbed fruit #24 from the tree.
Picker 2: dropped fruit #24 into the crate (12 of 12 filled).
Loader: Crate #2 loaded to truck contain [12 fruits]: [13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
```

```
Picker 1: grabbed fruit #25 from the tree.
Picker 1: dropped fruit #25 into the crate (1 of 12 filled).
Picker 3: grabbed fruit #26 from the tree.
Picker 3: dropped fruit #26 into the crate (2 of 12 filled).
Picker 2: grabbed fruit #27 from the tree.
Picker 2: dropped fruit #27 into the crate (3 of 12 filled).
Picker 1: grabbed fruit #28 from the tree.
Picker 1: dropped fruit #28 into the crate (4 of 12 filled).
Picker 3: grabbed fruit #29 from the tree.
Picker 3: dropped fruit #29 into the crate (5 of 12 filled).
Picker 2: grabbed fruit #30 from the tree.
Picker 2: dropped fruit #30 into the crate (6 of 12 filled).
Picker 2: grabbed fruit #31 from the tree.
Picker 2: dropped fruit #31 into the crate (7 of 12 filled).
Picker 3: grabbed fruit #33 from the tree.
Picker 3: dropped fruit #33 into the crate (8 of 12 filled).
Picker 1: grabbed fruit #32 from the tree.
Picker 1: dropped fruit #32 into the crate (9 of 12 filled).
Picker 2: grabbed fruit #34 from the tree.
Picker 2: dropped fruit #34 into the crate (10 of 12 filled).
Picker 3: grabbed fruit #35 from the tree.
Picker 3: dropped fruit #35 into the crate (11 of 12 filled).
Loader: Crate #3 loaded to truck contain [11 fruits]: [25, 26, 27, 28, 29, 30, 31, 33, 32, 34, 35]
Loader: All crates loaded to truck!.
WORK DONE! All fruit has been picked and loaded for transportation.
```