

Slot: L3+L4
Date: 02/11/2020

Name: Swaranjana Nayak
Reg. No.: 19BCE0977

LAB FAT

Problem 1 CIGARETTE SMOKER'S PROBLEM

Aim:

To implement Cigarette Smoker's problem using Semaphore

Algorithm:

There are 4 processes - agent, smoker1, smoker2, smoker3.

Note: This program uses UNIX STANDARD library, POSIX THREAD library and SEMAPHORE library. The program has been executed on VMware in the terminal as shown in screenshots.

- Let there be following semaphores
 - Agent_ready: an agent semaphore that represents items on the table.
 - Smoker_semaphores: each smoker semaphore represents when a smoker has the items they need
 - Pusher_semaphores: while pushing
 - Pusher_lock: exclusive access to the items on table
- Let there be following boolean/flag/information variables
 - Smoker_types: strings describing what each smoker type needs.
 - Items_on_table: this list represents item types that are on the table. Respectively for smoker_types. Each item is the one the smoker has. Paper, then tobacco, then matches.
- The smoker function handles waiting for the items that they need and then smokes. Since its given the process goes on forever, just for example, here its done 3 times. It waits on semaphore Smoker_semaphore[No] for his ingredients, notifies the agent that the ingredients have been taken and the table is free once the needed ingredients are available, then executes for loop to simulate smoking for a while, and then goes back for the next round.
- Pusher function handles releasing of proper smoker's semaphores when the right items are on the table.
- The agent function handles putting items on the table. It waits on the agent_ready semaphore
- The main handles agent's arbitration of putting items on the table.

Program:

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

sem_t agent_ready;

sem_t smoker_semaphors[3];

char* smoker_types[3] = { "matches & tobacco", "matches & paper", "tobacco & paper" };

bool items_on_table[3] = { false, false, false };

sem_t pusher_semaphores[3];

sem_t pusher_lock;

void* smoker(void* arg)
{
    int smoker_id = *(int*) arg;
```

```

int type_id = smoker_id % 3;

// Smoke 3 times
for (int i = 0; i < 3; ++i)
{
    printf("\033[0;37mSmoker %d \033[0;31m>>\033[0m Waiting for %s\n", smoker_id, smoker_types[type_id]);

    // Wait for the proper combination of items to be on the table
    sem_wait(&smoker_semaphors[type_id]);

    // Make the cigarette before releasing the agent
    printf("\033[0;37mSmoker %d \033[0;32m<<\033[0m Now making the a cigarette\n", smoker_id);
    usleep(rand() % 50000);
    sem_post(&agent_ready);

    // Smoking now
    printf("\033[0;37mSmoker %d \033[0;37m--\033[0m Now smoking\n", smoker_id);
    usleep(rand() % 50000);
}

return NULL;
}

void* pusher(void* arg)
{
    int pusher_id = *(int*) arg;

    for (int i = 0; i < 12; ++i)
    {
        // Wait for this pusher to be needed
        sem_wait(&pusher_semaphores[pusher_id]);
        sem_wait(&pusher_lock);

        // Check if the other item we need is on the table
        if (items_on_table[(pusher_id + 1) % 3])
        {
            items_on_table[(pusher_id + 1) % 3] = false;
            sem_post(&smoker_semaphors[(pusher_id + 2) % 3]);
        }
        else if (items_on_table[(pusher_id + 2) % 3])
        {
            items_on_table[(pusher_id + 2) % 3] = false;
            sem_post(&smoker_semaphors[(pusher_id + 1) % 3]);
        }
        else
        {
            // The other item's aren't on the table yet
            items_on_table[pusher_id] = true;
        }

        sem_post(&pusher_lock);
    }

    return NULL;
}

void* agent(void* arg)
{
    int agent_id = *(int*) arg;

    for (int i = 0; i < 6; ++i)
    {
        usleep(rand() % 200000);

        // Wait for a lock on the agent
        sem_wait(&agent_ready);

        // Items this agent gives out
        sem_post(&pusher_semaphores[agent_id]);
        sem_post(&pusher_semaphores[(agent_id + 1) % 3]);

        // Print type of items we put on the table
        printf("\033[0;35m==> \033[0;33mAgent %d giving out %s\033[0;0m\n",
            agent_id, smoker_types[(agent_id + 2) % 3]);
    }

    return NULL;
}

```

```

}

int main(int argc, char* argv[])
{
    srand(time(NULL));

    // A values of 1 = nothing on the table
    sem_init(&agent_ready, 0, 1);

    // Initialize the pusher lock semaphore
    sem_init(&pusher_lock, 0, 1);

    // Initialize the semaphores for the smokers and pusher
    for (int i = 0; i < 3; ++i)
    {
        sem_init(&smoker_semaphors[i], 0, 0);
        sem_init(&pusher_semaphores[i], 0, 0);
    }

    int smoker_ids[6];

    pthread_t smoker_threads[6];

    // Create the 6 smoker threads with IDs
    for (int i = 0; i < 6; ++i)
    {
        smoker_ids[i] = i;

        if (pthread_create(&smoker_threads[i], NULL, smoker, &smoker_ids[i]) == EAGAIN)
        {
            perror("Insufficient resources to create thread");
            return 0;
        }
    }

    int pusher_ids[6];

    pthread_t pusher_threads[6];

    for (int i = 0; i < 3; ++i)
    {
        pusher_ids[i] = i;

        if (pthread_create(&pusher_threads[i], NULL, pusher, &pusher_ids[i]) == EAGAIN)
        {
            perror("Insufficient resources to create thread");
            return 0;
        }
    }

    int agent_ids[6];

    pthread_t agent_threads[6];

    for (int i = 0; i < 3; ++i)
    {
        agent_ids[i] = i;

        if (pthread_create(&agent_threads[i], NULL, agent, &agent_ids[i]) == EAGAIN)
        {
            perror("Insufficient resources to create thread");
            return 0;
        }
    }

    // Make sure all the smokers are done smoking
    for (int i = 0; i < 6; ++i)
    {
        pthread_join(smoker_threads[i], NULL);
    }

    return 0;
}

```

Output:

```
Ubuntu 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
Activities Terminal
Nov 2 10:45
swaranjana@swaranjana-virtual-machine: ~/Documents/19BCE0977 Lab FAT
(base) swaranjana@swaranjana-virtual-machine:~/Documents/19BCE0977 Lab FAT$ gcc 19BCE0977_q1.c -lpthread
(base) swaranjana@swaranjana-virtual-machine:~/Documents/19BCE0977 Lab FAT$ ./a.out
Snoker 5 >>> Waiting for tobacco & paper
Snoker 4 >>> Waiting for matches & tobacco
Snoker 3 >>> Waiting for matches & paper
Snoker 2 >>> Waiting for tobacco & paper
Snoker 1 >>> Waiting for matches & paper
Snoker 0 >>> Waiting for matches & tobacco
==> Agent 1 giving out matches & tobacco
Snoker 3 <<< Now making the a cigarette
Snoker 3 -- Now smoking
==> Agent 2 giving out matches & paper
Snoker 4 <<< Now making the a cigarette
Snoker 4 -- Now making the a cigarette
Snoker 3 >>> Waiting for matches & tobacco
Snoker 4 >>> Now smoking
==> Agent 0 giving out tobacco & paper
Snoker 5 <<< Now making the a cigarette
Snoker 5 -- Now smoking
Snoker 5 >>> Waiting for tobacco & paper
Snoker 4 >>> Waiting for matches & paper
==> Agent 0 giving out tobacco & paper
Snoker 2 <<< Now making the a cigarette
Snoker 2 -- Now smoking
==> Agent 1 giving out matches & tobacco
Snoker 0 <<< Now making the a cigarette
Snoker 0 -- Now smoking
==> Agent 2 giving out matches & paper
Snoker 1 <<< Now making the a cigarette
Snoker 0 >>> Waiting for matches & tobacco
Snoker 2 >>> Waiting for tobacco & paper
Snoker 1 >>> Now smoking
==> Agent 1 giving out matches & tobacco
Snoker 3 <<< Now making the a cigarette
Snoker 3 -- Now smoking
Snoker 3 >>> Waiting for matches & tobacco
==> Agent 2 giving out matches & paper
Snoker 4 <<< Now making the a cigarette
Snoker 4 <<< Now making the a cigarette
Snoker 1 >>> Waiting for matches & paper
Snoker 4 >>> Now smoking
==> Agent 1 giving out matches & tobacco
Snoker 0 <<< Now making the a cigarette
Snoker 4 >>> Waiting for matches & paper
Snoker 0 >>> Now smoking
==> Agent 0 giving out tobacco & paper
Snoker 5 <<< Now making the a cigarette
Snoker 0 >>> Waiting for matches & tobacco
Snoker 5 >>> Now smoking
==> Agent 1 giving out matches & tobacco
Snoker 3 <<< Now making the a cigarette
Snoker 3 <<< Now making the a cigarette
```

```
Ubuntu 64-bit - VMware Workstation 15 Player (Non-commercial use only)
Player
Activities Terminal
Nov 2 10:45
swaranjana@swaranjana-virtual-machine: ~/Documents/19BCE0977 Lab FAT
Snoker 1 <<< Now making the a cigarette
Snoker 0 >>> Waiting for matches & tobacco
Snoker 2 >>> Waiting for tobacco & paper
Snoker 1 >>> Now smoking
==> Agent 1 giving out matches & tobacco
Snoker 3 <<< Now making the a cigarette
Snoker 3 -- Now smoking
Snoker 3 >>> Waiting for matches & tobacco
==> Agent 2 giving out matches & paper
Snoker 4 <<< Now making the a cigarette
Snoker 1 >>> Waiting for matches & paper
Snoker 4 >>> Now smoking
==> Agent 1 giving out matches & tobacco
Snoker 0 <<< Now making the a cigarette
Snoker 4 >>> Waiting for matches & paper
Snoker 0 >>> Now smoking
==> Agent 0 giving out tobacco & paper
Snoker 5 <<< Now making the a cigarette
Snoker 0 >>> Waiting for matches & tobacco
Snoker 5 >>> Now smoking
==> Agent 1 giving out matches & tobacco
Snoker 3 <<< Now making the a cigarette
Snoker 5 >>> Waiting for tobacco & paper
Snoker 3 >>> Now smoking
==> Agent 1 giving out matches & tobacco
Snoker 0 <<< Now making the a cigarette
Snoker 0 >>> Now smoking
==> Agent 2 giving out matches & paper
Snoker 1 <<< Now making the a cigarette
Snoker 1 >>> Now smoking
==> Agent 0 giving out tobacco & paper
Snoker 2 <<< Now making the a cigarette
Snoker 1 >>> Waiting for matches & paper
Snoker 2 >>> Now smoking
==> Agent 0 giving out tobacco & paper
Snoker 5 <<< Now making the a cigarette
Snoker 5 >>> Now smoking
Snoker 2 >>> Waiting for tobacco & paper
==> Agent 2 giving out matches & paper
Snoker 4 <<< Now making the a cigarette
Snoker 4 >>> Now smoking
==> Agent 2 giving out matches & paper
Snoker 1 <<< Now making the a cigarette
Snoker 1 >>> Now smoking
==> Agent 0 giving out tobacco & paper
Snoker 2 <<< Now making the a cigarette
Snoker 2 >>> Now smoking
(base) swaranjana@swaranjana-virtual-machine:~/Documents/19BCE0977 Lab FAT$
```

Problem 2

CSCAN DISK SCHEDULING ALGORITHM

Aim:

To write and execute a C program to implement CSCAN disk scheduling, and test it on given inputs.

Algorithm:

1. Let 'q_size' be number of disk positions to be read.
2. Let 'head' be position of the disk head.
3. Let queue1 array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival, which come after the initial head position.
4. Let queue2 array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival, which come before the initial head position.
5. Because size of disk movement wasn't given, the head services only in the left direction from size of the disk to 0.
6. While moving in the right direction do not service any of the tracks.
7. When we reach at the end(right end), reverse the direction.
8. While moving in left direction it services all tracks one by one.
9. While moving in left direction calculate the absolute distance of the track from the head.
10. Increment the total head movement count ('seek') with this distance.
11. Currently serviced track position now becomes the new head position.
12. Go to step 6 until we reach at left end of the disk.
13. If we reach at the left end of the disk reverse the direction and go to step 3 until all tracks in request array have not been serviced.

Program:

```
#include <stdlib.h>
#include <stdio.h>
#define UP 4999
#define DOWN 0

int main()
{
    int queue[20], q_size, head, i, j, seek = 0, diff, max, temp, queue1[20], queue2[20], temp1 = 0, temp2 = 0;

    printf("Input no of disk locations: ");
    scanf("%d", &q_size);

    printf("Enter initial head position: ");
    scanf("%d", &head);

    printf("Enter disk positions to be read: ");
    for (i = 0; i < q_size; i++)
    {
        scanf("%d", &temp);
        if (temp >= head)
        {
            queue1[temp1] = temp;
            temp1++;
        }
        else
        {
            queue2[temp2] = temp;
            temp2++;
        }
    }
}
```

```

    }
}
//sort both arrays
for (i = 0; i < temp1 - 1; i++)
{
    for (j = i + 1; j < temp1; j++)
    {
        if (queue1[i] > queue1[j])
        {
            temp = queue1[i];
            queue1[i] = queue1[j];
            queue1[j] = temp;
        }
    }
}

for (i = 0; i < temp2 - 1; i++)
{
    for (j = i + 1; j < temp2; j++)
    {
        if (queue2[i] > queue2[j])
        {
            temp = queue2[i];
            queue2[i] = queue2[j];
            queue2[j] = temp;
        }
    }
}

//calculate closest edge
if (abs(head - DOWN) >= abs(head - UP))
{
    for (i = 1, j = 0; j < temp1; i++, j++)
    {
        queue[i] = queue1[j];
    }

    queue[i] = UP;
    queue[i + 1] = 0;

    for (i = temp1 + 3, j = 0; j < temp2; i++, j++)
    {
        queue[i] = queue2[j];
    }
}
else
{
    for (i = 1, j = temp2 - 1; j >= 0; i++, j--)
    {
        queue[i] = queue2[j];
    }

    queue[i] = DOWN;
    queue[i + 1] = UP;

    for (i = temp2 + 3, j = temp1 - 1; j >= 0; i++, j--)
    {
        queue[i] = queue1[j];
    }
}

```

```

queue[0] = head;

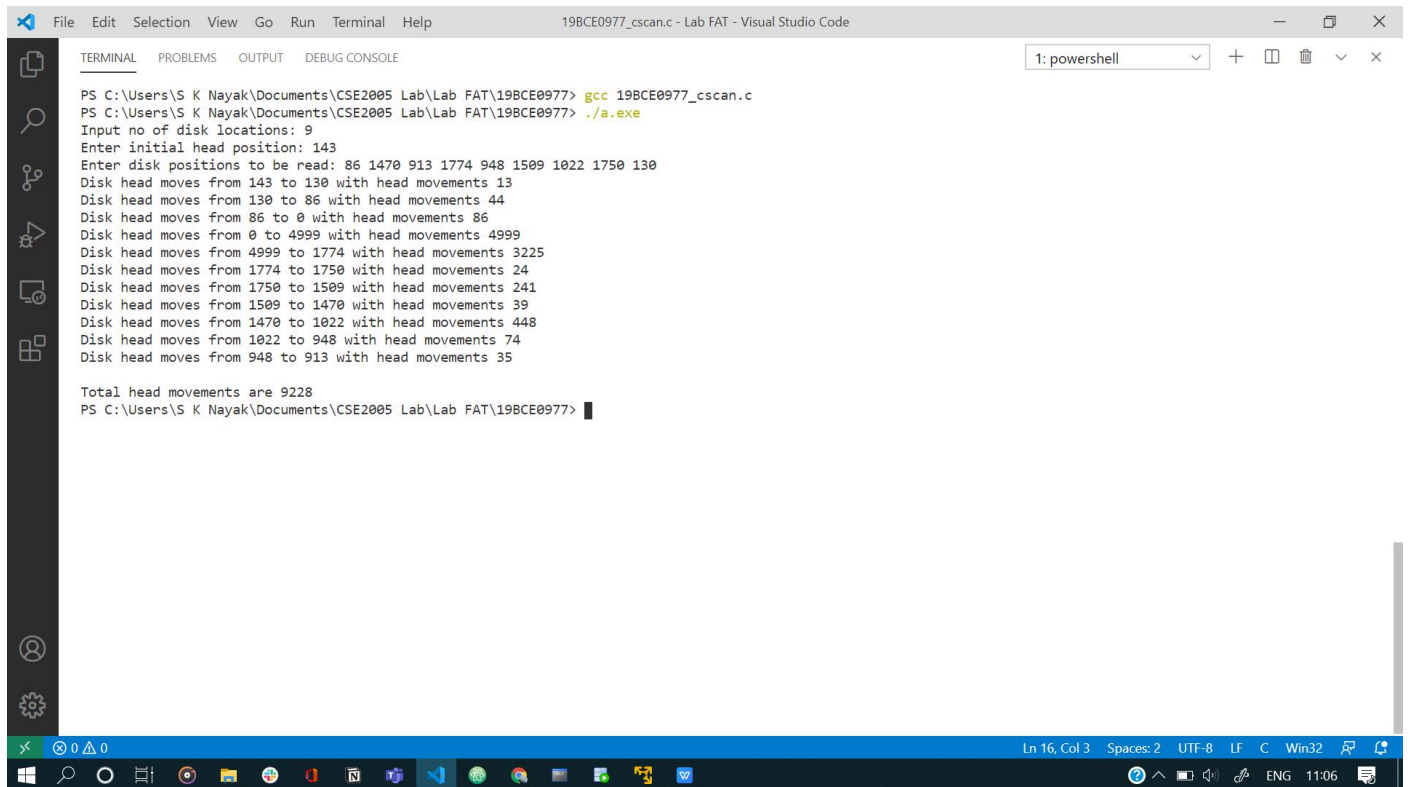
for (j = 0; j <= q_size + 1; j++)
{
    diff = abs(queue[j + 1] - queue[j]);
    seek += diff;
    printf("Disk head moves from %d to %d with head movements %d\n", queue[j], queue[j + 1]
, diff);
}

printf("\nTotal head movements are %d\n", seek);

return 0;
}

```

Output:



```

19BCE0977_cscan.c - Lab FAT - Visual Studio Code
1: powershell
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE
PS C:\Users\S K Nayak\Documents\CSE2005 Lab\Lab FAT\19BCE0977> gcc 19BCE0977_cscan.c
PS C:\Users\S K Nayak\Documents\CSE2005 Lab\Lab FAT\19BCE0977> ./a.exe
Input no of disk locations: 9
Enter initial head position: 143
Enter disk positions to be read: 86 1470 913 1774 948 1509 1022 1750 130
Disk head moves from 143 to 130 with head movements 13
Disk head moves from 130 to 86 with head movements 44
Disk head moves from 86 to 0 with head movements 86
Disk head moves from 0 to 4999 with head movements 4999
Disk head moves from 4999 to 1774 with head movements 3225
Disk head moves from 1774 to 1750 with head movements 24
Disk head moves from 1750 to 1509 with head movements 241
Disk head moves from 1509 to 1470 with head movements 39
Disk head moves from 1470 to 1022 with head movements 448
Disk head moves from 1022 to 948 with head movements 74
Disk head moves from 948 to 913 with head movements 35

Total head movements are 9228
PS C:\Users\S K Nayak\Documents\CSE2005 Lab\Lab FAT\19BCE0977>

```