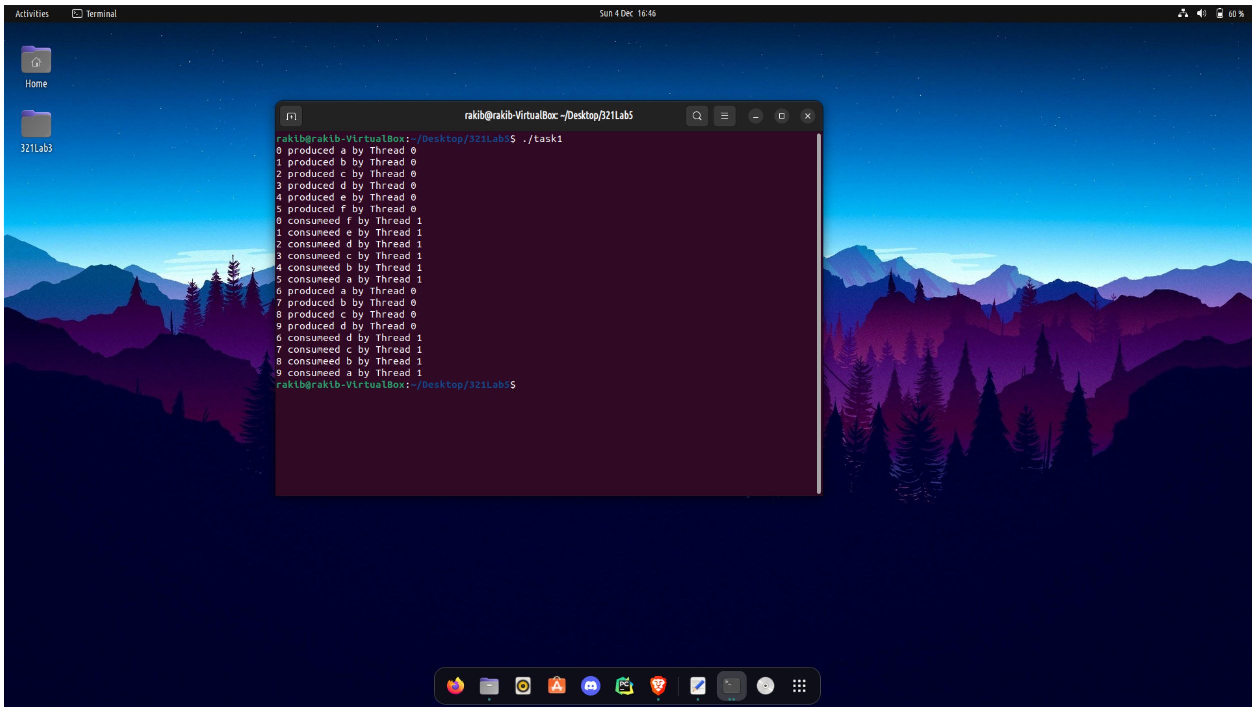
Task-1:

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

#define MAX 10
#define BUFLEN 6
#define NUMTHREAD 2

void *consumer(void *id);
void *producer(void *id);

char buffer[BUFLEN + 1];
char source[BUFLEN + 1];

int pCount = 0;
int cCount = 0;
int buflen;

pthread_mutex_t count_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t empty = PTHREAD_COND_INITIALIZER;
pthread_cond_t full = PTHREAD_COND_INITIALIZER;

int thread_id[NUMTHREAD] = {0, 1};

int src_index = 0;  // for accessing values
int count = 0;      // for ..

int main() {
    pthread_t thread[NUMTHREAD];

    strcpy(source, "abcdef");
    buflen = strlen(source);

    for (int i = 0; i < NUMTHREAD; i++) {
        int *a = malloc(sizeof(int));
        *a = thread_id[i];
        if (i == 0) {
            pthread_create(&thread[i], NULL, &producer, a);
        } else {
            pthread_create(&thread[i], NULL, &consumer, a);
        }
    }

    for (int i = 0; i < NUMTHREAD; i++) {
        pthread_join(thread[i], NULL);
    }
}

// producer
void *producer(void *id) {
    for (int i = 0; i < MAX; i++) {
        pthread_mutex_lock(&count_mutex);

        while (count >= BUFLEN) {
            pthread_cond_wait(&full, &count_mutex);
        }

        char x = source[src_index % BUFLEN];
        buffer[count] = x;
        src_index++;
        count++;
        pCount++;

        printf("%d produced %c by Thread %d\n", i, x, *(int *)id);

        pthread_cond_signal(&empty);
        pthread_mutex_unlock(&count_mutex);
    }
    pthread_exit(0);
}
// consumer
void *consumer(void *id) {
    for (int i = 0; i < MAX; i++) {
        pthread_mutex_lock(&count_mutex);

        while (count <= 0) {
            pthread_cond_wait(&empty, &count_mutex);
        }
        char y = buffer[count - 1];
        count--;
        cCount++;

        printf("%d consumeed %c by Thread %d\n", i, y, *(int *)id);
        pthread_cond_signal(&full);
        pthread_mutex_unlock(&count_mutex);
    }
    pthread_exit(0);
}
```

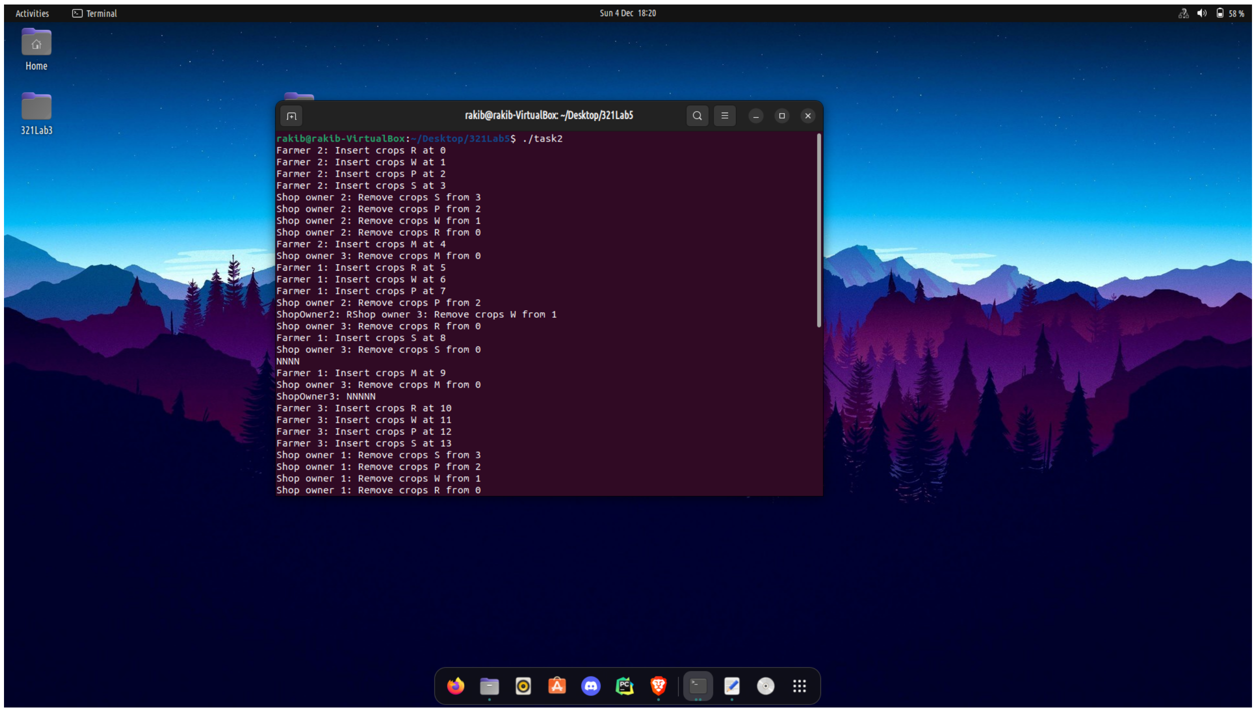


```
rakib@rakib-VirtualBox:~/Desktop/321Lab5$ ./task1
0 produced a by Thread 0
1 produced b by Thread 0
2 produced c by Thread 0
3 produced d by Thread 0
4 produced e by Thread 0
5 produced f by Thread 0
0 consumeed f by Thread 1
1 consumeed e by Thread 1
2 consumeed d by Thread 1
3 consumeed c by Thread 1
4 consumeed b by Thread 1
5 consumeed a by Thread 1
6 produced a by Thread 0
7 produced b by Thread 0
8 produced c by Thread 0
9 produced d by Thread 0
6 consumeed d by Thread 1
7 consumeed c by Thread 1
8 consumeed b by Thread 1
9 consumeed a by Thread 1
rakib@rakib-VirtualBox:~/Desktop/321Lab5$
```

<u>Task-2:</u>

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

#define MaxCrops 5
#define warehouseSize 5

sem_t empty;
sem_t full;

int c_index = 0;
int w_index = 0;
```

```
char crops[warehouseSize] = {
    'R', 'W', 'P', 'S', 'M',
};


char warehouse[warehouseSize] = {
    'N', 'N', 'N', 'N', 'N',
};

pthread_mutex_t mutex;

void *Farmer(void *far) {


    for (int i = 0; i < MaxCrops; i++) {

        sem_wait(&empty);
        pthread_mutex_lock(&mutex);
        // CT
        char x = crops[c_index % 5];
        warehouse[w_index] = x;

        printf("Farmer %d: Insert crops %c at %d\n", *(int *)far, x,
c_index);

        c_index++;
        w_index++;

        pthread_mutex_unlock(&mutex);
        sem_post(&full);
    }
    free(far);
    pthread_exit(0);
}
void *ShopOwner(void *sho) {


    for (int i = 0; i < MaxCrops; i++) {
        sem_wait(&full);
        pthread_mutex_lock(&mutex);

        char y = warehouse[w_index - 1];
        warehouse[w_index - 1] = 'N';
        int w = w_index - 1;
        printf("Shop owner %d: Remove crops %c from %d\n", *(int *)sho, y,
w);
        w_index--;

        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
    }
    printf("ShopOwner%d: ", *(int *)sho);
    for (int i = 0; i < warehouseSize; i++) {
        printf("%c", warehouse[i]);
    }
    printf("\n");

    free(sho);
    pthread_exit(0);
}
int main() {
    // intializing thread,mutex,semaphore
    pthread_t Far[5], Sho[5];
    pthread_mutex_init(&mutex, NULL);

    sem_init(&empty, 0, warehouseSize);
    sem_init(&full, 0, 0);


    int a[5] = {
        1, 2, 3, 4, 5,
    };


    for (int i = 0; i < 5; i++) {

        int *p = malloc(sizeof(int));
        int *q = malloc(sizeof(int));
        *p = a[i];
        *q = a[i];

        pthread_create(&Far[i], NULL, &Farmer, p);
        pthread_create(&Sho[i], NULL, &ShopOwner, q);
    }

    for (int i = 0; i < 5; i++) {
        pthread_join(Far[i], NULL);
        pthread_join(Sho[i], NULL);
    }

    pthread_mutex_destroy(&mutex);
    sem_destroy(&empty);
    sem_destroy(&full);

    return 0;
}
```



```
rakib@rakib-VirtualBox:~/Desktop/321Lab5$ ./task2
Farmer 2: Insert crops R at 0
Farmer 2: Insert crops W at 1
Farmer 2: Insert crops P at 2
Farmer 2: Insert crops S at 3
Shop owner 2: Remove crops S from 3
Shop owner 2: Remove crops P from 2
Shop owner 2: Remove crops W from 1
Shop owner 2: Remove crops R from 0
Farmer 2: Insert crops M at 4
Shop owner 3: Remove crops M from 0
Farmer 1: Insert crops R at 5
Farmer 1: Insert crops W at 6
Farmer 1: Insert crops P at 7
Shop owner 2: Remove crops P from 2
ShopOwner2: RShop owner 3: Remove crops W from 1
Shop owner 3: Remove crops R from 0
Farmer 1: Insert crops S at 8
Shop owner 3: Remove crops S from 0
NNNN
Farmer 1: Insert crops M at 9
Shop owner 3: Remove crops M from 0
ShopOwner3: NNNNN
Farmer 3: Insert crops R at 10
Farmer 3: Insert crops W at 11
Farmer 3: Insert crops P at 12
Farmer 3: Insert crops S at 13
Shop owner 1: Remove crops S from 3
Shop owner 1: Remove crops P from 2
Shop owner 1: Remove crops W from 1
Shop owner 1: Remove crops R from 0
```