

Program 5: Process/Thread Synchronization:

1. Dining Philosopher Problem

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

#define NUM_PHIL 5

sem_t chopstick[NUM_PHIL];

void *eat_phil(void *arg) {
    long int index = (long int) arg;
    if (index % 2 == 0) {
        sem_wait(&chopstick[(index+1) % NUM_PHIL]);
        sem_wait(&chopstick[index]);
    } else {
        sem_wait(&chopstick[index]);
        sem_wait(&chopstick[(index+1) % NUM_PHIL]);
    }
    printf("Philosopher %d is eating\n", index);
    sem_post(&chopstick[index]);
    sem_post(&chopstick[(index+1) % NUM_PHIL]);
    return NULL;
}

int main() {
    long int i;
    pthread_t threads[NUM_PHIL];
    printf("Enter the order in which threads should start: ");

    for (i = 0; i < NUM_PHIL; i++) {
        sem_init(&chopstick[i], 0, 1);
    }

    for (i = 0; i < NUM_PHIL; i++) {
        long int n;
        scanf("%ld", &n);
        if (n < 0 || n >= NUM_PHIL) {
            printf("Number %d not in range!!\n", n);
            return 1;
        }
        pthread_create(&threads[i], NULL, eat_phil, (void *)n);
    }

    for (i = 0; i < NUM_PHIL; i++) {
        pthread_join(threads[i], NULL);
    }

    return 0;
}
```

Output:

```
OS : bash — Konsole
File Edit View Bookmarks Settings Help
mahesh@mahesh:~/Code/Lab/OS$ ./dining
Enter the order in which threads should start: 4 1 3 2 0
Philosopher 4 is eating
Philosopher 0 is eating
Philosopher 3 is eating
Philosopher 1 is eating
Philosopher 2 is eating
mahesh@mahesh:~/Code/Lab/OS$ ./dining
Enter the order in which threads should start: 1 1 0 0 1
Philosopher 1 is eating
Philosopher 1 is eating
Philosopher 1 is eating
Philosopher 0 is eating
Philosopher 0 is eating
mahesh@mahesh:~/Code/Lab/OS$ ./dining
Enter the order in which threads should start: 4 2 3 1 0
Philosopher 2 is eating
Philosopher 1 is eating
Philosopher 0 is eating
Philosopher 3 is eating
Philosopher 4 is eating
```

2. Reader-writer problem:

```
#include<stdio.h>
#include<pthread.h>
#include<sys/types.h>
#include<semaphore.h>
#include<string.h>

char data[40];

sem_t mutex, rw_mutex;
int read_count = 0;

char messages[10][40];

void *writer(void *arg) {
    long int wn = (long int)arg;
    char *s = messages[wn];
    size_t sz = strlen(s);
    sem_wait(&rw_mutex);
    memcpy(data,s,sz + 1); // writing
    printf("Writer %ld Wrote: %s\n", wn, data);
    sem_post(&rw_mutex);
    return NULL;
}

void *reader(void *arg) {
    long int rn = (long int) arg;
    sem_wait(&mutex);
    read_count ++;
    if (read_count == 1) {
        sem_wait(&rw_mutex);
    }
    sem_post(&mutex);
    // Reading
    printf("Reader %ld read: %s\n", rn, data);
    sem_wait(&mutex);
    read_count--;
    if (read_count == 0) sem_post(&rw_mutex);
}
```

```

        sem_post(&mutex);
        return NULL;
    }

int main() {
    int nr, nw, i, j;
    pthread_t readers[10], writers[10];
    sem_init(&mutex, 0, 1);
    sem_init(&rw_mutex, 0, 1);

    memcpy(data, "(Nothing)", 10);
    printf("Numbers of readers & writers: ");
    scanf("%d %d", &nr, &nw);
    if (nr > 10 || nr <= 0 || nw > 10 || nw <= 0) {
        printf("Enter valid numbers!!\n");
        return 1;
    }
    for (i = 0; i < nw; i++) {
        printf("Data to write from writer %d: ", i);
        scanf(" %38[^\n]", messages[i]);
    }

    for (i = 0; i < nr; i++) {
        pthread_create(&readers[i], NULL, reader, (void *)i);
    }

    for (i = 0; i < nw; i++) {
        pthread_create(&writers[i], NULL, writer, (void *)i);
    }

    for (i = 0; i < nr; i++) {
        pthread_join(readers[i], NULL);
    }
    for (i = 0; i < nw; i++) {
        pthread_join(writers[i], NULL);
    }
}

```

```

OS : bash — Konsole
File Edit View Bookmarks Settings Help
mahesh@mahesh:~/Code/Lab/OS$ ./rdwr
Numbers of readers & writers: 2 6
Data to write from writer 0: String0
Data to write from writer 1: String1
Data to write from writer 2: String2
Data to write from writer 3: String3
Data to write from writer 4: String4
Data to write from writer 5: String5
Reader 1 read: (Nothing)
Writer 1 Wrote: String1
Writer 4 Wrote: String4
Writer 2 Wrote: String2
Writer 0 Wrote: String0
Reader 0 read: String0
Writer 5 Wrote: String5
Writer 3 Wrote: String3
mahesh@mahesh:~/Code/Lab/OS$ ./rdwr
Numbers of readers & writers: 1 1
Data to write from writer 0: Hey
Reader 0 read: (Nothing)
Writer 0 Wrote: Hey
mahesh@mahesh:~/Code/Lab/OS$

```

3. Producer-Consumer problem:

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<semaphore.h>
#include<sys/types.h>
#include<assert.h>

#define MAX 5

sem_t empty, full, mutex;

int pt, ct;

typedef struct queue {
    int items[MAX];
    int size, begin;
} Queue;

Queue *q;

int dequeue(Queue *q) {
    int r;
    if (q->size == 0) return -1;

    r = q->items[q->begin];
    q->size--;
    q->begin++;
    if(q->begin == MAX) q->begin = 0;
    return r;
}

int enqueue(Queue *q, int i) {
    if (q->size == MAX) {
        // Queue Full
        return -1;
    }
    q->items[(q->begin+q->size) % MAX] = i;
    q->size++;
    return 0;
}

void *prod(void *arg) {
    int pn = (int) arg;
    int i;
    for(i = 0; i < ct; i++) {
        sem_wait(&empty);
        sem_wait(&mutex);
        printf("Prod %d produced %d\n", pn, pn*10+i);
        enqueue(q, pn*10+i);
        sem_post(&mutex);
        sem_post(&full);
    }
    return NULL;
}

void *cons(void *arg) {
    int cn = (int) arg;
    int i, n;
    for (i = 0; i < pt; i++) {
        sem_wait(&full);
        sem_wait(&mutex);
```

```

        printf(" Cons %d consumed %d\n", cn, dequeue(q));
        sem_post(&mutex);
        sem_post(&empty);
    }
    return NULL;
}

int main() {
    Queue qm;
    int i;
    pthread_t prods[10], conss[10];
    qm.size = 0;
    q = &qm;
    sem_init(&full, 0, 0);
    sem_init(&mutex, 0, 1);
    sem_init(&empty, 0, MAX);
    printf("Number of producers & consumers: ");
    scanf("%d %d", &pt, &ct);
    assert(pt <= 10 && ct <= 10);

    for(i = 0; i < pt; i++) {
        pthread_create(&prods[i], NULL, prod, (void *)i);
    }

    for(i = 0; i < ct; i++) {
        pthread_create(&conss[i], NULL, cons, (void *)i);
    }

    for(i = 0; i < pt; i++) {
        pthread_join(prods[i], NULL);
        pthread_join(conss[i], NULL);
    }

    return 0;
}

```

```

OS : bash — Konsole
File Edit View Bookmarks Settings Help
mahesh@mahesh:~/Code/Lab/OS$ ./prodcons
Number of producers & consumers: 3 4
Prod 1 produced 10
Prod 1 produced 11
Prod 1 produced 12
Prod 1 produced 13
Cons 3 consumed 10
Cons 3 consumed 11
Prod 2 produced 20
Prod 2 produced 21
Prod 2 produced 22
Cons 3 consumed 12
Cons 1 consumed 13
Cons 1 consumed 20
Prod 2 produced 23
Cons 1 consumed 21
Cons 2 consumed 22
Cons 0 consumed 23
Prod 0 produced 0
Prod 0 produced 1
Prod 0 produced 2
Prod 0 produced 3
Cons 2 consumed 0
Cons 2 consumed 1
Cons 0 consumed 2
Cons 0 consumed 3
mahesh@mahesh:~/Code/Lab/OS$

```