Dining Philosophers Problem

Readers-Writers problem

Producer-Consumers Problem

Dining Philosopher's Problem

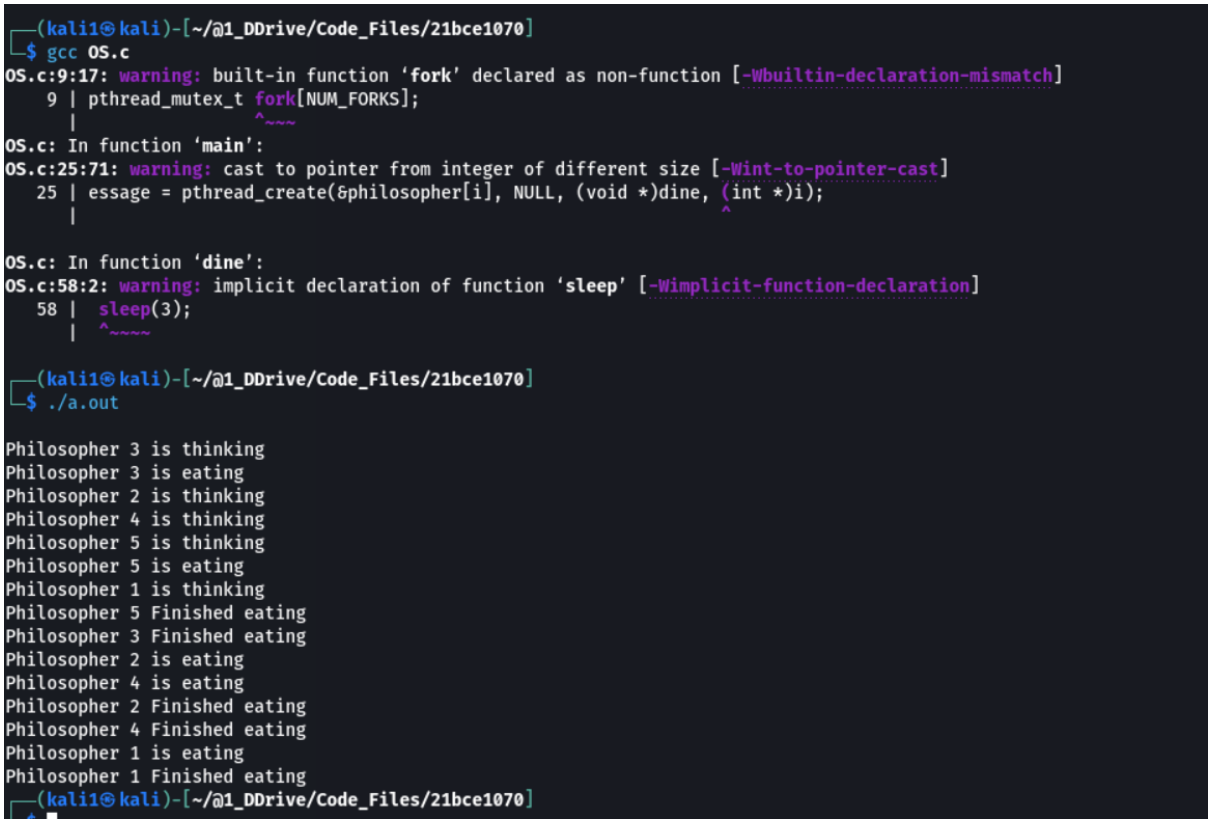
Code

```
#define NUM_PHILOSOPHERS 5
#define NUM_FORKS 5
void dine(int n);
pthread_t philosopher[NUM_PHILOSOPHERS];
pthread_mutex_t fork[NUM_FORKS];
int main()
{
 int i, status_message;
 void *msg;
 for (i = 1; i <= NUM_FORKS; i++)
 {
 status_message = pthread_mutex_init(&fork[i], NULL);
 if (status_message == -1)
 {
 printf("\n Mutex initialization failed");
 exit(1);
 }
 }
 for (i = 1; i <= NUM_PHILOSOPHERS; i++)
 {
 status_message = pthread_create(&philosopher[i], NULL, (void *)dine, (int *)i);
 if (status_message != 0)
 {
```

```
printf("\n Thread creation error \n");
exit(1);
}
}
for (i = 1; i <= NUM_PHILOSOPHERS; i++)
{
status_message = pthread_join(philosopher[i], &msg);
if (status_message != 0)
{
printf("\n Thread join failed \n");
exit(1);
}
}
for (i = 1; i <= NUM_FORKS; i++)
{
status_message = pthread_mutex_destroy(&fork[i]);
if (status_message != 0)
{
printf("\n Mutex Destroyed \n");
exit(1);
}
}
return 0;
}
void dine(int n)
{
printf("\nPhilosopher %d is thinking ", n);
pthread_mutex_lock(&fork[n]);
pthread_mutex_lock(&fork[(n + 1) % NUM_FORKS]);
printf("\nPhilosopher %d is eating ", n);
sleep(3);
```

```
 pthread_mutex_unlock(&fork[n]);
 pthread_mutex_unlock(&fork[(n + 1) % NUM_FORKS]);
 printf("\nPhilosopher %d Finished eating ", n);
}
```

Output



Readers-Writers Problem

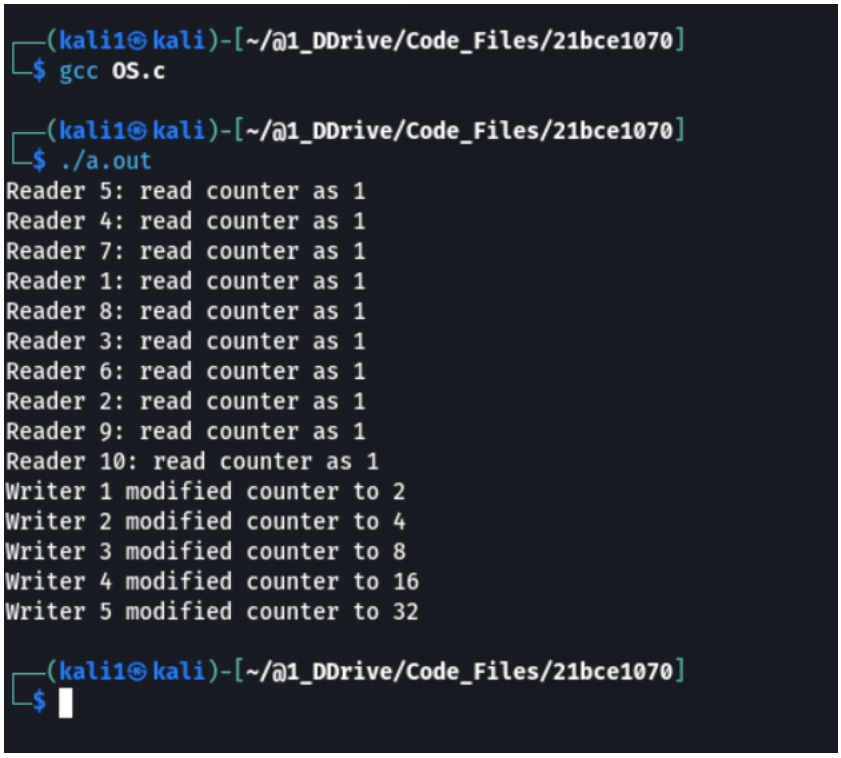
Code

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

```
sem_t writerSemaphore;
pthread_mutex_t mutex;
int counter = 1;
int readerCount = 0;

void *writer(void *wno)
{
	sem_wait(&writerSemaphore);
	counter = counter * 2;
	printf("Writer %d modified counter to %d\n", (*((int *)wno)), counter);
	sem_post(&writerSemaphore);
}

void *reader(void *rno)
{
	pthread_mutex_lock(&mutex);
	readerCount++;
	if (readerCount == 1) {
		sem_wait(&writerSemaphore);
	}
	pthread_mutex_unlock(&mutex);
	printf("Reader %d: read counter as %d\n", *((int *)rno), counter);
	pthread_mutex_lock(&mutex);
	readerCount--;
	if (readerCount == 0) {
		sem_post(&writerSemaphore);
	}
	pthread_mutex_unlock(&mutex);
}
```

```
int main()
{
	pthread_t read[10], write[5];
	pthread_mutex_init(&mutex, NULL);
	sem_init(&writerSemaphore, 0, 1);
	int arr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

	for (int i = 0; i < 10; i++) {
		pthread_create(&read[i], NULL, (void *)reader, (void *)&arr[i]);
	}

	for (int i = 0; i < 5; i++) {
		pthread_create(&write[i], NULL, (void *)writer, (void *)&arr[i]);
	}

	for (int i = 0; i < 10; i++) {
		pthread_join(read[i], NULL);
	}

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

	pthread_mutex_destroy(&mutex);
	sem_destroy(&writerSemaphore);
	return 0;
}
```

Output



Producer Consumer Problem

Code

```
#include <stdio.h>
#include <stdlib.h>
int mutex = 1;
int full = 0;
int empty = 4, data = 0;
void producer()
{
 --mutex;
 ++full;
 --empty;
 data++;
 printf("\nProducer produces item number: %d\n", data);
```

```
 ++mutex;
}
void consumer()
{
 --mutex;
 --full;
 ++empty;
 printf("\nConsumer consumes item number: %d.\n", data);
 data--;
 ++mutex;
}
int main()
{
 int n, i;
 printf("\n1. Enter 1 for Producer"
 "\n2. Enter 2 for Consumer"
 "\n3. Enter 3 to Exit");
 for (i = 1; i > 0; i++)
 {
 printf("\nEnter your choice: ");
 scanf("%d", &n);
 switch (n)
 {
 case 1:
 if ((mutex == 1) && (empty != 0))
 {
 producer();
 }
 else
 {
 printf("The Buffer is full. New data cannot be produced!");
```

```
}
break;
case 2:
if ((mutex == 1) && (full != 0))
{
consumer();
}
else
{
printf("The Buffer is empty! New data cannot be consumed!");
}
break;
case 3:
exit(0);
break;
}
}
}
```

Output

```
┌──(kali1㉿kali)-[~/@1_DDrive/Code_Files/21bce1070]
└─$ gcc OS.c

┌──(kali1㉿kali)-[~/@1_DDrive/Code_Files/21bce1070]
└─$ ./a.out

1. Enter 1 for Producer
2. Enter 2 for Consumer
3. Enter 3 to Exit
Enter your choice: 1

Producer produces item number: 1

Enter your choice: 2

Consumer consumes item number: 1.

Enter your choice: 1

Producer produces item number: 1

Enter your choice: 2

Consumer consumes item number: 1.

Enter your choice: 2
The Buffer is empty! New data cannot be consumed!
Enter your choice: 1

Producer produces item number: 1

Enter your choice: 1

Producer produces item number: 2

Enter your choice: 1

Producer produces item number: 3

Enter your choice: 2

Consumer consumes item number: 3.

Enter your choice: 2

Consumer consumes item number: 2.

Enter your choice: 2

Consumer consumes item number: 1.

Enter your choice: 2
The Buffer is empty! New data cannot be consumed!
Enter your choice: 3
```