Main Idea

Each barber is their own process. Each barber has shared access to the waiting queue of customers and each child has access to the waiting queue and the count variable to count when a customer is turned away.
For multiple processes to access a shared resource without a racing condition, we use semaphores to insure one process accesses the shared resource at a time and everybody else waits their turn.
We create 1 set of 4 semaphores: 1 for the customers, 1 for each type of barber (Male Only, Female Only and Male and Female) and a mutual exclusion semaphore.
During initialization of the semaphores, we set 0 as the init value for Customers, because there are no customers. When a customer sits on a chair, the customer semaphore is increased by 1. When a barber takes a customer from the queue, the semaphore is decreased by 1.
For Barber semaphores, we initialize the value to the number of barbers of each type in the shop. When a barber is in use, the barber count is decreased. When it is 0, the customers should wait until their barber is available. When a barber finishes, the barber count is increased by 1 to signal availability.
Mutex semaphore is set to 1 so no two processes accessing the shared resource at the same time.
The resources are shared across processes by using a mmap() to define the address in memory.
The queue is an array where each variable is the sex of the customer. When a barber is available, they check if there is a customer they can serve, if they can, the customer is taken out of the queue and all items after it go 1 place forward, leaving the last chair empty for the next customer which enters the shop.

Limitation

The barbers are light sleepers, each customer which enters wakes them up and the barbers check if they can serve the new customer.

Example Program Output with Commentary (1) - One of each Barber

```
#define BARBERSCOUNT_M 1 // (N1)
```

```
#define BARBERSCOUNT_F 1 // (N2)
#define BARBERSCOUNT_MF 1 // (N3)
#define CHAIRS 5 // (M)


// initialize the queue array
There are 5 chairs.
Customer 1 is seated.
// customer of type 2 (F) is added to the array at the start, the rest of the seats remain empty
array[0]: 2
array[1]: 0
array[2]: 0
array[3]: 0
array[4]: 0

// Barber_M has no customers (customers 1 are male) and is going to sleep.
barber[16817]: type 1 has no customers and is going to sleep
Customer 2 is seated.
// another customer is seated, again it is a type 2, so Female
array[0]: 2
array[1]: 2
array[2]: 0
array[3]: 0
array[4]: 0

// When Customer is seated,  the barber checks if they can serve them
barber[16817]: type 1 has no customers and is going to sleep
// Barber of type 2, can serve them
barber[16818]: type 2 is now cutting hair of customer: 2
Customer 3 is seated.
// when a barber serves the customer, they are removed from the queue
// customer 3 is seated in place [1] and is Female (type 2)
array[0]: 2
array[1]: 2
array[2]: 0
array[3]: 0
array[4]: 0

// barber of type 3 (can serve male and female) gets the first customer they can serve
barber[16819]: type 3 is now cutting hair of customer: 2
Customer 4 is seated.
// customer 4 is Male (type 1)
array[0]: 2
array[1]: 1
array[2]: 0
array[3]: 0
array[4]: 0

//when the customer entered, the sleeping barber checked if they can serve them, they could
barber[16817]: type 1 is now cutting hair of customer: 1
Customer 5 is seated.
array[0]: 2
array[1]: 1
array[2]: 0
```

```
array[3]: 0
array[4]: 0

Customer 6 is seated.
array[0]: 2
array[1]: 1
array[2]: 1
array[3]: 0
array[4]: 0

Customer 7 is seated.
array[0]: 2
array[1]: 1
array[2]: 1
array[3]: 1
array[4]: 0

barber[16818]: type 2 is now cutting hair of customer: 2
Customer 8 is seated.
array[0]: 1
array[1]: 1
array[2]: 1
array[3]: 1
array[4]: 0

barber[16819]: type 3 is now cutting hair of customer: 1
Customer 9 is seated.
array[0]: 1
array[1]: 1
array[2]: 1
array[3]: 2
array[4]: 0

Customer 10 is seated.
array[0]: 1
array[1]: 1
array[2]: 1
array[3]: 2
array[4]: 2

barber[16817]: type 1 is now cutting hair of customer: 1
Customer 11 is seated.
array[0]: 1
array[1]: 1
array[2]: 2
array[3]: 2
array[4]: 1
// the queue is full
Customer 12 left the shop.
Customer 13 left the shop.
// barber found the first female they can serve in the queue and took them
barber[16818]: type 2 is now cutting hair of customer: 2
Customer 14 is seated.
array[0]: 1
```

```
array[1]: 1
array[2]: 2
array[3]: 1
array[4]: 2

barber[16819]: type 3 is now cutting hair of customer: 1
Customer 15 is seated.
array[0]: 1
array[1]: 2
array[2]: 1
array[3]: 2
array[4]: 1

Customer 16 left the shop.
barber[16817]: type 1 is now cutting hair of customer: 1
Customer 17 is seated.
array[0]: 2
array[1]: 1
array[2]: 2
array[3]: 1
array[4]: 1

Customer 18 left the shop.
// signal interrupt ctrl-c
Program ended with exit code: 9
```

Example Program Output with Commentary (2) - Some barbers are null

With more chairs and no only male or only female barber, the MF barbers take on the workload.

```
#define BARBERSCOUNT_M 0 // (N1)
#define BARBERSCOUNT_F 0 // (N2)
#define BARBERSCOUNT_MF 2 // (N3)
#define CHAIRS 10 // (M)
```

```
There are 10 chairs.
Customer 1 is seated.
array[0]: 2
array[1]: 0
array[2]: 0
array[3]: 0
array[4]: 0
array[5]: 0
array[6]: 0
array[7]: 0
array[8]: 0
array[9]: 0

barber[17085]: type 3 is now cutting hair of customer: 2
Customer 2 is seated.
array[0]: 2
```

```
array[1]: 0
array[2]: 0
array[3]: 0
array[4]: 0
array[5]: 0
array[6]: 0
array[7]: 0
array[8]: 0
array[9]: 0

barber[17086]: type 3 is now cutting hair of customer: 2
Customer 3 is seated.
array[0]: 2
array[1]: 0
array[2]: 0
array[3]: 0
array[4]: 0
array[5]: 0
array[6]: 0
array[7]: 0
array[8]: 0
array[9]: 0

Customer 4 is seated.
array[0]: 2
array[1]: 1
array[2]: 0
array[3]: 0
array[4]: 0
array[5]: 0
array[6]: 0
array[7]: 0
array[8]: 0
array[9]: 0

Customer 5 is seated.
array[0]: 2
array[1]: 1
array[2]: 1
array[3]: 0
array[4]: 0
array[5]: 0
array[6]: 0
array[7]: 0
array[8]: 0
array[9]: 0

Customer 6 is seated.
array[0]: 2
array[1]: 1
array[2]: 1
array[3]: 1
array[4]: 0
array[5]: 0
```

```
array[6]: 0
array[7]: 0
array[8]: 0
array[9]: 0

barber[17085]: type 3 is now cutting hair of customer: 2
Customer 7 is seated.
array[0]: 1
array[1]: 1
array[2]: 1
array[3]: 1
array[4]: 0
array[5]: 0
array[6]: 0
array[7]: 0
array[8]: 0
array[9]: 0

barber[17086]: type 3 is now cutting hair of customer: 1
Customer 8 is seated.
array[0]: 1
array[1]: 1
array[2]: 1
array[3]: 1
array[4]: 0
array[5]: 0
array[6]: 0
array[7]: 0
array[8]: 0
array[9]: 0

Customer 9 is seated.
array[0]: 1
array[1]: 1
array[2]: 1
array[3]: 1
array[4]: 2
array[5]: 0
array[6]: 0
array[7]: 0
array[8]: 0
array[9]: 0

Customer 10 is seated.
array[0]: 1
array[1]: 1
array[2]: 1
array[3]: 1
array[4]: 2
array[5]: 2
array[6]: 0
array[7]: 0
array[8]: 0
array[9]: 0
```

```
Customer 11 is seated.
array[0]: 1
array[1]: 1
array[2]: 1
array[3]: 1
array[4]: 2
array[5]: 2
array[6]: 1
array[7]: 0
array[8]: 0
array[9]: 0

Customer 12 is seated.
array[0]: 1
array[1]: 1
array[2]: 1
array[3]: 1
array[4]: 2
array[5]: 2
array[6]: 1
array[7]: 2
array[8]: 0
array[9]: 0

barber[17085]: type 3 is now cutting hair of customer: 1
Customer 13 is seated.
array[0]: 1
array[1]: 1
array[2]: 1
array[3]: 2
array[4]: 2
array[5]: 1
array[6]: 2
array[7]: 1
array[8]: 0
array[9]: 0

barber[17086]: type 3 is now cutting hair of customer: 1
Customer 14 is seated.
array[0]: 1
array[1]: 1
array[2]: 2
array[3]: 2
array[4]: 1
array[5]: 2
array[6]: 1
array[7]: 1
array[8]: 0
array[9]: 0

Customer 15 is seated.
array[0]: 1
array[1]: 1
```

```
array[2]: 2
array[3]: 2
array[4]: 1
array[5]: 2
array[6]: 1
array[7]: 1
array[8]: 2
array[9]: 0

Customer 16 is seated.
array[0]: 1
array[1]: 1
array[2]: 2
array[3]: 2
array[4]: 1
array[5]: 2
array[6]: 1
array[7]: 1
array[8]: 2
array[9]: 2

Customer 17 left the shop.
Customer 18 left the shop.
barber[17085]: type 3 is now cutting hair of customer: 1
Customer 19 is seated.
array[0]: 1
array[1]: 2
array[2]: 2
array[3]: 1
array[4]: 2
array[5]: 1
array[6]: 1
array[7]: 2
array[8]: 2
array[9]: 2

Program ended with exit code: 9
```

Example Program Output with Commentary (3) - 2 of each Barber

Multiple barbers of each type

```
#define BARBERSCOUNT_M 2 // (N1)
#define BARBERSCOUNT_F 2 // (N2)
#define BARBERSCOUNT_MF 2 // (N3)
#define CHAIRS 3 // (M)
```

```
There are 3 chairs.
Customer 1 is seated.
array[0]: 2
array[1]: 0
array[2]: 0
```

```
// limitation, type 2 barber should be up and working on it.
barber[17149]: type 1 has no customers and is going to sleep
Customer 2 is seated.
array[0]: 2
array[1]: 2
array[2]: 0

// barber[17154] wakes up and serves customer
barber[17154]: type 3 is now cutting hair of customer: 2
Customer 3 is seated.
array[0]: 2
array[1]: 2
array[2]: 0

barber[17149]: type 1 has no customers and is going to sleep
barber[17152]: type 2 is now cutting hair of customer: 2
Customer 4 is seated.
array[0]: 2
array[1]: 1
array[2]: 0

barber[17150]: type 1 is now cutting hair of customer: 1
// second barber of type 3 is cutting the hair of the customer, because the 1st barber is occupied
barber[17153]: type 3 is now cutting hair of customer: 2
Customer 5 is seated.
array[0]: 1
array[1]: 0
array[2]: 0

barber[17149]: type 1 is now cutting hair of customer: 1
barber[17151]: type 2 has no customers and is going to sleep
Customer 6 is seated.
array[0]: 1
array[1]: 0
array[2]: 0

barber[17151]: type 2 has no customers and is going to sleep
Customer 7 is seated.
array[0]: 1
array[1]: 1
array[2]: 0

barber[17151]: type 2 has no customers and is going to sleep
Customer 8 is seated.
array[0]: 1
array[1]: 1
array[2]: 1

barber[17151]: type 2 has no customers and is going to sleep
Customer 9 left the shop.
Customer 10 left the shop.
Customer 11 left the shop.
Customer 12 left the shop.
Program ended with exit code: 9
```