AOS Assignment 5

List of files created:

1) future.h: This is the header file. It includes declaration for the following:
    - Future states – FUTURE_EMPTY, FUTURE_WAITING, FUTURE_VALID
    - Future modes – FUTURE_EXCLUSIVE, FUTURE_SHARED, FUTURE_QUEUE
      FUTURE_EXCLUSIVE: This flag involves only 2 threads executing at a time and doesnot clude any queue implementation.

      FUTURE_SHARED:
      With this flag, there are many threads executing and there exists one-to-many relationship between threads calling future_set() and future_get(). Queues are implemented for all waiting threads.

      FUTURE_QUEUE:
      With this flag, there are many threads executing and there is many to many relationship between threads calling future_set() and future_get().Queues are implemented for all waiting threads.

    - Structure of future changed to include set_queue and get_queue queues
    - System calls – future_alloc(), future_free(), future_get(), future_set()

    Created by: Amruta and Ajinkya

2) xsh_prodcons.c: This file is modified such that when command prodcons with argument -f is executed futures with all three modes are created and scheduled. Without this argument, producer-consumer synchronization would be executed. It also involves various error handling and help functionality.
   Created by: Ajinkya and Amruta

3) future_cons.c:
   The consumer process calls future_get() to consume the value if present within the future. If the value is not present, then it waits for the producer process to produce the value in the future. Once the value is consumed i.e. printed on console, future_free() is called.
   Created by:Ajinkya

4) future_prod.c:
   This is the code for producer process which produces values and calls future_set() to set the value within the future. After the value is set, it signals the consumer to consume the value.
   Created by: Amruta

5) System calls:
   future_alloc.c: Allocates memory for the future and both queues by calling getmem().
   future_free.c: Clears memory used by the future and queues by calling freemem().
   Created by: Ajinkya and Amruta

6) f_queue.h: This is the header file for queue implementation. The structure of queue is :

```
struct futqueue{
int parray[QUEUE_SIZE];
int front;
int rear;
int count;
} queue;
```

   Created by: Amruta

7) f_queue.c: This file includes all queue functionalities i.e. enqueue, dequeue and check if empty. To make the functionality modular we have used queue pointer instead of queue.
   Created by: Ajinkya