**Report Assignment 5**

We are using a circular queue to implement future with FUTURE\_SHARED and FUTURE\_QUEUE functionality through struct futque. Our queue is of size 10 and it holds the process ID’s of all the waiting processes.

**Assumption 1:** At max 10 process can wait i.e. 10 can be in set queue and 10 in get queue.

**Assumption 2:** IN FUTURE\_SHARED mode once a producer has produced a value and has resumed all the waiting consumer and further when all the consumer has consumed the value, the state of future is changed to FUTURE\_EMPTY.

|  |  |
| --- | --- |
| **File Name** | **Description** |
| Future.h | This header file contains the definition of the futen structure, constants for the state and mode of future as well as the prototypes of the following functions used:  future\_alloc  future\_free  future\_get  future\_set |
| future\_alloc.c | This is the system call that allocates memory to a future. Here the parameter to the future is the mode of the future, which as follows:  -FUTURE\_EXCLUSIVE: This mode only allows one to one relation between the producer and the consumer, allowing only one producer to produce and only one consumer to wait at a given time.  -FUTURE\_SHARED: This mode allows many to one relation, allowing many consumers to wait for a future to get a value while only one producer produces value.  -FUTURE\_QUEUE: This mode allows many to many relation between the producer and consumer. |
| future\_free.c | This is a system call which de allocates memory assigned to a future. This function checks for any waiting processes on the future or if any process has written something to the future before de allocation. i.e. if any process is waiting on the future then future\_free will kill all the waiting process and will free the memory . This is because these process would be wasting computational resources as well as memory by waiting for an unavailable value. |
| future\_get.c | This function is called by any procedure if it wants to read any value from the future. If the state of the future is FUTURE\_EMPTY then the calling function undergoes busy waiting and the state of the future changes to FUTURE\_WAIT until and unless some value is produced and assigned to the future making its state FUTURE\_VALID.  When FUTURE\_VALID state is achieved, the value of the future is returned by making a call to it and the state is changed back to FUTURE\_EMPTY.  IF the state is currently FUTURE\_WAIT, then a call to the function would result in an error as the mode is EXCLUSIVE.  In case of any other modes, the process will enter the queue futqueue as a waiting process and its state will change to PR\_WAIT. |
| future\_set.c | This system call is called by any value which wants to assign value to the future. Initially as the future is in FUTURE\_EMPTY state, a call to this procedure will assign a value to future and change its value to FUTURE\_VALID in EXCLUSIVE or SHARED mode. But in case of queue mode the state still remains EMPTY as the producer needs to wait for the consumer.  When the state of the future is FUTURE\_VALID, a call to the function will result into an error when the mode is EXCLUSIVE or SHARED. But if the mode is queue the producer will be put in the set queue.  If a call is made when future is in FUTURE\_WAIT state then it will change the value and the state to FUTURE\_VALID. |
| future\_prod.c | This is the producer which produces a value and assigns it to the future by calling the future\_set() system call. |
| future\_cons.c | This is the consumer which reads values from the future using the future\_get() system call. |
| xsh\_prodcons.c | This is the main class which can be invoked directly via the command line argument. In order to check future implementation this class must be invoked by passing the –f. |
| Fqueue.h | This file contains the declaration of the queue and its related structures.  #ifndef NMAX  #define NMAX 10  #endif  struct futque{  pid32 futqueue[NMAX];  int32 head;  int32 tail;  int32 count;  };  Here maximum size futqueue is 10. |

**Responsibilities/Tasks Member ID**

Future.h pandeyh

Fqueue.h sshalabh

Future\_alloc.c sshalabh

Future\_free.c pandeyh

Future\_set.c sshalabh

Future\_get.c sshalabh

Future\_prods.c pandeyh

Future\_cons.c pandeyh

Xsh\_prodcons.c sshalabh

Creation of Test Cases pandeyh

Testing and Execution of code sshalabh

Bug Fixing sshalabh

Report Creation pandeyh