The objective of assignment 4 is to understand the functionality of futures by implementing them by developing system calls. Futures are basically a mechanism to synchronously or asynchronously produce data and in this process improving the ability to manage concurrent processes. Futures can be regarded as placeholders for values that are not yet generated.

The future developed in our assignment is asynchronous in nature, i.e. when the producer produces a value, it stores it in the future and exits. The producer does not informs the consumer that is associated with the same future about the produced value. The producer will produce values independently of the consumers. The consumer hence checks for any change in state of the future. If the consumer has to consume any value it will enter in the waiting state and will not inform the producer about it.

|  |  |
| --- | --- |
| **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 functions used:  future\_alloc  future\_free  future\_get  future\_set |
| Prodcons.h | This header file contains the definition of future\_prod and future\_cons |
| 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 is FUTURE\_EXCLUSIVE in our case. 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 a t a given time. |
| future\_free.c | This is a system call which de allocates memory assigned to a future. This system call deallocated memory assigned to a future only if future is in FUTURE\_EMPTY state. |
| 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 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 and the state is hanged 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. |
| 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.  When the state of the future is FUTURE\_VALID, a call to the function will result into an error as the mode is EXCLUSIVE.  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. |

**Responsibilities/Tasks Member ID**

Prodcons.h sshalabh

Future.h pandeyh

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