OS Assignment-4: Modified philosophers problem

Karanjot Singh
2019050

Semaphore:

Semaphores cant be created with mutexes without the help of condition variable.
Hence, A semaphore structure is defined and named my_semaphore. This structure contains attributes of type pthread_cond_t (count)and pthreaad_mutex_t (mutex).

The wait and signal functions of a semaphore are implemented with the help of :

1. pthread_cond_signal() wakes up exactly one thread among the threads that are blocked on the condition variable
2. pthread_cond_wait atomically unlocks the mutex (as per pthread_unlock_mutex) and waits for the condition variable cond to be signaled.

For blocking:
Mutex is locked by calling pthread_mutex_lock(). If the mutex is already locked, the calling thread shall block until the mutex becomes available.

For non-blocking:
The pthread_mutex_trylock() function attempts to acquire ownership of the mutex specified without blocking the calling thread. If the mutex is currently locked by another thread, the call to pthread_mutex_trylock() returns an error of EBUSY.

Philosopher's problem:

My_semaphore is used to implement the philosophers' problem, Each bowl is allotted a semaphore and the forks are also represented by an array of semaphores.
To avoid deadlocks, Only those threads which acquire both the forks can acquire one of the bowls.
Only after which a philosopher can eat. Also, the bowls are released in a reverse order than they were taken to avoid deadlocks.

References:

Man pages

Implemented as Assignment 4 in CSE231 - Operating Systems at IIIT Delhi