Jigar Makwana

Assignment 3

Purpose of the project:  
Purpose of this assignment is to simulate memory management unit of the operating system and show how first fit, best fit and worst fit algorithm.



Background of the assignment:

Mutex are used to synchronize the threads accessing the critical sections on the program. It has can be either 1 or 0. 1 if thread has lock to access the critical section where as 0 then it must wait for the lock. On other hand Semaphore can allow multiple threads to access the critical section. It can have more than two values unlike mutex. If it is 0, thread must wait for the semaphore, if it is >0 thread can take the semaphore to access the critical section. Also, thread can post the value to the semaphore after it done with the critical section. Memory management unit manages the memory blocks requests in a system according to the algorithms first fit, best fit and worst fit. It also takes care of the situation where it run out of memory blocks, in such situation it combines the small memory blocks available to make a larger block. And also it kicks out the lower priority thread to allocate memory to higher priority thread.

Algorithms/Functions used in assignment:

- Produce MMS and User Threads  
- User Threads requests the MMS thread for a memory  
- The MMS thread assigns the user thread a memory  
- The MMS thread follow algorithms such as first fit, best fit and worst fit to manage the memory  
- This operation occurs in an infinite loop

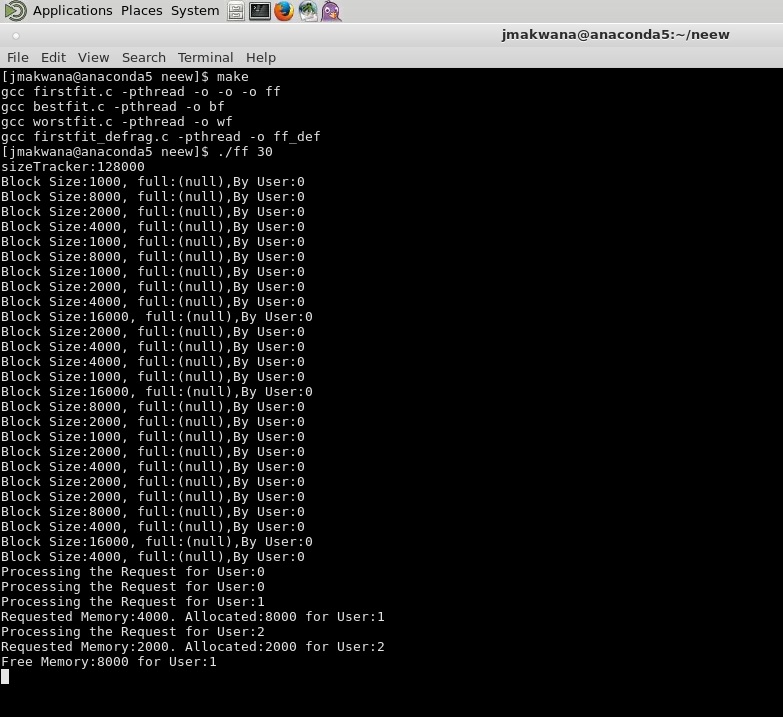
Functions:

pthread\_mutex\_init() - to initialize mutex sem\_init() – to initialize semaphore malloc() – to allocate memory pthread\_create() – to create pthreads pthread\_join() – to wait for threads to complete and join pthread\_mutex\_destroy() – destroy mutex sem\_destroy() – destroy semaphore sem\_wait() – wait on semaphore sem\_post () – post the semaphore pthread\_mutex\_lock() – mutex lock pthread\_mutex\_unlock() -mutex unlock

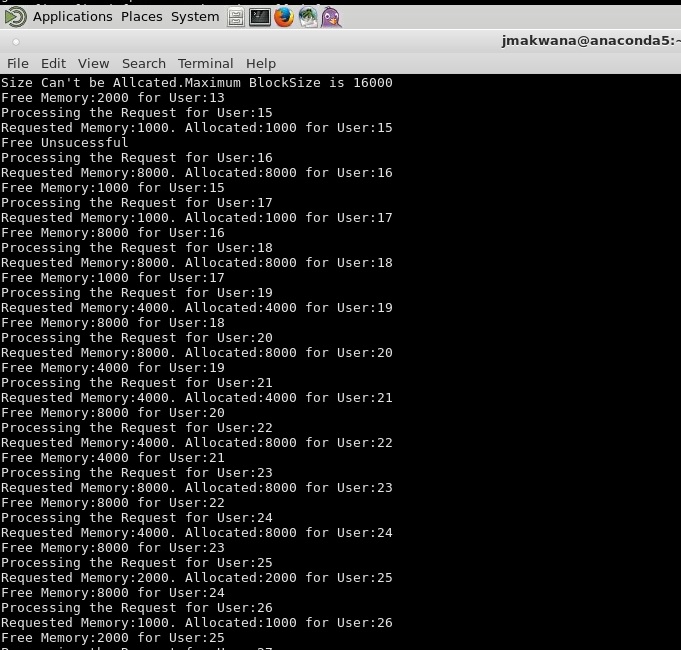
Userdefined functions: Manage() – to manage memory blocks Request() – to request MMS to allocate or free memory block memory\_malloc() – to place the request on to the request buffer Process\_Request() – to read the memory block request by the user from the request buffer memory\_free() – to place the free memory block request on to the free buffer Process\_Free() – to read the memory free request by the user from the free buffer Free\_mBlock() – to free the memory block in the linked list First\_Fit() – to assign memory blocks using first fit algorithm Best\_Fit() – to assign memory blocks using best fit algorithm Worst\_Fit() – to assign memory blocks using worst fit algorithm.

Result:

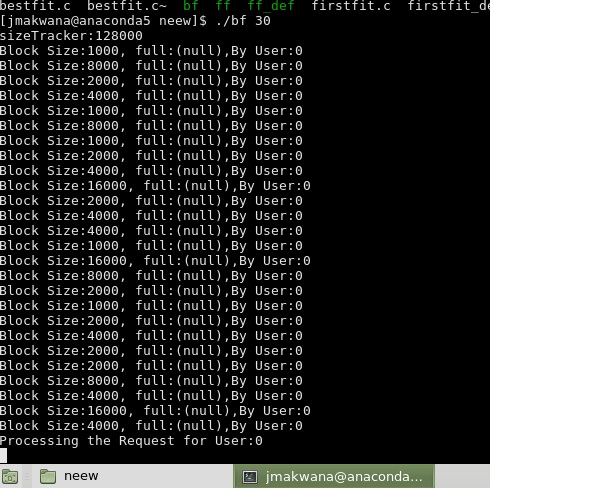
First fit with 30 user threads. You can see the memory blocks initialized during the start



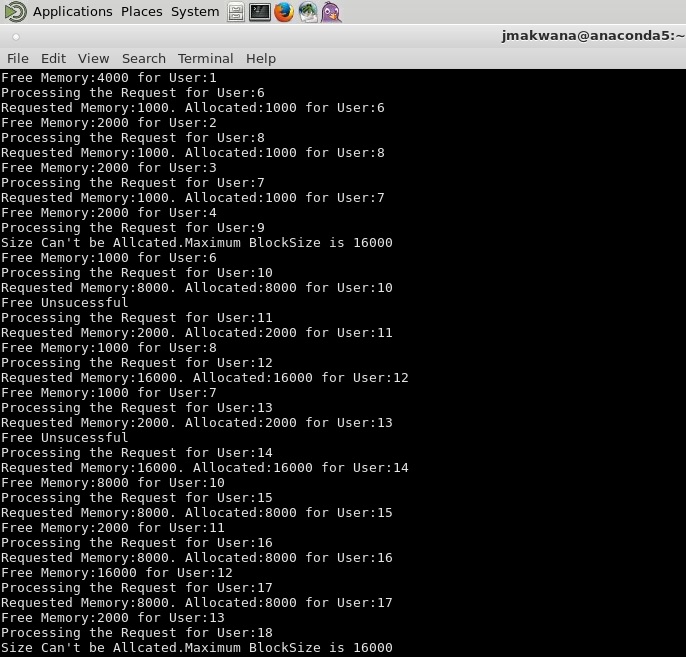
First fit: Below you can see the memory blocks assigned accordingly. Defragment is not implemented in this case.



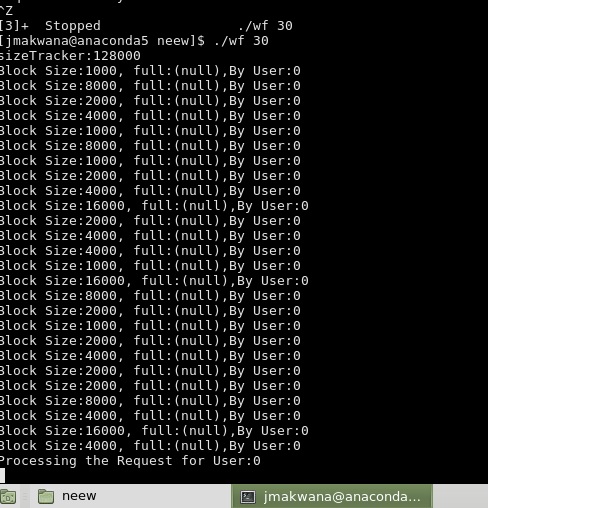
Best fit with 30 user threads. You can see the memory blocks initialized during the start



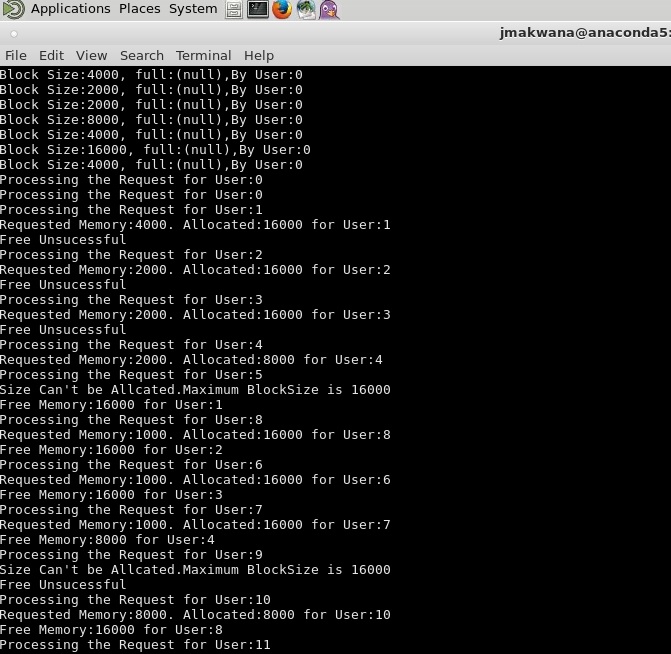
Best fit: Below you can see the memory blocks assigned accordingly. Defragment is not implemented in this case.



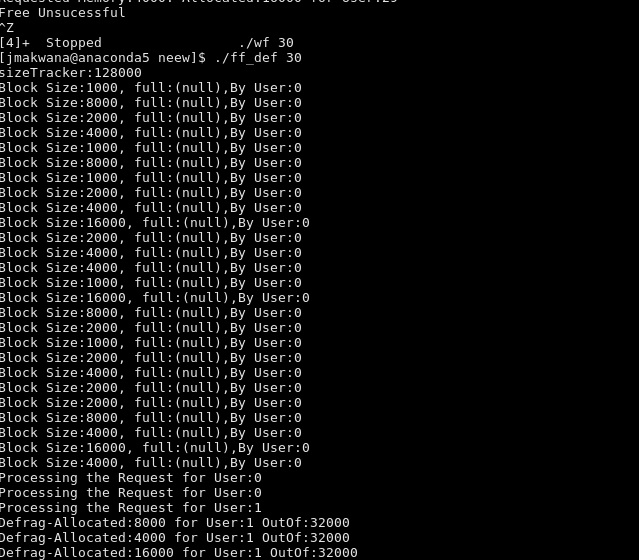
Worst fit with 30 user threads. You can see the memory blocks initialized during the start.



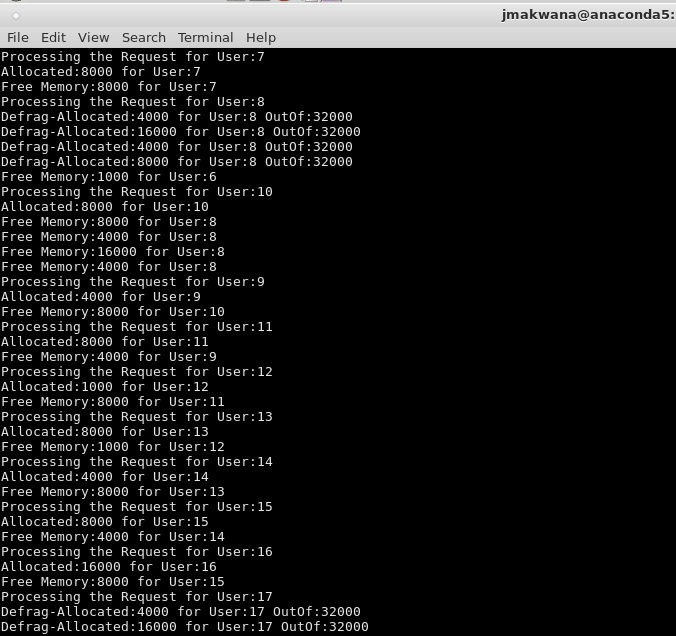
Worst fit: Below you can see the memory blocks assigned accordingly. Defragment is not implemented in this case.



First fit: Below you can see the memory blocks assigned accordingly. Defragment is implemented in this case. See the following three figs.



First fit: Below you can see the memory blocks assigned accordingly. Defragment is implemented in this case. You can see the kicking out the lower priority thread to get free memory block.



Observations:

Buffers are very useful in managing the communication between the threads. And Linked list best useful in this case to manage the memory blocks.

Conclusion:

Buffers are very useful in managing the communication between the threads. And Linked list best useful in this case to manage the memory blocks.

Source Code: