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

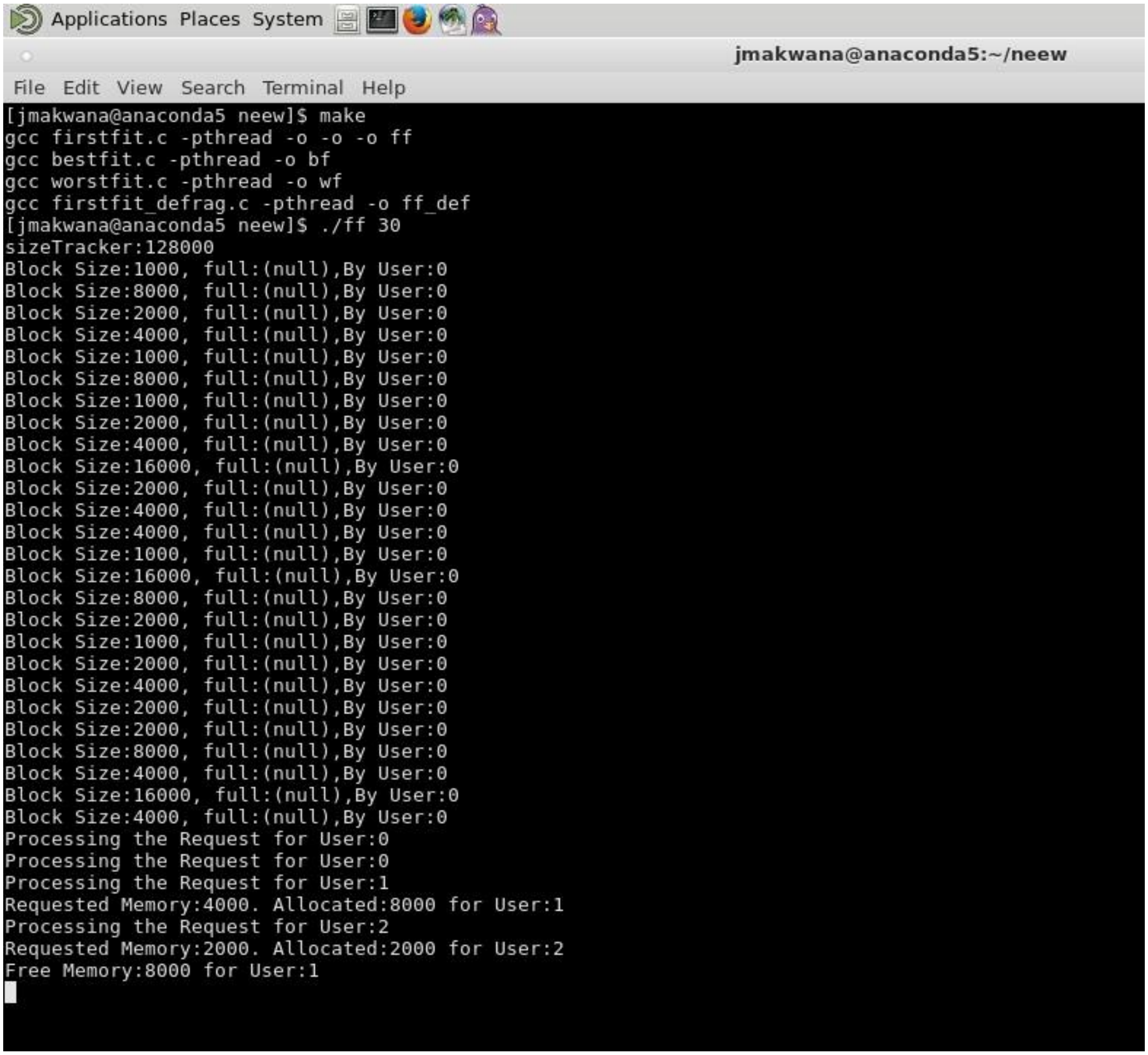
Functions:

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:

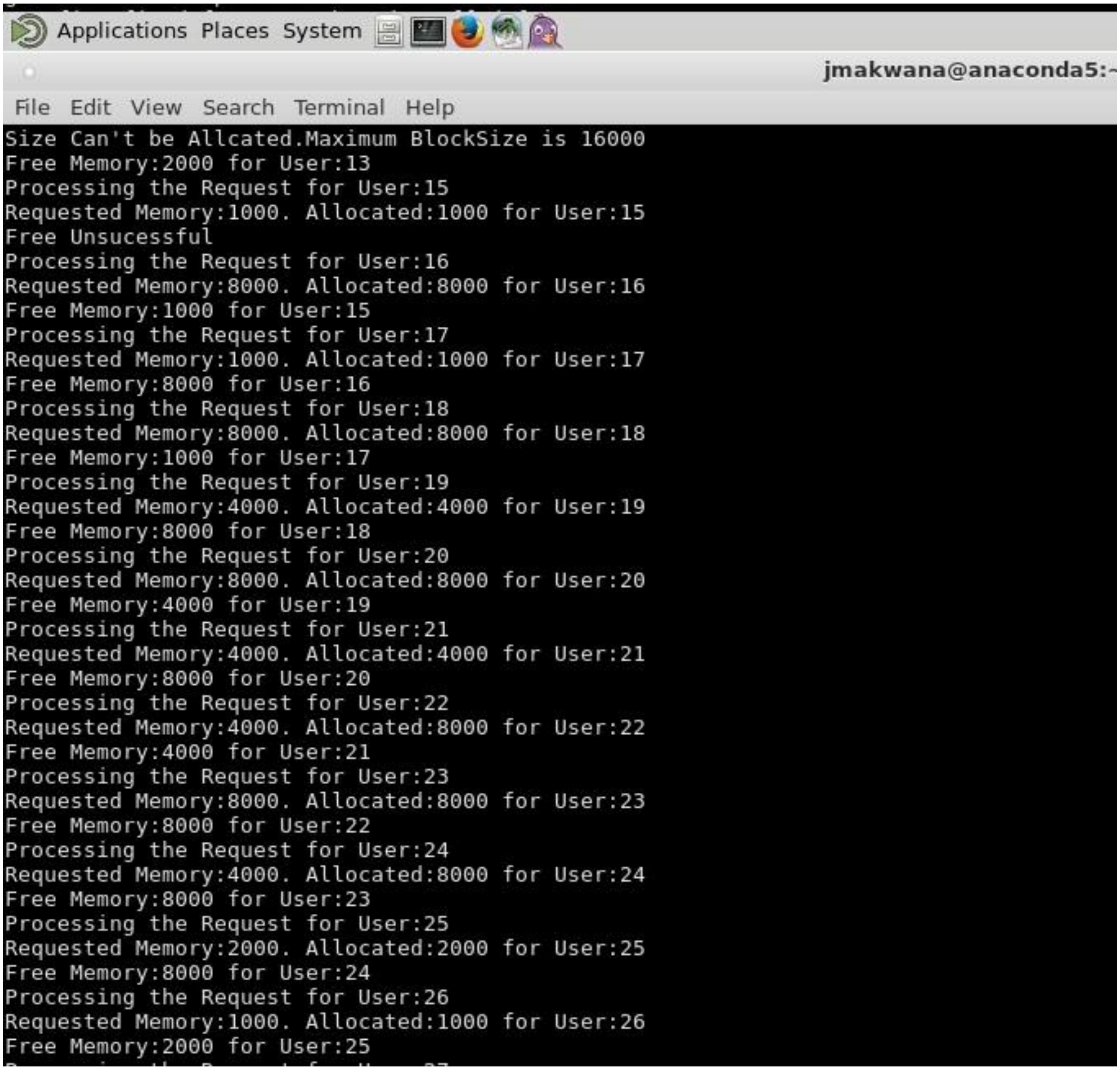
First fit with 30 user threads. You can see the memory blocks initialized during the start



Applications Places System
jmakwana@anaconda5:~/neew
File Edit View Search Terminal Help

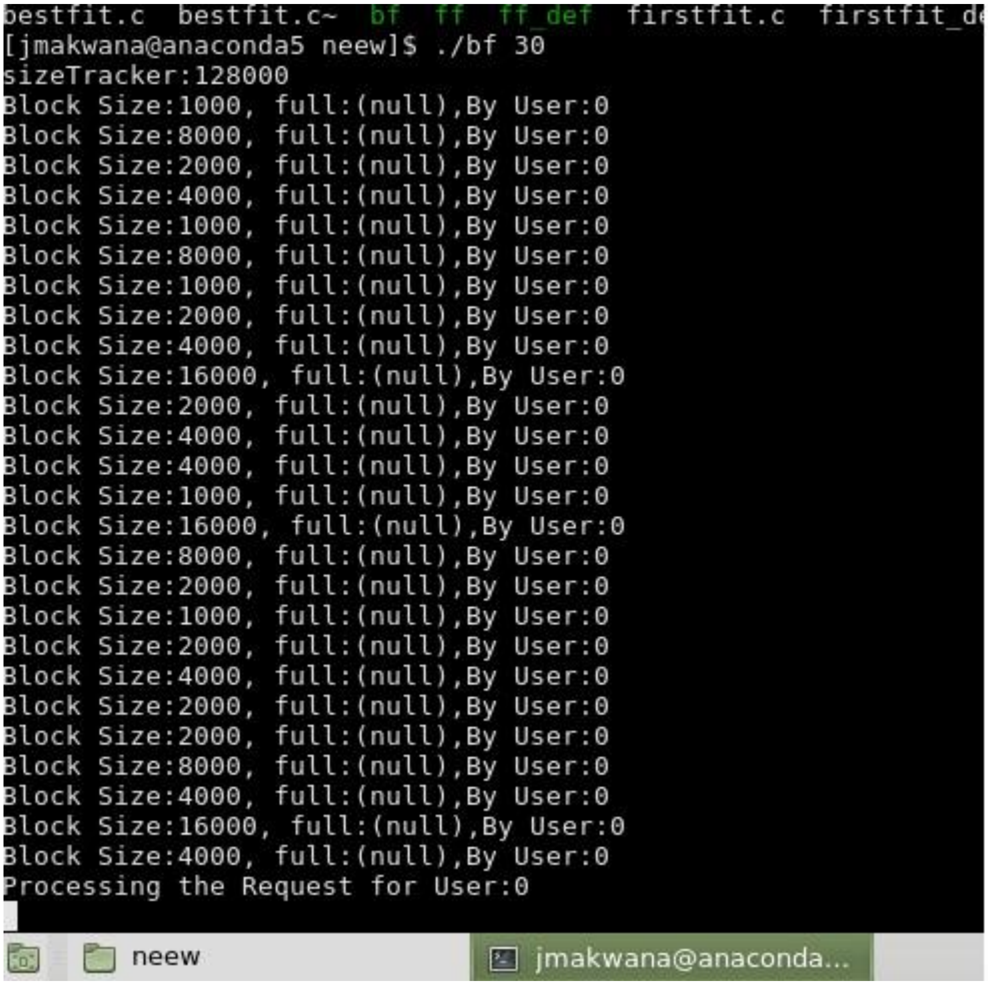
```
[jmakwana@anaconda5 neew]$ make
gcc firstfit.c -pthread -o -o -o ff
gcc bestfit.c -pthread -o bf
gcc worstfit.c -pthread -o wf
gcc firstfit_defrag.c -pthread -o ff_def
[jmakwana@anaconda5 neew]$ ./ff 30
sizeTracker:128000
Block Size:1000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:16000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:16000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:16000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Processing the Request for User:0
Processing the Request for User:0
Processing the Request for User:1
Requested Memory:4000. Allocated:8000 for User:1
Processing the Request for User:2
Requested Memory:2000. Allocated:2000 for User:2
Free Memory:8000 for User:1
█
```

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


Applications Places System
jmakwana@anaconda5:~
File Edit View Search Terminal Help

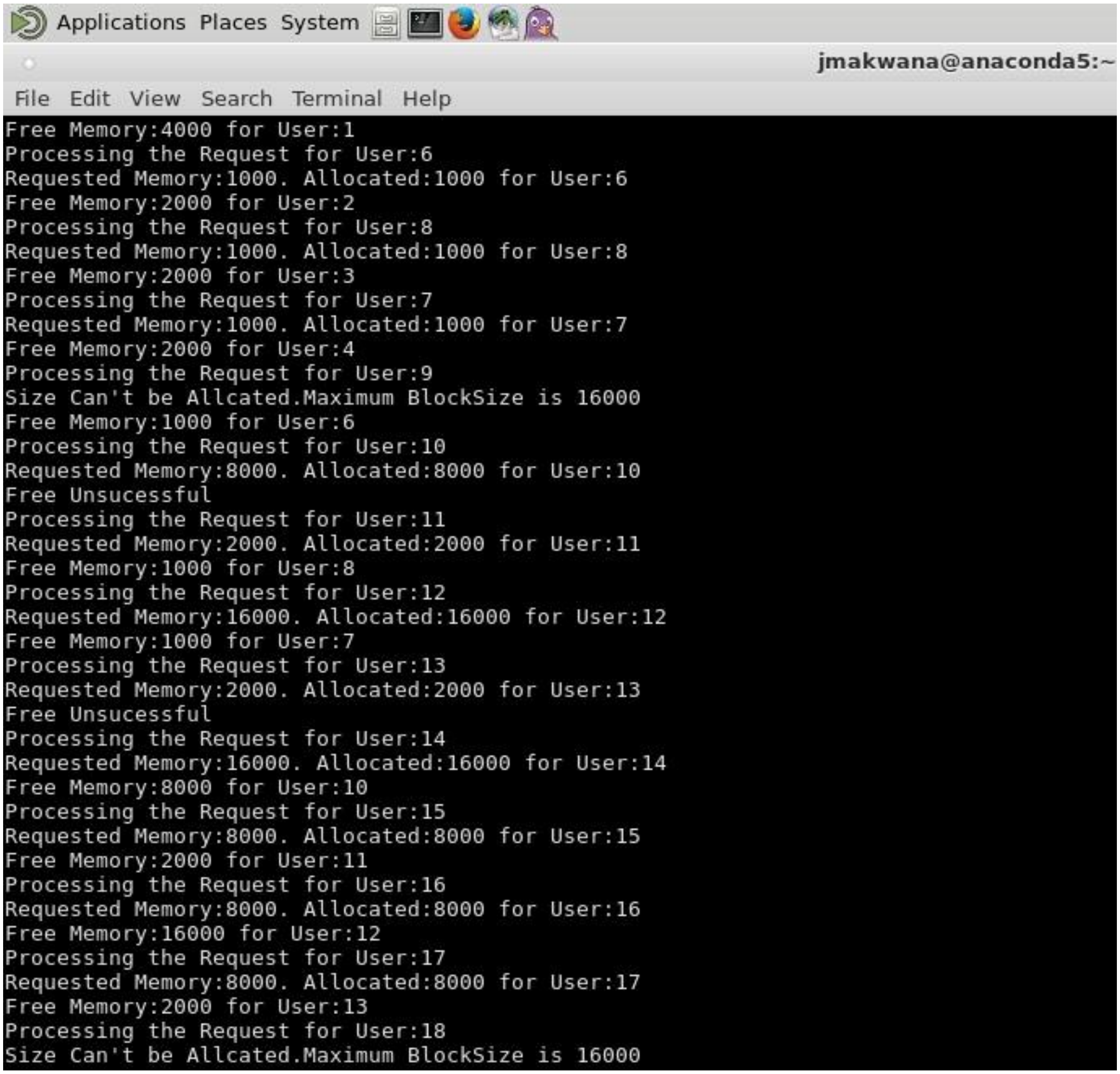
```
Size Can't be Allcated.Maximum BlockSize is 16000
Free Memory:2000 for User:13
Processing the Request for User:15
Requested Memory:1000. Allocated:1000 for User:15
Free Unsucessful
Processing the Request for User:16
Requested Memory:8000. Allocated:8000 for User:16
Free Memory:1000 for User:15
Processing the Request for User:17
Requested Memory:1000. Allocated:1000 for User:17
Free Memory:8000 for User:16
Processing the Request for User:18
Requested Memory:8000. Allocated:8000 for User:18
Free Memory:1000 for User:17
Processing the Request for User:19
Requested Memory:4000. Allocated:4000 for User:19
Free Memory:8000 for User:18
Processing the Request for User:20
Requested Memory:8000. Allocated:8000 for User:20
Free Memory:4000 for User:19
Processing the Request for User:21
Requested Memory:4000. Allocated:4000 for User:21
Free Memory:8000 for User:20
Processing the Request for User:22
Requested Memory:4000. Allocated:8000 for User:22
Free Memory:4000 for User:21
Processing the Request for User:23
Requested Memory:8000. Allocated:8000 for User:23
Free Memory:8000 for User:22
Processing the Request for User:24
Requested Memory:4000. Allocated:8000 for User:24
Free Memory:8000 for User:23
Processing the Request for User:25
Requested Memory:2000. Allocated:2000 for User:25
Free Memory:8000 for User:24
Processing the Request for User:26
Requested Memory:1000. Allocated:1000 for User:26
Free Memory:2000 for User:25
Processing the Request for User:27
```

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



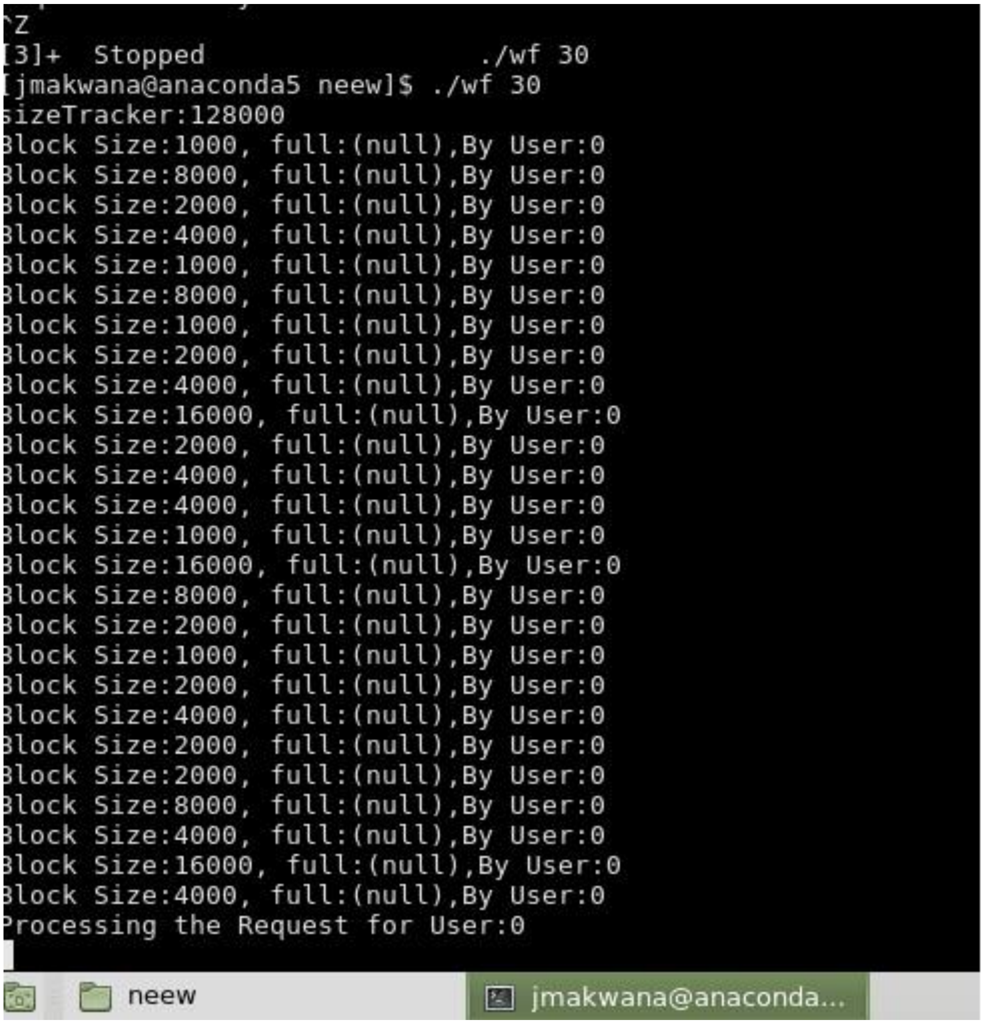
```
bestfit.c  bestfit.c~  bf  ff  ff_def  firstfit.c  firstfit_de
[jmakwana@anaconda5 neew]$ ./bf 30
sizeTracker:128000
Block Size:1000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:16000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:16000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:16000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Processing the Request for User:0
```

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


Applications Places System
jmakwana@anaconda5:~
File Edit View Search Terminal Help

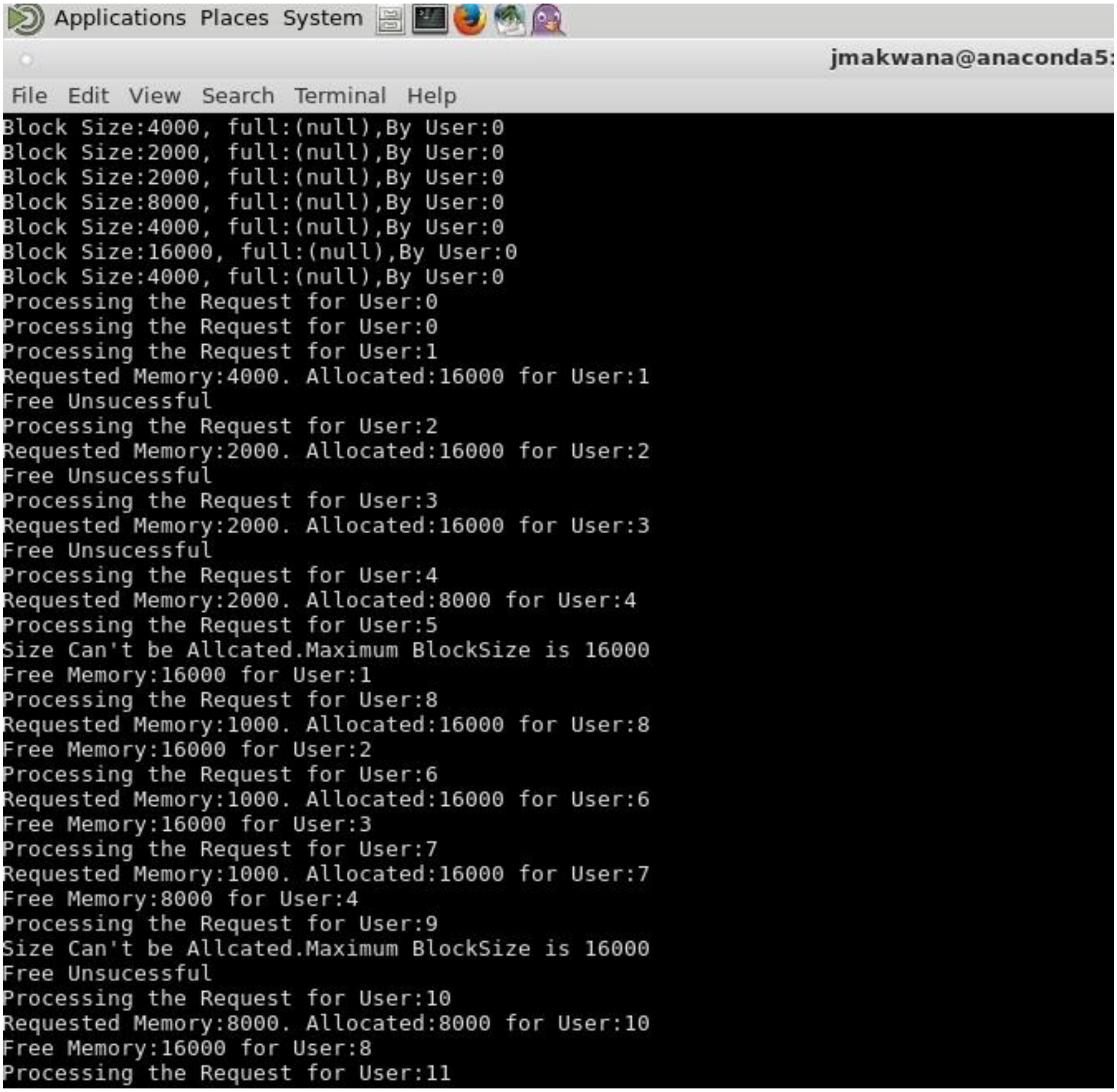
```
Free Memory:4000 for User:1
Processing the Request for User:6
Requested Memory:1000. Allocated:1000 for User:6
Free Memory:2000 for User:2
Processing the Request for User:8
Requested Memory:1000. Allocated:1000 for User:8
Free Memory:2000 for User:3
Processing the Request for User:7
Requested Memory:1000. Allocated:1000 for User:7
Free Memory:2000 for User:4
Processing the Request for User:9
Size Can't be Allcated.Maximum BlockSize is 16000
Free Memory:1000 for User:6
Processing the Request for User:10
Requested Memory:8000. Allocated:8000 for User:10
Free Unsucessful
Processing the Request for User:11
Requested Memory:2000. Allocated:2000 for User:11
Free Memory:1000 for User:8
Processing the Request for User:12
Requested Memory:16000. Allocated:16000 for User:12
Free Memory:1000 for User:7
Processing the Request for User:13
Requested Memory:2000. Allocated:2000 for User:13
Free Unsucessful
Processing the Request for User:14
Requested Memory:16000. Allocated:16000 for User:14
Free Memory:8000 for User:10
Processing the Request for User:15
Requested Memory:8000. Allocated:8000 for User:15
Free Memory:2000 for User:11
Processing the Request for User:16
Requested Memory:8000. Allocated:8000 for User:16
Free Memory:16000 for User:12
Processing the Request for User:17
Requested Memory:8000. Allocated:8000 for User:17
Free Memory:2000 for User:13
Processing the Request for User:18
Size Can't be Allcated.Maximum BlockSize is 16000
```

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



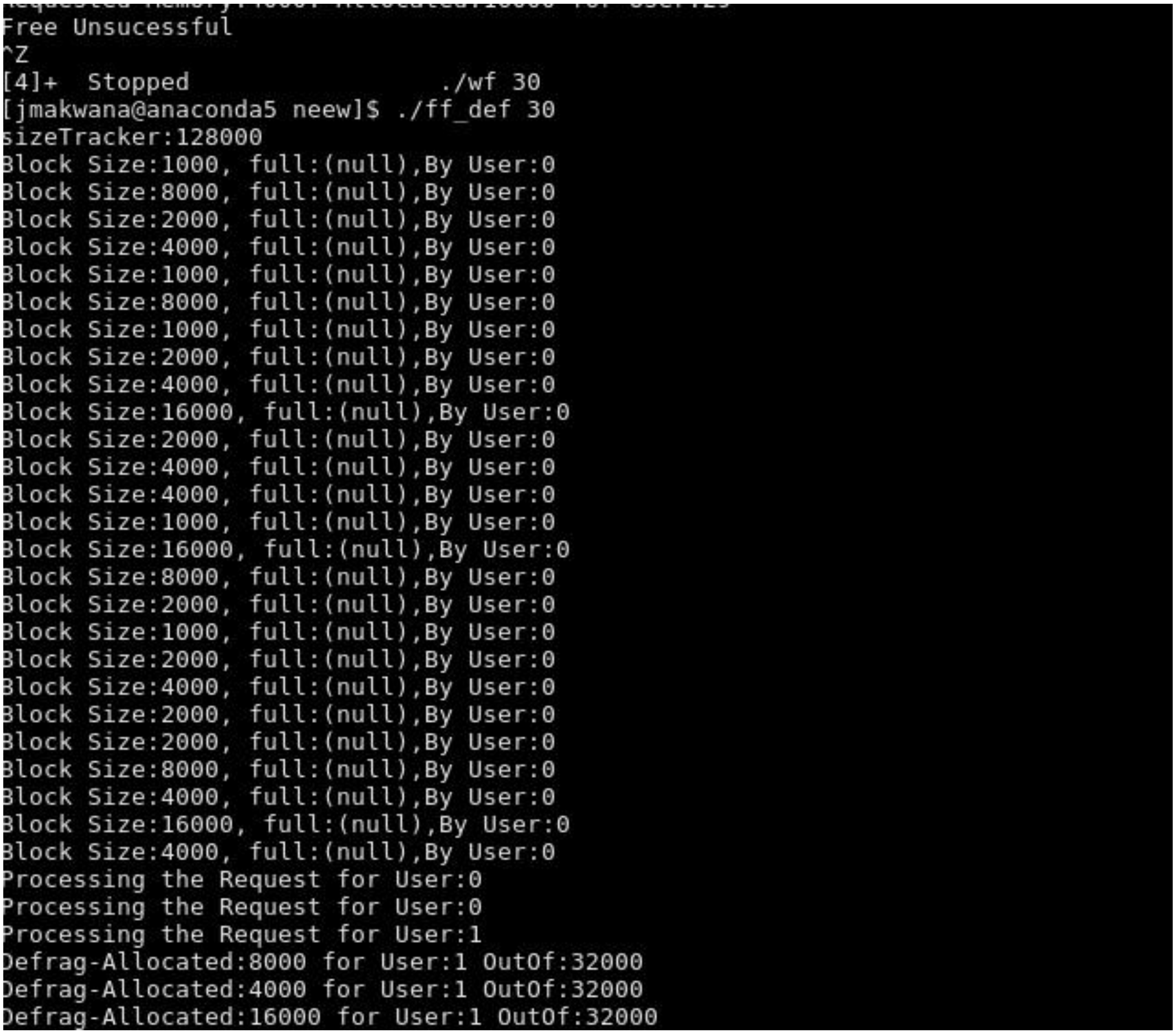
```
^Z
[3]+  Stopped                 ./wf 30
[jmakwana@anaconda5 neew]$ ./wf 30
sizeTracker:128000
Block Size:1000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:16000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:16000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:16000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Processing the Request for User:0
```

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



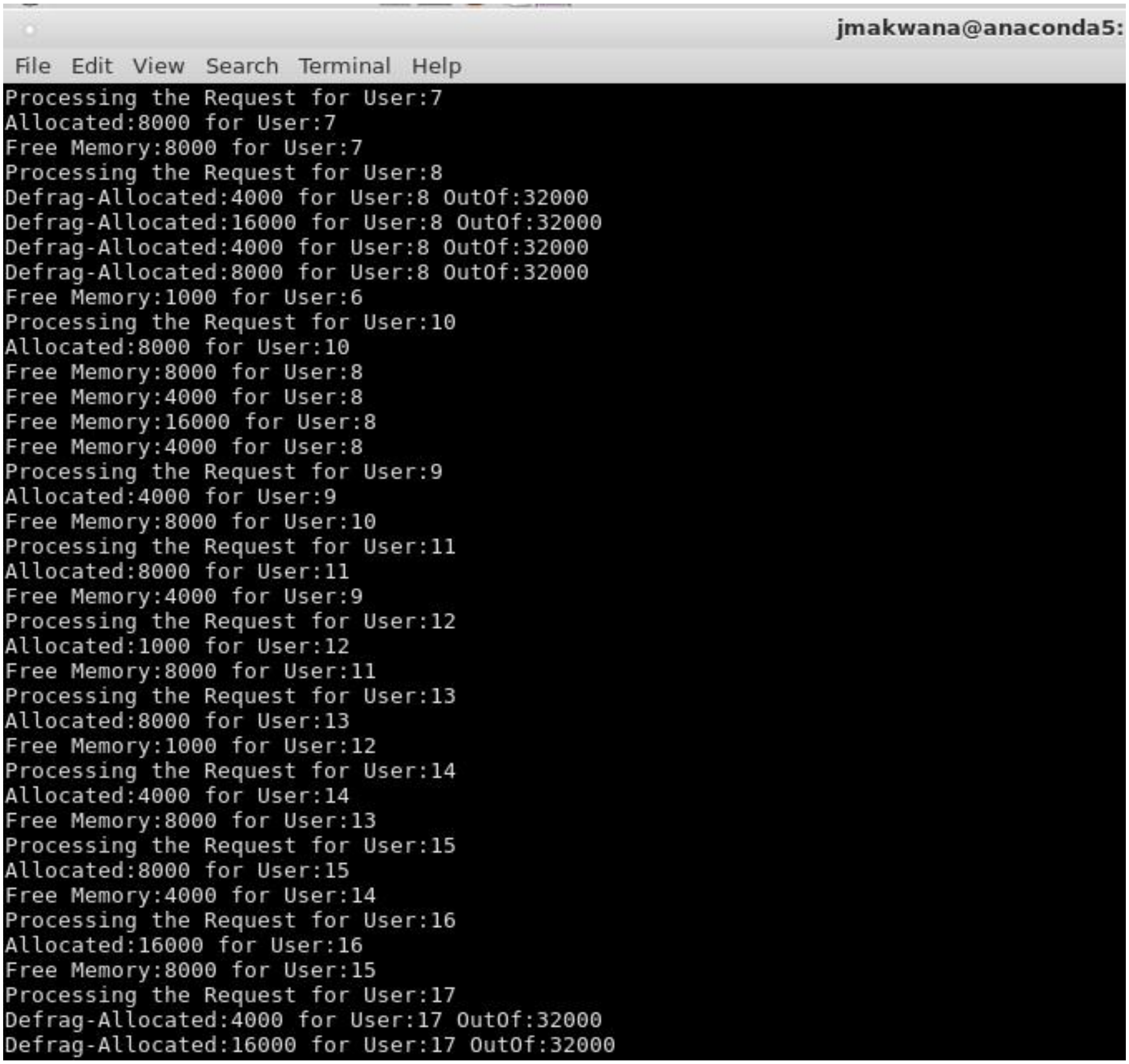
```
Block Size:4000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:16000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Processing the Request for User:0
Processing the Request for User:0
Processing the Request for User:1
Requested Memory:4000. Allocated:16000 for User:1
Free Unsucessful
Processing the Request for User:2
Requested Memory:2000. Allocated:16000 for User:2
Free Unsucessful
Processing the Request for User:3
Requested Memory:2000. Allocated:16000 for User:3
Free Unsucessful
Processing the Request for User:4
Requested Memory:2000. Allocated:8000 for User:4
Processing the Request for User:5
Size Can't be Allcated.Maximum BlockSize is 16000
Free Memory:16000 for User:1
Processing the Request for User:8
Requested Memory:1000. Allocated:16000 for User:8
Free Memory:16000 for User:2
Processing the Request for User:6
Requested Memory:1000. Allocated:16000 for User:6
Free Memory:16000 for User:3
Processing the Request for User:7
Requested Memory:1000. Allocated:16000 for User:7
Free Memory:8000 for User:4
Processing the Request for User:9
Size Can't be Allcated.Maximum BlockSize is 16000
Free Unsucessful
Processing the Request for User:10
Requested Memory:8000. Allocated:8000 for User:10
Free Memory:16000 for User:8
Processing the Request for User:11
```

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



```
Requested Memory:4000: Allocated:16000 for User:29
Free Unsucessful
^Z
[4]+  Stopped                 ./wf 30
[jmakwana@anaconda5 neew]$ ./ff_def 30
sizeTracker:128000
Block Size:1000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:16000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:16000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:1000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:2000, full:(null),By User:0
Block Size:8000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Block Size:16000, full:(null),By User:0
Block Size:4000, full:(null),By User:0
Processing the Request for User:0
Processing the Request for User:0
Processing the Request for User:1
Defrag-Allocated:8000 for User:1 OutOf:32000
Defrag-Allocated:4000 for User:1 OutOf:32000
Defrag-Allocated:16000 for User:1 OutOf:32000
```

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.



```
Processing the Request for User:7
Allocated:8000 for User:7
Free Memory:8000 for User:7
Processing the Request for User:8
Defrag-Allocated:4000 for User:8 OutOf:32000
Defrag-Allocated:16000 for User:8 OutOf:32000
Defrag-Allocated:4000 for User:8 OutOf:32000
Defrag-Allocated:8000 for User:8 OutOf:32000
Free Memory:1000 for User:6
Processing the Request for User:10
Allocated:8000 for User:10
Free Memory:8000 for User:8
Free Memory:4000 for User:8
Free Memory:16000 for User:8
Free Memory:4000 for User:8
Processing the Request for User:9
Allocated:4000 for User:9
Free Memory:8000 for User:10
Processing the Request for User:11
Allocated:8000 for User:11
Free Memory:4000 for User:9
Processing the Request for User:12
Allocated:1000 for User:12
Free Memory:8000 for User:11
Processing the Request for User:13
Allocated:8000 for User:13
Free Memory:1000 for User:12
Processing the Request for User:14
Allocated:4000 for User:14
Free Memory:8000 for User:13
Processing the Request for User:15
Allocated:8000 for User:15
Free Memory:4000 for User:14
Processing the Request for User:16
Allocated:16000 for User:16
Free Memory:8000 for User:15
Processing the Request for User:17
Defrag-Allocated:4000 for User:17 OutOf:32000
Defrag-Allocated:16000 for User:17 OutOf:32000
```

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: