CSc 360: Operating Systems (Summer 2012)
Assignment 2: Multi-Flow Scheduler (MFS) - Deliverable 1

By Derek Roberts (v00698880)

1. Threads will total the number of flows plus one for a controller.
2. They will not work independently and require a controller thread for transmission.
3. There will be a mutex for protecting the total number of flows in the waiting list and possibly another for protecting the total number of remaining threads. Although it does not reflect reality for a router the program may end when all flows have been processed.
4. The main thread will not be idle since it contains the controller and will constantly be waiting for flows to arrive and be prioritized or to finish transmission.
5. Flows will be represented by a single thread representing the flow in transmission. A wait or similar command can chew up time in that thread, leaving the producer and consumer active. Flows will be referenced by the controller in an array that contains the flow's address and details for deciding order of transmission.
6. A mutex will protect the total number of flows waiting for transmission, not including those that have not arrived. Depending on implementation another mutex may be required to prevent the token passed to the flow that's in transmission.
7. (a) Convars will be used to track whether a flow has arrived. Since transmitted flows' threads will end a convar is not required to show that. Another convar may be used to pass the token back from the terminating flow to the controller.
   (b) A mutex in the controller will prevent the waiting flow count from being reduced while other flows are being added in. Otherwise the queue may appear empty.
   (c) The controller should look at all arrived flows, including those that arrived while waiting on the last flow to transmit, and the next transmission chosen.
8. Algorithm

- Read input file
- Create a thread for every single flow
  - All flows will be unable to request transmission until their arrival times
  - All times calculated against application-set launch time and arrive accordingly
  - Upon arrival flows will pass their availability to a controller thread and wait
    - Availability will include that number with all the data from txt file
  - When flow is eventually told to transmit use usleep to kill time
  - Tell controller transmission is complete, release mutex/token and end thread
- Use a controller thread that looks at arrived flows and passes a mutex/token to transmit
  - Take arrived flows and create a priority queue based on assignment criteria
  - Tell one flow to transmit and allow it to hold the token until transmission ends
  - The total number of threads in the waiting list will be protected with a mutex
- Run until all flows have arrived and transmitted or when process interrupted
  - In real world scenario the controller would keep running, since flows not static