Project Description:

1. There are two queues: ready and wait. The ready queue is a queue for the CPU and the wait queue is the queue for the disk. We have one CPU and one disk; one process is served at a time.
2. The queues are implemented as single linked lists (i.e., append nodes at the end, remove nodes from the front) of PCB representing the corresponding processes.
3. We will be using the round robin algorithm in the scheduling pipeline.

4. PCB structure: int state, int process id /pid (from 1 to 10 as we have 10 processes), int program\_counter (this will point to the line number that the pid is currently working on), ptr memory info which will point to the array for the memory management.

5. For each process, we will read in a file which is 100 lines long. One line contains 10 characters. 1 page consists of 8 lines (80 characters), and each process will need multiple pages / frames to read in the whole file.

6. There will be two time slices defined in the system: 1) time slice in the CPU will range from 1 to 5 time units (since we are simulating the CPU, this is simply idling in a loop); 2) time slice in the disk will range from 5 to 8 units (8 units allows to read in the whole page; less than 8 units will lead to internal fragmentation due to the process being kicked-out before finishing the whole page). Given the time slice for the disk can be as low as 50 time units, each process may need up to 100/5=20 pages (worst case) and 100/8=13 pages (best case) to read in the whole file.

7. CPU clock is an incrementing integer.

8. Memory management. We will use an array of pointers 10x1 where each cell point to the head of a linked list containing nodes for one process with the following information in Memory structure: 1) pid, 2) page number, 3) the number of read in characters in the page (this will be useful in cases when the process was kicked-out due to the time slice expiration in the disk, and we need to complete the read-in of the data); 3) string representing the read in data for that frame; 4) cpu time when the current frame was started to be read into the memory. Array cell 0 contains head to the nodes for the list of process 1; cell 1, points to the head of the list for process 2, etc.

9. In the process management, the process state integer representation (in the PCB) is aligned with the following pipeline:

0. Process is newly created and is assigned to the ready queue;

1. Waiting in the ready queue;

2. On route to or in CPU;

3. Left the CPU, checking if it is done (if done, go to 9; else go to 4)

4. Check if it needs to use the disk. If yes, route the wait queue (go to 5), else to the ready queue (go to 1);

5. Waiting in the wait queue;

6. Left the wait queue and en route to the disk;

7. In disk and reading in a file;

8. Left the disk and en route back to the ready queue;

9. The process is done.

10. Definition of done: the process is done when all 100 lines were read in. In main(), we will stop after all 10 processes are done.

11. Files. The data in the files should be preferably such that when it is printed we can visually examine the output and find out which process (file) and line it is associated with. We have 10 processes and 100 lines. Let each line be represented with an integer and each process can be associated with an alphabetic letter: A for process 1, B for process 2, … J for process 10.

For example for process 1, the file should be (10 characters):

For line 1, the 10 characters are: A1A1A1A1A1

For line 2, the 10 characters are: A2A2A2A2A2

…

For line 100, the 10 characters are: A100A100A1

For example for process 10, the file should be (10 characters):

For line 1, the 10 characters are: J1J1J1J1J1

For line 2, the 10 characters are: J2J2J2J2J2

…

For line 100, the 10 characters are: J100J100J1

12. From the main(), we should print the following as we the program executes:

1. Print instantaneous information (from the PCB) for each process,time such as:

CPU time | pid1 | pid2 | pid3 | pid4 | pid5 | pid6 | pid7 | pid8 | pid9 | pid10

1 3 2 1 1 1 1 1 1 1 1 1

….

N 9 9 9 9 9 9 9 9 9 9 9

This information should also be written to a text file so that it can be analyzed and visualized (separately).

b) Upon the completion, we can traverse the memory list for each process and print the contents of what was read in and when. For example,

For process 1, we read in the following:

CPU time. | Frame | Read in string.

N 1 A1A1A1A1A1… A8A8A8A8A8

….

N 1 A95A95A95A…… A100A100A1

13. Program Skeleton: We have two linked lists and one hash table. We will use the functions for inserting and removing from a linked list as well as two structures (PCB in 4 and Memory in 8). We can start with the state machine describing the process management; the memory piece will branch off from the step 7 in the process management.