1. Processes management
   1. Create processes
      1. Each process will have a unique ID
      2. Users will define the process priority
      3. Processes can access shared memory
         1. The user can specify when the process accesses the shared memory (when the process enters its critical section)
      4. The user can specify when the process accesses an IO channel
      5. Assign memory to the process
      6. Users should be able to specify how many instructions are in the process
         1. The process can keep on iterating, or just stop after finishing the instructions
   2. Schedule processes
      1. Be able to schedule processes to multiple CPUs
      2. The user can choose a scheduling algorithm
         1. Round-robin, preemptive, …
   3. Processes must have a state (ready, running, blocked)
      1. When a process is running, it should contain the CPU ID it is running on
      2. When a process is blocked, it should contain the reason of its block
   4. Users can kill processes
   5. Processes can have child processes
      1. If a parent process is killed, the children will die too, unless specified
2. Memory management
   1. The simulated system can contain shared resources
      1. Should be able to specify a way of achieving mutual exclusion
         1. A shared resource can have a semaphore
   2. Virtual memory simulation
      1. Simulate frames in the physical memory and secondary memory
      2. Users can choose a page swapping strategy (FIFO, LRU)
         1. Swapping’s overhead should be simulated
            1. Strategies can have varied overheads

E.g. LRU runs slower than FIFO

* + - * 1. The overhead will be in the kernel process

1. Hardware management
   1. Users can create any number of CPUs
      1. Each CPU should have a unique ID
      2. Each CPU has a clock speed, defined in terms of the whole simulation clock
   2. Users can specify the size of physical memory
      1. Physical memory might have a delay
   3. Users can specify the size and speed of secondary memory
      1. Speed is defined in terms of simulation clock
   4. Users can create many IO channels
      1. Each IO will have its own delay, defined in terms of the simulation clock
2. Simulation
   1. The system should visually represent its state
      1. Visually represent every CPU in the simulated system
         1. Show which process is currently in the CPU
         2. The system should show when the kernel is in CPU
      2. Visually represent memory, physical and secondary
         1. Specify the contents of the memory and its ownership
   2. The system should have a text log of the OS events
      1. The log should have a timestamp for each event
         1. The timestamp should be in terms of the simulation clock
      2. Users should be able to save the log
   3. Users can initiate the software from a file
      1. Users can save the current system to a file
      2. The file should be simple enough to be created by hand if needed
   4. Each CPU should have its own clock, and the whole simulation software should have a clock of its own
      1. Users should be able to “step” the environment clock, or just make it run automatically
         1. Users can enter a specific simulation time, and the simulation will jump to that
   5. The system should create a Gant chart of the processes and which one runs on which CPU
      1. Users can save the Gant chart
      2. The Gant chart will be updated with each environment clock tick