1. It continues representing the currently running process, as was the case with the basic manager version. Thus, typing any of the commands introduced in Section 2.6 will have the same effect as before. In addition, a process is now able to issue an additional call, *Request_IO()*.
2. The terminal also represents the hardware. Notably, the user is able to emulate the generation of the two types of interrupts—timeout and I/O completion—by triggering the corresponding functions of the manager.

To incorporate the above extensions into the shell, three new commands must be provided to invoke the functions *Request_IO()*, *IO_competion()*, and *Timeout()*. None of these functions require any parameters.

4 SUMMARY OF SPECIFIC TASKS

1. Design and implement the process and resource manager; the basic version includes the functions *Create()*, *Destroy()*, *Suspend()*, *Activate()*, *Request()*, and *Release()*, together with the underlying data structures; the extended version provides the additional functions *Request_IO()*, *IO_completion()*, and *Timeout()*.
2. Define a command language for the presentation shell. Then design and implement the shell so that you can test and demonstrate the functionality of your process and resource manager. For each command, the shell should respond by displaying the name of the currently running process, and any errors and other relevant system events that may have occurred.
3. Instantiate the manager to include the following at start-up:
   - **(a)** A RL with at least three priorities.
   - **(b)** A single process, called *Init*, that runs at the lowest-priority level and is always ready to run. This may create other processes at higher levels.
   - **(c)** At least three fixed resources, say A, B, C, that processes can request and release.
   - **(d)** An IO resource that processes may request and become blocked on until the next I/O completion interrupt.
4. Test the manager using a variety of command sequences to explore all aspects of its behavior. Demonstrate also a deadlock situation.

5 IDEAS FOR ADDITIONAL TASKS

1. Extend the concept of a resource to include multiple identical units. That is, each RCB represents a *class* of resources. Instead of the simple *Status* field, it has an "*inventory*" field that keeps track of which or how many units of the given resource are currently available. A process then may specify how many units of a resource class it needs. The request is granted when enough units are available; otherwise, the process is blocked. Similarly, a process may release some or all of the units it has previously acquired. This may now potentially unblock multiple processes at once.
2. Incorporate main memory as another resource class in the process and resource manager. That means, define a RCB for memory. The *Status* field is replaced by a