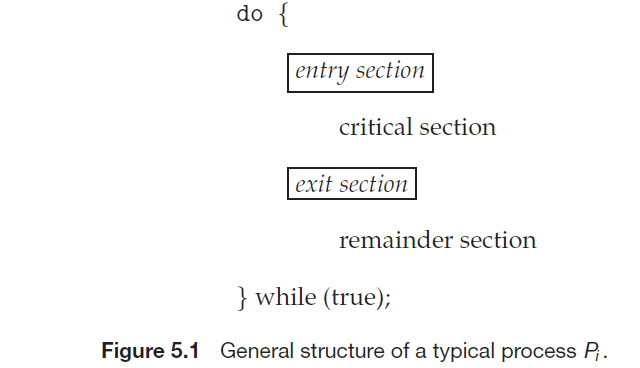
**CRITICAL SECTION PROBLEM**

Consider a system consisting of *n* processes {*P*0, *P*1, ..., *Pn*−1}. Each process has a segment of code, called a **critical section**, in which the process may be changing common variables, updating a table, writing a file, and so on. when one process is executing in its critical section, no other process is allowed to execute in its critical section. That is, no two processes are executing in their critical sections at the same time.

The ***critical-section problem*** is to design a protocol that the processes can use to cooperate. Each process must request permission to enter its critical section. The section of code implementing this request is the **entry section**. The critical section may be followed by an **exit** **section**. The remaining code is the **remainder section**. The general structure of a typical process *Pi* is shown in below Figure.



A solution to the critical-section problem must satisfy the following three requirements:

**1. Mutual exclusion:** If process Pi is executing in its critical section, then no other processes can be executing in their critical sections.

**2. Progress:** If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder sections can participate in deciding which will enter its critical section next, and this selection cannot be postponed indefinitely.

**3. Bounded waiting:** There exists a bound, or limit, on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.