```
processType processDescriptor[maxProcs];
int executing[maxProcs];   # one entry per processor
declarations of free, ready, and waiting lists and their locks;

SVC_Handler: {
  # entered with interrupts inhibited on processor i
  save state of executing[i];
  determine which primitive was invoked, then call it;
}

Timer_Handler: {
  # entered with interrupts inhibited on processor i
  lock ready list; insert executing[i] at end; unlock ready list;
  executing[i] = 0;
  dispatcher();
}

procedure fork(initial process state) {
  lock free list; remove a descriptor; unlock free list;
  initialize the descriptor;
  lock ready list; insert descriptor at end; unlock ready list;
  dispatcher();
}

procedure quit() {
  lock free list; insert executing[i] at end; unlock free list;
  record that executing[i] has quit; executing[i] = 0;
  if (parent process is waiting) {
    lock waiting list; remove parent from that list; unlock waiting list;
    lock ready list; put parent on ready list; unlock ready list;
  }
  dispatcher();
}

procedure join(name of child process) {
  if (child has already quit)
    return;
  lock waiting list; put executing[i] on that list; unlock waiting list;
  dispatcher();
}
```

```
procedure dispatcher() {
  if (executing[i] == 0) {
    lock ready list;
    if (ready list not empty) {
      remove descriptor from ready list;
      set executing[i] to point to it;
    }
    else    # ready list is empty
      set executing[i] to point to Idle process;
    unlock ready list;
  }
  if (executing[i] is not the Idle process)
    start timer on processor i;
  load state of executing[i];    # with interrupts enabled
}
```

Figure 6.4 Outline of a kernel for a shared-memory multiprocessor.