```
bool free = true;
sem e = 1, b[n] = ([n] 0);  # for entry and delay
typedef Pairs = set of (int, int);
Pairs pairs = ∅;
## SJN:  pairs is an ordered set ∧ free ⇒ (pairs == ∅)
request(time,id):
  P(e);
  if (!free) {
    insert (time,id) in pairs;
    V(e);           # release entry lock
    P(b[id]);       # wait to be awakened
  }
  free = false;
  V(e);    # optimized since free is false here
release():
  P(e);
  free = true;
  if (P != ∅) {
    remove first pair (time,id) from pairs;
    V(b[id]);  # pass baton to process id
  }
  else V(e);
```

Figure 4.14 Shortest-job-next allocation using semaphores.