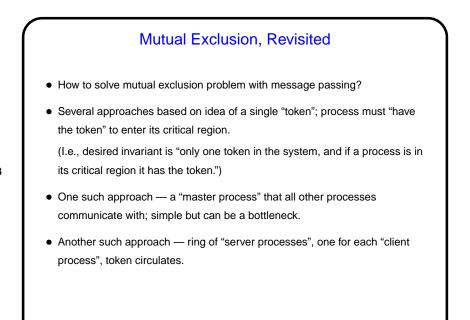


Message Passing as a Synchronization Mechanism — Recap

• Previous synchronization mechanisms all involve shared variables; okay in some circumstances but not very feasible in others (e.g., multiple-processor system without shared memory).

Slide 2

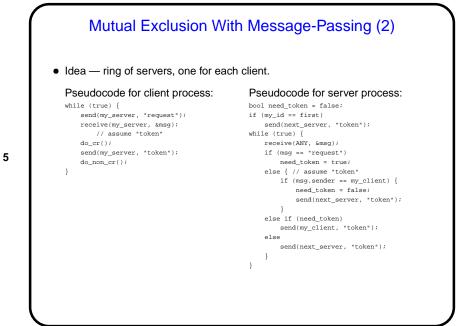
- Idea of message passing each process has a unique ID; two basic operations:
 - Send specify destination ID, data to send (message).
 - Receive specify source ID, buffer to hold received data. Usually some way to let source ID be "any".



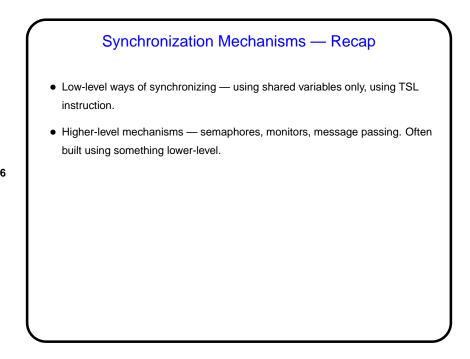
Slide 3

Mutual Exclusion With Message-Passing (1) • Idea — have "master process" (centralized control). Pseudocode for master process: Pseudocode for client process: while (true) { bool have_token = true; send(master, "request"); queue waitQ; receive(master, frequest),
 receive(master, &msg);
 // assume "token" while (true) { receive(ANY, &msg); if (msg == "request") { do_cr(); send(master, "token"); if (have_token) { send(msg.sender, "token");
have_token = false; do_non_cr(); else enqueue(sender, waitQ); else { // assume "token"
 if (empty(waitQ)) have_token = true; else { p = dequeue(waitQ); send(p, "token"); } } }

Slide 4



Slide 5



Slide 6

