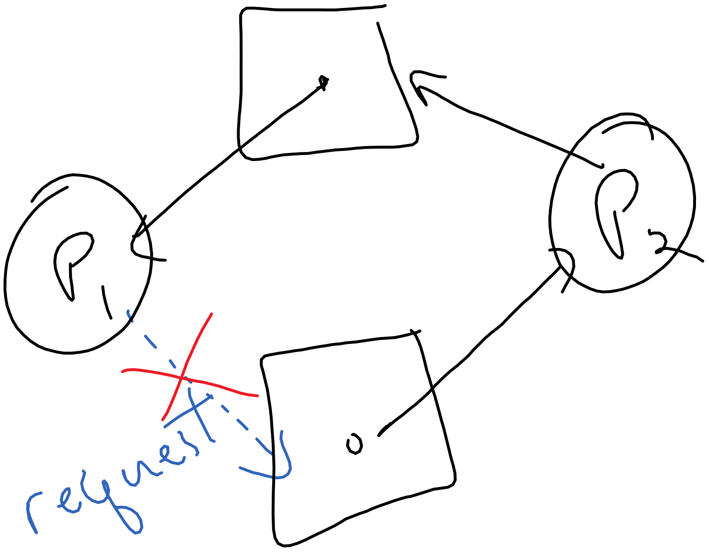
Thursday, March 17, 2016
1:09 PM

Deadlock Avoidance

- safe state - no deadlock
  unsafe - deadlock could occur

Thursday, March 17, 2016
1:18 PM

Single instance of each resource



Thursday, March 17, 2016
1:23 PM

Banker's Algorithm

Data:

int Available[ ] – for each res.

int max [i][j] - max res. of type j used by proc. i

int allocation [i][j] - # of res. j alloc to proc i

int need[ ][ ] = max - allocation

Thursday, March 17, 2016
1:31 PM

Safety Algorithm

boolean finish[ ] – for each proc. (false)

work[ ] = available

```
- find process i s.t.
      finish[i] == false
  and need[i][j] ≤ work[j]
              for all j
  if no such i, safe if all are
              finished, otherwise unsafe
  else
      finish[i] = true
      work[j] += allocation[i][j]
              for all j
```

When proc i makes a request

request[j] - # of res.

① request[j] ≤ need[i][j] for all j

(error if not)

② request[j] ≤ avail[j] ∀j

if not, wait

③ Simulate granting the request

avail[j] -= request[j] ∀j

alloc[i][j] += request[j]

need[i][j] -= request[j]

---

check if safe

yes - allocate

no - deny , fix arrays

Thursday, March 17, 2016
2:00 PM

Detection

- allow deadlock
- recognize it
- fix it

---

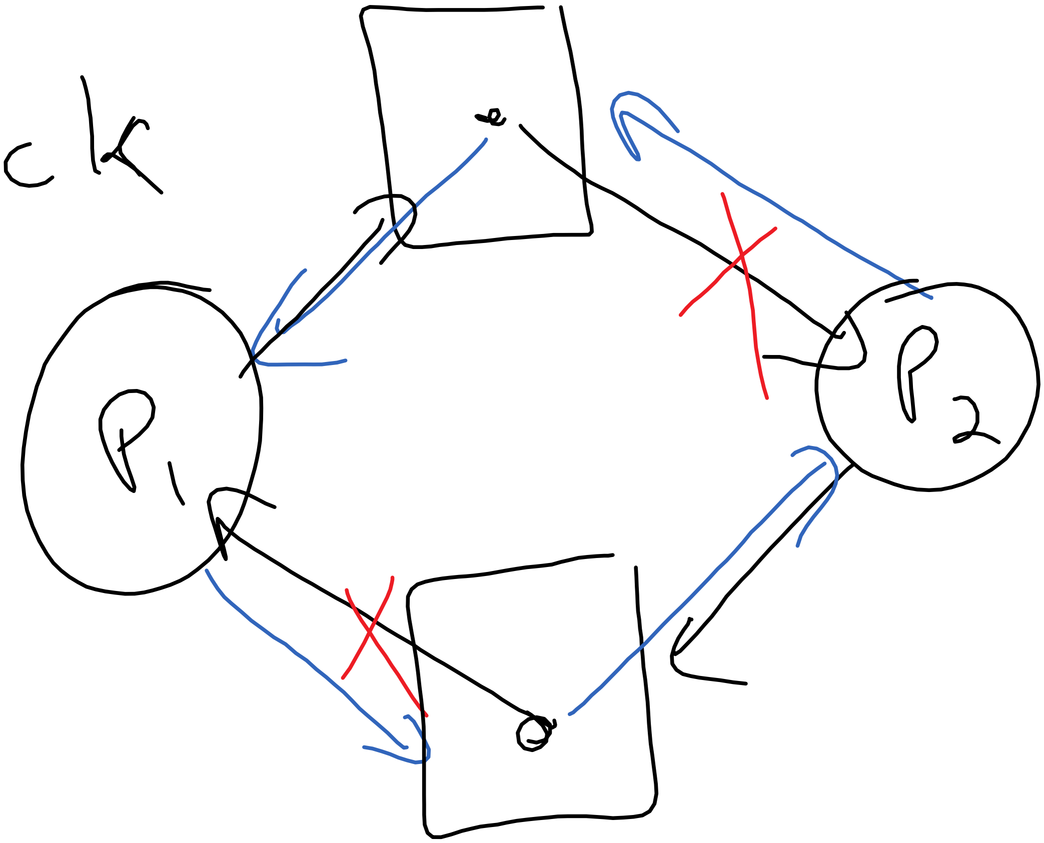
check for deadlock (cycles in resource graph)

---

have locks expire

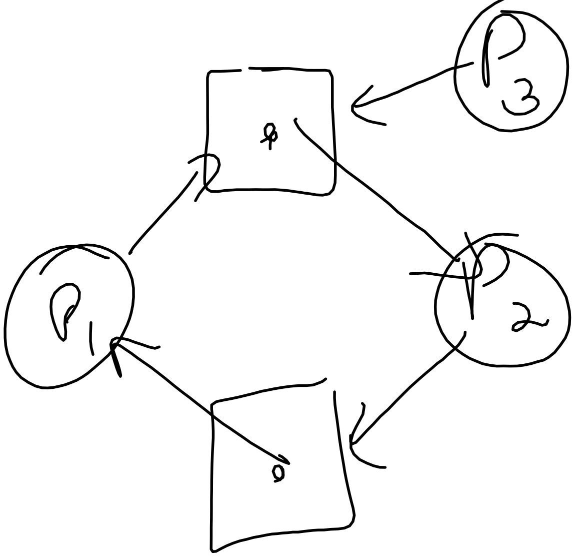
Thursday, March 17, 2016
2:05 PM

Livelock



Thursday, March 17, 2016
2:09 PM

Recovery



1. Kill all involved P_1, P_2, P_3?

2. Kill one at a time

3. ϕ