10/25/23 Readlocks

-> cycle of waiting threads blacked on each other

t1 t2 lock (C+) lock (B) lock (B) lock (B)

mutually recursive locking

deadlack Wont locks

ti tr

but a get but b get)

but_b. put() but_a. put()

resource = produing de consuming power_

-> Necessary Conditions For Veadlock D Bounded Resources : finite instances of each resource D No Preemption : resource can't be tonibly taken away 3 Holdse wait i hold on to resource while waiting & Circular wat : upde of waiting

Dining Philosophers 5 philosophers 5 dropstide (resource) each needs 2 chopsticks teat Chopstick o chopstick Philosopher -> gras one from left & grab one from right 'hilosopher philosopht -> Necessary condition for Readlock but not sufficient Unopstick Chopstick for Acadle de · ^{184doso}II4d , audosolind, · Single instance => Sufficient · multi instance =7 not Chopstick 2 (t) to Chopstick)

What to do? (limit code behavior) when writing code, break one of the 4 conditions for deadlock D Prevention : system controls resource access le scheeluling to avoid D'Awidance = let things be, detect & recover when there's deadlock. 3 Vetection:

Verdlock Prevention (reserve some resources to deal (w/ cases before running out of resources) provide sufficient resources Bounded resources: have preemption No Preesuption = release while wait Hold & wait: ordening. Lading ordening Ciralar wort

A have a way to grant all Deadlock Avoidance -> system grants alless to resame, can delay furfilling requests more requests = Safe. -> needs to know maximum resources each thread needs. Nerstrahy. (2)ABCDE avail= 5 allows [010201010] Atl 10 0 0 00 \sim avail = 4, to be avail = 5 , 5 avail: 3avail-2. to be awal = 2, 3, 4,5 Dt 1) (1) (1) (1) 2 avan = 1. to be anound => -> $E = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 \\ 2 & 2 & - \\ 2 & - & - \\ 2 &$ avort: D.

