# CISC 3595 Operating Systems
Fall 2022

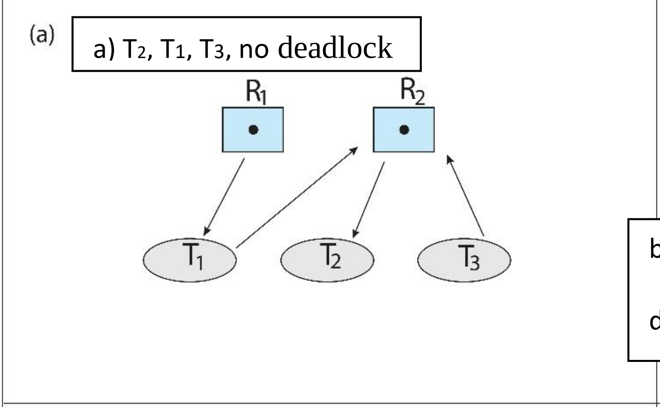
Assignment

Out 11/20, In 11/29

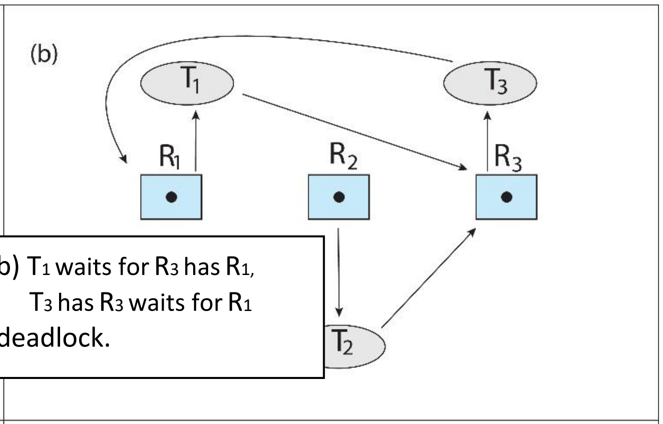
Q1. For each of the following Resource Allocation graphs, use the Graph Reduction algorithm described in the lectures to determine the following: (24 pts total)

i) List a possible order in which vertices are removed from the graph (3 pts each)

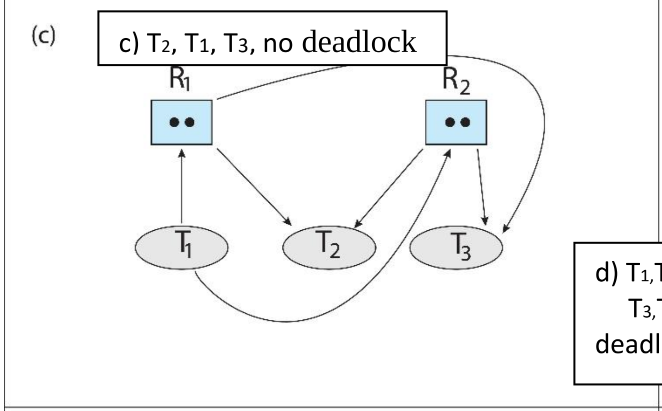
ii) Is there a deadlock? (1 pts each)



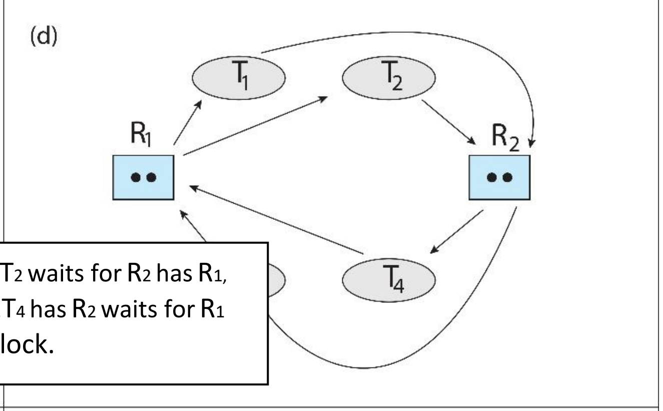
a) $T_2$, $T_1$, $T_3$, no deadlock



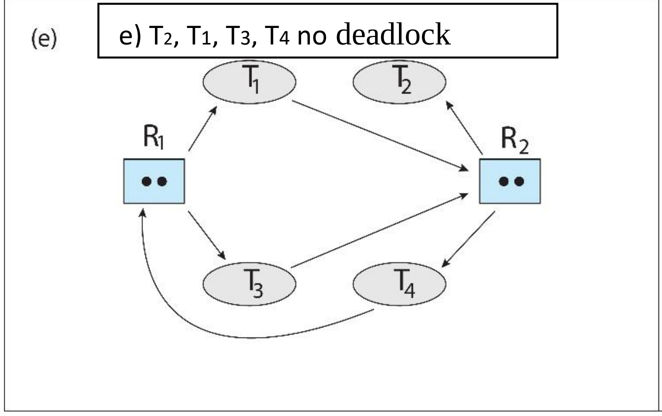
b) $T_1$ waits for $R_3$ has $R_1$,
$T_3$ has $R_3$ waits for $R_1$
deadlock.



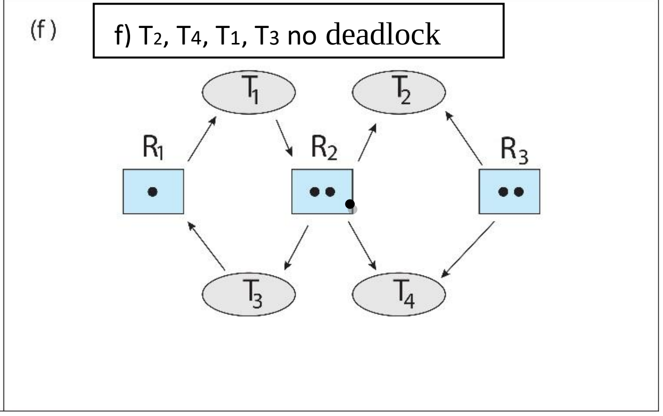
c) $T_2$, $T_1$, $T_3$, no deadlock



d) $T_1$,$T_2$ waits for $R_2$ has $R_1$,
$T_3$,$T_4$ has $R_2$ waits for $R_1$
deadlock.



e) $T_2$, $T_1$, $T_3$, $T_4$ no deadlock



f) $T_2$, $T_4$, $T_1$, $T_3$ no deadlock

Q2. Consider the following system allocation matrix with accompanying maximum requirements of each thread for each resource. (15 pts total)

i) fill in the Need matrix for each scenario (3 pts each)

ii) Is the scenario safe? (1 pts each)

iii) List the order in which the threads can complete or show where the allocation cannot be satisfied. (2 pts each)

*Available = (0,3,0,1) T2,T1,# NOT SAFE because all remaining need 3 of D.*
*Available = (1,0,0,2)T1,T2,T3,T4,T0 or T1,T2,T0,T3,T4 SAFE*

| | Max | | | | Allocated | | | | Need | | | | *Available* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Thread | A | B | C | D | A | B | C | D | A | B | C | D | *A* | *B* | *C* | *D* |
| T0 | 5 | 1 | 1 | 7 | 3 | 0 | 1 | 4 | 2 | 1 | 0 | 3 | | | | |
| T1 | 3 | 2 | 1 | 1 | 2 | 2 | 1 | 0 | 1 | 0 | 0 | 1 | *5* | *6* | *3* | *2* |
| T2 | 3 | 3 | 2 | 1 | 3 | 1 | 2 | 1 | 0 | 2 | 0 | 0 | *3* | *4* | *2* | *2* |
| T3 | 4 | 6 | 1 | 3 | 0 | 5 | 1 | 0 | 4 | 1 | 0 | 3 | | | | |
| T4 | 6 | 3 | 2 | 5 | 4 | 2 | 1 | 2 | 2 | 1 | 1 | 3 | | | | |

| | Max | | | | Allocated | | | | Need | | | | *Available* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Thread | A | B | C | D | A | B | C | D | A | B | C | D | *A* | *B* | *C* | *D* |
| T0 | 5 | 1 | 1 | 7 | 3 | 0 | 1 | 4 | 2 | 1 | 0 | 3 | *13* | *10* | *6* | *9* |
| T1 | 3 | 2 | 1 | 1 | 2 | 2 | 1 | 0 | 1 | 0 | 0 | 1 | *3* | *2* | *1* | *2* |
| T2 | 3 | 3 | 2 | 1 | 3 | 1 | 2 | 1 | 0 | 2 | 0 | 0 | *6* | *3* | *3* | *3* |
| T3 | 4 | 6 | 1 | 3 | 0 | 5 | 1 | 0 | 4 | 1 | 0 | 3 | *6* | *8* | *4* | *3* |
| T4 | 6 | 3 | 2 | 5 | 4 | 2 | 1 | 2 | 2 | 1 | 1 | 3 | *10* | *10* | *5* | *5* |

Available = (1,5,2,0) *T0,T2,T3,T4,T1 SAFE*

| | Max | | | | Allocated | | | | Need | | | | *Available* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Thread | A | B | C | D | A | B | C | D | A | B | C | D | *A* | *B* | *C* | *D* |
| T0 | 0 | 0 | 1 | 2 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | *1* | *5* | *3* | *2* |
| T1 | 1 | 7 | 5 | 0 | 1 | 0 | 0 | 0 | 0 | 7 | 5 | 0 | *3* | *14* | *12* | *12* |
| T2 | 2 | 3 | 5 | 6 | 1 | 3 | 5 | 4 | 1 | 0 | 0 | 2 | *2* | *8* | *8* | *6* |
| T3 | 0 | 6 | 5 | 2 | 0 | 6 | 3 | 2 | 0 | 0 | 2 | 0 | *2* | *14* | *11* | *8* |
| T4 | 0 | 6 | 5 | 6 | 0 | 0 | 1 | 4 | 0 | 6 | 4 | 2 | *2* | *14* | *12* | *12* |

Q3. *Explain* how *Logical Addresses* are mapped to *Physical Addresses* for each of

i) contiguous allocation,

ii) linked allocation

iii) indexed allocation

iv) FAT approaches for representing files.

by showing for each approach how to calculate the physical address from the logical address using the examples address 4209, 3034, 256, 18166, for a system with block size 512. (1 pts each – 16 total)

*Contiguous Allocation 4209 block# 8 offset: 113 CA: FCB contains start block and extent. traverse start blk + 8 blocks, offset 113*

*Linked Allocation 4209 LK: block# 8 offset: 149 LK: FCB contains start block and last block. traverse start blk + 8 links, offset: 145+4*

*Indexed Allocation 4209 IN: index# 8 offset: 113 IN: FCB contains index block# and number of blocks.  traverse to index 8, offset: 113*

*FAT Allocation 4209 FAT: block# 8 offset: 113 FAT: FCB contains start index and last index. traverse from first index, use contents for next index, 8 times. Offset 113*

*Contiguous Allocation 3034 block# 5 offset: 474 CA: FCB contains start block and extent. traverse start blk + 5 blocks, offset 474*

*Linked Allocation 3034 LK: block# 5 offset: 498 LK: FCB contains start block and last block. traverse start blk + 5 links, offset: 494+4*

*Indexed Allocation 3034 IN: index# 5 offset: 474 IN: FCB contains index block# and number of blocks.  traverse to index 5, offset: 474*

*FAT Allocation 3034 FAT: block# 5 offset: 474 FAT: FCB contains start index and last index. traverse from first index, use contents for next index, 5 times. Offset 474*

*Contiguous Allocation 262 block# 0 offset: 262 CA: FCB contains start block and extent. traverse start blk + 0 blocks, offset 262*

*Linked Allocation 262 LK: block# 0 offset: 266 LK: FCB contains start block and last block. traverse start blk + 0 links, offset: 262+4*

*Indexed Allocation 262 IN: index# 0 offset: 262 IN: FCB contains index block# and number of blocks.  traverse to index 0, offset: 262*

*FAT Allocation 262 FAT: block# 0 offset: 262 FAT: FCB contains start index and last index. traverse from first index, use contents for next index, 0 times. Offset 262*

*Contiguous Allocation 18176 block# 35 offset: 256 CA: FCB contains start block and extent. traverse start blk + 35 blocks, offset 256*

*Linked Allocation 18176 LK: block# 35 offset: 400 LK: FCB contains start block and last block. traverse start blk + 35 links, offset: 396+4*

*Indexed Allocation 18176 IN: index# 35 offset: 256 IN: FCB contains index block# and number of blocks. traverse to index 35, offset: 256*

*FAT Allocation 18176 FAT: block# 35 offset: 256 FAT: FCB contains start index and last index. traverse from first index, use contents for next index, 35 times. Offset 194*

| *LA* | Contiguous | | *Linked* | | *Index* | | *FAT* | |
|---|---|---|---|---|---|---|---|---|
| *Addr* | *Block#* | *Offset* | *Block#* | *Offset* | *Block#* | *Offset* | *Block#* | *Offset* |
| 4209 | *8* | *113* | *8* | *(145+4)149* | *8* | *113* | *8* | *113* |
| 3034 | *5* | *474* | *5* | *(494+4)498* | *5* | *474* | *5* | *474* |
| 262 | *0* | *262* | *0* | *(262+4)266* | *0* | *262* | *0* | *262* |
| 18176 | *35* | *256* | *35* | *(396+4)400* | *35* | *256* | *35* | *256* |

Q4. Consider a file system that uses a modified contiguous-allocation scheme with support for extents. A file is a collection of extents, with each extent corresponding to a contiguous set of blocks. A key issue in such systems is the degree of variability in the size of the extents. Assuming internal fragmentation means that allocated blocks are not completely used and external fragmentation means that contiguous unallocated blocks are not enough for any file, what are the advantages and disadvantages of the following schemes? (18 pts)

a. All extents are of the same size, and the size is predetermined.

*Easiest to implement when the extents are all the same size, but the degree of internal fragmentation will increase as the extent size increases (and decrease as the extent size decreases). If extents are fixed sized and do not need to be contiguous, it eliminates external fragmentation.*

b. Extents can be of any size and are allocated dynamically.

*Hardest to implement because variable extents can be divided into smaller extents and combined back into larger extents. Internal fragmentation all but disappears except in as much as the smallest unit is a block, so all file allocation schemes have some internal fragmentation. External fragmentation is more of an issue. Even though small extents can be allocated, it would be inefficient to store files in many small extents. In this caseall of the advantages of contiguous allocation can be lost.*

*c.* Extents can be of a few fixed sizes, and these sizes are predetermined

*More complicated than a single size extent. It can limit internal fragmentation because of a few fixed sized extents, however if the number of extents for each size are not allocated to match the needs of the files stored, then there can be more internal fragmentation. External fragmentation is not an issue since all extents can be allocated.*