

BIRZEIT UNIVERSITY

Electrical and Computer Engineering Department

ENCS 339 Operating Systems

First Semester, 2020-2021 HW#2 Due Date: Thursday November 5th 2020

1. Deadlock Prevention
   1. Describe a real-life deadlock situation NOT in traffic management. Explain why it satisfies the four necessary conditions (mutual exclusion, hold-and-wait, non-preemption, circular wait). How does one resolve that situation? Which condition becomes false?
   2. Give a real life example (better related to the example you gave for a) of a **starvation** situation and show how it differs from the deadlock in part a.
   3. Give an example, where the system is not in a safe state, but if the processes of the system are allowed to be executed, then they will be successfully complete (unsafe state without Deadlock).

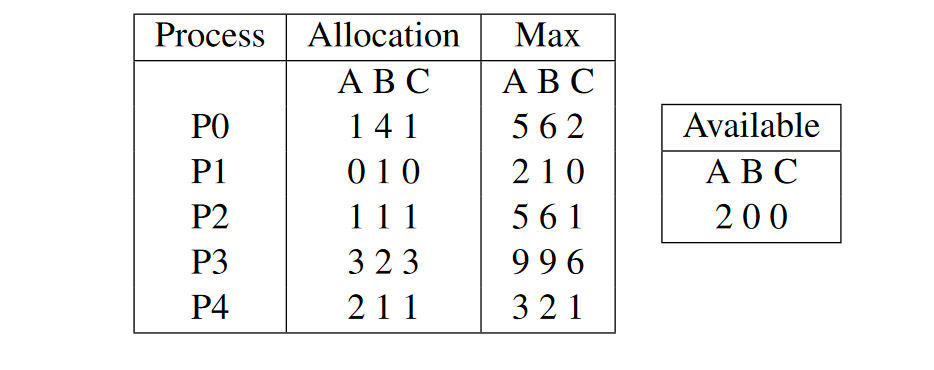
2 Deadlock Avoidance

Consider the following snapshot of a system:

1. Is the system in a safe or unsafe state. Execute the Banker's algorithm to check. Show the different values of the work vector after every iteration. What is the sequence of processes that the algorithm implicitly created? Any other ordering possible in this case?
2. Do the same if P0 had the allocation vector (3 2 2) instead of (1 4 0).

Which processes are NOT able to finish (if any)?

1. If a request from process P0 [in the original case] arrives for (1,0,0,), can the request be granted immediately? Justify your answer.
2. If instead a request from process P4 arrives for (2,0,0,), can the request be granted immediately? Justify your answer.
3. Can we reduce the overall number of resource instances and still be in a safe state? Why? Why Not?
4. Can we increase the number of resource instances so that we are always in safe states without the need for testing? Why? Why Not?



1. Synchronization:
   1. Is the following a correct way for solving the critical section problem (modified Peterson). Why and why not?

Process Pi:

do {

flag[i] = TRUE;

turn = i;

while (flag[j] && turn == j);

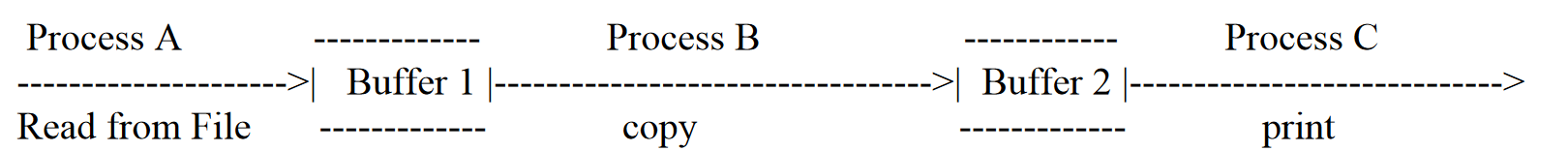
<critical section>

flag[i] = FALSE;

<remainder section>

} while (TRUE);

* 1. Three processes are involved in printing a file (pictured below). Process A reads the file data from the disk to Buffer 1, Process B copies the data from Buffer 1 to Buffer 2, finally Process C takes the data from Buffer 2 and print it.



Assume all three processes operate on one (file) record at a time. Buffer capacity is one record. Show that the program below coordinates the work of the three processes based on the 4 semaphores.

Answer:

semaphore empty1 = 1;

semaphore empty2 = 1;

semaphore full11 = 0;

semaphore full12 = 0;

Process\_A () {

while(1) {

wait(empty1);

read(next\_file(), Buffer\_1);

signal(full1);

}

}

Process\_B () {

while(1) {

wait(full1);

wait(empty2);

copy(Buffer\_2, Buffer\_1);

signal(empty1);

signal(full2);

}

}

Process\_C () {

while(1) {

wait(full2);

print(Buffer\_2);

signal(empty2);

}

}

Good Luck