Solutions
15-410, Spring 2022, Homework Assignment 1.

1 Tape drives (10 pts.)

1.1 3 pts

Is the following state safe? Why or why not?

Though the system has zero tape drives free and two processes entitled to request tape drives, this state is safe. Process B has everything it needs, so it can run to completion and release two tape drives. Then Process C could use those two tape drives to run to completion, at which point Process A would have two drives available despite being entitled to request only one. At least one other safe sequence exists.

1.2 3 pts

Explain why the following state is not safe.

There is no safe sequence because the following situation is true for each process in the system: it is allowed to request one more tape drive, but there are no tape drives available, so it cannot necessarily proceed to completion with the resources it holds plus the resources currently free. If no process can be the first one in a safe sequence, there can't be a safe sequence.

Less formally, each process is allowed to request one tape drive; because there are none free, any such request must block; because *every* process is allowed to block in this fashion before releasing tape drives, *any* blocking can be permanent; if *everybody* blocks, everybody will be waiting for everybody else.

1.3 4 pts

Would this system work with only two tape drives? If so, how well would it work?

The efficiency investigator noticed that all observations of the system had two tape drives busy and one idle. This is because the processes typically need only one tape drive each and because, as we have just shown, it is not safe to have three processes each running with one tape drive. That situation might work out ok, and in fact we know that any process needing two tape drives is a rare case, so the deadlock opportunity, where all three need two tape drives at the same time, is very rare. But an avoidance-aware allocator, once it has given one tape drive each to any two processes, will hold back the third process and leave the third tape drive idle in case the system ends up in the unusual case. So the behavior observed most of the time will be three tape drives, one idle.

If the efficiency investigator is allowed to remove the "idle" tape drive, the good news is that the system will still work. The key to it working is that no process is allowed to individually request more than two tape drives, and we have two, so an avoidance-aware allocator can make that work.

The bad news is that once the avoidance-aware allocator has given out one tape drive to any one process, it will need to keep the other tape drive available for that one process in case it needs it. So the behavior observed most of the time will be two tape drives, one idle. The efficiency investigator will be even less happy: instead of 33% waste, what will be observed is 50% waste.

2 Transfer trouble (10 pts.)

Using the tabular format presented in class, show how concurrent invocations of the "mathematically safe" operation `transfer(A, A, 500)` can double the amount of money in account `A`.

Execution Trace

time	Thread A	Thread B
0	a = 500	
1	if (...)	a = 500
2	a-= 500	if (...)
3	f->v = a /*0*/	a-= 500
4		f->v = a /*0*/
5	t->v += amount /*0=>500*/	
6		t->v += amount /*500=>1000*/

There are many other valid solutions—you can have fewer steps with more things "happening at the same time," or more steps with fewer things "happening at the same time."

Note

This homework assignment was probably "too easy" compared to most exam questions. But hopefully it served its purpose, which was to focus you on working through some examples, as a warm-up or inducement to look at old exams.

Also, hopefully it provides some guidance as to what we consider to be a clear execution trace. *Clarity—providing easy-to-evaluate evidence of your claim—is very important.* Please look over your traces and verify that they are as easy to read as ours. Note that in most cases it is necessary to show more lines of code (i.e., it is often not possible to collapse entire loops into a single line). But sometimes it is... abbreviation and other notational devices are fine as long as it is clear to the reader that you understand the execution flow. *If you are arguing that a trace shows that an execution pattern can repeat, be sure to show exact repetition! The repeating part does not need to be the whole trace (usually it isn't), but you need to ensure that all data structures and all program counters have returned to a previous state.*