CS 4348 Assignment 3

Daniel Crawford

1) How does interrupt disabling provide mutual exclusion?

By disabling interrupts, processes can only have interleaved execution and cannot overlap on a uniprocessor system. Mutual Exclusion can be guaranteed by preventing the process from being interrupted. In this method, the critical section of the code cannot be interrupted, which means that no other process can interfere with this process. Thus, mutual exclusion is achieved by simply preventing the other processes from accessing the same critical data. However, there are drawbacks. The performance and efficiency can be significantly degraded because the process will lose its ability to interleave processes effectively. Another massive drawback is that this approach only works on a uniprocessor system. On a multiple processor system, more than one process can be running at a time, so the computer can see the same mutual exclusion problem.

2) In the Compare&Swap instruction, why must the instruction execute atomically?

Inside the Compare&Swap instruction, the sources of data are critical. The instruction needs to be done atomically(no chance of being interrupted) because if any of the data is overridden during this process then values could be swapped with another value that was not intended. There are three variables in this function that can be affected and hurt the integrity of the data: testval(int), word(int\*), and newval(int). One big concern that can already be seen is that word is a pointer, and its reference to memory can be changed from another process. Besides this, if the values inside of testval and newval are changed, the entire operation can go through incorrectly. So for these reasons, its important to make the Compare&Swap instruction atomic.

3)Coordinate the actions of the two threads below by inserting wait/signal commands on semaphores so that the Give\_order() happens first, then the Take\_order(), then the Serve\_meal(), then the Eat\_meal(). You can assume one thread each.

Semaphores, waiterArrives = 0, placedOrder = 1, orderServed=2.

Customer() {

wait(waiterArrives)

Give\_order()

signal(placedOrder)

wait(orderServed)

Eat\_meal()

}

Waiter() {

signal(waiterArrives)

while(true) {

wait(placedOrder)

Take\_order()

Serve\_meal()

signal(orderServed)

signal(waiterArrives)

}

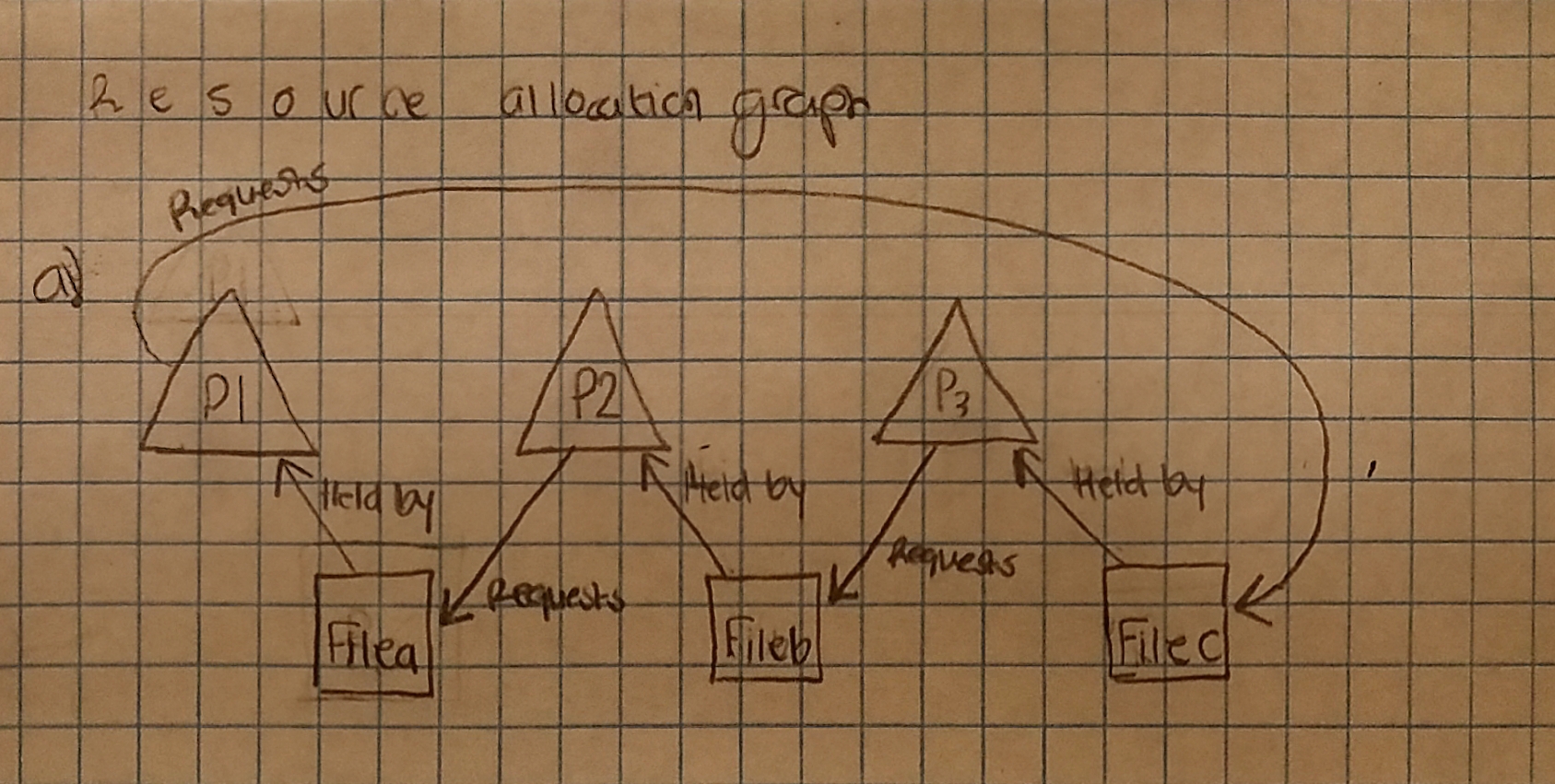
}

4) Why is the mutual exclusion condition necessary for deadlock to occur?

Mutual exclusion is necessary for processes because it does not want another process of higher priority or that interrupted it to interfere with its data. This successfully protects it data by preventing interference. However, this is what causes a deadlock. Once this other process reaches that point, it will enter a infinite loop where it is waiting to gain access to that point in memory. The other process will never finish this critical section because it is waiting for the process of higher priority to finish its execution. Again, this process cannot end its execution because it is waiting to gain access to the critical data, but the other process will never finish using the critical data. Because of this situation, Mutual Exclusion causes deadlocks to occur because it makes two processes depend on each other to finish execution by sharing a memory space.

5) Suppose there are three processes, P1, P2, and P3, and three files, FileA, FileB, and FileC. P1 has FileA and wants FileC. P2 has FileB and wants FileA. P3 has FileC and wants FileB.

a) Draw a resource allocation graph for this situation.

b) Does the graph represent a deadlock? Why or why not? Explain.

Yes, this graph can represent a deadlock. Take this situation for example. Process hold is currently held by file a, and will not release it until it gets file c. However, Process 3 is currently holding file c and will not release until it had file b, and process 2 will not release file b until it gets file a. By this logic, all three processes are waiting on each other to finish their use with the resources, but they never will because every process will never finish since they are waiting on each other. So this is indeed a diagram of a deadlock.

6) Is the following setup safe or unsafe according to the Banker’s algorithm? Show your work and explain your result.

This setup is not safe because two processes are needing more of process 1, but process 1 has no more resources to allocate. For resource 1, P1 and P2 are combined everything in R1, but they require more to finish. Because Resource 1 does not offer enough to let a process end, this setup would be considered unsafe in Banker’s Algorithm.