Operating Systems Textbooks are Wrong about Deadlocks

Kenneth Baclawski

October 2000

Abstract

The Coffman conditions for a deadlock are in every operating systems textbook. In spite of the claim that the four Coffman conditions are necessary and sufficient for the possibility of a deadlock, one can easily give counter-examples of deadlocks that do not satisfy the four Coffman conditions. Indeed, the Coffman article gives such a counter-example. Accordingly, operating systems textbooks should correct their mistake.

Introduction

Every textbook on Operating Systems that I have ever reviewed has a section on deadlocks that includes the four Coffman conditions for a deadlock. It is stated that these four conditions are both necessary and sufficient for a deadlock to occur. This result is puzzling because it is easy to give counter-examples to prove that the result is not correct. Yet, every operating systems book gives exactly the same statement. Some textbooks spend a great deal of time and effort elaborating on the consequences of this result.

In this article we discuss what Coffman actually claimed and give counter-examples to the claims in Operating Systems textbooks.

The Coffman Conditions

The first step in my investigation was to obtain a copy of the Coffman paper and to read it carefully [1]. What I found was quite shocking: *Coffman and his co-authors never proved the result that has been repeated so faithfully in every operating systems textbook.* The four Coffman conditions certainly do appear in the paper, and operating systems textbooks faithfully reproduce them. However, [1] does not prove that these conditions are equivalent to the presence of a deadlock. In fact, it is the other way around: the concept of a (certain kind of) deadlock is defined by the four conditions. More precisely,

> The existence of these conditions effectively defines a state of deadlock.

Even more amazingly, in the very same paper, they give a counter example! They consider the case of a system in which resources are partitioned into types such that a request for a resource of a given type may be satisfied by any resource of that type (e.g., memory pages). The authors then go on to state:

> With this definition a circuit in the state graph is a necessary, but no longer a sufficient condition for the existence of deadlocks.

It seems clear that the intention of the paper is to examine two kinds of deadlock situations. The paper does not cover all possible deadlock situations. Indeed, [1] never defines any of their terms with enough precision to state any theorems at all. Yet no operating systems textbook mentions that there are any limits to the applicability of the first kind of deadlock situation in [1].

Although [1] does not prove any theorems, they do cite some papers that do have proofs. If the four Coffman conditions are denoted by C1, C2, C3 and C4, then the Coffman result (as stated in operating systems textbooks) is the following:

(C1 and C2 and C3 and C4) is equivalent to the presence of a deadlock.

Whereas the actual result claimed by Coffman is something like this:

If C1 and C2 and C3 hold, then C4 is equivalent to the presence of a deadlock.

Needless to say, this is a much weaker result than the statement in operating systems textbooks. As we discuss in the next section, this is the most one can say in general about deadlocks.

Accordingly, all current Operating Systems textbooks are simply wrong. They claim that one can avoid deadlocks by violating one of the

four Coffman conditions. In fact, the only case that actually works is C4. If C1, C2 and C3 all hold but C4 does not, then no deadlock can occur. Indeed, there is an example in [1] which shows that if C2, C3 and C4 occur, but that C1 does not occur, then a deadlock is still possible.

Counter-Examples

We now give some examples to show that none of C1, C2 and C3 are necessary for a deadlock.

C1 (mutual exclusion condition) A counter-example to this condition is given in [1].

C2 (hold-and-wait condition) The most common deadlock scenario in database systems is the following:

- Two processes hold shared (non-exclusive) locks on a record.
- Both processes attempt to acquire an exclusive lock on the record.

If we further assume that neither process has any other locks, then neither process is holding exclusive resources while waiting for other resources. So condition C2 does not hold. The argument that the shared locks are somehow "exclusive" and therefore are being held while waiting for other locks is conclusively contradicted by the statement of condition C1 which defines the meaning of exclusive access to a resource. Examples in the second half of [1] also clarify this point.

C3 (non-preemption condition) Consumable resource deadlocks are a good example of a deadlock scenario in which preemption would not break a deadlock. For more about consumable resource deadlocks, see [2]

Conclusion

This investigation raises a number of issues, such as whether textbooks should continue to discuss the Coffman conditions. If the counter-examples to the Coffman result were obscure and peculiar, then it might be argued that it is better to give students a rough idea of how deadlocks work than to spend time discussing subtle distinctions. However, the counter-examples are neither subtle nor obscure. One can give a nice counter-example to the Coffman result that not only actually occurs in

database systems, but also represents the most commonly occurring deadlock situation by far. This is hardly a peculiar or unusual situation. Giving students the impression that one can avoid deadlocks by invoking the Coffman result could result in systems being built that are seriously flawed.

Another question is how this situation came about at all. It is hard to believe that the authors of operating systems textbooks could have made such a glaring mistake for so many years. This is not a small part of most operating systems textbooks. As many as 10 to 20 pages are devoted to a discussion of the Coffman result in typical texts. One can only presume that the authors have simply copied the ideas that have been repeated so often that they have achieved the status of dogma.

References

[1] E. Coffman, M. Elphick and A. Shoshani (1971). "System Deadlocks," *ACM Computing Surveys* 3(2)67–78.

[2] Singhal and Shivaratri, *Advanced Concepts in Operating Systems*