Deadlocks

Deadlocks is when competing processes in the user space are all waiting for a set of resources and prevents sets of concurrent process from completing. It occurs only if four conditions are held true. The first condition is mutual exclusion. Mutual exclusion is when a resource is not sharable, this means that resources cannot be used by multiple processes. The second is hold and wait. Hold and wait is when a single process is holding a resource requested by another processes. Third is no preemption. No preemption means a that a resource will only be let go after using it. Lastly, and most importantly, circular wait. Circular wait is when a process is waiting for a resource while holding a resource that is also being waited on by another process and this continuation keeps going. For example, if the first process is waiting on the second process and the second process is waiting on the first process. This in terms makes a circular order of wait.

There are multiple possible solutions to avoiding a deadlock. One such solution for deadlock avoidance is the requirement of a priori. One than can set a maximum number of resources that each process may have, thus limiting the possibilities of deadlocks. A second type is an algorithm that dynamically examines how resources are being allocated and making sure that this does not occur. There are two types of algorithms that would work, for single instances, using a resource-allocation graph would help solve the problem, for multiple instances of a resource we then use banker’s algorithm.