**CSC-3044 Operating Systems & System Programming**

**Laboratory 4 Report**

**NAME:** Joseph Leskey

1. **At what number do you see the first problem?**  
     
   There is a duplicated 1.
2. **Which statements in the script make up the critical region of the program?**  
     
   n=`tail -1 numbers` # retrieve last line of file  
   ((n = n + 1))  
   echo $n >>numbers # append n at end of file
3. **What is the purpose of a lock file?**  
     
   A lock file indicates that an associated resource is in use. It is used to block concurrent access that could lead to race conditions and unreliability.
4. **We saw the lack of a lock file create a race condition, but could introducing a lock file lead to a deadlock? If so, how; if not, how is the situation avoided?**  
     
   Lock files can lead to deadlock. If processes apply more than one lock without uniformity to shared resources, there is a risk that a process will place a lock on a resource that a different process needs before it can release its own lock on a resource that the first process needs. This circular wait would result is a standstill that would typically require manual intervention.
5. **Are the locks enforced by the operating system (mandatory) or serve as a warning (advisory) on Linux systems?**  
     
   While it may be possible to set a mandatory lock in Linux, locks are generally advisory on Linux systems. Even typical locks created by system call are designed to be voluntarily checked by programs that access the associated resources.
6. **What about Microsoft Windows?**  
     
   Locks in Windows can often be mandatory. Open files, depending on how they were opened, may be automatically inaccessible to other process. This does not have to be the case, as advisory locks are possible.
7. **If you had implemented the numberseq.sh as a C program, what system calls would have helped you when working with lock files?**  
     
   `flock` or possibly `fcntl` would be the main ones.