Aaron Delahoussaye (C00449804)

Allen Sanders (C00496790)

Michael Vedol (C00459436)

Riley Young (C00445412)

Project 2 Report

Task 1 – Access Control Matrix (Michael Vedol):

The Access Control Matrix class in Task 1 uses an N x (N + (N-1)) sized matrix generated from random values from 3 to 7 for the N and M values. An array “fileContents” is used to populate the array “access\_Matrix” array if the values are either read or write. Semaphores are used to access “fileContents” associated with files in the access\_Matrix array ass well as mutex and area semaphores to manage the reading and writer access to files in the access\_Matrix

Methods: .

The file\_Access method accepts an integer value and returns a string value. The integer value is a spot in the access matrix. A random number generator in the range of 0 to 3 is used to determine if the element will be given file access. 0 being no access, 1 giving read access, 2 giving write access, and 3 giving read and write access.

Domain\_Access follows a similar premise to file\_Access. It uses a random number generator from 0 to 1; 0 returning a blank value with no access, 1 giving access to switch to a different domain.

The threadRequestManager method acts as an overhead for determining if files or domains are trying to be accessed, and it manages the number of requests each thread can complete. With requestCount equaling five or greater, a random number (target) is generated between the number of files and the number of other domains, oDomains in the code, to select whether files or domains are accessed. If the target is less than the number of files, files are attempting to be accessed, else domains are trying to be accessed.

threadFileAction takes in an integer value as the file index of the access\_matrix and checks it’s read and/or write privileges. If there are read or read/write privileges, the semaphores fileSem, area, and mutex are used to read objects and offer synchronization to access and guard the file. The semaphores are released after the file is read. If there are write or read/write, the area and fileSem semaphores are used to write contents to the file and offer synchronization to the file being written to. Semaphores are released.

threadDomainAction accepts an integer value targetDomain. The method checks the permission of the current domain to access the target domain it is attempting to switch to. If domain is set to “allow”, the domain is granted permission to switch to its target domain. If permission is set as “N/A”, the domain is not allowed to switch since it cannot switch into itself. Lastly, if the permission is neither of the two options above, the domain is not granted access to switch.

Question:

1. Is there any chance of deadlock in this simulation? What changes could cause deadlocks.  
 If the number of domains and files were on the minimum range and all the threads were attempting to access the same file, I think there may be a chance of deadlocking. If the yields were not set to the correct value, or if the release function was never called deadlocks would occur. Also, if the Semaphores were not initialized properly.

Task 2 (Riley Young):

It doesn't differ greatly; it simply requires changing certain values to do different things. I cannot say which task was easier to implement, considering they were handled by separate people. I can say I didn’t have a lot of trouble with this task.

Task 3 (Aaron Delahoussaye):

What disadvantages do capability lists introduce given a more domain-based approach?

A key disadvantage of capability-based access control is the potential for privilege escalation. If a domain gains unauthorized access to a capability, it can misuse it. Careful management of capabilities is crucial.

Task 4 (Allen Sanders)

Task 5(Questions answered by Allen Sanders):

What is the importance of protection in a multi-user system?

Protection in a multi-user system ensures data security and user privacy. It prevents unauthorized access to sensitive information, making sure it doesn’t fall into the wrong hands and is used for malicious purposes or is even accidentally altered.

What could happen if all users were granted equal permission?

There would be a higher possibility of data breaches because then anyone could access or modify sensitive information. Also, there would be less accountability because it would be harder to tell who performed a certain action due to the fact that everyone would have the same permissions to do the same things.