Obligatory Assignment #2: Race Conditions

Brage Celius - 120920

17. september 2014

Race Conditions

Race conditions occur when different processes simultaneously want to use the same resource at the same time. This can lead to unintended results in one or both processes. For example say one process is running a script that checks if a file exists before opening it. The scripts runs up to the point of checking if the file exists. Then another process running a different script is given CPU time and deletes that file. When the first process resumes execution of the script, it will try to open a file that no longer exists, causing the script to fail.

E-Voting, Electronic ID Solution

The Imaginary solution of my choosing is an e-voting solution where users are allowed to vote for the political faction of their choosing. Only one vote can be cast per user.

Race Condition 1

The first race condition concerns the login sequence. The current sequence does not check if a user is already logged in. This will allow the user to establish several different sessions from different devices or even different browser windows on the same device.

Login Sequence

Client Side	**Server Side**
1. User logs on to the application	
2. Send Login Information to Server	
	3. Authenticate user and return true/false
	4. If true, establish session, else return failure message to user and start over

Solution

Add a check to the login sequence to see if user is already logged in. If that is the case, use the same sessionID.

Client Side	**Server Side**
1. User logs on to the application	
2. Send Login Information to Server	
	3. Authenticate user and return true/false
	4. If true, check if the user has an active session. Else, return failure message to user and start over.
	5. If user has an active session, use the active sessionID. Else, create a new session.

However, this solution adds a potential TOCTTOU(Time of Check to Time of Use") issue in the sequence. Say Bill is trying to logon to the application. However, Bill is a quirky guy so he decides to do it on two devices at exactly the same time. The server runs the first login process(p1). However, when the process has run the sequence through 4. and checked whether Bill has any active sessions(which returns false), another process(p2) handling the second login sequence is given CPU time. This process finishes the sequence and successfully establishes a session(Bill is now marked with an active session). Now, p1 is given CPU time and finishes the sequence. However, since p1 already decided that there were no active sessions for Bill, it goes ahead and creates another session. Bill is now logged in to two different sessions, and can potentially vote 2 times. To fix this problem, a mutex or a binary semaphore should be added around step 4 and 5 of the login sequence. By doing this, we ensure that step 4 and 5 are executed together without interference.

Race Condition 2

Vote Sequence

Client Side	**Server Side**
1. A session is established	
2. User is presented with and chooses a candidate	
	3. Check if user have already voted, if so abort and show error message. Else continue to process vote.
	4. Total votes for chosen candidate is incremented by 1
	5. Mark user as "Already Voted"

Assuming Race Condition 1 is not yet fixed, Bill has (through the above-mentioned scenario) logged in to two sessions with the same account. Bill is still as quirky as ever, and continues on to choose candidates on both sessions, locking in his choice on both devices at the same time. The server runs the first process(p1), checking whether the user have already voted. Seeing as Bill has not voted before, the vote continues to be processed. However, after executing step 3 of the vote sequence, another process(p2) handling Bill's second session is given CPU time. Seeing as p1 was stopped short of marking Bill as "Already Voted", Bill is allowed to vote once more. p2 finishes running and CPU time is given back to p1. p1 goes on to mark Bill as "Already Voted". Bill has now unintentionally registered two votes from the same account.

Solution

A mutex or a binary semaphore should be added around step 3-5. This assures that these steps will finish running uninterrupted. If this solution had been implemented from before, Bill would have gotten an error message on the session handled by p2 and only the first vote would have been registered.