Joakim Harbitz
Studentid: 120924 / 12HBISA

Obligatory exercise #2

Date: 2014-09-17

Task

- Imagine an electronic id solution (e.g. login to e-voting).
- Think about two (or more) possible race conditions and their effects.
- Describe countermeasures for the race conditions you observe.

What is a race condition

In a multithreaded system where different processes share the same processor, race conditions may occur. Since the algorithm used by the thread scheduling process may change the running threads at any time, the program running never knows what happened in between the change. What happen is that program A checks a variable and then gets swapped out with program B, which then changes the same variable. When program A gets the processor and acts on the variable, problems might happen since, program A doesn't know that program B has changed the variable. [1]

Race condition 1

User A is logging into a website and receives a session-id from the web server. The session-id identifies the user to the web server so that the user don't have to log in every time there is a page refresh.

User B browses to the same website and logs in as User A. User B receives a new session-id, but signs out of the website right after. The website deletes the session ids stored for User A's account and User B is signed out. This means that user A will be signed out as well since every session ids are deleted when a logged in user clicks the sign out button.

Solution: If the website checks which session-id requested the sign out, not all of the session-ids will be deleted and user A will stay logged in even though user B logs in and out with his or hers credentials.

Race condition 2

At a post office, a new queue system has been installed. The customer pushes a button on the ticket issuer by the door, which adds the new number issued to the user to a queue on the ticket server. The counters has a button which activates the process that will go to the beginning of the queue on the ticket server, fetch the number, remove the last one and then display it above the counter. A race condition may occur if the process fetching the first number in the queue gets an interrupt right after fetching the number. If another process interrupts at this stage, fetches the same number as the first process and then removes it before the first process gets the processor. The first process will then remove the last number in the queue which in fact is a number that has never been fetched. This means that the person in line will have to wait forever.

Solution: If the program fetching the queue numbers had entered critical section while fetching and removing the numbers in the queue, the process would be sure that no other process would fetch and delete the number the process was going to delete while inactive.

1. http://support.microsoft.com/kb/317723