NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
(KARACHI CAMPUS)
FAST School of Computing
Fall 2023

Title: System Call for Semaphore- Chain Smoker Problem

Team Members:

- 21K-4503 Muhammad Tahir
- 21K-3279 Insha Javed
- 21K-4606 Sabika Shameel

Department:

BSCS (4-E)

Problem Statement:

Suhas Patil first brought up the issue of chain smoking and stated that semaphores could not be used to resolve it. Although there are some caveats to that statement, the issue is nonetheless intriguing and difficult. An agent and three smokers are two of the four threads concerned. In an endless cycle, smokers prepare and smoke cigarettes after waiting for the necessary components. Matches, paper, and tobacco make up the components. We suppose that the agent has an infinite supply of each of the three ingredients and that each smoker also has an infinite supply of one of the ingredients, i.e., each smoker has matches, paper, and tobacco. The agent provides the smokers with two different components on a regular basis, picking them at random. Depending on the selection of components.

Constraints

The agent (which represents an operating system) is subject to the following constraints in order to model a resource-management issue of operating systems in realistic situations:

- Only using condition variables to indicate whether a resource is available, the agent is permitted to interact.
- Smokers are not allowed to ask the agent what is available because the agent is not allowed to reveal resource availability in any other way.
- The agent is not allowed to be aware of the resource requirements of smokers, so the agent is not allowed to wake up a smoker immediately.
- The agent can only make two resources accessible at a time, and it must wait for a smoker to light up before making any more resources available.

Problem analysis

Complexity of the problem

Every smoker thread may use one of the two items that the agent makes accessible, but only one may use both. For instance, if the agent makes paper and matches accessible, both people who smoke with paper and matches will want one of these, but neither has tobacco, so neither can smoke. However, if either of them awakens and consumes a resource, it will stop the tobacco thread from starting to burn and, as a result, stop the agent from awakening to give more resources. If this occurs, the system will become stuck and no process will be able to move forward.

Methodology

The agent signals two events in each iteration, and smokers must wait for both events to occur before deciding how to act appropriately.

However, a monitor's condition wait action can only be used for one event at a time. In each smoker function, stacking two condition wait operations doesn't assist because doing so would be impossible because it would require us to know the exact timing of the two events. As a result, we require a middle device that can intercept the two events the agent signals, combine them into one event, and then alert the smokers. (i.e. wake up the correct smoker).

Implementation

To put the above-mentioned plan into practice, there are various methods. The intermediate mechanism is implemented by three listeners sharing a single global variable named sum in my approach. Each listener is only accountable for a specific event category (paper, match, or tobacco). I use the total to keep track of the activities that listeners have heard about thus far. Each pairing of two occurrences results in a different value. Listeners will start a new event, alert the smokers, and wake up the appropriate smoker once the sum hits one of the characteristic values.

★ The programming language we use is C, and the uthread and pthread libraries are also used.