Varan Sharma

CSc 332 – Project 2 Report

The Cigarette Smoker’s Problem

The Cigarette Smoker’s Problem has a total of 4 processes. One is the agent, which puts two of three items required to make a cigarette on the table and the other three are the smokers. They each have one of the three items required to make a cigarette. Each time the agent places items on the table, the smoker with the remaining items creates a cigarette and smokes it.

The issue with this is that the execution order of different processes cannot be guaranteed, but there are constraints to the order in which actions are to be performed. For example, a smoker cannot create a cigarette until only the remaining items are on the table. The agent cannot place items on the table until a smoker picks the items up and rolls his cigarette. These constraints require semaphores for synchronization.

How the program works is that it creates three child processes, each being a different smoker. The parent process is the agent. There is a semaphore *mutex* for placing/retrieving items on/from the table. There is also a semaphore for each process (three smokers and one agent). Each process semaphore is initialized to 0. The agent begins by waiting for the table to be open for use (*P(mutex)*). Then, the agent places items on the table (store data in *buffer*). After that is done, depending on the items placed, the agent will signal the smoker that will be able to make a cigarette and then waits (*P(<semaphore for smoker>)*). Meanwhile, each smoker has been waiting. When a smoker receives the signal (*P(<semaphore for smoker>)* and the table is not being used (*P(<mutex>)*), it will create a cigarette using the items on the table. After doing so, the smoker signals the agent (*V(mutex)*) that it is done using the items on the table. It also signals the agent to continue with what it was doing (*V(<semaphore for agent>)*). This sequence of events is then repeated as many times as necessary for the smokers to pass away from lung failure.