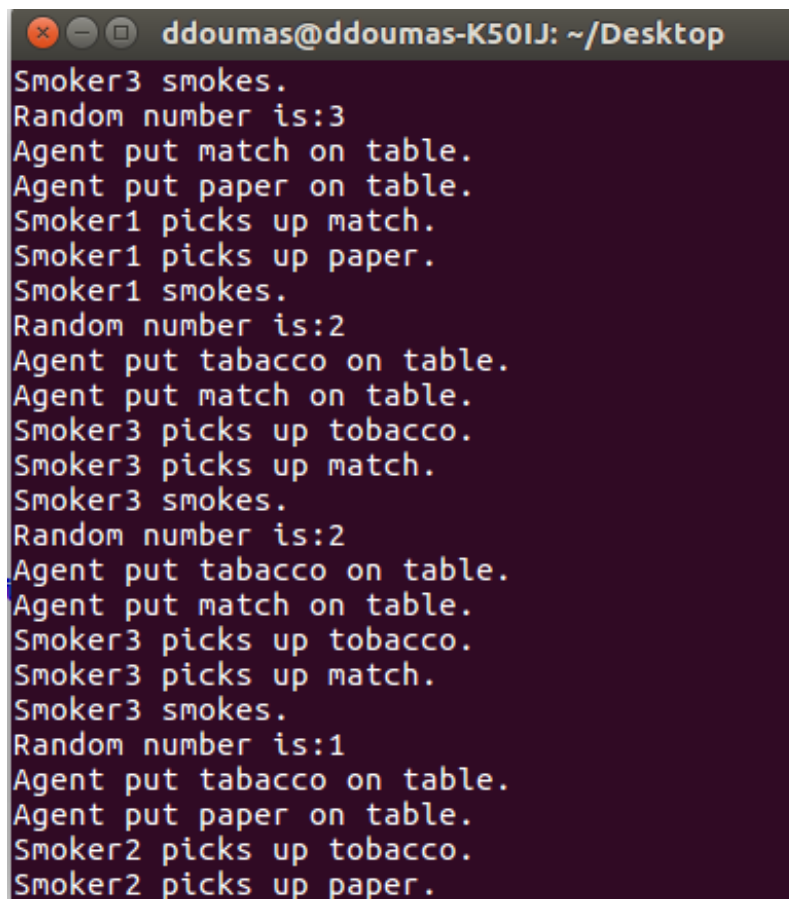


CSC332 Task6 Report

There are challenges when dealing with synchronizing several process or threads in the Cigarette Smoker problem. There are two ways to solve the race condition issue using semaphores and threads (in particular using the pthread libraries). There are four processes in total; the agent, and the three smokers. The agent place one out of the three material need to smoke the cigarette randomly on a table and one of the three smokers will pick it up and begin to smoke. The process repeats. The output of the two solutions is shown below (both have the same outputs).

There similarities and differences between the semaphore solution and the thread solutions that have their own merits and demerits. They both have the capability of waking or making a process/thread sleep. They both have the capability of enclosing a critical section. The difference lie between forking a process and creating threads. Forking a process copies its parent data, where as threads share the same data. So mutual exclusive variables in threads can't be used in a fork solution based program.

A terminal window with a dark background and light-colored text. The window title bar shows a red close button, a yellow minimize button, and a green maximize button, followed by the text 'ddoumas@ddoumas-K50IJ: ~/Desktop'. The output text is as follows:

```
Smoker3 smokes.  
Random number is:3  
Agent put match on table.  
Agent put paper on table.  
Smoker1 picks up match.  
Smoker1 picks up paper.  
Smoker1 smokes.  
Random number is:2  
Agent put tobacco on table.  
Agent put match on table.  
Smoker3 picks up tobacco.  
Smoker3 picks up match.  
Smoker3 smokes.  
Random number is:2  
Agent put tobacco on table.  
Agent put match on table.  
Smoker3 picks up tobacco.  
Smoker3 picks up match.  
Smoker3 smokes.  
Random number is:1  
Agent put tobacco on table.  
Agent put paper on table.  
Smoker2 picks up tobacco.  
Smoker2 picks up paper.
```