
# System Programming

*Homework 3 Report | 2021 - 2022*

Ahmet Tuğkan Ayhan

1901042692

# Problem

This problem between chiefs and the wholesaler is actually based on the cigarette smokers problem introduced in the 1970s. To explain the problem simply, chiefs have 2 of the 4 ingredients used in making Güllaç, and they can only get the remaining ingredients through wholesalers. However, the problem here is that possibility of causing deadlock. This problem occurs when the wholesaler brings 2 ingredients at a time and the chief who needs these 2 ingredients from the chiefs cannot get the ingredients(because other chiefs get them first but they can't finish the dessert neither).

## Solving the Problem with Named Semaphores

Although there are multiple solutions for this problem, one can solve it with named semaphores. In my program, I used 12 named semaphores to solve problem. One semaphore for each chief(6 total), one semaphore for the wholesaler, one semaphore for each pusher(4 total and 1 additional for mutual exclusion). I implemented the same solving method of cigarette-smoker problem. Lifecycle of the program is like this:

1. Wholesaler reads two ingredients and stores the ingredients inside a char array(uses shared memoy)
2. Wholesaler calls **sem_post** for the ingredients. (For example if it reads **SW** then it calls **sem_post(sem_sugar)** and **sem_post(sem_walnuts)**)
3. These semaphores are used in pushers. When a pusher wake up (if we use SW again, the pushers; **pusherSugar** and **pusherWalnuts** will wake up)
4. Then these pushers will look for flags(there are total of 4 flags for each ingredient and they are stored in shared memory). If another flag other than pusher's ingredient is assigned before this pusher, then it will wake up the related chef.
5. We can think last step as, **SW** is readen by the **wholesaler** and wholesaler first woke **pusherSugar** up. pusherSugar looks for other ingredient flags and nothing is assigned yet, then it changes it's own flag which is **sugarFlag**. Then, pusherWalnuts wakes up and it notices that **sugarFlag** is assigned. Finally, it wokes related chef up and clear flags.
6. When chef process wake up, it prints necessary information to the console, increments his/her own güllaç count, unlocks wholesaler using

**sem_post** and waits again until another pusher wakes him/her up.

7. Wholesaler reads the next ingredient again and if there is no ingredient left, then wholesaler breaks out the infinite for loop.
8. After breaking the for loop it sends kill signal(SIGINT signal) to all other chiefs/pushers and waits for them to return their total güllaç count(only chefs return a value). Then they will be collected by the wholesaler.
9. After all chiefs terminate, wholesaler reads the güllaç count and terminates.

Using the input given in the pdf file, result comes as this

```
tgknyhn@ubuntu:~/Desktop/hw3$ make named
gcc -ggdb3 -Wall -pthread source/namedsem.c source/file.c -o hw3named && ./hw3named -i "ingredients.txt"
 -n "milk" "flour" "walnuts" "sugar" "pusherlock" "wholesaler" "ch0" "ch1" "ch2" "ch3" "ch4" "ch5"
chef0 (pid 60006) is waiting for Walnuts and Sugar (  )
chef1 (pid 60007) is waiting for Flour and Walnuts (  )
chef2 (pid 60008) is waiting for Sugar and Flour (  )
chef3 (pid 60009) is waiting for Milk and Flour (  )
chef4 (pid 60010) is waiting for Milk and Walnuts (  )
chef5 (pid 60011) is waiting for Sugar and Milk (  )
the wholesaler (pid 60001) delivers Milk and Sugar (MS)
the wholesaler (pid 60001) is waiting for the dessert (MS)
chef5 (pid 60011) has taken the Sugar (MS)
chef5 (pid 60011) has taken the Milk (M )
chef5 (pid 60011) is preparing the dessert (  )
chef5 (pid 60011) has delivered the dessert (  )
the wholesaler (pid 60001) has obtained the dessert and left (  )
the wholesaler (pid 60001) delivers Flour and Milk (FM)
the wholesaler (pid 60001) is waiting for the dessert (FM)
chef3 (pid 60009) has taken the Milk (FM)
chef3 (pid 60009) has taken the Flour (F )
chef3 (pid 60009) is preparing the dessert (  )
chef3 (pid 60009) has delivered the dessert (  )
the wholesaler (pid 60001) has obtained the dessert and left (  )
the wholesaler (pid 60001) delivers Walnuts and Sugar (WS)
the wholesaler (pid 60001) is waiting for the dessert (WS)
chef0 (pid 60006) has taken the Walnuts (WS)
chef0 (pid 60006) has taken the Sugar ( S)
chef0 (pid 60006) is preparing the dessert (  )
chef0 (pid 60006) has delivered the dessert (  )
the wholesaler (pid 60001) has obtained the dessert and left (  )
the wholesaler (pid 60001) delivers Sugar and Milk (SM)
the wholesaler (pid 60001) is waiting for the dessert (SM)
chef5 (pid 60011) has taken the Sugar (SM)
chef5 (pid 60011) has taken the Milk ( M)
chef5 (pid 60011) is preparing the dessert (  )
chef5 (pid 60011) has delivered the dessert (  )
the wholesaler (pid 60001) has obtained the dessert and left (  )
chef0 (pid 60006) is exiting (  )
chef2 (pid 60008) is exiting (  )
chef4 (pid 60010) is exiting (  )
chef3 (pid 60009) is exiting (  )
chef5 (pid 60011) is exiting (  )
chef1 (pid 60007) is exiting (  )
the wholesaler (pid 60001) is done (total desserts: 4) (  )
tgknyhn@ubuntu:~/Desktop/hw3$
```

## Solving the Problem with Unnamed Semaphores

Solving the problem with unnamed semaphores is very similar to named semaphores. But instead of using semaphores through a file which it's name declared with command line arguments, we use semaphores which instantiated in shared memory. Since lifecycle of two program is similar I won't write the cycle again. Different from named semaphore I used **mmap** to store semaphores inside shared memory and initialized them with **sem_init**.(I used **sem_open** with named semaphores).

```
tgknyhn@ubuntu:~/Desktop/hw3$ make unnamed
gcc -ggdb3 -Wall -pthread source/unnamedsem.c source/file.c -o hw3unnamed && ./hw3unnamed -i "ingredient
s.txt"
chef0 (pid 60040) is waiting for Walnuts and Sugar (  )
chef2 (pid 60042) is waiting for Sugar and Flour (  )
chef3 (pid 60043) is waiting for Milk and Flour (  )
chef1 (pid 60041) is waiting for Flour and Walnuts (  )
chef4 (pid 60044) is waiting for Milk and Walnuts (  )
chef5 (pid 60045) is waiting for Sugar and Milk (  )
the wholesaler (pid 60035) delivers Milk and Sugar (MS)
the wholesaler (pid 60035) is waiting for the dessert (MS)
chef5 (pid 60045) has taken the Sugar (MS)
chef5 (pid 60045) has taken the Milk (M )
chef5 (pid 60045) is preparing the dessert (  )
chef5 (pid 60045) has delivered the dessert (  )
the wholesaler (pid 60035) has obtained the dessert and left (  )
the wholesaler (pid 60035) delivers Flour and Milk (FM)
the wholesaler (pid 60035) is waiting for the dessert (FM)
chef3 (pid 60043) has taken the Milk (FM)
chef3 (pid 60043) has taken the Flour (F )
chef3 (pid 60043) is preparing the dessert (  )
chef3 (pid 60043) has delivered the dessert (  )
the wholesaler (pid 60035) has obtained the dessert and left (  )
the wholesaler (pid 60035) delivers Walnuts and Sugar (WS)
the wholesaler (pid 60035) is waiting for the dessert (WS)
chef0 (pid 60040) has taken the Walnuts (WS)
chef0 (pid 60040) has taken the Sugar ( S)
chef0 (pid 60040) is preparing the dessert (  )
chef0 (pid 60040) has delivered the dessert (  )
the wholesaler (pid 60035) has obtained the dessert and left (  )
the wholesaler (pid 60035) delivers Sugar and Milk (SM)
the wholesaler (pid 60035) is waiting for the dessert (SM)
chef5 (pid 60045) has taken the Sugar (SM)
chef5 (pid 60045) has taken the Milk ( M)
chef5 (pid 60045) is preparing the dessert (  )
chef5 (pid 60045) has delivered the dessert (  )
the wholesaler (pid 60035) has obtained the dessert and left (  )
chef3 (pid 60043) is exiting (  )
chef4 (pid 60044) is exiting (  )
chef1 (pid 60041) is exiting (  )
chef0 (pid 60040) is exiting (  )
chef2 (pid 60042) is exiting (  )
chef5 (pid 60045) is exiting (  )
the wholesaler (pid 60035) is done (total desserts: 4) (  )
tgknyhn@ubuntu:~/Desktop/hw3$
```