CIS 452 – Operating Systems

Project 2 – Bake Off!

April 12th, 2023

Evan Johns, Alexis Webster

INTRODUCTION

For this project, we had to implement a kitchen for bakers to use to bake a specific set of recipes: cookies, pancakes, pizza dough, soft pretzels, and cinnamon rolls. These bakers are competing against one another to bake 1 of each of the recipes the fastest. The kitchen had a number of shared resources: 2 mixers, 1 pantry, 2 refrigerators, 3 bowls, 5 spoons, and 1 oven. These resources had to be represented using counting semaphores to keep track of their usage, and each baker was represented by a pthread.

DESIGN

As for the design of the project, the code starts with including necessary header files, and then defines semaphores for each shared resource. The number of each resource is defined using integer variables. A string array containing ingredient names is provided, followed by a *Recipe* struct that includes a recipe name, a list of required ingredients, and the number of ingredients. Five different recipes are created, each with its own list of required ingredients as described by the project requirements.

In the *main* function of the code, we start by initializing the semaphores, getting user input for the number of bakers, and creating a separate *pthread* for each baker. Each *pthread* is then assigned the *bake* function as its entry point. The *main* function then waits for all the threads to finish before cleaning up the semaphores and freeing allocated memory.

The *bake* function simulates the process of a baker baking all the recipes in a predefined list. For each recipe, the baker first collects the required ingredients, prepares the recipe by using the tools (bowls, spoons, and mixers), and finally bakes the recipe in the oven. The *getIngredient*, *prepareRecipe*, and *bakeRecipe* functions are responsible for handling the actual resource acquisition and release using semaphores.

The *getIngredient* function checks whether the ingredient is a pantry or refrigerator item and then acquires and releases the corresponding semaphore. The *prepareRecipe* function acquires and releases semaphores for bowls, spoons, and mixers, simulating the use of these tools in the process. Finally, the *bakeRecipe* function uses the oven semaphore to simulate the time spent baking in the oven.

In the design of this project, it was important to promote concurrency while ensuring proper resource management. Each baker operates independently, and semaphores are used to enforce exclusive access to resources as needed. This design helps prevent race conditions and ensures a fair distribution of resources among the bakers.

OBSTACLES

One of the obstacles we faced had to do with the many header files we had to include. We used so many constructs that we’ve learned in class all in one file, so we kept having to add header files, and using so many concepts at once in one program also makes things pretty confusing at times.

Another issue we ran into was passing in the baker’s ID as the pthreads were being created. We struggled to figure out what to make the last pthread\_create() argument, as well as what type to assign to the bakerId variable in our bake() method, but we ended up figuring it out. We started out by using the index i as our pthread identifiers, and that was also our end result, but we needed to cast it to a different type in order for it to be passed in to the pthread.