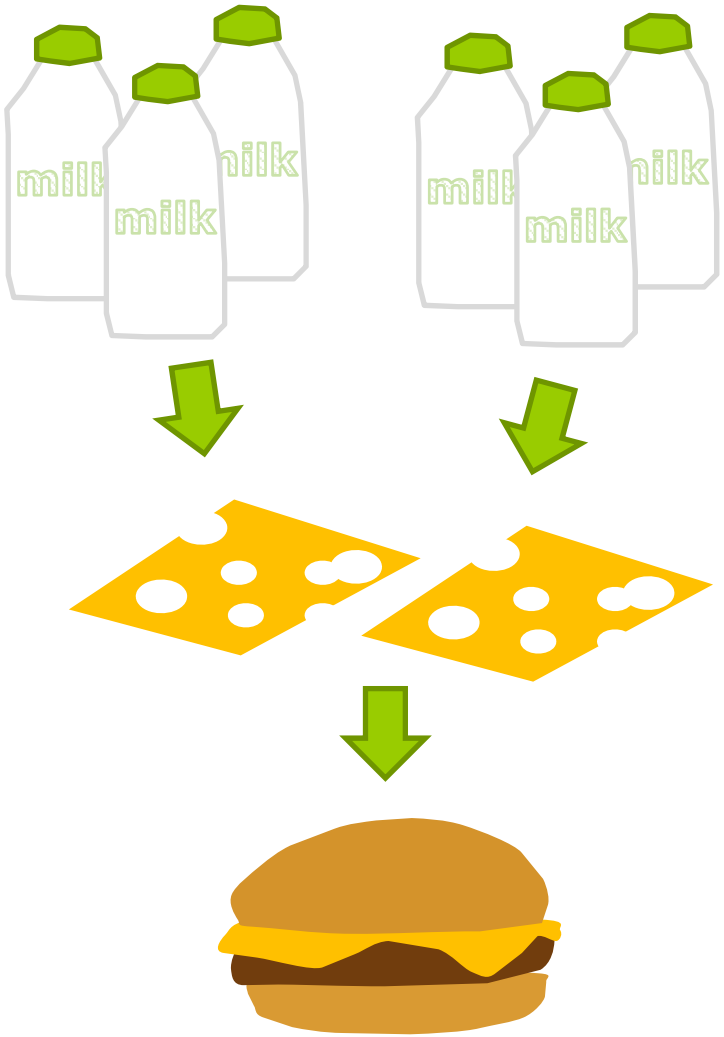
Programming Assignment #2



3 Milk producers

int Buffer_milk[9] = {0, }; // shared by all producers

2 Cheese producers

int Buffer_cheese[4] = {0, }; // shared by all producers

1 cheeseburger producer

Program Description

- You must create 6 threads in your program: 3 milk producer threads, 2 cheese producer threads, and 1 cheeseburger producer thread.
- When your program is started, user will type in the number of cheeseburgers.
- Then, your program calculates the required numbers of items for each thread. (see the previous slide)
  - e.g. To make 1 cheeseburgers => 2 slices of cheese => 6 bottles of milk
- Pass the ID of each producer and the number of items assigned to each thread: IDs of milk producers (1, 2, 3), cheese producers (4, 5)
  - pthread_create(&tid, NULL, runner, (void*) args);
- The main program waits for the completion of threads using: pthread_join(&tid, NULL)

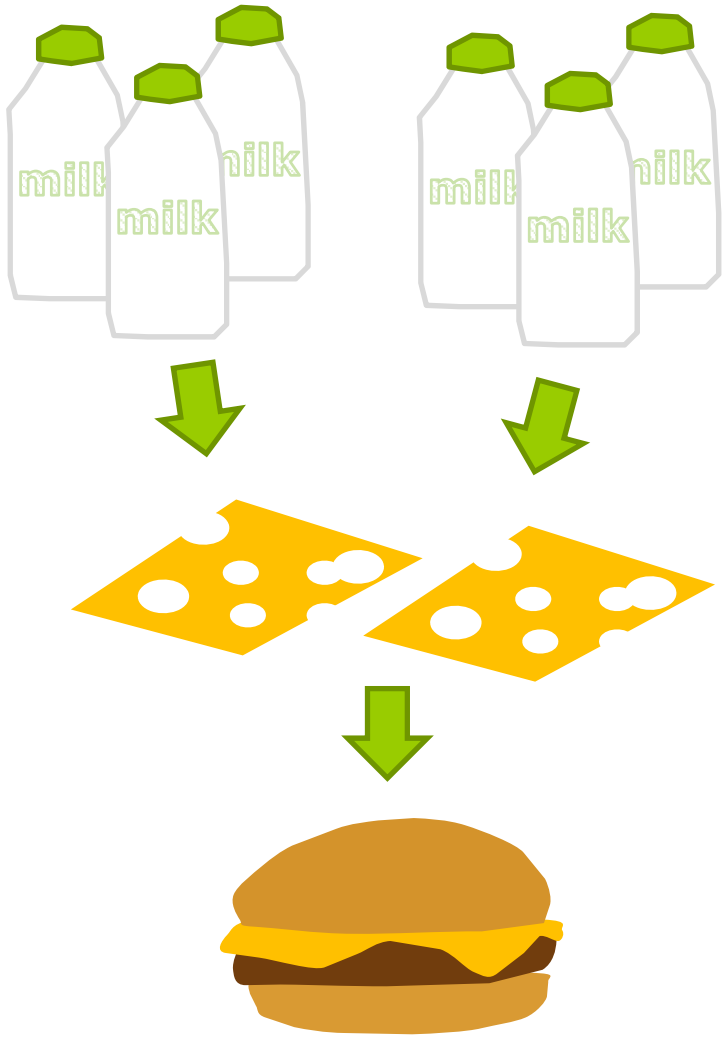
- Input of the program:
  - How many burgers do you want?

- Create 3 milk_producer threads
  - Arguments: producer ID (1, 2, 3), number of items to produce
  - Produce a bottle of milk and put it in the refrigerator (buffer_milk: size 9)
  - The buffer access (by milk_producer, cheese_producer) is protected by a mutex lock (use a semaphore to implement mutex)
    - Can produce one item per buffer access
    - 0 indicates no item; 1, 2 and 3 represent items produced by the producer IDs 1, 2, and 3, respectively
  - Use semaphores to synchronize milk production (three) and consumption (by cheese_producer)
  - The producer waits if the buffer is full (use a semaphore and carefully set an initial value)

- Create 2 cheese_producer threads
  - Arguments: producer ID (4, 5), number of items to produce
  - If there are less than three milk bottles, the thread waits (use semaphore)
  - If there are three available milk bottles in the “buffer_milk”, produce a slice of cheese
    - The cheese slice is represented by a 4-digit integer value
    - Three milk bottles are from the producers 2, 1, 3 and if the current cheese producer ID is 4 then 2134 is the produced item
    - The cheese_buffer is protected by a mutex lock implemented by a semaphore. (Only one thread can access the buffer at a time.)
  - The cheese_producer is a consumer (of milk) and a producer

- Create 1 cheeseburger_producer threads
  - Arguments: number of items to produce
  - Cheeseburger_producer is the consumer of cheese slices
  - If there are less than two slices of cheese, it waits (use semaphore)
  - If there are two or more cheese slices, produce a cheeseburger
  - Print the cheeseburger ID: concatenation of two cheese slice IDs
    - e.g. 213431125 is made with two slices of cheese 2134 and 1125
- The main thread waits for the termination of threads (three milk producers, two cheese producers, one cheeseburger producer) using pthread_join() operation
- Output of the program:
  - Printed cheeseburger IDs

Programming Assignment #2



3 Milk producers (ID: 1, 2, 3)

int Buffer_milk[9] = {0, }; // shared by all producers

2	3	1	3	2	2			

2 Cheese producers (ID: 4, 5)

int Buffer_cheese[4] = {0, }; // shared by all producers

2314	3225		

Cheese ID: three milk IDs (3digits) cheese producer ID (1digit)

1 cheeseburger producer

Cheeseburger ID: Cheese ID + Cheese ID
e.g. 23143225

What to turn in

- Submit a zip file (filename: firstname_lastname.zip) containing the following files:
  - Your source and header files (e.g., ‘c’ or ‘cpp’ file)
  - A makefile (if you used one)
  - All other files that are needed to run your program
- **A report** in which you have to include:
  - (1) a description of how to run your code,
  - (2) a **‘brief’** description of your program design, and
  - (3) a screenshot of your project management tool.
- Submit your zip file to a dropbox folder called ‘Programming Assignment 2’ on D2L.

Evaluation Criteria

- Your project will be evaluated based on the following:
  - **Documentation 10%** - your report clearly describes how to run your code (including screenshot(s)), and your thought process to address the project requirements.
  - **Project Management Tool 10%** - you have learned and used a project management tool to manage all your files for this assignment; a screenshot of your file revision history has been included in your report.
  - **Compilation 5%** - must compile without errors or warnings.
  - **Correctness 70%**
  - **Readability and Misc. 5%** - your source code must be well commented. Your zip file contains all files that are needed to run your program.