OPERATING SYSTEMS TUTORIAL 5

Question

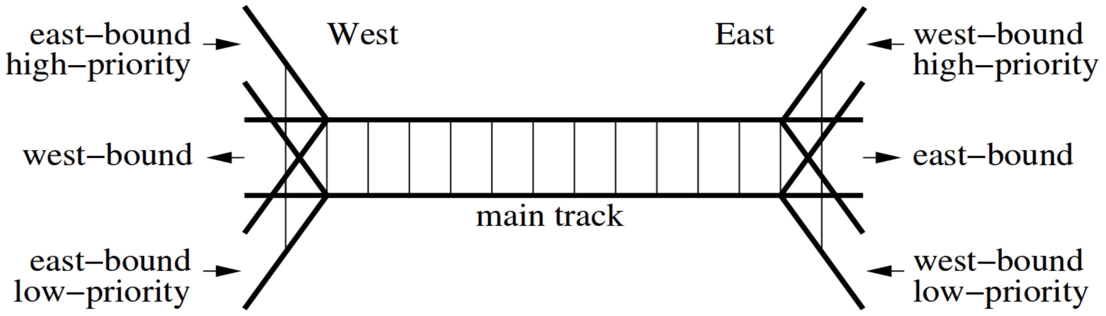
- mutex: error_check -> tryLock
- wait (conv_var, mutex)
- why need condition variable

Outline

- **P2 Specification go-through**
- Input, output & functions
- Design hints
  - Create Trains
  - Start Trains
  - Run Dispatcher

Multi-Thread Scheduling (MTS)

- An automated control system for the railway track
  - Two stations (for high and low priority trains) on each side of the main track.
  - At each station, one or more trains will be loading with commodities.
  - Each train commences its loading process at a common start time 0 of the simulation.
  - Some trains take more time to load, some less.
  - After train is loaded, it patiently awaits permission to cross the main track, subject to the requirements specified in the next slides.
  - After a train finishes crossing, it magically disappears.



Rules

The rules enforced by the automated control system are:

1. Only one train is on the main track at any given time.
2. Only loaded trains can cross the main track.
3. If there are multiple loaded trains, the one with the high priority crosses.
4. If two loaded trains have the same priority, then:
   a) If they are both traveling in the same direction, the train which finished loading first gets the clearance to cross first. If they finished loading at the same time, the one appeared first in the input file gets the clearance to cross first.
   b) If they are traveling in opposite directions, pick the train which will travel in the direction opposite of which the last train to cross the main track traveled. If no trains have crossed the main track yet, the Eastbound train has the priority.
   c) If there are four trains in the same direction traveled through the main track back to back, the trains waiting in the opposite direction get a chance to dispatch one train if any.

Train No.	Priority	Direction	Loading Time	Crossing Time
0	low	East	1.0s	0.6s
1	high	West	0.6s	0.7s
2	high	East	0.3s	1.0s

Train No.	Priority	Direction	Loading Time	Crossing Time
0	low	East	1.0s	0.6s
1	high	West	0.6s	0.7s
2	high	East	0.3s	1.0s

```
00:00:00.3 Train  2 is ready to go East
00:00:00.3 Train  2 is ON the main track going East
00:00:00.6 Train  1 is ready to go West
00:00:01.0 Train  0 is ready to go East
00:00:01.3 Train  2 is OFF the main track after going East
00:00:01.3 Train  1 is ON the main track going West
00:00:02.0 Train  1 is OFF the main track after going West
00:00:02.0 Train  0 is ON the main track going East
00:00:02.6 Train  0 is OFF the main track after going East
```

Deliverables

1. Design document (due: Feb. 28, 2022)
2. Codes (due: Mar. 11, 2022)

Deliverable 1 - A design document Due: Feb 28, 2022

1. How many threads are you going to use? Specify the work that you intend each thread to perform.
2. Do the threads work independently? Or, is there an overall “controller” thread?
3. How many mutexes are you going to use? Specify the operation that each mutex will guard.
4. Will the main thread be idle? If not, what will it be doing?
5. How are you going to represent stations (which are collections of loaded trains ready to depart)? That is, what type of data structure will you use?
6. How are you going to ensure that data structures in your program will not be modified concurrently?

Deliverable 1 - cont'd

7. How many convars are you going to use? For each convar:

   (a) Describe the condition that the convar will represent.

   (b) Which mutex is associated with the convar? Why?

   (c) What operation should be performed once pthread_cond_wait() has been unblocked and re-acquired the mutex?

8. In 15 lines or less, briefly sketch the overall algorithm you will use. You may use sentences such as:

   If train is loaded, get station mutex, put into queue, release station mutex.

Note:

2 pages maximum, pdf.

Bonus features before/in Deliverable 1.

Deliverable 2 - Code Due: March 11, 2022

1. The name of the submission file must be p2.tar.gz.
2. p2.tar.gz must contain all your files in a directory named p2
3. Inside the directory p2, there must be a Makefile. Also there shall be a test input file created by you.
4. Invoking make on it must result in an executable named mts being built, without user intervention.
5. You may not submit the assignment with a compiled executable and/or object (.o) files; the script will delete them before invoking make.

Outline

- P2 Specification go-through
- **Input, output & functions**
- Design hints
  - –Create Trains
  - –Start Trains
  - –Run Dispatcher

Input & output

Input file

```
e 10 6
W 6 7
E 3 10
```

Each train, which will be simulated by a thread

- Direction, Priority, Loading Time, Crossing Time
- Program simulates train crossing delays in 10ths of a second

Train No.	Priority	Direction	Loading Time	Crossing Time
0	low	East	1.0s	0.6s
1	high	West	0.6s	0.7s
2	high	East	0.3s	1.0s

./mts input.txt 3

```
00:00:00.3 Train  2 is ready to go East
00:00:00.3 Train  2 is ON the main track going East
00:00:00.6 Train  1 is ready to go West
00:00:01.0 Train  0 is ready to go East
00:00:01.3 Train  2 is OFF the main track after going East
00:00:01.3 Train  1 is ON the main track going West
00:00:02.0 Train  1 is OFF the main track after going West
00:00:02.0 Train  0 is ON the main track going East
00:00:02.6 Train  0 is OFF the main track after going East
```

Functions

1. File access functions:
   (a) `atoi`
   (b) `fopen`
   (c) `feof`
   (d) `fgets` and `strtok`
   (e) `fclose`

   and more efficiently you can use fscanf

2. Thread creation functions:
   (a) `pthread_create`
   (b) `pthread_exit`
   (c) `pthread_join`

3. Mutex manipulation functions:
   (a) `pthread_mutex_init`
   (b) `pthread_mutex_lock`
   (c) `pthread_mutex_unlock`

4. Condition variable manipulation functions:
   (a) `pthread_cond_init`
   (b) `pthread_cond_wait`
   (c) `pthread_cond_broadcast`
   (d) `pthread_cond_signal`

Outline

- P2 Specification go-through
- Input, output & functions
- **Design hints**
  - –Create Trains
  - –Start Trains
  - –Run Dispatcher

Create Trains

- Checking the Input Argument.
- **Load** the Input File & **Create/Initialize** Train Objects.
- Simulation clock

```
clock_gettime (CLOCK_MONOTONIC,  &begin)
clock_gettime (CLOCK_MONOTONIC,  &end)
```

Start Trains

- Loading
- Ready
- Granted
- Crossing
- Gone

Run Dispatcher

- If all trains are “Ready”…
- If all trains have “Gone”…
- If there exists a train whose status is “Granted”…
- If there is no train whose status is “Ready”…
- …