BIL461 - Operating Systems

Homework 3: Ferry Crossing

1 Overview

Synchronization is one of the fundamental concepts in Operating Systems which is about organization of threads, processes needing communication with each other. Synchronization can be applied with IPC, shared memory or locks. In this assignment, you will implement Ferry Crossing Problem which is one of the barrier synchronization problems using threads and locks.

The program will be invoked as follows:

```
./ferry_cross
```

The program will be implemented using threads and locks

2 Functional Requirements

- sem_board: A "ticket" to enter. The Ferry "gives" these tickets (signals). Cars "take" them (wait).
- sem_full: A "ready to go" button. The Ferry waits for this button to be pressed. The last car to enter presses (signals) it.
- sem_unboard: A "safe to exit" light. The Ferry turns this on (signals) when it docks. Cars wait for this light.
- sem_empty: A "clean sweep" confirmation. The Ferry waits for this. The last car to leave presses (signals) it.
- mutex: A lock to ensure that when cars count themselves (cars_on_board++ or --), two cars don't modify the number at the exact same time.

- The Ferry controls when cars enter.
- The Cars control when the Ferry leaves (by telling it the boat is full).
- The Ferry controls when cars leave.
- The Cars control when the Ferry resets (by telling it the boat is empty).

The ferry will initially wait at a dock. The cars will be at dock randomly (uniformly) between 0 and 1 seconds. For example, first car will be at dock at 0.4-th seconds, second car will be at dock at 0.7-th seconds (0.4 + 0.3), third car will be at dock at 1.3-th seconds (0.4 + 0.3 + 0.6)

After ferry is fulled, it will leave the dock and will travel to the new dock by 3 seconds. Random generation will stop, new cars will not be waiting at the new dock in this travel time. After ferry is arrived to new dock, the cars will leave the ferry, and random generation will be stopping in this duration too. After all cars leave the ferry, cars will be waiting again at the new dock randomly (random generation will start). Ferry will travel between multiple docks like this process. Program will be terminated after 1 minute.

3 Output Requirements

Standard Output (stdout):

[Clock : $CLOCK] Ferry arrives to new dock
[Clock : $CLOCK] Car $CAR_NUM entered the ferry
[Clock : $CLOCK] Car $CAR_NUM entered the ferry
[Clock : $CLOCK] Car $CAR_NUM entered the ferry
[Clock : $CLOCK] Car $CAR_NUM entered the ferry
[Clock : $CLOCK] Car $CAR_NUM entered the ferry
[Clock : $CLOCK] Ferry leaves the dock
[Clock : $CLOCK] Ferry arrives to new dock
[Clock : $CLOCK] Car $CAR_NUM left the ferry
[Clock : $CLOCK] Car $CAR_NUM left the ferry
[Clock : $CLOCK] Car $CAR_NUM left the ferry
[Clock : $CLOCK] Car $CAR_NUM left the ferry
[Clock : $CLOCK] Car $CAR_NUM left the ferry
[Clock : $CLOCK] Car $CAR_NUM entered the ferry
[Clock : $CLOCK] Car $CAR_NUM entered the ferry
[Clock : $CLOCK] Car $CAR_NUM entered the ferry
[Clock : $CLOCK] Car $CAR_NUM entered the ferry
[Clock : $CLOCK] Car $CAR_NUM entered the ferry
[Clock : $CLOCK] Ferry leaves the dock
[Clock : $CLOCK] Ferry arrives to new dock

4 Technical Constraints

Program will be written in C.
stdio, stdlib, unistd, pthread, semaphore, sys/time, errno, stdbool libraries can be used.
Other libraries are forbidden.
All system calls must be checked for errors. On failure, print a descriptive message to ‘stderr‘ via ‘perror‘ and exit with a non-zero status code.

5 Grading

Preconditions (15 Points)

Code compiles with ‘make‘ follows all technical constraints and it is runnable: 15

Functional Requirements (85 Points)

Thread Management (25 pts)

General car and ferry threads management-synchronization and random generation implementation

Output and Correctness (60 pts)

Correct terminal result and format

No points will be given if program does not satisfy preconditions. Deductions of up to 10 points for poor code style or lack of comments.

6 Compilation Requirement

You must provide a Makefile. Running make should compile your source code into an executable named ferry_cross.

Example Makefile :

```
CC = gcc
CFLAGS = -Wall -Wextra std=c99
TARGET = ferry_cross
SOURCES = ferry_cross.c

$(TARGET): $(SOURCES)
    $(CC) $(CFLAGS) -o $(TARGET) $(SOURCES)

clean:
    rm -f $(TARGET)

.PHONY: clean
```

7 Submission Guidelines

- Submit all your source code files (e.g., ferry_cross.c, any .h files) together with the make file.
- Do not submit compiled object files or executables.
- Package all files into a single .zip or .tar.gz archive named <your_student_id>_hw3.zip (or .tar.gz).
- Ensure your code compiles and runs.

8 Late Submission Policy

Maximum points you can get if you do a late submission :

- 1 day: 80 points out of 100.
- 2 days: 60 points out of 100.
- 3 days: 30 points out of 100.
- 4 days or more: 0 points out of 100.

9 Academic Integrity

You can use LLMs as a study partner while learning throughout the homework HOWEVER, all submitted work must be your own. Plagiarism detection software will be used if the source code is suspicious after it is reviewed. Do not share your code with others or copy code from online sources. Any instance of academic dishonesty will be dealt with according to university policy and grading of 0.