Gebze Technical University

Department of Computer Engineering

CSE344 System Programming

Homework 3 Report

Burak Yıldırım
1901042609

Contents

1 Introduction **3**
- 1.1 Project Description . . . 3
- 1.2 Compilation . . . 3

2 General Structure **4**

3 Implementation **4**
- 3.1 Structs and Enums . . . 4
- 3.2 Macros . . . 5
- 3.3 Global Variables . . . 5
- 3.4 Steps . . . 6
- 3.5 Details . . . 6
  - 3.5.1 carOwner . . . 6
  - 3.5.2 carAttendant . . . 7
  - 3.5.3 Signal Handling . . . 7

4 Valgrind Results **8**
- 4.1 Memory Leak Check . . . 8
  - 4.1.1 Normal Execution . . . 8
  - 4.1.2 Execution with SIGINT . . . 8
- 4.2 Thread Error Check . . . 8
  - 4.2.1 Normal Execution . . . 8
  - 4.2.2 Execution with SIGINT . . . 9

5 Testing **9**
- 5.1 Happy Path With 12 Cars . . . 9
- 5.2 No Available Spot . . . 10

1 Introduction

1.1 Project Description

The task of this project is to design and implement a parking lot system with finite capacity using threads and semaphores.

1.2 Compilation

```
CC = gcc
CFLAGS = -w
DFLAGS = -g
TFLAGS = -pthread
INCDIR = include
SRCDIR = src
TARGET = main
SOURCES = $(wildcard $(SRCDIR)/*.c) $(wildcard $(SRCDIR)/**/*.c)
HEADERS = $(wildcard $(INCDIR)/*.h) $(wildcard $(INCDIR)/**/*.h)
VALGRIND_MEMORY_OPTIONS = --leak-check=full --show-leak-kinds=all --track-origins=yes
VALGRIND_THREAD_OPTIONS = --tool=helgrind

all: $(TARGET)

$(TARGET): $(SOURCES) $(HEADERS)
	$(CC) $(CFLAGS) -o $@ $(SOURCES) -I$(INCDIR) $(TFLAGS)

debug: CFLAGS += $(DFLAGS)
debug: $(TARGET)

clean:
	rm -f $(TARGET)

valgrind_memory: debug
	valgrind $(VALGRIND_MEMORY_OPTIONS) ./$(TARGET)

valgrind_thread: debug
	valgrind $(VALGRIND_THREAD_OPTIONS) ./$(TARGET)
```

The provided Makefile automates the build process: executing **make** compiles the project, **make clean** removes the executable, **make valgrind_memory** launches the programs in Valgrind for memory leak checks, and **make valgrind_thread** launches the programs in Valgrind for thread errors.

2 General Structure

```
Midterm
├── include
│   ├── utils
│   │   ├── array.h
│   │   ├── io.h
│   │   ├── string.h
│   │   └── type.h
│   └── main.h
├── src
│   ├── utils
│   │   ├── array.c
│   │   ├── io.c
│   │   └── string.c
│   └── main.c
└── Makefile
```

This is the folder structure of the project.

- utils: utility functions and macros used by multiple files.
- main: main functions and macros.
- Makefile: compile project and clean executables, logs and lock files.

3 Implementation

3.1 Structs and Enums

```c
typedef enum {
    PARK,
    RETRIEVE
} Action;

typedef enum {
    AUTOMOBILE,
    PICKUP
} Car;

typedef enum {
    FALSE = 0,
    TRUE = 1
} Bool;
```

```c
typedef struct {
    Car type;
    int id;
    int time;
} CarOwner;
```

3.2 Macros

- `MAX_OUTPUT`: Sets the maximum output buffer size to 256 characters.
- `MAX_AUTOMOBILE_SPOT`: Sets the maximum number of automobile parking spots to 8.
- `MAX_PICKUP_SPOT`: Sets the maximum number of pickup parking spots to 4.
- `MAX_CAR_OWNER`: Sets the maximum number of car owner threads to 2.
- `MAX_ATTENDANT`: Sets the maximum number of car attendant threads to 2.
- `MAX_CAR`: Sets the maximum number of cars to be generated to 12.

3.3 Global Variables

- `Bool isFinished`: Flag to indicate there are no more car owners. Initial value is `FALSE`.
- `int mFree_automobile`: Number of free automobile spots. Initial value is `MAX_AUTOMOBILE_SPOT`.
- `int mFree_pickup`: Number of free pickup spots. Initial value is `MAX_PICKUP_SPOT`.
- `int mLeft_cars`: Number of cars that left because there were no spots available. Initial value is 0.
- `CarOwner gCarOwner`: Global variable to store the current car owner.
- `CarOwner parked`: Array to store the parked cars. It's size is `MAX_AUTOMOBILE_SPOT + MAX_PICKUP_SPOT`.
- `sem_t mutex`: Semaphore used to limit access to global variables to one thread at a time.
- `sem_t print_mutex`: Semaphore used to limit output to one thread at a time.
- `sem_t newAutomobile`: Semaphore used to signal that a new automobile has arrived.
- `sem_t newPickup`: Semaphore used to signal that a new pickup has arrived.
- `sem_t inChangeforAutomobile`: Semaphore used to signal that the car attendant for automobile is changing the automobile spots.
- `sem_t inChangeforPickup`: Semaphore used to signal that the car attendant for pickup is changing the pickup spots.
- `sem_t retrieveCar`: Semaphore used to signal that a car owner wants to retrieve their car.

3.4 Steps

The main program is implemented following these steps:

1. `SIGINT` is handled.
2. Random number generator is seeded with the current time.
3. A carAttendant for automobile is created using `pthread_create`.
4. A carAttendant for pickup is created `pthread_create`.
5. A carOwner for parking is created `pthread_create`.
6. A carOwner for retrieving is created `pthread_create`.
7. carOwner threads are waited using `pthread_join`.
8. carAttendant threads are cancelled and joined using `pthread_cancel` and `pthread_join`.
9. `cleanup` function is called to destroy all semaphores.
10. Program finishes.

3.5 Details

All access to the global variables are done between `sem_wait(&mutex)` and `sem_post(&mutex)`.

3.5.1 carOwner

carOwner function takes an Action type argument and according to its value it either parks a car or retrieves a car. The parking carOwner runs in a loop for `MAX_CAR` times with 1 second delays to simulate staggered arrival of cars. At each iteration it randomly generates a Car and checks if the generated Car type has an available space. If not, it increases the `mLeft_cars` and continues to the next iteration. If there is an available space, it updates the `gCarOwner` with type of the generated Car as its type, current index of the loop `i` as its id, and `i` plus a random integer between 1 and 5 as its wait time. `i` is added to the wait time to calculate the time passed from the start of the program in order to simulate async wait, e.g. a car that arrived at the 3^{rd} second and waits for 1 second will be retrieved at the same time as a car that arrived at the 1^{st} second and waits for 3 seconds. Then it posts `newAutomobile`, and waits the carAttendant's response with `inChangeforAutomobile`. After that it posts the `retrieveCar` to notify the retrieving carOwner. After the loop is finished it changes the `isFinished` to `TRUE`. The retrieving carOwner first initializes the integer variable `waitedTime` to 0. It then runs in a loop unless the `isFinished` is `TRUE` and the first element of the `parked` is not valid, e.g. `parked[0].time` is 0. At each iteration it waits `retrieveCar`, then sleeps for `timeToSleep = parked[0].time - waitedTime` seconds to simulate async wait for each parked car. If `timeToSleep` is negative it means that there is at least one car that is overdue, so `timeToSleep` is set to 0. After sleeping, it retrieves all cars from the parking lot that have the same wait time as `parked[0]` by shifting the `parked` to the left and setting the wait time of the last element to 0, and increases the number of free spots for each retrieved car's type. After retrievals it adds the `timeToSleep` to the `waitedTime`.

3.5.2 carAttendant

carAttendant function takes a Car type argument and according to its value it either manages automobiles or pickups. carAttendants run in an infinite loop. At each iteration, they wait for their respective new car semaphore, e.g. newAutomobile for automobile and newPickup for pickup. Then they calculate the index to insert the `gCarOwner` to the `parked`. Index points to the first empty spot in the `parked`. After that, the `parked` is sorted according to the wait time of its elements. This way it's ensured that the ones with the minimum wait time are retrieved first no matter the order they are inserted. Then they decrement their respective free car spot number, e.g. `mFree_automobile` for automobile and `mFree_pickup` for pickup. Finally they post their respective in change for car semaphores, e.g. `inChangeforAutomobile` for automobile and `inChangeforPickup` for pickup.

3.5.3 Signal Handling

SIGINT

```
void sigint_handler(int sig) {
    for (int i = 0; i < MAX_CAR_OWNER; i++) {
        pthread_cancel(carOwners[i]);
        pthread_join(carOwners[i], NULL);
    }

    for (int i = 0; i < MAX_ATTENDANT; i++) {
        pthread_cancel(carAttendants[i]);
        pthread_join(carAttendants[i], NULL);
    }

    cleanup();
    my_printf("\nReceived SIGINT. Exiting...\n");
    exit(EXIT_SUCCESS);
}
```

In the `SIGINT` handler, all threads are cancelled and joined. Then, the `cleanup` function is called to destroy all semaphores. Finally, the program exits successfully.

4 Valgrind Results

4.1 Memory Leak Check

Command:
`valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes ./main`

4.1.1 Normal Execution

```
==24444==
==24444== HEAP SUMMARY:
==24444==     in use at exit: 0 bytes in 0 blocks
==24444==   total heap usage: 9 allocs, 9 frees, 2,782 bytes allocated
==24444==
==24444== All heap blocks were freed -- no leaks are possible
==24444==
==24444== For counts of detected and suppressed errors, rerun with: -v
==24444== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

4.1.2 Execution with SIGINT

```
^C
Received SIGINT. Exiting...
==24450==
==24450== HEAP SUMMARY:
==24450==     in use at exit: 0 bytes in 0 blocks
==24450==   total heap usage: 9 allocs, 9 frees, 2,782 bytes allocated
==24450==
==24450== All heap blocks were freed -- no leaks are possible
==24450==
==24450== For counts of detected and suppressed errors, rerun with: -v
==24450== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

4.2 Thread Error Check

Command:
`valgrind --tool=helgrind ./main`

4.2.1 Normal Execution

```
==24465==
==24465== For counts of detected and suppressed errors, rerun with: -v
==24465== Use --history-level=approx or =none to gain increased speed, at
==24465== the cost of reduced accuracy of conflicting-access information
==24465== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 230 from 29)
```

4.2.2 Execution with SIGINT

```
^C
Received SIGINT. Exiting...
==24471==
==24471== For counts of detected and suppressed errors, rerun with: -v
==24471== Use --history-level=approx or =none to gain increased speed, at
==24471== the cost of reduced accuracy of conflicting-access information
==24471== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 386 from 29)
```

5 Testing

5.1 Happy Path With 12 Cars

```
Owner 1 (pickup) arrives
Attendant parked the pickup of owner 1. Remaining pickup spots: 3
Owner 2 (automobile) arrives
Attendant parked the automobile of owner 2. Remaining automobile spots: 7
Owner 1 retrieved their pickup after 2 seconds. Remaining pickup spots: 4
Owner 3 (pickup) arrives
Attendant parked the pickup of owner 3. Remaining pickup spots: 3
Owner 4 (pickup) arrives
Attendant parked the pickup of owner 4. Remaining pickup spots: 2
Owner 3 retrieved their pickup after 1 second. Remaining pickup spots: 3
Owner 2 retrieved their automobile after 3 seconds. Remaining automobile spots: 8
Owner 5 (automobile) arrives
Attendant parked the automobile of owner 5. Remaining automobile spots: 7
Owner 4 retrieved their pickup after 2 seconds. Remaining pickup spots: 4
Owner 6 (automobile) arrives
Attendant parked the automobile of owner 6. Remaining automobile spots: 6
Owner 7 (automobile) arrives
Attendant parked the automobile of owner 7. Remaining automobile spots: 5
Owner 5 retrieved their automobile after 3 seconds. Remaining automobile spots: 6
Owner 7 retrieved their automobile after 1 second. Remaining automobile spots: 7
Owner 8 (automobile) arrives
Attendant parked the automobile of owner 8. Remaining automobile spots: 6
Owner 6 retrieved their automobile after 3 seconds. Remaining automobile spots: 7
Owner 8 retrieved their automobile after 1 second. Remaining automobile spots: 8
Owner 9 (automobile) arrives
Attendant parked the automobile of owner 9. Remaining automobile spots: 7
Owner 10 (pickup) arrives
Attendant parked the pickup of owner 10. Remaining pickup spots: 3
Owner 9 retrieved their automobile after 1 second. Remaining automobile spots: 8
Owner 11 (pickup) arrives
Attendant parked the pickup of owner 11. Remaining pickup spots: 2
Owner 12 (pickup) arrives
Attendant parked the pickup of owner 12. Remaining pickup spots: 1
Owner 10 retrieved their pickup after 3 seconds. Remaining pickup spots: 2
Owner 11 retrieved their pickup after 5 seconds. Remaining pickup spots: 3
Owner 12 retrieved their pickup after 4 seconds. Remaining pickup spots: 4
```

5.2 No Available Spot

For this test case, I deliberately changed the maximum waiting time of 1-5 seconds to 5-9 seconds.

```
Owner 1 (pickup) arrives
Attendant parked the pickup of owner 1. Remaining pickup spots: 3
Owner 2 (pickup) arrives
Attendant parked the pickup of owner 2. Remaining pickup spots: 2
Owner 3 (automobile) arrives
Attendant parked the automobile of owner 3. Remaining automobile spots: 7
Owner 4 (pickup) arrives
Attendant parked the pickup of owner 4. Remaining pickup spots: 1
Owner 5 (automobile) arrives
Attendant parked the automobile of owner 5. Remaining automobile spots: 6
Owner 6 (automobile) arrives
Attendant parked the automobile of owner 6. Remaining automobile spots: 5
Owner 7 (pickup) arrives
Attendant parked the pickup of owner 7. Remaining pickup spots: 0
No spot available for pickup. Owner 8 leaves.
No spot available for pickup. Owner 9 leaves.
Owner 2 retrieved their pickup after 6 seconds. Remaining pickup spots: 1
Owner 1 retrieved their pickup after 9 seconds. Remaining pickup spots: 2
Owner 4 retrieved their pickup after 6 seconds. Remaining pickup spots: 3
Owner 10 (pickup) arrives
Attendant parked the pickup of owner 10. Remaining pickup spots: 2
Owner 3 retrieved their automobile after 8 seconds. Remaining automobile spots: 6
Owner 5 retrieved their automobile after 6 seconds. Remaining automobile spots: 7
Owner 11 (pickup) arrives
Attendant parked the pickup of owner 11. Remaining pickup spots: 1
Owner 12 (automobile) arrives
Attendant parked the automobile of owner 12. Remaining automobile spots: 6
Owner 6 retrieved their automobile after 9 seconds. Remaining automobile spots: 7
Owner 10 retrieved their pickup after 5 seconds. Remaining pickup spots: 2
Owner 7 retrieved their pickup after 9 seconds. Remaining pickup spots: 3
Owner 11 retrieved their pickup after 8 seconds. Remaining pickup spots: 4
Owner 12 retrieved their automobile after 9 seconds. Remaining automobile spots: 8
```