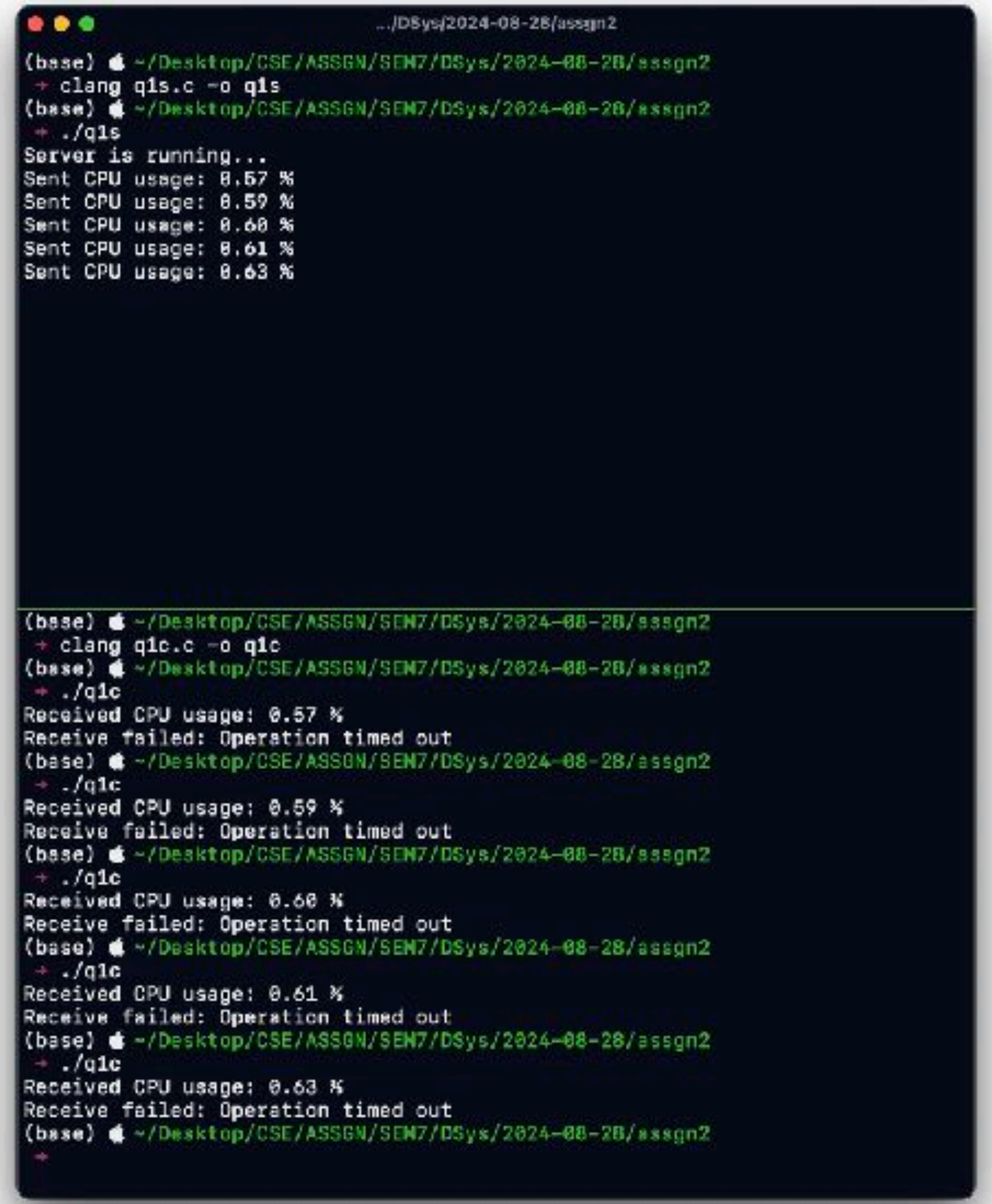
ASSIGNMENT 2
02/09/24

NAME : SHRESTH SONKAR
REGNO : 20214272
GROUP : CS7D
TOPIC : DISTRIBUTED SYSTEM
CODE : CS-17201

```c
//q1c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
    int sock = 0;
    struct sockaddr_in serv_addr;
    char buffer[BUFFER_SIZE] = {0};

    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("\n Socket creation error \n");
        return -1;
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, "127.0.0.1",
&serv_addr.sin_addr) <= 0) {
        printf("\nInvalid address/ Address not
supported \n");
        return -1;
    }
    if (connect(sock, (struct sockaddr *) &serv_addr,
sizeof(serv_addr)) < 0) {
        printf("\nConnection Failed \n");
        return -1;
    }

    read(sock, buffer, BUFFER_SIZE);
    printf("Received CPU usage: %.2f %% ", buffer);
    close(sock);
    return 0;
}
```

```c
//q1s
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/socket.h>

#define PORT 8080
#define BUFFER_SIZE 1024

float get_cpu_load() {
    long double a[4], b[4], loadavg;
    FILE *fp;

    fp = fopen("/proc/stat", "r");
    if (fp == NULL) {
        perror("Failed to open /proc/stat");
        return -1;
    }

    fscanf(fp, "cpu %Lf %Lf %Lf %Lf", &a[0], &a[1],
&a[2], &a[3]);
    fclose(fp);

    sleep(1);

    fp = fopen("/proc/stat", "r");
    if (fp == NULL) {
        perror("Failed to open /proc/stat");
        return -1;
    }

    fscanf(fp, "cpu %Lf %Lf %Lf %Lf", &b[0], &b[1],
&b[2], &b[3]);
    fclose(fp);

    loadavg = ((b[0] + b[1] + b[2]) - (a[0] + a[1] +
a[2])) /
              ((b[0] + b[1] + b[2] + b[3]) - (a[0] +
a[1] + a[2] + a[3]));

    return loadavg * 100;
}
```

```c
int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[BUFFER_SIZE] = {0};

    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0))
== 0) {
        perror("Socket failed");
        exit(EXIT_FAILURE);
    }

    if (setsockopt(server_fd, SOL_SOCKET,
SO_REUSEADDR , &opt, sizeof(opt))) {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    if (bind(server_fd, (struct sockaddr *) &address,
sizeof(address)) < 0) {
        perror("Bind failed");
        exit(EXIT_FAILURE);
    }

    if (listen(server_fd, 3) < 0) {
        perror("Listen");
        exit(EXIT_FAILURE);
    }

    printf("Server started...\n");

    if ((new_socket = accept(server_fd, (struct
sockaddr *) &address, (socklen_t * ) & addrlen)) < 0) {
        perror("Accept");
        exit(EXIT_FAILURE);
    }

    float cpu_load = get_cpu_load();
    snprintf(buffer, sizeof(buffer), "Received CPU
Load: %.2f%%\n", cpu_load);

    send(new_socket, buffer, strlen(buffer), 0);
    printf("Sent CPU usage: %f %% ");

    close(new_socket);
    close(server_fd);
    return 0;
}
```

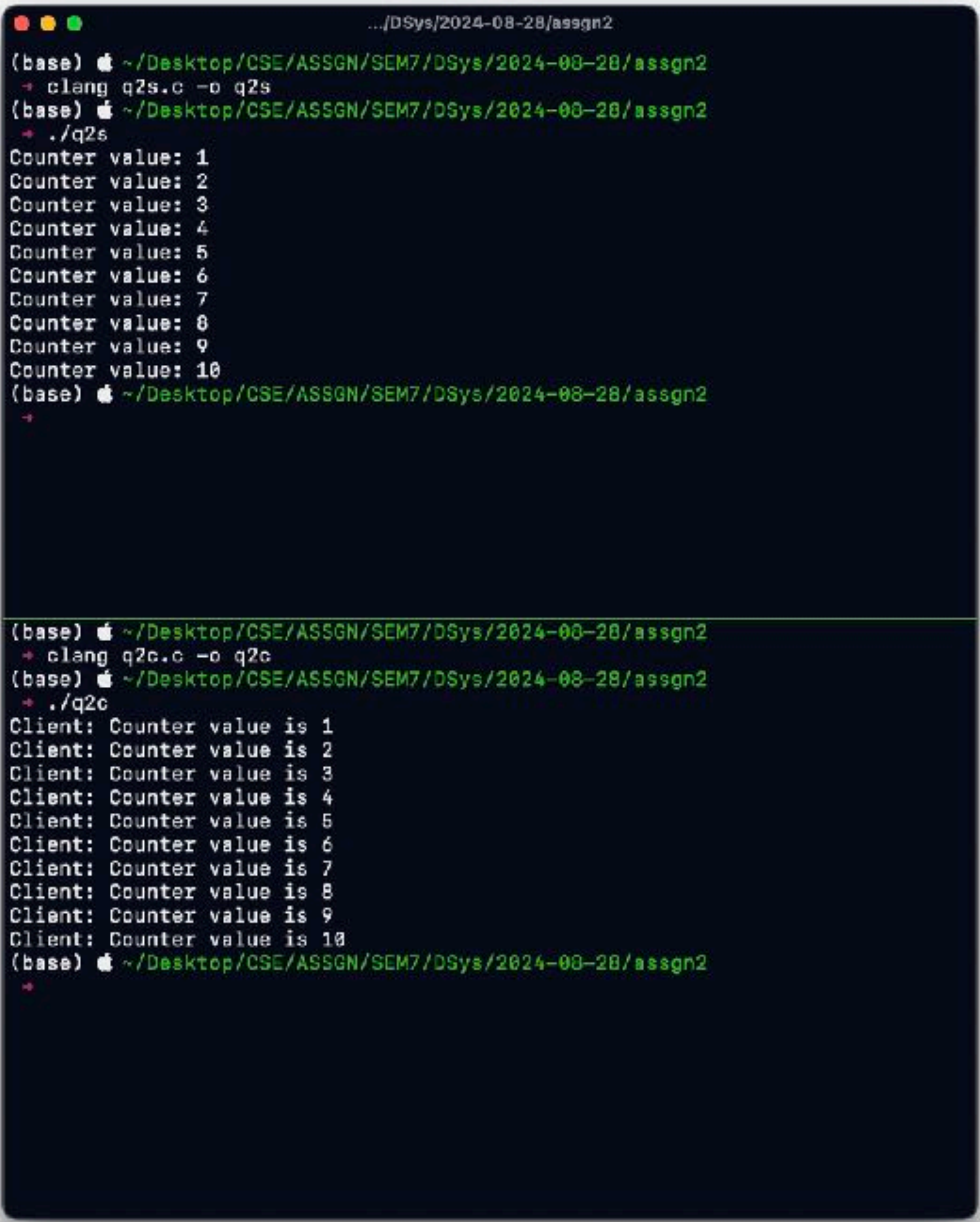


```
(base) ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
→ clang q1s.c -o q1s
(base) ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
→ ./q1s
Server is running...
Sent CPU usage: 0.57 %
Sent CPU usage: 0.59 %
Sent CPU usage: 0.60 %
Sent CPU usage: 0.61 %
Sent CPU usage: 0.63 %
```

```
(base) ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
→ clang q1c.c -o q1c
(base) ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
→ ./q1c
Received CPU usage: 0.57 %
Receive failed: Operation timed out
(base) ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
→ ./q1c
Received CPU usage: 0.59 %
Receive failed: Operation timed out
(base) ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
→ ./q1c
Received CPU usage: 0.60 %
Receive failed: Operation timed out
(base) ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
→ ./q1c
Received CPU usage: 0.61 %
Receive failed: Operation timed out
(base) ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
→ ./q1c
Received CPU usage: 0.63 %
Receive failed: Operation timed out
(base) ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
→
```

```c
//q2c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <semaphore.h>
#include <sys/stat.h>
#include <string.h>
#include <errno.h>

#define SHARED_MEM_SIZE sizeof(int)
#define SEM_NAME "/my_semaphore"

int main() {
    int fd;
    int *shared_counter;
    sem_t *sem;

    fd = shm_open("/my_shared_memory", O_RDWR, 0666);
    if (fd == -1) {
        perror("shm_open failed");
        exit(EXIT_FAILURE);
    }

    shared_counter = mmap(NULL, SHARED_MEM_SIZE,
PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
    if (shared_counter == MAP_FAILED) {
        perror("mmap failed");
        close(fd);
        exit(EXIT_FAILURE);
    }

    sem = sem_open(SEM_NAME, 0);
    if (sem == SEM_FAILED) {
        perror("sem_open failed");
        munmap(shared_counter, SHARED_MEM_SIZE);
        close(fd);
        exit(EXIT_FAILURE);
    }

    for (int i = 0; i < 10; i++) {
        sem_wait(sem);
        printf("Client: Counter value is %d\n",
*shared_counter);
```

```c
        sem_post(sem);
        sleep(1);
    }

    sem_close(sem);
    munmap(shared_counter, SHARED_MEM_SIZE);
    close(fd);
    return 0;
}

//q2s
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <semaphore.h>
#include <sys/stat.h>
#include <string.h>
#include <errno.h>

#define SHARED_MEM_SIZE sizeof(int)
#define SEM_NAME "/my_semaphore"

int main() {
    int fd;
    int *shared_counter;
    sem_t *sem;

    fd = shm_open("/my_shared_memory", O_CREAT |
O_RDWR, 0666);
    if (fd == -1) {
        perror("shm_open failed");
        exit(EXIT_FAILURE);
    }

    if (ftruncate(fd, SHARED_MEM_SIZE) == -1) {
        perror("ftruncate failed");
        close(fd);
        exit(EXIT_FAILURE);
    }

    shared_counter = mmap(NULL, SHARED_MEM_SIZE,
PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
    if (shared_counter == MAP_FAILED) {
        perror("mmap failed");
        close(fd);
        exit(EXIT_FAILURE);
    }

    *shared_counter = 0;

    sem = sem_open(SEM_NAME, O_CREAT, 0666, 1);
    if (sem == SEM_FAILED) {
        perror("sem_open failed");
        munmap(shared_counter, SHARED_MEM_SIZE);
        close(fd);
        exit(EXIT_FAILURE);
    }

    for (int i = 0; i < 10; i++) {
        sem_wait(sem);

        (*shared_counter)++;
        printf("Counter value: %d\n", *shared_counter);

        sem_post(sem);

        sleep(1);
    }

    sem_close(sem);
    sem_unlink(SEM_NAME);
    munmap(shared_counter, SHARED_MEM_SIZE);
    close(fd);
    shm_unlink("/my_shared_memory");
    return 0;
}
```



```
.../DSys/2024-08-28/assgn2

(base)  ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
 → clang q2s.c -o q2s
(base)  ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
 → ./q2s
Counter value: 1
Counter value: 2
Counter value: 3
Counter value: 4
Counter value: 5
Counter value: 6
Counter value: 7
Counter value: 8
Counter value: 9
Counter value: 10
(base)  ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
 →
```

```
(base)  ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
 → clang q2c.c -o q2c
(base)  ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
 → ./q2c
Client: Counter value is 1
Client: Counter value is 2
Client: Counter value is 3
Client: Counter value is 4
Client: Counter value is 5
Client: Counter value is 6
Client: Counter value is 7
Client: Counter value is 8
Client: Counter value is 9
Client: Counter value is 10
(base)  ~/Desktop/CSE/ASSGN/SEM7/DSys/2024-08-28/assgn2
 →
```