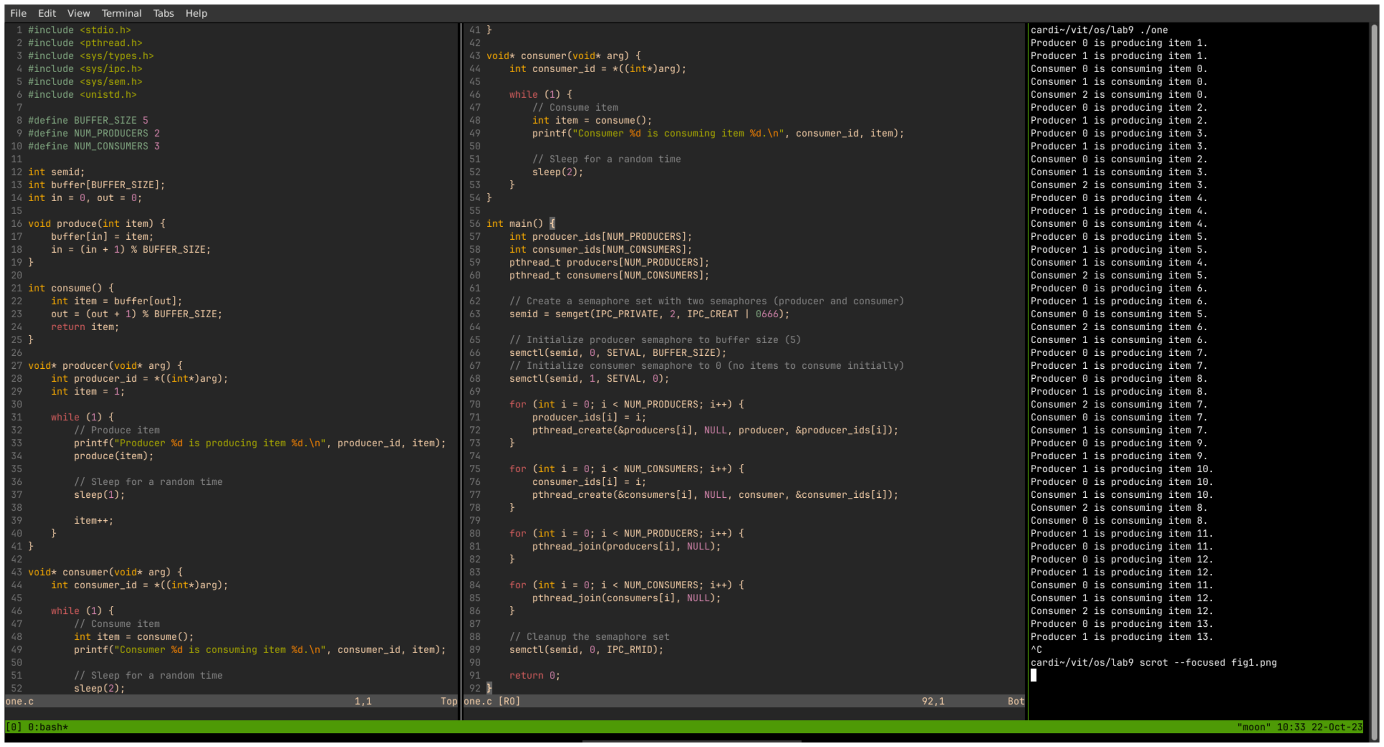
$22bps1059$

$first$



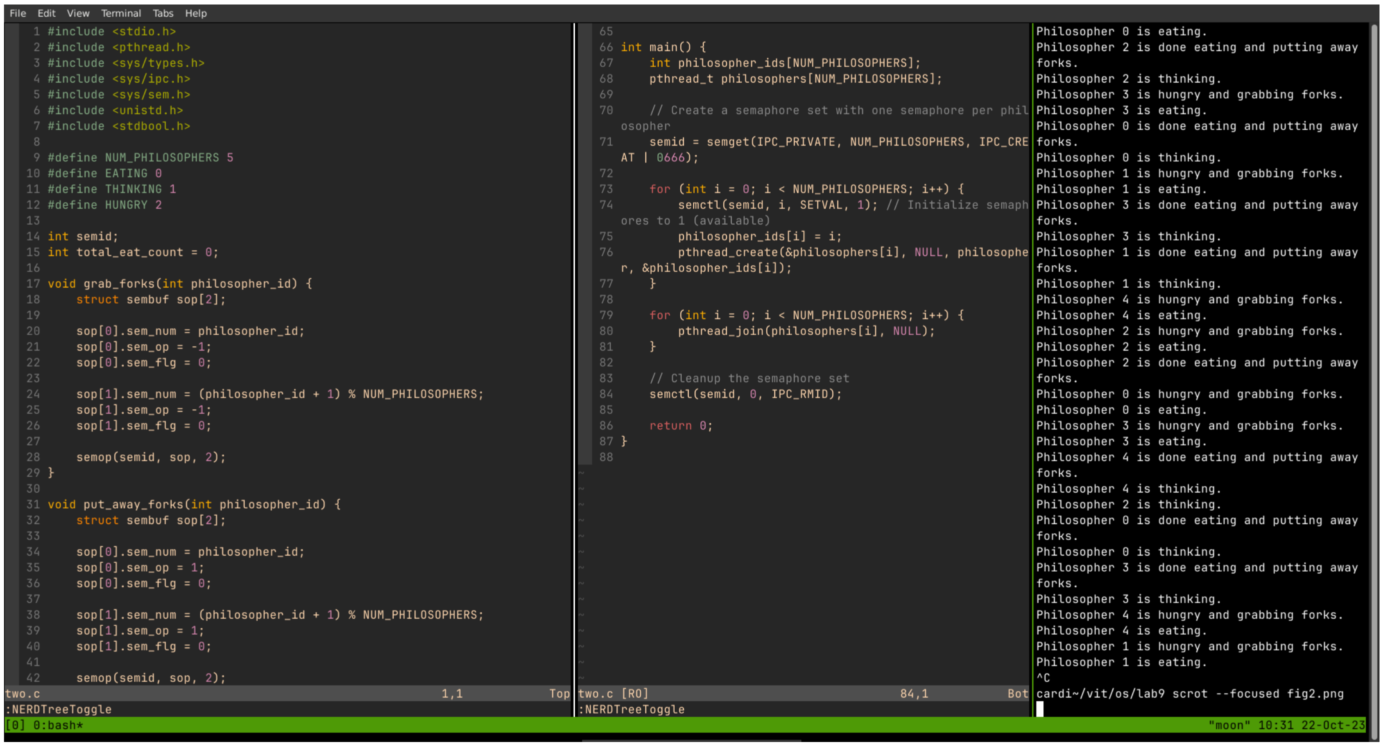
```c
#include <stdio.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <unistd.h>

#define BUFFER_SIZE 5
#define NUM_PRODUCERS 2
#define NUM_CONSUMERS 3

int semid;
int buffer[BUFFER_SIZE];
int in = 0, out = 0;

void produce(int item) {
    buffer[in] = item;
    in = (in + 1) % BUFFER_SIZE;
}

int consume() {
    int item = buffer[out];
    out = (out + 1) % BUFFER_SIZE;
    return item;
}

void* producer(void* arg) {
    int producer_id = *((int*)arg);
    int item = 1;

    while (1) {
        // Produce item
        printf("Producer %d is producing item %d.\n", producer_id, item);
        produce(item);

        // Sleep for a random time
        sleep(1);

        item++;
    }
}

void* consumer(void* arg) {
    int consumer_id = *((int*)arg);

    while (1) {
        // Consume item
        int item = consume();
        printf("Consumer %d is consuming item %d.\n", consumer_id, item);

        // Sleep for a random time
        sleep(2);
one.c                                                    1,1            Top
```

```c
}

void* consumer(void* arg) {
    int consumer_id = *((int*)arg);

    while (1) {
        // Consume item
        int item = consume();
        printf("Consumer %d is consuming item %d.\n", consumer_id, item);

        // Sleep for a random time
        sleep(2);
    }
}

int main() {
    int producer_ids[NUM_PRODUCERS];
    int consumer_ids[NUM_CONSUMERS];
    pthread_t producers[NUM_PRODUCERS];
    pthread_t consumers[NUM_CONSUMERS];

    // Create a semaphore set with two semaphores (producer and consumer)
    semid = semget(IPC_PRIVATE, 2, IPC_CREAT | 0666);

    // Initialize producer semaphore to buffer size (5)
    semctl(semid, 0, SETVAL, BUFFER_SIZE);
    // Initialize consumer semaphore to 0 (no items to consume initially)
    semctl(semid, 1, SETVAL, 0);

    for (int i = 0; i < NUM_PRODUCERS; i++) {
        producer_ids[i] = i;
        pthread_create(&producers[i], NULL, producer, &producer_ids[i]);
    }

    for (int i = 0; i < NUM_CONSUMERS; i++) {
        consumer_ids[i] = i;
        pthread_create(&consumers[i], NULL, consumer, &consumer_ids[i]);
    }

    for (int i = 0; i < NUM_PRODUCERS; i++) {
        pthread_join(producers[i], NULL);
    }

    for (int i = 0; i < NUM_CONSUMERS; i++) {
        pthread_join(consumers[i], NULL);
    }

    // Cleanup the semaphore set
    semctl(semid, 0, IPC_RMID);

    return 0;
}
one.c [RO]                                                      92,1           Bot
```

```
cardi~/vit/os/lab9 ./one
Producer 0 is producing item 1.
Producer 1 is producing item 1.
Consumer 0 is consuming item 0.
Consumer 1 is consuming item 0.
Consumer 2 is consuming item 0.
Producer 0 is producing item 2.
Producer 1 is producing item 2.
Producer 0 is producing item 3.
Producer 1 is producing item 3.
Consumer 0 is consuming item 2.
Consumer 1 is consuming item 3.
Consumer 2 is consuming item 3.
Producer 0 is producing item 4.
Producer 1 is producing item 4.
Consumer 0 is consuming item 4.
Producer 0 is producing item 5.
Producer 1 is producing item 5.
Consumer 1 is consuming item 4.
Consumer 2 is consuming item 5.
Producer 0 is producing item 6.
Producer 1 is producing item 6.
Consumer 0 is consuming item 5.
Consumer 2 is consuming item 6.
Consumer 1 is consuming item 6.
Producer 0 is producing item 7.
Producer 1 is producing item 7.
Producer 0 is producing item 8.
Producer 1 is producing item 8.
Consumer 2 is consuming item 7.
Consumer 0 is consuming item 7.
Consumer 1 is consuming item 7.
Producer 0 is producing item 9.
Producer 1 is producing item 9.
Producer 1 is producing item 10.
Producer 0 is producing item 10.
Consumer 1 is consuming item 10.
Consumer 2 is consuming item 8.
Consumer 0 is consuming item 8.
Producer 1 is producing item 11.
Producer 0 is producing item 11.
Producer 0 is producing item 12.
Producer 1 is producing item 12.
Consumer 0 is consuming item 11.
Consumer 1 is consuming item 12.
Consumer 2 is consuming item 12.
Producer 0 is producing item 13.
Producer 1 is producing item 13.
^C
cardi~/vit/os/lab9 scrot --focused fig1.png
```

```
[0] 0:bash*                                              "moon" 10:33 22-Oct-23
```

second



```c
#include <stdio.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <unistd.h>
#include <stdbool.h>

#define NUM_PHILOSOPHERS 5
#define EATING 0
#define THINKING 1
#define HUNGRY 2

int semid;
int total_eat_count = 0;

void grab_forks(int philosopher_id) {
    struct sembuf sop[2];

    sop[0].sem_num = philosopher_id;
    sop[0].sem_op = -1;
    sop[0].sem_flg = 0;

    sop[1].sem_num = (philosopher_id + 1) % NUM_PHILOSOPHERS;
    sop[1].sem_op = -1;
    sop[1].sem_flg = 0;

    semop(semid, sop, 2);
}

void put_away_forks(int philosopher_id) {
    struct sembuf sop[2];

    sop[0].sem_num = philosopher_id;
    sop[0].sem_op = 1;
    sop[0].sem_flg = 0;

    sop[1].sem_num = (philosopher_id + 1) % NUM_PHILOSOPHERS;
    sop[1].sem_op = 1;
    sop[1].sem_flg = 0;

    semop(semid, sop, 2);
```

```
two.c                                   1,1            Top
:NERDTreeToggle
```

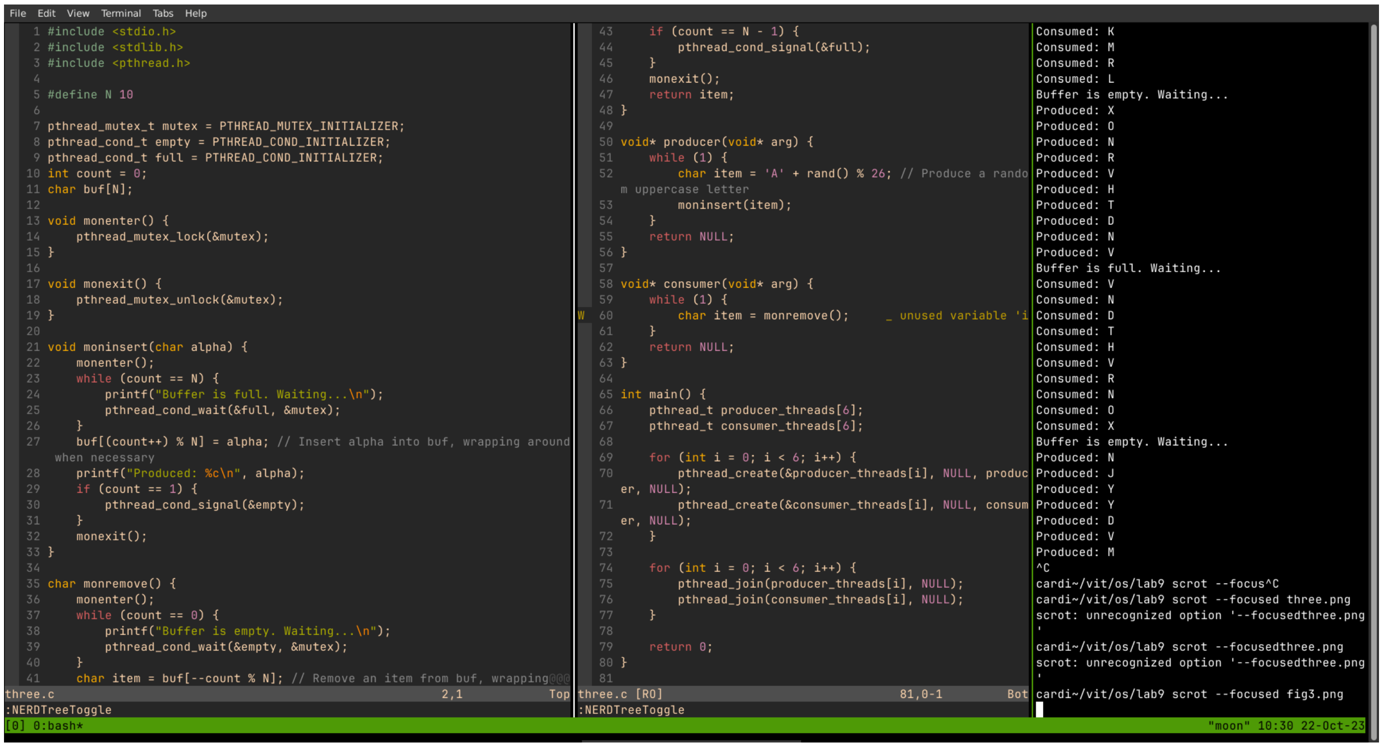
```c

int main() {
    int philosopher_ids[NUM_PHILOSOPHERS];
    pthread_t philosophers[NUM_PHILOSOPHERS];

    // Create a semaphore set with one semaphore per philosopher
    semid = semget(IPC_PRIVATE, NUM_PHILOSOPHERS, IPC_CREAT | 0666);

    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        semctl(semid, i, SETVAL, 1); // Initialize semaphores to 1 (available)
        philosopher_ids[i] = i;
        pthread_create(&philosophers[i], NULL, philosopher, &philosopher_ids[i]);
    }

    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        pthread_join(philosophers[i], NULL);
    }

    // Cleanup the semaphore set
    semctl(semid, 0, IPC_RMID);

    return 0;
}

```

```
two.c [RO]                              84,1           Bot
:NERDTreeToggle
```

```
Philosopher 0 is eating.
Philosopher 2 is done eating and putting away
forks.
Philosopher 2 is thinking.
Philosopher 3 is hungry and grabbing forks.
Philosopher 3 is eating.
Philosopher 0 is done eating and putting away
forks.
Philosopher 0 is thinking.
Philosopher 1 is hungry and grabbing forks.
Philosopher 1 is eating.
Philosopher 3 is done eating and putting away
forks.
Philosopher 3 is thinking.
Philosopher 1 is done eating and putting away
forks.
Philosopher 1 is thinking.
Philosopher 4 is hungry and grabbing forks.
Philosopher 4 is eating.
Philosopher 2 is hungry and grabbing forks.
Philosopher 2 is eating.
Philosopher 2 is done eating and putting away
forks.
Philosopher 0 is hungry and grabbing forks.
Philosopher 0 is eating.
Philosopher 3 is hungry and grabbing forks.
Philosopher 3 is eating.
Philosopher 4 is done eating and putting away
forks.
Philosopher 4 is thinking.
Philosopher 2 is thinking.
Philosopher 0 is done eating and putting away
forks.
Philosopher 0 is thinking.
Philosopher 3 is done eating and putting away
forks.
Philosopher 3 is thinking.
Philosopher 4 is hungry and grabbing forks.
Philosopher 4 is eating.
Philosopher 1 is hungry and grabbing forks.
Philosopher 1 is eating.
^C
cardi~/vit/os/lab9 scrot --focused fig2.png
```

```
[0] 0:bash*                                        "moon" 10:31 22-Oct-23
```

third



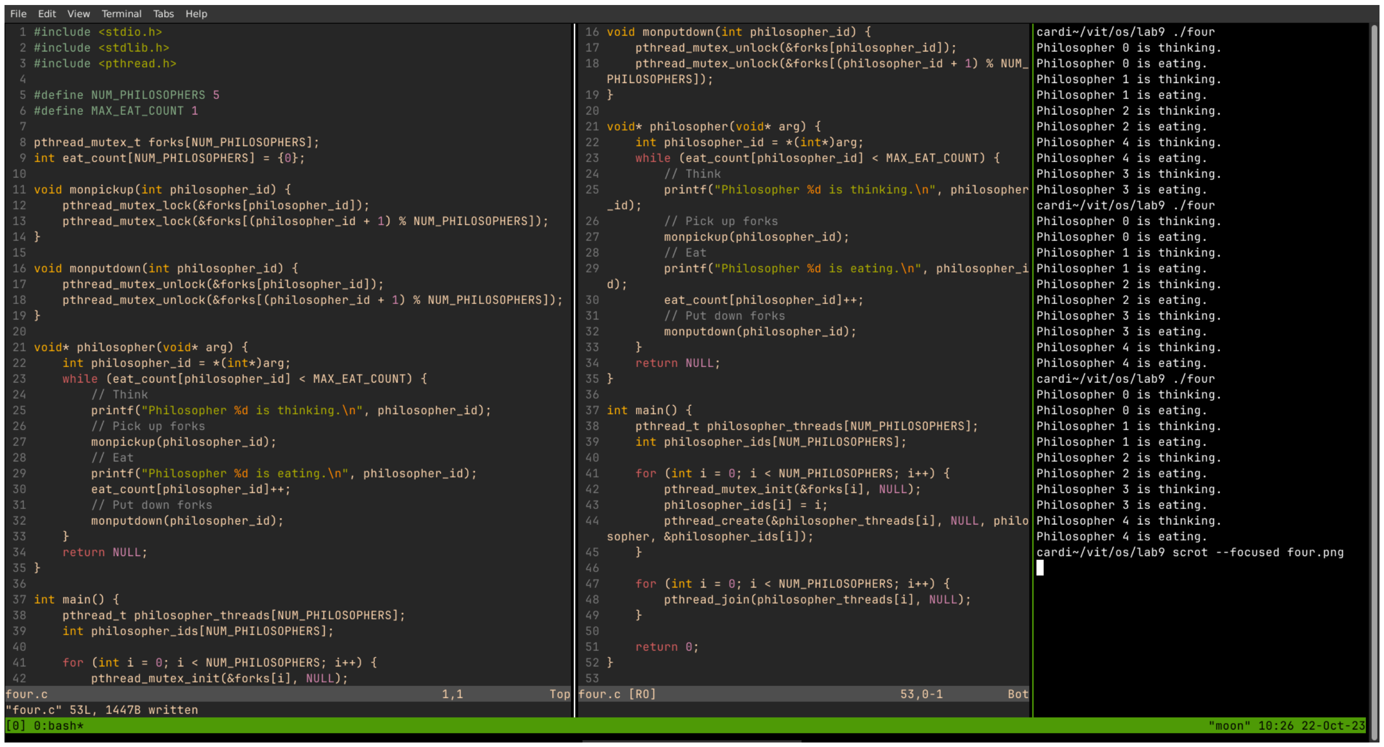
```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

#define N 10

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t empty = PTHREAD_COND_INITIALIZER;
pthread_cond_t full = PTHREAD_COND_INITIALIZER;
int count = 0;
char buf[N];

void monenter() {
    pthread_mutex_lock(&mutex);
}

void monexit() {
    pthread_mutex_unlock(&mutex);
}

void moninsert(char alpha) {
    monenter();
    while (count == N) {
        printf("Buffer is full. Waiting...\n");
        pthread_cond_wait(&full, &mutex);
    }
    buf[(count++) % N] = alpha; // Insert alpha into buf, wrapping around
 when necessary
    printf("Produced: %c\n", alpha);
    if (count == 1) {
        pthread_cond_signal(&empty);
    }
    monexit();
}

char monremove() {
    monenter();
    while (count == 0) {
        printf("Buffer is empty. Waiting...\n");
        pthread_cond_wait(&empty, &mutex);
    }
    char item = buf[--count % N]; // Remove an item from buf, wrapping@@@
three.c                                                2,1          Top
:NERDTreeToggle
```

```c
    if (count == N - 1) {
        pthread_cond_signal(&full);
    }
    monexit();
    return item;
}

void* producer(void* arg) {
    while (1) {
        char item = 'A' + rand() % 26; // Produce a rando
m uppercase letter
        moninsert(item);
    }
    return NULL;
}

void* consumer(void* arg) {
    while (1) {
W       char item = monremove();      _ unused variable 'i
    }
    return NULL;
}

int main() {
    pthread_t producer_threads[6];
    pthread_t consumer_threads[6];

    for (int i = 0; i < 6; i++) {
        pthread_create(&producer_threads[i], NULL, produc
er, NULL);
        pthread_create(&consumer_threads[i], NULL, consum
er, NULL);
    }

    for (int i = 0; i < 6; i++) {
        pthread_join(producer_threads[i], NULL);
        pthread_join(consumer_threads[i], NULL);
    }

    return 0;
}

three.c [RO]                              81,0-1        Bot
:NERDTreeToggle
```

```
Consumed: K
Consumed: M
Consumed: R
Consumed: L
Buffer is empty. Waiting...
Produced: X
Produced: O
Produced: N
Produced: R
Produced: V
Produced: H
Produced: T
Produced: D
Produced: N
Produced: V
Buffer is full. Waiting...
Consumed: V
Consumed: N
Consumed: D
Consumed: T
Consumed: H
Consumed: V
Consumed: R
Consumed: N
Consumed: O
Consumed: X
Buffer is empty. Waiting...
Produced: N
Produced: J
Produced: Y
Produced: Y
Produced: D
Produced: V
Produced: M
^C
cardi~/vit/os/lab9 scrot --focus^C
cardi~/vit/os/lab9 scrot --focused three.png
scrot: unrecognized option '--focusedthree.png
'
cardi~/vit/os/lab9 scrot --focusedthree.png
scrot: unrecognized option '--focusedthree.png
'
cardi~/vit/os/lab9 scrot --focused fig3.png
```

```
[0] 0:bash*                                        "moon" 10:30 22-Oct-23
```

fourth



```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

#define NUM_PHILOSOPHERS 5
#define MAX_EAT_COUNT 1

pthread_mutex_t forks[NUM_PHILOSOPHERS];
int eat_count[NUM_PHILOSOPHERS] = {0};

void monpickup(int philosopher_id) {
    pthread_mutex_lock(&forks[philosopher_id]);
    pthread_mutex_lock(&forks[(philosopher_id + 1) % NUM_PHILOSOPHERS]);
}

void monputdown(int philosopher_id) {
    pthread_mutex_unlock(&forks[philosopher_id]);
    pthread_mutex_unlock(&forks[(philosopher_id + 1) % NUM_PHILOSOPHERS]);
}

void* philosopher(void* arg) {
    int philosopher_id = *(int*)arg;
    while (eat_count[philosopher_id] < MAX_EAT_COUNT) {
        // Think
        printf("Philosopher %d is thinking.\n", philosopher_id);
        // Pick up forks
        monpickup(philosopher_id);
        // Eat
        printf("Philosopher %d is eating.\n", philosopher_id);
        eat_count[philosopher_id]++;
        // Put down forks
        monputdown(philosopher_id);
    }
    return NULL;
}

int main() {
    pthread_t philosopher_threads[NUM_PHILOSOPHERS];
    int philosopher_ids[NUM_PHILOSOPHERS];

    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        pthread_mutex_init(&forks[i], NULL);
```

```
four.c                                              1,1            Top
"four.c" 53L, 1447B written
```

```c
void monputdown(int philosopher_id) {
    pthread_mutex_unlock(&forks[philosopher_id]);
    pthread_mutex_unlock(&forks[(philosopher_id + 1) % NUM_PHILOSOPHERS]);
}

void* philosopher(void* arg) {
    int philosopher_id = *(int*)arg;
    while (eat_count[philosopher_id] < MAX_EAT_COUNT) {
        // Think
        printf("Philosopher %d is thinking.\n", philosopher_id);
        // Pick up forks
        monpickup(philosopher_id);
        // Eat
        printf("Philosopher %d is eating.\n", philosopher_id);
        eat_count[philosopher_id]++;
        // Put down forks
        monputdown(philosopher_id);
    }
    return NULL;
}

int main() {
    pthread_t philosopher_threads[NUM_PHILOSOPHERS];
    int philosopher_ids[NUM_PHILOSOPHERS];

    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        pthread_mutex_init(&forks[i], NULL);
```

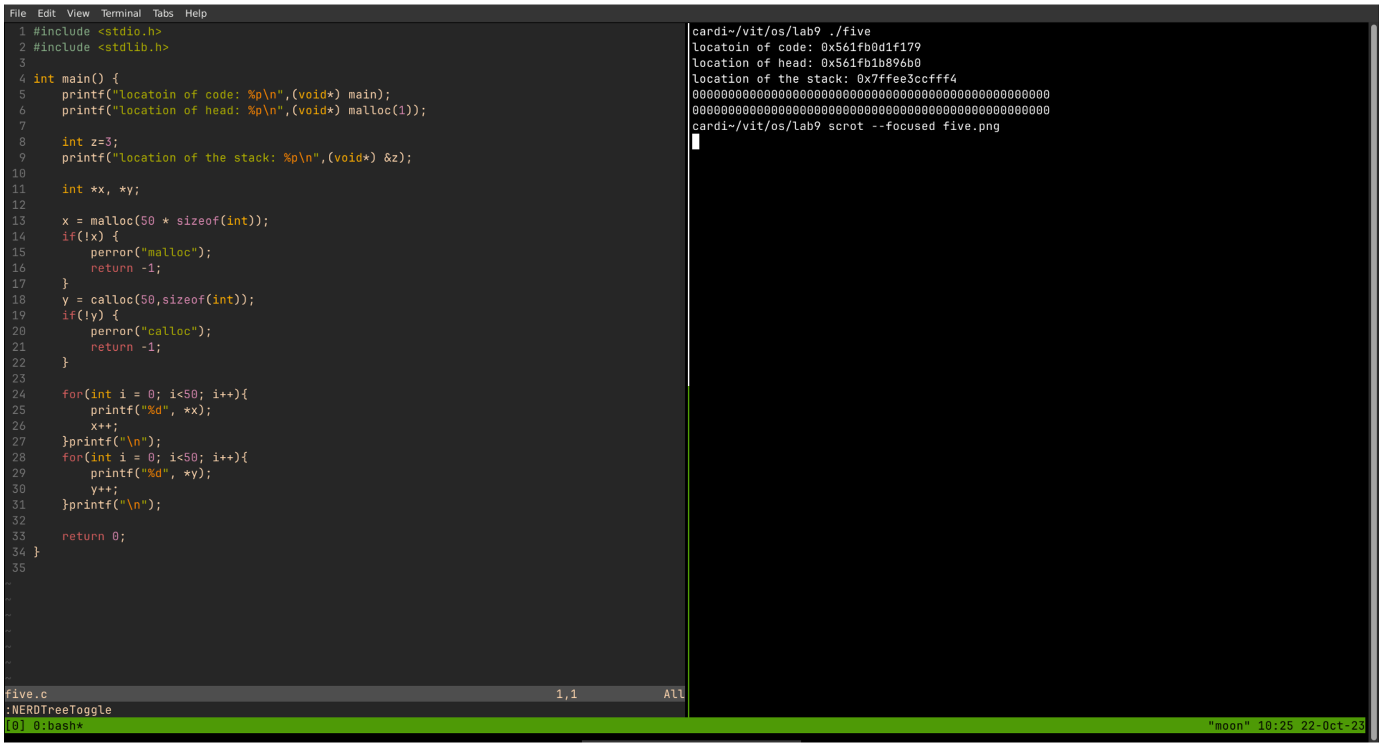
```c
        philosopher_ids[i] = i;
        pthread_create(&philosopher_threads[i], NULL, philosopher, &philosopher_ids[i]);
    }

    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        pthread_join(philosopher_threads[i], NULL);
    }

    return 0;
}
```

```
four.c [RO]                                53,0-1         Bot
```

```
cardi~/vit/os/lab9 ./four
Philosopher 0 is thinking.
Philosopher 0 is eating.
Philosopher 1 is thinking.
Philosopher 1 is eating.
Philosopher 2 is thinking.
Philosopher 2 is eating.
Philosopher 4 is thinking.
Philosopher 4 is eating.
Philosopher 3 is thinking.
Philosopher 3 is eating.
cardi~/vit/os/lab9 ./four
Philosopher 0 is thinking.
Philosopher 0 is eating.
Philosopher 1 is thinking.
Philosopher 1 is eating.
Philosopher 2 is thinking.
Philosopher 2 is eating.
Philosopher 3 is thinking.
Philosopher 3 is eating.
Philosopher 4 is thinking.
Philosopher 4 is eating.
cardi~/vit/os/lab9 ./four
Philosopher 0 is thinking.
Philosopher 0 is eating.
Philosopher 1 is thinking.
Philosopher 1 is eating.
Philosopher 2 is thinking.
Philosopher 2 is eating.
Philosopher 3 is thinking.
Philosopher 3 is eating.
Philosopher 4 is thinking.
Philosopher 4 is eating.
cardi~/vit/os/lab9 scrot --focused four.png
```

```
[0] 0:bash*                                                "moon" 10:26 22-Oct-23
```

fifth



```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    printf("locatoin of code: %p\n",(void*) main);
    printf("location of head: %p\n",(void*) malloc(1));

    int z=3;
    printf("location of the stack: %p\n",(void*) &z);

    int *x, *y;

    x = malloc(50 * sizeof(int));
    if(!x) {
        perror("malloc");
        return -1;
    }
    y = calloc(50,sizeof(int));
    if(!y) {
        perror("calloc");
        return -1;
    }

    for(int i = 0; i<50; i++){
        printf("%d", *x);
        x++;
    }printf("\n");
    for(int i = 0; i<50; i++){
        printf("%d", *y);
        y++;
    }printf("\n");

    return 0;
}

```

```
five.c                                          1,1            All
:NERDTreeToggle
```

```
cardi~/vit/os/lab9 ./five
locatoin of code: 0x561fb0d1f179
location of head: 0x561fb1b896b0
location of the stack: 0x7ffee3ccfff4
00000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000
cardi~/vit/os/lab9 scrot --focused five.png
```

```
[0] 0:bash*                                        "moon" 10:25 22-Oct-23
```