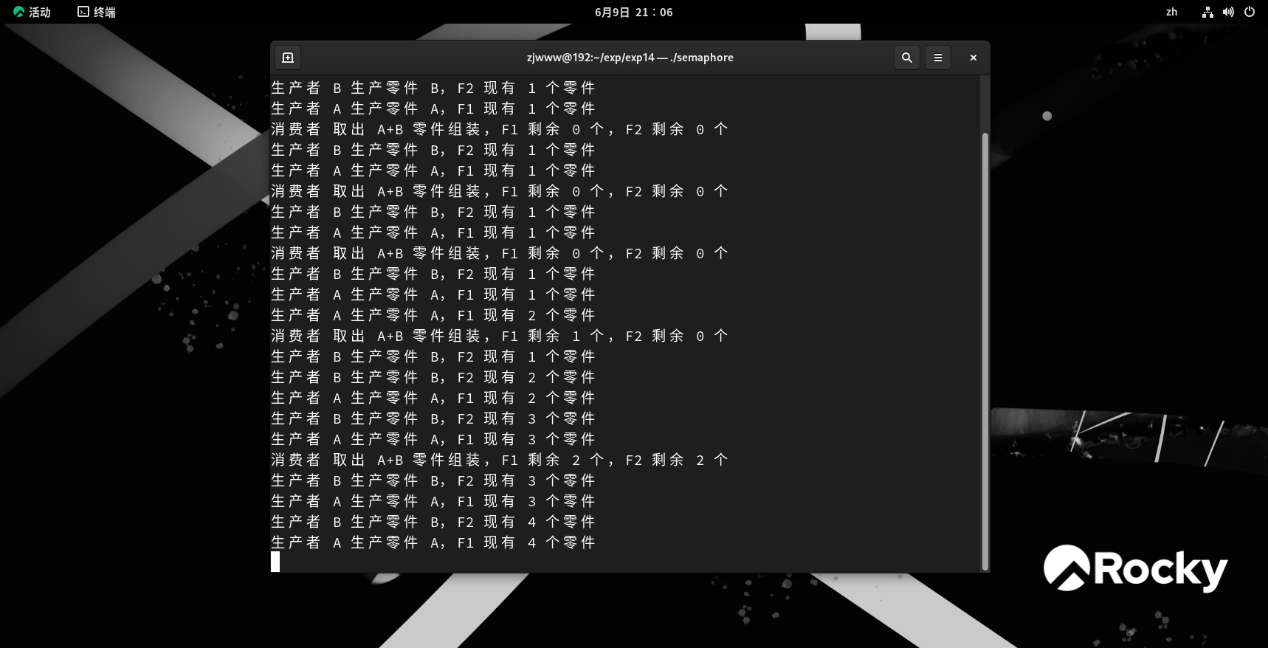
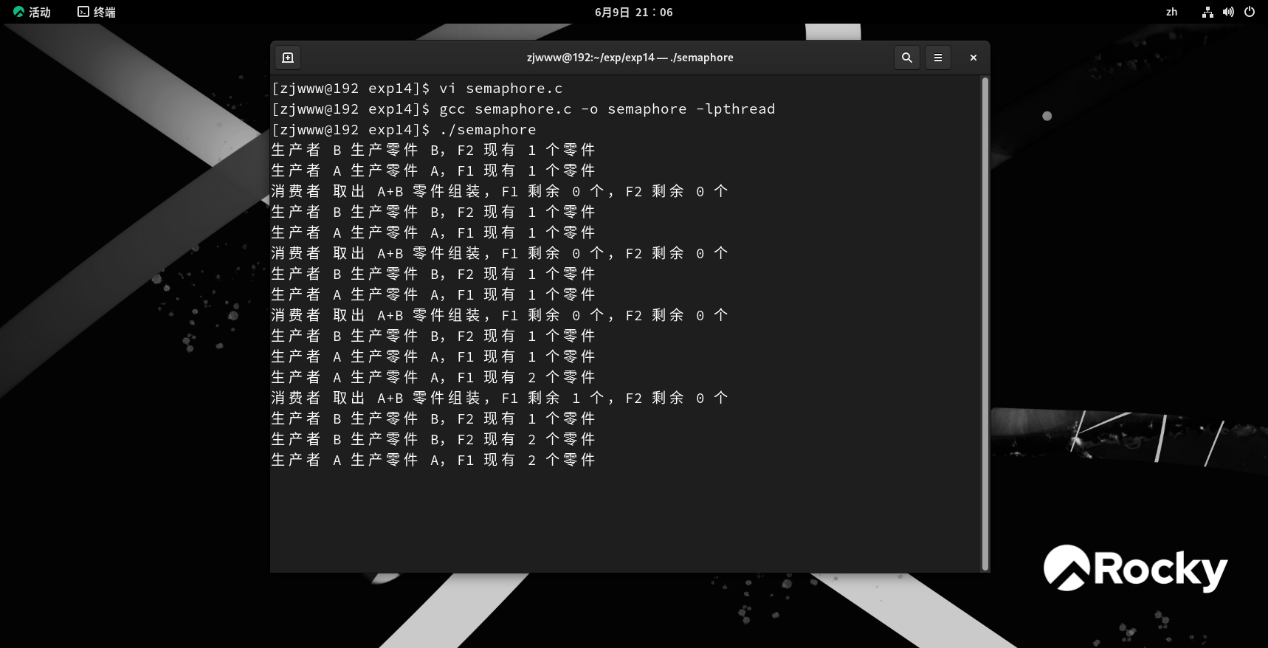
**实验14**

1. **创建并编写代码**

****

1. **编译并运行**

**电脑的屏幕截图

AI 生成的内容可能不正确。**

电脑萤幕画面

AI 生成的内容可能不正确。

**附录**

源代码

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

// 信号量定义

sem\_t empty1, full1;   // F1 货架（容量 4）

sem\_t empty2, full2;   // F2 货架（容量 6）

sem\_t mutex;           // 互斥锁

// 模拟货架（用计数器表示零件数量）

int shelf1 = 0;  // 零件 A 数量

int shelf2 = 0;  // 零件 B 数量

// 生产者 A：生产零件 A

void\* producer\_A(void\* arg) {

    while (1) {

        // 模拟生产零件 A 的时间

        sleep(rand() % 2 + 1);

        // 申请 F1 空槽位

        sem\_wait(&empty1);

        // 申请互斥锁

        sem\_wait(&mutex);

        shelf1++;

        printf("生产者 A 生产零件 A，F1 现有 %d 个零件\n", shelf1);

        // 释放互斥锁

        sem\_post(&mutex);

        // 唤醒消费者（F1 有零件）

        sem\_post(&full1);

    }

    return NULL;

}

// 生产者 B：生产零件 B

void\* producer\_B(void\* arg) {

    while (1) {

        // 模拟生产零件 B 的时间

        sleep(rand() % 2 + 1);

        // 申请 F2 空槽位

        sem\_wait(&empty2);

        // 申请互斥锁

        sem\_wait(&mutex);

        shelf2++;

        printf("生产者 B 生产零件 B，F2 现有 %d 个零件\n", shelf2);

        // 释放互斥锁

        sem\_post(&mutex);

        // 唤醒消费者（F2 有零件）

        sem\_post(&full2);

    }

    return NULL;

}

// 消费者：装配车间

void\* consumer(void\* arg) {

    while (1) {

        // 等待 F1 和 F2 都有零件

        sem\_wait(&full1);

        sem\_wait(&full2);

        // 申请互斥锁

        sem\_wait(&mutex);

        shelf1--;

        shelf2--;

        printf("消费者 取出 A+B 零件组装，F1 剩余 %d 个，F2 剩余 %d 个\n", shelf1, shelf2);

        // 释放互斥锁

        sem\_post(&mutex);

        // 唤醒生产者（F1/F2 有空槽位）

        sem\_post(&empty1);

        sem\_post(&empty2);

        // 模拟组装时间

        sleep(rand() % 3 + 1);

    }

    return NULL;

}

int main() {

    // 初始化信号量

    sem\_init(&empty1, 0, 4);  // F1 空槽位初始值 4

    sem\_init(&full1, 0, 0);   // F1 零件数量初始值 0

    sem\_init(&empty2, 0, 6);  // F2 空槽位初始值 6

    sem\_init(&full2, 0, 0);   // F2 零件数量初始值 0

    sem\_init(&mutex, 0, 1);   // 互斥锁初始值 1

    pthread\_t tid\_A, tid\_B, tid\_consumer;

    // 创建生产者线程

    pthread\_create(&tid\_A, NULL, producer\_A, NULL);

    pthread\_create(&tid\_B, NULL, producer\_B, NULL);

    // 创建消费者线程

    pthread\_create(&tid\_consumer, NULL, consumer, NULL);

    // 等待线程结束（按 Ctrl+C 终止程序）

    pthread\_join(tid\_A, NULL);

    pthread\_join(tid\_B, NULL);

    pthread\_join(tid\_consumer, NULL);

    // 销毁信号量

    sem\_destroy(&empty1);

    sem\_destroy(&full1);

    sem\_destroy(&empty2);

    sem\_destroy(&full2);

    sem\_destroy(&mutex);

    return 0;

}