

گزارش تمرین امتیازی مربوط به آزمایش ششم آزمایشگاه سیستم‌های عامل

نگار موقتیان، ۹۸۳۱۰۶۲

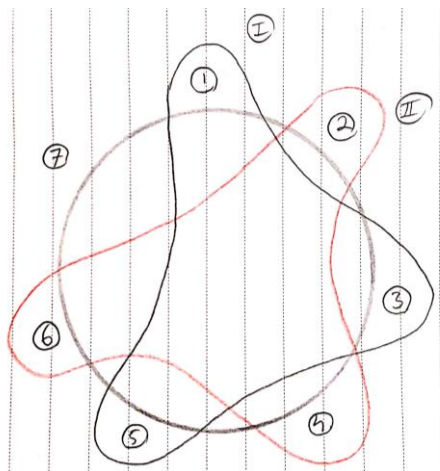
۱. ارائه الگوریتمی برای زمانبندی غذاخوردن فیلسوف‌ها

در این قسمت از آزمایش می‌خواهیم برنامه‌ای ارائه دهیم که فیلسوف‌ها (که تعداد آن‌ها عددی فرد است) بتوانند مطابق آن تصمیم بگیرند در هر لحظه باید فکر کنند و یا غذا بخورند.

در ابتدای برنامه کتابخانه‌های مورد نیاز اضافه شده‌اند.

سپس متغیرهای برنامه تعریف شده‌اند. متغیر n ورودی‌ای است که در ابتدای برنامه از کاربر دریافت می‌شود. تعداد فیلسوف‌ها در ادامه برابر خواهد بود با $N = 2n + 1$. سپس متغیر p با مقدار اولیه صفر تعریف شده که در هر نوبت اولین فیلسوفی که انتخاب می‌کنیم را نشان می‌دهد. این متغیر از اولین فیلسوف شروع کرده و در هر نوبت به صورت چرخشی یک عدد به جلو حرکت می‌کند.

در تابع `main` برنامه ابتدا ورودی گرفته شده و سپس در یک حلقه بی‌نهایت هر بار آرایه s در یک حلقه N تایی پر می‌شود. به ازای هر خانه i ام از آرایه اگر مقدار خانه صفر بود فیلسوف باید فکر کند و اگر یک بود می‌تواند غذا بخورد. به این ترتیب تصمیم گرفته می‌شود که در هر نوبت کدام یک از فیلسوف‌ها می‌توانند با هم غذا بخورند (میزهای مجازی مشخص می‌شوند). شیوه انتخاب فیلسوف‌ها به طور کلی مانند زیر است. در این شکل دو نوبت از γ نوبت نشان داده شده‌است که در آن p برابر با ۰ و ۱ قرار می‌گیرد. این فریم (که به رنگ مشکی و قرمز نمایش داده شده‌است و همان میزهای مجازی ما هستند) به ترتیب همراه با متغیر p می‌چرخد.



```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>
#include <string.h>

int n, N, p, counter;

int main() {
    // get n as input. there will be N = 2n + 1 philosophers at the table
    printf("> Please enter the value of 'n' (there will be 2n + 1 philosophers
at the table): ");
    scanf("%d", &n);

    N = 2 * n + 1; // number of philosophers
    int s[N];       // for each philosopher: 0 -> think, 1 -> eat

    while (1) { // decide which philosophers can eat at the same time in each
round
        // fill the s array, which indicates which philosophers should be
eating and which philosophers should be thinking
        memset(s, 0, sizeof s);
        int curr = p;
        if (N == 1)
            s[0] = 1;
        else
            while (curr != ((p - 1) + N) % N)
                s[curr] = 1, curr = (curr + 2) % N;
        p = (p + 1) % N;

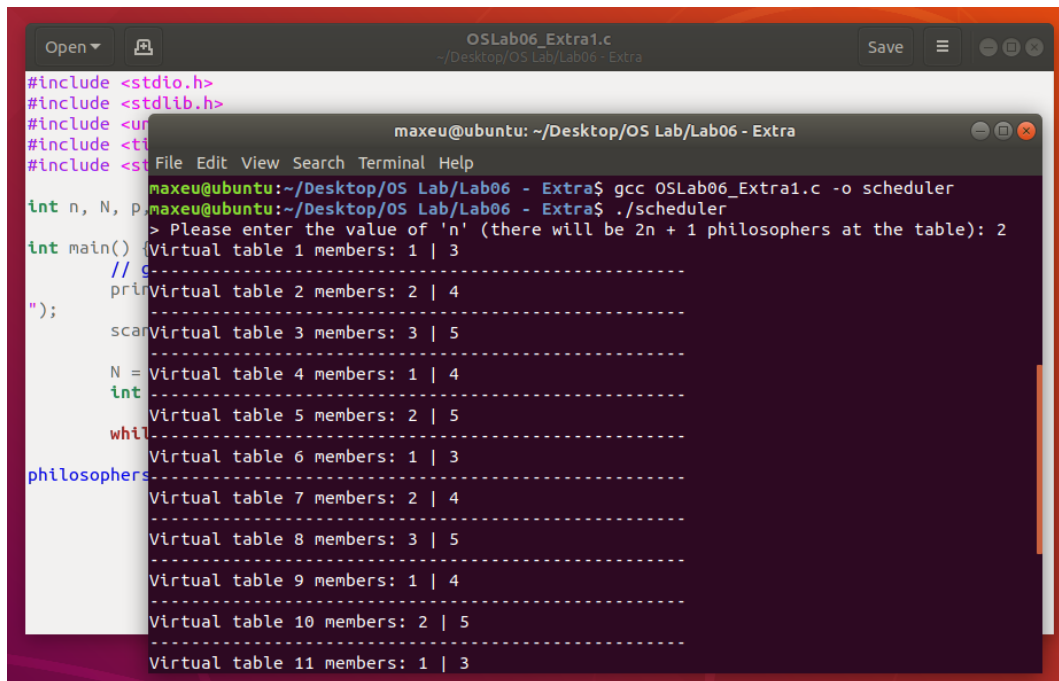
        // print the virtual table's info
        printf("Virtual table %d members: ", ++counter);
        for (int i=0; i<N; i++)
            if (s[i])
                printf("%d | ", i + 1);
        printf("\b\b \n-----
\n");

        // it takes 1 second for philosophers at the virtual table to finish
eating
        sleep(1);
    }

    return 0;
}

```

حال می‌توانیم برنامه فوق را اجرا کنیم. خروجی برنامه به ازای ۵ فیلسوف مانند زیر است.



The screenshot shows a code editor with a file named `OSLab06_Extra1.c` and a terminal window. The code in the editor includes headers `<stdio.h>` and `<stdlib.h>`, and defines `int n, N, p;`. The `main` function prompts the user to enter the value of 'n' (the number of philosophers). The terminal shows the user entering '2', followed by the program outputting 11 virtual table configurations, each with two members. The configurations are as follows:

| Table | Members |
|------------------|----------------|
| Virtual table 1 | members: 1 3 |
| Virtual table 2 | members: 2 4 |
| Virtual table 3 | members: 3 5 |
| Virtual table 4 | members: 1 4 |
| Virtual table 5 | members: 2 5 |
| Virtual table 6 | members: 1 3 |
| Virtual table 7 | members: 2 4 |
| Virtual table 8 | members: 3 5 |
| Virtual table 9 | members: 1 4 |
| Virtual table 10 | members: 2 5 |
| Virtual table 11 | members: 1 3 |

انتخاب این گروه‌ها منحصر به فرد نیست، به علاوه به این شیوه هر N نوبت یکبار فیلسوف‌هایی که باید همزمان مشغول غذاخوردن باشند تکرار می‌شوند.

در این روش به دلیل شیوه انتخاب میزهای مجازی می‌توانیم مطمئن باشیم دیگر deadlock رخ نخواهد داد زیرا هرگز دو فیلسوف که کنار یکدیگر نشسته‌اند همزمان برای غذا خوردن اقدام نمی‌کنند، لذا فیلسوفی که نوبت آن است می‌تواند بدون نگرانی دو چنگال دو طرف خود را بردارد.

۲. استفاده thread ها از خروجی قسمت قبل

حال می‌خواهیم از نتیجه برنامه ریزی قسمت قبل برای تعیین زمانی که thread فیلسوفان غذا می‌خورند و یا فکر می‌کنند استفاده کنیم.

تابع `main` این برنامه ترکیبی از کد مربوط به قسمت اصلی آزمایش ششم و کد قسمت قبل است. از کد قسمت قبل برای پر کردن آرایه `s` استفاده می‌کنیم و از آن در تابع `thinkAndEat` استفاده می‌کنیم (در واقع تابع `main` مانند یک زمانبند برای اجرای thread ها عمل می‌کند). در این تابع هر فیلسوف منتظر می‌شود تا خانه مربوط به آن از آرایه `s` برابر با یک شود، یعنی نوبت غذاخوردن آن شود. سپس قفل دو چنگال دو طرفش را بدست می‌گیرد و تا زمانی که خانه مربوط به آن از آرایه `s` برابر با یک است و اجازه غذاخوردن دارد، غذا می‌خورد و سپس قفل مربوط به چنگال‌هایش را آزاد می‌کند.

```

#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>

# define MAX 100
pthread_t philosophers[MAX];
pthread_mutex_t chopstick[MAX];
int s[MAX]; // for each philosopher: 0 -> think, 1 -> eat
int n, N, p;

void *thinkAndEat(int n) {
    while (1) {
        printf("philosopher %d is thinking!!\n", n + 1);

        while (!s[n]);

        // acquire the chopsticks
        pthread_mutex_lock(&chopstick[n]);
        pthread_mutex_lock(&chopstick[(n + 1) % N]);

        printf("philosopher %d is eating using chopstick[%d] and chopstick[%d]!!\n", n + 1, n, (n + 1) % N);
        while (s[n]);

        printf("philosopher %d finished eating!!\n", n + 1);

        // release the chopsticks
        pthread_mutex_unlock(&chopstick[n]);
        pthread_mutex_unlock(&chopstick[(n + 1) % N]);
    }
}

int main() {
    // get n as input. there will be N = 2n + 1 philosophers at the table
    printf("> Please enter the value of 'n' (there will be 2n + 1 philosophers at the table): ");
    scanf("%d", &n);
    N = 2 * n + 1; // number of philosophers

    // initiate the mutex locks used for accessing the chopsticks
    for (int i=0; i<N; i++)
        pthread_mutex_init(&chopstick[i], NULL);

    // create the threads representing the philosophers
    for (int i=0; i<N; i++)
        pthread_create(&philosophers[i], NULL, (void *) thinkAndEat, (void *) (intptr_t) i);
}

```

```

        while (1) { // decide which philosophers can eat at the same time in each
round
            // fill the s array, which indicates which philosophers should be
eating and which philosophers should be thinking
            memset(s, 0, sizeof s);
            int curr = p;
            if (N == 1)
                s[0] = 1;
            else
                while (curr != ((p - 1) + N) % N)
                    s[curr] = 1, curr = (curr + 2) % N;
            p = (p + 1) % N;

            // we give 1 second to the philosophers at the virtual table to finish
eating
            sleep(1);
        }

        return 0;
}

```

با این روش دیگر مشکل deadlock نخواهیم داشت زیرا طبق زمانبندی مشخص شده دو فیلسوف کنار هم همزمان اقدام به غذا خوردن نخواهند کرد. البته همچنان نیاز به قفل کردن چنگال‌ها تا اتمام غذا خوردن داریم زیرا ترتیب اجرای thread ها غیر قابل پیش‌بینی است و ممکن است در main آرایه s آپدیت شده باشد اما thread قبلی همچنان از آن با خبر نشده و مشغول غذا خوردن است، پس باید تا متوجه شدن آن و آزادسازی منابع منتظر بمانیم.

خروجی این برنامه به ازای ۵ فیلسوف مانند زیر خواهد بود. همانطور که مشاهده می‌شود غذا خوردن فیلسوف‌ها دقیقا مطابق با جدول زمانبندی قسمت قبل به ازای ۵ فیلسوف انجام شده‌است.

ابتدا فیلسوف‌های ۱ و ۳، سپس ۲ و ۴، سپس ۳ و ۵ و ... غذا خورده‌اند.

File Edit View Search Terminal Help

```
maxeu@ubuntu:~/Desktop/OS Lab/Lab06 - Extra$ gcc OSLab06_Extra2.c -o philosophers -pthread
```

```
maxeu@ubuntu:~/Desktop/OS Lab/Lab06 - Extra$ ./philosophers
```

```
> Please enter the value of 'n' (there will be 2n + 1 philosophers at the table): 2
```

```
philosopher 1 is thinking!!  
philosopher 1 is eating using chopstick[0] and chopstick[1]!!  
philosopher 5 is thinking!!  
philosopher 4 is thinking!!  
philosopher 3 is thinking!!  
philosopher 3 is eating using chopstick[2] and chopstick[3]!!  
philosopher 2 is thinking!!  
philosopher 1 finished eating!!  
philosopher 1 is thinking!!  
philosopher 3 finished eating!!  
philosopher 2 is eating using chopstick[1] and chopstick[2]!!  
philosopher 3 is thinking!!  
philosopher 4 is eating using chopstick[3] and chopstick[4]!!  
philosopher 4 finished eating!!  
philosopher 4 is thinking!!  
philosopher 2 finished eating!!  
philosopher 2 is thinking!!  
philosopher 3 is eating using chopstick[2] and chopstick[3]!!  
philosopher 5 is eating using chopstick[4] and chopstick[0]!!  
philosopher 5 finished eating!!  
philosopher 3 finished eating!!  
philosopher 3 is thinking!!  
philosopher 4 is eating using chopstick[3] and chopstick[4]!!  
philosopher 5 is thinking!!  
philosopher 1 is eating using chopstick[0] and chopstick[1]!!  
philosopher 4 finished eating!!  
philosopher 4 is thinking!!  
philosopher 1 finished eating!!  
philosopher 5 is eating using chopstick[4] and chopstick[0]!!  
philosopher 2 is eating using chopstick[1] and chopstick[2]!!  
philosopher 1 is thinking!!  
philosopher 5 finished eating!!  
philosopher 5 is thinking!!  
philosopher 2 finished eating!!  
philosopher 2 is thinking!!
```