

Michał Bronikowski

Systemy Operacyjne 2017
Problem złodzieja jabłek

9 października 2017

Spis treści

1	Opis zadania	3
1.1	Problem złodzieja jabłek	3
1.2	Rozwiązanie	3
2	Opis implementacji	3
3	Testowanie	5

1 Opis zadania

1.1 Problem złodzieja jabłek

Problem został przeze mnie wymyślony na potrzeby zadania. Wzorowałem się na klasycznym problemie synchronizacji tj. problemie producenta i konsumenta. Powiedzmy, że jesteśmy złodziejem jabłek i w naszej miejscowości są dwa sady A i B, w których w dość szybkim tempie przybywa jabłek. Kradniemy jabłka z sadu A lub z sadu B. W międzyczasie w sadach, dzięki działaniu właścicieli i natury jabłek przybywa. Problem polega na takim zsynchronizowaniu procesów sadów A i B oraz procesu złodzieja, żeby złodziej nie ukradł wszystkich jabłek z sadów, ponieważ to zwróciłoby uwagę właściciela i nasz złodziej mógłby mieć problemy.

1.2 Rozwiązanie

Rozwiązanie składa się z trzech procesów. Procesu złodzieja i dwóch procesów reprezentujących sady. Do synchronizacji procesów użyję semaforów. W celu zabezpieczenia dostępu do sekcji krytycznych programu użyję Mutex'ów.

2 Opis implementacji

Program składa się z jednego pliku źródłowego **sem.c**.

W programie możemy wyróżnić następującą strukturę:

```
int main()
{
    srand(time(0));
    pthread_attr_t attr;
    pthread_attr_init(&attr);
    pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);

    pthread_mutex_init(&mutex, NULL);
    sem_init(&skradziono, 0, 5);
    sem_init(&skradziono2, 0, 5);
    sem_init(&uroslo, 0, 0);

    pthread_t threads[3];
    pthread_create(&threads[0], &attr, sadA, NULL);
    pthread_create(&threads[2], &attr, sadB, NULL);
    pthread_create(&threads[1], &attr, zlodziej, NULL);
    pthread_create(&threads[2], &attr, sadB, NULL);
    pthread_join(threads[0], NULL);
    pthread_join(threads[1], NULL);
    pthread_join(threads[2], NULL);

    pthread_attr_destroy(&attr);
    pthread_mutex_destroy(&mutex);
    sem_destroy(&skradziono);
    sem_destroy(&uroslo);
    sem_destroy(&skradziono2);

    pthread_exit(NULL);
    return 0;
}
```

Funkcja główna programu

```

void *zlodziej()
{
    while(1)
    {
        sem_wait(&uroslo);
        pthread_mutex_lock(&mutex);
        int sad = (rand()%2)+1;
        if(sad == 1)
        {
            if(jablko_z_A < 1)
            {
                printf("Nie ukradnę z sadu A - nie mają już jabłek\n");
                return 0;
            }
            else
            {
                --jablko_z_A;
                zA++;
                printf("Kradnę jabłko z sadu A mam już %d jabłek z sadu A i %d jabłek z sadu B \n ",zA,zB);
            }
            sem_post(&skradziono2);
        }
        else
        {
            if(jablko_z_B < 1)
            {
                printf("Nie ukradnę z sadu B - nie mają już jabłek\n");
                return 0;
            }
            else
            {
                --jablko_z_B;
                zB++;
                printf("Kradnę jabłko z sadu B mam już %d jabłek z sadu A i %d jabłek z sadu B \n ",zA,zB);
            }
            sem_post(&skradziono);
        }
        pthread_mutex_unlock(&mutex);
    }
}

```

Funkcja zarządzająca procesem złodzieja

```

void *sadA() {
    while (1)
    {
        sem_wait(&skradziono2);
        pthread_mutex_lock(&mutex);
        printf("Jestem z sadu A i mam %d jabłek\n", ++jablko_z_A);
        pthread_mutex_unlock(&mutex);
        sem_post(&uroslo);
    }
}

```

Funkcja zarządzająca procesem sadu A

```

void *sadB() {
    while (1)
    {
        sem_wait(&skradziono);
        pthread_mutex_lock(&mutex);
        printf("Jestem z sadu B i mam %d jabłek\n", ++jablko_z_B);
        pthread_mutex_unlock(&mutex);
        sem_post(&uroslo);
    }
}

```

Funkcja zarządzająca procesem sadu B

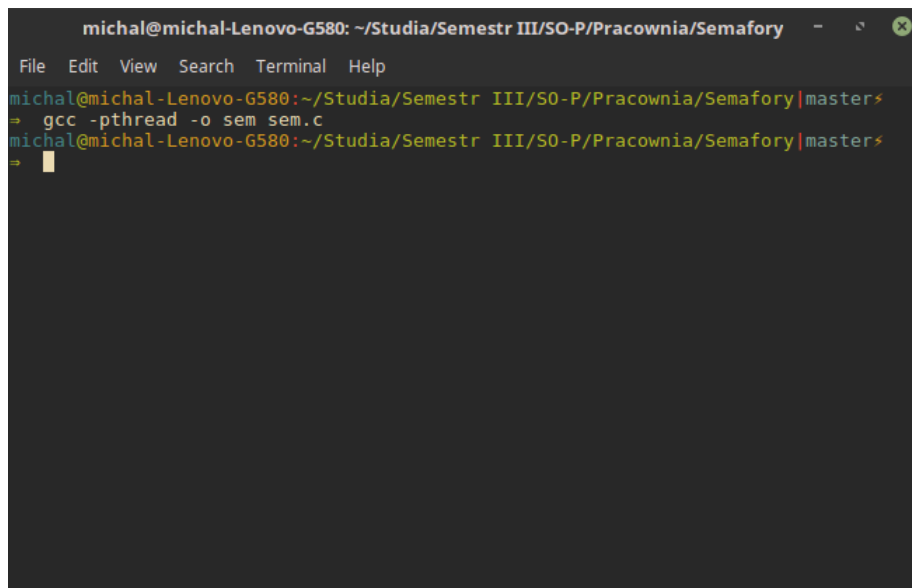
Na początku inicjalizuję semafony o nazwach: **skradziono**,**uroslo**,**skradziono2**. Inicjalizuję również mutex dający dostęp do zasobów tylko jednemu wątkowi. Następnie tworzę 3 wątki: **sadA**,**sadB**,**zlodziej**. Procesy odpowiadające za „produkcję” jabłek w sadach są zwalniane przez semafony **skradziono** i **skradziono2**. Każdy z procesów sadów informuje o tym, że zwiększyła się ilość dostępnych jabłek. Proces złodzieja losowo wybiera, z którego sadu chce ukraść jabłko, jeżeli w danym sadzie nie ma już jabłek oznacza to, że synchronizacja nie

działa poprawnie i wtedy program kończy swoje działanie. Każdorazowo po kradzieży złodziej zwalnia odpowiedni semafor przypisany do sadu A lub sadu B. Na starcie w każdym sadzie jest dziesięć jabłek oraz wartości odpowiadających im semaforom ustawione są na pięć, a złodzieja na zero. Taki zabieg pozwala nam uniknąć sytuacji w której złodziej będzie chciał ukraść jabłka z sadu, w którym tych jabłek nie ma.

3 Testowanie

Program należy uruchamiać na systemie operacyjnym LINUX. Do kompilacji służy komenda:

gcc -pthread -o sem sem.c

A screenshot of a terminal window with a dark background. The title bar at the top reads "michal@michal-Lenovo-G580: ~/Studia/Semestr III/SO-P/Pracownia/Semafory". Below the title bar is a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal shows a prompt "michal@michal-Lenovo-G580:~/Studia/Semestr III/SO-P/Pracownia/Semafory|master\$" followed by the command "gcc -pthread -o sem sem.c" being entered and executed. The prompt then changes to "michal@michal-Lenovo-G580:~/Studia/Semestr III/SO-P/Pracownia/Semafory|master\$" again, with a cursor at the end of the line.

Kompilacja

Wersja binarna programu będzie dostępna pod nazwą **sem**, uruchamiamy poleceniem:

./sem

```
michal@michal-Lenovo-G580: ~/Studia/Semestr III/SO-P/Pracownia/Semafory
File Edit View Search Terminal Help
michal@michal-Lenovo-G580:~/Studia/Semestr III/SO-P/Pracownia/Semafory$ ./sem
Desten z sadu B i mam 11 jabłek
Desten z sadu A i mam 11 jabłek
Desten z sadu A i mam 12 jabłek
Desten z sadu A i mam 13 jabłek
Desten z sadu A i mam 14 jabłek
Desten z sadu A i mam 15 jabłek
Desten z sadu B i mam 12 jabłek
Desten z sadu B i mam 13 jabłek
Desten z sadu B i mam 14 jabłek
Desten z sadu B i mam 15 jabłek
Kradnę jabłko z sadu B mam już 0 jabłek z sadu A i 1 jabłek z sadu B
Kradnę jabłko z sadu B mam już 0 jabłek z sadu A i 2 jabłek z sadu B
Kradnę jabłko z sadu A mam już 1 jabłek z sadu A i 2 jabłek z sadu B
Kradnę jabłko z sadu A mam już 2 jabłek z sadu A i 2 jabłek z sadu B
Kradnę jabłko z sadu A mam już 3 jabłek z sadu A i 2 jabłek z sadu B
Kradnę jabłko z sadu B mam już 3 jabłek z sadu A i 3 jabłek z sadu B
Kradnę jabłko z sadu B mam już 3 jabłek z sadu A i 4 jabłek z sadu B
Kradnę jabłko z sadu B mam już 3 jabłek z sadu A i 5 jabłek z sadu B
Kradnę jabłko z sadu B mam już 3 jabłek z sadu A i 6 jabłek z sadu B
Desten z sadu B i mam 10 jabłek
Kradnę jabłko z sadu A mam już 4 jabłek z sadu A i 6 jabłek z sadu B
Desten z sadu B i mam 11 jabłek
Desten z sadu A i mam 12 jabłek
Kradnę jabłko z sadu B mam już 4 jabłek z sadu A i 7 jabłek z sadu B
Kradnę jabłko z sadu B mam już 4 jabłek z sadu A i 8 jabłek z sadu B
Desten z sadu A i mam 13 jabłek
Desten z sadu A i mam 14 jabłek
Desten z sadu B i mam 10 jabłek
Desten z sadu B i mam 11 jabłek
Desten z sadu B i mam 12 jabłek
Desten z sadu B i mam 13 jabłek
Desten z sadu B i mam 14 jabłek
Desten z sadu B i mam 15 jabłek
Kradnę jabłko z sadu B mam już 4 jabłek z sadu A i 10 jabłek z sadu B
Kradnę jabłko z sadu B mam już 4 jabłek z sadu A i 11 jabłek z sadu B
Kradnę jabłko z sadu A mam już 5 jabłek z sadu A i 11 jabłek z sadu B
Kradnę jabłko z sadu A mam już 6 jabłek z sadu A i 11 jabłek z sadu B
Kradnę jabłko z sadu A mam już 7 jabłek z sadu A i 11 jabłek z sadu B
Kradnę jabłko z sadu B mam już 7 jabłek z sadu A i 12 jabłek z sadu B
Kradnę jabłko z sadu B mam już 7 jabłek z sadu A i 13 jabłek z sadu B
Kradnę jabłko z sadu B mam już 7 jabłek z sadu A i 14 jabłek z sadu B
Kradnę jabłko z sadu B mam już 7 jabłek z sadu A i 15 jabłek z sadu B
```

Uruchomienie

Program powinien działać poprawnie ze względu na wartości startowe semaforów odpowiadających za sady. Tyle jeżeli chodzi o teorię, wykonałem jeszcze test, polegający na tym że uruchomiłem ten program w tle i przez 5h się nie zatrzymał.