Santa Claus Problem

Bruna Zamith
Henrique Frajacomo
João Victor Pacheco
Marcos Faglioni

Universidade Federal de São Carlos

17 de maio de 2018

1. Problema

Stand Claus sleeps in his shop at the North Pole and can only be awakened by either (1) all nine reindeer being back from their vacation in the South Pacific, or (2) some of the elves having difficulty making toys; to allow Santa to get some sleep, the elves can only wake him when three of them have problems. When three elves are having their problems solved, any other elves wishing to visit Santa must wait for those elves to return. If Santa wakes up to find three elves waiting at his shop's door, along with the last reindeer having come back from the tropics, Santa has decided that the elves can wait until after Christmas, because it is more important to get his sleigh ready. (It is assumed that the reindeer do not want to leave the tropics, and therefore they stay there until the last possible moment.) The last reindeer to arrive must get Santa while the others wait in a warming hut before being harnessed to the sleigh.

2. Tópicos para Solução

1. Devemos ter duas threads: Uma para as Renas e uma para os Elfos;
2. O Papai Noel (Santa Claus) é uma função que pode ser invocada pelas Renas ou pelos Elfos;
3. O número atual de Elfos e o número atual de Renas deve ser incrementado em suas respectivas threads;
4. O número máximo de Renas é 9 e este é o critério para chamada da função do Papai Noel pelas Renas;
5. O número máximo de Elfos é determinado pelo usuário, mas os Elfos são atendidos de 3 em 3, e este é o critério para chamada de função do Papai Noel pelos Elfos;
6. O Papai Noel sempre dá prioridade para as Renas;
7. As threads acabam quando os números máximos definidos para as Renas e os Elfos são atingidos.

3. Tópicos para Tratamento de Deadlocks

1. Alterações em contadores de Renas e Elfos não podem ter concorrência;
2. A função do Papai Noel não pode ser invocada por duas ou mais threads simultaneamente;
3. Elfos não podem pedir ajuda do Papai Noel enquanto ele já está ajudando outros elfos.

4. Conceitos Utilizados

- Threads;
- Semáforos;
- Mutex;
- Deadlocks.

5. Solução

1. *Devemos ter duas threads: Uma para as Renas e uma para os Elfos*

```
//Cria as threads e instanciacao das funcoes elfos e renas
pthread_create(&thr_elfos, NULL, funcao_elfos, NULL);
pthread_create(&thr_renas, NULL, funcao_renas, NULL);

//Join - Sincroniza todas as threads a fim de serem finalizadas antes
// de encerrar o codigo
pthread_join(thr_elfos, NULL);
pthread_join(thr_renas, NULL);
```

5. Solução

2. *O Papai Noel (Santa Claus) é uma função que pode ser invocada pelas Renas ou pelos Elfos*

```
void* funcao_renas(void* v){
...
      funcao_santa(NULL);
...
  return NULL;
}

void* funcao_elfos(void* v){
...
      funcao_santa(NULL);
...
  return NULL;
}
```

5. Solução

3. *O número atual de Elfos e o número atual de Renas deve ser incrementado em suas respectivas threads*

```
void* funcao_renas(void* v){
...
    renas++;
...
    return NULL;
}

void* funcao_elfos(void* v){
...
    elfos++;
...
    return NULL;
}
```

5. Solução

4. *O número máximo de Renas é 9 e este é o critério para chamada da função do Papai Noel pelas Renas*

```
#define max_renas 9

void* funcao_renas(void* v){
...
    if(renas < max_renas){
        renas++;
    }

    if(renas == max_renas){
        funcao_santa(NULL);
    }
...
    return NULL;
}
```

5. Solução

5. *O número máximo de Elfos é determinado pelo usuário, mas os Elfos são atendidos de 3 em 3, e este é o critério para chamada de função do Papai Noel pelos Elfos*

```
#define max_elfos 3

void* funcao_elfos(void* v){
    for(int i=0; i<elfos_total; i++){
        if(elfos < max_elfos){
            elfos++;
        }

        if(elfos == max_elfos){
            funcao_santa(NULL);
        }
    }
    return NULL;
}
```

5. Solução

6. *O Papai Noel sempre dá prioridade para as Renas*

```
void* funcao_santa(void* v){
...
    if(renas == max_renas){
        ...
    }

    else if(elfos == max_elfos){
        ...
    }
...
    return NULL;
}

void* funcao_elfos(void* v){
...
    if(elfos == max_elfos && renas<max_renas){
        funcao_santa(NULL);
    }
...
    return NULL;
}
```

5. Solução

7. *As threads acabam quando os números máximos definidos para as Renas e os Elfos são atingidos*

```
void* funcao_renas(void* v){
    for(int i=0; i<max_renas; i++){
        ...
    }
    return NULL;
}

void* funcao_elfos(void* v){
    for(int i=0; i<elfos_total; i++){
        ...
    }
    return NULL;
}
```

6. Tratamento de Deadlocks

1. *Alterações em contadores de Renas e Elfos não podem ter concorrência*

```
sem_t mutex_contadores;

void* funcao_santa(void* v){
    sem_wait(&mutex_contadores);
...
    sem_post(&mutex_contadores);
    return NULL;
}

void* funcao_renas(void* v){
    sem_wait(&mutex_contadores);
...
    sem_post(&mutex_contadores);
    return NULL;
}

void* funcao_elfos(void* v){
    sem_wait(&mutex_contadores);
...
    sem_post(&mutex_contadores);
    return NULL;
}
```

6. Tratamento de Deadlocks

2. *A função do Papai Noel não pode ser invocada por duas ou mais threads simultaneamente*

```
sem_t semaforo_santa;

void* funcao_santa(void* v){
    sem_wait(&semaforo_santa);
...
    sem_post(&semaforo_santa);
    return NULL;
}
```

6. Tratamento de Deadlocks

3. *Elfos não podem pedir ajuda do Papai Noel enquanto ele já está ajudando outros elfos*

```
sem_t semaforo_elfos;

void* funcao_elfos(void* v){
...
    if(elfos == max_elfos && renas<max_renas){
        sem_wait(&semaforo_elfos);
        funcao_santa(NULL);
        sem_post(&semaforo_elfos);
    }
...
}
```

Referências

Allen B. Downey (V 2.2.1)

The Little Book of Semaphores

Fim