Universidade Federal de São Carlos
Departamento de Computação

Primeira Prova de SO1

Professora Dra. Kelen Vivaldini

Bruna Zamith Santos (RA: 628093)
Henrique Cordeiro Frajacomo (RA: 726536)
João Victor Pacheco (RA: 594970)
Marcos Augusto Faglioni Junior (RA: 628301)

05/2018
São Carlos - SP, Brasil

Conteúdo

1 Introdução

Este relatório visa descrever a solução do desafio ”Santa Claus”e a lógica por trás da proposta de solução pelos autores deste documento (Grupo 5).

O documento está organizado como segue: Seção 2 introduz o problema do Santa Claus; a Seção 3 expõe as principais ideias para resolução geral do problema e suas implementações; a Seção 4 expõe os deadlocks possíveis identificados e as principais ideias para tratá-los; na Seção 5 o código completo é mostrado, assim como as principais funções; por fim, a Seção 6 apresenta como executar o programa.

2 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.

3 Solução

De acordo com a descrição do problema apresentada na Seção 2, identificamos alguns tópicos essenciais. Eles são descritos a seguir, assim os trechos de códigos com ideias para suas implementações.

Para essas implementações e, futuramente, o tratamento de deadlocks, usamos os conceitos de Threads, Semáforos, Mutex e Deadlocks.

3.1 Threads: Uma para as Renas e uma para os Elfos

As renas e os elfos podem ser tratadas como duas threads separadas, que chamam um comando central (Santa Claus).

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

```
4
5     //Join - Sincroniza todas as threads a fim de serem finalizadas
6     // antes de encerrar o codigo
7     pthread_join(thr_elfos, NULL);
8     pthread_join(thr_renas, NULL);
```

3.2 Função para o Papai Noel (Santa Claus)

A função do Papai Noel só pode ser invocada pelas Renas ou pelos Elfos.

```
1     void* funcao_renas(void* v){
2     ...
3          funcao_santa(NULL);
4     ...
5       return NULL;
6     }
7
8     void* funcao_elfos(void* v){
9     ...
10         funcao_santa(NULL);
11    ...
12      return NULL;
13    }
```

3.3 Incremento do número de Renas e de Elfos

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

```
1     void* funcao_renas(void* v){
2     ...
3         renas++;
4     ...
5         return NULL;
6     }
7
8     void* funcao_elfos(void* v){
9     ...
10        elfos++;
11    ...
12        return NULL;
13    }
```

3.4 Número máximo de Renas

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;
}
```

3.5 Número máximo de Elfos

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;
}
```

3.6 Prioridade de atendimento do Papai Noel

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;
    }
```

3.7 Término das threads

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_renasl; i++){
            ...
        }
        return NULL;
    }

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

4 Tratamento de Deadlocks

4.1 Contadores

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

```
    sem_t mutex_contadores;

```

```
3     void* funcao_santa(void* v){
4         sem_wait(&mutex_contadores);
5     ...
6         sem_post(&mutex_contadores);
7         return NULL;
8     }
9
10    void* funcao_renas(void* v){
11        sem_wait(&mutex_contadores);
12    ...
13        sem_post(&mutex_contadores);
14        return NULL;
15    }
16
17    void* funcao_elfos(void* v){
18        sem_wait(&mutex_contadores);
19    ...
20        sem_post(&mutex_contadores);
21        return NULL;
22    }
```

4.2 Concorrência pelo Papai Noel

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

```
1     sem_t semaforo_santa;
2
3     void* funcao_santa(void* v){
4         sem_wait(&semaforo_santa);
5     ...
6         sem_post(&semaforo_santa);
7         return NULL;
8     }
```

4.3 Concorrência entre Elfos

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

```
1     sem_t semaforo_elfos;
2
3     void* funcao_elfos(void* v){
4     ...
5         if(elfos == max_elfos && renas<max_renas){
6             sem_wait(&semaforo_elfos);
7             funcao_santa(NULL);
```

```
8               sem_post(&semaforo_elfos);
9           }
10      ...
11      }
```

5 Código Final

A seguir é apresentado o código completo do santa_claus.c; sua explicação pode ser encontrada nos comentários do código.

```
1  //Definicoes
2  #define _USE_GNU 1
3  #define max_renas 9 //Numero maximo total de renas
4  #define max_elfos 3 //Numero maximo de elfos ajudados por vez
5
6  //Bibliotecas
7  #include <pthread.h>
8  #include <stdio.h>
9  #include <stdlib.h>
10 #include <unistd.h>
11 #include <semaphore.h>
12
13 //Semaforos
14 sem_t mutex_contadores; //Mutex para controlar a alteracao dos
       contadores (de elfos e de renas)
15 sem_t semaforo_elfos; //Previne que mais elfos entrem enquanto tres
       estao sendo ajudados
16 sem_t semaforo_santa; //Semaforo para indicar quando Santa Claus esta
       disponivel
17
18
19 //Variaveis globais
20 int elfos = 0; //Numero de elfos
21 int renas = 0; //Numero de renas
22 int elfos_total = 0; //Numero maximo de elfos a ser indicado pelo
       usuario
23
24 //Funcao da Santa Claus
25 void* funcao_santa(void* v){
26   sem_wait(&semaforo_santa); //Espera semaforo_santa (garante que outra
       thread nao esta usando ela)
27   sem_wait(&mutex_contadores); //Espera mutex_contador (para ter
       certeza que nao vai ter concorrencia de alteracao)
28   if(renas == max_renas){ //Se o numero de renas for igual ao numero
       maximo, o treno pode ser preparado
29     printf("Treno preparado!\n"); //Prepara o treno
```

```
30      for(int i=0; i<max_renas; i++){ //Depois que o treno foi preparado, o numero de renas eh decrementado ate 0
31        renas--;
32      }
33    } else if(elfos == max_elfos){ //Se o numero de elfos for igual ao numero maximo (3), os elfos podem ser ajudados
34      printf("Elfos Ajudados!\n"); //Ajuda os elfos
35      for(int i=0; i<max_elfos; i++){ //Depois que os elfos foram ajudados, o numero de elfos eh decrementado ate 0
36        elfos--;
37      }
38    }
39    sem_post(&mutex_contadores); //Libera o mutex_contador
40    sem_post(&semaforo_santa); //Libera o semaforo_santa
41    return NULL;
42  }
43
44  //Funcao dos Elfos
45  // Sem parametros e sem retorno
46  void* funcao_elfos(void* v){
47    for(int i=0; i<elfos_total; i++){ //Elfos_total eh definido pelo usuario, numero maximo de execucoes da funcao dos elfos
48      if(elfos < max_elfos){ //Se o numero de elfos for menor que o numero maximo
49        sem_wait(&mutex_contadores); //Espera mutex_contador
50        elfos ++; //Incrementa o numero de elfos
51        printf("O numero de elfos eh: %i\n",elfos); //Printa o numero de elfos atual
52        sem_post(&mutex_contadores); //Libera mutex_contador
53      }
54      if(elfos == max_elfos && renas<max_renas){ //Se atingir o numero maximo de elfos e o numero de renas tambem nao for o maximo
55        sem_wait(&semaforo_elfos); //Espera para ver se os elfos nao estao ja sendo ajudados
56        printf("Os elfos chamaram Santa!\n"); //Senao, chama a Santa
57        funcao_santa(NULL);    //Se numero de elfos for igual ao numero maximo de elfos, chama a funcao funcao_santa afim de liberar as elfos
58        printf("%i elfos ajudados!\n",max_elfos); //Printa o numero de elfos ajudados
59        sem_post(&semaforo_elfos); //Libera o semaforo_elfos
60      }
61      sleep(rand() % 3); //Tempo aleatorio para a execucao de outras threads
62    }
63    return NULL;
64  }
```

```
65
66 //Funcao das Renas:
67 // Sem parametros e sem retorno
68 void* funcao_renas(void* v){
69   for(int i=0; i<max_renas; i++){  //Enquanto o numero maximo de renas
     nao for atingido, roda o loop
70     if(renas < max_renas){ //Se o numero de renas for menor que o
     numero maximo
71       sem_wait(&mutex_contadores); //Espera mutex_contador
72       renas ++; //Incrementa o numero de renas
73     printf("O numero de renas eh: %i\n",renas); //Printa o numero de
     renas atual
74       sem_post(&mutex_contadores); //Libera mutex_contador
75     }
76     if(renas == max_renas){ //Se atingir o numero maximo de renas
77       printf("As renas chamaram Santa!\n"); //Chama a Santa
78       funcao_santa(NULL);  //Se numero de renas for igual ao numero
     maximo de renas, chama a funcao funcao_santa afim de liberar as
     renas
79       printf("As renas foram amarradas e liberadas!\n"); //As renas sao
      amarradas e liberadas
80     }
81     sleep(rand() % 3); //Tempo aleatorio para a execucao de outras
     threads
82   }
83   return NULL;
84 }
85
86 //Funcao principal
87 int main(int argc, char *argv[]){
88   //Declara as threads
89   pthread_t thr_santa, thr_elfos, thr_renas;
90
91   //Recebe o numero de elfos como argumento do processo
92   if(argc != 2){
93     printf("Digite o numero de elfos na chamada da funcao!\n");
94   return -1;
95    }
96    elfos_total = atoi(argv[1]);
97
98   //Inicializa o mutex com 1
99   sem_init(&mutex_contadores, 0, 1);
100
101  //Inicializa os semaforos com 1
102  sem_init(&semaforo_santa, 0, 1);
103  sem_init(&semaforo_elfos, 0, 1);
104
```

```
105   //Cria as threads e instanciacao das funcoes elfos e renas
106   pthread_create(&thr_elfos, NULL, funcao_elfos, NULL);
107   pthread_create(&thr_renas, NULL, funcao_renas, NULL);
108
109   //Join - Sincroniza todas as threads a fim de serem finalizadas antes
        de encerrar o codigo
110   pthread_join(thr_elfos, NULL);
111   pthread_join(thr_renas, NULL);
112
113   //Destroi os semaforos e mutexes
114   sem_destroy(&mutex_contadores);
115   sem_destroy(&semaforo_elfos);
116   sem_destroy(&semaforo_santa);
117
118   //Finaliza o programa
119   return 0;
120 }
```

6 Execução

A solução implementada pode ser acessada através da execução do código santa_claus, encontrado no diretório Códigos/ a partir do diretório onde se encontra o READ-ME. Para compilar o código, acesse o diretório Códigos/ e abra o terminal. Digite o comando:

gcc -pthread santa_claus.c -o santa_claus -std=c99

Após a compilação, execute o código digitando:

./santa_claus N

Sendo N a quantidade total de elfos desejada pelo usuário.

Após isso o código executará e será encerrado automaticamente.

A Figura mostra o código sendo executado.

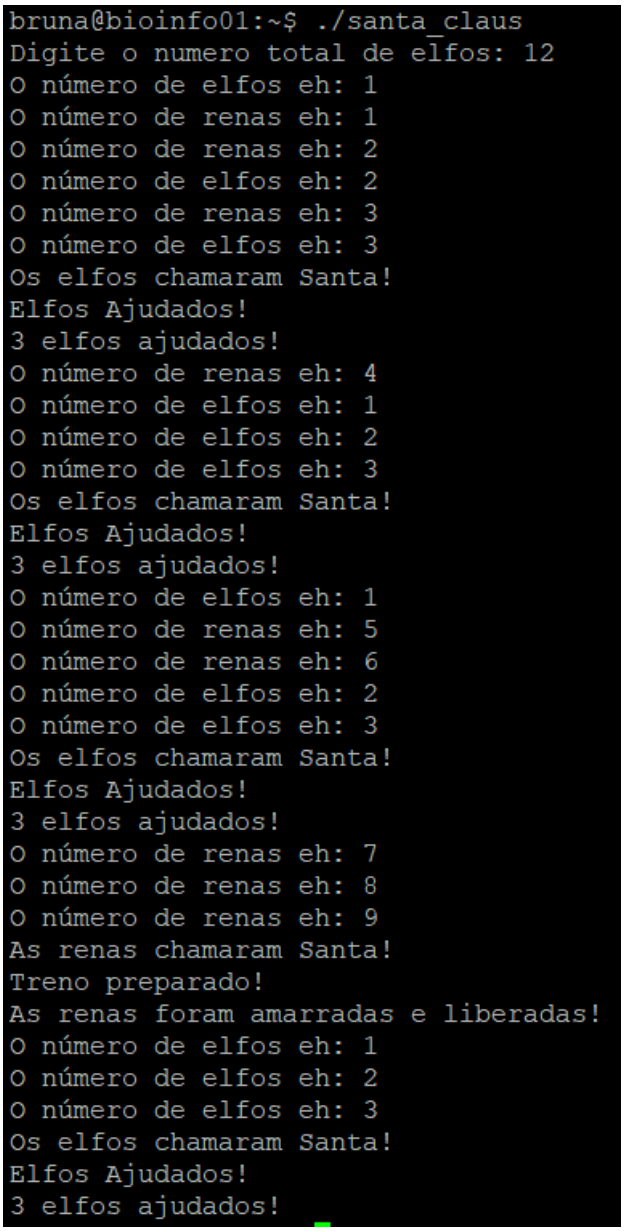

Figura 1: Código santa_claus sendo executado