# The child care problem

## Disciplina: Sistemas Operacionais 01 - Profª Kellen Vivaldini

•••

Leonardo de Oliveira Peralta - 726556
Rebeca Lima Rocha - 486060
Renata Sarmet Smiderle Mendes - 726586

# O problema:

Em uma creche, os regulamentos estaduais exigem que haja sempre 1 adulto presente para cada 3 crianças.

Para as crianças entrarem na sala, 1 adulto já deve estar lá. Nesse ponto, até mais 2 crianças podem entrar. Uma 4ª criança deve esperar que outro adulto apareça. O mesmo acontece quando um adulto quer sair.

# Semáforos Utilizados

```
// Proporção de 1 adulto para cada 3 crianças

sem_t sem_adulto;
sem_t sem_crianca;

// Como produtor/consumidor para o adulto

sem_t pos_vazia_adulto;
sem_t pos_ocupada_adulto;

// Como produtor/consumidor para a criança

sem_t pos_vazia_crianca;
sem_t pos_ocupada_crianca;
```

```
// Inicia com 1 adulto
sem_init(&sem_adulto, 0, 1);

// Inicia com 0 criancas
sem_init(&sem_crianca, 0, 0);

// Inicia com N_ITENS vazios
sem_init(&pos_vazia_adulto, 0, N_ITENS);

// Inicia com 0 itens ocupados
sem_init(&pos_ocupada_adulto, 0, 0);

// Inicia com 3 * N_ITENS vazios
sem_init(&pos_vazia_crianca, 0, (3 * N_ITENS));

// Inicia com 0 itens ocupados
sem_init(&pos_ocupada_crianca, 0, 0);
```

# Threads Utilizadas

```
void* entra_adulto(void *v) {

  int i;

  for(i=0;i<N_ITENS;i++){
    sem_wait(&pos_vazia_adulto);
    sem_wait(&sem_adulto);
    qtd_adulto += 1;

    fim_adulto = (fim_adulto + 1) % N_ITENS;
    buffer_adulto[fim_adulto] = i;

    sem_post(&pos_ocupada_adulto);

    sem_post(&sem_crianca);
    sem_post(&sem_crianca);
    sem_post(&sem_crianca);

    sleep(random() % 3);
  }
  return NULL;
}
```

```
void* entra_crianca(void *v) {

  int quantas_criancas, i;
  for(i=0;i< 3 * N_ITENS;i++){
    sem_wait(&pos_vazia_crianca);
    sem_wait(&sem_crianca);

    qtd_crianca += 1;

    fim_crianca = (fim_crianca + 1) % (3 * N_ITENS);
    buffer_crianca[fim_crianca] = i;

    sem_post(&pos_ocupada_crianca);
    sem_getvalue(&sem_crianca,&quantas_criancas);

    if (quantas_criancas == 0){
      sem_post(&sem_adulto);
    }
    sleep(random() % 2);
  }
  return NULL;
}
```

# Threads Utilizadas

```
void* sai_adulto(void *v){

  int i, j;

  for (i = 0; i < N_ITENS; i++){
    sem_wait(&pos_ocupada_adulto);

    while (qtd_crianca > 3 * (qtd_adulto - 1));

    for (j = 0; j < 3; j++){
      sem_wait(&sem_crianca);
    }

    inicio_adulto = (inicio_adulto + 1) % N_ITENS;
    qtd_adulto -= 1;

    sem_post(&pos_vazia_adulto);
    if ((qtd_adulto <= 0)||(3 * qtd_adulto == qtd_crianca))
      sem_post(&sem_adulto);

    sleep(random() % 10);
  }
  return NULL;
}
```

```
void* sai_crianca(void *v){

  int i, j;

  for (i = 0; i < N_ITENS; i++){
    sem_wait(&pos_ocupada_crianca);
    inicio_crianca = (inicio_crianca + 1) % (3 *
N_ITENS);
    qtd_crianca -= 1;

    sem_post(&pos_vazia_crianca);

    while ((qtd_adulto - 1) * 3 - 1 >= qtd_crianca){
      sem_wait(&pos_ocupada_adulto);
      for (j = 0; j < 3; j++)
            sem_wait(&sem_crianca);
      inicio_adulto = (inicio_adulto + 1) % N_ITENS;
      qtd_adulto -= 1;
    }

    sem_post(&sem_crianca);
    sleep(random() % 13);
  }
  return NULL;
}
```

## Controle de deadlocks :

- Mutual exclusion
  - Sair 1 adulto sempre que entrar 1 adulto e consequentemente não permitir que entrem crianças;
  - Entrar 1 criança sempre que sair 1 criança e consequentemente não permitir que saiam adultos.

- Hold and wait
  - Nunca ter uma posição vazia para entrar adultos;
  - Nunca ter uma posição vazia para entrar crianças;
  - Não garantir que vai entrar 1 adulto para que possam entrar crianças;
  - Necessidade de entrar mais 1 criança mas nunca entrar mais 1 adulto.

## Controle de deadlocks :

- No preemption
  - Não liberar para que entrem 3 crianças assim que entrar 1 adulto;
  - Não aumentar a demanda de 1 adulto se não puder entrar mais crianças.

- Circular wait
  - Precisar entrar 1 adulto mas o buffer de adulto estar cheio e ter de sair 1 adulto antes, mas para sair 1 adulto necessitar sair previamente 3 crianças;
  - Precisar entrar 1 criança e não poder entrar mais crianças antes de entrar 1 adulto, mas o buffer de adulto estar cheio e ter de sair 1 adulto antes, mas para sair 1 adulto necessitar sair previamente 3 crianças.

github.com/renatasarmet/childCareProblem