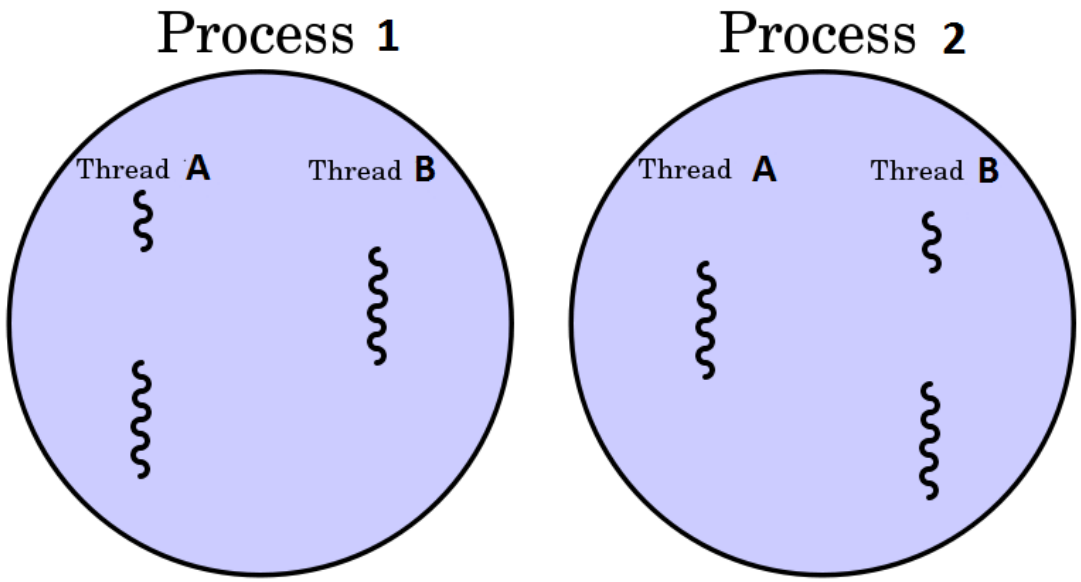
Trabalho 02 - Threads

Carlos Alberto

1. O que são threads?

- São desvios de rotinas de execução de programas que são executados dentro de um processo e compartilham dos mesmos espaços de memória do seu processo pai.



2. Como criar uma thread

- Se quisermos criar uma thread em C em SO do tipo GNU/Linux precisamos dos seguintes comandos:

```
pthread_create(id,atr_thread,pointer_function,thread_args);
```

- id é o identificador da thread
- atr_thread é um ponteiro para as variáveis de comportamento da thread.
- Pointer_function recebe um ponteiro para uma função que será executada na thread.
- Thread_args são os argumentos para a função da thread.

Um exemplo de uma implementação de threads n-ésimo multiplo de um inteiro

```
91    struct params parametro3;
92    parametro3.value = 7;
93
94    pthread_t thread_id1;
95    pthread_t thread_id2;
96
97    /*inicializando o semaforo*/
98    //sem_init(&semaforo,0,0);
99    pthread_create (&thread_id2, NULL, &mmc_function, &parametro1);
100   pthread_create (&thread_id1, NULL, &mmc_function, &parametro2);
101
102
103   pthread_join (thread_id1, NULL);
104   pthread_join (thread_id2, NULL);
105
106  // printf("MMC: %d \n",mmc);
107   //sem_destroy(&semaforo);
108  return 0;
109 }
```

3. Caracteristicas observadas nos testes

- As threads podem perder o "foco" da CPU a qualquer momento e ficar em um estado de "pausa" para que a CPU programe a execução de outras threads.
- Neste exemplo eu criei 2 threas para trabalharem de forma concorrente, mas na verdade são criadas no mínimo 4 threas neste exemplo, a que executa a main e a thread gerenciadora.
- As variáveis que se encontram publicas são livremente acessadas por qualquer thread, isso é bom, e é ruim, porque, em certos momentos, eu perdia dados pois estes eram sobrescrevidos por outras threads.

4. Ferramentas para resolução do problema de acesso á variáveis públicas

- Uso do Mutex
- Mutex é uma ferramenta da própria biblioteca pthread.h. Ele serve para resolver o problema chamado de "condição de corrida", que é quando existe a possibilidade de várias threads manipularem uma variável antes de outras de forma não programada.

- Usando o Mutex você garante que só uma thread tenha acesso aquelas variáveis e impede que outras threads consigam manipula-las.

Exemplo Multiplos já com o problema resolvido.

```
59    for(i = 1; i < nth_value; i++){
60      if(!(i%data->value)){
61
62        /*Aqui queremos evitar que outras threads acessem o mmc durante esta execução*/
63        pthread_mutex_lock (&mutex);
64        if(mmc == 0){
65           mmc = i;
66        }else{
67          /*testa se o valor que está no mmc é um mútiplo seu*/
68          if(!(mmc%data->value)){
69               if(sem_trywait(&semaforo) == EAGAIN){
70                sem_wait(&semaforo);
71                printf("Erro %d\n",sem_trywait(&semaforo));
72               };
73          }
74        }
75        pthread_mutex_unlock (&mutex);
76        /*libera a variavel mmc*/
77
```

```
➜  ~ gcc -pthread Trabalho02.c
➜  ~ ./a.out
Multiplos de 5
 5, 10, 15, 20, 25, 30, 35,
Multiplos de 3
 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39,
➜  ~
```

5. Ferramentas para sincronização de threads

Semaphores

- É um tipo de ferramenta de sincronização de threads. Através dela podemos controlar qual thread vai executar e qual thread deve esperar por sua vez.

- Usa um contador como indicador para as threads.

- Suas principais funções são : sem_wait(), sem_post().

Exemplo mmc com uso de Semaphores

```
59    for(i = 1; i < nth_value; i++){
60      if(!(i%data->value)){
61
62        /*Aqui queremos evitar que outras threads acessem o mmc durante esta execução*/
63        pthread_mutex_lock (&mutex);
64        if(mmc == 0){
65           mmc = i;
66        }else{
67          /*testa se o valor que está no mmc é um mútiplo seu*/
68          if(!(mmc%data->value)){
69               if(sem_trywait(&semaforo) == EAGAIN){
70                sem_wait(&semaforo);
71                printf("Erro %d\n",sem_trywait(&semaforo));
72               };
73          }
74        }
75        pthread_mutex_unlock (&mutex);
76        /*libera a variavel mmc*/
77
```

```
➜  ~ ./a.out
Multiplos de 5
 5, 10, 15, 20, 25, 30, 35,
Multiplos de 3
 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39,
MMC: 15
➜  ~
```

Referências

- Material cedido em aula "alp-ch04-threads".
- SILBERSCHATZ, Abraham; GALVIN, Peter B; GAGNE, Greg., Sistemas operacionais com java, Elsevier , 2ed , 2008

- Link do projeto: https://github.com/CarlosAlbertoUFS/SOProjects