UNIVERSIDADE FEDERAL
DE SANTA CATARINA
Campus de Blumenau

Sistemas Computacionais - 2021.1

Trabalho Prático 1 (TP1)

Prof. Carlos R. Moratelli
Entrega 01/06/2022

Roteiro

Objetivos

1. Utilizar programação multithreading.
2. Praticar técnicas de sincronização de processos e transferência de dados.
3. Estudar e implementar problemas clássicos de sincronização.

Problema: Implementação de uma biblioteca de lista encadeada genérica e threadsafe.

Uma biblioteca threadsafe consiste em um conjunto de rotinas que podem ser chamadas de um contexto multithread de maneira segura (livre de condições de corrida ou deadlocks). Essas rotinas implementam os mecanismos de sincronização internamente, de forma transparente ao programador. Assim, este trabalho requer a implementação de um conjunto de rotinas para manipulação de uma lista encadeada em contexto de multithread. A lista encadeada deve ter a estrutura conforme a Figura 1. Onde, o decritor é um nodo especial que possui ponteiros para o início e fim da lista, um contador de itens alocados e mecanismos de sincronização (mutex e semáforos).

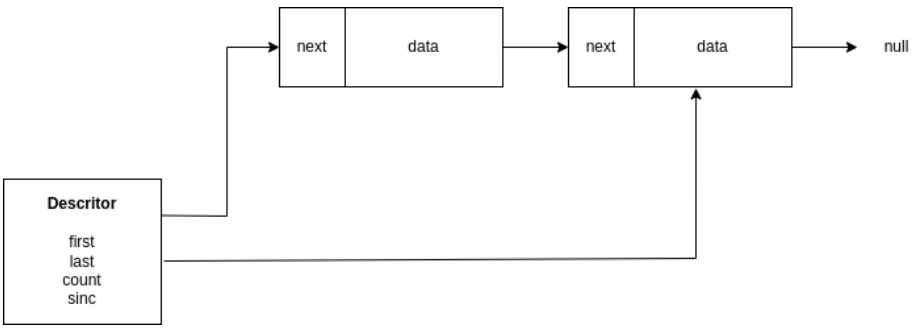

Figura 1. Estrutura da lista encadeada.

Observe que cada nodo da lista encadeada possui um campo *next* (próximo nodo) e um campo *data*. O campo *data* deve ser genérico e suportar qualquer tipo de dado, desde um simples inteiro até uma estrutura definida pelo usuário de qualquer tamanho. Desta forma, o campo data deve ser alocado conforme o tamanho do dado que a lista irá suportar.

As seguintes rotinas devem ser implementadas:

- *struct linkedlist_t *linkedlist_create(unsigned int data_size, unsigned int max_length);*

  Alocar o nodo descritor retornando um ponteiro para ele. Entradas:
  data_size: quantidade de bytes do campo data dos nodos.
  length: quantidade máxima de nodos suportados pela lista.

  Retorno: Ponteiro para o descritor recém alocado.

- *void linkedlist_insert_tail(struct linkedlist_t *l, void *e);*

  Insere um item no fim da lista. Deve bloquear a thread chamadora caso a lista contenha *length* elementos alocados. Entradas:
  l: Descritor da lista (retornado por *linkedlist_create()*).
  e: ponteiro para o dado a ser alocado na lista. O dado deve ser copiado para o campo data do nodo. A quantidade a ser copiada é definida por *data_size*.

- *void linkedlist_insert_head(struct linkedlist_t *l, void *e);*

  Mesmo comportamento de *linkedlist_insert_tail()*, contudo, inserindo no início da lista.

- *void linkedlist_remove_head(struct linkedlist_t *l, void* e);*

  Remove o elemento do início da lista. Deve bloquear a thread chamadora caso não exista mais elementos na lista.

  l: Descritor da lista (retornado por *linkedlist_create()*).
  e: ponteiro para área que memória que irá receber o conteúdo do nodo. A rotina deve copiar os dados do nodo para a área apontada e desalocar o nodo, mantendo a

consistência da lista. A quantidade a ser copiada é definida por *data_size*.

- *void linkedlist_remove_tail(struct linkedlist_t *l, void *e);*

  Mesmo comportamento de *linkedlist_remove_head()*, contudo, removendo do final da lista.

- *int linkedlist_size(struct linkedlist_t *l);*

  Retorna a quantidade de elementos da lista. Entrada:
  l: Descritor da lista (retornado por *linkedlist_create()*).

- *void linkedlist_destroy(struct linkedlist_t **l);*

  Desaloca completamente a lista, inclusive o nodo descritor. Nenhuma thread pode estar utilizando a lista quando esta função é chamada.
  l: Descritor da lista (retornado por *linkedlist_create()*).

A implementação da biblioteca deve ser seguida de uma aplicação de testes multithread que faça sua utilização. Exemplos de aplicação:

- ping para uma lista de hosts, determinar;
- determinar, por força bruta, o inteiro usado para a geração de um hash;
- descobrir se um número é primo;
- Calcular o determinante de uma matriz, entre outros.

Será disponibilizado um esqueleto de código para ser utilizado como base para a implementação.

Entrega

O trabalho pode ser implementado em linguagem C e realizado em grupos com no máximo 2 alunos. É permitido realizar o trabalho individualmente. A entrega do trabalho deve ser realizada pelo Moodle (código-fonte). Cópias de trabalho implicará na anulação da nota para todos os envolvidos.