CET-088, II Semestre de 2017
Data Lab: Laboratório C
Enviado: Segunda, 18 de Setembro, Entrega: Sexta, 30 de Setembro, 23:59

Leard Fernandes (`lofernandes@uesc.br`) é o responsável por esta atividade

1 Logistica

Não há dias para atraso, feriados ou qualquer extensão para esta atividade. Planeje iniciar o mais rápido possível, para que esteja pronto na data de entrega. Um estudante bem preparado pode completar esta tarefa em 1-2 horas, mas pode requerer maior tempo se você não programa em C.

Este é um projeto individual. Todas as tarefas são eletrônicas. Você pode fazer esta tarefa em qualquer máquina que você quiser. Ele foi testado em diferentes máquinas utilizando Linux. A sua avaliação será realizada em máquinas utilizando Linux. Esclarecimentos e correções serão postados na página do Curso/Grupo.

2 Visão Geral

Este laboratório irá lhe proporcionar a prática no estilo de programação que você irá precisar para estar hábil, e irá precisar para as atividades futuras na disciplina. O material coberto aqui deverá ser uma revisão para você. Algumas habilidade que serão testadas:

- Gerenciamento explicito de memória, como necessário em C.
- Criando e manipulando estruturas de dados baseadas em ponteiros.
- Implementando código robusto que opera corretamente com argumentos inválidos, incluindo ponteiros NULL.

O laboratório envolve a implementação de uma fila, suportando ambas as operações da estrutura de fila last-in, first-out (LIFO), e first-in-first-out (FIFO). A estrutura de dados em questão é uma lista ligada única, melhorada para realizar algumas operações mais eficientes.

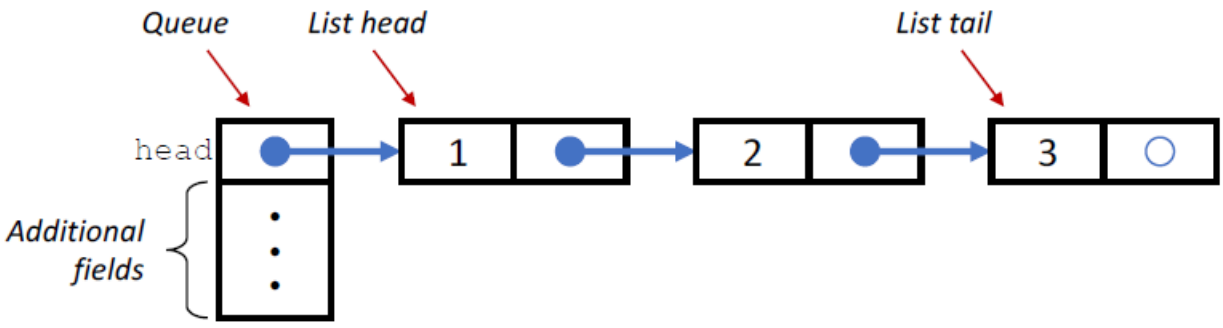

Figura 1: Implmentação de uma lista ligada de uma fila

3 Baixando o arquivo

Inicie copiando o arquivo `datalab-handout.tar` para um diretório protegido numa máquina Linux no qual você planeje realizar o seu trabalho. Então dê o comando

```
unix> tar xvf datalab-handout.tar.
```

Isso irá causar o desempacotamento dos arquivos no diretório. Consulte o README para mais informações

4 Visão geral

O arquivo `queue.h` contém declarações das seguintes estruturas:

```
/* Elemento da lista ligada */
typedef struct ELE {
  int value;
  struct ELE *next;
} list_ele_t;

/* Estrutura da fila */
typedef struct {
  list_ele_t *head; /* Elementos da lista ligada */
} queue_t;
```

Estas estruturas são combinadas para implementar uma fila, como pode ser visto na 1. A representação de alto nível de uma fila é uma estrutura do tipo `queue_t` . No inicio do código, esta estrutura contém apenas um único campo "head", mas que você irá adicionar outros campos. Os conteúdos da fila são representados como uma lista ligada única, com cada elemento representado por uma estrutura do tipo `list_ele_t`,

tendo os campos "value"e "next", armazenando um valor e um ponteiro na fila, respectivamente. Você pode adicionar mais campos nesta estrutura caso necessário.

Em seu código C, uma fila é um ponteiro do tipo `queue_t *`. Nós temos dois casos especiais: Uma fila NULL é uma que o ponteiro é configurado como NULL. Uma fila vazia é uma apontando para uma estrutura `queue_t` válida com um campo `head` configurado como NULL. Seu código precisará lidar propriamente com ambos os casos, assim como a fila contendo um ou mais elementos.

5 Tarefa de programação o arquivo

Sua tarefa é modificar o código em `queue.h` e `queue.c` para implementar totalmente as seguintes funções:

`q_new`: Cria uma nova, fila vazia.

`q_free`: Libera todo o armazenamento utilizado pela fila.

`q_insert_head`: Tenta inserir um novo elemento na cabeça da fila (LIFO).

`q_insert_tail`: Tenta inserir um novo elemento na calda da fila (FIFO).

`q_remove_head`: Tenta remover um elemento na cabeça da fila.

`q_size`: Calcula o número de elementos na fila.

`q_reverse`: Reorganizar a fila, tal que o elementos fiquem em ordem inversa.

Mais detalhes podem ser encontrados nos comentários nestes dois arquivos, incluindo como manipular operações inválidas, e quais os efeitos colaterais e valores de retorno a funções deverão ter.

A seguir algumas importantes notas que você deverá seguir ao implementar estas funções:

- Duas das funções: `q_insert_tail` e `q_size` irá precisar de um maior esforço seu, para atender os requisitos de performance. Implementações simples requerem O(n) passos para uma fila com n elementos. É exigido que sua implementação opere no tempo O(1), isto é, que o tempo necessário seja independente do tamanho da fila. Você pode fazer isto incluindo outros campos na estrutura de dados `queue_t` e gerenciando estes valores propriamente quando elementos da lista ligada são inseridos, removidos ou invertidos.
- Você deverá implementar `q_reverse` de forma que não seja necessário alocar qualquer memória adicional. Ao invés disso, seu código deverá modificar os ponteiros na lista existente. Implementações que necessitem alocar novos elementos na lista irão falhar no teste de performance, uma vez que os testes podem monitorar as chamadas `free` e `malloc`.
- Seu programa será testado sobre filas com mais de 1000000 de elementos. Você irá perceber que não pode processar longas listas utilizando funções recursivas, uma vez que isso requer muito espaço na pilha.

6 Testando

Siga as descrições no arquivo README

7 Avaliação

Seu programa será avaliado utilizando 14 registros de rastreamento. Você terá um crédito (7 ou 8 pontos, dependendo do registro) para cada um que seja executado corretamente, somando ao score máximo de 100 pontos.

O programa controlador `driver.py` executa `qtest` sobre os arquivos de rastreamento e computa o seu score.

8 Atividade

Utilizando `make` para gerar `qtest` também possui o efeito de gerar o arquivo handin.tar. Você deverá fazer isso numa máquina Linux. Você deverá enviar este arquivo (apenas este arquivo!) para o email do instrutor, com o seguinte titulo [SWB-LAB0], no corpo do seu email deverá conter o seu nome e matrícula. Somente emails com o título e o arquivo com extensão .tar serão avaliados.

9 Reflexão

Esta deverá ser uma atividade bem direta para os estudantes que estão totalmente preparados para realizar este curso. Se você encontrar algum problema para escrever este código ou ter alguma dificuldade para fazê-lo funcionar propriamente, isto pode ser um indicador de que você precise melhorar a sua habilidade de programação C até o próximo mês. Você irá precisar escrever programas que necessitam de domínio das habilidades testadas nesta atividade.