计算机系统基础 2019 春季学期

C Programming Lab:

Assessing Your C Programming Skills

1、概要

本实验主要是锻炼你的编程能力，你需要能够熟练地完成，尤其是在课程的后期作业中，这篇材料的内容你应该过一遍。这个实验主要检测以下这些技能：

- C 语言中要求的显式内存管理；
- 创建和操作基于指针的数据结构；
- 实现输入无效参数也能正确运行的健壮代码，包括 NULL 指针；
- 在 Makefile 中创建规则。

这个实验需要实现一个同时支持后进先出（LIFO）和先进先出（FIFO）的队列。提供的底层数据结构是一个单链表，你需要改进这些代码使一些操作更加高效。

2、下载作业

你的实验所需材料包含在名为 cprogramminglab-handout.tar 的 Linux 压缩文件中。登录 Linux 机器然后操作以下命令：

linux> tar xvf cprogramminglab-handout.tar

这会创建一个 cprogramminglab-handout 的目录，里面包含着一些实验所需的文件。有关这些文件的说明，请参阅 README。

3、底层数据结构

文件 queue.h 定义的结构如下：

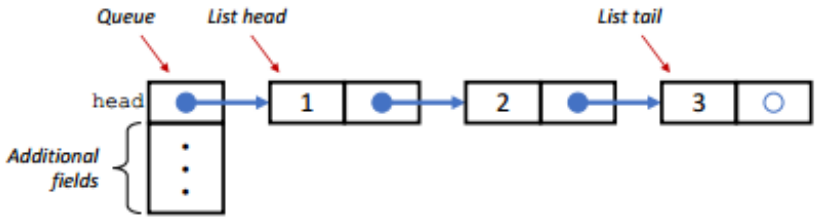

Figure 1: Linked-list implementation of a queue

```
/* Linked list element */
typedef struct ELE {
        int value;
        struct ELE *next;
 } list_ele_t;

   /* Queue structure */
    typedef struct {
        list_ele_t *head; /* Linked list of elements */
 } queue_t;
```

如图 1 所示，这些组合起来实现一个队列。队列的顶层表示是类型为 queue_t 的结构，在最初代码中，这个结构只包含一个单独的字段“head”，你需要添加其他字段。队列被表示为单链表，其中每个元素由具有字段“value”和“next”的 list_ele_t 结构表示，分别存储队列值和一个指向下一个元素的指针。尽管可以在不更改 list_ele_t 的情况下实现要求的所有函数，但欢迎你对 list_ele_t 进行更改，例如将单链表转换为双链表。

在给出的 C 代码中，队列是一个类型为 queue_t *的指针。 我们区分两种特殊情况：一个 **NULL 队列**是指针设置为 NULL 的队列。 而一个**空（empty）队列**是一个指向一个有效的 queue_t 结构的队列，该 queue_t 结构的头部字段被设置为 NULL。除了包含一个或多个元素的队列，你的代码还要能正确处理这两种情况。

4、编程任务

你的任务是修改 queue.h 和 queue.c 中的代码以实现以下功能：

q_new：创建一个新的空队列；

q_free: 释放队列使用的所有存储空间；

q_insert_head：在队列的头部插入新元素（LIFO）；

q_insert_tail：在队列的尾部插入新元素（FIFO）；

q_remove_head：从队列的头部删除一个元素；

q_size: 计算队列中元素的个数；

q_reverse：对列表重新排列，使得队列中元素的顺序都反转过来。

更多细节可以在这两个文件的注释中找到，包括如何处理无效操作（例如，从空队列或 NULL 队列中移除元素），以及函数应该具有哪些副作用和返回值。

以下是关于如何实现这些功能的一些重要注意事项：

(1) 当 q_new 和 q_insert_head 需要动态分配内存时，使用 malloc 或 calloc 来实现；

(2) 其中 q_insert_tail 和 q_size 需要满足时间复杂度为 O（1），即所需的时间与队列大小无关。你可以通过在 queue_t 数据结构中添加其他字段来完成此操作；

(3) q_reverse 的实现不需要分配额外的内存。相反，你的代码应该修改现有列表中的指针。 作为反向操作的一部分，直接或间接调用 malloc，calloc 或 free 的实现将导致错误；

(4) 你的程序将在超过 1,000,000 个元素的队列中进行测试，所以你不能使用递归函数来遍历这样的长列表，因为这需要太多的栈空间。

5、Build

在实验开始时，敲 make 命令并不会开始构建你的代码：

```
linux> make
```

你需要更新用于构建 qtest 的 Makefile 文件，将添加为 queue.o 要求，并在链接过程中包含 queue.o 。有关 Makefile 规则的说明请参考 https://www.cs.swarthmore.edu/~newhall/unixhelp/howto_makefiles.html。其中的“An example of building an executable from multiple .o files”部分对本作业很有帮助。正确的修改 Makefile 能使得在输入 make 命令时，如果 queue.o 发生

了变化，就重新生成 qtest。

6、测试

如果你正确编写了 Makefile，编译器将生成一个可执行程序 qtest，提供一个命令界面，你可以使用该界面创建，修改和检查队列。 关于可用命令的文档可以通过启动该程序并运行 help 命令找到：

```
linux> ./qtest
cmd>help
```

以下文件（traces/trace-eg.cmd）演示了一个示例命令序列：

```
# Demonstration of queue testing framework
# Initial queue is NULL.
show
# Create empty queue
new
# Fill it with some values. First at the head
ih 2
ih 1
ih 3
# Now at the tail
it 5
it 1
# Reverse it
reverse
# See how long it is
size
# Delete queue. Goes back to a NULL queue.
free
# Exit program
quit
```

在批处理模式下运行 qtest 能看到这些命令的效果：

linux> ./qtest -f traces/trace-eg.cmd

在初始代码中很多操作都没有正确实现。

traces 目录包含 11 个 trace 文件，其格式为 trace-k-cat.txt，其中 k 是 trace 号，cat 指明它测试的属性的类别。每个 trace 由一系列命令组成，与上面显示的相似，来测试程序的正确性、健壮性和性能的不同方面。你可以使用这写 trace 文件直接与 qtest 进行交互，以测试和调试你的程序。

7、评价

你的程序将使用 15 个 trace 文件进行评估，除了已经发给你们的 11 个外，每种类型新增加一个 trace（共 4 个）。每一个 trace 有一个分数值（6 或 7 分不等），总计为 100 分。每执行正确一个结果获得相应的分数。程序 driver.py 在 trace 上运行 qtest 并计算分数。你可以使用以下命令直接调用它：

linux> ./driver.py

或者

linux> make test

8、提交

用 make 生成 qtest 同时也会生成 handin.tar 文件。**将该文件重命名为 clab-handin-学号-姓名.tar**，例如 clab-handin-2014202110021-郭凯.tar。

将 tar 文件发送至邮箱：icscswhu@163.com，截止时间：2018 年 3 月 15 日 14:00。