Staque: A multi-file stack-queue application

- We build linked-list implementations of the stack and queue data structures.
- We write the following files.

defs.h	Defines a node data type.
stack.h	Defines the stack data type and the stack function prototypes.
queue.h	Defines the queue data type and the queue function prototypes.
stack.c	The implementations of the stack functions.
queue.c	The implementations of the queue functions.
staquecheck.c	A sample application with the *main* function.

The header file defs.h

- Both stacks and queues use nodes defined as follows.

```
typedef struct _node {
    int data;
    struct _node *next;
} node;

typedef node *nodep;
```

- Write these data-type definitions in **defs.h**.

The header file stack.h

```
typedef nodep stack;                      // Pointer to the beginning of the linked list

stack initstack ( ) ;                     // Create a new empty stack
int emptystack ( stack ) ;                // Check whether the input stack is empty
int top ( stack ) ;                       // Return the top of a stack (if non-empty)
stack push ( stack , int ) ;              // Push an integer to a stack
stack pop ( stack ) ;                     // Pop from a (non-empty) stack
void printstack ( stack ) ;               // Print the elements of a stack from top to bottom
stack destroystack ( stack ) ;            // Delete all the nodes from a stack
```

The header file queue.h

```
typedef struct {
    nodep front;                              // Pointer to the beginning of the linked list
    nodep back;                               // Pointer to the end of the linked list
} queue;

queue initqueue ( ) ;                         // Create a new empty queue
int emptyqueue ( queue ) ;                    // Check whether a queue is empty
int front ( queue ) ;                         // Return the element at the front of a queue (if non-empty)
queue enqueue ( queue , int ) ;               // Insert an integer at the front of a queue
queue dequeue ( queue ) ;                     // Delete an element from the back of a (non-empty) queue
void printqueue ( queue ) ;                   // Print the elements of a queue from front to back
queue destroyqueue ( queue ) ;                // Delete all the nodes from a queue
```

The file stack.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "defs.h"
#include "stack.h"

stack initstack ( )
{
    stack S;
    S = (stack)malloc(sizeof(node));
    S -> data = 0; S -> next = NULL;
    return S;
}
...
stack destroystack ( stack S )
{
    node *p;
    while (S) {
        p = S; S = S -> next; free(p);
    }
    return NULL;
}
```

The file queue.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "defs.h"
#include "queue.h"

queue initqueue ( )
{
    queue Q;
    node *p;
    p = (node *)malloc(sizeof(node));
    p -> data = 0;
    p -> next = NULL;
    Q.front = Q.back = p;
    return Q;
}
...
queue destroyqueue ( queue Q )
{
    node *p;
    while (Q.front) {
        p = Q.front;
        Q.front = (Q.front) -> next;
        free(p);
    }
    Q.front = Q.back = NULL;
    return Q;
}
```

The application staquecheck.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "defs.h"
#include "stack.h"
#include "queue.h"

#define ITER_CNT 10

int main ( )
{
    stack S;
    queue Q;
    int i;
    S = initstack();
    for (i=0; i<ITER_CNT; ++i) { S = push(S, rand() % 100); printstack(S); }
    S = destroystack(S);

    Q = initqueue();
    for (i=0; i<ITER_CNT; ++i) { Q = enqueue(Q, rand() % 100); printqueue(Q); }
    Q = destroyqueue(Q);

    exit(0);
}
```

Compile in one shot

```
$ gcc -Wall staquecheck.c stack.c queue.c
$ ls -l
total 48
-rwxr-xr-x 1 abhij abhij 17640 Dec 23 20:40 a.out
-rw-r--r-- 1 abhij abhij   152 Dec 23 19:43 defs.h
-rw-r--r-- 1 abhij abhij  1262 Dec 23 19:45 queue.c
-rw-r--r-- 1 abhij abhij   360 Dec 23 19:43 queue.h
-rw-r--r-- 1 abhij abhij  1098 Dec 23 19:45 stack.c
-rw-r--r-- 1 abhij abhij   315 Dec 23 19:43 stack.h
-rw-r--r-- 1 abhij abhij   983 Dec 23 20:34 staquecheck.c
$ ./a.out
...
$
```

- The option `-Wall` generates most of the relevant warning messages.
- Instead of `a.out`, you can generate an executable file of any name by the `-o` option.

```
$ gcc -Wall -o myapp staquecheck.c stack.c queue.c
$ ./myapp
```

- Never forget an executable name after –o. Writing the C source file name after –o will replace the file.

Generating individual object files

- Compile using the `-c` option.
- Does not require a *main* function.
- This does not generate an executable file (even if *main* is there).

```
$ gcc -Wall -c stack.c
$ gcc -Wall -c queue.c
$ gcc -Wall -o myapp staquecheck.c stack.o queue.o
$ ls -l
-rw-r--r-- 1 abhij abhij   152 Dec 23 19:43 defs.h
-rwxr-xr-x 1 abhij abhij 17640 Dec 23 21:01 myapp
-rw-r--r-- 1 abhij abhij  1262 Dec 23 19:45 queue.c
-rw-r--r-- 1 abhij abhij   360 Dec 23 19:43 queue.h
-rw-r--r-- 1 abhij abhij  3424 Dec 23 21:01 queue.o
-rw-r--r-- 1 abhij abhij  1098 Dec 23 19:45 stack.c
-rw-r--r-- 1 abhij abhij   315 Dec 23 19:43 stack.h
-rw-r--r-- 1 abhij abhij  3248 Dec 23 21:01 stack.o
-rw-r--r-- 1 abhij abhij   983 Dec 23 20:34 staquecheck.c
$ ./myapp
...
$
```

Difference between #include <. . .> and #include ". . ."

- There are default (system-dependent) directories for C header files.
  - /usr/include
  - /usr/local/include
- Header files residing in non-default directories should be included by the `#include "..."` directive.
- You can add to the list of default include directories by the `-I` option.

```
$ gcc -Wall -c -I. stack.c
$ gcc -Wall -c -I. queue.c
$ gcc -Wall -o myapp -I. staquecheck.c stack.o queue.o
```

- These compilations add the current directory to the list of include directories.
- You can now use `#include <defs.h>`, `#include <stack.h>`, and `#include <queue.h>` in the source codes.

The environment variable C_INCLUDE_PATH

- You can avoid the `-I` flag if you set `C_INCLUDE_PATH`.
- Multiple directories can be added as a colon-separated list DIR1:DIR2:DIR3:. . .
- . (the current directory) can be one of these directories.
- In bourne shell, this can be done as:

```
$ export C_INCLUDE_PATH=".:/home/foobar/include:/opt/users/foobar/include"
$
```

- C shell users should do this:

```
% setenv C_INCLUDE_PATH ".:/home/foobar/include:/opt/users/foobar/include"
%
```

Building the static staque library

- We have the files *defs.h*, *stack.h*, *queue.h*, *stack.c*, and *queue.c* as before.
- We want to build the static library *libstaque.a*. This will contain all the stack and queue functions as listed earlier.
- The library is not meant to contain any *main* function.
- Application programs like *staquecheck.c* will contain the *main* functions as needed.

- Compile individual source files with the –c option to generate the object files.
- Combine the object files into an archive *libstaque.a* using the command *ar*.

Generate libstaque.a

```
$ gcc -Wall -c stack.c
$ gcc -Wall -c queue.c
$ ar rcs libstaque.a stack.o queue.o
$ ls -l
-rw-r--r-- 1 abhij abhij  152 Dec 23 19:43 defs.h
-rw-r--r-- 1 abhij abhij 7046 Dec 24 18:25 libstaque.a
-rw-r--r-- 1 abhij abhij 1262 Dec 23 19:45 queue.c
-rw-r--r-- 1 abhij abhij  360 Dec 23 19:43 queue.h
-rw-r--r-- 1 abhij abhij 3424 Dec 24 18:23 queue.o
-rw-r--r-- 1 abhij abhij 1098 Dec 23 19:45 stack.c
-rw-r--r-- 1 abhij abhij  315 Dec 23 19:43 stack.h
-rw-r--r-- 1 abhij abhij 3248 Dec 24 18:23 stack.o
-rw-r--r-- 1 abhij abhij  144 Dec 23 19:43 staque.h
$
```

How to use the library

- To compile the application program *staquecheck.c* as given earlier.
- Include the header files *defs.h*, *stack.h*, and *queue.h*.
- A straightforward compilation fails.

```
$ gcc -Wall staquecheck.c
/usr/bin/ld: /tmp/ccIr2q5J.o: in function ‘main’:
staquecheck.c:(.text+0x12): undefined reference to ‘initstack’
/usr/bin/ld: staquecheck.c:(.text+0x57): undefined reference to ‘push’
/usr/bin/ld: staquecheck.c:(.text+0x67): undefined reference to ‘printstack’
/usr/bin/ld: staquecheck.c:(.text+0x7d): undefined reference to ‘destroystack’
/usr/bin/ld: staquecheck.c:(.text+0x8b): undefined reference to ‘initqueue’
/usr/bin/ld: staquecheck.c:(.text+0xdb): undefined reference to ‘enqueue’
/usr/bin/ld: staquecheck.c:(.text+0xf6): undefined reference to ‘printqueue’
/usr/bin/ld: staquecheck.c:(.text+0x113): undefined reference to ‘destroyqueue’
collect2: error: ld returned 1 exit status
$
```

How to link the library

- Like *–lm*, you should compile with *–lstaque*.

```
$ gcc -Wall staquecheck.c -lstaque
/usr/bin/ld: cannot find -lstaque
collect2: error: ld returned 1 exit status
$
```

- The linker *ld* does not look in the current directory for searching libraries.
- The *–L* option advises the linker to add directories to the library path.

```
$ gcc -Wall -L. staquecheck.c -lstaque
$ ls -l
-rwxr-xr-x 1 abhij abhij 17536 Dec 24 18:52 a.out
-rw-r--r-- 1 abhij abhij   152 Dec 23 19:43 defs.h
-rw-r--r-- 1 abhij abhij  7046 Dec 24 18:25 libstaque.a
-rw-r--r-- 1 abhij abhij  1262 Dec 23 19:45 queue.c
-rw-r--r-- 1 abhij abhij   360 Dec 23 19:43 queue.h
-rw-r--r-- 1 abhij abhij  3424 Dec 24 18:23 queue.o
-rw-r--r-- 1 abhij abhij  1098 Dec 23 19:45 stack.c
-rw-r--r-- 1 abhij abhij   315 Dec 23 19:43 stack.h
-rw-r--r-- 1 abhij abhij  3248 Dec 24 18:23 stack.o
-rw-r--r-- 1 abhij abhij   473 Dec 24 18:52 staquecheck.c
-rw-r--r-- 1 abhij abhij   144 Dec 23 19:43 staque.h
$
```