Separate Compilation

Yonglei Tao

Separate Compilation – stack.h

```
// A linked list implementation of a stack

#include <stdio.h>
#include <stdlib.h>

#define   EMPTY      0
#define   FULL       10000

typedef   char   data;
typedef   enum {false, true}   boolean;

struct node {
   data          d;
   struct node   *next;
};

typedef   struct node   node;
```

```
struct stack {
   int     cnt;
   node    *top;
};

typedef   struct stack   stack;

boolean   empty(const stack *stk);
boolean   full(const stack *stk);
void      initialize(stack *stk);
void      push(data d, stack *stk);
data      pop(stack *stk);
data      top(stack *stk);
```

Separate Compilation – stack.c

```
#include "stack.h"

boolean empty(const stack *stk) {
   return ((boolean) (stk -> cnt == EMPTY));
}

boolean full(const stack *stk) {
   return ((boolean) (stk -> cnt == FULL));
}

void initialize(stack *stk) {
   stk -> cnt = 0;
   stk -> top = NULL;
}

data top(stack *stk) {
   return (stk -> top -> d);
}
```

```
void push(data d, stack *stk) {
   node   *p;

   p = malloc(sizeof(node));
   p -> d = d;
   p -> next = stk -> top;
   stk -> top = p;
   stk -> cnt++;
}

data pop(stack *stk) {
   data   d;
   node   *p;

   d = stk -> top -> d;
   p = stk -> top;
   stk -> top = stk -> top -> next;
   stk -> cnt--;
   free(p);
   return d;
}
```

Separate Compilation – main.c

```
// Test the stack implementation
// by reversing a string.

#include "stack.h"

int main(void) {
   char    str[] = "My name is Tom!";
   int     i;
   stack   s;

   initialize(&s);
   printf(" In the string: %s\n", str);

   for (i = 0; str[i] != '\0'; ++i) {
      if ( !full (&s) )
         push (str[i], &s);
   }
   printf("From the stack: ");

   while ( !empty (&s) )
      putchar (pop(&s));
   putchar ('\n');

   return 0;
}
```

Managing Dependencies - makefile

```
main: main.o stack.o
      gcc main.o stack.o –o main
main.o: main.c stack.h
stack.o: stcak.c stack.h
```