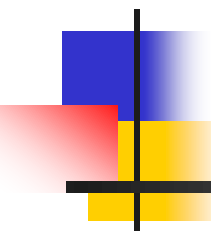


SEEM3460 Tutorial

Multi-module programming in C



SEEM3460 Tutorial

Multi-module programming in C

Before we start:

1. **Please log on to the server**
2. **Please recall how to edit text files on Linux/Unix (vi/vim/nano)**



Why Multi-module

- Some functions can be reused in many places.
- Writing all the code in one file will make the code too long and not easy to maintain.
- It is quite time consuming and we may need to replace every copy of the code if we plan to revise this function

```

/* main.c */
#include <string.h>
void func_1(...);
...
void func_n(...);

int main(...) {
    ...
}

void func_1(...) {
    ...
}

...
void func_n(...) {
    ...
}

```

```

/* func.c */
#include <string.h>
void func_1(...);
...
void func_n(...);

void func_1(...) {
    ...
}

...
void func_n(...) {
    ...
}

```

```

/* main.c */
#include <string.h>
void func_1(...);
...
void func_n(...);

int main(...) {
    ...
}

```

```

/* func.h */
void func_1(...);
...
void func_n(...);

```

```

/* func.c */
#include <string.h>
#include "func.h"

void func_1(...) {
    ...
}

...
void func_n(...) {
    ...
}

```

```

/* main.c */
#include <string.h>
#include "func.h"

int main(...) {
    ...
}

```



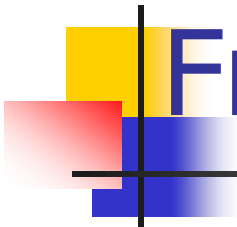
What we will do today

- We have two programs "ask_reverse" and "ask_palindrome" . But both of them have all the code in one single file. We will partition them into several modules.
- Task 1: Partition "ask_reverse" into two parts. One is the main program while another is the "reverse" function.
- Task 2: Partition "ask_palindrome" into three parts, i.e. the main program, the "reverse" function and "palindrome" function. We can reuse the "reverse" function in Task 1.

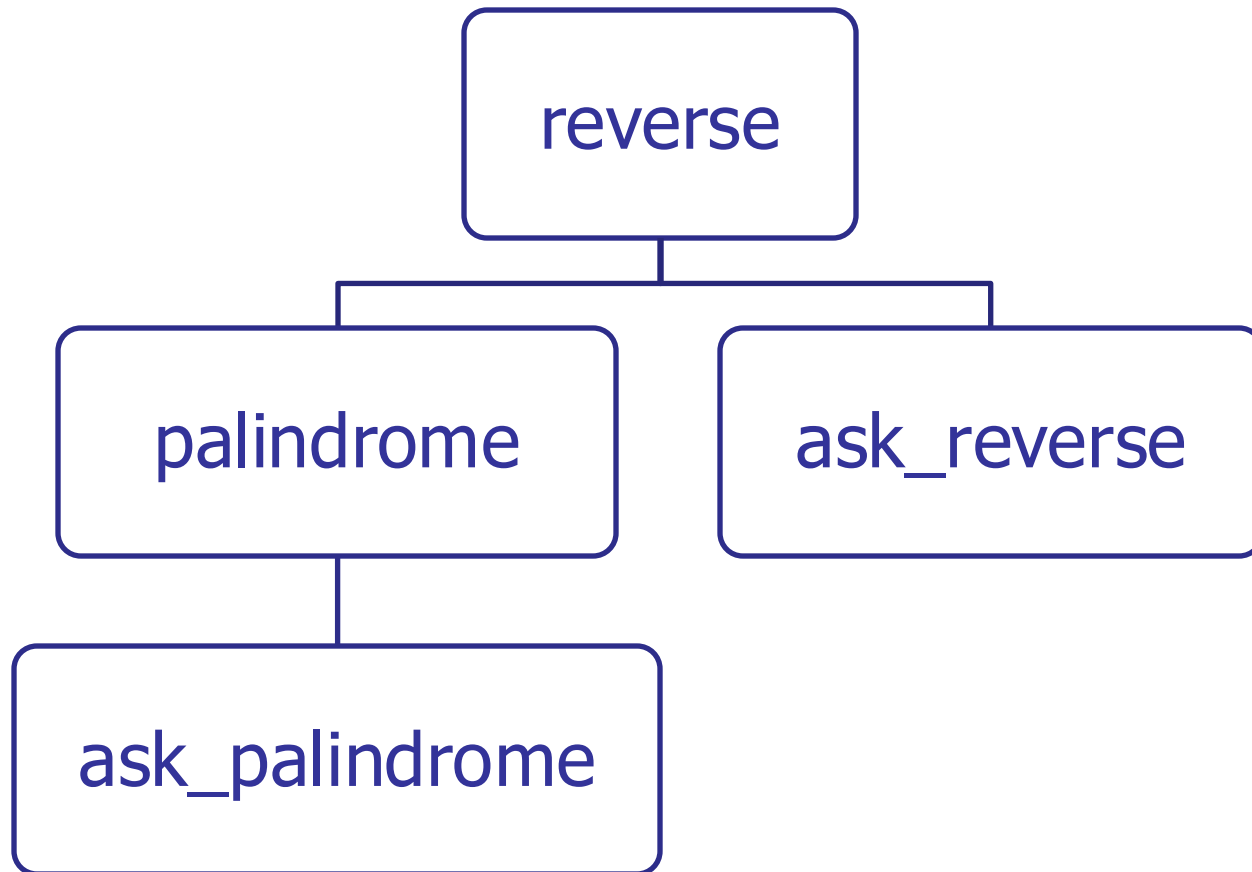


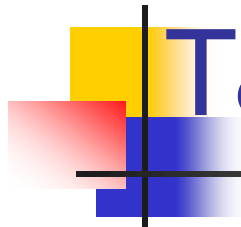
Copy the material

- Create the directory
 - `mkdir c_multi`
 - `cd c_multi`
- Copy files
 - `cp ~seem3460/distribute/c_multi-module/ask/ask_reverse.c ./`
 - `cp ~seem3460/distribute/c_multi-module/ask/ask_palindrome.c ./`

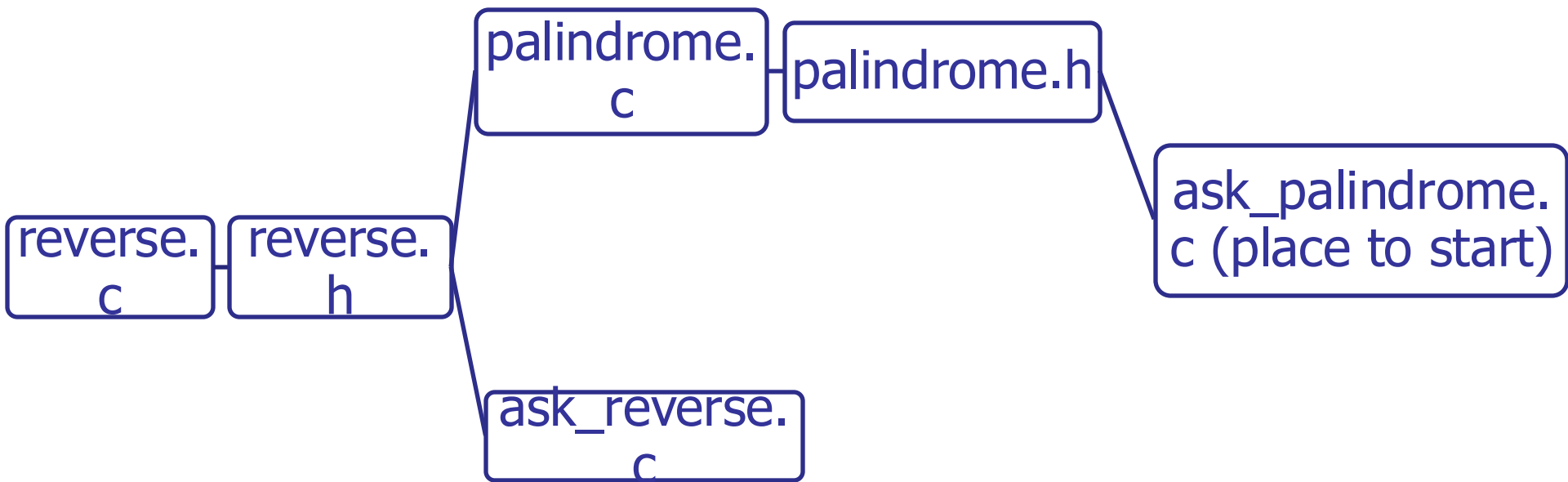


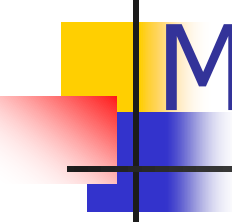
Framework 1





Target Structure





Modularizing Reverse

- Create "reverse.h" and "reverse.c"
 - How to paste under unix:
 - Shift+insert, right click (not in x-win32)
 - With the help of copy and paste
 - >cp ask_reverse.c reverse.c
 - >cp reverse.c reverse.h
 - Edit "reverse.c" and "reverse.h"
- Edit "ask_reverse.c"

Modularizing Reverse

reverse.h

```
void reverse(...);
```

reverse.c

```
#include <string.h>  
#include "reverse.h"
```

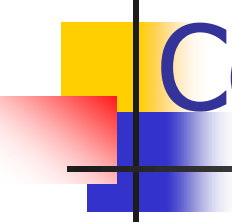
```
void reverse(...) {  
    ...  
}
```

ask_reverse.c

```
#include <stdio.h>  
#include <string.h>  
#include "reverse.h"
```

```
int main() {  
    ...  
}
```

```
void reverse(...) {  
    ...  
}
```



Compile

- Create .o files:
 - `gcc -c reverse.c`
 - `gcc -c ask_reverse.c`
- Link .o files as executables:
 - `gcc -o ask_reverse ask_reverse.o reverse.o`
- One code for all:
 - `gcc -o ask_reverse reverse.c ask_reverse.c`

Modularizing palindrome

palindrome.h

```
int palindrome(...);
```

palindrome.c

```
#include <string.h>
#include "palindrome.h"
#include "reverse.h"

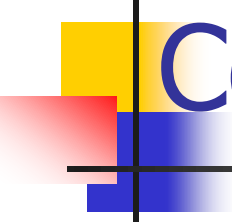
int palindrome(...) {
    ...
}
```

ask_palindrome.c

```
#include <stdio.h>
#include <string.h>
#include "palindrome.h"
int palindrome(...) {
    ...
}

void reverse(...) {
    ...
}

int main(...) {
    ...
}
```



Compile

- Create .o files:
 - `gcc -c palindrome.c`
 - `gcc -c ask_palindrome.c`
- Link .o files as executables:
 - `gcc -o ask_palindrome ask_palindrome.o
palindrome.o reverse.o`
- One code for all:
 - `gcc -o ask_palindrome reverse.c palindrome.c
ask_palindrome.c`