

Tutorial 9

Miscellaneous

Jiawei Li (Michael)

Office hours: Monday 1:00-3:00pm

Office: PMB426

Email: jiawei.li@Nottingham.edu.cn

Static variables

- A static variable remains in memory while the program is running.
- Static variables are allocated memory in data segment, not stack segment.
- A static variable is initialized only once.
- Static variables (like global variables) are initialized as 0 if not initialized explicitly.
- static variables can only be initialized using constant literals.

```
static int x = 1; // can be either global or local
```

Exercise 1

```
#include<stdio.h>

void func()
{
    static int value = 0;
    value++;
    printf("value: %d\n",value);
}

int main()
{
    for(int i=0;i<5;i++)
        func();
}
```

```
#include<stdio.h>
int initializer(void)
{
    return 50;
}

int main()
{
    static int i = initializer();
    printf(" value of i = %d", i);
    getchar();
    return 0;
}
```

Static functions

- Access to static functions is restricted to the file where they are declared. Therefore, when we want to restrict access to functions, we make them static.

```
static int fun(void)
{
    printf("I am a static function ");
}
```

```
void main ()
{
    fun();
}
```

Use multiple files

- Multiple files can be combined together by using the keyword 'include'.

```
// This is the file of "fun.c"  
#include <stdio.h>  
int fun(void)  
{  
    printf("I am in fun.c file.\n");  
    return 1;  
}
```

```
// This is the file "main.c"  
#include "fun.c"  
int main()  
{  
    fun();  
    return 1;  
}
```

- Compile multiple files together to create an executable program (so no need to 'include' them).

Compile multiple files

- `gcc -std=c99 -lm -Wformat -Wwrite-strings main.c fun.c -o test`

```
// This is the file of "fun.c"
```

```
#include <stdio.h>
int fun(void)
{
    printf("I am in fun.c file.\n");
    return 1;
}
```

```
// This is the file "main.c"
```

```
int fun(void);
int main()
{
    fun();
    return 1;
}
```

Global variables vs. static variables

- Similarity: memory, life span
- Difference: declaration, access, initialization

```
int global_var = 1;
...

void func()
{
    static int static_var = 2;
    ...
}
```

Const variables

- A constant variable cannot be changed.

```
const double PI = 3.1415;  
...  
PI = 3.14; // This is an error
```

- A const pointer means that the address cannot be changed.

```
char * const ptr = &string1;  
ptr = &string2; // This is wrong
```

- A pointer to const value means that the pointed value cannot be changed.

```
const char *ptr = &string1;  
string1[0] = 's'; // This is wrong  
ptr = &string2; // This is okay
```


Macro

- Whenever a macro name is used, C preprocessor will replace it with the body of the macro.

```
#define PI 3.1415
```

- You can also define macros that works like a function call, known as function-like macros.

```
#define circleArea(r) (3.1415*(r)*(r))
```

- Predefined Macros: `__DATE__`, `__TIME__`, `__FILE__`

```
printf("Current time: %s %s\n", __DATE__, __TIME__);
```

Task: Complete a Trie

- You are given a partially-completed program which implements a Trie. (`trie_incomplete.c` in the Tutorial section)
- Your task is to complete the function `trie_contains()` so that the program print "Test passed" when run.
- The `trie_contains` function should check if a given string is present in the trie or not.
- `assert(expression)` is a macro in C. If `expression` is true, it does nothing. If `expression` is false, it terminates the program.