Tutorial 9
Miscellaneous

Jiawei Li (Michael)

Office hours: Tuesday 3:00-5:30pm
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 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 change.

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

- A const pointer means that the value of the pointer cannot change.

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

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

```
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__);
```

File I/O

- File pointer:

```
FILE *fp;
fp=fopen (“filename”, ”‘mode”);
```

fscanf()

fgetc () function reads a character from file.

fgets () function reads string from a file, one line at a time.

- We need to know the location of the file pointer when reading or writing a file.

feof(fp) returns TRUE if it reaches the end of a file.

fseek() function is used to move file pointer position to the given location.

What is an EOF?

- EOF is NOT a char.
- Why can we find the end of a file via EOF?
- How to detect EOF?

```
if(fscanf(fp,“%c”, &ch) != EOF)
…
```

Task: file operation

The word guessing game in Lab10:

A dictionary file containing a large amount of words is here: /usr/share/dict/words

Each word is saved as a string in the file. Different words are separated by '\n'.

Your task is to check how many words the dictionary contains.