CS 2211
Systems Programming

Part Eight (a): Variables

VARIABLES in C

Why have different *kinds* of variables in programming language

Some information is used only within one function.

Some information is used by multiple functions

Recall that **Java** has **4 different *kinds*** of variables:

Class variables (**long term** information storage)

Instance variables (**long term** information storage)

Local variables (**short term** information storage)

Parameter variables (**short term** information storage)

VARIABLES in C

C has ***five* (5) kinds of variables** - divided into ***2 categories***:

***compile-time* (allocated) variables:**

Global variables --- accessible everywhere
***Static* global variables** --- accessible within the ***same* C program file**
***Static* local variables** --- accessible within the ***same* C function**

Common property:
The life time of these kinds of variables is the whole execution period of the C program

***run-time* variables:**

Local variables --- accessible only in the block they are declared in
Parameter variables --- accessible within the called ***function***

Common property:
The **memory space** used for these **kinds** of **variables** are **allocated (reserved)** ***during*** the **execution (running)** of the program

They are **created** on the **system stack**.

VARIABLES in C

LOCAL VARIABLES

```
#include <stdio.h>

int divideMe( int k, n )     // k and n local to this function only
{
     int z;                  // z local to this function only
     z = k / n;
     return(z);
}


int main(int argc, char *argv[])
{
      int y,d,;    // local to just main

      x=9;
      d=2;
      for (int i = 0; i <10; i++)
      {
         y = x * i;  // i only local to this for loop block
      }
      x = divideMe(y,d);
}
```

VARIABLES in C

GLOBAL VARIABLES

```
#include <stdio.h>

int z;
int n;

int divideMe( int k )     // k local to this function only
{
     z = k / n;           // z and n global to the entire code
     return(10);
}


int main(int argc, char *argv[])
{
      int x, y,;      // local to just main

      x=9;
      n=2;             // n global to the entire code
      for (int i = 0; i <10; i++)
      {
         y = x * i;   // i only local to this for loop block
      }
      x = divideMe(y);
      y = (x = 10) ? z : -1; // z global to the entire code
}
```

VARIABLES in C

STATIC LOCAL VARIABLES

```
#include <stdio.h>

void foo()
{
    int a = 10;
    static int sa = 10;

    a += 5;
    sa += 5;


}


int main()
{
    int i;

    for (i = 0; i < 10; ++i)
        foo();

    printf("a = %d, sa = %d\n", a, sa);
}
```

留在函数中下次使用.

```
a = 15, sa = 15
a = 15, sa = 20
a = 15, sa = 25
a = 15, sa = 30
a = 15, sa = 35
a = 15, sa = 40
a = 15, sa = 45
a = 15, sa = 50
a = 15, sa = 55
a = 15, sa = 60
```

VARIABLES in C

STATIC GLOBAL VARIABLES

```
#include <stdio.h>

static int z;
static int n;

int divideMe( int k )     // k local to this function only
{
    z = k / n;            // z and n global to the entire code
    return(10);
}


int main(int argc, char *argv[])
{
     int x, y,;      // local to just main

     x=9;
     n=2;             // n global to the entire code
     for (int i = 0; i <10; i++)
     {
        y = x * i;   // i only local to this for loop block
     }
     x = divideMe(y);
     y = (x = 10) ? z : -1; // z global to the entire code
}
```

Variable Scope in C

END OF SECTION

Static：静态：在此源文件中有效且在此函数被调用后仍保有数据.

Global：全局：在所有源文件中有效/在所有函数中有效.

Static global：只在定义该函数的源文件中所有函数有效.