COMP 4511

Bridging From C++ to C

Goals

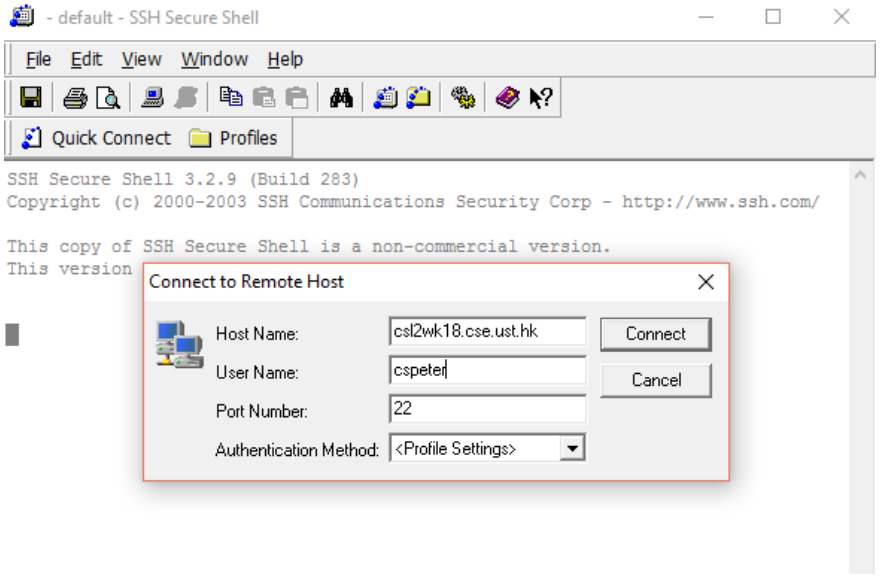
- The lab is not teaching C programming within a few hours!
- It tries to bridge you from C++ to C
- Key comparisons between C and C++ are highlighted

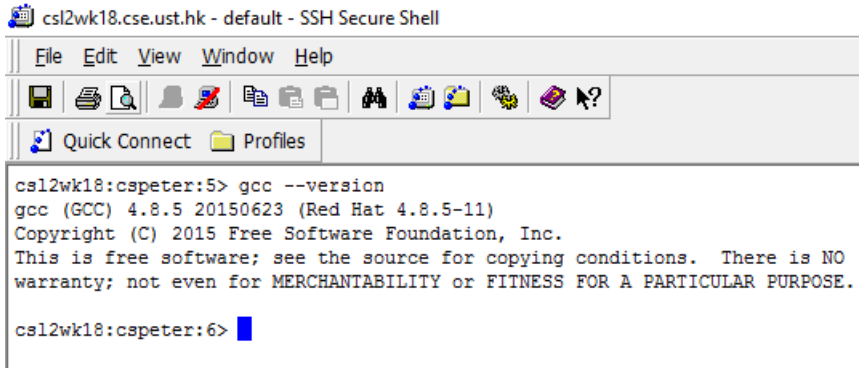
Why C Programming?

- C is a powerful, portable and efficient programming language
- The Linux kernel is written in the version of the C programming language (plus assembly language as well)
- To add/modify features in the Linux kernel in this course, you MUST know how to do C programming
- Comparing with C++, C is not a completely new language. You can pick it up fast if you are familiar with C++

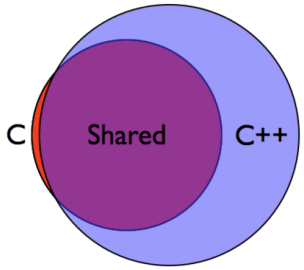
Getting Started

- In this lab, we can use SSH Secure Shell client to connect to any CS Linux Lab 2 machine. In Mac OS X, you can open a terminal and type in SSH command.
  - Host Name: csl2wkXX.cse.ust.hk, where XX=01..40
  - User Name: Your ITSC user name
  - Port Number: 22 (default)
  - Ensure that `gcc (GNU compiler collections)` is installed





Overview: C and C++



Shared features in C/C++

- Variables and primitive data types
- Operators (i.e. +, -, *…)
- Expressions and Statements
  - Example expression: a>4
  - Example statement: i=2+3
- Branching statements and loops (i.e. If-else, switch-case, while-loop, for-loop, ...)
- Functions
- Pointers
- Arrays

Things that are different

- Functions: No pass by reference in C
- Class: No class in C
- Struct in C
  - No member functions are allowed No access modifiers (i.e. private, public, protected) or inheritance
- String manipulation: No string class. Use char* directly
- Dynamic memory: No new/delete operators! Use malloc(), free()… instead
- No Standard Template Library (STL) in C
- I/O library: No iostream/fstream, cout, cin...

Quick Start: HelloWorld in C and C++

- HelloWorld in C

hello.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    cout << "Hello World!" << endl;
    return 0;
}
```

- HelloWorld in C++

hello.c

```c
#include <stdio.h>

int main() {
    printf("Hello World!\n");
    return 0;
}
```

Compile and Link a source file

- In C++ (Using g++)
  - g++ -c hello.cpp
  - g++ -o hello_cpp hello.o
- In C (Using gcc)
  - gcc –c hello.c
  - gcc –o hello_c hello.o
- The compiled C program is usually smaller

```
 74B Dec 26 11:38 hello.c
105B Dec 26 11:37 hello.cpp
8.2K Dec 26 11:45 hello_c
 15K Dec 26 11:45 hello_cpp
```

- C++ has a much larger library than C, something may not be automatically linked in C
- Example: A program using Math-related functions
- C++
  - g++ math_program.cpp
- C
  - Explicit linking maybe necessary
  - gcc math_program.c -lm

Functions in C: Pass-by-value and No pass-by-reference

- Pass-by-reference in C++

pass_by_reference.cpp

```cpp
#include <iostream>
using namespace std;
void swapInt(int& a, int& b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
int main() {
    int a = 3;
    int b = 2;
    cout << "Before: " << a << " " << b << endl;
    swapInt(a, b);
    cout << "After : " << a << " " << b << endl;
    return 0;
}
```

- Equivalent implementation of pass-by-reference using pointers

pass_by_ref_equivalent.c

```c
#include <stdio.h>
void swapInt(int *a, int *b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
int main()
{
    int a = 3;
    int b = 2;
    printf("Before: %d %d\n", a, b);
    swapInt(&a, &b);
    printf("After : %d %d\n", a, b);
    return 0;
}
```

C Struct

- C++ struct (with member functions and access modifiers)
  - Note: Invalid in C!
- C struct
  - separated functions are required
  - no access modifiers

```
struct IntPair {
public:
    IntPair(): first(0), second(0) {}
    void setFirst(int a) { first=a; }
    void setSecond(int b) { second=b; }
    int getFirst() {return first; }
    int getSecond() {return second; }
    int sum() {return first+second; }
private:
    int first, second;
};
```

```
struct IntPair {
    int first, second;
};

int IntPair_sum(struct IntPair pair)
{
    return pair.first + pair.second;
}
```

String manipulation in C: string storage and manipulation

- C++ (string object is pretty convenience)
- Equivalent in C (NULL-terminated character array)

string_storage.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string str = "Hello World!";
    cout << "String: " << str << endl ;
    cout << "First char: " << str[0] << endl;
    cout << "Last char: " << str[str.size()-1] << endl;
    return 0;
}
```

string_storage.c

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char str[100];
    strcpy(str, "Hello World!");

    printf("String: %s\n", str);
    printf("First char: %c\n", str[0]);

    int i = 0;
    while ( str[i] != '\0' && i < 100 )
        i++;
    printf("Last char: %c\n", str[i-1] );

    return 0;
}
```

C string library reference: http://www.cplusplus.com/reference/cstring/

Dynamic Memory in C: malloc/calloc

- C++ (using new operator)

```
dynamic_mem.cpp
1  #include <iostream>
2  using namespace std;
3
4  struct IntPair
5  {
6      int first, second;
7  };
8
9  int main()
10 {
11     int *intVar = new int;
12     int *intArr = new int[100];
13     IntPair *intPair = new IntPair;
14
15     *intVar = 10;
16     for (int i=0; i<100; i++)
17         intArr[i] = i;
18     intPair->first = 1;
19     intPair->second = 2;
```

- C (using malloc/calloc)
  - Unlike calloc, malloc won't initialize all bits as 0s

```
dynamic_mem.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct IntPair
5  {
6      int first, second;
7  };
8
9  int main()
10 {
11     int *intVar = malloc(sizeof(int));
12     int *intArr = malloc(sizeof(int)*100);
13     struct IntPair *intPair = malloc(sizeof(struct IntPair));
14
15     *intVar = 10;
16     for (int i=0; i<100; i++)
17         intArr[i] = i;
18     intPair->first = 1;
19     intPair->second = 2;
```

malloc reference: http://www.cplusplus.com/reference/cstdlib/malloc/
calloc reference: http://www.cplusplus.com/reference/cstdlib/calloc/

Dynamic Memory in C: free

- C++ (using delete operator)

```
delete intVar;
delete [] intArr;
delete intPair;
```

- C (using free)

```
free(intVar);
free(intArr);
free(intPair);
```

I/O Library: Standard IO (scanf/printf)

- C++

```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
    int age;
    string name;

    cout << "Enter your name: " ;
    cin >> name ;
    cout << "Enter your age: " ;
    cin >> age;

    cout << "Welcome " << name << "! "
         << "You are " << age << " years old!"
         << endl;

    return 0;
```

- C

```c
#include <stdio.h>

int main()
{
    int age;
    char name[100];

    printf("Enter your name: ");
    scanf("%s", name);
    printf("Enter your age: ");
    scanf("%d", &age);

    printf("Welcome %s! You are %d years old!\n",
        name, age);

    return 0;
}
```

A comprehensive "scanf" tutorial: http://www.cplusplus.com/reference/cstdio/scanf/
A comprehensive "printf" tutorial: http://www.cplusplus.com/reference/cstdio/printf/

String manipulation: sscanf, sprintf

- In C, you can manipulate string using scanf-printf like functions: sscanf and sprintf
- You only need to specify an extra parameter, which is a pointer pointing to a character array

- Example:

sscanf_sprintf.c

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char line[100], linuxText[20], machineName[20], kernelText[50];
    int kernelMajor, kernelMinor;

    strcpy(line, "Linux csl2wk19 3.10.0-327.3.1.el7.x86_64");

    sscanf(line, "%s %s %s", linuxText, machineName, kernelText);
    sscanf(kernelText, "%d.%d", &kernelMajor, &kernelMinor);

    printf("Version: %d.%d\n", kernelMajor, kernelMinor);

    return 0;

}
```

File I/O Example: Sum up 2 numbers from a file and put the result to a file

- C++ (ifstream/ofstream)

io_file.cpp

```cpp
#include <fstream>
using namespace std;

int main()
{
    int a, b;

    ifstream fin("input.txt");
    fin >> a >> b ;
    fin.close();

    int result = a + b;
    ofstream fout("output.txt");
    fout << "The sum of " << a << " & "
         << b << " is " << result << endl;
    fout.close();

    return 0;
}
```

- C (fopen, fclose, fscanf, sprintf)

io_file.c

```c
#include <stdio.h>

int main()
{
    int a, b;
    FILE *fin = fopen("input.txt", "r");
    fscanf(fin, "%d %d", &a, &b);
    fclose(fin);

    int result = a + b;
    FILE *fout = fopen("output.txt","w");
    fprintf(fout, "The sum of %d & %d is %d\n",
            a, b, result);
    fclose(fout);

    return 0;
}
```

Reference: http://www.cplusplus.com/reference/cstdio/fscanf/