Chapter Ten

Developing UNIX Applications
In C and C++

Lesson A

C Programming in a UNIX Environment

Objectives

- Create simple C programs
- Understand C program structure
- Use the make utility to help with program development and maintenance
Introducing C Programming

- C is the language in which UNIX was developed and refined
- Known for its efficiency and brevity
- C program is set of functions
  - Must have main function
- C programs are stored in
  - Header files: .h
  - Body files: .c

Creating a C Program

```c
#include <stdio.h>

main() {
    printf("hello world\n");
}
```

- Stored as: one.c
- Compiled and linked as: gcc one.c
- Run as: ./a.out
Compilation Process

A simple C program

```c
#include <stdio.h>
main(int argc, char* argv[]) {
    if (argc > 1)
        printf("hello %s\n", argv[1]);
    else
        printf("hello world\n");
}
```

command line arguments

A simple C program

- to compile:
  - gcc two.c
    produces: a.out
  - or
  - gcc -c two.c
    produces: two.o
A simple C function

```c
#include <math.h>

int compute(int x, int y) {
    double tx = x, ty = y, tr;
    tr = pow(tx, ty);
    return (int) tr;
}
```

- uses math library

Multiple source files

```
#include <stdio.h>

main() {
    int x, y, p;
    printf("This program computes x^y\n");
    printf("Enter x: ");
    scanf("%d", &x);
    printf("Enter y: ");
    scanf("%d", &y);
    p = compute(x, y);
    printf("Result: %d\n", p);
}
```

- asks for user input
Steps to create an executable

- to compile:
  - gcc -c compute.c
  - gcc -c power.c
  - gcc compute.o power.o -lm -o power
- to run
  - ./power

make utility

- has configuration file that specifies dependencies: makefile
- checks timestamps on files
- recompiles necessary files

makefile

```
makefile

power: power.o compute.o
    gcc power.o compute.o -o power -lm
power.o: power.c
    gcc -c power.c
compute.o: compute.c
    gcc -c compute.c
```

```
Lesson B
C++ Programming in a UNIX Environment

Objectives
- Create a simple C++
- Understand multi-file organization of a C++ program
- Make

Introducing C++ Programming
- C++ is a programming language that builds on C to add object-oriented capabilities
- C and C++ are similar in many ways
Creating a C++ Program

```cpp
#include <iostream>

main() {
    cout << "hello world" << endl;
}
```

- Stored as: one.cc
- Compiled and linked as: g++ one.cc
- Run as: ./a.out

---

A simple C++ program

```cpp
#include <iostream>

main(int argc, char* argv[]) {
    if (argc > 1)
        cout << "hello " << argv[1] << endl;
    else
        cout << "hello world\n";
}
```

- Command line arguments

---

A simple C++ program

- To compile:
  - g++ two.cc
    - Produces: a.out
  - Or
  - g++ -c two.cc
    - Produces: two.o
C++ classes

- Class header and body
- Header in header file .h
- Body in body file .cc or .C
  - includes header file
- main function still required

A simple C++ class header

```cpp
#ifndef COUNTER_H
#define COUNTER_H
class Counter {
  int value;
  public:
    Counter();
    void increment(int);
    void reset();
    int getValue();
};
#endif
```

A simple C++ class body

```cpp
#include "Counter.h"
Counter::Counter() {
    reset();
}
void Counter::increment(int n=1) {
    value += n;
}
void Counter::reset() {
    value = 0;
}
int Counter::getValue() {
    return value;
}
```
main C++ program

```c++
#include <iostream>
#include "Counter.h"
main() {
    Counter c;
    int value;
    cout << "enter value: ";
    cin >> value;
    c.increment(value);
    cout << "counter now: "
    << c.getValue() << endl;
}
```

makefile

```
count: Counter.o main.o
    g++ Counter.o main.o -o count
Counter.o: Counter.C
    g++ -c Counter.C
main.o: main.C
    g++ -c main.C
```

Chapter Summary

- The major difference between C and C++ is that C follows procedural principles and C++ primarily follows object-oriented programming.
- The make utility is used to maintain the application's source files.