RIPHAH INTERNATIONAL UNIVERSITY, ISLAMABAD

Bachelors of Computer Science – 6th Semester

Subject:OS

LAB#6

Submitted by: Ammara tahir –41345

QUESTION#1

Let’s say I have a simple C program named `hello.c`:

```
#include <stdio.h>


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

To compile and run this program:

1. Save it as `hello.c`.
2. Open the terminal and type:

```
gcc -o hello hello.c
```

To run the program, type:

```
./hello
```

This will output:

```
Hello, World!
```

QUESTION#2

Let's suppose the example of question 1

Without `-o` Option:

If you compile the program without specifying `-o`, like this:

```
gcc example.c
```

This will produce an output file named `a.out`. You would run the program using:

```
./a.out
```

With `-o` Option:

If you compile the program with the `-o` option, like this:

```
gcc -o program Hello.c
```

This will produce an output file named `program`. You would run the program using:

```
./program
```

QUESTION#3

gcc:

- Purpose: `gcc` (GNU Compiler Collection) is used primarily for compiling C programs, though it can also compile C++ programs with the right options.
- Behavior: When used with C code, it treats the code as C code by default. You can use `gcc` for C++ code.
- Usage: Use `gcc` when you are working with C programs
- Use `gcc` when you are working with C code or when you want to compile C++ code but have more control over linking

gcc -o program program.c

g++:

- Purpose: `g++` is specifically designed to compile C++ programs. It automatically links the C++ standard library without requiring additional flags.
- Behavior: When compiling with `g++`, it treats the code as C++ by default. It's more convenient when working with C++ code since it handles all C++-specific features (like linking to C++ libraries) automatically.

- Usage: Use `g++` when you are writing and compiling C++ programs.
- Use `g++` when you are compiling C++ code, as it handles C++-specific libraries and conventions automatically.

```
g++ -o program program.cpp
```

QUESTION#4

Compile the C++ program:

```
g++ -o output_Hello program.cpp
```

1. `output_Hello`
2. `program.cpp`

Run the compiled program:

```
./output_Hello
```

QUESTION#5

```
#include <iostream>

using namespace std;


// Function template to swap two variables of any type

template <typename T>

void swapValues(T& a, T& b) {

   T temp = a;

   a = b;

   b = temp;

}
```

```
int main() {
    int x = 5, y = 10;
    cout << "Before swap: x = " << x << ", y = " << y << endl;
    swapValues(x, y);  // Uses the template for int
    cout << "After swap: x = " << x << ", y = " << y << endl;

    double m = 1.2, n = 3.4;
    cout << "Before swap: m = " << m << ", n = " << n << endl;
    swapValues(m, n);  // Uses the template for double
    cout << "After swap: m = " << m << ", n = " << n << endl;

    return 0;
}
```

Explanation:

- `template <typename T>`: This defines a template where `T` is a placeholder for any data type (like `int`, `float`, etc.).
- `swapValues(T& a, T& b)`: This function takes two arguments of type `T` (which could be any type, such as `int`, `float`, etc.) and swaps their values.

QUESTION#6

In C programming, file extensions like `.c` (for C) and `.cpp` (for C++) help the compiler identify the correct language and apply proper syntax rules. They ensure the right standard libraries are linked, prevent compilation errors, and improve code readability. IDEs and tools rely on extensions for features like syntax highlighting and debugging. Using incorrect extensions can cause compilation issues, especially in mixed C/C++

projects. Therefore, proper extensions ensure smooth compilation and clear distinction between C and C++ code.

QUESTION#7

1. Syntax Errors:

- Description: These occur due to mistakes in the structure of the code, such as missing semicolons, unmatched parentheses, or incorrect variable declarations.
- Resolution: Check the compiler error messages to identify the line number where the issue occurred, then correct the syntax by adding missing characters or fixing typos.

Example: Missing semicolon at the end of a statement.

```
int x = 5  // Error: missing semicolon
```

2. Undeclared Variables:

- Description: This happens when a variable is used without being declared.
- Resolution: Declare the variable before using it, making sure to specify the correct data type.

Example:

```
printf("%d", x);  // Error: 'x' is undeclared
```

Fix:

```
int x = 5;
printf("%d", x);
```

3. Missing Header Files:

- Description: This occurs when you use a function or feature without including the appropriate header file.
- Resolution: Include the necessary headers at the top of your file (e.g., `#include <stdio.h>` for input/output functions).

Example:

```
printf("Hello, World!");  // Error: implicit declaration of 'printf'
```

Fix:

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

4. Type Mismatch Errors:

- Description: Occurs when you try to assign a value of one type to a variable of a different type, or when passing incorrect types to functions.
- Resolution: Ensure that variables and function parameters have compatible data types.

Example:

```
int x = 3.5;  // Error: assigning float to an int
```

5. Linking Errors (Undefined References):

- Description: These happen when you call a function, but the compiler cannot find its definition (often occurs if a library is not linked properly).
- Resolution: Ensure that all functions are defined, and if external libraries are used, link them correctly (e.g., `gcc -o program program.c -lm` for math functions).

Example:

```
sqrt(9);  // Error: undefined reference to `sqrt`
```

Fix:

```
#include <math.h>
```

```
gcc -o program program.c -lm
```

QUESTION#8

Key Permissions:

- Read (r), Write (w), Execute (x).
- Apply to User (u), Group (g), and Others (o).

Common `chmod` Commands:

Grant execute permission to the user:

```
chmod u+x filename
```

1.Grant execute permission to everyone:

```
chmod a+x filename
```

2.Remove execute permission for everyone:

```
chmod a-x filename
```

3.Numeric Mode:

To set executable permissions using numbers (e.g., `777` for full user access):

```
chmod 777 filename
```

QUESTION#9

A tarball is a compressed archive file (`.tar.gz` or `.tar.bz2`) used in Linux to bundle and distribute multiple files.

Advantages:

- Bundling & Compression: Simplifies distribution and reduces file size.
- Portability: Works across different systems without relying on package managers.

- Source Distribution: Ideal for sharing source code for custom builds.

Limitations:

- Manual Installation: Requires manual steps for extraction, compilation, and installation.
- No Dependency Management: Users must manually handle software dependencies.
- No Version Control or Uninstall: Difficult to manage versions or uninstall compared to package managers.

Tarballs are useful for software distribution but lack the convenience and automation of modern package management systems.

QUESTION#10

The RPM (Red Hat Package Manager) package format is designed for managing software on Red Hat-based Linux distributions (like Fedora and CentOS). It automates software installation, updates, and management, addressing the shortcomings of tarballs.

Key Features:

1. Automated Installation: Simplifies the installation process by correctly placing and configuring files.
2. Dependency Management: Automatically resolves and installs necessary libraries, preventing installation failures.
3. Version Control: Tracks installed package versions for easy upgrades or rollbacks.
4. Easy Uninstallation: Provides a standard method for cleanly removing software.
5. Integrity Checks: Verifies the authenticity of packages to reduce the risk of corruption.

Advantages Over Tarballs:

- Ease of Use: Automates processes compared to manual tarball installations.
- Dependency Resolution: Handles dependencies automatically, unlike tarballs.
- Package Tracking: Maintains a database of installed software, simplifying version management.
- Clean Removal: Offers straightforward uninstallation, which tarballs typically lack.

RPM provides a more efficient and user-friendly approach to software management than tarballs.