**VIETNAM NATIONAL UNIVERSITY**

**HO CHI MINH UNIVERSITY OF TECHNOLOGY**

****

**REPORT FOR OPERATING SYSTEM LABORATORY (CO2008)**

**CC07 - Semester 242**

|  |  |
| --- | --- |
| **Lecturer** | Le Thanh Van |
| **Name:** | Nguyen Quang Linh |
| **Student ID** | 2352679 |

**Ho Chi Minh City, March 8th 2025**

TABLE OF CONTENTS

[I. ANSWER THE QUESTION LAB 1 2](#_Toc192336634)

[II. EXPLAIN CODE IN PART 3.6 2](#_Toc192336635)

[III. EXPLAIN CODE IN PART 5.6 2](#_Toc192336636)

# 

# ANSWER THE QUESTION LAB 1

**Question 1**: List some other popular Linux shells and describe their highlighted features

|  |  |
| --- | --- |
| **Linux shells** | **Highighted features** |
| Zsh (Z shell) | Enhanced interactice use, Customization and scripting |
| Fish (Friendly Interactive Shell) | User-Friendly Interface, Modern design and Improved usability |
| Ksh (Korn Shell) | Efficiency and compatibility, robust scripting, legacy and stability |
| Tcsh (an enhanced C Shell) | Interactive enhancements, history and aliasing |
| Dash (Debian Almquist Shell) | Speed and Efficiency, POSIX Compliance |

# Question 2: Compare the Output Redirection (>/>>) with the Piping (|) technique.

**Answer:**

* *Overwrite a file (>)*: Redirects the output to a file, replacing its existing contents, e.g. echo “Hi, my name is Tom” > hello\_world.txt
* *Append to a File (>>):* Adds the output to the end of a file without altering its current contents, e.g. echo “I’m 20 years old.” >> hello\_world.txt
* *Piping (|)*: Connects the output of one command directly to the input of another, enabling the creation of complex command sequences, e.g. cat file.txt | grep “search\_term”. It means reads file.txt and filter lines containing “search\_term”

**Question 3:** Compare the sudo and the su command

**Answer:**

They are commands used to execute tasks with elevated privileges, but they differ in functionality and usage.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Command** | **Purpose** | **Usage** | **Authentication** | **Security** | **Configuration** |
| Sudo | Execute a single command with elevated privileges | Sudo command | Current user’s password | Executes specific commands as root, minimizes risk | Configured in /etc/sudoers; allows fine-grained access control |
| su | Switch to another use account (commonly root) for a session | Su or su - | Target user’s password | Provides fill access to the target user’s environment; high risk if misused | No additional configuratuin needed; relies on the target user’s credentials |

**Question 4:** Discuss about the 777 permissions on critical services (web hostings, sensitive databases, …)

**Answer:**

777 grands full read, write and execute permissions to the owner, group and others.

If we use 777 permissions, we will get risks

* Unauthorized access: It allows any user on the system to read, modify or execute files, increasing the risk of unauthorized access or tampering.
* Malware and exploits: If an attacker gains access to the system, they can exploit files with 777 permissions to install malware or escalate privileges
* Data Integrity Issues: Permits any user to alter or delete critical files, potentially leading to data corruption or loss.

So we shouldn’t set permission to 777. We always prioritize security by adhering to best practices and regularly reviewing system configurations.

**Question 5:**

**5.1 What are the advantages of Makefile? Give examples?**

**Answer**: There are many advantages of Makefile:

* *Automated builds*: They automate the compilation and build process, reducing the need for manual intervention. This automation ensures consistency, especially in large projects with numerous source files.
* *Incremental Compilation*: They track dependencies between source files and their corresponding object files. When a source file is modified, only the affected file and its dependent targets are recompiled, optimizing build times.
* *Simplified build management*: They centralize build instructions in a single file, making it easier to manage complex build processes. Developers can define variables, set compiler flags, and specify build targets systematically.
* *Consistency*: By using Makefiles, teams ensure that all developers follow the same build procedures, leading to consistent build results and reducing discrepancies between development environments.

Makefiles enhance effiency and contribute to the overall quality and maintainability of software projects.

**5.2 Compiling a program in the first time usually takes a longer time in comparison to the next re-compiling. What is the reason?**

**Answer:** There are many reasons why it happens:

* Initial compilation of all source code
  + First-time compilation: The compiler processes all source files, translating them into object files and linking them to create the executable. This comprehensive process naturally requires more time.
  + Subsequent compilation: Only modified source files are recompiled, leveraging existing object files, which significantly reduces the overall compilation time.
* Linking process
  + The linking phase combines object files into a single executable. During the initial build, all necessary object files are linked, which can be time-consuming. In contrast, incremental builds focus on linking only the affected components, expediting the process.

**5.3 Is there any Makefile mechanism for other programming languages? If it has, give an example.**

Answer: Yes, they can be utilized with various programming languages beyond their traidition use in C and C++.

Example

* Python:

A black rectangular object with a black border

AI-generated content may be incorrect.

* JavaScript / TypeScript

A black and white screen

AI-generated content may be incorrect.

* Java

A black and white screen with white text

AI-generated content may be incorrect.

# EXPLAIN CODE IN PART 3.6

# EXPLAIN CODE IN PART 5.3