“Київський фаховий коледж зв’язку”

Циклова комісія комп’ютерної та програмної інженерії

**ЗВІТ ПО ВИКОНАННЮ**

**ЛАБОРАТОРНОЇ РОБОТИ №1**

з дисципліни: «Операційні системи»

**Тема: «Ознайомлення з робочим середовищем віртуальних машин та операційних систем різних сімейств»**

Виконав(ла/ли) студент(ка/и)

групи: КСМ-23а

The Awkward Turtles:

Когут Б.М., Михайленко О.О.,

Трощинський Я.П.

Перевірила викладач

Сушанова В.С.

Київ 2024

**Відповіді на контрольні запитання**

***Готував матеріал студент Михайленко О.О.***

Hypervisors (or virtual machine monitors) are divided into two main types: **Type 1 hypervisors** (bare-metal) and **Type 2 hypervisors** (hosted). They differ in architecture, how they operate, and their application areas.

**Type 1 Hypervisors (bare-metal):**

1. **Architecture:**
   * Operate directly on the hardware (server or computer).
   * They are the primary software layer on which virtual machine operating systems run.
2. **Features:**
   * Provide higher performance as they do not require an intermediary operating system.
   * Direct access to hardware ensures more efficient resource utilization.
   * More stable as they reduce the number of intermediaries between hardware and virtual machines.
3. **Examples:**
   * VMware ESXi
   * Microsoft Hyper-V
   * Xen
4. **Application Area:**
   * Used in data centers, for cloud solutions, and corporate environments.
   * Ideal for large-scale virtualization and cloud infrastructures (e.g., private and public clouds).

**Type 2 Hypervisors (hosted):**

1. **Architecture:**
   * Run on top of a host operating system (host OS) that is already installed on the computer or server.
   * Execute as applications that launch virtual machines within the host OS environment.
2. **Features:**
   * Easier to set up and use, as they integrate with the existing operating system.
   * Have lower performance compared to Type 1 hypervisors since they operate through the host OS.
   * Less efficient under heavy loads due to the resource constraints imposed by the host OS.
3. **Examples:**
   * VMware Workstation
   * Oracle VirtualBox
   * Parallels Desktop
4. **Application Area:**
   * Suitable for personal use, software development, testing, and small-scale projects.
   * Used on workstations to run multiple operating systems for testing or educational purposes.

**Key Differences:**

1. **Performance:**
   * Type 1 hypervisors offer better performance due to direct hardware access.
   * Type 2 hypervisors have slightly lower performance since they operate through the host OS.
2. **Resource Utilization:**
   * Type 1 is more efficient in utilizing system resources.
   * Type 2 depends on the host OS, which can limit its efficiency.
3. **Application:**
   * Type 1 is suited for enterprise solutions and servers.
   * Type 2 is suited for personal or less critical use.

2.

The **GNU General Public License (GNU GPL)** is a widely used free software license, which guarantees end users the freedom to run, study, share, and modify the software. It was originally written by Richard Stallman for the GNU Project, and it is the most prominent example of a "copyleft" license, which means that derived works can only be distributed under the same or compatible license terms.

**Key Concepts of GNU GPL:**

1. **Freedom to Use the Software:**
   * Users are free to run the software for any purpose, without any restrictions.
2. **Freedom to Study and Modify:**
   * The license grants users access to the source code, allowing them to study how the program works and modify it according to their needs.
3. **Freedom to Share:**
   * Users are free to redistribute copies of the software, both in its original form and with modifications, ensuring that others also benefit from the software's freedom.
4. **Copyleft Principle:**
   * Any derivative work or modification of GPL-licensed software must also be distributed under the same GPL license. This ensures that future versions of the software remain free and open, protecting users' rights even if the software evolves.
5. **Prohibition of Proprietary Relicensing:**
   * Software distributed under the GPL cannot be converted into proprietary software. If someone modifies and redistributes the software, they must also provide the source code and allow further modification and redistribution under the GPL terms.

**Objective:**

The main goal of the GNU GPL is to promote software freedom by preventing restrictions on how software can be used, shared, or modified. It aims to build a collaborative and open community, where software evolves through contributions while maintaining users' rights.

3. **Open-source software (OSS)** refers to software with source code that anyone can inspect, modify, and enhance. The key aspect of open-source software is its **transparency** and **collaborative nature**, allowing developers and users from around the world to contribute to its development and improvement.

**Core Principles of Open-Source Software:**

1. **Access to Source Code:**
   * The source code is freely available to anyone. This allows users to study how the software works, identify bugs, or suggest improvements.
2. **Freedom to Modify:**
   * Users can modify the software to suit their own needs, customize features, or add new functionalities. This adaptability is a major advantage of open-source software.
3. **Redistribution:**
   * Users are free to redistribute both the original and modified versions of the software. This fosters a community of sharing and collaboration, where improvements are shared back with the public.
4. **Collaborative Development:**
   * Open-source software is often developed in a decentralized and collaborative manner. Developers from different backgrounds and locations can work together to fix bugs, improve security, and add features, making the software more reliable and secure over time.
5. **Licensing:**
   * Open-source software is distributed under licenses that align with open-source principles. Popular licenses include the **GNU General Public License (GPL)**, **MIT License**, and **Apache License**, each ensuring that the software remains open and available for public use and modification.

**Benefits of Open-Source Software:**

* **Transparency:** Users can see exactly how the software works, which enhances trust, especially for security-sensitive applications.
* **Cost-Effectiveness:** Most open-source software is free to use, reducing the cost of licensing and software acquisition.
* **Community Support:** A strong community often forms around open-source projects, providing support, documentation, and improvements.
* **Innovation:** With many contributors, open-source software tends to evolve quickly, adopting new features and technologies.

**Examples of Open-Source Software:**

* **Linux** (operating system)
* **Mozilla Firefox** (web browser)
* **Apache** (web server)
* **LibreOffice** (office suite)

4.A **distribution** (or **distro**) refers to a version of software, typically used in the context of **Linux distributions**. It is a complete operating system package that includes the Linux kernel along with a set of supporting software, libraries, and utilities, all bundled together to form a functional system.

5. There are many system administration tasks that can be implemented using Linux operating systems. Here are some key tasks:

1. **User and Group Management**: Creating, managing, and deleting user accounts and groups. Assigning permissions and controlling access using commands like useradd, groupadd, usermod, and passwd.
2. **File System Management**: Setting up, mounting, and managing file systems (ext4, XFS, etc.). Monitoring disk space usage with tools like df and du, and managing partitions with fdisk or parted.
3. **Network Configuration**: Setting up and managing network interfaces, configuring IP addresses, DNS, routing, and firewall rules using tools like ifconfig, ip, iptables, and firewalld.
4. **Service and Daemon Management**: Starting, stopping, enabling, and monitoring system services and daemons. Linux administrators use systemctl and service to manage services like Apache, Nginx, SSH, and more.
5. **Package Management**: Installing, updating, and removing software packages via package managers such as apt, yum, dnf, or zypper depending on the Linux distribution.
6. **Security Management**: Implementing security measures, configuring firewalls, SELinux, or AppArmor. Setting up access control lists (ACLs), permissions, and ensuring secure authentication methods.
7. **Monitoring and Performance Tuning**: Monitoring system performance, processes, memory, and CPU usage using tools like top, htop, vmstat, and iotop. Tuning the system for better performance by adjusting kernel parameters.
8. **Backup and Recovery**: Implementing backup strategies using tools like rsync, tar, or specialized backup software. Setting up automated backups and ensuring recovery procedures.
9. **Automation with Scripts**: Writing and implementing scripts (Bash, Python, etc.) to automate repetitive tasks, system updates, user creation, backups, and log rotation.
10. **Log Management**: Configuring and analyzing system logs for troubleshooting using journalctl, syslog, or rsyslog to ensure proper log rotation and monitoring.

6. Android and Linux are closely related, with Android being built on top of the Linux kernel. Here's how they are connected:

1. **Linux Kernel Foundation**: Android uses the Linux kernel as its core. The kernel is responsible for managing hardware resources such as CPU, memory, and devices. Android uses a modified version of the Linux kernel to suit the needs of mobile devices, with additional support for mobile-specific hardware.
2. **Custom Layering on Linux Kernel**: While Android uses the Linux kernel, it does not use many of the typical Linux libraries, tools, and GNU components that are common in standard Linux distributions (e.g., Ubuntu or Fedora). Instead, Android provides its own user-space libraries, such as Bionic (its custom C library), and a Java-based application framework.
3. **Different User Interfaces**: Android has a completely different user interface and interaction model compared to traditional Linux systems. It is designed primarily for touchscreens and mobile environments, while most Linux distributions are designed for desktops or servers.
4. **App Ecosystem**: Android applications are typically developed using Java (or Kotlin), and they run within a managed environment (the Android Runtime, or ART), whereas Linux applications are usually developed using C, C++, Python, etc., and run directly on the system.
5. **Security Model**: Android incorporates a robust security model based on the Linux kernel's user separation and permission-based control. Every Android app runs in its own process, with its own user ID, which isolates apps from one another—a feature inherited from Linux’s multi-user design.
6. **Open Source Roots**: Both Linux and Android are open source projects. The Linux kernel is maintained by the Linux community, while Android is developed by Google and released as open source through the Android Open Source Project (AOSP). However, Android includes many proprietary components and services, especially in commercial versions distributed with Google services.

7. **Embedded Linux** refers to a Linux-based operating system designed for embedded systems—small, specialized computing devices that perform specific tasks within larger systems. Here are the key features and usage areas of Embedded Linux:

### ****Key Features of Embedded Linux:****

1. **Lightweight and Customizable**: Embedded Linux is highly customizable, allowing developers to strip down the OS to only the necessary components. This makes it lightweight, which is essential for devices with limited resources like CPU, memory, and storage.
2. **Real-time Support**: Embedded Linux can be configured with real-time capabilities (via Real-Time Linux patches) to meet the strict timing requirements of applications like automotive control systems, industrial automation, and robotics.
3. **Open Source and Cost-effective**: As an open-source system, Linux provides significant cost advantages. There are no licensing fees, and the large Linux community contributes to ongoing development and security updates.
4. **Broad Hardware Support**: Linux supports a wide variety of hardware architectures (ARM, x86, MIPS, PowerPC), making it a versatile choice for different embedded platforms such as microcontrollers and system-on-chips (SoCs).
5. **Security Features**: Linux includes robust security features such as user permissions, process isolation, and support for modern cryptographic libraries, making it suitable for applications where security is crucial (e.g., medical devices, IoT).
6. **Networking Capabilities**: Embedded Linux has built-in support for networking protocols (TCP/IP, UDP, Bluetooth, Wi-Fi), making it ideal for connected devices like routers, IoT gateways, and smart home systems.
7. **File System Flexibility**: Linux supports various file systems (ext4, FAT, JFFS2) that can be optimized for embedded devices depending on their storage needs and performance requirements.
8. **Development Tools and Community Support**: With a rich set of development tools (GCC, GDB, cross-compilers), debugging and development for Embedded Linux are straightforward. The large open-source community also provides extensive documentation and support.

### ****Use Cases of Embedded Linux:****

1. **Consumer Electronics**: Embedded Linux powers many consumer devices such as smart TVs, set-top boxes, digital cameras, and home appliances (e.g., smart refrigerators).
2. **Internet of Things (IoT)**: Due to its flexibility and networking capabilities, Embedded Linux is commonly used in IoT devices like smart thermostats, wearable technology, and home automation systems.
3. **Automotive Systems**: Embedded Linux is used in automotive infotainment systems, navigation systems, and advanced driver assistance systems (ADAS). Automotive-grade Linux (AGL) is a specific Linux distribution tailored for car manufacturers.
4. **Industrial Automation**: Embedded Linux is employed in industrial control systems, robotics, and factory automation due to its real-time capabilities, robustness, and ability to handle complex tasks.
5. **Medical Devices**: Many medical devices, from patient monitoring systems to portable diagnostic tools, use Embedded Linux because of its security, reliability, and flexibility.
6. **Telecommunications**: Embedded Linux is used in routers, switches, and other networking equipment due to its solid networking stack and ability to manage complex communications protocols.
7. **Aerospace and Defense**: Embedded systems in drones, satellites, and military equipment often rely on Embedded Linux for its real-time capabilities, high reliability, and support for a wide range of hardware platforms.

8. To change the boot target of a Linux system between text mode (runlevel 3) and graphical mode (runlevel 5), you can use the following steps:

1. **For a one-time boot:**
   * During boot, on the GRUB menu, press e to edit the boot parameters.
   * Find the line that starts with linux and append 3 at the end to boot into text mode or 5 to boot into graphical mode.
   * Press Ctrl + X or F10 to boot with the modified parameters.
2. **To change the default boot target permanently:**
   * Open a terminal with root privileges.
   * Use the command:

systemctl set-default multi-user.target

for text mode (runlevel 3) or:

systemctl set-default graphical.target

for graphical mode (runlevel 5).

* + Reboot the system for the changes to take effect.

**Difference between CLI and GUI:**

* **CLI (Command Line Interface):**
  + A text-based interface where users interact with the system by typing commands.
  + CLI is more resource-efficient, faster, and often preferred for servers, system administration, and scripting tasks.
  + It requires knowledge of specific commands and syntax.
* **GUI (Graphical User Interface):**
  + A visual-based interface where users interact with the system through graphical elements like windows, icons, and menus.
  + GUI is more user-friendly, especially for regular users or those new to the system.
  + It is more resource-intensive compared to CLI and is commonly used on desktop environments.