**兰州理工大学**

|  |  |
| --- | --- |
| D:\小米备份\Xiaomi\Pictures\PS\兰州理工大学.png兰州理工大学 |  |
| Lanzhou University of Technology |

**《GNU/Linux操作系统》实验报告**

期末实验: **Final experimental report**

**院(系):计算机与通信学院**

**专业年级:**

**姓 名:**

**学 号:**

**指导教师:**

**20年月日**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **《GNU/Linux操作系统》实验报告** | | | | | |
| 题目 | Final experimental report | 姓名 |  | 学号 |  |
| Ⅰ.Purpose of the experiment:  1.熟悉GNU/Linux操作系统：通过GNU/Linux基本操作实验，我们可以熟悉GNU/Linux操作系统的基本界面、命令行操作和文件系统结构，了解GNU/Linux的特点和优势。  （Familiarity with the GNU/Linux operating system: Through the GNU/Linux Basic Operating Experiment, We are familiar with the basic interface, command-line operations, and file system structure of the GNU/Linux operating system, Learn about the features and benefits of GNU/Linux.）  2.培养操作和管理技能：通过GNU/Linux基本操作实验可以帮助我们掌握GNU/Linux的基本命令，如文件和目录管理、权限管理、进程管理、软件安装和配置等技能，从而能够在GNU/Linux环境下进行常见的操作和管理任务。  （Develop operational and management skills: The GNU/Linux Basic Operations Experiment can help us master the basic GNU/Linux commands, such as file and directory management, rights management, process management, software installation and configuration, to enable common operational and administrative tasks in the GNU/Linux environment.）  3.理解GNU/Linux安全性：通过Linux基本操作实验可以帮助我们了解GNU/Linux的安全性特点，如用户和权限管理，以及如何保护系统免受潜在威胁的影响。  （Understanding GNU / Linux Security: Basic Linux operating experiments can help us understand GNU / Linux security features, such as user and rights management, and how to protect the system from potential threats.） | | | | | |
| II. Experimental Environment：  Software Environment: Raspberry Pi OS (64-bit)    Hardware Environment: Raspberry Pi 3 Model B+ | | | | | |
| III. Experimental Content：  However, the following experiments are all conducted on the command line, so here I use SSH tools (such as Xshell) instead of GUI operations.  i. General  1. Run `top` in one terminal while you do the exercises in another.  Command: `top`  Response: Execute this command to view real-time information about processes, system load, and resource usage.  2. Run the `ps` command.  Command: `ps`  Response: Use this command to display a snapshot of currently running processes.  3. Read the man pages to find out how to display all your processes.  Command: `man ps`  Response: Review the manual pages for the `ps` command to learn how to display all processes.  4. Run the command `find /`. What effect does it have on system load? Stop this command.  Command: `find /`  Response: Execute this command to search for files starting from the root directory. Observe its impact on system load and stop the command.  5. In graphical mode, start the `xclock` program in the foreground. Then let it run in the background. Stop the program using the `kill` command.  Command: `xclock` (in one terminal), `xclock &` (in another terminal), `kill <xclock\_process\_id>`  Response: Start the `xclock` program in the foreground and background, then stop it using the `kill` command.  6. Run the `xcalc` directly in the background, so that the prompt of the issuing terminal is released.  Command: `xcalc &`  Response: Run `xcalc` in the background, freeing the terminal prompt for other commands.  7. What does `kill -9 -1` do?  Command: `kill -9 -1`  Response: Execute this command to forcefully terminate all processes except for the initiating process and the system processes.  8. Open two terminals or terminal windows again and use `write` to send a message from one to the other.  Command: `write <username>`  Response: Use `write` to send a message from one terminal to another.  9. Issue the `dmesg` command. What does it tell?  Command: `dmesg`  Response: Run this command to display kernel-related messages and information about system hardware and devices.  10. How long does it take to execute `ls` in the current directory?  Command: `time ls`  Response: Measure the execution time of the `ls` command in the current directory using this command.  11. Based on process entries in `/proc`, owned by your UID, how would you work to find out which processes these actually represent?  Response: Explore the `/proc` directory entries corresponding to your UID to identify and understand the associated processes.  12. How long has your system been running?  Command: `uptime`  Response: Check the system uptime to determine how long your system has been running.  13. Which is your current TTY?  Command: `tty`  Response: Use this command to find out the current terminal (TTY) you are using.  14. Name 3 processes that couldn't have had `init` as an initial parent.  Command: `ps -eo pid,ppid,comm | grep -v init`  Response: Identify three processes that did not have `init` as their initial parent using this command.  15. Name 3 commands which use SUID mode. Explain why this is so.  Command: `find / -type f -perm -4000`  Response: List three commands using SUID mode and understand why SUID is necessary for their functionality.  16. Name the commands that are generally causing the highest load on your system.  Command: `ps aux --sort=-%cpu | head`  Response: Identify commands causing the highest CPU load using this command.  ii. Booting, init etc.  1. Can you reboot the system as a normal user? Why is that?  Response: No, a normal user cannot reboot the system because rebooting requires administrative privileges for system-wide changes.  2. According to your current run level, name the steps that are taken during shutdown. How do you change the system run level? Switch from your default run level to run level 1 and vice versa.  Response: Describe the shutdown steps based on the current run level. Change the run level using `telinit` or `init` commands.  3. Make a list of all the services and daemons that are started up when your system has booted.  Command: `systemctl list-units --type=service --state=running`  Response: Generate a list of running services and daemons using this command.  4. Which kernel is currently loaded at startup?  Command: `uname -r`  Response: Identify the currently loaded kernel version using this command.  5. Suppose you have to start some exotic server at boot time. What do you have to do in order to have the service start up automatically in run level 4, which you defined for this purpose only?  Response: Create a service script and configure it in the appropriate run level directory, such as `/etc/rc4.d/`, to ensure automatic startup.  iii. Scheduling  1. Use `sleep` to create a reminder that your pasta is ready in ten minutes.  Command: `sleep 600 && echo "Pasta is ready!"`  Response: Set a reminder for pasta readiness using the `sleep` command.  2. Create an `at` job that copies all files in your home directory to /var/tmp within half an hour. You may want to create a sub-directory in  /var/tmp.  Command: `at now + 30 minutes; cp -r ~/ /var/tmp/backup`  Response: Schedule a task to copy home directory files to /var/tmp/backup in half an hour using the `at` command.  3. Make a `cronjob` that does this task every Monday to Friday during lunch. Check that it works.  Command: `crontab -e` (add `0 12 1-5 cp -r ~/ /var/tmp/backup`)  Response: Create a cron job to copy files every weekday at lunchtime and verify its functionality.  4. Make a mistake in the crontab entry, like issuing the nonexistent command `coppy` instead of `cp`. What happens upon execution of the task?  Response: The cron job will fail because the command `coppy` does not exist, and an error message will be generated.  iv. I/O Redirection and Command Composition Exercises  1. Use the `cut` command on the output of a long directory listing in order to display only the file permissions. Then pipe this output to `sort` and `uniq` to filter out any double lines. Then use `wc` to count the different permission types in this directory.  Commands: `ls -l | cut -c 1-10 | sort | uniq | wc -l`  Response: Extract file permissions from a directory listing, sort and filter them, then count the different permission types.  2. Put the output of `date` in a file. Append the output of `ls` to this file. Send this file to your local mailbox. When using Bash, you will see a new mail notice upon success.  Commands: `date > my\_file.txt; ls >> my\_file.txt; mail -s "Subject" your\_username < my\_file.txt`  Response: Save the current date, append the directory listing to a file, and send it to the local mailbox, triggering a new mail notice.  3. List the devices in `/dev` which are currently used by your UID. Pipe through `less` to view them properly.  Commands: `ls -l /dev | grep $(whoami) | less`  Response: List devices in `/dev` used by your user and view them properly using `less`.  4. Issue the following commands as a non-privileged user. Determine standard input, output, and error for each command.  Commands:  - `cat nonexistentfile`  - `file /sbin/ifconfig`  - `grep root /etc/passwd /etc/nofiles > grepresults`  - `/etc/init.d/sshd start > /var/tmp/output`  - `/etc/init.d/crond start > /var/tmp/output 2>&1`    Response: Execute commands and determine standard input, output, and error for each.  5. Now check your results by issuing the commands again, now redirecting standard output to the file `/var/tmp/output` and standard error to the file `/var/tmp/error`.  Commands:  - `/etc/init.d/sshd start > /var/tmp/output 2> /var/tmp/error`  - `/etc/init.d/crond start > /var/tmp/output 2> /var/tmp/error`  Response: Reissue commands, redirecting standard output to one file and standard error to another.  6. How many processes are you currently running?  Command: `ps aux | wc -l`  Response: Count the number of processes currently running.  7. How many invisible files are in your home directory?  Command: `ls -a | grep "^\." | wc -l`  Response: Count the number of invisible files (starting with a dot) in the home directory.  8. Use `locate` to find documentation about the kernel.  Command: `locate kernel | grep doc`  Response: Locate and display documentation related to the kernel.  9. Find out which file contains the following entry: `root:x:0:0:root:/root:/bin/bash` And this one: `system: root`  Command: `grep -r "root:x:0:0:root:/root:/bin/bash" /etc`  Response: Identify the file containing the specified entry.  10. See what happens upon issuing this command: `> time; date >> time; cat < time`  Command: `> time; date >> time; cat < time`  Response: Execute the command and observe the result.  11. What command would you use to check which script in `/etc/init.d` starts a given process?  Command: `ls -l /etc/init.d | grep "process\_name"`  Response: List scripts in `/etc/init.d` and identify the one associated with the given process name. | | | | | |
| IV. Experiment summary and experience  Exploring Linux Commands  In the exploration of Linux commands and system functionalities, the exercises in sections 4.6 and 5.5 provided valuable hands-on experience for a computer science and technology student like myself. Here is a summary of the key learnings and experiences from these exercises:   * Section i-iii: Processes * General Insights:   Process Monitoring: Utilizing commands like top and ps allowed me to gain real-time insights into running processes on the system.  Kernel Information: Investigating kernel information through commands such as uname -r enhanced my understanding of the system's foundational components.   * System Interaction:   Run Levels: Experimenting with system run levels and understanding the steps during shutdown provided practical knowledge about system initialization and termination.  Service Configuration: Configuring a service script to start automatically during boot in a specific run level was a valuable lesson in system customization.   * Scheduled Tasks:   Scheduling Jobs: Exploring the use of at and cron for scheduling tasks showcased the power of automation in managing recurring processes.  Error Handling: Creating intentional errors in cron entries and observing the outcomes increased my awareness of error handling in scheduled tasks.   * Section iv: I/O Redirection and Command Composition * Command Mastery:   I/O Redirection: The exercises in this section honed my skills in redirecting input and output streams, enhancing my proficiency in managing data flow.  Command Composition: Combining commands such as cut, sort, and uniq showcased the efficiency of chaining operations for complex tasks.   * File Operations:   File Manipulation: Appending outputs to files, sending files via email, and redirecting standard output and error expanded my understanding of file operations in a Linux environment.  Device Exploration: Listing devices in /dev and understanding their usage by my user ID deepened my knowledge of device management.   * Process Analysis:   Standard Streams: Analyzing standard input, output, and error for various commands, especially during intentional errors, improved my troubleshooting skills.  Process Identification: Learning to identify processes associated with specific functionalities using commands like ps and grep was enlightening.   * Overall Reflection:   These exercises provided a holistic experience in navigating Linux systems, from understanding core processes to mastering advanced command-line techniques. The emphasis on practical application, error analysis, and system exploration has significantly contributed to my proficiency in Linux system administration. These skills are not only crucial for academic pursuits but also lay a strong foundation for real-world scenarios in the field of computer science and technology. | | | | | |