Laboratory manual

Systems Administration

UbiLAB

Универзитет "Св. Кирил и Методиј" во Скопје
ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И ИНФОРМАЦИСКИ ТЕХНОЛОГИИ

Hochschule Anhalt
Anhalt University of Applied Sciences

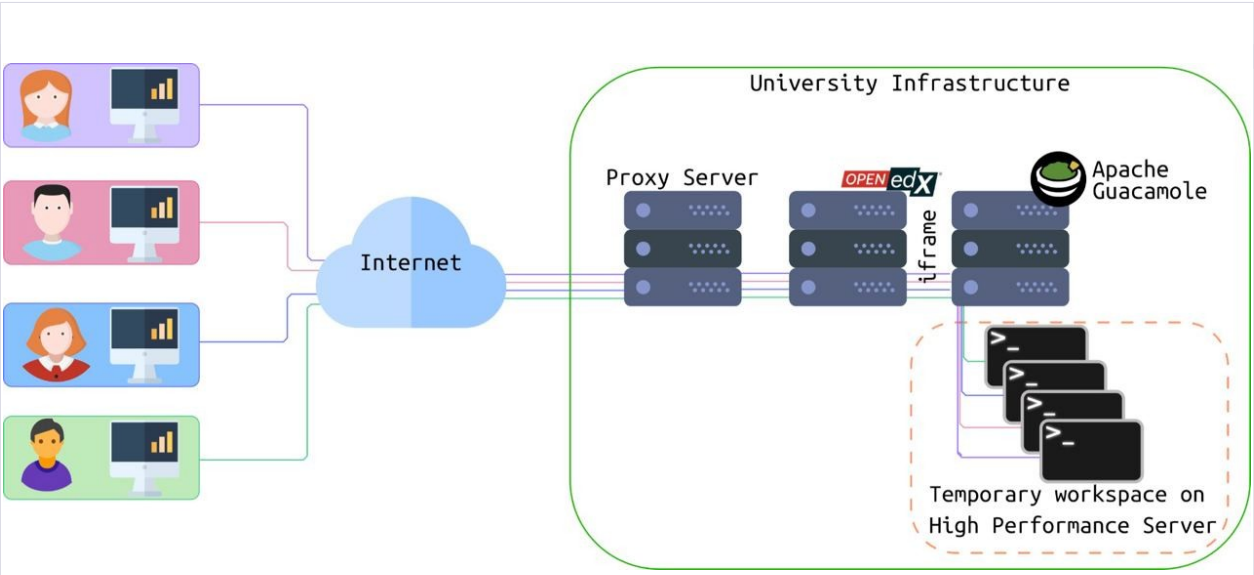
University of Maribor

Erasmus+

NATIONAL AGENCY
for European Educational
Programmes and Mobility

Content

Introduction



The setup enabling remote access to the laboratory experiments involves several stages of development and integration of various technologies. Here's a detailed description of each component:

1. **Apache Proxy Server**: The first line of communication with the outside world, the Apache proxy server, is the backbone of our setup. It's strategically placed between the internet and our internal network, serving as the border point. This configuration not only maintains our internal network's security but also manages and controls the flow of traffic. Any requests from users first hit this server, where they are verified and then routed to the appropriate service in our internal network.
2. **OpenEdx and Guacamole Services**: Both these services are deployed within the secure boundaries of our internal network and are responsible for the core functionality of our laboratory setup. OpenEdx is a feature-rich platform for online learning, serving course material, quizzes, and providing a user-friendly interface for learners to interact with the course content. Guacamole is used to facilitate the connection to our high-performance server. It provides a Virtual Network Computing (VNC) service that allows users to access the lab environment directly from their web browsers.

3. **iFrame Integration**: To provide a seamless user experience, Guacamole is integrated into OpenEdx using an iFrame. This allows the laboratory exercises to be loaded directly within the OpenEdx platform, offering a unified interface to the students. The students can access the lab exercises and OpenEdx course materials simultaneously, leading to a more interactive and efficient learning experience.
4. **High-Performance Server and Docker Containers**: The laboratory environment is hosted on a high-performance Linux server. The server uses Docker, a platform-as-a-service product that delivers software in packages known as containers. These containers are temporary instances that contain all the necessary elements to run the desired application, in this case, the laboratory exercises. Each student gets their Docker container, providing an isolated environment for their work. This ensures that students' experiments do not interfere with each other and can be scaled as per the needs.
5. **Apache Guacamole VNC**: Apache Guacamole's VNC is used to allow users to interact with their Docker containers. It provides a remote desktop interface, directly accessible from a web browser, eliminating the need for users to install any software or navigate complex network configurations.
6. **Automatic Exercise Checking Script**: The grading of exercises is automated through a script. Upon completion of an exercise, this script packages the student's work into a tar file and computes its MD5 checksum for verification purposes. The tar file is then removed, ensuring that only the checksum remains for assessment. This setup allows for efficient, scalable, and consistent grading of student work.
7. **Integration with Moodle and OpenEdx Platforms**: The entire laboratory setup is designed to connect seamlessly with Moodle and OpenEdx learning management systems. These integrations allow us to deliver laboratory exercises and feedback directly through these platforms, making it easier for students to access, complete, and receive feedback on their work. This setup provides a comprehensive learning experience to the students, right from accessing the course material to getting their exercises evaluated.

This setup, with a combination of Apache Proxy Server, OpenEdx, Guacamole, Docker, and integration with popular learning management systems, ensures a scalable, accessible, and interactive environment for students to carry out their laboratory experiments remotely. The focus on security, usability, and seamless integration provides a superior learning experience for students.

How to gain remote access

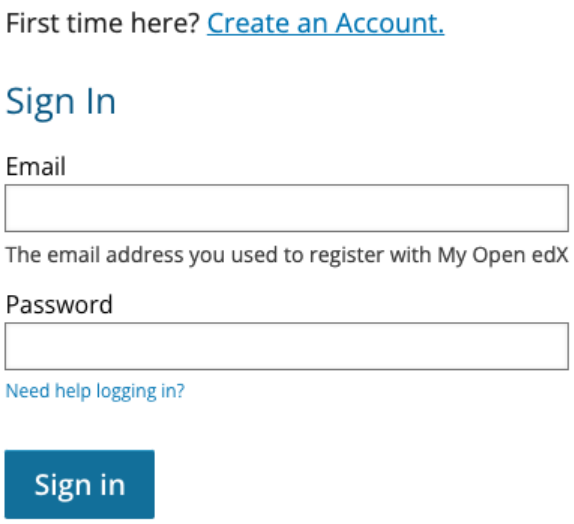


Figure 1: OpenEdx Login Screen: Enter your credentials to begin your learning journey.

1. Login: First, navigate to the OpenEdx platform using the domain name openedx.fila-lab.de in your web browser. You will be greeted by a login screen. Enter your credentials to proceed. If you don't have an account yet, you'll need to create one.

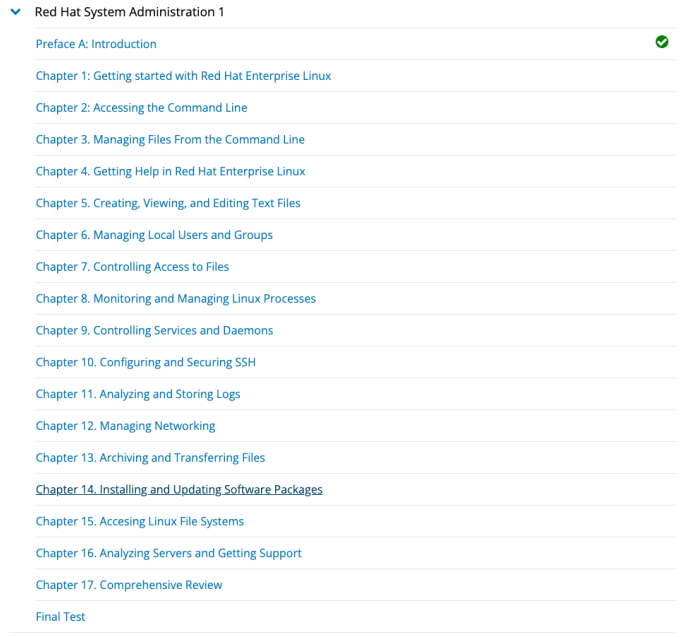


Figure 2: Course Dashboard: A wide range of courses are at your disposal. Select one to

2. Course Selection: After logging in, you will be directed to the main dashboard, where you can see all the available courses. Browse through the list and select the course you want to participate in.

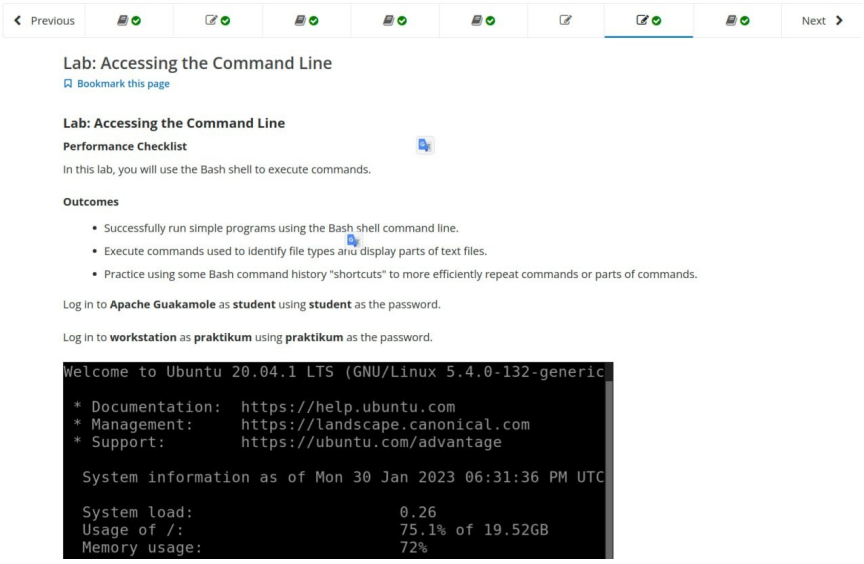


Figure 3: Interactive Laboratory Exercise: Perform your exercise in the embedded Linux environment through the Apache Guacamole VNC interface.

3. Sections and Subsections: After selecting a course, you will be presented with the course outline. This includes various sections and subsections, each representing different topics and subtopics within the course. Navigate through these sections and subsections to find the laboratory exercises.

4. Accessing Laboratory Exercises: The laboratory exercises are embedded within the course using an iframe. This iframe contains the Apache Guacamole VNC interface, which provides remote access to a Linux environment where the exercises are to be performed. Click on the specific laboratory exercise, and it will be loaded within the iframe.

Please remember to save your work regularly while performing the exercises to avoid data loss. Once you have completed your exercise, it will be automatically graded and the feedback will be available on the OpenEdx platform. Enjoy your learning experience!

Laboratory experiment 1: Accessing the Command Line

This laboratory experiment introduces students to the Bash shell, a popular command-line interpreter for Unix-based systems. In the scope of this lab, students will execute a variety of commands, manipulate text files, and experience the efficiency of Bash command history shortcuts.

Experimental setup

The experiment is conducted on a Linux environment, accessed remotely via Apache Guacamole, integrated into the OpenEdx platform. Students log in to the environment as 'praktikum' with the same as the password, allowing them to execute commands in the Bash shell.

Goals of the experiment

The objectives of this experiment are to:

1. Familiarize students with the Bash shell command line.
2. Execute commands to identify file types and display parts of text files.
3. Learn and practice using Bash command history shortcuts for command repetition and modification.

Experimental results

Upon completion of this experiment, students will gain:

1. A clear understanding of how to navigate the Bash shell command line.

2. Experience with executing commands to manipulate and inspect files.
3. Proficiency in using Bash command history shortcuts to repeat and modify previous commands.
4. After each step, students can input their command results into the lab's interactive grader for immediate feedback and validation.

Task:

Step 1: Login

Firstly, log in to Apache Guacamole using the credentials: username - 'student', password - 'student'. Then, log in to the workstation with the credentials: username - 'praktikum', password - 'praktikum'.

Step 2: Display Current Date and Time

Execute the following command in the Bash shell to display the current time and date:

```
date
```

Step 3: Display Current Time in 12-Hour Clock Format

To display the current time in a 12-hour clock format, use the following command:

```
date +%r
```

Step 4: Identify File Type

Determine the file type of /home/praktikum/zcat and whether it is readable by humans using:

```
file /home/praktikum/zcat
```

Step 5: Display the Size of the File

Use the wc command along with Bash shortcuts to display the size of zcat:

```
wc -c /home/praktikum/zcat
```

Step 6: Display the First 10 Lines of the File

To display the first 10 lines of zcat, execute:

```
head /home/praktikum/zcat
```

Step 7: Display the Last 10 Lines of the File

Display the last 10 lines of the zcat file using the following command:

```
tail /home/praktikum/zcat
```

Step 8: Repeat the Previous Command

To repeat the previous command exactly with three or fewer keystrokes, use Bash history command:

```
!!
```

Step 9: Display the Last 20 Lines of the File

Edit the previous command to display the last 20 lines of the file with a minimal number of keystrokes:

```
!-1 -n 20
```

Step 10: Run the date +%r Command Again

Use the shell history to run the date +%r command again:

```
!date
```

Please remember, after each step, to input your command results into the fields provided for verification. This will help you ensure the correctness of your work.

Remember to follow each step carefully and good luck with your experiment!

Laboratory experiment 2: Managing Files from the Command Line

In this lab, you will efficiently create, move, and remove files and directories by using the shell and a variety of file name matching techniques.

Experimental setup

1. Log into Apache Guacamole as student using 'student' as the password.
2. Log into the workstation as 'praktikum' using 'praktikum' as the password.

Goals of the experiment

The aim of this experiment is to:

1. Familiarize you with file and directory manipulation from the command line.
2. Teach you to use wildcards for locating and handling files.

Task

1. Before you create project files, use the mkdir command with brace expansion to create empty project planning documents in the /tmp/answer/Documents/project_plans directory. (Hint: if /tmp/answer/Documents does not exist, the -p option for the mkdir command will create it.) Create two empty files in the /tmp/answer/Documents/project_plans directory: season1_project_plan.odf and season2_project_plan.odf .
2. Create sets of empty practice files to use in this lab. If you do not immediately recognize the intended shell expansion shortcut, use the solution to learn and practice. Use shell tab completion to locate file path names easily. Create a total of 12 files with names tv_seasonX_episodeY.ogg . Replace X with the season number and Y with that season's episode, for two seasons of six episodes each.

3. As the author of a successful series of mystery novels, your next bestseller's chapters are being edited for publishing. Create a total of eight files with names mystery_chapterX. Replace X with the numbers 1 through 8.
4. Use a single command to create two subdirectories named season1 and season2 under the Videos directory, to organize the TV episodes.
5. To check the task [student@workstation ~]$ bash /tmp/check.sh /tmp/answer/
6. Paste the output of the script into the text box
7. Move the appropriate TV episodes into the season subdirectories. Use only two commands, specifying destinations using relative syntax.
8. Create a 2-level directory hierarchy with a single command to organize the mystery book chapters. Create my_bestseller under the Documents directory, and chapters under the new my_bestseller directory.
9. Create three more subdirectories directly under the my_bestseller directory using a single command. Name these subdirectories editor , changes , and vacation . The -p option (create parents) is not needed because the my_bestseller parent directory already exists.
10. Change to the chapters directory. Move all book chapters to the chapters directory, which is now your current directory. What is the simplest syntax to specify the destination directory?
11. To check the task: bash /tmp/check.sh /tmp/answer/
12. Paste the output of the script into the text box
13. You sent the first two chapters to the editor for review. Move only those two chapters to the editor directory to avoid modifying them during the review. Starting from the chapters subdirectory, use brace expansion with a range to specify the chapter file names to move and a relative path for the destination directory.
14. While on vacation you intend to write chapters 7 and 8. Use a single command to move the files from the chapters directory to the vacation directory. Specify the chapter file names using brace expansion with a list of strings and without using wildcard characters.
15. Change your working directory to /tmp/answer/Videos/season2 , and then copy the first episode of the season to the vacation directory.
16. Use a single cd command to change from your working directory to the /tmp/answer/Documents/my_bestseller/vacation directory. List its files. Use the previous

working directory argument to return to the season2 directory. (This will succeed if the last directory change with the cd command was accomplished with one command rather than several cd commands.) From the season2 directory, copy the episode 2 file into the vacation directory. Use the shortcut again to return to the vacation directory

17. To check the task: bash /tmp/check.sh /tmp/answer/
18. Paste the output of the script into the text box
19. The authors of chapters 5 and 6 want to experiment with possible changes. Copy both files from the /tmp/answer/Documents/my_bestseller/chapters directory to the /tmp/answer/Documents/my_bestseller/changes directory to prevent these changes from modifying original files. Navigate to the /tmp/answer/Documents/my_bestseller directory. Use square-bracket pattern matching to specify which chapter numbers to match in the filename argument of the cp command.
20. Change your current directory to the changes directory. Use the date +%F command with command substitution to copy mystery_chapter5 to a new file which includes the full date. The name should have the form mystery_chapter5_YYYY-MM-DD. Make another copy of mystery_chapter5 , appending the current time stamp (as the number of seconds since the epoch, 1970-01-01 00:00 UTC) to ensure a unique file name. Use command substitution with the date +%s command to accomplish this.
21. After further review, you decide that the plot changes are not necessary. Delete the changes directory. If necessary, navigate to the changes directory and delete all the files within the directory. You cannot delete a directory while it is the current working directory. Change to the parent directory of the changes directory. Try to delete the empty directory using the rm command without the -r recursive option. This attempt should fail. Finally, use the rmdir command to delete the empty directory, which will succeed.
22. When the vacation is over, the vacation directory is no longer needed. Delete it using the rm command with the recursive option. Return to /tmp/answer directory.
23. Create a hard link to the /tmp/answer/Documents/project_plans/season2_project_plan.odf file named /tmp/answer/Documents/backups/season2_project_plan.odf.back . A hard link will protect against accidental deletion of the original file and will keep the backup file updated as changes are made to the original.
24. On workstation, run the check.sh script to confirm success on this lab. $ cd /tmp/ ; bash check.sh answer/

25. Paste the output of the script into the text box

Experimental results

You should be able to use wildcards to locate and manipulate files.

Laboratory experiment 3: Editing Files Using Vim's Visual Mode

In this lab, you'll learn how to simplify repetitive edits using Vim's visual mode. You'll manipulate a file containing a list of selected files and tabular data, which will help you become more familiar with the required utilities and techniques for file editing.

Experimental setup

Log into the server as 'student' and begin in the student's home directory.

Goals of the experiment

he objective of this experiment is to enhance your skills in:

1. Redirecting output to a file or program.
2. Editing text files from the shell prompt using Vim.
3. Editing text files with a graphical editor.

Experimental results

At the completion of this lab experiment, you should have achieved the following outcomes:

1. File Redirection: You should have successfully redirected the long listing of all content in the student's home directory into a file named editing_final_lab.txt. This redirection helps in logging the directory structure for future reference or for tracking changes over time.
2. Vim Editing: You should be able to demonstrate your proficiency in using Vim's visual mode. This was achieved by editing editing_final_lab.txt and manipulating lines and columns within the file. Your skills in using the line-based, single-line, and block selection modes in Vim should have been honed.

3. File Manipulation: By removing specific lines and columns from the file, you have demonstrated an understanding of how to selectively manipulate file content. This includes modifying file permissions and removing certain rows.
4. File Backup and Emailing: You created a backup of your file using a timestamp to ensure a unique filename. You then emailed the contents of this file to 'student'. This showcases your ability to safeguard your work and to share your work via email.
5. File and Process Appending: You added a dashed line to the file to denote the beginning of new content. Subsequently, you appended a process listing to the file and validated its presence at the end of the file. This demonstrates your ability to dynamically add content to a file, including system process information.

Overall, these outcomes signify that you've acquired valuable skills in redirecting output to a file, editing text files from the shell prompt using Vim, and using an editor in a graphical desktop environment to modify file content.

Task:

Log in to workstation as student using student as the password.

On workstation, run the edit-review start command.

[student@workstation ~]$ lab edit-review start

1. Redirect a long listing of all content in the student's home directory, including hidden directories and files, into a file named editing_final_lab.txt .
2. Edit the file using Vim.
3. Remove the first three lines. Enter line-based visual mode with uppercase V.
4. Remove columns on the first line. Enter visual mode with lowercase v. Lowercase v selects characters on a single line only. The columns after -rw- should be deleted.
5. Remove columns, and the subsequent dot (".") on the remaining lines. Use the visual block mode. Enter visual block with the control sequence Ctrl+V. Use this key sequence to select a block of characters on multiple lines. The columns after -rw-should be deleted.

6. Use visual block mode to remove the fourth column.
7. Use visual block mode to remove the time column, leaving the month and day on all lines.
8. Remove the Desktop and Public rows. Enter visual line mode with uppercase V.
9. Use the :wq command to save and exit the file. Make a backup, using the date (in seconds) to create a unique file name.
10. Append a dashed line to the file. The dashed line should contain at least 12 dashes
11. Append a directory listing of the Documents directory. List the directory listing on the terminal and send it to the editing_final_lab.txt file with one command line.
12. Confirm that the directory listing is at the bottom of the lab file.

Conclusion

In conclusion, this lab exercise provided a hands-on experience with fundamental operations in Unix/Linux command line environments, including redirecting output to a file, editing text files using Vim, and utilizing a graphical editor for altering file content.

The practice of using Vim's visual modes allowed for enhanced understanding and skill development in manipulating file data effectively. The selective removal of lines and columns within a file served as an example of how file content could be manipulated for specific needs, such as refining file permissions or modifying specific rows.

Moreover, the experiment emphasized the importance of good file management practices, particularly in creating backups and ensuring files are shared securely via email.

Appending a process listing to a file demonstrated how system process information could be logged and accessed, underlining the versatility of Unix/Linux commands in gathering system insights.

Ultimately, the lab has equipped the participants with essential command line skills that will be beneficial in a variety of scenarios, particularly in system administration, software development, and data management. These skills are not only crucial in dealing with Unix/Linux environments but also are transferrable to other computing contexts, which makes them valuable in today's increasingly digital and data-driven world.