™

Andrew Shahini, Kishaan Thalayasingam, and Melanie Swain

Abstract

Management System done in Linux to help the administrators to obtain information and manage the system efficiently.

FINAL UNIX PROJECT

Table of Contents

[Project Overview 2](#_Toc184658194)

[Project Requirements and Deliverables 2](#_Toc184658195)

[Task Assignment 2](#_Toc184658196)

[Project Components and Solutions 3](#_Toc184658197)

[System Status: 3](#_Toc184658198)

[Backup: 4](#_Toc184658199)

[Network: 5](#_Toc184658200)

[Services 6](#_Toc184658201)

[User Management: 7](#_Toc184658202)

[File Management: 8](#_Toc184658203)

[Detailed Section for Each Part 9](#_Toc184658204)

[System Status: 9](#_Toc184658205)

[Backup: 9](#_Toc184658206)

[Network: 10](#_Toc184658207)

[Services: 10](#_Toc184658208)

[User Management: 10](#_Toc184658209)

[File Management: 12](#_Toc184658210)

[GitHub 13](#_Toc184658211)

# Project Overview

The UNIX Management System Shell Script project is designed to create a user-friendly tool for system administrators. This script simplifies essential system management tasks by showcasing an interface. It enables administrators to gather system information, manage resources such as backups, configure networks, handle user management, and perform file management tasks effectively. The project emphasizes functionality, user-friendliness, and proper documentation as it’s the main criteria in this project.

# Project Requirements and Deliverables

|  |  |
| --- | --- |
| **DELIVERIES** | **DELIVERY DATES** |
| Delivery 1: Presentation to demonstrate what has been completed so far. | November 27th, 2024 |
| Delivery 2: Final presentation. | December 4th, 2024 |
| Delivery 3: Final submission of the script and documentation. | December 9th, 2024 |

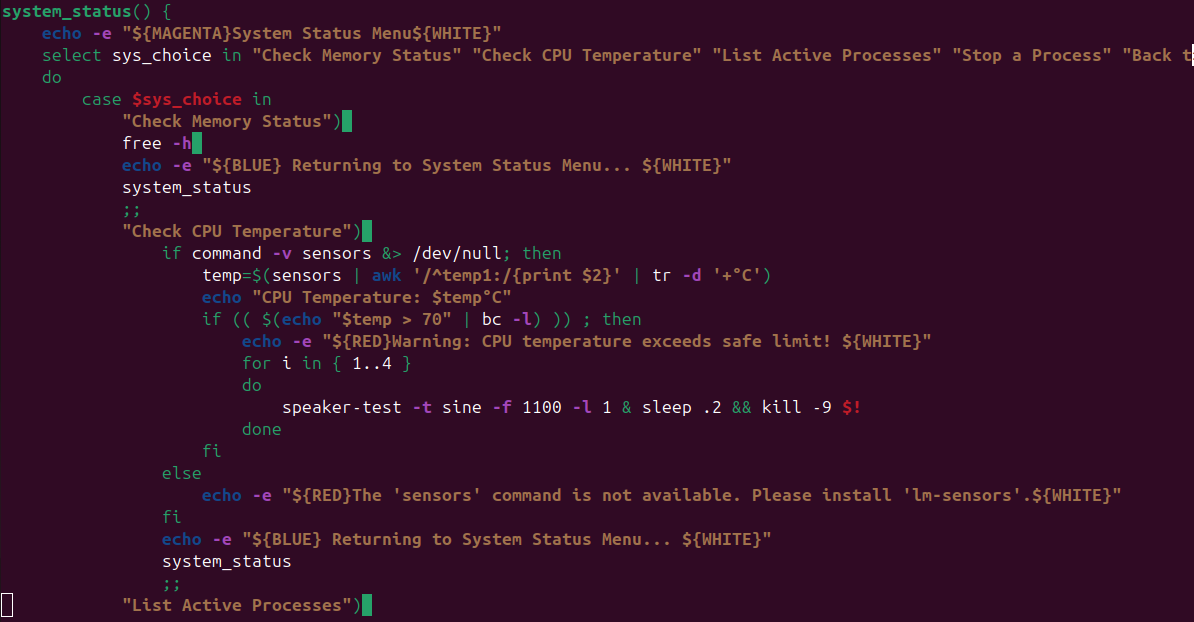
# Task Assignment

A breakdown of how the tasks were assigned to everyone.

|  |  |
| --- | --- |
| **TEAM MEMBERS** | **TASKS** |
| Andrew Shahini | * Implemented the main menu. * Developed the system status. * Implemented the services section. * Designed the documentation layout. |
| Kishaan Thalaysingam | * Added colours to the menus * Developed the networking function. * Implemented the back up function. |
| Melanie Swain | * Implemented the file management function. * Implemented the user management function. |

# Project Components and Solutions

## System Status:

A screenshot of a computer program

Description automatically generated

## Backup:

A computer screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

## Network:

A computer screen shot of a program code

Description automatically generated

A computer screen shot of text

Description automatically generated

A screenshot of a computer program

Description automatically generated

Services:A computer screen shot of a program code

Description automatically generated

## User Management:

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

A black background with yellow text

Description automatically generated

A screen shot of a computer program

Description automatically generated

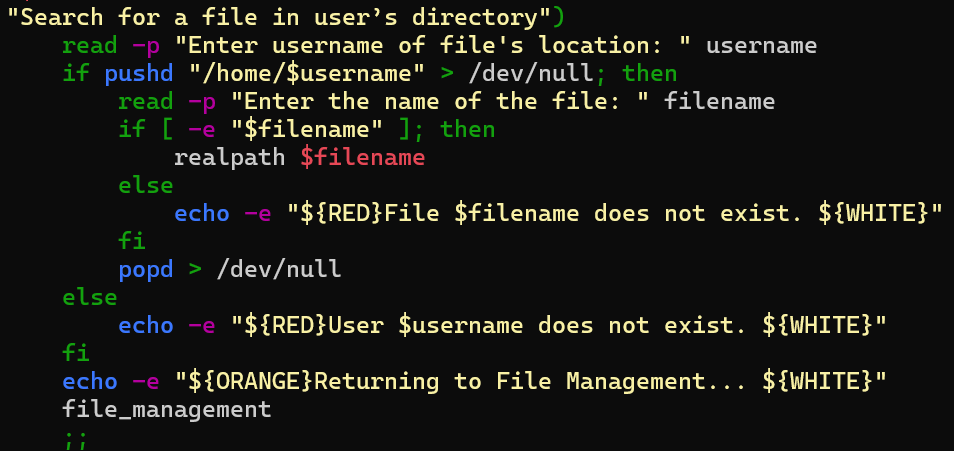
A screenshot of a computer code

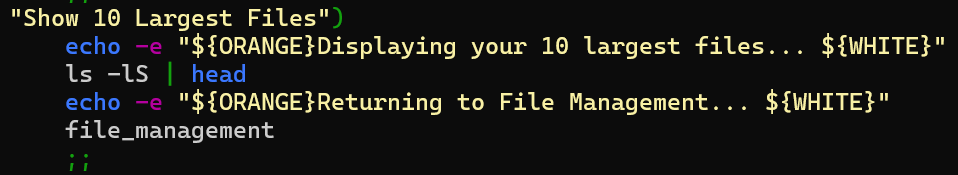
Description automatically generated

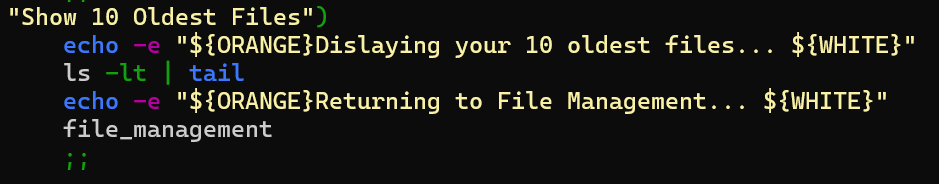
A screen shot of a computer program

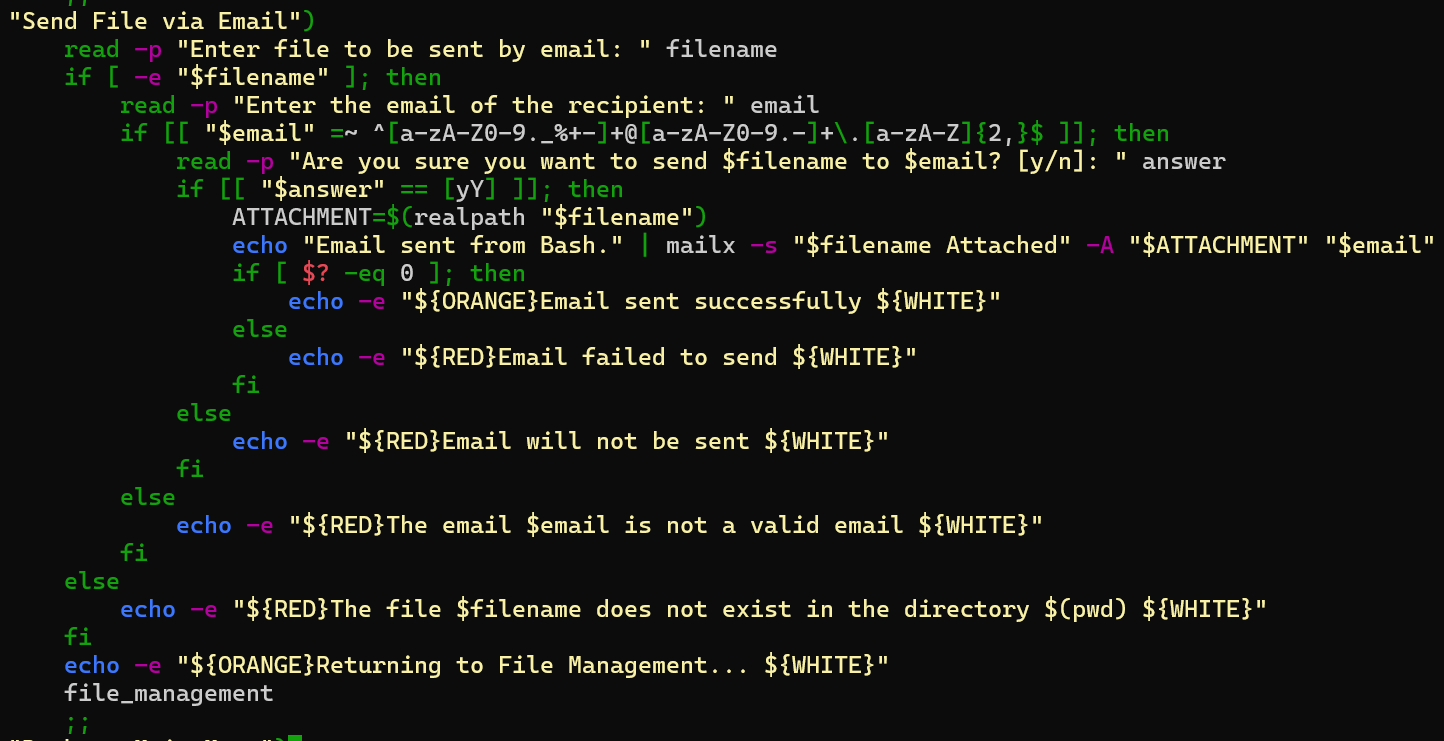
Description automatically generated

## File Management:









# Detailed Section for Each Part

## System Status:

* Select and case was used for the structure of the menu layout for the system status.
* **Check Memory Status**
* Free -h: this command displays the system’s memory usage in a format that is meant for humans.
* **Check Temperature**
* Sensors: provides data about the hardware, we used it to provide the CPU’s temperature.
* Awk: a text tool for formatting an output, we used it to format the text of the CPU information to just show the temperature and the list of the PIDs.
* Tr -d: to delete characters from the awk output.
* Speaker-test: Plays test tones through the system’s speaker to allude a warning for the user’s CPU temperature surpasses 70°C three times by wrapping in a for-loop.
* **List active processes**
* Ps aux: displays all the running processes.
* **Stop a process**
* Kill: This command is used to stop a specific process by giving its PID.

## Backup:

* Select Case allows user to choose the action they’d like to perform:
* **Schedule a Backup:**
* Usage of the -e and –d in if statements assured the existence of the back file and destination directory
* cronjob="$minute $hour \* \* $day cp $file\_path $backup\_dest usage of cron job string command in combination with date info previously asked for to create a back of the file into destination.
* echo \"$(date): Backup completed for $file\_path\" >> ~/backup\_log.txt: Logs the backup completion into backup\_log.txt file
* (crontab -l 2>/dev/null; echo "$cronjob") | crontab: lists all existing cron jobs of current user, outputs the new cron job string and combines the output of the existing cron jobs and the new cron job into an updated list of cron jobs
* **Show Last backup:**
* tail -n 1 ~/backup\_log.txt: “Show last backup” load last line inputted into the backup\_log.txt file which retrieves most recent backup that was logged

## Network:

* Select Case allows user to choose the action they’d like to perform
* **Show Network Detail:**
* If commands checks if ip addr show command is viable, if yes it will run it, if not it will run the ifconfig command.
* Ip addr show: retrieves and displays the network interface information
* If user does not have ip command’s package, script will run the ifconfig which generates same results.
* **Show Network Detail:**
* Usage of Network Manager package allows for more modern view of the interface names and makes it more convenient to user. If user does not have the package, script will demand user to install it to be able to disable/enable network card, to set the IP address and to list & connect to WIFI
* nmcli device status | awk '{print $1}': prints the names of the available interfaces using the Network Manager package
* if ip link show "$interface" > /dev/null: checks if the interface provided is valid
* sudo ip link set $interface up/down: depending on the user choice, the command will either/disable or enable the interface chosen
* **Set IP Address:**
  + sudo ip addr add $ip\_address dev $interface: sets the ip address of the chosen interface to the new ip address entered by user
* **List WIFI and Connect:**
  + nmcli dev Wi-Fi list: lists all available Wi-Fi network in range, if device does not have a Wi-Fi card, the list will not appear
  + nmcli dev Wi-Fi connect: connect device to the SSID name of the Wi-Fi entered by user.

## Services:

* Select and case was used for the structure of the menu layout for the services.
* **Show sevices**
  + Systemctl list-units –type=service: this command lists all the active system services.
* **Start a service**
* Awk: a text tool for formatting an output, we used it to format the text of the list of systems to just show their names.
* Read: reads the user input for the service name.
* Sudo systemctl start: starts a system service by the given name.
* **Start a service**
* Sudo systemctl stop: stops a aystem service by the given name.

## User Management:

* Select and case is used to allow the user to seamlessly pick the action that they’d like to perform
* The line *if id “$username” &>/dev/null;...* is frequently used to check if the username inputted by the user exists on the system
* Add User
  + *sudo useradd –m $username*: *useradd* adds a new user to the system (which requires superuser permissions) while the *–m* option tells the *useradd* command to create a new home directory for the newly created user
  + *sudo passwd $username*: *passwd* sets the password for the newly created user
* Give Root Permission to User
  + *awk –F: ‘{ print $1 }’ /etc/passwd*: prints the first column of the */etc/passwd* file, which shows all the users on the system
  + *sudo usermod –a –G root $username*: *usermod* modifies a user’s account, while the *–a* adds inputted user to the group specified by the *–G* option
* Delete User
  + *sudo userdel –r $username 2>/dev/null*: *userdel* deletes a user, while the *–r* option deletes the user’s home directory. *2>/dev/null* redirects any error messages so that they aren’t displayed to the user while they are executing the file
* Show Connected Users
  + *who*: displays the users currently logged into the system as well as some of their information
* Show User Groups
  + *groups $username*: displays the groups of which a user is a part
* Disconnect Remote User
  + *if who | grep –q “$username”*: the *grep –q* command and option quiets the output of the *grep* command and only returns its exit status, therefore, it check if the specified username exists in the list of users who are currently logged into the system (given by the *who* command)
  + *terminal=$(who | awk –v user=“$username” ‘$1 == user { print $2 }’)*:
    - The *who* commands gets the list of users on the system
    - The pipe operator redirects its output to the input of the *awk* command
    - The *–v user=“$username”* assigns the new variable user to the value of the shell variable *username*
    - *$1 == user* finds the line where the first column is the value of user
    - *{ print $2 }* collects the second column of this line
    - The value of the new variable *terminal* is set to value of the command inside of *$(...)*, effectively finding the terminal of the variable *username*
  + *sudo pkill –t “$terminal”*: *pkill* terminates a process, while the *–t* option specifies the terminal where the task to be killed is located
* Change User Group
  + *sudo usermod –g $groupname $username*: *usermod* modifies the user, while the *–g* option specifies the group to which to add the user, therefore this command changes the primary group of the user

## File Management:

* Select and case is used to allow the user to seamlessly pick the action that they’d like to perform
* Search for a File in a User’s Home Directory
  + *pushd* is used to save the current status of a stack, which, in this case, is the home directory of the user that the program is currently running on (/home/current username).
  + *popd* is used to revert the system back to what is saved on the stack, which is the home directory of the old username. These commands are used so that if the system changes to a different user in order to search for a file, it can switch back to where it was before.
  + *if [ -e “$filename” ]* checks to see if the inputted file exists
  + *realpath $filename* print the absolute path of the file that has been searched
* Show 10 Largest Files
  + *ls –lS | head*: *ls* lists the files of the current directory and the –*lS* option provides detailed information and sorts the output by size, while the *head* command displays the first 10 lines of the input it receives, which is the 10 largest files
* Show 10 Oldest Files
  + *ls –lt | tail*: *ls* lists the files of the current directory and the –*lt* option provides detailed information and sorts the output by modification time, while the *tail* command displays the last 10 lines, which is the 10 oldest files
* Send File via Email
  + In order to use *mailx*, you must first install the *mailutils* package by using the following commands for a Debian-based system:
    - *sudo apt-get update*
    - *sudo apt-get install mailutils*
  + i*f [[ "$email" =~ ^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$ ]];* checks if the input matches the pattern of a normal email address
  + *echo "Email sent from Bash." | mailx -s "$filename Attached" -A "$ATTACHMENT" "$email"*: uses the *mailx* command to send an email, where the *–s “$filename Attached” sets the subject of the email, –A “$ATTACHMENT”* attaches the inputed file to the email, and sends it to the email of *“$email”*
  + *if [ $? -eq 0 ]* checks if the exit status of the last executed command is 0 (ie: Was the email successfully sent?)

# GitHub

The link to our GitHub project: <https://github.com/AndrewShahini/FileMangerSystem>