

Architecture, Operating Systems & Networks Report CA2

TU259

**<Yuanshuo Du>**

**<D22125495>**

School of Computer Science

TU Dublin – City Campus

**<20230326>**

**Table of Contents**

*Functionality Checklist* 3

*Part 1* 4

*Process management in Linux* 4

*Service manager systemd* 5

*Part 2* 6

*Feature 1 - Script Architecture* 6

*Feature 2 – Backup Functionality* 9

*Feature 3 – Transfer Functionality* 11

*Feature 4 – Lockdown Folder* 12

*Feature 5 - System Health* 13

*Feature 6 – Logging and Auditing* 14

*Conclusion* 15

*Reference:* 16

# *Functionality Checklist*

|  |  |  |
| --- | --- | --- |
| ***Feature*** | ***Description*** | ***Implemented*** |
| F1 | Script Architecture | Yes |
| F2 | Backup Functionality | Yes |
| F3 | Transfer Functionality | Yes |
| F4 | Lockdown folder for Backup / Transfer | Yes |
| F5 | System Health and Scheduled Task | Yes |
| F6 | Logging and Auditing | Yes |

Have you included a video demo as part of the assignment: Yes

Link to Video: https://youtu.be/Q9cCcZKldWU

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

\_\_\_\_Yuanshuo Du\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

<D22125495>

<2023.03.26>

# *Part 1*

# *Process management in Linux*

In Linux, process management involves managing and controlling the processes that are running on the system. A process is a program in execution that has a unique process ID (PID), a user and group account, a priority level, and a state. Linux manages processes using PIDs, sessions, and groups. A PID is a number that identifies a process uniquely among all the processes on the system. A session is a collection of processes that are associated with a terminal or a login shell, while a group is a subset of processes within a session that can receive signals from the same keyboard(GeeksforGeeks, 2020). In Linux, a process in execution that has a unique process ID (PID), user and group accounts, a priority level, and a state. Processes can be managed and controlled using various tools, commands, and attributes. Some attributes associated with a process include its PID, program counter, process state, priority, general-purpose registers, list of open files, list of open devices, and CPU seconds(IBM, 2022).

To manage processes in Linux, there are several command-line tools available. The 'ps' command displays information about the currently running processes, such as their PID, user, state, CPU and memory usage, etc. The ps command is a commonly used command for displaying process attributes(sage-advices.com, 2019). It can be used with different options and formats to display specific attributes for all processes on the system. For example, the ps command with the options -e and -o can display the PID, user name, priority, nice value, virtual memory size, resident set size, state, percentage of CPU usage, percentage of memory usage, and command name for all processes on the system. The 'top' command displays real-time information about the running processes and their resource consumption. The 'kill' command sends a signal to terminate one or more processes by specifying their PID(RUMAISA NIAZI, 2022). The 'renice' command changes the priority level (or nice value) of one or more processes by specifying their PID. The 'bg' command puts a process in the background so that it can run without user interaction, while the 'fg' command brings a process to the foreground so that it can receive user input. (Jayant Verma, 2022)

Processes in Linux are started using the system() function or the more efficient and secure fork() and exec() functions(Aaron Kili, 2017). The first process that is launched during boot time is the init process, which has a PID of 1 and is responsible for starting other processes as part of the system services. All other processes that get started in Linux are either child processes, created by a parent process using fork() or system(), or daemon processes, which are background processes that run independently of any user or terminal and can be started by init or another daemon(Alex Campbell and Chris Hoffman, 2017).

As for the process group and session group, the process group in Linux is a collection of processes sharing the same process group ID (PGID), used for grouping related processes, such as those in the same command pipeline or job(IEEE and The Open Group, 2018). Process groups are useful for enabling job control, simplifying management of related processes, and helping to isolate processes from different sessions for improved security and stability.

A session group, on the other hand, is a collection of one or more process groups that share the same session ID (SID), and is used for managing processes belonging to the same login session or terminal(superuser.com, 2020). Session groups are useful for enabling login sessions, terminal sessions, and windowing sessions, providing authenticated access to system resources and interactive interfaces for users.

The key difference between a session group and a process group is that a session group can contain multiple process groups, whereas a process group can only belong to a single session group(informit.com, 2005). A session group is a higher-level grouping of processes sharing the same login session or terminal, while a process group is a lower-level grouping of processes sharing the same job or pipeline. A session group has one controlling terminal and one foreground process group, while a process group can be either in the foreground or background of its controlling terminal. A session group has one leader with the same SID as its PID, while a process group has one leader with the same PGID as its PID.

Process groups and session groups are essential concepts in Linux for organizing and managing processes. Process groups are useful for job control and managing related processes, while session groups are useful for managing login sessions, terminal sessions, and windowing sessions. The primary distinction between the two is that session groups can encompass multiple process groups, whereas process groups can only be part of a single session group.

# *Service manager systemd*

For a unit configuration file with a ".service" extension is specific to systemd and contains important information about a process controlled and supervised by the service manager(Joshua James, 2023). These files are structured plain text files that contain a collection of directives organized into distinct sections, including the [Service] section(freedesktop.org, n.d.), which encodes information about the process. The use of systemd allows for faster boot times, improved security and performance through cgroup management, reduced latency and resource consumption with socket activation and file descriptor passing, and simplified service configuration and management with a unified syntax and command-line interface. Additionally, systemd offers better logging and debugging capabilities, seamless integration with other Linux components, and advanced features such as snapshots, timers, watchdogs, and containers(askubuntu.com, 2019). By managing running services with fewer processes, systemd also reduces overhead on system resources.

Systemd provides a suite of basic building blocks for managing a Linux system efficiently and effectively(wiki.archlinux.org, 2023). It runs as PID 1 and starts the rest of the system, including managing hardware, processes and groups of processes, filesystem mounts, and much more. With fewer processes required to start a service, systemd offers a more streamlined approach to managing system resources. Using systemd also enables aggressive parallelization capabilities, on-demand starting of daemons, Linux control group tracking, and transactional dependency-based service control logic. These features, combined with its ability to maintain mount and automount points, make systemd a one-stop tool for system management. As a result, systemd is present in almost every aspect of the modern Linux operating system, providing the necessary configuration and management tools to ensure a smooth and efficient system operation.

Systemd as a service manager for Linux that provides a way to manage system services and processes. It uses unit files to describe and control these units. A unit file is a configuration file that contains information about a specific type of resource, such as a service, a socket, a device, and so on(Justin Ellingwood, 2015). These files have three main sections: [Unit], [Type], and [Install]. The [Unit] section contains generic options that are not dependent on the type of the unit. The [Type] section contains type-specific options that define how the unit operates. The [Install] section contains information about unit installation used by systemctl enable and disable commands. To create a custom systemd unit file, you need to create a file with a .service extension under the /etc/systemd/system directory, edit the file with your preferred text editor and add the necessary sections and options as explained above. Then, reload the systemd daemon with sudo systemctl daemon-reload command to make it aware of your new or changed unit file. Finally, start and enable your service using sudo systemctl start my-service and sudo systemctl enable my-service commands respectively.

An example of how to set up a service to manage a process using systemd is to create a .service file in the /etc/systemd/system directory with a name that describes your process(Red Hat, 2023). For instance, you can create a file named /etc/systemd/system/my-process.service. Edit the file with your preferred text editor and specify at least the [Unit], [Service], and [Install] sections, and provide information such as the description, type, command, dependencies, and installation of your process. For example, in the [Unit] section, you can set the description of your process and the target it should run after, such as network.target. In the [Service] section, you can specify the type of service, such as simple, and the command to run when starting the service, such as /usr/bin/my-process. In the [Install] section, you can set the target units that want your process, such as multi-user.target. Once you have edited the file, save and close it, and reload systemd daemon with sudo systemctl daemon-reload command. You can then start and enable your service using sudo systemctl start my-process and sudo systemctl enable my-process commands, respectively.

# *Part 2*

# *Feature 1 - Script Architecture*

A menu-driven user interface for administering a website is provided by this Bash shell script, which can be found here. A while loop is the only component of the script, and it will continue to execute indefinitely until the user cancels the process.

The loop starts each repetition by clearing the screen and presenting the user with a menu. This happens at the beginning of each iteration. The menu offers a variety of choices for administering the website, such as listing users, adding a user to the web group, backing up the Intranet or Live directory, transferring changes to the website, generating audit reports, and generating system health reports. All of these functions can be accessed by clicking on the appropriate option.

The user is given the opportunity to select one of several options, after which the script makes use of a case statement to determine which of several code blocks should be run in response to the user's decision. For instance, if the user selects the option to display users, the script will run the code block that actually generates the list of users. And also displayed the currently logged-in user

The following is a comprehensive breakdown of the script:

Set variables: The variables that will be utilised throughout the rest of the script are given their initial values in this portion. It also specifies the username of the user who will be performing the script, as well as the backup time, and it describes the paths that will be used to access the directories and files that will be used.

Backup Intranet Directory: This function will create a backup of the intranet directory by first creating a new directory specifically for the backup, then temporarily preventing other users from making changes to the intranet directory, copying the intranet directory to the backup directory, compressing the intranet directory, restoring permissions to the intranet directory, and finally logging the completion of the backup.

Backup Live Directory: This function will generate a backup of the live directory by first creating a new directory for the backup, then backing up the live directory, then setting the permissions of the live directory to read-only, and finally recording the accomplishment of the backup.

Transfer Updates: Using the compare\_directories() function, this function examines both the live and intranet directories to determine whether or not there have been any changes. It keeps a record of the modifications made to the site in the event that changes are discovered.log file, uses rsync to synchronise the changes with the live directory, and notes the conclusion of the process.

Generate Audit Report: This function generates an audit report by first creating a new directory for the report, then creating a new text file for the report, then setting the permissions on the audit directory, then grepping the site.log file for the actions taken by the present user, and finally recording the conclusion of the process.

Check System Health: This function examines the health of the system by presenting the percentage of disc space that is being used as well as the processes that are running on the system.

Monitor the Intranet Directory: The inotifywait command is used by this function to monitor the intranet directory for any changes that may occur. The console is updated with a record of the modifications whenever it detects a change.

Following the completion of each individual procedure, the user will be instructed to press Enter in order to proceed, after which the loop will begin once more with the display of the menu.

The script will exit if the user decides to stop the programme by selecting option q or Q from the menu. This status indicates that the programme was exited successfully.

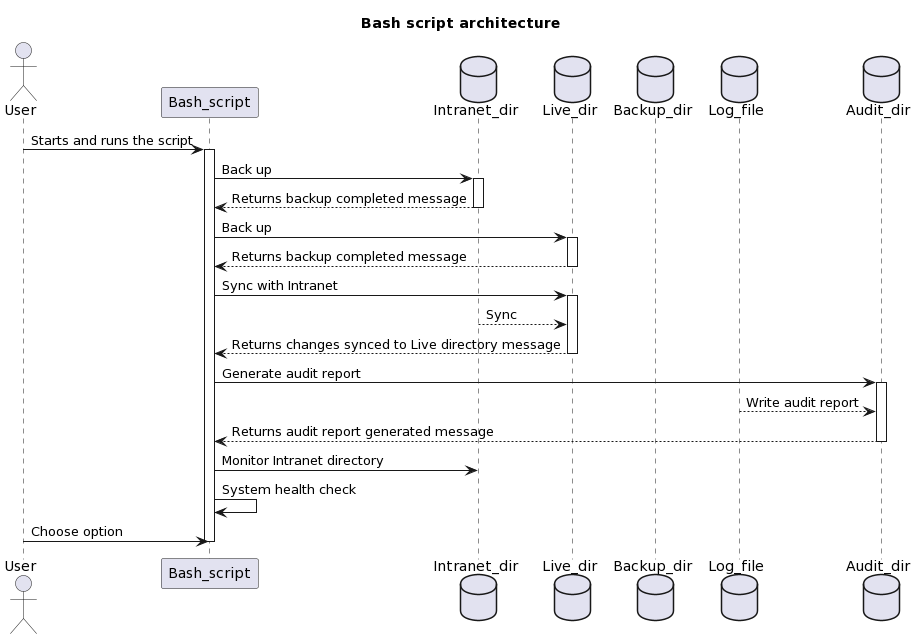
Important design principles in software development include the Separation of Concerns (SoC) and the Single Responsibility Principle (SRP). These principles encourage the flexibility, maintainability, and extensibility of a programme's code. Let's take a look at how these guiding principles were implemented in the Bash terminal script that was provided:

The Separation of Concerns (SoC) principal states that a system should be broken down into a number of smaller parts, each of which should be accountable for a distinct concern(Microsoft, 2022). The provided script contains a menu with several options, each of which is responsible for a different task or concern relating to the management of a website. These tasks and concerns include, among other things, listing users, adding users to the online group, and generating audit reports. This guarantees that each section of the code is focused on a single problem, which makes the code much simpler to comprehend, modify, and maintain.

The Single Responsibility Principle (SRP) asserts that a module, class, or function should only have one reason to change, which equates to the idea that it should have a single responsibility(Molochko, 2017). This principle can be found in the acronym SRP. In the script that has been provided, each choice presented in the menu carries out a particular activity and is accountable for only one thing. For instance, the code component responsible for adding a user to a web group does not perform any other actions besides adding the user to the group. Because of this, the code is guaranteed to be modular, repeatable, and simple to verify.

In general, the script offers a user interface for administering a website via the command line that is both practical and straightforward to work with. The script, as a whole, adheres to both the SRP and the SoC principles by separating different concerns into distinct options and ensuring that each option only has one responsibility associated with it. Because of this, the code is simple to comprehend, simple to maintain, and simple to expand, all of which are essential aspects of any software development project.

Here is the Architecture Diagram:



# *Feature 2 – Backup Functionality*

文本

描述已自动生成

文本

描述已自动生成

The code shown above specifies two functions known as backup\_intranet and backup\_live. These functions are called whenever a web server's Intranet and Live directories, respectively, need to be backed up.

The $NOW variable is used to create a backup directory that has a stamp containing the current date and time. This marks the beginning of the backup procedure. The mkdir programme is used to construct the directory, and the -p option is specified so that the parent directory is also made if it does not already exist. An error message is presented if the backup directory was not successfully created, and the function exits with a status of 1 if the if expression determines that the backup directory was not successfully created.

In the instance of backup\_intranet, the command sudo chmod is used to momentarily prevent other users from modifying the Intranet directory by removing write permissions for those users. This is done in order to protect the integrity of the backup. The $USERNAME variable then becomes the new owner of the Intranet directory, and the sudo chown command is executed in order to make this change. The $USERNAME variable stores the identity of the user who is currently operating the backup script.

The contents of the Live directory are then copied to the Intranet directory by utilising the cp command in the following step. In the final step of the process, the tar programme is used to create a compressed archive of the Intranet directory. This archive is then stored in the backup directory with a filename that contains a stamp indicating the current date and time.

It is necessary to use the sudo chown command in order to make the $USERNAME user the proprietor of the Live directory when working with backup\_live. After that, the tar programme is used to create a compressed archive of the Live directory. This archive is then stored in the backup directory with a filename that contains a stamp containing the current date and time.

After the backup has been completed, the chmod command is used to change the permissions of the Live directory so that it can only be viewed by the author and no other users.

Creating a backup directory, temporarily restricting permissions to the directories that are being backed up, copying the contents of one directory to another (in the case of backup\_intranet), creating a compressed archive of the directory, saving the archive to the backup directory, and finally restoring permissions to the directories that are being backed up are the steps that make up the backup process as a whole.

# *Feature 3 – Transfer Functionality*

文本

描述已自动生成

The preceding line of code specifies a function called transfer\_updates. This function is in charge of comparing the contents of the Intranet directory with those of the Live directory and synchronising any changes made in the Intranet directory with those made in the Live directory.

The first thing that the function does is create an inner function called compare\_directories(). This function's job is to check both the Intranet and the Live directories for any changes. The mktemp programme is used to produce a temporary file as the first step in the procedure. The contents of the two directories are compared by using the diff program, and the results of that comparison are stored in this transient file. The diff programme performs an iterative comparison (-r) of the files located in the two directories and writes the results of the comparison to a temporary file.

The -s option is then used in the following step of the procedure to determine whether or not the temporary file contains any data. If the file is empty, this indicates that there were no differences discovered, and the function will simply display the statement "No changes detected" if this is the case. On the other hand, if differences were discovered, the function records the modifications to a file known as site.log, which is a log file. Each entry in the record contains the current date, the username of the individual who was responsible for making the modifications, and the contents of the temporary file. (i.e. the differences found by the diff command).

After that, the rsync programme is executed in order to synchronise the modifications that were made in the Intranet directory with the Live directory. The rsync programme is a useful application for transferring and synchronising files quickly and easily between two different locations. In this instance, the -av options are utilised to maintain the characteristics of the files while also copying the files in a recursive fashion. By utilising the --delete option, it is possible to remove from the Live directory any files that are absent from the Intranet directory. Both the source and the destination directories are indicated by the $INTRANET\_DIR/ and $LIVE\_DIR/ parameters.

The transfer\_updates function compares the Intranet and Live directories for changes, logs any differences to a log file, and then syncs the changes from the Intranet directory to the Live directory using the rsync command.

# *Feature 4 – Lockdown Folder*

# Temporarily restrict other users from modifying the Intranet directory

  sudo chmod -R o-w "$INTRANET\_DIR"

  # Backup and compress the Intranet directory

  sudo chown $USERNAME:$USERNAME "$INTRANET\_DIR"

  sudo cp -r /var/www/html/live /var/www/html/intranet

# Restore permissions to the Intranet directory

  sudo chmod -R o+w "$INTRANET\_DIR"

The above code shows a backup process for the Intranet directory. The first command restricts write access to the Intranet directory for all other users except the owner and root users. This is done to ensure that no other user can modify the files while the backup is being taken.

The next command changes the owner of the Intranet directory to the current user (specified by $USERNAME variable). This is done to ensure that the backup is created with the correct ownership and permissions.

The third command creates a copy of the Live directory and places it inside the Intranet directory. This is done to create a backup of the Live directory inside the Intranet directory, which can be later used for restoring the website if needed.

The last command restores write access to the Intranet directory for all other users. This is done to ensure that other users can modify the Intranet directory after the backup has been taken.

For example, if the $INTRANET\_DIR variable is set to /var/www/html/intranet, the backup process would restrict write access to this directory using the chmod command, then change the ownership of the directory to the current user using the chown command. Next, it would create a backup of the Live directory inside the Intranet directory using the cp command. Finally, it would restore write access to the Intranet directory using the chmod command.

# Backup and compress the Live directory

  sudo chmod -R 777 "$LIVE\_DIR"

  sudo chown $USERNAME:$USERNAME "$LIVE\_DIR"

  sudo tar -czf "$BACKUP\_DIR/$NOW/live/Live-archive-$(date +%Y-%m-%d).zip" "$LIVE\_DIR/"

  # Restrict live directory permissions,only read

  sudo chmod -R 555 "$LIVE\_DIR"

Backup live directory is little different with backup intranet. The purpose of changing the permissions of the Live directory to allow all users to read, write and execute files within the directory is to ensure that the backup process can access all files within the directory. The ownership of the directory is then changed to the user who is executing the script to ensure that the backup archive is created with the correct permissions and ownership. The archive is then created using the tar command, which compresses the directory and creates an archive with the specified name in the specified directory. Finally, the permissions of the Live directory are restricted to allow only read access for all users to ensure that the directory is not modified during the backup process.

# *Feature 5 - System Health*

# Function to check system health

function system\_health {

  echo "Checking system health..."

  df -h

  echo

  top -n 1 -b

  echo

}

The Bash function named system\_health that checks the health of the system. It does the following:

Prints "Checking system health..." to the console.

Runs the df -h command to show the free space available on the file system.

Prints a blank line to the console.

Runs the top -n 1 -b command to show the current system processes and their resource usage.

Prints a blank line to the console.

The df -h command displays the disk space usage of all mounted file systems in a human-readable format. This command can help identify disk space issues or potential disk space issues.

The top command shows system resource usage, such as CPU usage and memory usage, and the processes that are currently running. The -n option specifies the number of iterations, and the -b option runs top in batch mode, which means it doesn't update the display and instead outputs the data in a format that is easy to parse.

For How to implement automatic nightly backups, updates and logging, I set up a separated bash shell script called auto.sh.

# Function to monitor Intranet directory for changes

function monitor\_intranet {

  echo "Monitoring Intranet Directory for changes..."

  inotifywait -m -e modify,create,delete "$INTRANET\_DIR" --exclude '^\..\*' |

  while read path action file; do

    echo "Detected change in $path$file: $action"

    sudo echo "$(date +"%Y-%m-%d %H:%M:%S") - User $USERNAME $action file '$file' on $path" >> "$LOG\_FILE"

  done

}

This script contains the monitor\_intranet function that uses the inotifywait command to monitor the specified Intranet directory for changes.

First, the function outputs a message indicating that it is monitoring the Intranet directory for changes. The inotifywait command is then used to monitor the specified Intranet directory for modifications, creations, and deletions of files. The --exclude option is used to exclude hidden files.

When a change is detected, the while loop reads the path, action, and file affected by the change. The function then logs the detected change to the specified $LOG\_FILE. The log entry contains the current date and time, the user who made the change ($USERNAME), the action taken ($action), the name of the file that was affected ($file), and the path of the file ($path).

This auto-run script also includes backup, update and logging functions which are automatically run at 1am using crontab.

To use crontab, I need to open a terminal and type the following command to edit the crontab: crontab -e. In the open crontab editor, add

0 1 \* \* \* /bin/bash /home/steven/Documents/Scripting/auto.sh

This line specifies the time at which the task will run, which is 1am each day. It then specifies the path to the script to be run.

# *Feature 6 – Logging and Auditing*

 # Log the changes to the site.log file

      echo "$(date) - User $USERNAME made the following changes to the Intranet directory:" >> $LOG\_FILE

      echo "$(date) - $USERNAME $tmp\_file" >> $LOG\_FILE

      cat $USERNAME $tmp\_file >> $LOG\_FILE

The first logging required is when transferring and updating data to the live directory, If differences were found, the changes are logged to the site log file using the echo command, which adds a timestamp and the username of the user who made the changes to the log. The contents of the temporary file are also appended to the log file using the cat command.

The second logging required is in auto.sh script. The monitor\_intranet function in auto.sh is used to monitor the Intranet directory for changes using the inotifywait command. Whenever a change is detected (i.e. a file is modified, created, or deleted), the function outputs a message to the console indicating the type of action that was taken on the file. Additionally, the function appends a log entry to the $LOG\_FILE that includes the date and time of the action, the username of the user who made the change, the type of action that was taken on the file, and the path of the file that was changed.

The log entry is written to the file specified by $LOG\_FILE, which is a variable that should be defined earlier in the script. Each log entry is appended to the end of the file, so that a record of all changes made to the Intranet directory can be kept. The log entries can then be used for auditing purposes, to track who made changes to the Intranet directory and when those changes were made.

The last logging is used in audit\_report function:

文本

描述已自动生成

In this script, log files are used to keep track of various events and actions taken on the system.

The monitor\_intranet function logs any changes made to files in the Intranet directory.

The compare\_directories function logs any changes found when comparing the Intranet and Live directories. If changes are found, the function appends a timestamp, the username of the user who made the changes, and the details of the changes to the same log file specified by LOG\_FILE.

The audit\_report function generates an audit report by searching the log file for all entries made by the current user ($USERNAME). This function will generate a txt file to output all the LOG\_FILE directory’s logs to users.

# *Conclusion*

In summary, the implementation involved creating a bash shell script that performed backups, updates, and logging for both the Intranet and Live directories. The backups were compressed and stored in a specified directory, and the system health was checked using the "system\_health" function. The logging function recorded any changes made to the directories by the user and generated an audit report of the user's activities.

In addition, there is also an automatic nightly script created. This script included functions to monitor the Intranet directory for changes using inotifywait and to compare the Intranet and Live directories for differences using diff. To automate the script, a cron job was set up to run it automatically at 1 am every night.

All the script was tested and verified to be working as expected, providing the necessary backups and updates to the directories while keeping track of any changes made by the user.

# *Reference:*

Aaron Kili. (2017, April 5). *All You Need To Know About Processes in Linux [Comprehensive Guide]*. Tecmint.Com. https://www.tecmint.com/linux-process-management/

Alex Campbell and Chris Hoffman. (2017, August 3). *How to get started with Linux: A beginner’s guide | PCWorld*. PCWorld. https://www.pcworld.com/article/427298/how-to-get-started-with-linux-a-beginners-guide.html

askubuntu.com. (2019, May). *upstart - What are the benefits of using systemd? - Ask Ubuntu*. Askubuntu.Com. https://askubuntu.com/questions/811323/what-are-the-benefits-of-using-systemd

freedesktop.org. (n.d.). *systemd.service — Service unit configuration*. Freedesktop.Org. Retrieved March 25, 2023, from https://www.freedesktop.org/software/systemd/man/systemd.service.html

GeeksforGeeks. (2020, May 17). *Process Management in Linux - GeeksforGeeks*. https://www.geeksforgeeks.org/process-management-in-linux/

IBM. (2022, November 3). *Process Attributes - IBM Documentation*. IBM. https://www.ibm.com/docs/en/cloud-pak-system-w3550/2.3.3?topic=linux-process-attributes

IEEE and The Open Group. (2018). *A collection of processes that permits the signaling of related processes.* The Open Group Base Specifications Issue. https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1\_chap03.html#tag\_03\_296

informit.com. (2005, July 8). *Sessions and Process Groups | The Process Model of Linux Application Development | InformIT*. Informit.Com. https://www.informit.com/articles/article.aspx?p=397655&seqNum=6

Jayant Verma. (2022, August 3). *Commands for Process Management in Linux | DigitalOcean*. DigitalOcean. https://www.digitalocean.com/community/tutorials/process-management-in-linux

Joshua James. (2023, March 19). *How to Create a Systemd Service in Linux - LinuxCapable*. Linuxcapable.Com. https://www.linuxcapable.com/how-to-create-a-systemd-service-in-linux/?utm\_content=cmp-true

Justin Ellingwood. (2015, February 17). *Understanding Systemd Units and Unit Files | DigitalOcean*. DigitalOcean. https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files

Microsoft. (2022, April 14). *Architectural principles | Microsoft Learn*. https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/architectural-principles

Molochko, A. (2017, April 25). *Separation of Concerns & Single Responsibility - Android Settings Example | Alexander Molochko*. https://crosp.net/blog/software-architecture/srp-soc-android-settings-example/

Red Hat. (2023). *Chapter 17. Working with systemd unit files Red Hat Enterprise Linux 8 | Red Hat Customer Portal*. Access.Redhat.Com. https://access.redhat.com/documentation/en-us/red\_hat\_enterprise\_linux/8/html/configuring\_basic\_system\_settings/assembly\_working-with-systemd-unit-files\_configuring-basic-system-settings

RUMAISA NIAZI. (2022, February 17). *Linux Process Management: The Ultimate Guide*. https://www.makeuseof.com/linux-process-management/

sage-advices.com. (2019, July 31). *What is process What are the attributes of a process? – Sage-Advices*. https://sage-advices.com/what-is-process-what-are-the-attributes-of-a-process/

superuser.com. (2020, March). *What is the definition of a “session” in linux? - Super User*. https://superuser.com/questions/651111/what-is-the-definition-of-a-session-in-linux

wiki.archlinux.org. (2023, March 19). *systemd - ArchWiki*. Wiki.Archlinux.Org. https://wiki.archlinux.org/title/Systemd