



**UNIVERSIDAD AUTÓNOMA DEL ESTADO DE
HIDALGO**

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

**LICENCIATURA EN INGENIERÍA EN
TELECOMUNICACIONES**

TESIS

**AUTOMATIZACIÓN DEL MONITOREO DE RECURSOS
CRÍTICOS EN SISTEMAS DE TELECOMUNICACIONES DE
ALTO RENDIMIENTO**

**Para obtener el título de
Licenciada en Ingeniería en Telecomunicaciones**

PRESENTA:

Erika Nayeli Medina Jiménez

Director:

Ing. Benigno Martínez Gutiérrez

Mineral de la Reforma, Hidalgo, México, 5 de agosto del 2026



Universidad Autónoma del Estado de Hidalgo

Instituto de Ciencias Básicas e Ingeniería

School of Engineering and Basic Sciences

Dirección

Office of the Director

Mineral de la Reforma, Hgo., a 24 de agosto de 2026

Número de control: ICBI-D/1428/2026

Asunto: Autorización de impresión.

DRA. OJUKY DEL ROCÍO ISLAS MALDONADO
DIRECTORA DE ADMINISTRACIÓN ESCOLAR DE LA UA EH

Con Título Quinto, Capítulo II, Capítulo V, Artículo 51 Fracción IX del Estatuto General de nuestra Institución, por este medio, le comunico que el Jurado asignado a la egresada de la Licenciatura en Ingeniería en Telecomunicaciones **Erika Nayeli Medina Jiménez**, quien presenta el trabajo de titulación "**AUTOMATIZACIÓN DEL MONITOREO DE RECURSOS CRÍTICOS EN SISTEMAS DE TELECOMUNICACIONES DE ALTO RENDIMIENTO**", ha decidido, después de revisar fundamento en lo dispuesto en el Título Tercero, Capítulo I, Artículo 18 Fracción IV; dicho trabajo en la reunión de sinodales, **autorizar la impresión del mismo**, una vez realizadas las correcciones acordadas.

A continuación, firman de conformidad los integrantes del Jurado:

Presidente: Ing. Emmanuel Gutiérrez Rojas

Secretario: Ing. Mauricio Hernández Castillo

Vocal: Ing. Benigno Martínez Gutiérrez

Suplente: Ing. Claudio Iván Durán Marroquín

Sin otro particular por el momento, reciba un cordial saludo.

Atentamente
"Amor, Orden y Progreso"

Dr. Gabriel Vergara Rodríguez
Director del ICBI

GVR/MMM

"Amor, Orden y Progreso"

Ciudad del Conocimiento, Carretera Pachuca-Tulancingo
Km. 4.5 Colonia Carboneras, Mineral de la Reforma, Hidalgo,
México. C.P 42184
Teléfono: 52 (771) 71 720 00 Ext. 40001
direccion_icbi@uaeh.edu.mx, vergara@uaeh.edu.mx



uaeh.edu.mx

Automatización del monitoreo de recursos críticos en sistemas de telecomunicaciones de alto rendimiento

Erika Nayeli Medina Jiménez

5 de agosto de 2026

Índice general

1. Introducción	8
1.1. Planteamiento del problema	10
1.2. Hipótesis	11
1.3. Objetivo	11
1.3.1. Objetivo General	11
1.3.2. Objetivos Específicos	11
1.4. Justificación	12
2. Marco Teórico	14
2.1. Estructura y gestión de archivos en Linux	14
2.1.1. Subdirectorios	14
2.2. Clasificación de Datos	14
2.2.1. Compartidos vs No Compartidos	15
2.2.2. Variable y estático	16
2.3. FHS	17
2.3.1. Directorios principales del sistema de archivos	17
2.3.2. Directorio raíz (/)	19
2.3.3. Directorio /bin	19
2.3.4. Directorio /boot	20
2.3.5. Directorio /dev	21
2.3.6. Directorio /etc	22
2.3.7. Directorio /home	24
2.4. Monitoreo y análisis del uso y consumo de almacenamiento en el sistema	24
2.5. Monitoreo del sistema	26
2.5.1. Herramientas de monitoreo	26
2.5.2. Archivos de registro del sistema	30
2.6. Monitoreo de procesos	30

2.6.1.	Monitoreo de procesos críticos en telecomunicaciones .	30
2.6.2.	Monitoreo de procesos con ps	32
2.6.3.	Personalización de la salida ps	34
2.6.4.	Uso del comando pstree	39
2.6.5.	Uso del comando top	40
2.7.	Automatización de tareas por medio de <i>Bash scripting</i>	41
2.7.1.	Fundamentos de la automatización mediante scripts en <i>Bash</i>	41
2.7.2.	Herramientas y Comandos Usados en la Automatización	43
2.8.	Automatización de tareas con cron	44
2.8.1.	Funcionamiento de Cron	44
2.8.2.	Integración de cron con otras herramientas	46
3.	Implementación del sistema monitoreo automatizado	47
3.1.	Configuración del sistema de notificaciones	47
3.1.1.	Importancia de las notificaciones en sistemas de tele- comunicaciones	48
3.1.2.	Instalación y Configuración de msmtpt	49
3.2.	Implementación de los scripts de monitoreo	51
3.2.1.	Monitoreo del Uso de Red	51
3.2.2.	Monitoreo de CPU y limpieza de procesos	57
3.2.3.	Monitoreo y optimización de memoria RAM	65
3.3.	Automatización de tareas de monitoreo mediante cron	76
3.3.1.	Configuración de crontab para ejecución automática .	76
3.3.2.	Verificación del funcionamiento de la automatización .	79
3.3.3.	Integración de los monitoreos automatizados con cron .	80
4.	Conclusiones	84

Índice de cuadros

2.1.	Estructura de directorios del sistema de archivos en Linux [20].	18
2.2.	Herramientas de monitoreo del sistema [15].	27
2.3.	Herramientas para monitoreo de la memoria [31, 32].	28
2.4.	Herramientas para el monitoreo del rendimiento del sistema [32].	28
2.5.	Herramientas para el monitoreo del tráfico de red [33].	29
2.6.	Herramientas para el monitoreo en las Telecomunicaciones [34].	29
2.7.	Archivos de registro del sistema [9, 32].	30
2.8.	Monitoreo de procesos con ps [9, 37].	33
2.9.	Descripción de campos de la salida del comando ps [9].	34
2.10.	Campos utilizados en el archivo crontab y su significado [9, 27].	45
2.11.	Valores especiales en crontab y su equivalencia en la sintaxis numérica [40].	46
3.1.	Opciones para liberar memoria caché en Linux.	71

Índice de figuras

2.1.	Captura de la salida del comando <code>ls</code> en el directorio <code>/bin</code> .	20
2.2.	Captura de la salida de comando <code>ls -lhtr</code> desde el directorio <code>/boot</code> .	21
2.3.	Captura de la salida de comando <code>ls -lhtr</code> desde el directorio <code>/dev</code> .	22
2.4.	Captura de la salida de comando <code>ls -lhtr</code> desde el directorio <code>/etc</code> .	23
2.5.	Captura de la salida de comando <code>ls -lhtr</code> desde el directorio <code>/home/erika</code> .	24
2.6.	Captura de la salida de comando <code>du -shxc --exclude=proc * sort -rh</code> desde el directorio raíz de sistema.	25
2.7.	Arquitectura simplificada de un sistema OSS/NMS utilizado para la supervisión y monitoreo centralizada de elementos de red en una infraestructura de telecomunicaciones.	32
2.8.	Captura de la salida de comando <code>ps aux</code> .	33
2.9.	Captura de la salida del comando <code>ps aux grep vim</code> .	34
2.10.	Captura de la salida del comando <code>ps -o pid,uid,comm,mem,cpu</code> .	35
2.11.	Captura de la salida del comando <code>ps</code> con campos personalizados: PID, usuario, CPU, memoria y comando.	36
2.12.	Captura del comando <code>ps</code> -eo ascendente.	36
2.13.	Captura del comando <code>ps</code> -eo descendente.	37
2.14.	Captura del comando <code>ps -u</code> .	38
2.15.	Captura del comando <code>ps</code> .	39
2.16.	Captura de la salida de comando <code>pstree</code> .	40
2.17.	Captura de la salida de comando <code>top -u erika -n 1</code> .	41
3.1.	Captura de la salida de comando <code>ifconfig</code> .	53

3.2. Captura de Configuración desde la GUI.	54
3.3. Captura del comando rx_bytes	54
3.4. Captura del comando tx_bytes	55
3.5. Captura del correo recibido después de exceder el límite de uso de Red en el Sistema.	56
3.6. Captura de la salida del comando <code>mpstat -P ALL 10 1</code>	60
3.7. Captura de la salida de comando <code>ps aux -sort=-%cpu awk '\$3 >80'</code>	61
3.8. Captura de la interfaz Graphical User Interface (GUI) System Monitor.	62
3.9. Captura del correo recibido después de exceder el límite de uso del Uso de CPU en el Sistema.	63
3.10. Captura del log adjunto al correo.	64
3.11. Captura de la salida de comando <code>free awk '/Mem/ print int((3/2) * 100)'</code>	70
3.12. Captura de la salida de comando <code>free y free -h</code>	70
3.13. Captura de la salida del comando <code>ps aux grep bash y ps -o rss= -C bash</code>	72
3.14. Captura de la salida del comando utilizado para calcular el consumo total de memoria RAM de los procesos Bash.	73
3.15. Captura del correo recibido después de exceder el límite de uso del Uso de la memoria RAM en el Sistema.	74
3.16. Captura del log adjunto al correo.	75
3.17. Captura de la salida de comando <code>crontab -e</code> en el sistema.	77
3.18. Captura de la salida de comando <code>crontab -l</code>	77
3.19. Captura de la salida de comando <code>ll /etc/ grep cron</code>	78
3.20. Captura de la salida de comando <code>cat /etc/crontab</code>	79
3.21. Captura de la salida de comando <code>grep CRON /var/log/cron</code>	80
3.22. Captura de la salida de ejecución de script bash sin indicar la ruta de la ubicación.	81
3.23. Captura de la salida de ejecución de script bash sin indicar la ruta de la ubicación.	82

Acrónimos

Bash Bourne Again Shell. 9

CD-ROM Compact Disc Read-Only Memory. 21

CLI Command Line Interface. 9

CPU Central Processing Unit. 12

DNS Domain Name System. 23

EMS Element Management System. 64

FCAPS Fault, Configuration, Accounting, Performance, and Security. 48

FHS Filesystem Hierarchy Standard. 17

GB Gigabyte. 26

GUI Graphical User Interface. 5, 62

HTTP Hypertext Transfer Protocol. 31

ID Identifier. 43

IP Internet Protocol. 16

KB Kilobyte. 26

KPI Key Performance Indicator. 31

MB Megabyte. 26

-
- NMS** Network Management System. 31
- NUMA** Non-Uniform Memory Access. 27
- OSS** Operations Support System. 31
- PID** Process Identifier. 34
- QoS** Quality of Service. 8
- RAM** Random Access Memory. 12
- RSS** Resident Set Size. 34
- SFTP** SSH File Transfer Protocol. 31
- SMTP** Simple Mail Transfer Protocol. 46
- SNMP** Simple Network Management Protocol. 31
- SO** Sistema Operativo. 9
- SSH** Secure Shell. 23
- STARTTLS** Simple Mail Transfer Protocol Secure. 50
- TLS** Transport Layer Security. 50
- USB** Universal Serial Bus. 21
- VoIP** Voice over Internet Protocol. 65

Capítulo 1

Introducción

En las telecomunicaciones es fundamental contar con sistemas en óptimas condiciones con el fin de brindar un servicio en donde la calidad destaque frente a los competidores en el mercado [1]. Con el avance constante de la tecnología, el número de aplicaciones, procesos corriendo simultáneamente y tareas que dependen de un servidor ha aumentado exponencialmente [1, 2]. Por ello, una infraestructura básica ya no es suficiente para satisfacer altas demandas de llamadas, mensajería, navegación o videoconferencias y se requiere de un equipo más sofisticado que brinde las herramientas necesarias para realizar las actividades de manera eficiente y sin contratiempos [3].

En entornos donde se gestionan procesos como servicios de voz, centrales de datos, radio bases, gestión de nodos y datos, la eficiencia y rendimiento de los sistemas, están principalmente basados en la calidad de servicio Quality of Service (QoS) que un proveedor de telecomunicaciones ofrece al usuario [2]. Cualquier fallo en los recursos críticos del sistema puede provocar pérdidas significativas por interrupción en el servicio, afectando al proveedor y perjudicando a los usuarios finales con pérdida de datos, interrupciones en llamadas, latencia, etc [4].

Para garantizar el óptimo desempeño de la red de un operador de telefonía móvil, se consideran aspectos como la velocidad, seguridad, continuidad en el servicio, disponibilidad, capacidad, licenciamiento, infraestructura y mantenimiento (incluyendo costos) [3]. Es crucial que un servidor sea capaz de ejecutar múltiples tareas en el menor tiempo posible, evitando disminución en la productividad y en la calidad de servicio brindado. A su vez, el operador debe mantener la red completamente segura, previniendo ataques que comprometan la privacidad del usuario, y provocar la pérdida de datos, robo

de información, ralentización del sistema y accesos no autorizados [5].

Otro aspecto para considerar es el costo asociado al mantenimiento. No llevar a cabo tareas preventivas y correctivas a tiempo puede derivar en problemas técnicos, pérdida de información, lentitud en el equipo, reparaciones o reemplazos, así como pérdida total del equipo o servidor, afectando la disponibilidad de los servicios que dependen de él [6].

Para mantener los sistemas en condiciones óptimas, es fundamental adoptar prácticas de mantenimiento constantes, organizadas en tareas diarias, semanales o en periodos más extensos, según el nivel de criticidad [7]. Actualmente en la mayoría de los sistemas de telefonía móvil se cuenta con gestores que permiten centralizar los nodos envueltos en la red de un operador móvil encargados de instalación de actualizaciones del sistema operativo y del software, monitoreo del espacio en disco, creación de copias de seguridad, eliminación de archivos innecesarios, supervisión de la temperatura del equipo, revisión de alarmas, por mencionar algunas tareas [8].

Hoy en día, existen diversos sistemas operativos encargados de la gestión del hardware y software de servidor. Entre los más utilizados se encuentran Windows, Mac OS y Linux [9, 10]. Este último cuenta con múltiples distribuciones, como Red Hat, Fedora de licenciamiento y Ubuntu o Debian que son Sistema Operativo (SO) de código abierto y no requieren licencias para su uso [9, 11–13].

Linux presenta varias ventajas en comparación con Windows, entre ellas destacan su acceso gratuito en la mayoría de las distribuciones [12, 14], menor vulnerabilidad a ataques de seguridad por su robusta arquitectura y su alto nivel de personalización, tanto en la interfaz gráfica como en la línea de comandos (Command Line Interface (CLI)), *Command Line Interface*). Además, ofrece compatibilidad con diversos lenguajes de programación, como Bourne Again Shell (Bash), Python, Java y JavaScript, proporcionando a los administradores un control total sobre el sistema [12].

El lenguaje de programación Bash consiste en la ejecución de secuencias de órdenes contenidas en archivos también denominados *scripts* que tienen el fin de realizar tareas específicas que son proporcionadas por administradores, usuarios o simplemente automatizaciones del sistema donde la interacción de un usuario no es indispensable [9, 15, 16].

Por ello, esta tesis se enfocará en la prevención de errores y el mantenimiento oportuno de equipos de alto rendimiento que requieren ser controlados constantemente debido a su alta demanda de tráfico y ejecución de tareas, mediante el monitoreo automatizado de puntos críticos del sistema

en entornos relacionados con las redes móviles.

El servidor será gestionado por el sistema operativo Linux, y el monitoreo automático se realizará a través de la automatización de archivos de instrucciones en Linux (scripts) por medio del lenguaje de programación Bash dentro de la interfaz de línea de comandos [15, 16]. Esto permitirá que el operador y administrador de la red de manera periódica, tenga información detallada sobre el estado del sistema y pueda tomar acciones preventivas que contribuyan al mantenimiento de un servicio estable, seguro y eficiente.

1.1. Planteamiento del problema

El rendimiento óptimo de un servidor de alto desempeño como lo son los equipos de un operador de telefonía móvil, puede verse afectado por diversos factores, entre los que destacan el uso excesivo de recursos, acumulación de archivos innecesarios por generación de logs, ejecución simultánea de procesos, problemas de conectividad, actualizaciones no gestionadas y configuraciones inseguras que vulneran la información del sistema. Además, la falta de mantenimiento preventivo y correctivo puede ocasionar fallos en el sistema, disminuyendo su estabilidad, seguridad y eficiencia [1].

El mantenimiento oportuno es sumamente importante en una red de telecomunicaciones, porque el fin es garantizar la calidad y continuidad del servicio entorno a los sistemas de alto rendimiento donde se ofrecen principalmente servicios de transmisión de voz, video, datos y gestión de información [17]. Cualquier degradación en el rendimiento de un nodo, puede ocasionar latencia, pérdida de paquetes, interrupciones en la comunicación o fallas en la disponibilidad de servicios, lo que impacta directamente la calidad del servicio (QoS) y la experiencia del consumidor [18].

Uno de los principales problemas es que muchos proveedores o incluso gestores de la red, desconocen el estado real de su infraestructura y la cantidad de recursos que se consumen en un momento determinado, ocupándose primordialmente de la configuración de servicios y dejando de lado el estado de salud de los equipos a nivel hardware y software [8]. La ausencia del monitoreo constante de recursos críticos del sistema, puede derivar en bajo rendimiento, pérdida de información, vulnerabilidades de seguridad y costos adicionales por mantenimiento correctivo o sustitución de hardware [19].

Ante esta problemática, surge la necesidad de una solución que automatice el monitoreo de los recursos críticos de la red, permitiendo identificar y

prevenir fallos antes de que afecten el funcionamiento. En las comunicaciones móviles, esto representa una ventaja invaluable en el mercado, ya que garantiza la calidad del servicio (QoS), minimiza interrupciones y optimiza la eficiencia operativa [8]. La automatización por medio de archivos que ejecutan órdenes se denominan “*scripts*”. En Linux los scripts facilitan la supervisión continua sin intervención manual [10], asegurando que los gestores reciban información oportuna sobre el estado de los nodos y con ello, realicen las acciones adecuadas a la brevedad posible.

1.2. Hipótesis

La implementación de *scripts* automatizados para el monitoreo del estado del sistema permitirá detectar y alertar al gestor de la red sobre problemas existentes mediante acciones y comandos específicos. Esto facilitará la identificación y corrección oportuna de puntos críticos, optimizando el rendimiento de los nodos, previniendo su mal funcionamiento y reduciendo fallos que afecten la productividad del administrador, así como la continuidad operativa de servicios de comunicaciones móviles. En consecuencia, se minimizan interrupciones, se mejora la calidad del servicio (QoS) y se evitan costos asociados a fallos no detectados a tiempo.

1.3. Objetivo

1.3.1. Objetivo General

Desarrollar *scripts* automatizados en Bash, ejecutados desde la línea de comandos de Linux, para monitorear los puntos críticos que afectan el rendimiento y funcionamiento óptimo de los sistemas utilizados en entornos de telecomunicaciones.

1.3.2. Objetivos Específicos

- Conocer los distintos lenguajes de programación que existen en Linux con estricto enfoque en Bash.
- Entender el sistema Operativo Linux, la gestión y monitoreo de los recursos críticos en un sistema de telecomunicaciones.

- Familiarizarse con la línea de comandos (CLI) en Linux para un eficiente uso al programar por medio del lenguaje Bash y la gestión de recursos existentes en una red.
- Conocer las principales tareas de un administrador de una red telecomunicaciones basados en sistemas Linux.
- Conocer los comandos y algunas herramientas de Linux para la revisión y monitoreo de puntos críticos que pueden afectar el rendimiento de un servidor en telecomunicaciones, como Central Processing Unit (CPU), Random Access Memory (RAM), almacenamiento y red.
- Ejecutar scripts automatizados para monitorear el estado del sistema, minimizando la intervención manual.
- Detectar y solucionar problemas derivados de un monitoreo ineficiente de un servicio de telecomunicaciones (latencia, pérdida de paquetes, etc.).
- Analizar los problemas de un sistema de telecomunicaciones obtenidos a través de los *scripts*, y realizar acciones necesarias para evitar inoperatividad.

1.4. Justificación

Implementar un sistema de monitoreo automatizado en Linux para una red de telecomunicaciones, basado en buenas prácticas y procedimientos programados, asegura un mantenimiento preventivo efectivo que crea las condiciones óptimas para una mejor experiencia del usuario. Al identificar los puntos críticos de rendimiento, estabilidad y funcionamiento para un sistema de telecomunicaciones, se mantiene un control adecuado en la gestión del sistema y en la calidad de servicio brindada al usuario final [1, 7].

Asimismo, la automatización de tareas facilita la detección de problemas, permitiendo que el administrador o gestor de la red tome acciones correctivas a tiempo sin necesidad de intervenciones manuales constantes, ayudando a optimizar tiempos y corregir puntualmente factores que afecten al desempeño de la red como lo son el consumo excesivo de recursos, acumulación de archivos generados por el sistema y usuarios, factores que provocan lentitud del sistema, degradación en la calidad del servicio, entre otros factores [9, 20].

Es necesario resaltar que la automatización del monitoreo de recursos críticos en un sistema de comunicaciones es indispensable porque evitará generar altos costos que son atribuidos a reemplazos constantes de hardware, afectación al servicio y vulnerabilidad de la información [5, 21].

Capítulo 2

Marco Teórico

2.1. Estructura y gestión de archivos en Linux

2.1.1. Subdirectorios

Linux ordena la información en una estructura jerárquica conocida como árbol de archivos. Esta estructura tiene como punto inicial el directorio raíz, comúnmente representado por una diagonal a partir del cual se derivan todos los demás directorios y archivos del sistema [22, 23].

Dentro del sistema de archivos existen diversos sistemas montados denominados subdirectorios. Los sistemas de archivos se encuentran en diferentes particiones del sistema, dispositivos o redes [10, 24].

Linux lleva a cabo procedimientos estandarizados para la gestión y organización de los sistemas de archivos, lo que facilita su administración y mejora la eficiencia del monitoreo [22, 25].

Es importante mencionar que el uso correcto de los subdirectorios en los sistemas de archivos ayudará a tener un control que mejorará la eficiencia del monitoreo [22, 25].

2.2. Clasificación de Datos

La principal diferencia entre la organización de archivos y datos en Linux es que permite clasificarlos de acuerdo con sus características y funciones. Entre las categorías más comunes se distinguen los datos compartidos y no

compartidos, así como datos variables y estáticos, criterios que contribuyen a una gestión eficiente de la información almacenada en el sistema [12, 24].

2.2.1. Compartidos vs No Compartidos

Los datos compartidos son aquellos que pueden ser accedidos y utilizados por diferentes sistemas o máquinas dentro de una red, mientras que los datos no compartidos son destinados al uso exclusivo de un único host [12, 24].

Un ejemplo de ello son los directorios creados por usuarios, los cuales pueden configurarse para permitir el acceso a múltiples usuarios o equipos mediante la asignación de permisos y privilegios según las políticas de seguridad establecidas.

En un sistema Linux con múltiples usuarios, es posible identificar y diferenciar los archivos compartidos de archivos exclusivos de cada usuario por medio del análisis de permisos y niveles de acceso asignados. Por ejemplo, un caso donde existen dos usuarios *UserA* y *UserB* en un sistema, algunos archivos o directorios pueden ser accesibles para ambos usuarios, en cambio otros permanecen restringidos al propietario correspondiente. Esta distinción permite comparar los archivos e identificar cuáles son compartidos y únicos para cada usuario.

- **UserA**

- Document.txt
- Example.jpg
- SecondDocument.txt

- **UserB**

- Document.txt
- Example.jpg
- FirstDocument.txt

Los datos compartidos entre UserA y UserB son:

- Document.txt
- Example.jpg

Los datos no compartidos del UserA son:

- SecondDocument.txt

Mientras que los datos no compartidos del UserB son:

- FirstDocument.txt

2.2.2. Variable y estático

Existe otra clasificación relevante dentro de los sistemas de archivos Linux que consiste en catalogar los datos variables y estáticos, los cuales se diferencian con la frecuencia en la que su contenido puede variar durante la gestión del sistema [12,20].

Los archivos estáticos son denominados así debido a que incluyen binarios, bibliotecas, documentación, entre otros archivos que no van a modificarse a menos que el administrador de sistema lo requiera.

Por ejemplo, un archivo de configuración del sistema que se utiliza para la resolución local de nombres de host es `/etc/hosts`. Este archivo es mayormente utilizado para asociar direcciones Internet Protocol (IP) fijas con nombres específicos que facilitan la comunicación interna de la red. Es importante señalar que las direcciones IP no cambian a menos que el administrador del sistema realice los cambios manuales [20,26].

Los archivos variables a diferencia de los estáticos, son aquellos archivos que pueden cambiar automáticamente como resultado de la ejecución de procesos, aplicaciones o servicios en la operación normal del sistema, inclusive sin la intervención de un administrador de la red [27].

Un archivo variable es `/var/log/syslog`, el cual almacena el registro de eventos, mensajes o errores en el sistema y sus servicios. Este archivo se actualiza continuamente conforme ocurren eventos, por lo que va cambiando de forma repetida en función al tipo de actividad que se registra [20,27].

Por lo tanto, se le denomina archivo estático cuando el contenido no cambia y permanece constante durante la operación común del sistema. En cambio, los archivos variable es nombrado así porque presenta cambios frecuentes los cuales dependen del sistema operativo, los servicios en ejecución o aplicaciones que actualizan información de manera variable [28].

2.3. FHS

El estándar Filesystem Hierarchy Standard (FHS) (Filesystem Hierarchy Standard) define la estructura y organización de sistema de archivos de Linux. Establece una jerarquía para la ubicación de directorios, facilitando la administración, mantenimiento y compatibilidad entre diferentes distribuciones [10].

FHS es el encargado de especificar directorios que deben estar presentes en un sistema, así como el uso y funcionalidad de cada uno de ellos dentro de la categoría del sistema de archivos. El objetivo de tener un estándar es simplificar la localización de los archivos existentes en el sistema para su correcta administración [10, 14].

2.3.1. Directorios principales del sistema de archivos

En la siguiente tabla se muestra un resumen de los directorios principales que están presentes en el estándar FHS:

Directorio	Función principal	Descripción
/	Raíz	Directorio raíz del sistema de archivos. Contiene todos los directorios y archivos del sistema.
/bin	Binario	Contiene comandos básicos que son accesibles para todos los usuarios y son utilizados para el funcionamiento del sistema.
/boot	Arranque	Contiene archivos que son utilizados para el arranque del sistema, incluyendo el kernel y archivos relacionados con el arranque.
/dev	Dispositivos	Archivos del dispositivo que representan el hardware y software.
/etc	Configuración	Archivos de configuración.
/home	Hogar	Directorios personales del usuario.
/lib	Bibliotecas	Bibliotecas compartidas.
/media	Media	Puntos de montaje para dispositivos removibles.
/mnt	Montaje	Montajes temporales manuales.
/opt	Opcionales	Paquetes de software opcionales.
/proc	Procesos	Sistema de archivos virtual que informa sobre los procesos corriendo en el sistema.
/root	Raíz del sistema	Directorio del usuario root.
/sbin	Superusuarios binarios	Binarios del sistema para la administración y el mantenimiento.
/tmp	Temporal	Archivos temporales.
/usr	Recursos de usuario	Aplicaciones, herramientas y datos del usuario.
/var	Variables	Datos variables que pueden cambiar (bases de datos, registros, etc.).

Cuadro 2.1: Estructura de directorios del sistema de archivos en Linux [20].

2.3.2. Directorio raíz (/)

El sistema de archivos es un árbol que facilita la navegación entre directorios, también permite mantener una estructura clara y simplificada de acuerdo con las funciones de cada uno de los archivos. En un sistema pueden existir diferentes particiones trabajando conjuntamente [9].

La partición raíz / contiene la información necesaria para arrancar el sistema y montar el resto de los sistemas de archivos [9, 29]. Este directorio requiere de archivos de configuración y datos esenciales para el inicio del sistema. Entre sus características más importantes se encuentran:

- Inicio del sistema [9].
- Restauración del sistema desde algún *backup* que se encuentre en un medio externo [9].
- Reparación del sistema [9].

2.3.3. Directorio /bin

El directorio /bin contiene programas y scripts esenciales del sistema que son ejecutados por los administradores del sistema y por los usuarios sin privilegios. Estos binarios son fundamentales para el correcto funcionamiento del sistema operativo [9, 14].

Algunos de los programas comunes que se encuentran en este directorio son:

- tar, gtar, xzcat, cd, command
- fg, jobs, less, gpasswd, tabs
- grep, kill, rm, dir, pr
- chmod, cp, groups, pwd, logname
- id, mkdir, sync, sum, tail
- touch, mount, su, ps, zip, nano, entre otros.

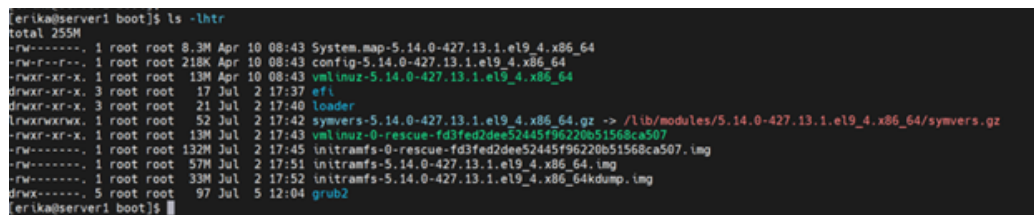
Estos programas se definen como comandos del sistema utilizados para la gestión y operación diaria de una infraestructura de telecomunicaciones basado en un sistema operativo Linux [9, 14].



Existen dos archivos esenciales que se encuentran en el directorio `/boot` denominados `vmlinuz` y `initramfs`. El archivo `vmlinuz` es el kernel del sistema operativo de Linux comprimido, también nombrado `initrd` en algunas distribuciones Linux, mientras que el archivo `initramfs` contiene un sistema de archivos temporal que es el encargado de la memoria RAM y su función principal es facilitar el arranque en el sistema y la resolución de problemas [9, 25].

El directorio `/boot` comprende de dos archivos que se utilizan para información y depuración: `config`, utilizado para la compilación del kernel, y `System.map` para depuración mediante tablas de símbolos [13].

En la figura 2.2 se muestran los archivos que incluyen el directorio `/boot`, `vmlinuz` y `initramfs`. En el caso de `vmlinuz` se tienen tres archivos, dos de ellos son la versión Kernel de Linux que tiene el sistema, ambos con una arquitectura de 64 bits y el archivo `vmlinuz-0-rescue` corresponde una imagen de kernel generada automáticamente para el rescate del sistema en caso de una falla por actualización o por errores que impidan que el sistema arranque correctamente.



```

erika@server1 boot$ ls -lhtr
total 255M
-rw-r----- 1 root root 8.3M Apr 10 08:43 System.map-5.14.0-427.13.1.el9_4.x86_64
-rw-r----- 1 root root 218K Apr 10 08:43 config-5.14.0-427.13.1.el9_4.x86_64
-rwxr-xr-x 1 root root 13M Apr 10 08:43 vmlinuz-5.14.0-427.13.1.el9_4.x86_64
drwxr-xr-x 3 root root 17 Jul 2 17:37 efi
drwxr-xr-x 3 root root 21 Jul 2 17:40 loader
lrwxrwxrwx 1 root root 52 Jul 2 17:42 symvers-5.14.0-427.13.1.el9_4.x86_64.gz -> /lib/modules/5.14.0-427.13.1.el9_4.x86_64/symvers.gz
-rwxr-xr-x 1 root root 13M Jul 2 17:45 vmlinuz-0-rescue-fd3fed2dee52445f96220b51568ca507
-rw-r----- 1 root root 132M Jul 2 17:45 initramfs-0-rescue-fd3fed2dee52445f96220b51568ca507.img
-rw-r----- 1 root root 57M Jul 2 17:51 initramfs-5.14.0-427.13.1.el9_4.x86_64.img
-rw-r----- 1 root root 33M Jul 2 17:52 initramfs-5.14.0-427.13.1.el9_4.x86_64kdump.img
drwx----- 5 root root 97 Jul 5 12:04 grub2
erika@server1 boot$

```

Figura 2.2: Captura de la salida de comando `ls -lhtr` desde el directorio `/boot`.

2.3.5. Directorio `/dev`

El directorio `/dev` almacena archivos especiales que permiten la comunicación de los dispositivos conectados al sistema incluidos los componenetes de hardware que son esenciales para el correcto funcionamiento del sistema. Entre estos se incluyen Compact Disc Read-Only Memory (CD-ROM), impresoras o terminales. Cada archivo de dispositivo actúa como una interfaz para el acceso y control del hardware correspondiente. Un ejemplo común de un archivo especial es `/dev/bus/usb`, que contiene subdirectorios de los dispositivos Universal Serial Bus (USB) detectados por el sistema [9, 10, 15].

El sistema `udev` es usado para la creación de los archivos en `/dev`.

```

[erika@server1 dev]$ ls -lhtr
total 0
drwxrwxrwt. 2 root root    40 Jul 10 12:46 tmp
drwxr-xr-x. 2 root root     0 Jul 10 12:46 pts
lrwxrwxrwx. 1 root root    15 Jul 10 12:46 stdout -> /proc/self/fd/1
lrwxrwxrwx. 1 root root    15 Jul 10 12:46 stdin -> /proc/self/fd/0
lrwxrwxrwx. 1 root root    15 Jul 10 12:46 stderr -> /proc/self/fd/2
lrwxrwxrwx. 1 root root    13 Jul 10 12:46 fd -> /proc/self/fd
lrwxrwxrwx. 1 root root    11 Jul 10 12:46 core -> /proc/kcore
lrwxrwxrwx. 1 root root    28 Jul 10 12:46 log -> /run/systemd/journal/dev-log
crw----- 1 root root   10, 239 Jul 10 12:46 uhid
drwxr-xr-x. 2 root root    80 Jul 10 12:46 bsg
drwxr-xr-x. 7 root root   140 Jul 10 12:46 disk
lrwxrwxrwx. 1 root root    12 Jul 10 12:46 initctl -> /run/initctl
drwxr-xr-x. 2 root root     0 Jul 10 12:46 hugepages
crw----- 1 root root   10, 137 Jul 10 12:46 vhci
drwxr-xr-x. 2 root root    60 Jul 10 12:46 vfiio
crw----- 1 root root  108,  0 Jul 10 12:46 ppp
drwxr-xr-x. 2 root root    60 Jul 10 12:46 net
crw-rw---- 1 root lp      6,  3 Jul 10 12:46 lp3
crw-rw---- 1 root lp      6,  2 Jul 10 12:46 lp2
crw-rw---- 1 root lp      6,  1 Jul 10 12:46 lp1
crw-rw---- 1 root lp      6,  0 Jul 10 12:46 lp0
crw-rw-rw-. 1 root kvm   10, 241 Jul 10 12:46 vhost-vsock
crw-rw-rw-. 1 root kvm   10, 238 Jul 10 12:46 vhost-net
crw-rw---- 1 root disk   10, 237 Jul 10 12:46 loop-control
crw-rw-rw-. 1 root tty    5,  0 Jul 10 12:46 tty
crw--w---- 1 root tty    5,  1 Jul 10 12:46 console
crw-rw---- 1 root dialout 4, 67 Jul 10 12:46 ttyS3
crw-rw---- 1 root dialout 4, 66 Jul 10 12:46 ttyS2
crw-rw---- 1 root dialout 4, 65 Jul 10 12:46 ttyS1
crw-rw---- 1 root dialout 4, 64 Jul 10 12:46 ttyS0
crw--w---- 1 root tty    4, 11 Jul 10 12:46 tty11
crw--w---- 1 root tty    4, 10 Jul 10 12:46 tty10
crw--w---- 1 root tty    4,  1 Jul 10 12:46 tty1
crw--w---- 1 root tty    4,  0 Jul 10 12:46 tty0
crw--w---- 1 root tty    4, 12 Jul 10 12:46 tty12

```

Figura 2.3: Captura de la salida de comando `ls -lhtr` desde el directorio `/dev`.

La salida del comando `ls -lhtr` en el directorio `/dev` mostrado en la figura 2.3 cuenta con diversos archivos como lo son: `tty`, `usbmon0`, `hpet`, `ppp`, `stdin`, entre otros.

2.3.6. Directorio `/etc`

El directorio `/etc` cuenta con archivos de configuración de la máquina local y scripts del arranque [11, 14]. En el caso de la distribución Red Hat, encontraremos el directorio `/etc/sysconfig` en donde se guardan archivos de la configuración del sistema [30]. Algunos otros subdirectorios son los siguientes:

- `/etc/passwd`: Almacena información sobre las cuentas de usuario en el sistema.
- `/etc/shadow`: Contiene contraseñas encriptadas del usuario.

- `/etc/hostname`: Archivo con el nombre del host del sistema o nombre del equipo en la red.
- `/etc/group`: Contiene los grupos del sistema y los miembros.
- `/etc/ssh`: Tiene la configuración del servidor/cliente Secure Shell (SSH) para acceso remoto a los sistemas gestionados.

La figura 2.4 contiene archivos de arranque y configuración de un sistema Linux. Es uno de los directorios fundamentales debido a que incluye servicios de red, seguridad y asignación de nombres (Domain Name System (DNS)) que son indispensables para la administración de servicios de telecomunicaciones como lo son las redes móviles o fijas [11].

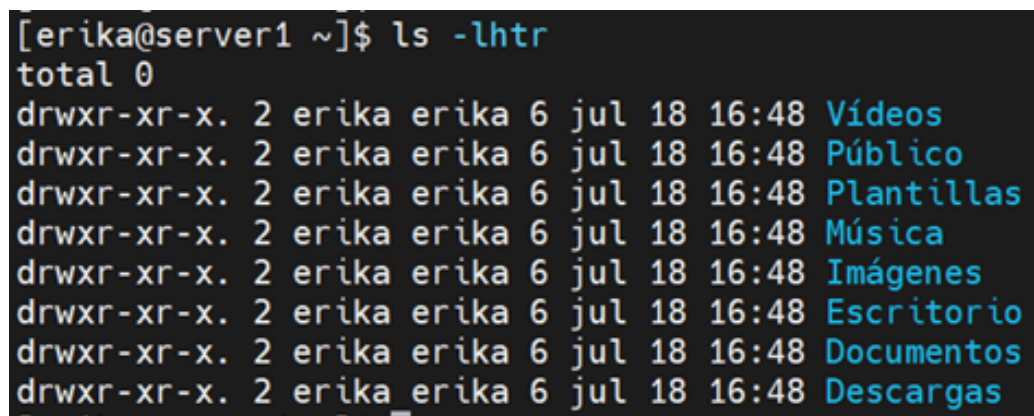
```
[erika@server1 etc]$ ls -lht
total 1.3M
-rw-r--r-- 1 root root 66K Apr 22 2020 mime.types
-rw-r--r-- 1 root root 272 Apr 22 2020 mailcap
-rw-r--r-- 1 root root 496 Jun 7 2020 logrotate.conf
-rw-r--r-- 1 root root 0 Jun 23 2020 subuid-
-rw-r--r-- 1 root root 0 Jun 23 2020 subgid-
-rw-r--r-- 1 root root 44 Jun 23 2020 shells
-rw-r--r-- 1 root root 677K Jun 23 2020 services
-rw-r--r-- 1 root root 6.5K Jun 23 2020 protocols
-rw-r--r-- 1 root root 233 Jun 23 2020 printcap
-rw-r--r-- 1 root root 58 Jun 23 2020 networks
-rw-r--r-- 1 root root 0 Jun 23 2020 motd
-rw-r--r-- 1 root root 943 Jun 23 2020 inputrc
-rw-r--r-- 1 root root 158 Jun 23 2020 hosts
-rw-r--r-- 1 root root 9 Jun 23 2020 host.conf
-rw-r--r-- 1 root root 66 Jun 23 2020 filesystems
-rw-r--r-- 1 root root 0 Jun 23 2020 exports
-rw-r--r-- 1 root root 1.4K Jun 23 2020 ethertypes
-rw-r--r-- 1 root root 1.5K Jun 23 2020 aliases
-rw-r--r-- 1 root root 2.4K Mar 1 2021 libuser.conf
-rw-r--r-- 1 root root 1.6K Aug 1 2021 rpc
-rw-r--r-- 1 root root 28 Aug 2 2021 ld.so.conf
-rw-r--r-- 1 root root 817 Aug 9 2021 xattr.conf
-rw-r--r-- 1 root root 29K Aug 9 2021 brltty.conf
-rw-r--r-- 1 root root 4.7K Aug 9 2021 encrypt.cfg
drwxr-xr-x 2 root root 6 Aug 9 2021 statetab.d
-rw-r--r-- 1 root root 94 Aug 9 2021 GREP_COLORS
-rw-r--r-- 1 root root 68 Aug 10 2021 papersize
drwxr-xr-x 2 root root 6 Aug 10 2021 libpaper.d
-rw-r--r-- 1 root root 624 Aug 10 2021 updatedb.conf
-rw-r--r-- 1 root root 11K Aug 10 2021 nanorc
-rw-r--r-- 1 root root 1.4K Aug 10 2021 pbm2ppa.conf
-rw-r--r-- 1 root root 6.2K Aug 10 2021 pnm2ppa.conf
-rw-r--r-- 1 root root 2.9K Aug 10 2021 pinforc
drwxr-xr-x 2 root root 6 Aug 10 2021 popt.d
```

Figura 2.4: Captura de la salida de comando `ls -lht` desde el directorio `/etc`.

2.3.7. Directorio /home

Los directorios personales de los usuarios se encuentran en `/home`, con excepción del usuario `root` puesto que habitualmente es encontrado en la ruta `/root` [9]. Por ejemplo, `/home/erika` mostrado en la figura 2.5 hace referencia a un usuario `/home/usuario` que tendrá acceso al contenido encontrado en esta carpeta sin requerir de permisos adicionales.

Es importante mencionar que cada usuario tendrá permisos de lectura, escritura o ejecución sobre su propio directorio personal. Si un usuario desea acceder al directorio perteneciente a un usuario diferente, deberá tener los permisos específicos para ingresar y así preservar la información y seguridad en el sistema [9].



```
[erika@server1 ~]$ ls -lht
total 0
drwxr-xr-x. 2 erika erika 6 jul 18 16:48 Vídeos
drwxr-xr-x. 2 erika erika 6 jul 18 16:48 Público
drwxr-xr-x. 2 erika erika 6 jul 18 16:48 Plantillas
drwxr-xr-x. 2 erika erika 6 jul 18 16:48 Música
drwxr-xr-x. 2 erika erika 6 jul 18 16:48 Imágenes
drwxr-xr-x. 2 erika erika 6 jul 18 16:48 Escritorio
drwxr-xr-x. 2 erika erika 6 jul 18 16:48 Documentos
drwxr-xr-x. 2 erika erika 6 jul 18 16:48 Descargas
```

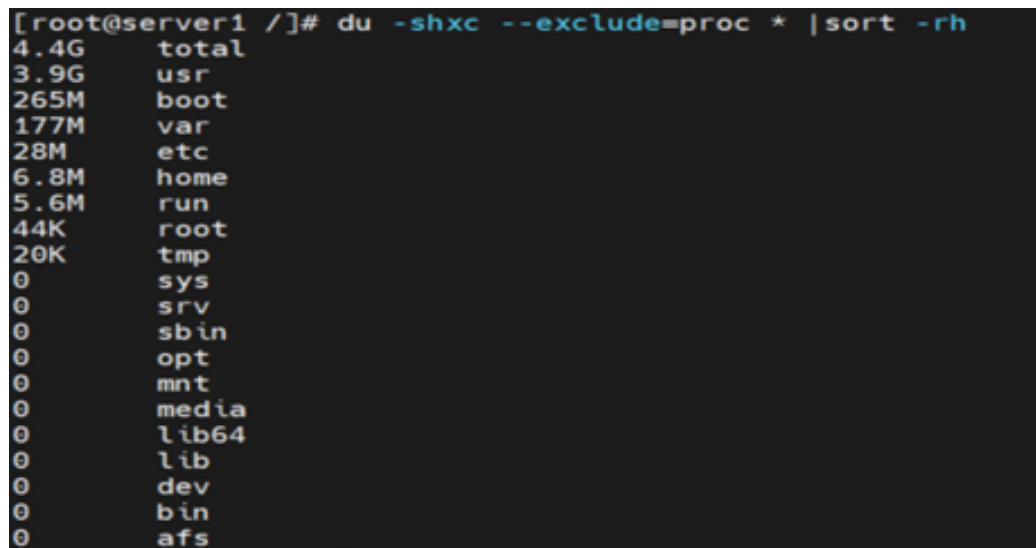
Figura 2.5: Captura de la salida de comando `ls -lht` desde el directorio `/home/erika`.

2.4. Monitoreo y análisis del uso y consumo de almacenamiento en el sistema

Es fundamental el monitoreo periódico para conocer el estado de almacenamiento en el que se encuentran los sistemas de archivos Linux [9]. Cada directorio tiene una función y contenido específico, sin embargo, algunos directorios consumen mayor espacio de almacenamiento que depende de diversos factores como la cantidad de archivos almacenados, la actividad del

sistema y los servicios en ejecución. A continuación, se presentan algunos directorios representativos:

- **/var**: Este directorio suele consumir mayor espacio debido a que almacena consigo logs de sistema o aplicaciones que no son administradas correctamente [27].
- **/home**: El directorio contiene archivos personales del usuario, si el usuario no gestiona de manera adecuada sus archivos, el consumo del espacio puede incrementar excesivamente [27].
- **/usr**: El directorio contiene la mayoría de las aplicaciones, librerías y otros recursos relacionados a software, por lo que se puede tener un alto consumo de espacio [27].



```
[root@server1 /]# du -shxc --exclude=proc * | sort -rh
4.4G    total
3.9G    usr
265M    boot
177M    var
28M     etc
6.8M    home
5.6M    run
44K     root
20K     tmp
0       sys
0       srv
0       sbin
0       opt
0       mnt
0       media
0       lib64
0       lib
0       dev
0       bin
0       afs
```

Figura 2.6: Captura de la salida de comando `du -shxc --exclude=proc * | sort -rh` desde el directorio raíz de sistema.

El comando `du` permite estimar el uso del espacio del disco en el directorio raíz y otras ubicaciones en el sistema. En la figura 2.6 se aprecia uno de los usos y opciones más comunes utilizadas del comando `du`. A continuación, se describe con detalle cada uno de los parámetros utilizados `du` [11]:

- **-s**: Muestra el tamaño total del directorio.
- **-h**: Muestra el tamaño del directorio en un formato más comprensible para el usuario (Kilobyte (KB), Megabyte (MB), Gigabyte (GB)).
- **-x**: Evita contar archivos montados de otro sistema de archivos en el mismo directorio.
- **-c**: Muestra el total de espacio ocupado por todos los directorios listados.
- **--exclude=proc**: Excluye el directorio `/proc`, ya que es un directorio virtual que no usa espacio en el disco.
- *****: Muestra los archivos y directorios del path actual.

El comando `sort` también puede ser utilizado para ordenar la salida del comando `du`:

- **-r**: Ordena la salida de mayor a menor.
- **-h**: Considera las unidades de tamaño humano (KB, MB, GB) al ordenar.
- **-r**: Ordena la salida de mayor a menor.
- **-h**: Considera las unidades de tamaño humano (KB, MB, GB) al ordenar.

2.5. Monitoreo del sistema

2.5.1. Herramientas de monitoreo

Herramientas de monitoreo de procesos y carga

Linux cuenta con herramientas instaladas para conocer cuál es el rendimiento y estado del sistema [9]. Estas herramientas utilizan pseudo-sistemas de archivos montados como lo son `/proc` y `/sys`. Los pseudo-sistemas se caracterizan porque la información es generada por el kernel del sistema operativo en tiempo real, por lo que los datos no son almacenados en el disco

duro. A través de una interfaz, estos sistemas acceden a procesos de ejecución, configuración de hardware o estadísticos del sistema [10].

Herramienta	Descripción	Paquete
top	Muestra los procesos activos en el sistema y uso de recursos.	procps
uptime	Tiempo de actividad y carga del sistema.	procps
ps	Muestra información de los procesos del sistema.	procps
pstree	Estructura de procesos en árbol.	pstree
mpstat	Estadísticas del uso de CPU en sistemas multiprocesador.	sysstat
iostat	Estadísticas sobre el rendimiento de CPU.	sysstat
sar	Monitorear y recopilar información de la actividad del sistema.	sysstat
numastat	Estadísticas de uso de memoria en sistemas con arquitectura NUMA.	numactl
strace	Rastrea llamadas y señales de un proceso.	strace

Cuadro 2.2: Herramientas de monitoreo del sistema [15].

Herramientas de monitoreo de memoria

Herramienta	Descripción	Paquete
free	Uso de memoria del sistema.	procps
vmstat	Estado de la memoria virtual del sistema.	procps
pmap	Muestra el mapa de memoria de un proceso	procps

Cuadro 2.3: Herramientas para monitoreo de la memoria [31, 32].

Herramientas de monitoreo de E/S

Herramienta	Descripción	Paquete
iostat	Estadísticas sobre el rendimiento de CPU.	sysstat
sar	Monitorear y recopilar información de la actividad del sistema.	sysstat
vmstat	Estado de la memoria virtual del sistema.	procps

Cuadro 2.4: Herramientas para el monitoreo del rendimiento del sistema [32].

Herramientas de monitoreo de red y tráfico de datos

Herramienta	Descripción	Paquete
netstat	Muestra información acerca de conexiones de red.	netstat
iptraf	Información sobre el tráfico de red en tiempo real.	iptraf
tcpdump	Muestra paquetes de datos que pasan a través de la red.	tcpdump
wireshark	Examina el tráfico de red en tiempo real.	wireshark

Cuadro 2.5: Herramientas para el monitoreo del tráfico de red [33].

Herramientas de monitoreo en telecomunicaciones

Herramienta	Descripción	Paquete
Grafana	Muestra errores, métricas y responde rápidamente a incidentes en una infraestructura. .	grafana
Nagios	Monitorea los servicios e Infraestructura de la red.	nagios
Prometheus	Recolección en tiempo real de datos de la red.	prometheus
Zabbix	Recopila datos e información de los elementos en una infraestructura.	zabbix-server

Cuadro 2.6: Herramientas para el monitoreo en las Telecomunicaciones [34].

2.5.2. Archivos de registro del sistema

La mayoría de los archivos de registro del sistema se encuentran en el directorio `/var/log`. Estos archivos son esenciales para el monitoreo y la solución de problemas, y son encargados de proporcionar información acerca del funcionamiento del sistema, así como el monitoreo de seguridad para garantizar un mantenimiento correcto [9, 11, 27].

El Daemon `syslogd` es el encargado de controlar la mensajería. Los mensajes se envían a los archivos de registro y también a la ventana de la consola del sistema. Con el comando `tail -f /var/log/messages` o `dmesg -w` se pueden supervisar los nuevos mensajes en tiempo real conforme se van generando los eventos en el sistema.

En la siguiente tabla se muestran algunos archivos de registro importantes encontrados en el directorio `/var/log`:

Herramienta	Descripción
<code>boot.log</code>	Registra mensajes sobre el arranque del sistema.
<code>dmesg</code>	Muestra mensajes del Kernel después del arranque del sistema.
<code>messages</code>	Mensajes del sistema y los servicios, incluidos advertencias y errores.
<code>secure</code>	Muestra problemas de seguridad en el sistema.

Cuadro 2.7: Archivos de registro del sistema [9, 32].

2.6. Monitoreo de procesos

2.6.1. Monitoreo de procesos críticos en telecomunicaciones

Los operadores de telefonía móvil requieren constante monitoreo y el mantenimiento de la red en condiciones óptimas para garantizar un servicio ininterrumpido [35]. Todo esto se logra a través de plataformas de gestión especializadas que permiten supervisar el estado y desempeño de la infraestructura.

El monitoreo de una infraestructura de telecomunicaciones puede llevarse a cabo en distintos niveles. El primero consiste en el monitoreo del nodo mediante CLI o por acceso a la GUI, en donde se visualizan alarmas y Key Performance Indicator (KPI)s básicos de cada uno de los nodos. Por otro lado, a nivel red, se realiza por medio de Operations Support System (OSS), Network Management System (NMS), que son plataformas utilizadas para monitorear y centralizar la supervisión y administración de cientos o miles de nodos de forma simultanea, facilitando el acceso, la gestión y el análisis general a cada uno de los elementos de la red [36].

La OSS o NMS utiliza distintos protocolos como Simple Network Management Protocol (SNMP), Hypertext Transfer Protocol (HTTP), SSH File Transfer Protocol (SFTP), Syslog, etc, los cuales sirven para monitorear, recopilar fallos y comportamientos de configuración [21].

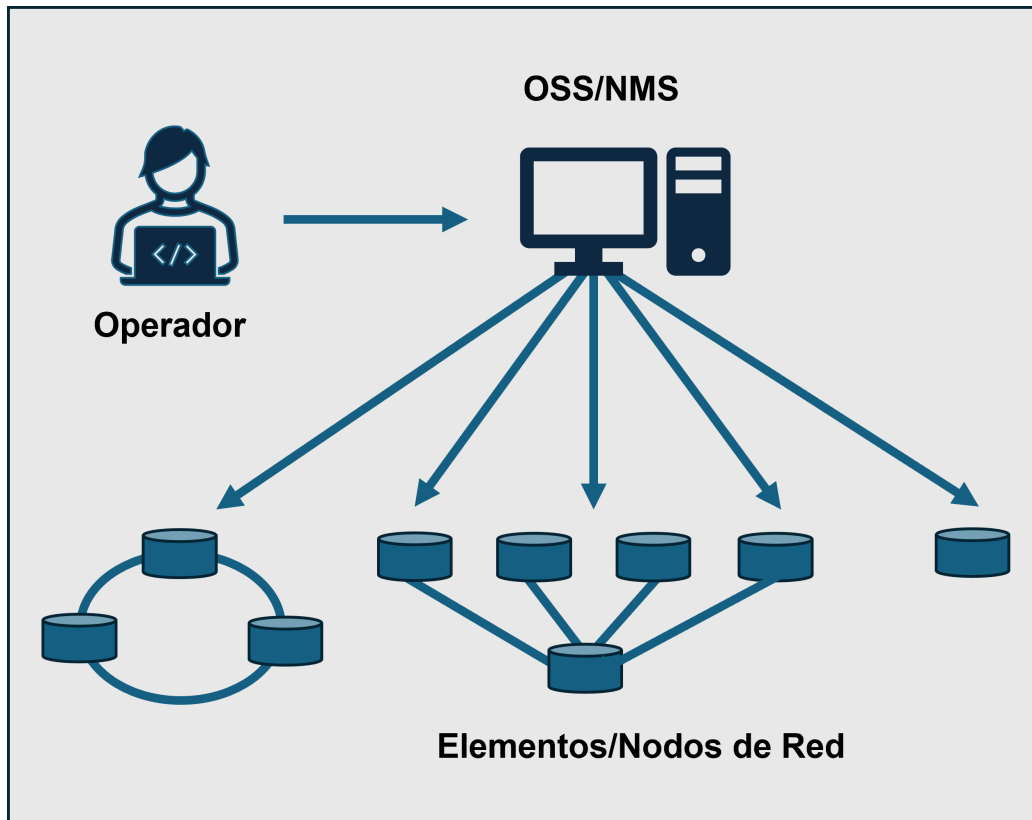


Figura 2.7: Arquitectura simplificada de un sistema OSS/NMS utilizado para la supervisión y monitoreo centralizada de elementos de red en una infraestructura de telecomunicaciones.

La figura 2.7 ejemplifica la arquitectura utilizada para la gestión de nodos y elementos en una red de telecomunicaciones, que puede variar en arquitectura (capas, niveles o sistemas intermedios) dependiendo la gestión del operador, tecnologías de los fabricantes de estas plataformas y por la cantidad de elementos de red existentes en toda la infraestructura de telecomunicaciones.

2.6.2. Monitoreo de procesos con `ps`

La herramienta `ps` es indispensable para obtener la información de los procesos que corren en el sistema, permite ver el ID del proceso, el usuario que lo ejecuta, el uso del CPU y la memoria [11].

Algunos de sus diferentes usos se mencionan a continuación:

ps uso	Descripción
ps	Procesos ejecutados en la terminal actual.
ps aux	Todos los procesos del sistema por todos los usuarios.
ps -elf	Detalle de los procesos.
ps -eL	Detalle del proceso e hilos.
ps -C "bash"	Información de los procesos que coinciden con el nombre especificado.

Cuadro 2.8: Monitoreo de procesos con `ps` [9,37].

```

(erika@server1 ~)$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.9 107576 16648 ?        Ss   17:14   0:00 /usr/lib/systemd/systemd rhgb --switched-root --system --de
root         2  0.0  0.0      0     0 ?        S    17:14   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        I<   17:14   0:00 [rcu_gp]
root         4  0.0  0.0      0     0 ?        I<   17:14   0:00 [rcu_par_gp]
root         5  0.0  0.0      0     0 ?        I<   17:14   0:00 [slub_flushwq]
root         6  0.0  0.0      0     0 ?        I<   17:14   0:00 [netns]
root         8  0.0  0.0      0     0 ?        I<   17:14   0:00 [kworker/0:0H-kblockd]
root         9  0.0  0.0      0     0 ?        I<   17:14   0:00 [kworker/u2:0-events_unbound]
root        10  0.0  0.0      0     0 ?        I<   17:14   0:00 [mm_percpu_wq]
root        11  0.0  0.0      0     0 ?        I<   17:14   0:00 [kworker/u2:1-events_unbound]
root        12  0.0  0.0      0     0 ?        I<   17:14   0:00 [rcu_tasks_kthre]
root        13  0.0  0.0      0     0 ?        I<   17:14   0:00 [rcu_tasks_rude_]
root        14  0.0  0.0      0     0 ?        I<   17:14   0:00 [rcu_tasks_trace]
root        15  0.0  0.0      0     0 ?        S    17:14   0:00 [ksoftirqd/0]
root        16  0.0  0.0      0     0 ?        S    17:14   0:00 [pr/tty0]
root        17  0.0  0.0      0     0 ?        I<   17:14   0:00 [rcu_preempt]
root        18  0.0  0.0      0     0 ?        S    17:14   0:00 [migration/0]
root        19  0.0  0.0      0     0 ?        S    17:14   0:00 [idle_inject/0]
root        21  0.0  0.0      0     0 ?        S    17:14   0:00 [cpuhp/0]
root        23  0.0  0.0      0     0 ?        S    17:14   0:00 [kdevtmpfs]
root        24  0.0  0.0      0     0 ?        I<   17:14   0:00 [inet_frag_wq]
root        25  0.0  0.0      0     0 ?        S    17:14   0:00 [kauditd]
root        26  0.0  0.0      0     0 ?        S    17:14   0:00 [khungtaskd]
root        27  0.0  0.0      0     0 ?        S    17:14   0:00 [oom_reaper]
root        28  0.0  0.0      0     0 ?        I<   17:14   0:00 [writeback]
root        29  0.0  0.0      0     0 ?        S    17:14   0:00 [kcompactd0]

```

Figura 2.8: Captura de la salida de comando `ps aux`.

En la figura 2.8 se observan todos los procesos corriendo en el servidor, en donde se incluye información detallada del usuario, número de identificador del proceso, consumo de CPU, memoria RAM, por mencionar algunos.

A continuación, se muestra un ejemplo específico de la herramienta `ps` con la opción `aux` que da detalle de todos los procesos del sistema.

Columna	Descripción
USER	Usuario que ejecuta el proceso.
PID	Identificador del proceso.
%CPU	Uso de la CPU en porcentaje.
%MEM	Uso de la memoria en porcentaje.
VSZ	Tamaño de memoria virtual del proceso en KB.
RSS	Memoria física que una tarea determinada utiliza en KB.
STAT	Estado del proceso. En la figura 2.8 se muestran dos estados: S (procesos dormidos) e I (proceso en estado de espera).
COMMAND	Comando que inició el proceso.

Cuadro 2.9: Descripción de campos de la salida del comando `ps` [9].

2.6.3. Personalización de la salida `ps`

El comando `ps` dispone de múltiples opciones y parámetros que permiten personalizar la información mostrada sobre los procesos de ejecución de acuerdo con los requerimientos específicos de monitoreo y administración del sistema.

A continuación, se muestran algunos ejemplos:

Filtro de proceso

El comando `ps aux` acompañado de un `grep vim` mostrado en la figura 2.9, sirve como auxiliar para listar únicamente el proceso solicitado, en este ejemplo, `vim` la salida contendrá toda la información relacionada a esa tarea como usuario, número de identificador, memoria RAM, CPU, el comando que inició el proceso e información relevante [9, 11].

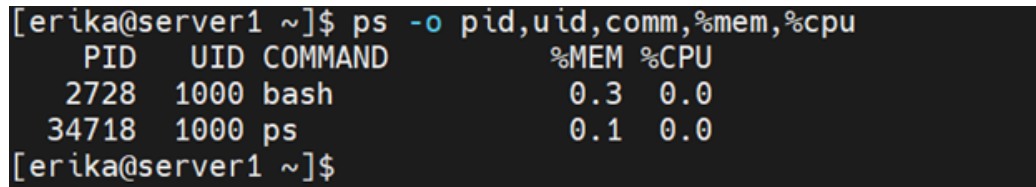
```
[erika@server1 ~]$ ps aux | grep vim
erika    38596  0.0  0.1 221672 2432 pts/0    S+   18:55   0:00 grep --color=auto vim
[erika@server1 ~]$
```

Figura 2.9: Captura de la salida del comando `ps aux | grep vim`.

En la figura 2.9 se muestra la salida del comando `texttt ps aux | grep vim`

Selección de columnas específicas

La opción `-o` sirve para mostrar las columnas deseadas.



```
[erika@server1 ~]$ ps -o pid,uid,comm,%mem,%cpu
  PID   UID COMMAND      %MEM %CPU
  2728  1000 bash          0.3  0.0
 34718  1000 ps            0.1  0.0
[erika@server1 ~]$
```

Figura 2.10: Captura de la salida del comando `ps -o pid,uid,comm,mem,cpu`

En la figura 2.10 se muestra únicamente la columna PID, UID, COMM, %MEM y %CPU, descartando el resto de la información que muestra el comando `ps` usualmente.

Formato personalizado

El comando `ps` puede acompañarse de diversas opciones dependiendo de la información que se desee visualizar en su salida. La opción `-eo` permite mostrar columnas personalizadas con encabezados definidos por el usuario [9–11, 16]. En la figura 2.11 se observa un ejemplo donde se utilizan campos personalizados como `pid`, cuyo encabezado es denominado *ID_Del_Proceso*; `user`, identificado como *Usuario*; `%cpu`, mostrado como *Uso_de_CPU*; `%mem`, presentado como *Uso_de_Memoria*; y `comm`, cuyo encabezado corresponde a *Comando*.

```
[erika@server1 ~]$ ps -eo pid=ID_Del_Proceso,user=Usuario,%cpu=Uso_De_CPU,%mem=Uso_de_Memoria,comm=Comando
ID_Del_Proceso Usuario  Uso_De_CPU  Uso_de_Memoria Comando
1 root          0.0          1.0 systemd
2 root          0.0          0.0 kthreadd
3 root          0.0          0.0 rcu_gp
4 root          0.0          0.0 rcu_par_gp
5 root          0.0          0.0 slub_flushwq
6 root          0.0          0.0 netns
8 root          0.0          0.0 kworker/0:0H-kblockd
10 root         0.0          0.0 mm_percpu_wq
12 root         0.0          0.0 rcu_tasks_kthre
13 root         0.0          0.0 rcu_tasks_rude_
14 root         0.0          0.0 rcu_tasks_trace
```

Figura 2.11: Captura de la salida del comando `ps` con campos personalizados: PID, usuario, CPU, memoria y comando.

Orden de la salida

Si se requiere personalizar la salida del comando `ps` en algún orden específico, lo recomendable es utilizar la opción `-o` descrito en la figura 2.11, la opción `-e` mostrará todos los procesos, además el comando `sort` servirá para mostrar la salida en un orden ascendente o descendente según el criterio especificado [15, 31].

```
[erika@centos ~]$ ps -eo %mem,user,%cpu,comm --sort=%cpu
%MEM USER      %CPU COMMAND
0.8 root        0.0 systemd
0.0 root        0.0 kthreadd
0.0 root        0.0 rcu_gp
0.0 root        0.0 rcu_par_gp
0.0 root        0.0 slub_flushwq
0.0 root        0.0 netns
0.0 root        0.0 kworker/u2:0-events_unbound
0.0 root        0.0 mm_percpu_wq
0.0 root        0.0 kworker/u2:1-events_unbound
0.0 root        0.0 rcu_tasks_kthre
0.0 root        0.0 rcu_tasks_rude_
```

Figura 2.12: Captura de la salida del comando `ps -eo %mem,user,%cpu,comm --sort=%cpu`

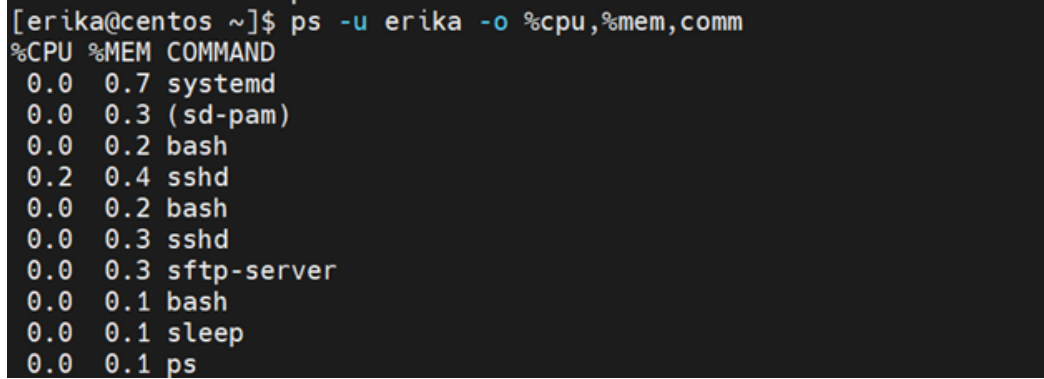
```
[erika@centos ~]$ ps -eo %mem,user,%cpu,comm --sort=-%cpu
%MEM USER      %CPU COMMAND
 0.4 erika      0.2 sshd
 0.0 root       0.1 kworker/0:3-ata_sff
 0.0 root       0.1 kworker/0:2-ata_sff
 0.0 root       0.1 kworker/0:1-events
 0.8 root       0.0 systemd
 0.0 root       0.0 kthreadd
 0.0 root       0.0 rcu_gp
 0.0 root       0.0 rcu_par_gp
 0.0 root       0.0 slub_flushwq
 0.0 root       0.0 netns
 0.0 root       0.0 kworker/u2:0-events_unbound
 0.0 root       0.0 mm_percpu_wq
 0.0 root       0.0 kworker/u2:1-events_unbound
 0.0 root       0.0 rcu_tasks_kthre
```

Figura 2.13: Captura de la salida del comando `ps -eo %mem,user,%cpu,comm --sort=-%cpu`

Las figuras 2.12 y 2.13 muestran la salida de los procesos con las columnas de uso de memoria, usuario, CPU y el proceso asociado. En la figura 2.12 los resultados se presentan de forma ascendente, desde el menor hasta el mayor uso de CPU, mientras que en la figura 2.13 se muestran en orden descendente.

Procesos de un solo usuario

El comando `ps` también muestra los procesos de un único usuario mediante la opción `-u`. En la figura 2.14 se observa la salida del comando `ps -u erika -o %cpu,%mem,comm` donde *erika* hace referencia al usuario propietario de los procesos ejecutados [16].



```
[erika@centos ~]$ ps -u erika -o %cpu,%mem,comm
%CPU %MEM COMMAND
0.0 0.7 systemd
0.0 0.3 (sd-pam)
0.0 0.2 bash
0.2 0.4 sshd
0.0 0.2 bash
0.0 0.3 sshd
0.0 0.3 sftp-server
0.0 0.1 bash
0.0 0.1 sleep
0.0 0.1 ps
```

Figura 2.14: Captura de la salida del comando `ps -u erika -o %cpu,%mem,comm`

Procesos por jerarquía

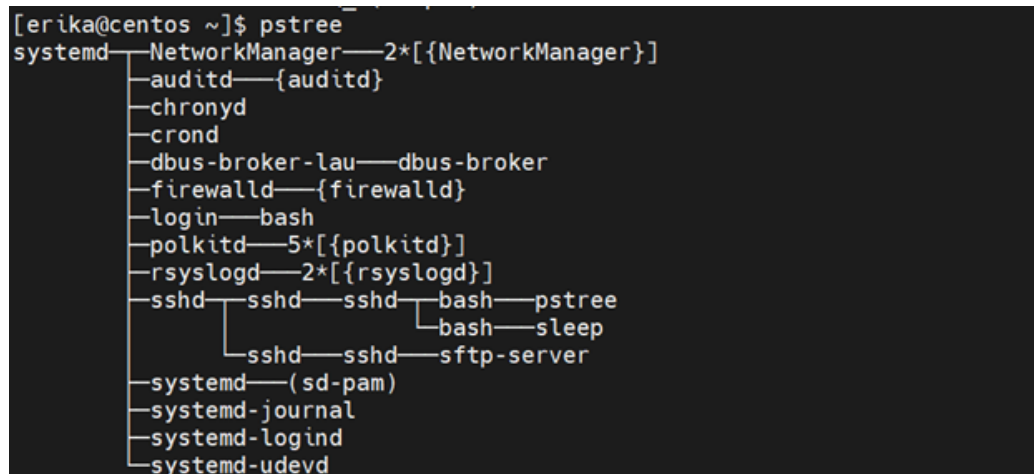
Para mostrar los procesos por orden jerárquico, es decir, la relación de cada uno de los procesos, existe la opción **forest** que desplegará una lista de las tareas en forma de árbol. En la figura 2.15 se observa la relación padre e hijo de los procesos existentes ordenados de forma jerárquica [11, 32].

```
[erika@centos ~]$ ps -eo %cpu,pid,user,comm --forest
%CPU    PID  USER      COMMAND
0.0      2   root      kthreadd
0.0      3   root      \_ rcu_gp
0.0      4   root      \_ rcu_par_gp
0.0      5   root      \_ slub_flushwq
0.0      6   root      \_ netns
0.0      9   root      \_ kworker/u2:0-events_unbound
0.0     10   root      \_ mm_percpu_wq
0.0     11   root      \_ kworker/u2:1-events_unbound
0.0     12   root      \_ rcu_tasks_kthre
0.0     13   root      \_ rcu_tasks_rude_
0.0     14   root      \_ rcu_tasks_trace
0.0     15   root      \_ ksoftirqd/0
0.0     16   root      \_ pr/tty0
0.0     17   root      \_ rcu_preempt
0.0     18   root      \_ migration/0
0.0     19   root      \_ idle_inject/0
0.0     21   root      \_ cpuhp/0
0.0     23   root      \_ kdevtmpfs
0.0     24   root      \_ inet_frag_wq
0.0     25   root      \_ kauditd
0.0     26   root      \_ khungtaskd
0.0     27   root      \_ oom_reaper
0.0     29   root      \_ writeback
0.0     30   root      \_ kcompactd0
```

Figura 2.15: Captura de la salida del comando `ps -eo %cpu,pid,user,comm --forest`

2.6.4. Uso del comando `pstree`

El comando `pstree` nos provee una descripción de la jerarquía de los procesos en forma de árbol. A diferencia del comando `ps` con la opción `--forest`, el comando `pstree` imprime la información de forma más clara y visual, lo que facilita entender cómo están relacionados cada uno de los procesos [11,32].



- Si se presiona la tecla **u**, se muestran los procesos de un usuario específico.
- Si se presiona la tecla **k**, se puede terminar un proceso.
- Si se presiona la tecla **p**, se ordenan los procesos por uso de CPU.

Además, el comando **top** se puede personalizar desde la línea de comandos sin necesidad de usar la interfaz interactiva.

```
Tasks: 110 total, 1 running, 108 sleeping, 1 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1775.7 total, 1303.3 free, 293.9 used, 325.2 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used, 1481.8 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1219	erika	20	0	22972	13912	10752	S	0.0	0.8	0:00.04	systemd
1221	erika	20	0	108248	7248	1920	S	0.0	0.4	0:00.00	(sd-pam)
1229	erika	20	0	7448	4224	3584	S	0.0	0.2	0:00.00	bash
1265	erika	20	0	19612	7588	5504	S	0.0	0.4	0:15.57	sshd
1267	erika	20	0	7448	4224	3584	S	0.0	0.2	0:00.05	bash
1295	erika	20	0	19456	7240	5376	S	0.0	0.4	0:00.00	sshd
1296	erika	20	0	8464	5888	5120	S	0.0	0.3	0:00.00	sftp-server
1316	erika	20	0	7260	3584	3328	S	0.0	0.2	0:06.00	bash
30747	erika	20	0	10572	4224	3456	T	0.0	0.2	0:00.02	top
34088	erika	20	0	5588	1920	1920	S	0.0	0.1	0:00.00	sleep
34089	erika	20	0	10572	4096	3456	R	0.0	0.2	0:00.00	top

Figura 2.17: Captura de la salida de comando **top -u erika -n 1**.

Como se muestra en la figura 2.17, la opción **-u** del comando **top** permite clasificar los procesos asociados a un usuario, mientras que la opción **-n 1** indica que la información se actualizará por una sola ocasión antes de finalizar la ejecución.

2.7. Automatización de tareas por medio de *Bash scripting*

2.7.1. Fundamentos de la automatización mediante scripts en *Bash*

La automatización de determinadas tareas por medio de *bash* permitirá el ahorro de tiempo al ejecutar tareas repetitivas a través de scripts programados para una ejecución periódica [38].

Uno de los principales beneficios de la automatización con *bash scripting* es asegurar que cada proceso o tarea será ejecutada de acuerdo a los requerimientos establecidos, eliminando principalmente el riesgo de inestabilidad cuando las tareas son ejecutadas manualmente. La reducción de costos, incremento en la eficiencia en los sistemas, escalabilidad y mayor seguridad son una de las mayores ventajas al realizar automatizaciones en los sistemas. [11, 32].

Bash scripting es una de las herramientas frecuentemente utilizadas para automatizar sistemas Linux dado que en la mayoría de las distribuciones de este sistema operativo los scripts bash se ejecutan nativamente por medio del programa *Shell* el cual permite dar ordenes a Linux usando comandos [38].

Para la ejecución automática de los scripts de monitoreo en Linux, la herramienta **cron** es comunmente utilizada puesto que permite programar la ejecución de scripts en determinados periodos. En la sección “Automatización de tareas por medio de Bash scripting” se explican algunos comandos útiles que pueden incorporarse en los scripts ejecutados por **cron** [11].

Los scripts de *bash* consisten en comandos que se ejecutan en serie para realizar una determinada acción. A través de *bash* se pueden automatizar diversas tareas de un administrador de sistemas como por ejemplo:

- **Respallos del sistema:** El uso de la automatización mediante scripts de *bash* para realizar respaldos periódicos de la información del sistema permite asegurar la manipulación correcta de los datos y evitar pérdidas importantes de información o configuraciones realizadas [11].
- **Administración de usuarios:** La gestión de usuarios es de suma importancia para mantener el sistema seguro y organizado. Desde la creación de usuarios hasta la modificación de permisos, mediante *bash* la repetición de estas tareas o errores en la asignación de permisos sería difícil de controlar cuando se cuenta con diversos usuarios [25].
- **Administración de archivos y directorios:** *Bash* puede automatizar tareas como la organización de archivos, desde su clasificación por tipo, hasta la eliminación de archivos temporales o demasiado grandes que consuman espacio en el sistema [25].
- **Instalación de programas:** Dependiendo del uso que se le dé al sistema, se podrán instalar o desinstalar programas de forma automática mediante *bash* [25].

- **Monitoreo de recursos:** En distribuciones Linux existen herramientas como `ps`, `top`, `df`, `free`, `ifconfig`, que pueden ser automatizadas para monitorear el uso de CPU, memoria RAM, espacio en disco o tráfico de red [37]. Los scripts podrán ejecutarse de manera repetitiva cada que se requiera, alertando al administrador del sistema en cuestión [16].
- **Mantenimientos preventivos:** Los scripts de *bash* se utilizan para programar tareas de mantenimiento del sistema, tales como comprobaciones de seguridad, optimización de recursos, revisión de aplicaciones instaladas, entre otros. Esto permitirá el correcto funcionamiento del sistema sin la intervención constante de algún usuario, evitando así mayores problemas a largo plazo [39].

2.7.2. Herramientas y Comandos Usados en la Automatización

- **ps:** Herramienta que permite visualizar el estado de los procesos que se están ejecutando en un sistema Linux, mostrando información como el estado actual del proceso, Identifier (ID) del proceso, prioridad, memoria utilizada, entre otros [12].
- **top:** Herramienta para visualizar en tiempo real los procesos que se están ejecutando en el sistema [12].
- **df:** Herramienta para visualizar el espacio disponible de los sistemas de archivos en el árbol del sistema Linux [39].
- **free:** Permite ver el uso de memoria RAM del sistema, así como la cantidad utilizada, la memoria libre, compartida, cacheada y disponible para iniciar nuevos procesos o aplicaciones [16].
- **ifconfig:** Utilizado para realizar configuraciones y mostrar el estado de la red, así como definir la dirección de red de interfaz en un sistema Linux [22].

2.8. Automatización de tareas con cron

2.8.1. Funcionamiento de Cron

cron es una herramienta en sistemas Linux que permite la ejecución automática de comandos o scripts en intervalos de tiempo específicos, como cada minuto, cada hora, diariamente o en cualquier otro período definido por el usuario [9, 20, 27].

Entre sus principales funciones, se destacan las siguientes:

- Automatización de la ejecución de comandos o scripts.
- Añadir tareas programadas para monitorear redes y sistemas en tiempo real.
- Programación de tareas en el sistema sin intervención manual.
- Recolección de logs del sistema.
- Creación de copias de seguridad de la información del sistema.
- Sincronización de archivos entre diferentes directorios o servidores.
- Descarga de aplicaciones o archivos de manera programada.
- Reducción de la necesidad de configuración manual por parte del usuario.
- Ejecución de tareas pendientes que deben completarse en un tiempo determinado.

La herramienta **cron** opera a través de un archivo de configuración llamado **crontab**, que contiene una lista de todas las tareas programadas junto con su respectivo intervalo de ejecución [9].

Cada línea dentro del archivo **crontab** representa una tarea específica y el tiempo de ejecución definido mediante cinco campos representado de izquierda a derecha:

Campo	Valores	Descripción
Minutos	0 a 59	Minuto en el que se ejecutará la tarea.
Horas	0 a 23	Hora en formato 24 horas.
Día del mes	1 a 31	Día del mes en el que se ejecutará la tarea.
Mes	1 a 12	Mes en el que se ejecutará la tarea.
Día de la semana	0 a 7	Día de la semana (0 y 7 representan el domingo, 1 = lunes, 2 = martes, etc.).

Cuadro 2.10: Campos utilizados en el archivo `crontab` y su significado [9,27].

Si se desea indicar todos los valores posibles para un campo, se usa el símbolo de asterisco (*) [16].

Por ejemplo, la siguiente configuración en `crontab` ejecutará un script cada minuto, sin importar la hora, el día o el mes:

```
* * * * * /home/student/primer_script.sh
```

La última columna de la configuración corresponde al comando o script que se ejecutará en el intervalo de tiempo definido [40]. Es recomendable probar manualmente el comando antes de agregarlo a `crontab` para asegurar su correcto funcionamiento.

Si se desea ejecutar un script cada 5 minutos, se utilizaría la siguiente sintaxis:

```
*/5 * * * * /home/student/monitoreo.sh
```

Intervalos especiales en `cron`

Además de la sintaxis estándar, `cron` permite el uso de valores especiales para simplificar la programación de tareas recurrentes. La siguiente tabla muestra algunos de los valores más utilizados y su equivalencia en `crontab`:

Valor	Intervalo de ejecución	Equivalencia
@yearly	Una vez cada año.	0 0 1 1 *
@monthly	Una vez cada mes	0 0 1 * *
@weekly	Una vez a la semana	0 0 * * 0
@daily	Una vez al día	0 0 * * *
@hourly	Una vez cada hora	0 * * * *

Cuadro 2.11: Valores especiales en **crontab** y su equivalencia en la sintaxis numérica [40].

El uso de estos valores permite escribir reglas de **crontab** de forma más sencilla, sin necesidad de recordar la sintaxis numérica completa.

2.8.2. Integración de **cron** con otras herramientas

En el monitoreo de recursos de Linux, **cron** es una herramienta adaptable, que fácilmente puede integrarse con diferentes herramientas existentes en un sistema de Linux, tales como:

- **msmtp**: Cliente Simple Mail Transfer Protocol (SMTP) que permite enviar correos electrónicos automáticos cuando algunos recursos exceden los umbrales recomendados en el sistema [13].
- **Logrotate**: Utilidad que gestiona la rotación automática de logs generados por las tareas de monitoreo, evitando la acumulación excesiva de archivos de registro [40].
- **rsyslog**: Servicio que permite el envío de logs a servidores remotos y garantiza la conservación de información crítica del sistema [38, 40].

Gracias a esta integración de herramientas, es posible la creación de monitoreos automatizados que envíen alertas a los usuarios, realicen registros y mantengan un control eficiente en los recursos del sistema [34].

Capítulo 3

Implementación del sistema monitoreo automatizado

3.1. Configuración del sistema de notificaciones

Para mantener un servidor de telecomunicaciones, en condiciones óptimas, es necesario monitorear constantemente los servicios, procesos y tareas en ejecución [41], sin embargo, la revisión manual de estos recursos puede ser una tarea extensa y propensa a errores. Por lo tanto, la automatización del monitoreo se presenta como una solución eficiente para mantener el rendimiento del sistema sin intervención continua. Esta automatización no solo facilita la detección temprana de problemas, sino que también mejora la gestión de los recursos del sistema, optimizando el uso del ancho de banda, la CPU y la memoria RAM. En este trabajo, se utilizarán scripts Bash para automatizar el monitoreo de los puntos más críticos del sistema, asegurando un rendimiento constante y la capacidad de reacción ante cualquier anomalía.

Al revisar periódicamente el hardware de los sistemas de telecomunicaciones, como la temperatura en la que se encuentra, el uso de CPU, la memoria RAM, el estado de los discos, es crucial para garantizar la disponibilidad del servicio [34, 42]. Además, la seguridad del sistema debe ser una prioridad para prevenir vulnerabilidades que puedan comprometer la integridad del sistema. Es fundamental realizar un control exhaustivo de los puertos abiertos, las conexiones remotas, el monitoreo de los inicios de sesión, la revisión de los usuarios, los archivos eliminados, entre otros aspectos.

Todas estas actividades requieren procesos extensos manuales para su revisión, los cuales pueden tomar horas o extenderse durante días y aumentar la probabilidad de errores humanos. Por lo tanto, la automatización del monitoreo se presenta como una solución eficiente para mantener el rendimiento del sistema sin intervención continua. Esta automatización no solo facilita la detección temprana de problemas, sino que también mejora la gestión de los recursos del sistema, optimizando el uso del ancho de banda, la CPU y la memoria RAM.

En este trabajo, se utilizarán scripts Bash para automatizar el monitoreo de los puntos más críticos del sistema, asegurando un rendimiento constante y la capacidad de reacción ante cualquier anomalía. Se enviará una notificación al administrador del sistema, quien podrá tomar las acciones necesarias para corregir los problemas antes de que afecten al sistema.

A continuación, se detallan algunos de los puntos clave que se abordarán:

- Monitoreo del uso de la red con el uso de herramientas como `ifconfig`, mencionadas en el marco teórico.
- Uso de la CPU por medio de herramientas como `ps` para gestionar el uso del CPU como se mencionó previamente.
- Monitoreo del uso de la RAM y los procesos con el objetivo de optimizar el rendimiento del sistema.

3.1.1. Importancia de las notificaciones en sistemas de telecomunicaciones

Al gestionar un sistema de telecomunicaciones es necesario se emitan notificaciones en respuesta a un evento interno. Las alertas no necesariamente son transmitidas fuera del nodo o elemento de red gestionado sino también es almacenado mediante la generación logs.

Las notificaciones son utilizadas para alertar sobre la ocurrencia de un evento específico, considerando que las alertas no siempre están relacionadas a fallos. Existen cinco áreas funcionales definidas para las notificaciones (Fault, Configuration, Accounting, Performance, and Security (FCAPS)), clasificadas por la *ITU-T* las cuales son fallos, contabilización, desempeño y seguridad, sin embargo, las notificaciones más elementales siempre son enfocadas a los fallos en los sistemas de telecomunicaciones [19, 43].

En una red de telecomunicaciones, el detectar anomalías en tiempo real es fundamental porque se implementan las medidas necesarias de manera oportuna a fallos en el tráfico de red [44].

3.1.2. Instalación y Configuración de `msmtp`

Como se ha mencionado, las automatizaciones consistirán en tareas repetitivas que se ejecutarán en intervalos determinados. En caso de detectar alguna anomalía, se enviará una notificación. Para que un correo electrónico sea enviado automáticamente, existen diversas formas de implementación. En este documento, se abordará únicamente el uso de `msmtp` para el envío de notificaciones por correo electrónico desde el sistema Linux.

En las telecomunicaciones el monitoreo continuo es crucial para identificar cualquier anomalía o falla de la infraestructura [1, 2]. Una vez detectado el evento se generan alertas o alarmas que permiten al operador realizar acciones preventivas y correctivas, con respuesta inmediata a los posibles fallos para garantizar la continuidad del servicio y calidad de la experiencia del usuario final. Sin embargo, cuando estas tareas son realizadas de forma manual puede haber retrasos, errores e ineficiencias en la gestión. Al automatizar este monitoreo permite optimizar la detección de fallas, mejorando la eficiencia operacional.

Instalación de `msmtp`

Para comenzar, es necesario instalar `msmtp` según la distribución de Linux utilizada. En este caso, se trabajará con CentOS, y el comando para la instalación será el siguiente:

```
sudo yum install msmtp
```

Configuración de `msmtp`

Una vez instalado `msmtp`, se requiere configurar el archivo `.msmtprc` que se encuentra en el directorio `home` del usuario [45]. Si el archivo no existe, se puede verificar mediante el comando `ls -a` para listar los archivos ocultos. Si no aparece, se debe crear uno nuevo utilizando un editor de texto. En este caso, se utilizará `vi` como editor:

```
vi ~/.msmtprc
```

A continuación, se muestra un ejemplo de configuración para utilizar una cuenta de Gmail con `msmtp`. La configuración se debe agregar al archivo `.msmtprc`:

```
account default
host smtp.gmail.com
port 587
from tesis_example@gmail.com
auth on
user tesis_example@gmail.com
password password_example
tls on
tls_starttls on
tls_trust_file /etc/ssl/certs/ca-bundle.crt
```

En la configuración anterior:

- **account:** Define la cuenta de correo electrónico que se utilizará para enviar los correos (en este caso, Gmail por defecto).
- **host:** Especifica el servidor SMTP al que `msmtp` se conectará. Para Gmail, se utiliza `smtp.gmail.com`.
- **port:** Indica el puerto utilizado por el servidor SMTP para la conexión. Para Gmail, el puerto recomendado es el 587, ya que utiliza una conexión segura Simple Mail Transfer Protocol Secure (STARTTLS). Si bien existen otros puertos (como 465 o 25), el puerto 587 es el más recomendado debido a su seguridad [45].
- **auth:** Define si se utilizará autenticación para conectarse al servidor de correo.
- **user:** Especifica el nombre de usuario (en este caso, la dirección de correo de Gmail).
- **password:** Es la contraseña de la cuenta de correo (en este caso, una contraseña de aplicación generada previamente).
- **tls:** Habilita la conexión segura Transport Layer Security (TLS) [32].
- **tls_starttls:** Activa la negociación de la conexión segura con STARTTLS.

- **tls_trust_file:** Especifica el archivo de certificados de confianza para la conexión TLS.

Verificación de la configuración

Una vez guardado el archivo de configuración, se debe verificar que el cliente msmtplib esté correctamente configurado para enviar correos a través de la línea de comandos.

Para comprobarlo, se puede ejecutar el siguiente comando:

```
echo "Prueba 1, configuración de msmtplib." | msmtplib ejemplo_correo@gmail.com
```

Este comando enviará un correo con el texto "Prueba 1, configuración de msmtplib." al correo electrónico especificado (en este caso, ejemplo_correo@gmail.com).

Si el correo aparece en la bandeja de entrada de la dirección de correo especificada, la configuración de msmtplib será correcta. Si no llega el mensaje, será necesario verificar que la configuración en el archivo `/.msmtplibrc` sea correcta.

Más adelante, se implementará un sistema de monitoreo automatizado donde cron ejecuta scripts de análisis de uso de recursos y envía alertas vía msmtplib cuando se detectan valores fuera de los umbrales establecidos.

3.2. Implementación de los scripts de monitoreo

3.2.1. Monitoreo del Uso de Red

El monitoreo del uso de red es fundamental para la prevención y detección de problemas de rendimiento, así como para garantizar la seguridad del sistema y el control de costos. Realizar un monitoreo continuo de las interfaces de red facilita la identificación de actividades sospechosas que puedan comprometer la seguridad del sistema. Además, permite gestionar de manera eficiente el uso del ancho de banda, no solo entre los usuarios, sino también en las aplicaciones que consumen una mayor cantidad de recursos, lo que contribuye a mejorar la experiencia del usuario [1].

El siguiente script está diseñado para monitorear el estado de la red en uso del sistema. Este script puede ser personalizado según las necesidades específicas de cada usuario.

Script de monitoreo del Uso de red

El siguiente script en Bash permite monitorear el uso de red de una interfaz específica del sistema y enviar una notificación por correo electrónico si se excede un umbral predefinido.

El script calcula el uso de datos recibidos y enviados en megabytes (MB) y, si alguno de los valores supera el umbral, se envía un correo electrónico de alerta al administrador del sistema.

Código 3.1: Script para monitoreo de uso de red

```
#!/bin/bash

Interfaz="enp0s3" # Interfaz a monitorear
Correo="ejemplo_correo@gmail.com"

Rx_B=$(cat /sys/class/net/$Interfaz/statistics/rx_bytes) # Bytes recibidos
Tx_B=$(cat /sys/class/net/$Interfaz/statistics/tx_bytes) # Bytes
      transmitidos

Rx_Mb=$((echo "scale=3; $Rx_B / 1024 / 1024" | bc)) # Convertir a MB
Tx_Mb=$((echo "scale=3; $Tx_B / 1024 / 1024" | bc)) # Convertir a MB

Umbral=20 # Umbral en MB

Uso_de_Red () {

    Asunto="Warning: Se ha excedido el uso de Red en el sistema $(hostname
    )"

    Mensaje="$1"

    echo -e "Subject: $Asunto\n\n$Mensaje" | msmtp $Correo
}

if (( $(echo "$Rx_Mb > $Umbral" | bc -l) )) || (( $(echo "$Tx_Mb > $Umbral
" | bc -l) )); then
    Uso_de_Red "Estimado usuario,\n\nSe ha usado más de $Umbral MB en la
    interfaz $Interfaz del sistema $(hostname)\n\nMB Recibidos: $Rx_Mb
    Mb\n\nMB Enviados: $Tx_Mb Mb\n\nPor favor, verifique esta informaci
    ón"
fi

echo "Uso de red total en $Interfaz - Total recibido: $Rx_Mb MB, Total
    enviado: $Tx_Mb MB. Se ha enviado un correo electrónico al correo"
```

\$Correo para más información"

Descripción del Script de monitoreo del Uso de Red

La variable `Interfaz` se obtiene directamente desde la línea de comandos de Linux mediante el comando `ifconfig` o `ip a` [34]. La elección entre uno u otro comando dependerá de la distribución de Linux utilizada. En el ejemplo mostrado en la figura 3.1, la interfaz se obtuvo desde una distribución *CentOS Stream 9*.

```
[erika@centos ~]$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.92 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 2806:103e:27:26dc:a00:27ff:feeb:4c81 prefixlen 64 scopeid 0x0<global>
    inet6 2806:103e:27:26dc::6 prefixlen 128 scopeid 0x0<global>
    inet6 fe80::a00:27ff:feeb:4c81 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:cb:4c:81 txqueuelen 1000 (Ethernet)
    RX packets 64258 bytes 60870732 (58.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 41821 bytes 4562768 (4.3 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[erika@centos ~]$
```

Figura 3.1: Captura de la salida de comando `ifconfig`.

Otra forma sencilla de identificar la interfaz de red activa es mediante el uso de la comunicación gráfica de usuario (GUI). Para ello, se puede acceder a la sección “**Configuración**” → “**Red**” → “**Interfaces**”, donde se listan todas las interfaces disponibles junto con su estado actual, la figura 3.2 muestra la configuración de la interfaz desde el entorno gráfico del sistema.

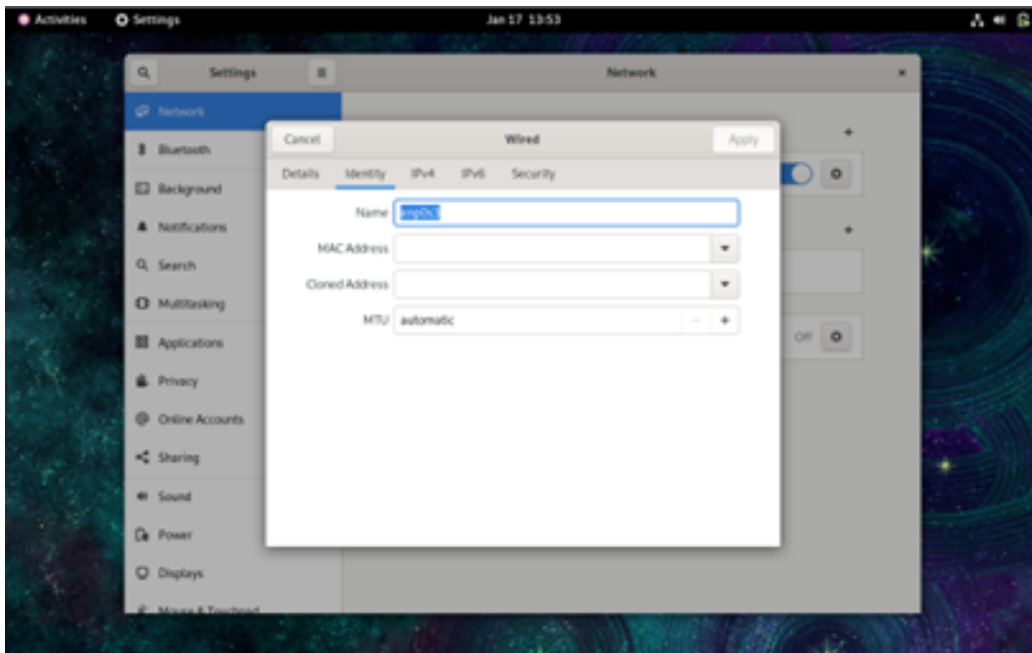


Figura 3.2: Captura de Configuración desde la GUI.

La variable **Correo** debe ser configurada con la dirección de correo electrónico a la cual se enviarán las alertas o notificaciones.

La variable **Rx_B** almacena el resultado del siguiente comando:

```
cat /sys/class/net/enp0s3/statistics/rx_bytes
```

```
[erika@centos ~]$ cat /sys/class/net/enp0s3/statistics/rx_bytes
65059533
[erika@centos ~]$
[erika@centos ~]$
```

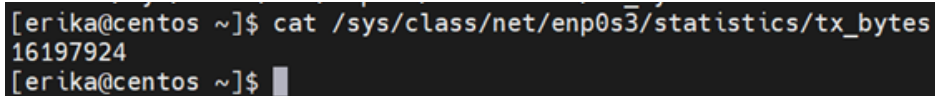
Figura 3.3: Captura de salida del comando `cat /sys/class/net/enp0s3/statistics/rx_bytes`.

La salida del comando mostrado en la figura 3.3 devuelve el total de bytes recibidos por la interfaz desde el inicio del sistema.

Mientras que la variable `Tx_B` contiene el siguiente comando, que se utiliza para conocer el total de bytes transmitidos mediante la interfaz `enp0s3`:

El resultado del comando es mostrado en la figura 3.4.

```
cat /sys/class/net/enp0s3/statistics/tx_bytes
```



```
[erika@centos ~]$ cat /sys/class/net/enp0s3/statistics/tx_bytes
16197924
[erika@centos ~]$
```

Figura 3.4: Captura de la salida del comando `cat /sys/class/net/enp0s3/statistics/tx_bytes`.

Es importante señalar que los valores obtenidos de ambos comandos están en **bytes**. La herramienta `bc` de Linux es útil para realizar cálculos, y en este caso, se emplea para convertir los valores de bytes a megabytes en las variables `Rx_B` y `Tx_B`.

La variable `Umbra1` es crucial, ya que define el valor máximo de MB que se desea permitir en el sistema. En este ejemplo, el umbral se establece en **20 MB**, pero este valor debe ajustarse según las características del sistema, como el ancho de banda disponible, el plan contratado con el proveedor de Internet y el tipo de uso (*doméstico o corporativo*). En entornos corporativos, por ejemplo, pueden existir restricciones más estrictas sobre el uso del ancho de banda.

Un ejemplo sencillo de cómo gestionar el uso de red es dividir el ancho de banda total disponible entre el número de usuarios que utilizan la red.

El script utiliza una condición `if` para enviar un correo automáticamente al gestor o usuario del sistema (según lo definido en la variable `Correo`) cuando el valor de la variable `Umbra1` es superado. En caso contrario, no se enviará ninguna notificación.



Figura 3.5: Captura del correo recibido después de exceder el límite de uso de Red en el Sistema.

La figura 3.5 contiene el correo electrónico recibido después de que se detectó por medio del script que el uso de red excedió el Umbral asignado.

La notificación contiene información relevante para la gestión oportuna en el sistema, incluyendo la cantidad de megabytes (MB) recibidos y enviados a través de la interfaz monitoreada.

Aplicación en sistemas de telecomunicaciones

El objetivo principal del monitoreo del uso de red es supervisar los datos transmitidos y recibidos en un servidor a través de las interfaces de comunicación existente en el sistema. Esta información permite identificar variaciones en el comportamiento normal del tráfico, facilitando la detección temprana de anomalías que puedan afectar el rendimiento y la disponibilidad de los servicios.

En las plataformas NMS de gestión, el monitoreo constante del uso de red permite verificar el intercambio de información entre el servidor y los elementos de red administrados. La supervisión de estas métricas contribuye a

mantener la operación adecuada de la plataforma y a detectar oportunamente posibles incidentes [36].

Cuando existe un incremento agresivo en la transmisión y recepción de datos, en comparación con los valores comunes, puede implicar que existen condiciones inusuales asociadas a posibles problemas de conectividad, generación excesiva de alarmas o fallas masivas en la infraestructura de telecomunicaciones. Por consecuencia, el monitoreo constante de uso de red mediante notificaciones juega un papel fundamental en la toma de decisiones y atención oportuna de incidentes [46].

3.2.2. Monitoreo de CPU y limpieza de procesos

Script de monitoreo de CPU y limpieza de recursos

El siguiente script ayudará a monitorear la carga total del CPU y a tomar medidas, eliminando los procesos que excedan el límite de uso establecido:

Código 3.2: Script para Monitoreo de Uso de CPU

```
#!/bin/bash

Log="/$HOME/Informacion_CPU_Uso"
Umbral=80
Monitoreo=10
Correo="ejemplo_correo@gmail.com"
hora=$(date +" %H: %M: %S")
fecha=$(date +" %Y- %m- %d")

if [ -f "$Log" ]; then
    rm "$Log"
fi

echo "[$(date)] Iniciando monitoreo del Uso de CPU en el sistema $(hostname)" >> "$Log"

mpstat -P ALL $Monitoreo 1 >> "$Log"

correo() {

    Asunto="Warning: Uso de CPU excedido"

    Mensaje="$1"
```

```
(echo -e "Subject: $Asunto\nMIME-Version: 1.0\nContent-Type: multipart/
mixed; boundary=frontier\n\n--frontier\nContent-Type: text/plain\n\
n$Mensaje.\n\n--frontier\nContent-Type: text/plain; name=cpu.log\
nContent-Transfer-Encoding: base64\nContent-Disposition: attachment;
filename=Informacion_CPU_Uso.log\n"; base64 $Log) | msmtplib $Correo

}

kill_proceso() {

if PID_Alto=$(ps aux --sort=-%cpu | awk '$3 > 80 {print $2}'); then

    for PID in $PID_Alto; do

        echo "[$(date)] Se ha detectado que el proceso con PID $PID excede
        el umbral de uso del CPU." >> "$Log"

        echo "[$(date)] Eliminando proceso con PID $PID." >> "$Log"

        kill -9 $PID

        if [ $? -eq 0 ]; then

            echo "[$(date)] Se ha eliminado el proceso con PID $PID
            satisfactoriamente." >> "$Log"

            ps aux --sort=-%cpu | awk '$3 > 80' >> "$Log"

        else

            echo "[$(date)] Ha surgido un error al eliminar el proceso con
            PID $PID." >> "$Log"

        fi
    done
fi

}

CPU=$(mpstat 1 1 | grep "all" | awk '{print $13}' | head -n 1)

CPU=$(echo "100 - $CPU" | bc)
```

```
if (( $(echo "$CPU > $Umbra1" | bc -l) )); then

    echo "[$(date)] ALERTA: Uso de CPU alto: $CPU%" >> "$Log"

    echo "[$(date)] Verificando procesos con alto uso de CPU." >> "$Log"
    "

    ps aux --sort=-%cpu | awk '$3 > 80' >> "$Log"

    proceso=$(ps aux --sort=-%cpu | awk '$3 > 80')

    kill_proceso

    correo "Estimado usuario, la utilización del CPU alcanzó el nivel máximo a
        las $hora con fecha $fecha en el sistema $(hostname).\n\nPorcentaje
        alcanzado: $CPU%\n\nSe realizó la limpieza de lo siguiente:\n\
        n$proceso.\n\n Para mayor información, consultar el log adjunto a este
        correo"

    echo "Se ha enviado un correo electrónico a $Correo con la información
        detallada."

else

    echo "[$(date)] No se ha detectado alta carga en el uso del CPU." >> "$Log"
    "

    echo "No se ha detectado alta carga en el uso del CPU."

fi
```

Descripción del script de monitoreo de CPU y limpieza de procesos

El script comienza definiendo varias variables, entre las cuales destaca Log, que indica el directorio donde se creará automáticamente un archivo log, tras ejecutar el script. Es importante notar que en el entorno de Linux existen variables predeterminadas, como \$HOME, que apunta al directorio del usuario que está ejecutando el script. De este modo, no es necesario escribir la ruta completa, sino que el sistema tomará automáticamente el directorio del usuario.

El directorio puede ser cualquier otro, pero se debe asegurar que el usuario tenga los permisos necesarios para leer, escribir y modificar archivos dentro de dicho directorio.

Una de las variables más importantes es `Umbra1`, que define el porcentaje máximo de uso que el CPU puede alcanzar antes de que se tomen acciones. Este valor debe ajustarse según las características y el uso del sistema. Como recomendación, no debe exceder el 80 % de utilización, ya que valores cercanos al 90-95 % podrían provocar una ralentización del sistema.

Funcionamiento del script de monitoreo de CPU y limpieza de procesos

El script realiza las siguientes tareas:

1. **Generación del archivo de log:** El script verifica si el archivo de log existe en el directorio seleccionado. Si no es así, lo crea. Si ya existe, lo elimina y lo vuelve a crear.
2. **Monitoreo en tiempo real del uso del CPU:** El comando `mpstat -P ALL Monitoreo1»"Log"`, mostrado en la figura 3.6 se utiliza para monitorear el uso del CPU en tiempo real. Aquí, `$Monitoreo` es una variable que define el intervalo de tiempo para la revisión. Este comando proporciona información detallada sobre el uso del CPU.

```
[erika@centos ~]$ mpstat -P ALL 10 1
Linux 5.14.0-547.el9.x86_64 (centos) 01/20/2025 _x86_64_ (4 CPU)
```

	CPU	%usr	%nice	%sys	%iowait	%irq	%soft	%steal	%guest	%gnice	%idle
05:26:44 PM	all	0.25	0.00	0.28	0.00	0.00	0.15	0.00	0.00	0.00	99.32
05:26:54 PM	0	0.30	0.00	0.20	0.00	0.00	0.10	0.00	0.00	0.00	99.39
05:26:54 PM	1	0.20	0.00	0.20	0.00	0.00	0.10	0.00	0.00	0.00	99.49
05:26:54 PM	2	0.20	0.00	0.30	0.00	0.00	0.10	0.00	0.00	0.00	99.39
05:26:54 PM	3	0.30	0.00	0.40	0.00	0.00	0.30	0.00	0.00	0.00	98.99
Average:	CPU	%usr	%nice	%sys	%iowait	%irq	%soft	%steal	%guest	%gnice	%idle
Average:	all	0.25	0.00	0.28	0.00	0.00	0.15	0.00	0.00	0.00	99.32
Average:	0	0.30	0.00	0.20	0.00	0.00	0.10	0.00	0.00	0.00	99.39
Average:	1	0.20	0.00	0.20	0.00	0.00	0.10	0.00	0.00	0.00	99.49
Average:	2	0.20	0.00	0.30	0.00	0.00	0.10	0.00	0.00	0.00	99.39
Average:	3	0.30	0.00	0.40	0.00	0.00	0.30	0.00	0.00	0.00	98.99

```
[erika@centos ~]$
```

Figura 3.6: Captura de la salida del comando `mpstat -P ALL 10 1`.

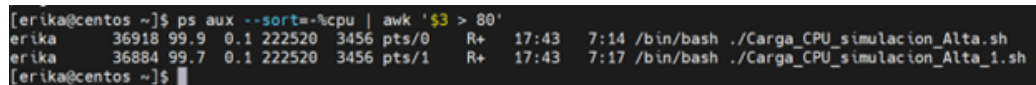
3. **Identificación de procesos que exceden el límite de CPU:** En la Figura 3.6, se puede identificar en la columna *idle* el porcentaje de uso inactivo de todas las CPUs disponibles en el sistema, siendo 99.32 % una cantidad óptima. El cálculo de la utilización de la CPU se obtiene

restando el valor de *%idle* al 100 %, tomando como referencia la línea `CPU all`, la cual promedia todas las CPUs físicas y lógicas del sistema.

En caso de que se detecte el uso del CPU superior al valor establecido en la variable `Umbral`, el script identificará los procesos que consumen más recursos mediante el siguiente comando:

```
ps aux --sort=-%cpu | awk '$3 > 80 {print $2}'
```

La salida mostrada en la figura 3.7 muestra todos los procesos en ejecución y los ordena de forma descendente según el porcentaje de uso del CPU (columna 3). Luego, utilizando `awk`, se filtran los procesos cuyo uso de CPU excede el 80 %. El identificador de proceso (PID), ubicado en la columna 2, se utiliza posteriormente para tomar acciones y finalizar los procesos problematicos. La herramienta `awk` permite extraer y procesar únicamente las columnas requeridas de la salida.



```
[erika@centos ~]$ ps aux --sort=-%cpu | awk '$3 > 80'
erika 36918 99.9 0.1 222520 3456 pts/0 R+ 17:43 7:14 /bin/bash ./Carga_CPU_simulacion_Alta.sh
erika 36884 99.7 0.1 222520 3456 pts/1 R+ 17:43 7:17 /bin/bash ./Carga_CPU_simulacion_Alta_1.sh
[erika@centos ~]$
```

Figura 3.7: Captura de la salida de comando `ps aux --sort=- %cpu | awk '$3 >80'`.

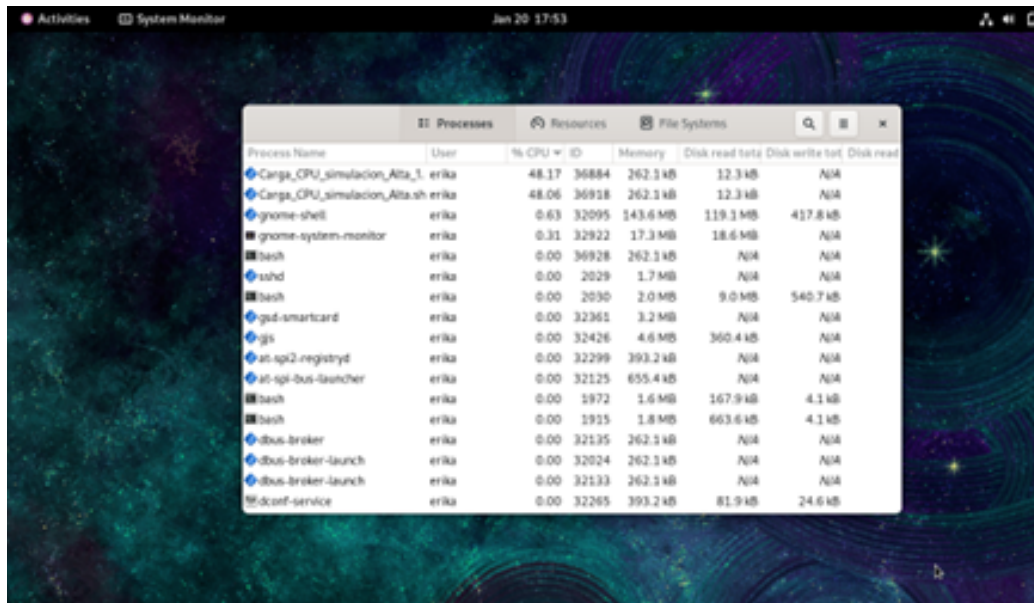


Figura 3.8: Captura de la interfaz GUI System Monitor.

Como se visualiza en la figura 3.8 la parte gráfica también es una herramienta útil para gestionar los procesos corriendo en el sistema, pero implica intervención manual.

4. **Eliminación de procesos problemáticos:** Una vez identificados los procesos o aplicaciones que están utilizando un alto porcentaje de la CPU, el script será capaz de eliminarlos automáticamente con el siguiente comando:

```
kill -9 $PID
```

En este comando, PID corresponde al identificador del proceso que está excediendo el umbral de uso de la CPU. Este paso es opcional, ya que depende de las necesidades y configuraciones del usuario. Si no se desean eliminar los procesos automáticamente, se puede **comentar**", es decir, insertar el símbolo "#" al inicio para agregar u omitir una línea y que el sistema no la ejecute.

5. **Notificación por correo electrónico:** Una vez que se haya optimizado el uso de la CPU, el script enviará una notificación por correo electrónico utilizando la herramienta `msmtp`, tal como se muestra en la figura 3.9. El mensaje notificará al usuario que el uso de la CPU fue excedido y que se eliminaron los procesos responsables del evento. Además, el correo incluirá información detallada sobre los procesos, su porcentaje de uso, la fecha de ejecución, así como un archivo de log que contendrá toda la información necesaria para futura referencia del usuario. Un ejemplo del contenido del log es mostrado en la figura 3.10.



Figura 3.9: Captura del correo recibido después de exceder el límite de uso del Uso de CPU en el Sistema.

```

[Mon Jan 20 04:21:30 PM EST 2025] Iniciando monitoreo del Uso de CPU en el sistema centos
Linux 5.14.0-547.el9.x86_64 (centos) 01/20/2025 _x86_64_ (4 CPU)

04:21:30 PM CPU %usr %nice %sys %iowait %irq %soft %steal %quest %gnice %idle
04:21:41 PM all 2.88 0.00 0.24 0.00 0.00 0.19 0.00 0.00 0.00 96.70
04:21:41 PM 0 0.19 0.00 0.19 0.00 0.00 0.00 0.00 0.00 0.00 99.61
04:21:41 PM 1 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
04:21:41 PM 2 0.00 0.00 0.29 0.00 0.00 0.00 0.00 0.00 0.00 99.71
04:21:41 PM 3 87.10 0.00 0.00 0.00 0.00 12.90 0.00 0.00 0.00 0.00

Average: CPU %usr %nice %sys %iowait %irq %soft %steal %quest %gnice %idle
Average: all 2.88 0.00 0.24 0.00 0.00 0.19 0.00 0.00 0.00 96.70
Average: 0 0.19 0.00 0.19 0.00 0.00 0.00 0.00 0.00 0.00 99.61
Average: 1 100.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Average: 2 0.00 0.00 0.29 0.00 0.00 0.00 0.00 0.00 0.00 99.71
Average: 3 87.10 0.00 0.00 0.00 0.00 12.90 0.00 0.00 0.00 0.00

[Mon Jan 20 04:21:47 PM EST 2025] ALERTA: Uso de CPU alto: 4.75%
[Mon Jan 20 04:21:47 PM EST 2025] Verificando procesos con alto uso de CPU.
erika 6993 99.7 0.1 222520 3456 pts/1 R+ 16:21 0:42 /bin/bash
./Carga_CPU_simulacion_Alta_1.sh
erika 7018 99.6 0.1 222520 3456 pts/0 R+ 16:21 0:40 /bin/bash
./Carga_CPU_simulacion_Alta.sh
[Mon Jan 20 04:21:47 PM EST 2025] Se ha detectado que el proceso con PID 6993 excede el umbral de uso
del CPU.
[Mon Jan 20 04:21:47 PM EST 2025] Eliminando proceso con PID 6993.
[Mon Jan 20 04:21:47 PM EST 2025] Se ha eliminado el proceso con PID 6993 satisfactoriamente.
erika 7018 99.8 0.1 222520 3456 pts/0 R+ 16:21 0:40 /bin/bash

```

Figura 3.10: Captura del log adjunto al correo.

Importancia del monitoreo de CPU en sistemas de telecomunicaciones

En las redes de telecomunicaciones es habitual que los nodos sean monitoreados constantemente para conocer el estado en condiciones de fallo y degradaciones en el servicio. Los equipos generan alarmas y eventos que son clasificados mayormente por su severidad (menores, mayores, críticas), las cuales suelen ser centralizadas por sistemas de gestión como los Element Management System (EMS) o NMS, dependiendo del proveedor de servicios [42].

El monitoreo y gestión genera una gran demanda de procesamiento de los recursos, entre ellos, el uso de CPU. Esto implica que un sistema procesa diversas alarmas en tiempo real y al monitorear el uso de CPU al mismo tiempo, se está monitoreando la capacidad de la red para detectar y reportar fallas de los nodos a tiempo.

Debido a la rapidez con la que se liberan nuevas versiones y configuraciones, las reglas de operación de los sistemas se modifican frecuentemente sin analizar completamente cómo esos cambios pueden afectar el desempeño general de la red o del sistema.

Es indispensable señalar que el uso de CPU en cada sistema dependerá del procesamiento de reglas, actualizaciones, alarmas y eventos de cada ope-

rador de telecomunicaciones. El umbral será proporcional a la arquitectura y plataformas de gestión existentes en la red. Por lo tanto, el monitoreo constante de CPU en un sistema de telecomunicaciones es fundamental, ya que es un elemento que permite que la red sea supervisada de forma continua y fiable [4].

3.2.3. Monitoreo y optimización de memoria RAM

El monitoreo de la memoria RAM es crucial para el funcionamiento eficiente de cualquier sistema, especialmente en el ámbito de las telecomunicaciones [6], donde los servicios deben mantenerse operativos de forma continua, las 24 horas del día, los 365 días del año. Un uso inadecuado o no controlado de la memoria RAM puede forzar al sistema a utilizar la memoria swap, es decir, extender la RAM hacia el disco duro. Esta práctica, conlleva una disminución considerable en el rendimiento del sistema, lo cual impacta directamente en aplicaciones sensibles como servidores de Voice over Internet Protocol (VoIP), conmutadores de llamadas de voz y la gestión en el tráfico de plano de usuario, control y datos.

Implementar un sistema de monitoreo de RAM permite identificar de manera proactiva problemas de uso de memoria antes de que afecten la estabilidad del sistema. Al prevenir el uso excesivo de RAM y la activación de la swap, se mejora el rendimiento general y se evitan interrupciones críticas en servicios de telecomunicaciones.

Además, el monitoreo constante facilita la detección de aplicaciones mal configuradas que consumen más memoria de la necesaria, permite realizar limpiezas periódicas de caché para prevenir sobrecargas, y ayuda a identificar posibles procesos maliciosos que consumen recursos de más, fortaleciendo de esta forma la seguridad del sistema.

El siguiente script, nos ayudará a monitorear y optimizar el uso de la memoria RAM en nuestro sistema.

Código 3.3: Script para monitoreo de uso de memoria RAM

```
#!/bin/bash

Log="/$HOME/Informacion_Uso_De_RAM.log"
Umbral=70
Correo="ejemplo_correo@gmail.com"
hora=$(date +" %H: %M: %S")
fecha=$(date +" %Y- %m- %d")
```

```
Umbral_Service=300

if [ -f "$Log" ]; then
    rm "$Log"
fi

correo() {
    Asunto="Warning: Uso de Memoria RAM excedido"

    Mensaje="$1"

    (echo -e "Subject: $Asunto\nMIME-Version: 1.0\nContent-Type: multipart
/mixed; boundary=frontier\n\n--frontier\nContent-Type: text/plain\n
\n$Mensaje.\n\n--frontier\nContent-Type: text/plain; name=
Uso_de_RAM.log\nContent-Transfer-Encoding: base64\nContent-
Disposition: attachment; filename=Informacion_Uso_De_RAM.log\n";
    base64 $Log) | msmtplib $Correo
}

Memoria=$(free | awk '/Mem/ {print int(($3/$2) * 100)}')

if [ "$Memoria" -gt "$Umbral" ]; then

    echo "$Memoria" >> "$Log"

    echo "El uso de la memoria ha rebasado el Umbral. Uso de memoria RAM:
$Memoria%" >> "$Log"

    echo "El uso de la memoria ha rebasado el Umbral. Uso de memoria RAM:
$Memoria%"

    echo "[$(date)] Iniciando lectura del estado de caché en el sistema $(
hostname)" >> "$Log"

    echo "[$(date)] Iniciando lectura del estado de caché en el sistema $(
hostname)"

    echo "[$(date)] Ejecutando free -h"

    echo "[$(date)] Ejecutando free -h" >> "$Log"

    free -h >> "$Log"

    echo "[$(date)] Limpiando memoria caché en el sistema $(hostname)" >> "
```

```
$Log"

echo "[$(date)] Limpiando memoria caché en el sistema $(hostname)"

sync

echo 1 | sudo tee /proc/sys/vm/drop_caches >> "$Log"

free -h >> "$Log"

echo "Memoria caché liberada a las $hora del dia $fecha" >> "$Log"

Servicios=("crond" "rsyslog" "cups" "postfix")

Servicios_reiniciados=()

for Servicio in "${Servicios[@]}; do

Uso_De_Memoria=$(ps -o rss= -C $Servicio | awk '{sum += $1} END {print sum
/ 1024}')

if (( $(echo "$Uso_De_Memoria > $Umbral_Service" | bc -l) )); then

echo "[$(date)] El proceso $Servicio está consumiendo ${Uso_De_Memoria}MB
de RAM." >> "$Log"

echo "[$(date)] El proceso $Servicio está consumiendo ${Uso_De_Memoria}MB
de RAM."

echo "[$(date)] Reiniciando proceso $Servicio" >> "$Log"

echo "[$(date)] Reiniciando proceso $Servicio"

systemctl restart $Servicio && Servicios_Reiniciados+=($Servicio) >> "$Log
"

echo "$Servicio reiniciado a las $(date)" >> "$Log"

echo "$Servicio reiniciado a las $(date)"

Memoria_Nueva=$(ps -o rss= -C $Servicio | awk '{sum += $1} END {print sum
/ 1024}')
```

```
echo "Se ha reiniciado satisfactoriamente. Uso de memoria en el $Servicio:
$Memoria_Nueva MB"
    fi
done

Memorial=$(free | awk '/Mem/ {print int(($3/$2) * 100)}')

correo "Estimado usuario, la utilización del la Memoria RAM alcanzó el
nivel máximo a las $hora con fecha $fecha en el sistema $(hostname).\n
\ Uso de Memoria alcanzado:$Memoria%\n\nSe realizó el reinicio de lo
siguiente:\n\n${Servicios_Reiniciados[@]}\n\n Uso de memoria RAM
actual:$Memorial& \n\nPara mayor información, consultar el log adjunto
a este correo"

echo "Se ha enviado un correo electrónico a $Correo con la información
detallada."

else

echo "[$(date)] No se ha detectado alta carga en el uso de la Memoria RAM.
" >> "$Log"

    echo "No se ha detectado alta carga en el uso de la Memoria RAM."
fi
```

Descripción del script

El script 3.3 está diseñado para realizar un monitoreo constante de la memoria RAM y tomar acciones en caso de que el uso supere un umbral predefinido. A continuación, se describen las variables clave utilizadas:

- **Log:** Archivo donde se guardará la información del estado de la memoria y las acciones realizadas.
- **Umbral:** Porcentaje máximo permitido del uso de memoria RAM. Se recomienda mantenerlo entre 70 % y 80 %.
- **Correo:** Dirección de correo electrónico a la que se enviará la alerta si se supera el umbral.
- **Hora:** Hora de ejecución del script.

- **Fecha:** Fecha de ejecución del script.
- **Umbral_Service:** Límite máximo de memoria (en MB) que puede utilizar un proceso antes de reiniciarlo.

Verificación de archivo de log

Se verifica si existe el archivo de log, y en caso afirmativo, se elimina y se vuelve a crear:

```
if [ -f "$Log" ]; then
    rm "$Log"
fi
```

Estructura del script

El script consta de cuatro partes fundamentales:

1. Verificación del estado de la memoria RAM
2. Limpieza de memoria caché
3. Reinicio de servicios
4. Envío de alertas por correo

Verificación del estado de la memoria RAM

Se utiliza el siguiente comando:

```
free | awk '/Mem/ {print int(($3/$2) * 100)}'
```

Donde:

- **free** muestra información de uso de la memoria.
- **awk** filtra la línea correspondiente a **Mem** y calcula el porcentaje de uso con la fórmula $(\$3/\$2)*100$.

Si el valor supera el umbral, se ejecutan tres acciones:

- Liberación de caché

- Reinicio de servicios que excedan el uso permitido
- Envío de alerta por correo electrónico

El comando de la figura 3.11 obtiene la información filtrada del porcentaje de memoria RAM, obteniendo un valor entero.

```
[root@centos ~]# free | awk '/Mem/ {print int(($3/$2) * 100)}'
```

43

```
[root@centos ~]#
```

Figura 3.11: Captura de la salida de comando `free | awk '/Mem/ print int((3/2) * 100)'`.

Si únicamente se hace uso del comando `free`, mostrará la información sobre el estado de memoria swap y RAM, total utilizada y disponible como se observa en la figura 3.12

```
[root@centos ~]# free
```

	total	used	free	shared	buff/cache	available
Mem:	1815924	791460	742212	11712	456272	1024464
Swap:	2097148	0	2097148			

```
[root@centos ~]# free -h
```

	total	used	free	shared	buff/cache	available
Mem:	1.7Gi	772Mi	724Mi	11Mi	445Mi	1.0Gi
Swap:	2.0Gi	0B	2.0Gi			

```
[root@centos ~]#
```

Figura 3.12: Captura de la salida de comando `free` y `free -h`.

El comando `free -h` mostrará la cantidad en formato legible para humanos (human readable).

Limpieza de memoria caché

En entornos de telecomunicaciones, mantener la memoria caché libre de datos innecesarios puede mejorar la capacidad de respuesta de sistemas que gestionan grandes volúmenes de tráfico en tiempo real [47]. El script realiza una limpieza de memoria caché utilizando el archivo `/proc/sys/vm/drop_caches`,

y se recomienda el valor **1** (liberación de caché de archivos) para evitar impactos negativos en el rendimiento.

El siguiente comando borra el caché de archivos en el sistema, liberando memoria que puede ser utilizada por aplicaciones de telecomunicaciones críticas.

```
echo 1 > /proc/sys/vm/drop_caches
```

A continuación, se muestra una tabla con los valores que se pueden escribir en el archivo `/proc/sys/vm/drop_caches` para liberar memoria caché en sistemas Linux:

Valor	Acción
1	Borra caché de archivos.
2	Borra estructuras del sistema de archivos.
3	Borra estructuras y caché (combinación de 1 y 2).

Cuadro 3.1: Opciones para liberar memoria caché en Linux.

Es importante mencionar que esto requiere permisos de súper usuario (`root`).

Reinicio de servicios

En sistemas de telecomunicaciones, algunos procesos pueden consumir una cantidad excesiva de memoria, lo que afecta la estabilidad y la capacidad de respuesta. El script monitoriza procesos individuales y, si alguno supera el `Umbral_Service` el proceso será reiniciado automáticamente para liberar recursos.

La siguiente parte del código, evaluará si algún servicio en ejecución ha excedido el `Umbral_Service` definido. La siguiente condición es declarada para definir el límite de uso de memoria.

```
echo "$Uso_De_Memoria > $Umbral_Service" | bc -l
```

Se monitoriza el uso de memoria de cada servicio con:

```
ps -o rss= -C <nombre_del_servicio> | awk '{sum += $1} END {print sum / 1024}'
```

Este comando es una combinación de `ps` y `awk`, donde `ps` servirá para buscar los procesos corriendo en el sistema y la opción `-C` ayudará a encontrarlo con el nombre indicado desde la columna `RSS`, que indicará la memoria de cada uno de los procesos.

A continuación, se desglosa con detalle el funcionamiento del comando:

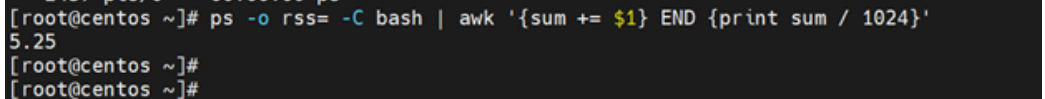
- `ps -o rss=`: Muestra solo la columna de memoria residente del proceso en cuestión (RSS).
- `-C <servicio>`: Filtra por nombre de proceso que se desea revisar.
- `sum += $1`: Suma todos los valores de la columna `RSS` si es que el servicio tiene múltiples instancias, de esta manera, sumará la memoria usada por todos los procesos corridos del servicio.
- `END {print sum / 1024}`: Convierte el total de la salida de la memoria `RSS` (memoria RAM del servicio) de KB a MB.

```
[root@centos ~]# ps aux | grep bash
root      2134  0.0  0.2 224156  5376 pts/0    Ss   15:34   0:00 -bash
root      2450  0.0  0.1 224156  3388 pts/0    D+   19:49   0:00 -bash
[root@centos ~]# ps -o rss= -C bash
5376
[root@centos ~]#
```

Figura 3.13: Captura de la salida del comando `ps aux | grep bash` y `ps -o rss= -C bash`.

La figura 3.13 da una idea de la diferencia del uso de `sum += $1` comparada con la salida simple del comando `ps` del proceso `bash`, en donde no hará un conteo total de los procesos totales que estén corriendo en el servicio, considerando únicamente de referencia el primer proceso del sistema.

La variable `Umbra1_Service` corresponde al valor en MB máximo alcanzado por cada servicio corriendo en el sistema de acuerdo con los requerimientos del usuario.



```
[root@centos ~]# ps -o rss= -C bash | awk '{sum += $1} END {print sum / 1024}'
5.25
[root@centos ~]#
[root@centos ~]#
```

Figura 3.14: Captura de la salida del comando utilizado para calcular el consumo total de memoria RAM de los procesos Bash.

En la figura 3.14 se busca un proceso con el nombre bash, y se imprimirá solo el valor de la memoria RAM utilizada convertida en MB.

En caso de que la condición se cumpla, entonces el script procederá a revisar proceso por proceso que esté corriendo en el sistema Linux, para posteriormente reiniciar el servicio que cumpla la condición con el siguiente comando:

```
systemctl restart <nombre_del_servicio> \&\& Servicios_Reiniciados+= <
nombre_del_servicio>
```

El comando utiliza la herramienta **systemctl** para reiniciar el servicio que excedió el uso de la memoria RAM, mientras que la variable **Servicios_Reiniciados**, se ejecuta únicamente si se ha realizado el reinicio del servicio, lo que implica añadir el valor de **<nombre_del_servicio>** a la variable **Servicios_Reiniciados**. Este proceso se repetirá con cada uno de los servicios que hayan excedido la memoria RAM del sistema.

Es importante enfatizar que, para el uso de la herramienta **systemctl** se requieren permisos de superusuario (**root**), por lo que deberá ejecutarse con este usuario o por medio de **texttsudo** ingresando las credenciales necesarias para la ejecución del comando. Posteriormente, se verificará el estado de la memoria RAM del sistema.

En entornos de telecomunicaciones, el reinicio de servicios en servidores de VoIP, bases de datos de tráfico o conmutadores telefónicos es una medida crítica cuando el consumo de memoria excede los límites establecidos, ya que contribuye a que el sistema recupere estabilidad en menor tiempo [2].

Envío de alerta

En caso de haber excedido el Umbral, un correo será enviado al usuario con un resumen de las acciones realizadas.

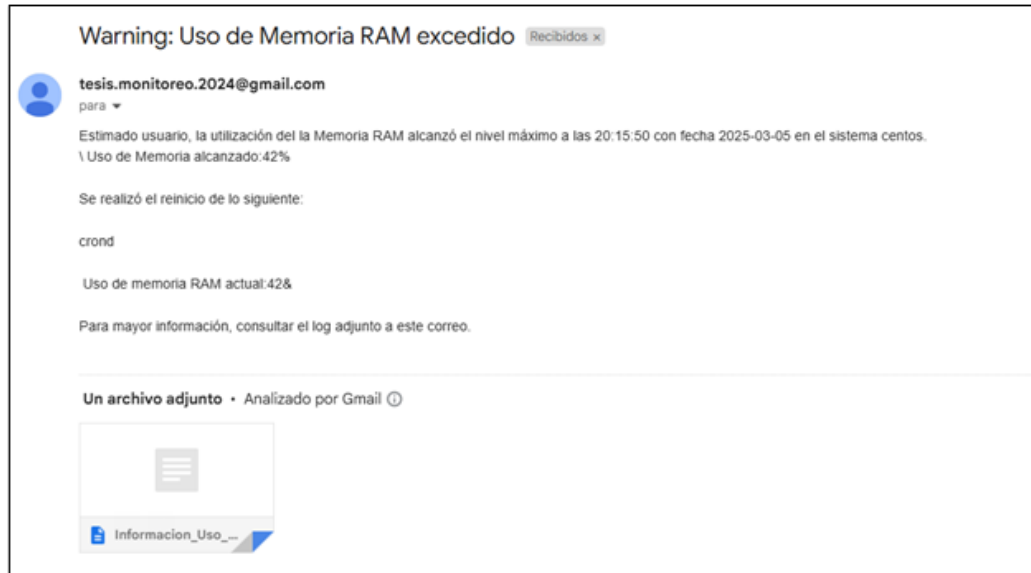


Figura 3.15: Captura del correo recibido después de exceder el límite de uso del Uso de la memoria RAM en el Sistema.

Es importante señalar que la figura 3.15 contiene el correo electrónico que se adapta a las condiciones específicas de cada sistema. En este caso, se trata de un entorno de laboratorio, por lo que las cifras y valores presentados no reflejan necesariamente las que se encontrarían en un entorno de producción.

En un entorno de prueba, el uso de memoria RAM puede no alcanzar el 70-80 %, ya que depende de la cantidad de procesos activos y archivos almacenados. Si no se ejecutan múltiples procesos que demanden recursos intensivos, el consumo de memoria será bajo. En este caso, se ha fijado un umbral de uso de RAM del 40 % y un límite de 1 MB por servicio. Sin embargo, estos valores varían según la configuración de cada sistema. Aunque este entorno de prueba no refleja completamente uno de producción, sí es posible que el uso de RAM alcance esos niveles si se simula una carga adecuada.

```

42
El uso de la memoria ha rebasado el Umbral. Uso de memoria RAM:42%
[Wed Mar  5 08:15:50 PM EST 2025] Iniciando lectura del estado de caché en el sistema centos
[Wed Mar  5 08:15:50 PM EST 2025] Ejecutando free -h
              total        used        free      shared  buff/cache   available
Mem:          1.7Gi         749Mi         931Mi         3.0Mi         255Mi         1.0Gi
Swap:          2.0Gi          13Mi          2.0Gi
[Wed Mar  5 08:15:50 PM EST 2025] Limpiando memoria caché en el sistema centos
1
              total        used        free      shared  buff/cache   available
Mem:          1.7Gi         746Mi         939Mi         3.0Mi         250Mi         1.0Gi
Swap:          2.0Gi          13Mi          2.0Gi
Memoria caché liberada a las 20:15:50 del día 2025-03-05
[Wed Mar  5 08:15:50 PM EST 2025] El proceso crond está consumiendo 3.625MB de RAM.
[Wed Mar  5 08:15:50 PM EST 2025] Reiniciando proceso crond
crond reiniciado a las Wed Mar  5 08:15:50 PM EST 2025

```

Figura 3.16: Captura del log adjunto al correo.

En la figura 3.16 se aprecia la información contenida en el archivo de registro (log) generado a partir de las notificaciones y acciones automáticas generadas por el script de monitoreo de memoria RAM. En el registro, se observan las acciones realizadas por el sistema, incluyendo la visualización del uso de memoria, la liberación de memoria caché cuando se alcanzan los umbrales definidos, así como el reinicio de servicios de alto consumo.

Aplicación en sistemas de telecomunicaciones

La memoria RAM es considerada un componente crítico del sistema, debido a que su funcionamiento afecta al desempeño de los servicios y aplicaciones. El monitoreo contante de la RAM, permite detectar fallas a tiempo para evitar latencia en los servicios, degradación de rendimiento, pérdida de sesiones o conexiones, así como reinicio repentino en el sistema.

Algunos sistemas o nodos en las telecomunicaciones son sensibles al control de memoria, cuando no se cuenta con suficientes recursos, el sistema recurre a balancear la carga con los recursos disponibles como lo es el uso de memoria SWAP lo que puede provocar una disminución notoria en el rendimiento. Por ello, es importante detectar a tiempo la disponibilidad de memoria, antes de que el sistema comience a degradarse.

3.3. Automatización de tareas de monitoreo mediante cron

El uso de **cron** juega un papel crucial en la automatización de los procesos del sistema, ya que evita la intervención manual por parte del usuario. Esta herramienta facilita la programación de scripts que supervisan diversos aspectos del sistema, tales como el uso de la red, la carga de CPU y la gestión de la memoria RAM. En el ámbito de las telecomunicaciones, donde la alta disponibilidad y el rendimiento óptimo son esenciales, este enfoque asegura que los componentes críticos del sistema, como servidores de VoIP, conmutadores y demás dispositivos clave de la infraestructura de red, sean continuamente monitoreados. Esta vigilancia constante permite la detección anticipada de posibles fallos, contribuye a la mejora de la calidad del servicio (QoS) y garantiza el funcionamiento eficiente de sistemas de alto rendimiento que deben manejar grandes volúmenes de datos y comunicaciones en tiempo real.

3.3.1. Configuración de crontab para ejecución automática

Para integrar los scripts con la herramienta **cron**, es necesario editar el archivo **crontab**. Este archivo debe contener las líneas de comandos que se ejecutarán, junto con el intervalo de tiempo requerido para cada uno. No es obligatorio utilizar el mismo intervalo de tiempo para todos los scripts; este dependerá completamente de las necesidades del usuario, basándose en sus requerimientos específicos. De esta manera, se podrá automatizar la ejecución periódica de mantenimientos y monitoreos en el sistema, adaptándolos al desempeño y uso que se le dé al mismo. El intervalo variará según las aplicaciones que se ejecuten, el uso del sistema, el almacenamiento disponible, entre otros factores [11].

Los pasos para integrar la ejecución automática de los monitoreos en el sistema mediante **cron** son los siguientes:

- Identificar si los comandos deben ejecutarse para todos los usuarios existentes en el sistema o solo para un usuario específico.

Ejecución para solo un usuario

- **Ejecutar el comando `crontab -e` (figura 3.17):** Este comando permitirá editar el archivo `crontab` para agregar las líneas correspondientes a las tareas que se desean programar. Una vez ejecutado el comando `crontab -e`, se deberá insertar la línea correspondiente a la tarea que se desea programar, utilizando la siguiente estructura:

<minuto><hora><día del mes><mes><día de la semana><comando o script>

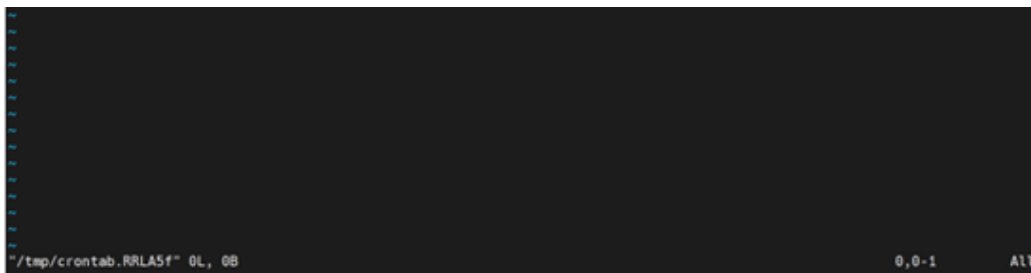


Figura 3.17: Captura de la salida de comando `crontab -e` en el sistema.

- **Ejecutar el comando `crontab -l`:** Este comando permite verificar que la tarea programada fue añadida correctamente al archivo `crontab`.

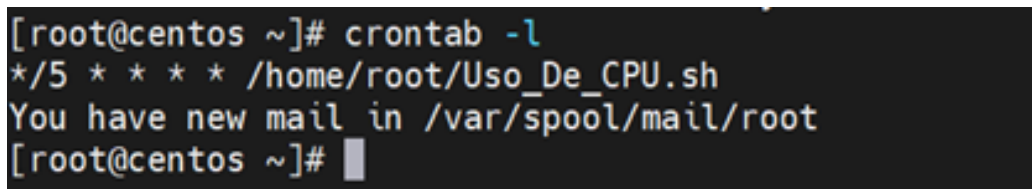


Figura 3.18: Captura de la salida de comando `crontab -l`.

En la figura 3.18, se puede observar que la línea fue añadida correctamente al archivo `crontab`, lo que confirma que la tarea programada se ha registrado de manera satisfactoria.

Ejecución para todos los usuarios

- **Ejecución de tareas a nivel de sistema:** Para programar tareas que afecten a todo el sistema y no solo a un usuario específico, es necesario editar el archivo global de `cron` en lugar de utilizar únicamente el comando `crontab -e`. En la mayoría de las distribuciones de Linux, dicho archivo se encuentra en la siguiente ruta:

```
/etc/crontab
```

Este archivo permite especificar tareas programadas que serán ejecutadas por el sistema independientemente del usuario actual.

- **Tareas programadas por intervalos predefinidos:** Para automatizar tareas que deben ejecutarse en intervalos de tiempo específicos (por ejemplo, diariamente, por hora, semanal o mensualmente), existen directorios dedicados ubicados generalmente en `/etc/cron`. Estos directorios incluyen:

- `/etc/cron.hourly`
- `/etc/cron.daily`
- `/etc/cron.weekly`
- `/etc/cron.monthly`

Cada uno de estos directorios contiene scripts que se ejecutan automáticamente en el intervalo correspondiente, lo cual facilita la administración periódica del sistema. En la figura 3.19 se filtran los archivos y directorios `cron` existentes en el sistema.

```
[root@centos etc]# ll /etc/ | grep cron
-rw-r--r--. 1 root root 541 Dec 11 08:34 anacrontab
drwxr-xr-x. 2 root root 21 Dec 11 08:34 cron.d
drwxr-xr-x. 2 root root 6 Aug 9 2021 cron.daily
-rw-r--r--. 1 root root 0 Dec 11 08:34 cron.deny
drwxr-xr-x. 2 root root 22 Jan 17 13:32 cron.hourly
drwxr-xr-x. 2 root root 6 Aug 9 2021 cron.monthly
-rw-r--r--. 1 root root 451 Aug 9 2021 crontab
drwxr-xr-x. 2 root root 6 Aug 9 2021 cron.weekly
```

Figura 3.19: Captura de la salida de comando `ll /etc/ | grep cron`.

```
[root@centos etc]# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
*/5 * * * * /home/root/Usa_De_CPU.sh
[root@centos etc]#
```

Figura 3.20: Captura de la salida de comando `cat /etc/crontab`.

Como se muestra en la figura 3.20, se añadió la siguiente línea al archivo `crontab`:

```
*/5 * * * * /home/root/Usa_De_CPU.sh
```

Esta línea configura la ejecución automática del script `Usa_De_CPU.sh` cada cinco minutos, aplicándose a todos los usuarios del sistema si es agregada al archivo global `/etc/crontab`.

Para la edición del archivo `crontab`, se utilizan los siguientes atajos de teclado comunes dentro del editor de texto por defecto (generalmente `vi` o `nano`, dependiendo de la configuración del sistema):

- **CTRL + I**: Insertar texto o ingresar al modo de edición.
- **CTRL + C**: Salir del modo de escritura.
- **:wq!:** Guardar los cambios realizados y salir del editor.

3.3.2. Verificación del funcionamiento de la automatización

Para verificar que el servicio `cron` está funcionando correctamente y que las tareas programadas (scripts) se están ejecutando según lo establecido, es posible consultar los registros del sistema ubicados en:

```
grep CRON /var/log/cron
```

Este comando filtra las entradas del archivo de log relacionadas con `cron`, permitiendo confirmar si las tareas han sido lanzadas en los tiempos definidos.

La revisión puede realizarse de forma manual o mediante un script en Bash. Para ello, basta con guardar el comando anterior en un archivo con extensión `.sh` y ejecutarlo cada vez que se desee verificar la actividad del programador de tareas. En la figura 3.21 se visualizan los eventos generados por el servicio `cron`

```
[root@centos ~]# grep CRON /var/log/cron
Mar  7 04:40:18 centos CROND[2695]: (root) CMD (/home/root/Usa_De_CPU.sh)
Mar  7 10:26:56 centos CROND[2662]: (root) CMDEND (/home/root/Usa_De_CPU.sh)
Mar  7 10:30:01 centos CROND[2743]: (root) CMD (/home/root/Usa_De_CPU.sh)
Mar  7 10:31:02 centos CROND[2740]: (root) CMDEND (/home/root/Usa_De_CPU.sh)
Mar  7 10:35:01 centos CROND[2754]: (root) CMD (/home/root/Usa_De_CPU.sh)
Mar  7 10:36:01 centos CROND[2752]: (root) CMDEND (/home/root/Usa_De_CPU.sh)
Mar  7 10:40:02 centos CROND[2790]: (root) CMD (/home/root/Usa_De_CPU.sh)
Mar  7 10:41:02 centos CROND[2767]: (root) CMDEND (/home/root/Usa_De_CPU.sh)
Mar  7 10:45:01 centos CROND[2816]: (root) CMD (/home/root/Usa_De_CPU.sh)
Mar  7 10:46:02 centos CROND[2797]: (root) CMDEND (/home/root/Usa_De_CPU.sh)
Mar  7 10:50:01 centos CROND[2847]: (root) CMD (/home/root/Usa_De_CPU.sh)
Mar  7 10:51:01 centos CROND[2825]: (root) CMDEND (/home/root/Usa_De_CPU.sh)
Mar  7 10:55:02 centos CROND[2874]: (root) CMD (/home/root/Usa_De_CPU.sh)
Mar  7 10:56:02 centos CROND[2855]: (root) CMDEND (/home/root/Usa_De_CPU.sh)
Mar  7 11:00:02 centos CROND[2908]: (root) CMD (/home/root/Usa_De_CPU.sh)
Mar  7 11:01:01 centos CROND[2912]: (root) CMD (run-parts /etc/cron.hourly)
Mar  7 11:01:01 centos CROND[2911]: (root) CMDEND (run-parts /etc/cron.hourly)
Mar  7 11:01:02 centos CROND[2886]: (root) CMDEND (/home/root/Usa_De_CPU.sh)
Mar  7 11:05:02 centos CROND[2953]: (root) CMD (/home/root/Usa_De_CPU.sh)
```

Figura 3.21: Captura de la salida de comando `grep CRON /var/log/cron`.

3.3.3. Integración de los monitoreos automatizados con `cron`

El directorio mayormente utilizado para almacenar scripts en un sistema es `/bin`, cuando lo único que se desea estén disponibles para un solo usuario. Sin embargo, si se pretende que los scripts sean ejecutados por el usuario `root` o por todos los usuarios del sistema, es recomendable almacenarlos en la ruta: `/usr/local/bin`.

El directorio `/bin` del sistema Linux, forma parte del estándar *Filesystem Hierarchy Standard* (FHS), que clasifica este directorio para almacenar programas y scripts que no forman parte del sistema base, pero que deben ser accesibles para todos los usuarios sin ningún riesgo de seguridad [14]. Esto asegura que los scripts sean fácilmente ejecutables desde cualquier terminal,

sin necesidad de especificar la ruta completa, ya que el sistema busca por defecto en este directorio.

Cuando un script es ubicado en `/bin`, se puede ejecutar simplemente utilizando su nombre, sin tener que escribir la ruta completa del archivo como se muestra en la figura 3.22

```
[root@centos ~]# pwd
/root
[root@centos ~]# ls -lhtr /usr/local/bin/
total 12K
-rwxr-xr-x. 1 root root 2.3K Mar 18 17:59 Uso_De_CPU.sh
-rwxr-xr-x. 1 root root 3.5K Mar 18 17:59 Uso_de_RAM.sh
-rwxr-xr-x. 1 root root 949 Mar 18 17:59 Uso_de_Red.sh
[root@centos ~]#
[root@centos ~]# ./Uso_De_CPU.sh
No se ha detectado alta carga en el uso del CPU.
[root@centos ~]#
```

Figura 3.22: Captura de la salida de ejecución de script bash sin indicar la ruta de la ubicación.

En caso de que los scripts de monitoreo del uso de RAM, CPU y red se encuentren almacenados en ubicaciones distintas dentro del sistema de archivos o bien en el directorio `/bin`, será necesario especificar la ruta completa de cada script en el archivo `crontab`, junto con su nombre exacto. Esto garantiza que el sistema pueda ejecutar los scripts correctamente sin importar su ubicación.

La frecuencia con la que se monitorean los recursos del sistema, como la RAM, la CPU y la red, dependerá de la carga y los requisitos específicos del sistema. En sistemas de alto rendimiento, es importante tener en cuenta que la ejecución continua de estos scripts puede generar una sobrecarga innecesaria, por lo que se recomienda evaluar cuidadosamente la carga del sistema antes de establecer una frecuencia demasiado alta. Sin embargo, como se mencionó previamente, la frecuencia dependerá del uso específico que se le dé al sistema [41].

A continuación, se presenta una frecuencia general recomendada para la ejecución de cada uno de los scripts de monitoreo:

- **Uso de CPU:** Se recomienda un intervalo de entre 15 a 20 minutos si

se desea un monitoreo detallado y continuo, pero evitando sobrecargar el sistema.

- **Uso de Red:** Un intervalo de entre 5 a 10 minutos, dado que el tráfico de red puede variar repentinamente y requiere un monitoreo más frecuente.
- **Uso de RAM:** Se sugiere un intervalo de entre 30 minutos a 1 hora, ya que el uso de la RAM generalmente es más estable y gradual en comparación con la CPU o la red.

```
[root@centos ~]# crontab -l
*/15 * * * * /usr/local/bin/Uso_De_CPU.sh
*/10 * * * * /usr/local/bin/Uso_de_Red.sh
*/30 * * * * /usr/local/bin/Uso_de_RAM.sh
[root@centos ~]#
```

Figura 3.23: Captura de la salida de ejecución de script bash sin indicar la ruta de la ubicación.

En el caso del laboratorio, se establece un monitoreo automatizado de 15 minutos para el uso del CPU, 10 minutos para el uso de red y, finalmente, 30 minutos para el seguimiento de uso de memoria RAM, como se muestra a la salida del comando `crontab -l` (figura 3.23).

Los tiempos de supervisión pueden variar; sin embargo, se recomienda evitar un monitoreo excesivamente intensivo, ya que podría afectar el rendimiento del sistema. El umbral es un valor que desempeña un papel muy importante, ya que evita la generación de alertas o notificaciones masivas, así como la ejecución continua de tareas innecesarias realizadas por el script.

Impacto del monitoreo automatizado en sistemas de telecomunicaciones

Las herramientas de monitorización en una red son indispensables para las telecomunicaciones, porque su función principal es la búsqueda de componentes internos defectuosos y problemáticos, que avisen a los administradores de la red de eventos ocurridos mediante la generación de alarmas con el fin

de realizar las correcciones a tiempo y evitar el mal comportamiento e interrupción del servicio.

Los sistemas de telecomunicaciones poseen una arquitectura compuesta por servidores, nodos, aplicaciones y servicios que operan de forma ininterrumpible en la red [42]. La supervisión de los recursos permite identificar los problemas y fallos a tiempo y evitar comprometer la estabilidad del sistema.

El monitoreo automatizado facilita la gestión de los sistemas de telecomunicaciones y permite la continuidad de los servicios al proporcionar información detallada sobre el comportamiento de los recursos críticos del sistema.

Capítulo 4

Conclusiones

En esta tesis se desarrolló un sistema automatizado para el monitoreo de recursos críticos en un entorno Linux de alto rendimiento orientado a sistemas de telecomunicaciones, enfocándose en el uso de memoria RAM, red y CPU mediante scripts personalizados en Bash. Esta implementación permite mantener entornos Linux en condiciones óptimas, identificando problemas de forma oportuna a través de revisiones periódicas y notificando al usuario para realizar las acciones correctivas y preventivas necesarias.

Se demostró que Bash scripting es una herramienta eficaz para este tipo de tareas, al aprovechar comandos nativos del sistema como ps, top, mpstat y free, lo que permite un monitoreo eficiente y de bajo consumo de recursos. Esta característica resulta especialmente valiosa en sistemas de telecomunicaciones, donde el rendimiento y la disponibilidad son esenciales para garantizar la calidad del servicio.

La integración de los scripts con la herramienta msmtplib permitió enviar notificaciones automáticas al usuario cuando se detectaban valores que superaban los umbrales establecidos, fortaleciendo la capacidad de respuesta ante incidentes. Además, el uso de cron posibilitó la ejecución periódica de los scripts, garantizando un monitoreo constante sin intervención manual.

Durante el desarrollo, una de las principales limitaciones fue la necesidad de definir umbrales de alerta más dinámicos y ajustables, ya que los valores fijos no siempre reflejan adecuadamente el comportamiento real del sistema. Adicionalmente, se recurrió a la simulación de cargas de trabajo para validar la efectividad del monitoreo en un entorno controlado de laboratorio.

Además, se cumplió con el objetivo de diseñar y desarrollar procesos automatizados para el monitoreo de memoria RAM, CPU y uso de red mediante

Bash scripting, integrando herramientas de programación de tareas y notificación que permiten supervisar continuamente el estado del sistema, para permitir el funcionamiento óptimo de los servicios.

Para investigaciones futuras, se sugiere la integración de herramientas más avanzadas como Prometheus o Grafana, que permiten la visualización en tiempo real y la generación de reportes detallados, facilitando la supervisión y el análisis a gran escala. Esta mejora sería especialmente útil en infraestructuras de telecomunicaciones que requieren una vigilancia continua y detallada del rendimiento.

En conclusión, el sistema desarrollado representa una solución práctica y adaptable para la automatización del monitoreo de recursos en entornos Linux de alto rendimiento. La reducción de tareas manuales, la prevención de fallos y la mejora en la eficiencia operativa lo convierten en una herramienta de gran valor para sistemas donde la estabilidad, disponibilidad y rendimiento son críticos, como es el caso en las telecomunicaciones.

Bibliografía

- [1] A. Oodan, *Telecommunications Quality of Service Management*. Institution of Engineering and Technology, 2003.
- [2] P. Bellavista, *Telecommunication Systems and Technologies*. EOLSS Publishers Company Limited, 2009.
- [3] J. E. Flood, *Telecommunication Networks*. Institution of Electrical Engineers, 1997.
- [4] H.-K. Kahng and S. Goto, Eds., *Information Networking: Networking Technologies for Broadband and Mobile Networks, International Conference ICOIN 2004, Busan, Korea, February 18–20, 2004, Revised Selected Papers*. Berlin, Germany: Springer, 2004.
- [5] B. S. Dhillon, *System Safety, Maintainability, and Maintenance for Engineers*. CRC Press, 2023.
- [6] K. Terplan and P. A. Morreale, Eds., *The Telecommunications Handbook*. CRC Press, 2018.
- [7] H. H. Wang, *Telecommunications Network Management*. McGraw-Hill, 1999.
- [8] J. G. Voinea, *Redes de Comunicaciones: Administración y Gestión*. Lulu.com, 2012.
- [9] C. Negus, *Linux Bible: The Comprehensive, Tutorial Resource*. Wiley, 2015.
- [10] N. Pons, *Linux: Principios básicos de uso del sistema*. Ediciones ENI, 2016.

- [11] W. E. Shotts Jr., *The Linux Command Line: A Complete Introduction*. No Starch Press, 2012.
- [12] C. Bresnahan and R. Blum, *LPI Linux Essentials Study Guide*. John Wiley & Sons, 2020.
- [13] Noite.pl, *The Structure of the File and Directory System: Linux Basic*. NOITE S.C.
- [14] Noite.pl, *The Basics of the Linux System Structure: Linux Basic. AL1-002*. NOITE S.C.
- [15] R. Blum, *CompTIA Linux+ Study Guide: Exam XK0-006*. John Wiley & Sons, 2025.
- [16] R. Blum and C. Bresnahan, *Linux Command Line and Shell Scripting Bible*. Wiley, 2008.
- [17] M. Liyanage, A. Gurtov, and M. Ylianttila, *Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture*. John Wiley & Sons, 2015.
- [18] A. R. Figueiras Vidal and A. Artés Rodríguez, *Una panorámica de las telecomunicaciones*. Pearson Educación, 2002.
- [19] M. N. Ellanti, S. S. Gorshe, L. G. Raman and W. D. Grover, *Next Generation Transport Networks: Data, Management, and Control Planes*. Springer, 2005.
- [20] A. Calcatinge and J. Balog, *Mastering Linux Administration*. Packt Publishing, 2021.
- [21] T. Plevyak and V. Sahin, *Next Generation Telecommunications Networks, Services, and Management*. Wiley, 2011.
- [22] P. Olushile, *Essential Linux Commands: 100 Linux Commands Every System Administrator Should Know*. Packt Publishing, 2023.
- [23] R. W. Smith, *Linux Samba Server Administration*. Wiley, 2006.
- [24] The Linux Foundation, *Essentials of Linux System Administration (LFS201)*. [En línea]. Disponible en: <https://trainingportal.linuxfoundation.org/learn> [Accedido: Aug. 13, 2024].

-
- [25] A. Mallett, *CentOS System Administration Essentials*. Packt Publishing, 2014.
 - [26] C. Sicam, R. Baclit, P. Membrey, and J. Newbigin, *Foundations of CentOS Linux: Enterprise Linux On the Cheap*. New York, NY, USA: Apress, 2009.
 - [27] A. Frisch, *Essential System Administration*. O'Reilly & Associates, 1995.
 - [28] M. Bar, *Linux File Systems*. Osborne/McGraw-Hill, 2001.
 - [29] B. Costales, E. Allman and N. Rickert, *Sendmail*. O'Reilly & Associates, 1993.
 - [30] M. Pérez Colino, P. Iranzo Gómez and S. McCarty, *Red Hat Enterprise Linux 8 Administration*. Packt Publishing, 2021.
 - [31] Z. Yap, *RHEL Troubleshooting for System Administrators*. Decoded Brand, 2026.
 - [32] R. Fox, *Linux with Operating System Concepts*. CRC Press, 2021.
 - [33] D. Liu, S. Miller, M. Lucas, A. Singh and J. Davis, *Firewall Policies and VPN Configurations*. Syngress, 2006.
 - [34] R. Botwright, *Service Desk Analyst Bootcamp: Maintaining, Configuring and Installing Hardware and Software*. Rob Botwright, 2024.
 - [35] W. R. Bomfim, *Ingeniería Completa de Redes Móviles: Optimización de Redes Móviles, Dominando las Redes 5G e Implementación de Redes Celulares*. Wagner Romano Bomfim, 2026.
 - [36] K. Misra, *OSS for Telecom Networks: An Introduction to Network Management*. Springer London, 2012.
 - [37] C. Newham and B. Rosenblatt, *Learning the Bash Shell: Unix Shell Programming*. O'Reilly Media, 2005.
 - [38] S. Kumari, *Linux Shell Scripting Essentials*. Packt Publishing, 2015.
 - [39] M. Tobler, *Inside Linux*. New Riders, 2001.

-
- [40] M. H. Jang, *Linux Transfer for Windows Network Admins: A Roadmap for Building a Linux File and Print Server*. Hentzenwerke Publishing, 2003.
 - [41] International Engineering Consortium, Ed., *Operations Support Systems 2002: Enabling the Next Generation Network*. International Engineering Consortium, 2002.
 - [42] N. J. Muller, *Desktop Encyclopedia of Telecommunications*. McGraw-Hill, 2002.
 - [43] T. Anttalainen, *Introduction to Telecommunications Network Engineering*. Artech House, 2003.
 - [44] L. D. Xu, *Enterprise Integration and Information Architecture: A Systems Perspective on Industrial Information Integration*. CRC Press, 2014.
 - [45] J. Andreu Gómez, *Gestión de servicios de correo electrónico (Servicios en red)*. Editorial Editex, 2011.
 - [46] T. S. Randhawa and S. Hardy, *Network Management in Wired and Wireless Networks*. Springer US, 2013.
 - [47] K. Terplan and P. A. Morreale, Eds., *CRC Handbook of Modern Telecommunications*. Taylor & Francis, 2018.