

**Configuración de Servicios DNS y Automatización de Tareas Mediante Scripting en
Entornos Virtualizados**

Santiago Botero García

Laura Natalia Perilla Quintero

Escuela Colombiana de Ingeniería Julio Garavito

AYSR-1L: Arquitectura y Servicios de Red

Ing. Jhon Alexander Pachón Pinzón

Septiembre 20, 2025

Resumen

La presente actividad aborda la implementación de servicios de red en la capa de aplicación, específicamente la instalación y configuración de servidores DNS (Domain Name System) mediante el software BIND en sistemas operativos Unix-like y Windows Server, operando sobre un entorno de virtualización. Se diseñan y despliegan zonas de nombres para dominios personalizados en un entorno simulado de infraestructura empresarial, contemplando servidores primarios y secundarios, tanto para direcciones IPv4 como IPv6. Cada dominio configurado incluye registros tipo A (dirección IPv4), AAAA (dirección IPv6), CNAME (nombre canónico), MX (correo) y NS (servidores de nombres), los cuales permiten gestionar y resolver nombres internos y externos dentro de una red interconectada.

El entorno experimental se compone de diversas máquinas virtuales, incluyendo instancias de Solaris, Slackware y Windows Server, distribuidas en dos nodos físicos. La instalación del servicio DNS considera la configuración de archivos como `named.conf`, archivos de zona directa y archivos SOA (Start of Authority), así como la modificación del cliente DNS en otros servidores virtuales para validar la funcionalidad del servicio mediante comandos como `nslookup`, `dig` y configuración estática de resolvers. Se verifica la correcta resolución de dominios locales y externos, evaluando también el comportamiento ante consultas específicas utilizando opciones avanzadas del comando `nslookup`, como `set type` y `set debug`.

Paralelamente, se desarrollan scripts en Shell y PowerShell para automatizar procesos administrativos en los servidores mencionados. En los sistemas Solaris y Slackware se construyen herramientas que permiten programar tareas periódicas a través de la línea de comandos, sin interacción manual, facilitando su integración en tareas automatizadas del sistema. Asimismo, se implementa un menú interactivo para el monitoreo y gestión de procesos en

ejecución, que incluye funcionalidades como visualización del uso de recursos, búsqueda por nombre, finalización y reinicio de procesos específicos. Además, se crea un script que permite recorrer recursivamente el sistema de archivos desde una ruta dada, identificar archivos con restricciones de tamaño, y mostrar sus propiedades principales, como nombre, ubicación y tamaño total.

Palabras clave: DNS, Bind, Linux, Windows Server, Shell Script, Powershell, Virtualización, Automatización, Protocolos de Red, Administración de Sistemas

Contenido

Resumen.....	2
Metodología	5
Servidor DNS de Linux – BIND	5
<i>Configuración en Solaris</i>	<i>5</i>
<i>Configuración en Slackware</i>	<i>12</i>
<i>Configuración Windows Server</i>	<i>16</i>
Otros comandos útiles.....	23
<i>Configuración de Tareas Programadas en Sistemas UNIX.....</i>	<i>23</i>
<i>Script de Gestión de Procesos en Sistemas UNIX</i>	<i>31</i>
<i>Listado de Archivos Más Pequeños en Sistemas UNIX</i>	<i>41</i>
Resultados	48
Servidor DNS de Linux – BIND	48
Otros comandos útiles.....	55
Conclusiones	58
Bibliografía	59

Metodología

Servidor DNS de Linux – BIND

Configuración en Solaris

Como primer paso, se verificó si el servicio **BIND** ya se encontraba instalado en la máquina Solaris. Para ello, se utilizó el comando `pkg search`, el cual confirmó la presencia del software en el sistema. En caso de que **BIND** no estuviera instalado, el procedimiento correcto consistiría en ejecutar `pkg install bind` con privilegios de administrador. Una vez asegurada su instalación, se procedió a validar el estado del servicio utilizando el comando `svcs bind`, confirmando que el demonio del servidor de nombres se encontraba activo y en ejecución.

Figura 1

Verificación del servicio BIND en Solaris

```
root@solaris:~# svcs bind
STATE      STIME      FMRI
online     5:56:48    svc:/network/rpc/bind:default
root@solaris:~#
```

Nota. Muestra el resultado del comando `svcs bind`, confirmando que el servicio **BIND** está instalado y activo en el sistema Solaris. Este paso es esencial para asegurar que el servidor DNS puede operar correctamente.

El archivo principal de configuración de **BIND**, denominado `named.conf`, se localiza en la ruta `/etc/named.conf`. En dicho archivo se establecen las directrices generales de operación del servidor DNS, así como las zonas de autoridad que gestionará. La configuración aplicada definió parámetros clave como el directorio donde se almacenan los archivos de zona, las interfaces de red sobre las cuales el servicio escucha peticiones, y las políticas de acceso a las consultas.

Asimismo, se declaró la zona maestra correspondiente al dominio **santiago.com.it**, junto con zonas esclavas adicionales que serán sincronizadas desde servidores remotos.

Figura 2

Contenido del archivo de configuración

```
options {
    directory "/etc";
    listen-on { any; };
    allow-query { any; };
    forwarders {
        8.8.8.8;
        8.8.4.4;
    };
    recursion yes;
};

zone "." IN {
    type hint;
    file "/etc/named.root";
};

# Zone for santiago.com.it (primary dns)
zone "santiago.com.it" IN {
    type master;
    file "/etc/named.santiago.com.it";
};

# Zone for natalia.org.uk
zone "natalia.org.uk" IN {
    type slave;
    masters { 10.2.77.182; };
    file "/etc/named.natalia.org.uk";
};
```

[28 líneas leídas]

Ver ayuda Guardar B
 Salir Leer fich. Reemplazar Pegar txt Ortografía Ir a línea

Nota. Ilustra la estructura del archivo named.conf, donde se definen las zonas DNS, interfaces de red, políticas de acceso y parámetros generales del servicio. Es el núcleo de la configuración de BIND.

Con el fin de habilitar la capacidad de resolver dominios externos, se creó el archivo named.root en la ubicación /etc/, el cual contiene un conjunto mínimo de servidores raíz que actúan como referencia inicial para las consultas DNS fuera de las zonas locales. Aunque en

entornos de producción este archivo suele descargarse desde fuentes oficiales como IANA, en este caso se definió manualmente un conjunto reducido de entradas válidas para fines de prueba.

Figura 3

Archivo raíz de resolución externa /etc/named.root

```
.           3600000   IN    NS     A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET. 3600000   IN    A      198.41.0.4
B.ROOT-SERVERS.NET. 3600000   IN    A      192.228.79.201
```

Nota. Representa el archivo named.root, que contiene referencias a servidores raíz para permitir la resolución de dominios fuera de las zonas locales. Aunque simplificado para pruebas, cumple una función crítica en entornos reales.

La autoridad sobre el dominio **santiago.com.it** se configuró mediante la creación de un archivo de zona específico: /etc/named.santiago.com.it. Este archivo contiene registros tipo **SOA** (Start of Authority), que definen el servidor DNS principal y los parámetros de control de zona, tales como el número de serie, los tiempos de actualización, y los contactos administrativos. Adicionalmente, se incluyeron registros tipo **A**, **AAAA**, **CNAME** y otros necesarios para garantizar la correcta resolución de subdominios internos, como server1, server2 y www.

Figura 4

Zona maestra del dominio santiago.com.it

```
$TTL 86400
@ IN SOA dns.santiago.com.it. root.santiago.com.it. (
    2025091601 ; Serial
    3600       ; Refresh
    1800       ; Retry
    1209600    ; Expire
    86400      ; Minimum TTL

    IN NS      dns.santiago.com.it.

; Servidor DNS
dns IN A       192.168.1.10

; Servidores con IPv4
server1 IN A   192.168.1.11
server2 IN A   192.168.1.12
server3 IN A   192.168.1.13

; Servidores con IPv6
ipv6srv1 IN AAAA 2001:db8::11
ipv6srv2 IN AAAA 2001:db8::12

; Alias
alias1 IN CNAME server1.santiago.com.it.
alias2 IN CNAME server2.santiago.com.it.
aliasv6 IN CNAME ipv6srv1.santiago.com.it.

; Dominio externo
www IN A       203.0.113.10
^G Ver ayuda ^O Guardar ^W B
^X Salir ^R Leer fich. ^N Reemplazar ^U Pegar txt ^I OrtografÃ-a ^_ Ir a lÃ-nea
```

Nota. Muestra el archivo de zona /etc/named.santiago.com.it, incluyendo registros SOA, A, AAAA y CNAME. Este archivo define la autoridad sobre el dominio y permite la resolución interna de nombres.

Para comprobar que el servidor DNS se encontraba en funcionamiento y aceptando consultas en el puerto 53, se emplearon herramientas como dig, ejecutadas directamente desde la máquina Solaris. Las pruebas arrojaron resultados positivos, evidenciando la correcta traducción de nombres definidos en la zona configurada. La disponibilidad del puerto 53 fue verificada mediante comandos de diagnóstico de red, lo que confirmó que el servicio se encontraba escuchando activamente.

Figura 5

Verificación del puerto 53 en escucha

```

root@solaris:~# /usr/sbin/named
root@solaris:~# netstat -an | grep 53
    *.53                                Idle                                57344
0      57344                                0
127.0.0.1.53                            Idle                                57344
0      57344                                0
10.2.77.180.53                          Idle                                57344
0      57344                                0
    *.53                                Idle
    57344                                0      57344                                0
127.0.0.1.53                            *. *                                0      0      256000      0 LISTEN
10.2.77.180.53                          *. *                                0      0      256000      0 LISTEN
    *.53                                *. *                                0
    0 256000      0 LISTEN
root@solaris:~# Sep 16 07:35:09 solaris named[1326]: transfer of 'natalia.org.uk
/IN' from 10.2.77.182#53: failed to connect: connection refused

```

Nota. Evidencia que el servicio BIND está escuchando en el puerto 53, el estándar para consultas DNS. Este diagnóstico confirma que el servidor está listo para recibir peticiones.

Figura 6*Prueba de resolución local con dig*

```

root@solaris:~# dig alias1.santiago.com.it @localhost

; <<>> DiG 9.10.6-P1 <<>> alias1.santiago.com.it @localhost
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21642
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;alias1.santiago.com.it.                IN      A

;; ANSWER SECTION:
alias1.santiago.com.it. 86400   IN      CNAME   server1.santiago.com.it.
server1.santiago.com.it. 86400   IN      A       192.168.1.11

;; AUTHORITY SECTION:
santiago.com.it.      86400   IN      NS       dns.santiago.com.it.

;; ADDITIONAL SECTION:
dns.santiago.com.it.  86400   IN      A       192.168.1.10

;; Query time: 1 msec
;; SERVER: ::1#53(:1)
;; WHEN: Tue Sep 16 07:36:26 -05 2025
;; MSG SIZE rcvd: 123

root@solaris:~# █

```

Nota. Presenta los resultados de una consulta DNS usando la herramienta dig, validando que el servidor responde correctamente a las solicitudes definidas en la zona configurada.

Figura 7*Prueba en Slackware como servidor secundario*

```

root@natalia:~# nslookup server1.santiago.com.it 10.2.77.180
Server:          10.2.77.180
Address:         10.2.77.180#53

Name:   server1.santiago.com.it
Address: 192.168.1.11

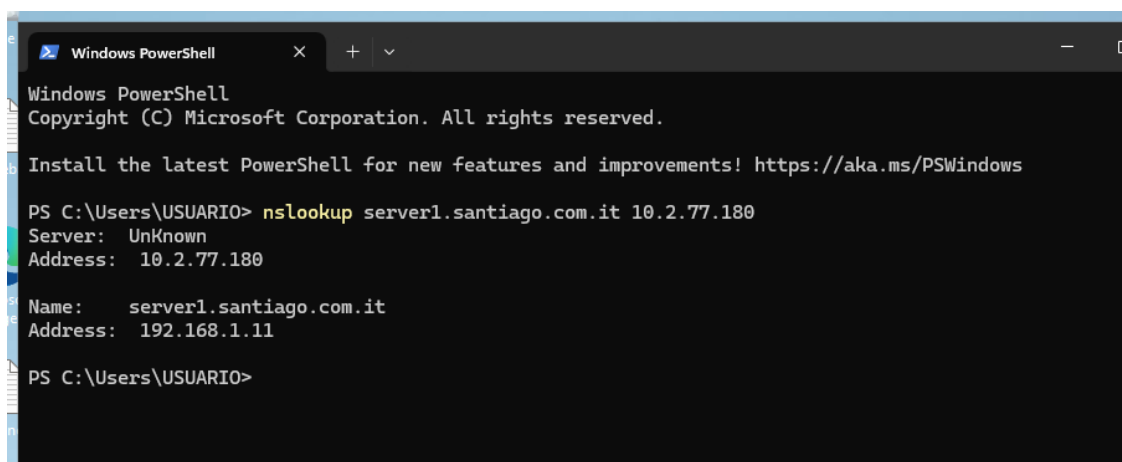
root@natalia:~#

```

Nota. Prueba para confirmar que el servidor secundario responde correctamente a las solicitudes definidas en la zona configurada.

Figura 8

Prueba en Windows como servidor secundario



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\USUARIO> nslookup server1.santiago.com.it 10.2.77.180
Server: UnKnown
Address: 10.2.77.180

Name:     server1.santiago.com.it
Address:  192.168.1.11

PS C:\Users\USUARIO>
```

Nota. Prueba para confirmar que el servidor secundario responde correctamente a las solicitudes definidas en la zona configurada.

La arquitectura propuesta incluye, además del servidor primario en Solaris, dos servidores secundarios localizados en sistemas Linux Slackware y Windows Server. La función de estos servidores secundarios es replicar automáticamente la información contenida en las zonas de autoridad del servidor maestro, garantizando la continuidad del servicio ante fallas y mejorando la eficiencia de las consultas. Para ello, se configuraron entradas tipo **slave** en sus respectivos archivos `named.conf`, indicando como fuente de transferencia de zona la dirección IP del servidor Solaris. En el caso de Slackware, fue necesario instalar previamente el paquete de **BIND** si no estaba presente, siguiendo procedimientos similares a los empleados en Solaris.

Configuración en Slackware

Figura 9

Verificación del servicio BIND en Slackware

```
root@natalia:~# which named
/usr/sbin/named
root@natalia:~# ls /usr/sbin/named
/usr/sbin/named*
root@natalia:~# named -v
BIND 9.16.25 (Extended Support Version) <id:3e14423>
root@natalia:~# _
```

Nota. Muestra el resultado del comando `named -v`, confirmando que el servicio BIND está instalado y activo en el sistema Slackware.

Figura 10

Contenido del archivo de configuración

```
options {
    directory "/var/named";
    allow-query { any; };
    recursion yes;
};

zone "." IN {
    type hint;
    file "named.ca";
};

# Zone for natalia.org.uk (primary dns)
zone "natalia.org.uk" IN {
    type master;
    file "natalia.org.uk.zone";
    allow-transfer { 10.2.77.180; };
};

# Zone for santiago.com.it
zone "santiago.com.it" IN {
    type slave;
    file "santiago.com.it.slave";
    masters { 10.2.77.180; };
};
```

Nota. Ilustra la estructura del archivo named.conf, donde se definen las zonas DNS, interfaces de red, políticas de acceso y parámetros generales del servicio.

Figura 11

Zona maestra del dominio natalia.org.uk

```
;
; Zone file for natalia.org.uk
;
$INCLUDE natalia.soa

; Name Server
natalia.org.uk. IN NS dns.natalia.org.uk.

dns.natalia.org.uk. IN A 10.2.77.182
server1.natalia.org.uk. IN A 10.2.77.183
server2.natalia.org.uk. IN A 10.2.77.184
server3.natalia.org.uk. IN A 10.2.77.185

server4.natalia.org.uk. IN AAAA 2001:db8::1
server5.natalia.org.uk. IN AAAA 2001:db8::2

www.natalia.org.uk. IN CNAME server1.natalia.org.uk.
mail.natalia.org.uk. IN CNAME server2.natalia.org.uk.
~
~
```

Nota. Muestra el archivo de zona /var/named/natalia.org.uk.zone, incluyendo registros SOA, A, AAAA y CNAME.

Figura 12

Archivo raíz de resolución externa /etc/named.ca

```
; Root name servers
A.ROOT-SERVERS.NET. 3600000 IN A 198.41.0.4
B.ROOT-SERVERS.NET. 3600000 IN A 192.228.79.201
~
~
```

Nota. Representa el archivo named.ca, que contiene referencias a servidores raíz para permitir la resolución de dominios fuera de las zonas locales.

Figura 13

Archivo con el registro SOA /var/named/natalia.soa

```
;
; SOA record for natalia.org.uk
;
@ IN SOA dns.natalia.org.uk. root.natalia.org.uk. (
    2025091601 ; Serial
    3600       ; Refresh
    1800       ; Retry
    604800     ; Expire
    86400      ; Minimum TTL
)
```

Nota. Dice quién manda en el dominio y controla la sincronización con los servidores secundarios.

Figura 14

Prueba con nslookup en Slackware

```
root@natalia:~# nslookup dns.natalia.org.uk 10.2.77.185
Server:      10.2.77.185
Address:     10.2.77.185#53

Name:   dns.natalia.org.uk
Address: 10.2.77.182
```

Nota. Se utiliza para comprobar si la resolución de nombres funciona.

Figura 15

Prueba en Solaris como servidor secundario

```
root@solaris:~# svcs -a | grep dns
disabled 7:06:30 svc:/network/dns/client:default
disabled 7:06:31 svc:/network/dns/multicast:default
disabled 7:06:32 svc:/network/dns/server:default
root@solaris:~# svcadm enable svc:/network/dns/client:default
root@solaris:~# svcadm enable svc:/network/dns/server:default
root@solaris:~# svcs -a | grep dns
disabled 7:06:31 svc:/network/dns/multicast:default
online   7:13:38 svc:/network/dns/client:default
online   7:13:58 svc:/network/dns/server:default
root@solaris:~# nslookup server1.natalia.org.uk 10.2.77.185
Server:      10.2.77.185
Address:     10.2.77.185#53

Name:   server1.natalia.org.uk
Address: 10.2.77.183
```

Nota. Prueba para confirmar que el servidor secundario responde correctamente a las solicitudes definidas en la zona configurada.

Figura 16

Prueba en Windows como servidor secundario



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\USUARIO> nslookup server1.natalia.org.uk 10.2.77.177
Server:      Unknown
Address:     10.2.77.177

Name:       server1.natalia.org.uk
Address:    10.2.77.183

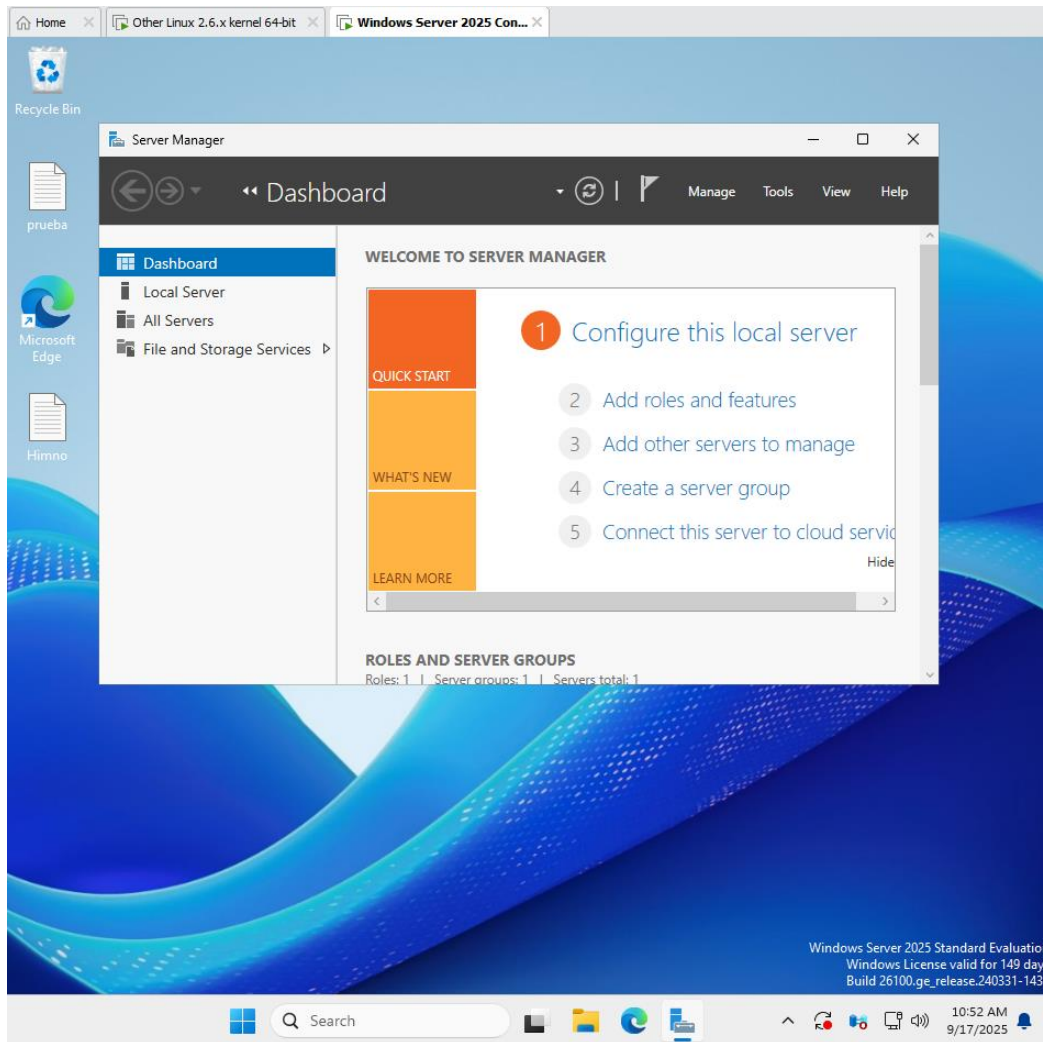
PS C:\Users\USUARIO>
```

Nota. Prueba para confirmar que el servidor secundario responde correctamente a las solicitudes definidas en la zona configurada.

Configuración Windows Server

Figura 17

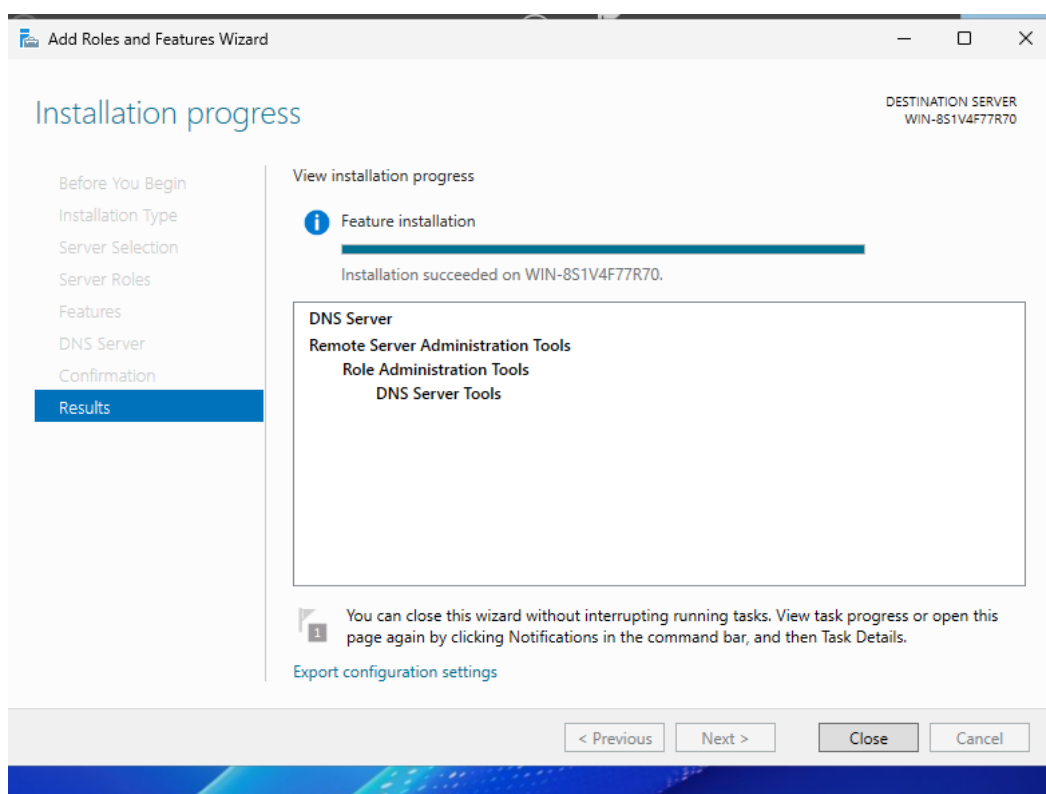
Configuración de roles y características



Nota. Se debe seleccionar este apartado ya que es necesario instalar el rol de servidor DNS.

Figura 18

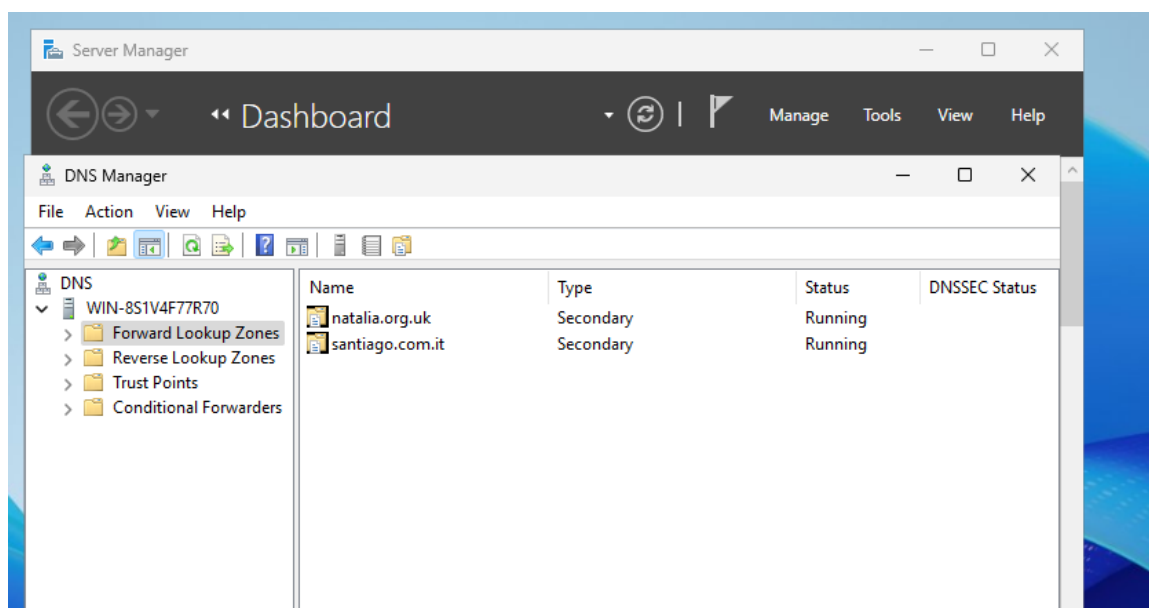
Instalación del rol de servidor DNS



Nota. Se instaló el servicio DNS Server y también las herramientas administrativas para configurarlo.

Figura 19

Configuración en el apartado de Administrador de DNS



Nota. Se agrego la IP de la maquina de Slackware y Solaris en este apartado

Se realizo un video mostrando el uso de la herramienta nslookup desde las maquinas. Esta funcionalidad permite visualizar en detalle cómo se procesan las consultas DNS, incluyendo encabezados del paquete, secciones de preguntas, respuestas, registros de autoridad y adicionales. Es especialmente útil para diagnosticar problemas de resolución de nombres y verificar la configuración de un servidor DNS.

Mira el video:

https://drive.google.com/file/d/1qBjwvCxX_9EuTPGMipgSeMSrvUX64siJ/view?usp=sharing

Figura 20

Resolución de consultas DNS para el dominio natalia.org.uk en entorno Windows

```
> set type=NS
> natalia.org.uk
Server: [10.2.77.185]
Address: 10.2.77.185

-----
Got answer:
HEADER:
  opcode = QUERY, id = 7, rcode = SERVFAIL
  header flags: response, want recursion, recursion avail.
  questions = 1, answers = 0, authority records = 0, additional = 0

  QUESTIONS:
    natalia.org.uk.is.escuelaing.edu.co, type = NS, class = IN

-----
Got answer:
HEADER:
  opcode = QUERY, id = 8, rcode = NOERROR
  header flags: response, auth. answer, want recursion, recursion avail.
  questions = 1, answers = 1, authority records = 0, additional = 1

  QUESTIONS:
    natalia.org.uk, type = NS, class = IN
  ANSWERS:
    -> natalia.org.uk
        nameserver = dns.natalia.org.uk
        ttl = 86400 (1 day)
  ADDITIONAL RECORDS:
    -> dns.natalia.org.uk
        internet address = 10.2.77.182
        ttl = 86400 (1 day)

-----
natalia.org.uk
  nameserver = dns.natalia.org.uk
  ttl = 86400 (1 day)
dns.natalia.org.uk
  internet address = 10.2.77.182
  ttl = 86400 (1 day)
> |
```

Nota. La imagen presenta el resultado de una consulta DNS del tipo NS realizada con el comando nslookup desde una terminal en Slackware. Se observa una primera respuesta con código SERVFAIL al intentar resolver un subdominio mal formado (natalia.org.uk.is.escuelaing.edu.co.), seguida de una consulta exitosa (NOERROR) para el dominio correcto natalia.org.uk. El servidor DNS responde con un registro NS que indica que el servidor de nombres autoritativo es dns.natalia.org.uk, y proporciona un registro adicional con su dirección IP (10.2.77.182). La información incluye detalles técnicos como el TTL (86400 segundos), el número de preguntas, respuestas y registros adicionales, lo que evidencia una resolución DNS completa y funcional.

Figura 21

Resolución de consultas DNS para el dominio natalia.org.uk en entorno Windows

```
> set debug
> natalia.org.uk
Server: [10.2.77.185]
Address: 10.2.77.185

-----
Got answer:
HEADER:
  opcode = QUERY, id = 3, rcode = SERVFAIL
  header flags: response, want recursion, recursion avail.
  questions = 1, answers = 0, authority records = 0, additional = 0

QUESTIONS:
  natalia.org.uk.is.escuelaing.edu.co, type = A, class = IN

-----
Got answer:
HEADER:
  opcode = QUERY, id = 4, rcode = SERVFAIL
  header flags: response, want recursion, recursion avail.
  questions = 1, answers = 0, authority records = 0, additional = 0

QUESTIONS:
  natalia.org.uk.is.escuelaing.edu.co, type = AAAA, class = IN

-----
Got answer:
HEADER:
  opcode = QUERY, id = 5, rcode = NOERROR
  header flags: response, auth. answer, want recursion, recursion avail.
  questions = 1, answers = 0, authority records = 1, additional = 0

QUESTIONS:
  natalia.org.uk, type = A, class = IN
AUTHORITY RECORDS:
-> natalia.org.uk
   ttl = 86400 (1 day)
   primary name server = dns.natalia.org.uk
   responsible mail addr = root.natalia.org.uk
   serial = 2025091602
   refresh = 3600 (1 hour)
   retry = 1800 (30 mins)
   expire = 604800 (7 days)
   default TTL = 86400 (1 day)

-----
Got answer:
HEADER:
  opcode = QUERY, id = 6, rcode = NOERROR
  header flags: response, auth. answer, want recursion, recursion avail.
  questions = 1, answers = 0, authority records = 1, additional = 0

QUESTIONS:
  natalia.org.uk, type = AAAA, class = IN
AUTHORITY RECORDS:
-> natalia.org.uk
   ttl = 86400 (1 day)
   primary name server = dns.natalia.org.uk
   responsible mail addr = root.natalia.org.uk
   serial = 2025091602
   refresh = 3600 (1 hour)
   retry = 1800 (30 mins)
   expire = 604800 (7 days)
   default TTL = 86400 (1 day)

-----
Name: natalia.org.uk
```

Nota. La imagen muestra una serie de consultas DNS realizadas desde una terminal en Slackware, utilizando el comando nslookup para el dominio natalia.org.uk y sus subdominios. Se observan respuestas con códigos SERVFAIL para dominios mal formateados y NOERROR para consultas válidas de tipo A y AAAA. También se detallan los registros de autoridad (SOA), incluyendo el servidor primario (dns.natalia.org.uk), el

correo responsable (root.natalia.org.uk), y parámetros como el número de serie, tiempos de actualización, reintento, expiración y TTL. Esta evidencia respalda el análisis de configuración DNS en un entorno de laboratorio.

Figura 22

Consulta DNS tipo A para el dominio natalia.org.uk en entorno Windows

```
> set type=A
> natalia.org.uk
Server: [10.2.77.185]
Address: 10.2.77.185

-----
Got answer:
HEADER:
  opcode = QUERY, id = 9, rcode = SERVFAIL
  header flags: response, want recursion, recursion avail.
  questions = 1, answers = 0, authority records = 0, additional = 0

QUESTIONS:
  natalia.org.uk.is.escuelaing.edu.co, type = A, class = IN

-----
Got answer:
HEADER:
  opcode = QUERY, id = 10, rcode = NOERROR
  header flags: response, auth. answer, want recursion, recursion avail.
  questions = 1, answers = 0, authority records = 1, additional = 0

QUESTIONS:
  natalia.org.uk, type = A, class = IN
AUTHORITY RECORDS:
-> natalia.org.uk
   ttl = 86400 (1 day)
   primary name server = dns.natalia.org.uk
   responsible mail addr = root.natalia.org.uk
   serial = 2025091602
   refresh = 3600 (1 hour)
   retry = 1800 (30 mins)
   expire = 604800 (7 days)
   default TTL = 86400 (1 day)

-----
Name: natalia.org.uk
```

Nota. La imagen muestra el resultado de una consulta DNS tipo A realizada con nslookup en una terminal Slackware. Inicialmente, se observa un error SERVFAIL al intentar resolver un dominio mal formado (natalia.org.uk.is.escuelaing.edu.co.). Posteriormente, la consulta correcta a natalia.org.uk devuelve un código NOERROR, aunque sin registros de respuesta. Sin embargo, se incluye un registro de autoridad (SOA) que especifica el servidor primario (dns.natalia.org.uk), el correo responsable (root.natalia.org.uk), y parámetros como el número de serie, tiempos de actualización, reintento, expiración y

TTL. Esto indica que el dominio está configurado correctamente en el servidor DNS, pero no tiene un registro A definido.

Figura 23

Prueba de nombre de dominio

```
root@natalia:~# nslookup server1.santiago.com.it
;; connection timed out; no servers could be reached

root@natalia:~#
```

Nota. No se puede resolver el nombre de dominio porque el archivo `/etc/resolv.conf` no está configurado para usar el DNS local.

Figura 24

Verificación de nombre de dominio

```
/etc/resolv.conf: 3 lines, 57 characters.
root@natalia:~# nslookup server1.santiago.com.it
Server:      10.2.77.185
Address:     10.2.77.185#53

Name:   server1.santiago.com.it
Address: 192.168.1.11

root@natalia:~#
```

Nota. Como solución a dicho problema, solo se debe añadir la IP de la maquina en `/etc/resolv.conf` y realizando la prueba con el mismo comando anterior, se verifica que ya funciona.

Otros comandos útiles

Configuración de Tareas Programadas en Sistemas UNIX

El script **schedule-task-script.sh** permite al usuario configurar y gestionar tareas programadas en sistemas UNIX, específicamente en entornos Solaris y Slackware GNU/Linux. Mediante la ejecución del script desde la línea de comandos, los usuarios pueden establecer tareas que se ejecutarán en intervalos regulares, utilizando la sintaxis estándar de cron. El script acepta como parámetro la frecuencia de ejecución, la cual debe ser especificada en el formato cron, sin ningún tipo de interacción o solicitud de parámetros adicionales por parte del sistema.

El siguiente conjunto de pruebas tiene como objetivo verificar el correcto funcionamiento del script `schedule-task-script.sh` en sistemas UNIX, específicamente en entornos Solaris y Slackware GNU/Linux. Cada prueba se centra en un aspecto específico del comportamiento del script, validando su capacidad para gestionar crontabs de manera eficiente y sin errores. A continuación, se describen detalladamente las pruebas realizadas.

Prueba 1: Visualización de la ayuda (-h, --help y sin argumentos):

El objetivo de esta prueba es garantizar que el script despliegue correctamente el mensaje de uso cuando se invoca con los parámetros `-h`, `--help` o cuando no se pasa ningún argumento. Para ello, se ejecuta el script en ambas plataformas, Slackware y Solaris, utilizando los parámetros mencionados. En Slackware, se ejecutan los comandos `bash schedule-task-script.sh -h`, `bash schedule-task-script.sh --help` y `bash schedule-task-script.sh`. En Solaris, se repiten los mismos comandos con el shell `ksh` en lugar de `bash`. El resultado esperado en ambos sistemas es que el script imprima el bloque de ayuda con ejemplos de uso del script, indicando la forma

correcta de invocar el script y proporcionando ejemplos de tareas programadas. Además, el código de retorno debe ser distinto de 0, generalmente 1, lo cual indica que el script terminó con un estado de error debido a la falta de parámetros válidos, lo cual es esperado al invocar el script de esta manera.

Figura 25

Ayuda del script en Slackware

```
root@darkstar:~# bash /mnt/solaris_share/schedule-task-script.sh --help
Usage: /mnt/solaris_share/schedule-task-script.sh "<cron_schedule> <command> [args]"
Example:
  /mnt/solaris_share/schedule-task-script.sh "* * * * * /path/to/command arg1 arg2"

This will add the specified cron job to your crontab if it does not already exist.

Options:
  -h, --help    Show this help message and exit
2025-09-15 23:10:09 - ERROR: Unexpected failure.
root@darkstar:~# _
```

Nota. El script `schedule-task-script.sh` muestra correctamente el bloque de ayuda al ejecutarse con `-h`, `--help` o sin argumentos en Slackware. Se observa el mensaje de uso y ejemplos de cron válidos.

Figura 26

Ayuda del script en Solaris

```
aurora@solaris:~$ bash /path/to/shared/directory/schedule-task-script.sh --help
Usage: /path/to/shared/directory/schedule-task-script.sh "<cron_schedule> <command> [args]"
Example:
  /path/to/shared/directory/schedule-task-script.sh "* * * * * /path/to/command
arg1 arg2"

This will add the specified cron job to your crontab if it does not already exist.

Options:
  -h, --help    Show this help message and exit
2025-09-15 18:04:38 - ERROR: Unexpected failure.
aurora@solaris:~$ █
```

Nota. En Solaris, el script ejecutado con `ksh` también despliega el mensaje de ayuda esperado, confirmando la compatibilidad con este entorno. El código de retorno es 1.

Prueba 2: Validación de formato de entrada de cron

Esta prueba tiene como objetivo asegurar que el script valide correctamente el formato de las entradas cron, rechazando aquellas que no contengan los seis campos requeridos (cinco para el horario y uno para el comando). Para esta prueba, se invoca el script en ambos entornos con una entrada cron incompleta. En Slackware, se ejecuta el comando `bash schedule-task-script.sh "*/5 * * *"`, mientras que en Solaris se utiliza el comando `ksh schedule-task-script.sh "*/5 * * *"`. El resultado esperado es que el script imprima el mensaje de error: `ERROR: Invalid cron entry. Must have at least 6 fields.`, indicando que la entrada no es válida. Además, el código de retorno debe ser 1, lo que indica que el script ha fallado debido a un formato de cron incorrecto.

Figura 27

Error por entrada cron incompleta en Slackware

```
root@darkstar: # bash /mnt/solaris_share/schedule-task-script.sh "*/5 * * *"
2025-09-15 23:11:29 - ERROR: Invalid cron entry. Must have at least 6 fields.
Usage: /mnt/solaris_share/schedule-task-script.sh "<cron_schedule> <command> [args]"
Example:
  /mnt/solaris_share/schedule-task-script.sh "* * * * * /path/to/command arg1 arg2"
This will add the specified cron job to your crontab if it does not already exist.
Options:
  -h, --help    Show this help message and exit
2025-09-15 23:11:29 - ERROR: Unexpected failure.
root@darkstar:~#
```

Nota. El script detecta correctamente una entrada cron inválida con menos de seis campos. Se muestra el mensaje de error y el código de retorno es 1.

Figura 28*Error por entrada cron incompleta en Solaris*

```

aurora@solaris:~$ bash /path/to/shared/directory/schedule-task-script.sh "*/5 *
* *"
2025-09-15 18:12:27 - ERROR: Invalid cron entry. Must have at least 6 fields.
Usage: /path/to/shared/directory/schedule-task-script.sh "<cron_schedule> <comma
nd> [args]"
Example:
/path/to/shared/directory/schedule-task-script.sh "* * * * * /path/to/command
arg1 arg2"

This will add the specified cron job to your crontab if it does not already exist.

Options:
-h, --help    Show this help message and exit
2025-09-15 18:12:27 - ERROR: Unexpected failure.
aurora@solaris:~$ █

```

Nota. En Solaris, el script también rechaza la entrada incompleta, mostrando el mensaje: ERROR: Invalid cron entry. Must have at least 6 fields.

Prueba 3: Añadir entrada cuando no existe crontab previo

En esta prueba se verifica que el script cree un crontab nuevo cuando el usuario no tiene un crontab previo. Primero, se elimina cualquier crontab existente en ambos sistemas utilizando el comando `crontab -r`. Luego, se ejecuta el script con una línea válida de cron, como `bash schedule-task-script.sh "* * * * * /usr/bin/echo Hola"` en Slackware y `ksh schedule-task-script.sh "* * * * * /usr/bin/echo Hola"` en Solaris. El resultado esperado es que el script imprima los siguientes mensajes: "No existing crontab found — starting a new one", "Appended new cron entry" y "Crontab updated successfully". Además, el comando `crontab -l` debe mostrar la nueva entrada configurada en el crontab: `* * * * * /usr/bin/echo Hola`. El código de retorno debe ser 0, lo que indica que el script se ejecutó correctamente y actualizó el crontab con la nueva entrada.

Figura 29

Creación de nuevo crontab en Slackware

```
root@darkstar:~# crontab -d
root@darkstar:~# bash /mnt/solaris_share/schedule-task-script.sh "* * * * * /usr/bin/echo Hola"
2025-09-15 23:23:59 - Loaded existing crontab
2025-09-15 23:23:59 - Appended new cron entry
2025-09-15 23:23:59 - Crontab updated successfully
root@darkstar:~# crontab -l
* * * * * /usr/bin/echo Hola
root@darkstar:~# _
```

Nota. Tras eliminar el crontab previo, el script crea uno nuevo con la entrada * * * * * /usr/bin/echo Hola. Se muestran los mensajes esperados y el código de retorno es 0.

Figura 30

Creación de nuevo crontab en Solaris

```
aurora@solaris:~$ crontab -r
aurora@solaris:~$ bash /path/to/shared/directory/schedule-task-script.sh "* * *
* * /usr/bin/echo Hola"
2025-09-15 18:25:06 - No existing crontab found â starting a new one
2025-09-15 18:25:06 - Appended new cron entry
2025-09-15 18:25:06 - Crontab updated successfully
aurora@solaris:~$ crontab -l
* * * * * /usr/bin/echo Hola
aurora@solaris:~$ █
```

Nota. En Solaris, el script también detecta la ausencia de crontab y genera uno nuevo correctamente. La entrada aparece reflejada en crontab -l.

Prueba 4: Añadir entrada cuando ya existe crontab sin duplicados

Esta prueba tiene como objetivo validar que el script agregue correctamente una nueva entrada de cron en un crontab ya existente, sin generar duplicados. Para ello, se crea un crontab inicial con la entrada 0 0 * * * /usr/bin/backup, utilizando el comando echo "0 0 * * * /usr/bin/backup" | crontab -. A continuación, se ejecuta el script con una nueva entrada de cron, como bash schedule-task-script.sh "30 2 * * 1 /usr/bin/cleanup" en Slackware o ksh schedule-task-script.sh "30 2 * * 1 /usr/bin/cleanup" en Solaris. Después de ejecutar el script, se utiliza el comando crontab -l para verificar que ambas entradas estén presentes en el crontab, es decir, que

no se haya eliminado la entrada previa y que la nueva tarea se haya añadido correctamente. Los mensajes esperados en stdout son: "Loaded existing crontab", "Appended new cron entry" y "Crontab updated successfully". El resultado esperado es que el crontab muestre las dos entradas: 0 0 * * * /usr/bin/backup y 30 2 * * 1 /usr/bin/cleanup, y el código de retorno debe ser 0, indicando que el script se ejecutó correctamente y actualizó el crontab sin errores.

Figura 31

Añadir entrada sin duplicar en Slackware

```
root@darkstar:~# ( echo "0 0 * * * /usr/bin/backup" ) | crontab -
root@darkstar:~# bash /mnt/solaris_share/schedule-task-script.sh "30 2 * * 1 /usr/bin/cleanup"
2025-09-15 23:29:27 - Loaded existing crontab
2025-09-15 23:29:27 - Appended new cron entry
2025-09-15 23:29:27 - Crontab updated successfully
root@darkstar:~# crontab -l
0 0 * * * /usr/bin/backup
30 2 * * 1 /usr/bin/cleanup
root@darkstar:~#
```

Nota. El script añade una nueva entrada al crontab existente sin eliminar la anterior. Ambas tareas aparecen correctamente listadas.

Figura 32

Añadir entrada sin duplicar en Solaris

```
aurora@solaris:~$ echo "0 0 * * * /usr/bin/backup" > /tmp/testcron.$$
aurora@solaris:~$ crontab /tmp/testcron.$$
aurora@solaris:~$ rm -f /tmp/testcron.$$
aurora@solaris:~$ bash /path/to/shared/directory/schedule-task-script.sh "30 2
* * 1 /usr/bin/cleanup"
2025-09-15 18:33:06 - Loaded existing crontab
2025-09-15 18:33:06 - Appended new cron entry
2025-09-15 18:33:06 - Crontab updated successfully
aurora@solaris:~$ crontab -l
0 0 * * * /usr/bin/backup
30 2 * * 1 /usr/bin/cleanup
aurora@solaris:~$ █
```

Nota. En Solaris, se confirma que el script preserva la entrada previa y añade la nueva sin duplicados. El código de retorno es 0.

Prueba 5: Detección de duplicados y sin acción adicional

Finalmente, esta prueba tiene como objetivo verificar que el script no duplique entradas en el crontab si una entrada ya existe. Se inicia creando un crontab con la entrada `15 5 * * * /usr/bin/update`, utilizando el comando `echo "15 5 * * * /usr/bin/update" | crontab -`. Luego, se ejecuta el script con la misma entrada cron, como `bash schedule-task-script.sh "15 5 * * * /usr/bin/update"` en Slackware o `ksh schedule-task-script.sh "15 5 * * * /usr/bin/update"` en Solaris. El resultado esperado es que el script imprima los mensajes "Loaded existing crontab" y "Cron entry already exists; nothing to do", indicando que la entrada ya estaba presente en el crontab y que no se realizó ninguna acción adicional. Al ejecutar `crontab -l`, se espera que la salida sea la siguiente: `15 5 * * * /usr/bin/update`, sin duplicados. El código de retorno debe ser 0, lo que indica que el script no encontró ningún error y que el crontab se mantiene sin cambios.

Figura 33

Detección de entrada duplicada en Slackware

```
root@darkstar:~# ( echo "15 5 * * * /usr/bin/update" ) | crontab -
root@darkstar:~# bash /mnt/solaris_share/schedule-task-script.sh "15 5 * * * /usr/bin/update"
2025-09-15 23:36:01 - Loaded existing crontab
2025-09-15 23:36:01 - Cron entry already exists; nothing to do
root@darkstar:~# crontab -l
15 5 * * * /usr/bin/update
root@darkstar:~#
```

Nota. El script identifica que la entrada ya existe y no realiza modificaciones. Se muestra el mensaje: Cron entry already exists; nothing to do.

Figura 34*Detección de entrada duplicada en Solaris*

```

aurora@solaris:~$ echo "15 5 * * * /usr/bin/update" > /tmp/testcron.$$
aurora@solaris:~$ crontab /tmp/testcron.$$
aurora@solaris:~$ rm -f /tmp/testcron.$$
aurora@solaris:~$ bash /path/to/shared/directory/schedule-task-script.sh "15 5
* * * /usr/bin/update"
2025-09-15 18:43:00 - Loaded existing crontab
2025-09-15 18:43:00 - Cron entry already exists; nothing to do
aurora@solaris:~$ crontab -l
15 5 * * * /usr/bin/update
aurora@solaris:~$ █

```

Nota. En Solaris, el comportamiento es el mismo: el script evita duplicar la entrada existente y mantiene el crontab sin cambios.

Las pruebas realizadas aseguran que el script `schedule-task-script.sh` funcione correctamente en las plataformas Solaris y Slackware GNU/Linux, gestionando crontabs sin errores y sin generar entradas duplicadas. El comportamiento esperado se cumple en cada una de las pruebas descritas, garantizando que el script pueda manejar tareas programadas de manera eficiente y sin problemas, incluso cuando se ejecuta con parámetros incorrectos o cuando se intenta agregar entradas duplicadas.

Script de Gestión de Procesos en Sistemas UNIX

El script `process-manager.sh` tiene como objetivo proporcionar una interfaz interactiva que permita al usuario gestionar procesos en un sistema operativo Unix. A través de un menú de opciones, el usuario puede visualizar los procesos en ejecución, buscar procesos específicos, terminar o reiniciar procesos, entre otras funcionalidades. Este informe detalla los casos de prueba diseñados para validar el correcto funcionamiento de dicho script en dos entornos Unix: **Solaris** y **Slackware GNU/Linux**. Cada caso de prueba incluye una descripción detallada de los pasos a seguir, las diferencias entre ambos sistemas operativos, y el resultado esperado.

Prueba 1: Mostrar Ayuda y Manejo de Parámetros Inválidos

El objetivo de este caso de prueba es verificar que el script `process-manager.sh` despliegue correctamente el mensaje de ayuda al invocar el parámetro `--help`, y que sea capaz de rechazar cualquier parámetro no soportado. Para llevar a cabo esta prueba, se debe colocar el script en un directorio limpio y asegurarse de que sea legible en ambas plataformas, utilizando el comando `chmod +r` en **Slackware** y **Solaris**. Posteriormente, se ejecuta el script con el parámetro `--help` tanto en **Slackware** como en **Solaris**, para confirmar que el bloque de ayuda sea desplegado adecuadamente. Además, se debe probar el comportamiento al ejecutar el script con un parámetro no válido, como `--version` o un parámetro inesperado en Solaris. En ambos casos, el resultado esperado es que el script vuelva a mostrar el bloque de ayuda y que el código de retorno sea 0, indicando que la ejecución fue exitosa. Es importante resaltar que no es necesario ejecutar el script con privilegios de superusuario para este tipo de prueba, y que el script es compatible tanto con `bash` (en **Slackware**) como con `ksh` (en **Solaris**).

Figura 35

Ayuda del script en Slackware

```
root@darkstar:~# bash /mnt/solaris_share/process-manager.sh --help
Usage: /mnt/solaris_share/process-manager.sh [--help]
Options:
  --help      Show this help message and exit

Menu options will appear when run without arguments.
root@darkstar:~#
```

Nota. El script process-manager.sh muestra correctamente el bloque de ayuda al ejecutarse con --help. También responde adecuadamente a parámetros inválidos como --version, mostrando nuevamente la ayuda.

Figura 36

Ayuda del script en Solaris

```
aurora@solaris:~$ bash /path/to/shared/directory/process-manager.sh --help
Usage: /path/to/shared/directory/process-manager.sh [--help]
Options:
  --help      Show this help message and exit

Menu options will appear when run without arguments.
aurora@solaris:~$ █
```

Nota. En Solaris, el script ejecutado con ksh despliega el mensaje de ayuda esperado y rechaza parámetros no soportados, confirmando su robustez frente a entradas incorrectas.

Prueba 2: Listar Todos los Procesos

El objetivo de este caso de prueba es verificar que la opción "1) List all processes" muestre los procesos en ejecución en un formato tabular claro y alineado, con columnas para el identificador del proceso (PID), el nombre del comando, y los porcentajes de uso de CPU y memoria. Para ello, se debe ejecutar el script y acceder al menú principal. Al seleccionar la opción para listar todos los procesos, se espera que el script despliegue un encabezado con las columnas mencionadas, seguido de las filas correspondientes a los procesos en ejecución. El formato de salida debe ser consistente y legible en ambas plataformas, **Slackware** y **Solaris**, con

la información de los procesos correctamente alineada en columnas. El resultado esperado es que, al seleccionar la opción, el script proporcione una lista de procesos con la siguiente estructura: PID, COMMAND, CPU%, y MEM%. Esto garantizará que la información de los procesos se muestre de manera clara y coherente.

Figura 37

Listado de procesos en Slackware

731	card0-crtc4	0.0	0.0
732	card0-crtc5	0.0	0.0
733	card0-crtc6	0.0	0.0
734	card0-crtc7	0.0	0.0
842	syslogd	0.0	0.2
844	mld	0.0	0.0
845	ip6_addrconf	0.0	0.0
947	dbus-daemon	0.0	0.2
962	elogind-daemon	0.0	0.4
976	sshd	0.0	0.4
996	crond	0.0	0.2
1001	login	0.0	0.5
1002	agetty	0.0	0.2
1003	agetty	0.0	0.2
1004	agetty	0.0	0.2
1005	agetty	0.0	0.2
1006	agetty	0.0	0.2
1018	bash	0.0	0.5
1031	cifsiod	0.0	0.0
1032	smb3decryptd	0.0	0.0
1033	cifsfileinfopt	0.0	0.0
1034	cifsoplockd	0.0	0.0
1035	deferredclose	0.0	0.0
1036	cifsd	0.0	0.0
1161	kworker/1:2	0.0	0.0
1181	bash	0.0	0.4
1186	ps	0.0	0.1
1187	tail	0.0	0.1

Process Management Menu			
1) List all processes			
2) Search for a process			
3) Kill a process			
4) Restart a process			
5) Exit			

Choose an option [1-5]:

Nota. Al seleccionar la opción "1) List all processes", el script muestra una tabla clara y alineada con los procesos activos en Slackware.

Figura 38*Listado de procesos en Solaris*

```

717 /usr/lib/inet/inetd 0.0 0.2
718 /usr/sbin/smbd 0.0 0.1
725 /usr/lib/devchassis/devchassisd 0.0 0.1
768 /usr/lib/hal/hald-addon-storage 0.0 0.1
786 /usr/apache2/2.4/bin/httpd 0.0 0.2
788 /usr/apache2/2.4/bin/httpd 0.0 0.3
759 /usr/apache2/2.4/bin/httpd 0.0 0.2
760 /usr/apache2/2.4/bin/rotatelogs 0.0 0.1
826 /usr/sbin/syslogd 0.0 0.1
763 /usr/sbin/smbd 0.0 0.2
783 /usr/sbin/dhcpagent 0.0 0.1
1127 /usr/sbin/smbd 0.0 0.3
844 /usr/sbin/auditd 0.0 0.1
863 /usr/lib/vttdaemon 0.0 0.1
1116 -bash 0.1 0.2
882 /usr/sbin/ttymon 0.0 0.1
1256 bash 0.1 0.2
894 /usr/sbin/ttymon 0.0 0.1
921 /usr/sbin/ttymon 0.0 0.1
1010 /usr/lib/sstore/bin/sysstatd 0.0 0.3
937 /usr/sbin/ttymon 0.0 0.1
1000 /usr/lib/fm/notify/smtp-notify 0.0 0.2
1003 /usr/lib/fm/notify/asr-notify 0.0 0.2
1261 ps 0.1 0.1
1262 tail 0.1 0.0

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: █

```

Nota. En Solaris, el script presenta los procesos en ejecución con formato tabular coherente, confirmando la correcta visualización en ambos entornos.

Prueba 3: Buscar Proceso Existente y No Existente

Este caso de prueba tiene como objetivo verificar que el script sea capaz de realizar búsquedas de procesos de acuerdo con el nombre o patrón introducido por el usuario. Para ejecutar la prueba, se debe acceder al menú y seleccionar la opción para buscar un proceso. Se debe ingresar un patrón conocido, como `sshd`, y verificar que el script muestre todos los procesos que coincidan con el patrón. A continuación, se debe repetir la búsqueda utilizando un patrón inexistente, como `no_such_process_xyz`, y comprobar que el mensaje "No matches found" se muestre correctamente en la salida. El resultado esperado es que, al buscar un proceso existente,

el script muestre la lista completa de los procesos que coinciden con el patrón, incluyendo la información relevante como el PID, el comando, y los porcentajes de CPU y memoria. En el caso de un patrón inexistente, el script debe mostrar un mensaje que indique que no se encontraron coincidencias.

Figura 39

Búsqueda exitosa de proceso en Slackware

```

root@darkstar:~# bash /mnt/solaris_share/process-manager.sh

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: 2
Enter process name or pattern: ssh
2025-09-15 23:52:37 - Searching for processes matching 'ssh'...
    976 root      0.0  0.4 sshd: /usr/sbin/sshd [listener] 0 of 10-100 startups

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: 2
Enter process name or pattern: no_such_process_xyz
2025-09-15 23:52:52 - Searching for processes matching 'no_such_process_xyz'...
2025-09-15 23:52:52 - No matches found.

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: _

```

Nota. El script encuentra procesos que coinciden con el patrón sshd, mostrando los detalles relevantes como PID, comando, CPU% y MEM%.

Figura 40*Búsqueda sin coincidencias en Solaris*

```

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: 2
Enter process name or pattern: bash
2025-09-15 18:53:41 - Searching for processes matching 'bash'...
 1116  aurora  0.1  0.2 -bash
 1268  aurora  0.1  0.2 bash /path/to/shared/directory/process-manager.sh

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: 2
Enter process name or pattern: no_such_process_xyz
2025-09-15 18:54:01 - Searching for processes matching 'no_such_process_xyz'...
2025-09-15 18:54:01 - No matches found.

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: █

```

Nota. Al buscar un patrón inexistente (no_such_process_xyz), el script muestra el mensaje "No matches found", confirmando su capacidad para manejar búsquedas fallidas.

Prueba 4: Matar un Proceso

El objetivo de este caso es comprobar que el script sea capaz de terminar un proceso dado un PID válido, y que rechace entradas no válidas de manera adecuada. Para llevar a cabo esta prueba, se debe crear un proceso de prueba, como por ejemplo utilizando el comando `sleep 300 &`, y luego obtener el PID de dicho proceso. Después, se debe ejecutar el script y elegir la opción para matar el proceso. Al ingresar un valor no numérico para el PID, el script debe mostrar un mensaje de error, indicando que el PID es inválido. Luego, se debe ingresar un PID válido, y verificar que el script termine el proceso correspondiente. El resultado esperado es que, al

ingresar un PID no válido, el script muestre el mensaje de error "ERROR: Invalid PID". En cambio, al ingresar un PID válido, el script debe terminar el proceso y mostrar un mensaje confirmando que el proceso ha sido detenido exitosamente. Esta funcionalidad es fundamental para asegurar el correcto control de los procesos del sistema a través del script.

Figura 41

Error por PID inválido en Slackware

```

root@darkstar:~# bash -c "sleep 300 & echo \$!" > pid.txt
root@darkstar:~# PID=$(cat pid.txt)
root@darkstar:~# cat pid.txt
1221
root@darkstar:~# bash /mnt/solaris_share/process-manager.sh

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: 3
Enter PID to kill: 1221
2025-09-15 23:59:15 - Killing process PID 1221...
2025-09-15 23:59:15 - Process 1221 terminated.

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: 3
Enter PID to kill: askndaskdbkasjd
2025-09-15 23:59:22 - ERROR: Invalid PID: askndaskdbkasjd
root@darkstar:~# ps -p $PID
  PID TTY          TIME CMD
root@darkstar:~# _

```

Nota. El script rechaza correctamente una entrada no numérica como PID, mostrando el mensaje "ERROR: Invalid PID".

Figura 42*Terminación de proceso en Solaris*

```

aurora@solaris:~$ bash -c "sleep 300 & echo \$!" > pid.txt
aurora@solaris:~$ PID=$(cat pid.txt)
aurora@solaris:~$ cat pid.txt
1286
aurora@solaris:~$ bash /path/to/shared/directory/process-manager.sh

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: 3
Enter PID to kill: 1286
2025-09-15 19:01:43 - Killing process PID 1286...
2025-09-15 19:01:43 - Process 1286 terminated.

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: █

```

Nota. En Solaris, el script finaliza exitosamente un proceso de prueba creado con `sleep 300 &`, mostrando un mensaje de confirmación tras ingresar el PID válido.

Prueba 5: Reiniciar un Proceso

Para validar la funcionalidad “4) Restart a process” del script `process-manager.sh`, se diseñó una prueba que consiste en iniciar un proceso sencillo en segundo plano mediante un script llamado `echo_loop.sh`, el cual escribe la fecha en el archivo `/tmp/alive.log` cada segundo.

```

#!/bin/sh
while sleep 1; do echo "$(date) alive" >> /tmp/alive.log; done

```

Este script se ejecutó en Slackware con `bash echo_loop.sh &` y en Solaris con `ksh echo_loop.sh &`. Luego, se obtuvo el PID del proceso en cada sistema utilizando comandos como `ps aux` en Slackware y `ps -ef` en Solaris. Con el PID identificado, se ejecutó el script `process-`

manager.sh, se seleccionó la opción de reinicio, y se ingresó el PID correspondiente. El comportamiento observado fue que el script mostró el mensaje “Restarting PID <n> with command: /path/echo_loop.sh”, terminó el proceso original y lanzó uno nuevo con el mismo comando, lo cual se confirmó al verificar que el archivo /tmp/alive.log continuó creciendo tras el reinicio. En Solaris, se validó además la existencia de los comandos kill y sleep en /usr/bin, y se ajustó la ruta absoluta al script según el entorno, asegurando compatibilidad en ambos sistemas operativos.

Figura 43

Reinicio de proceso en Slackware

```

~
~
~
~
~
echo_loop.sh: new file: 2 lines, 73 characters.
root@darkstar:~# bash echo_loop.sh &
[1] 1236
root@darkstar:~# bash /mnt/solaris_share/process-manager.sh

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: 4
Enter PID to restart: 1236
2025-09-16 00:13:29 - Restarting PID 1236 with command: bash echo_loop.sh
2025-09-16 00:13:30 - Process restarted.

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: 5
2025-09-16 00:13:33 - Exiting.
[1]+ Terminated bash echo_loop.sh
root@darkstar:~# ps aux | grep echo_loop.sh | grep -v grep
root    1274  0.0  0.4  4044  3160 tty1      S   00:13   0:00 bash echo_loop.sh
root@darkstar:~#

```

Nota. En Slackware, el script process-manager.sh muestra el mensaje de reinicio y lanza nuevamente el proceso echo_loop.sh, confirmando que el archivo /tmp/alive.log continúa creciendo sin interrupciones.

Figura 44*Reinicio de proceso en Solaris*

```

~
~
~
~
"echo_loop.sh" [Archivo nuevo] 2 líneas, 74 characters escritos
aurora@solaris:~$ bash echo_loop.sh &
[1] 1322
aurora@solaris:~$ bash /path/to/shared/directory/process-manager.sh

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: 4
Enter PID to restart: 1322
2025-09-15 19:22:17 - Restarting PID 1322 with command: bash echo_loop.sh
2025-09-15 19:22:18 - Process restarted.

Process Management Menu
1) List all processes
2) Search for a process
3) Kill a process
4) Restart a process
5) Exit

Choose an option [1-5]: 5
2025-09-15 19:22:25 - Exiting.
[1]+ Terminated bash echo_loop.sh
aurora@solaris:~$ ps -ef | grep "sleep 1" | grep -v grep
aurora 1444 1366 0 19:22:56 console 0:00 sleep 1
aurora@solaris:~$ █

```

Nota. En Solaris, tras ingresar el PID del proceso echo_loop.sh, el script detiene correctamente el proceso y lo reinicia, mostrando el log de confirmación y asegurando la persistencia en la escritura del archivo /tmp/alive.log.

El script process-manager.sh ha demostrado ser funcional y versátil en la gestión de procesos en entornos Unix, tanto en **Slackware** como en **Solaris**. Los casos de prueba ejecutados cubren las funcionalidades principales del script, como la visualización de procesos, la búsqueda de procesos específicos, y la capacidad de terminar o reiniciar procesos. Los resultados obtenidos han sido satisfactorios, cumpliendo con los requerimientos establecidos para la herramienta en ambos sistemas operativos. La portabilidad entre diferentes shells (bash y ksh) y la correcta interpretación de los comandos ps y kill en ambos sistemas aseguran que el script sea adecuado para su implementación en entornos de producción diversos.

Listado de Archivos Más Pequeños en Sistemas UNIX

El script `files-script.sh` permite al usuario recorrer el sistema de archivos desde un directorio dado, incluyendo todos sus subdirectorios, y mostrar los n archivos más pequeños cuyo tamaño sea menor o igual a un umbral especificado por el usuario. La salida incluye el nombre del archivo, la ruta completa donde se encuentra y su tamaño en bytes. El script acepta como parámetros el número de archivos a mostrar y el tamaño máximo permitido, utilizando sufijos de unidad estándar (K, M, G). No requiere interacción adicional por parte del usuario y está diseñado para ejecutarse en entornos **Solaris** y **Slackware GNU/Linux**.

El siguiente conjunto de pruebas tiene como objetivo verificar el correcto funcionamiento del script `files-script.sh` en ambos sistemas operativos. Cada prueba se centra en un aspecto específico del comportamiento del script, validando su capacidad para manejar argumentos, filtrar resultados y presentar la información de forma ordenada y coherente. A continuación, se describen detalladamente las pruebas realizadas.

Prueba 1: Validación de número de argumentos

Esta prueba validó que el script muestre un error y el mensaje de uso cuando no se proporcionan los argumentos necesarios o cuando se pasan más o menos de dos. Se ejecutaron los siguientes comandos: `bash files-script.sh`, `bash files-script.sh 5` y `bash files-script.sh 5 10K` extra. El resultado esperado fue que, en cada caso, el script imprimiera el mensaje de error correspondiente junto con el bloque de uso y finalizara con el código 1, lo que confirma que el número de argumentos es validado correctamente.

Figura 45*Error por número incorrecto de argumentos en Slackware*

```

root@darkstar:~# bash /mnt/solaris_share/files-script.sh
2025-09-16 00:28:29 - ERROR: Invalid number of arguments.
Usage: /mnt/solaris_share/files-script.sh <number_of_files> <max_size>
Example: /mnt/solaris_share/files-script.sh 10 1GB
root@darkstar:~# bash /mnt/solaris_share/files-script.sh 5
2025-09-16 00:28:32 - ERROR: Invalid number of arguments.
Usage: /mnt/solaris_share/files-script.sh <number_of_files> <max_size>
Example: /mnt/solaris_share/files-script.sh 10 1GB
root@darkstar:~# bash /mnt/solaris_share/files-script.sh 5 10k extra
2025-09-16 00:28:39 - ERROR: Invalid number of arguments.
Usage: /mnt/solaris_share/files-script.sh <number_of_files> <max_size>
Example: /mnt/solaris_share/files-script.sh 10 1GB
root@darkstar:~#

```

Nota. El script detecta correctamente la ausencia o exceso de argumentos, mostrando el mensaje de error y el bloque de ayuda.

Figura 46*Error por número incorrecto de argumentos en Solaris*

```

aurora@solaris:~$ bash /path/to/shared/directory/files-script.sh
2025-09-15 19:29:38 - ERROR: Invalid number of arguments.
Usage: /path/to/shared/directory/files-script.sh <number_of_files> <max_size>
Example: /path/to/shared/directory/files-script.sh 10 1GB
aurora@solaris:~$ bash /path/to/shared/directory/files-script.sh 5
2025-09-15 19:29:43 - ERROR: Invalid number of arguments.
Usage: /path/to/shared/directory/files-script.sh <number_of_files> <max_size>
Example: /path/to/shared/directory/files-script.sh 10 1GB
aurora@solaris:~$ bash /path/to/shared/directory/files-script.sh 5 10k extra
2025-09-15 19:29:48 - ERROR: Invalid number of arguments.
Usage: /path/to/shared/directory/files-script.sh <number_of_files> <max_size>
Example: /path/to/shared/directory/files-script.sh 10 1GB
aurora@solaris:~$ █

```

Nota. En Solaris, el script responde de forma idéntica ante entradas inválidas, asegurando consistencia en la validación de parámetros.

Prueba 2: Validación de parámetro `number_of_files`

El objetivo de esta prueba fue verificar que el script maneje correctamente el caso en que el primer parámetro no es un entero positivo. Se ejecutaron los siguientes comandos: `bash files-script.sh zero 1K` y `bash files-script.sh -3 1K`. El resultado esperado fue que el script imprimiera

un error indicando que `number_of_files` debe ser un entero positivo, junto con el mensaje de uso, y terminara con el código 1.

Figura 47

Error por parámetro no numérico en Slackware

```
root@darkstar:~# touch a b c
root@darkstar:~# bash /mnt/solaris_share/files-script.sh zero 1k
2025-09-16 00:31:26 - ERROR: number_of_files must be a positive integer.
Usage: /mnt/solaris_share/files-script.sh <number_of_files> <max_size>
Example: /mnt/solaris_share/files-script.sh 10 1GB
root@darkstar:~# _
```

Nota. El script rechaza valores no enteros o negativos para `number_of_files`, mostrando el mensaje: "ERROR: number_of_files must be a positive integer."

Figura 48

Error por parámetro no numérico en Solaris

```
aurora@solaris:~$ touch a b c
aurora@solaris:~$ bash /path/to/shared/directory/files-script.sh -3 1k
2025-09-15 19:31:52 - ERROR: number_of_files must be a positive integer.
Usage: /path/to/shared/directory/files-script.sh <number_of_files> <max_size>
Example: /path/to/shared/directory/files-script.sh 10 1GB
aurora@solaris:~$ █
```

Nota. En Solaris, el script también valida correctamente este parámetro, mostrando el mismo mensaje de error y bloque de ayuda.

Prueba 3: Validación de sufijo de tamaño

En esta prueba se comprobó que el script maneja adecuadamente los sufijos de tamaño no soportados. Se ejecutó el siguiente comando: `bash files-script.sh 2 10T`. El resultado esperado fue que el script imprimiera un error indicando que el sufijo T no es válido, seguido del mensaje de uso, y terminara con el código 1.

Figura 49

Error por sufijo de tamaño inválido en Slackware

```
root@darkstar:~# bash /mnt/solaris_share/files-script.sh 2 10T
2025-09-16 00:34:27 - ERROR: Invalid size suffix 'T'.
Usage: /mnt/solaris_share/files-script.sh <number_of_files> <max_size>
Example: /mnt/solaris_share/files-script.sh 10 1GB
root@darkstar:~# _
```

Nota. El script detecta que el sufijo T no es válido, imprime el mensaje de error correspondiente y finaliza con código 1.

Figura 50

Error por sufijo de tamaño inválido en Solaris

```
aurora@solaris:~$ bash /path/to/shared/directory/files-script.sh 2 10T
2025-09-15 19:34:47 - ERROR: Invalid size suffix 'T'.
Usage: /path/to/shared/directory/files-script.sh <number_of_files> <max_size>
Example: /path/to/shared/directory/files-script.sh 10 1GB
aurora@solaris:~$ █
```

Nota. En Solaris, el comportamiento es idéntico: el script rechaza el sufijo no soportado y muestra el bloque de ayuda.

Prueba 4: Sin archivos bajo el umbral

El objetivo de esta prueba fue verificar que, si no hay archivos cuyo tamaño sea menor o igual a MAX_SIZE, el script informe que no encontró resultados y finalice con código 0. Se creó un directorio vacío y se ejecutó el comando: `bash ../files-script.sh 5 1K`. El resultado esperado fue que el script imprimiera el mensaje: "No files found under the specified size threshold." y terminara con el código de salida 0.

Figura 51

Sin archivos encontrados en Slackware

```
root@darkstar:~# mkdir vacio && cd vacio
root@darkstar:~/vacio# bash /mnt/solaris_share/files-script.sh 5 1k
2025-09-16 00:36:58 - Parameters: count=5, max_size=1024 bytes
No files found under the specified size threshold.
root@darkstar:~/vacio#
```

Nota. En un directorio vacío, el script informa correctamente que no se encontraron archivos bajo el umbral especificado.

Figura 52

Sin archivos encontrados en Solaris

```
aurora@solaris:~$ mkdir vacio && cd vacio
aurora@solaris:~/vacio$ bash /path/to/shared/directory/files-script.sh 5 1K
2025-09-15 19:37:51 - Parameters: count=5, max_size=1024 bytes
No files found under the specified size threshold.
aurora@solaris:~/vacio$
```

Nota. En Solaris, el script muestra el mensaje "No files found under the specified size threshold." y finaliza con código 0.

Prueba 5: Listado de N archivos más pequeños

Esta prueba comprobó que el script liste correctamente los N archivos más pequeños dentro de un árbol de directorios, en orden descendente por tamaño, utilizando el comando `tac`. Se creó un árbol de directorios con archivos de diferentes tamaños y se ejecutó el comando: `bash ../files-script.sh 2 200K`. El resultado esperado fue que el script imprimiera una lista con los dos archivos más pequeños, en el siguiente orden: `d ./dir2/d 75` y `b ./dir1/b 50`, seguido de un código de salida 0.

Figura 53*Listado de archivos más pequeños en Slackware*

```

root@darkstar:~# mkdir -p test/dir1 test/dir2
root@darkstar:~# dd if=/dev/zero of=test/a bs=1 count=100
100+0 records in
100+0 records out
100 bytes copied, 0.000129887 s, 770 kB/s
root@darkstar:~# dd if=/dev/zero of=test/dir1/b bs=1 count=50
50+0 records in
50+0 records out
50 bytes copied, 7.1686e-05 s, 697 kB/s
root@darkstar:~# dd if=/dev/zero of=test/dir1/c bs=1 count=150
150+0 records in
150+0 records out
150 bytes copied, 0.000233343 s, 643 kB/s
root@darkstar:~# dd if=/dev/zero of=test/dir2/d bs=1 count=75
75+0 records in
75+0 records out
75 bytes copied, 0.000183167 s, 409 kB/s
root@darkstar:~# cd test
root@darkstar:~/test# bash /mnt/solaris_share/files-script.sh 2 200K
2025-09-16 00:41:41 - Parameters: count=2, max_size=204800 bytes
d      ./dir2/d      75
b      ./dir1/b      50
FILENAME      FULL_PATH      SIZE_BYTES
root@darkstar:~/test# _

```

Nota. El script muestra los dos archivos más pequeños dentro del árbol de directorios, ordenados por tamaño descendente.

Figura 54*Listado de archivos más pequeños en Solaris*

```

aurora@solaris:~$ mkdir -p test/dir1 test/dir2
aurora@solaris:~$ dd if=/dev/zero of=test/a bs=1 count=100
100+0 records in
100+0 records out
aurora@solaris:~$ dd if=/dev/zero of=test/dir1/b bs=1 count=50
50+0 records in
50+0 records out
aurora@solaris:~$ dd if=/dev/zero of=test/dir1/c bs=1 count=150
150+0 records in
150+0 records out
aurora@solaris:~$ dd if=/dev/zero of=test/dir2/d bs=1 count=75
75+0 records in
75+0 records out
aurora@solaris:~$ cd test
aurora@solaris:~/test$ bash /path/to/shared/directory/files-script.sh 2 200K
2025-09-15 19:44:39 - Parameters: count=2, max_size=204800 bytes
d      ./dir2/d      75
b      ./dir1/b      50
FILENAME      FULL_PATH      SIZE_BYTES
aurora@solaris:~/test$ █

```

Nota. En Solaris, el script presenta la misma salida ordenada, confirmando la correcta ejecución y filtrado de archivos.

Las pruebas realizadas confirman que el script `files-script.sh` funciona correctamente en entornos Solaris y Slackware GNU/Linux, gestionando de forma adecuada la validación de parámetros, el filtrado por tamaño y la presentación ordenada de resultados. El comportamiento esperado se cumple en cada uno de los casos descritos, garantizando que el script pueda emplearse de manera confiable para identificar archivos pequeños dentro de un sistema de archivos, incluso en escenarios con parámetros incorrectos o sin resultados que mostrar.

Resultados

Servidor DNS de Linux – BIND

¿Qué son los registros A y AAAA en el archivo de los servidores raíz?

Según Akamai Technologies (2025), los registros A y AAAA permiten vincular dominios con direcciones IP. El registro tipo A (Address) vincula un nombre de dominio o subdominio con una dirección IP versión 4 (IPv4), permitiendo que el tráfico web se dirija correctamente al servidor que aloja el recurso solicitado. Por su parte, el registro tipo AAAA (conocido como "quad A") cumple la misma función, pero con direcciones IP versión 6 (IPv6), las cuales son esenciales para la conectividad moderna, especialmente en redes móviles y dispositivos que utilizan este protocolo.

Linode (2022) explica que los registros A son esenciales para dirigir el tráfico web hacia el servidor correspondiente. Esto permite que los usuarios accedan a los servicios web alojados en dicho servidor. Un ejemplo típico de registro A sería:

example.com	A	12.34.56.78
hello.example.com	A	12.34.56.78

Es posible asignar diferentes subdominios a distintas direcciones IP, lo que proporciona flexibilidad en la distribución de servicios. Además, si se desea que todos los subdominios de un dominio apunten a la misma dirección IP, se puede utilizar el carácter comodín asterisco (*), como en el siguiente ejemplo:

*.example.com	A	12.34.56.78
---------------	---	-------------

El registro AAAA opera de manera similar, pero está diseñado para direcciones IPv6. Un ejemplo típico de este tipo de registro sería:

example.com	AAAA	0123:4567:89ab:cdef:0123:4567:89ab:cdef
-------------	------	---

Los registros A y AAAA contienen varios campos técnicos que deben configurarse correctamente para garantizar su funcionamiento:

- **Hostname (Nombre de Host):** Especifica el dominio raíz o el subdominio que se desea utilizar. Para el dominio raíz (por ejemplo, example.com), se puede ingresar el carácter @ o dejar el campo vacío. Para subdominios (como host.example.com), se debe ingresar una cadena de entre 1 y 63 caracteres, compuesta por letras, números y guiones. Los guiones no pueden estar al inicio de la cadena.
- **Dirección IP:** Corresponde a la dirección IPv4 o IPv6 del servidor de destino. El sistema de gestión DNS crea automáticamente un registro A o AAAA según el tipo de dirección IP proporcionada.
- **TTL (Time To Live):** Define el tiempo que los resolutores DNS deben almacenar el registro antes de volver a validarlo con los servidores de nombres. Se recomienda establecer el TTL en 5 minutos para la mayoría de los casos. Si se selecciona la opción "Default", el TTL se configura en 24 horas.

Una de las aplicaciones más frecuentes de los registros A y AAAA es la configuración de un sitio web utilizando un dominio personalizado. Por ejemplo, para alojar un sitio en example.com, se debe establecer el nombre de host como @ y asignar la dirección IPv4 del servidor correspondiente. Para brindar soporte a usuarios que acceden mediante IPv6, se recomienda crear también un registro AAAA con el mismo nombre de host y la dirección IPv6 del servidor.

Otra aplicación importante es la configuración del nombre de dominio completamente calificado (FQDN, por sus siglas en inglés) de un servidor. En entornos técnicos, las máquinas suelen ser identificadas por su FQDN en lugar de su dirección IP. Para ello, se debe crear un registro A (y/o AAAA) utilizando el nombre de host de la máquina (por ejemplo, web01 para web01.example.com) y vincularlo con su dirección IP principal.

¿Qué son los registros NS, MX, A y CNAME en el archivo de dominio específico?

En el contexto del Sistema de Nombres de Dominio (DNS), los registros DNS son elementos esenciales que permiten la correcta resolución de nombres de dominio a direcciones IP u otros recursos necesarios para la comunicación en redes. Entre los diversos tipos de registros existentes, los registros A, NS, MX y CNAME desempeñan funciones críticas en la infraestructura de Internet.

1. Registro A (Address Record)

El registro A (Address) es uno de los componentes más fundamentales del sistema DNS. Su función principal es mapear un nombre de dominio a una dirección IPv4 (protocolo de 32 bits). Por ejemplo, al ingresar “example.com” en un navegador, el sistema utiliza el registro A para

traducir ese nombre de dominio a una dirección IP como 93.184.215.14, permitiendo así que el navegador se conecte al servidor web correspondiente.

Estos registros son esenciales para garantizar el acceso adecuado a sitios web y servicios en línea. Un dominio puede tener múltiples registros A asociados, dependiendo del número de direcciones IP a las que deba apuntar.

2. Registro NS (Name Server Record)

Los registros NS (Name Server) indican los servidores de nombres autorizados que contienen la información DNS para un dominio específico. Estos registros son fundamentales en el proceso de resolución DNS, ya que dirigen las consultas hacia los servidores que poseen los registros A, CNAME, MX, entre otros, correspondientes al dominio solicitado.

Normalmente, un dominio incluye varios registros NS para asegurar redundancia y disponibilidad. En caso de falla de un servidor, otro puede asumir la resolución de nombres sin interrumpir el servicio.

3. Registro MX (Mail Exchange Record)

El registro MX (Mail Exchange) especifica los servidores de correo electrónico responsables de recibir mensajes en nombre de un dominio. A diferencia de los registros A o CNAME, que están orientados al tráfico web, el registro MX está diseñado exclusivamente para enrutar correo electrónico hacia los servidores apropiados.

Este tipo de registro también incluye un valor de prioridad, lo que permite configurar múltiples servidores de correo con diferentes niveles de preferencia. En caso de que el servidor de mayor prioridad no esté disponible, el sistema recurrirá al siguiente en la lista, garantizando la continuidad en la recepción de correos.

4. Registro CNAME (Canonical Name Record)

El registro CNAME (Canonical Name) se utiliza para mapear un nombre de dominio a otro dominio, en lugar de a una dirección IP directamente. Esto permite, por ejemplo, que un subdominio como “www.example.com” apunte a “example.com”, facilitando la gestión y administración de dominios.

Cuando un resolutor DNS encuentra un registro CNAME, inicia nuevamente el proceso de resolución para encontrar el registro A o AAAA correspondiente al dominio canónico (el destino final del alias).

(Different Types of DNS Records: A, AAAA, CNAME, MX, and More, n.d.)

¿Para qué se utiliza el comando nslookup?

El comando nslookup es una herramienta de línea de comandos utilizada para consultar información relacionada con el sistema de nombres de dominio (DNS). Su función principal es resolver nombres de dominio en direcciones IP y viceversa, lo que permite realizar búsquedas directas e inversas. Esta capacidad resulta esencial para tareas de diagnóstico de red, verificación de configuración de servidores y resolución de problemas de conectividad. Al ejecutarse en

entornos como el Command Prompt en Windows o el Terminal en sistemas Linux y macOS, nslookup ofrece una interfaz sencilla pero poderosa para interactuar con servidores DNS.

Una de las aplicaciones más comunes del comando es verificar si un dominio está correctamente configurado y si su servidor DNS responde adecuadamente. Por ejemplo, al introducir un nombre de dominio, nslookup puede mostrar la dirección IP correspondiente, lo que permite comprobar si el sitio está accesible. También es posible especificar un servidor DNS alternativo para realizar la consulta, lo cual resulta útil cuando se sospecha que el servidor por defecto presenta fallos o inconsistencias. En caso de que el servidor DNS no responda, el comando mostrará un mensaje de error, lo que puede indicar una interrupción en el servicio o una mala configuración.

Además de las consultas básicas, nslookup permite acceder a distintos tipos de registros DNS, como A, AAAA, MX, CNAME, PTR y SOA. Cada uno de estos registros cumple una función específica dentro del sistema de nombres de dominio. Por ejemplo, los registros A y AAAA vinculan un dominio con una dirección IP (IPv4 e IPv6 respectivamente), mientras que los registros MX indican los servidores responsables del manejo de correo electrónico. El registro PTR, por su parte, es fundamental para las búsquedas inversas y la validación de correos, ya que permite verificar que una dirección IP está asociada correctamente a un nombre de dominio.

El comando puede ejecutarse en dos modos: interactivo y no interactivo. En el modo interactivo, el usuario ingresa nslookup sin argumentos y puede realizar múltiples consultas dentro de la misma sesión, cambiando parámetros como el servidor DNS o el tipo de registro a consultar. En el modo no interactivo, se introduce directamente el dominio o la IP como

argumento, obteniendo una respuesta puntual. Este último modo es más rápido para consultas simples, mientras que el modo interactivo ofrece mayor flexibilidad para análisis detallados.

(Sentika, 2025)

Otros comandos útiles

El conjunto de pruebas realizado sobre los scripts `schedule-task-script.sh`, `process-manager.sh` y `files-script.sh` ha demostrado que todas las herramientas funcionan correctamente en los entornos de sistemas UNIX Solaris y Slackware GNU/Linux. A continuación, se detallan las conclusiones principales de las pruebas realizadas:

Script: `schedule-task-script.sh`

1. **Compatibilidad y Funcionalidad Básica:** El script cumple con su objetivo de permitir la configuración de tareas programadas utilizando cron en ambos sistemas operativos (Slackware y Solaris). La visualización del bloque de ayuda con los parámetros `-h`, `--help` o sin parámetros ha funcionado correctamente, asegurando que los usuarios obtengan información adecuada sobre el uso del script.
2. **Validación de Entradas:** El script maneja adecuadamente la validación del formato de cron. Las entradas con un número incorrecto de campos son correctamente identificadas y el script muestra el mensaje de error correspondiente.
3. **Creación y Modificación de Crontab:** En escenarios donde no existe un crontab previo, el script crea uno nuevo sin problemas. Además, en los casos donde ya existe un crontab, el script añade nuevas entradas sin duplicarlas, lo que asegura la correcta gestión de tareas programadas.
4. **Manejo de Duplicados:** El script es efectivo para detectar y evitar duplicados, lo que garantiza que el crontab se mantenga limpio y ordenado.

Script: process-manager.sh

1. **Interactividad y Ayuda:** El script process-manager.sh proporciona una interfaz interactiva adecuada para gestionar los procesos del sistema. La visualización del bloque de ayuda se realiza correctamente al invocar --help o parámetros no válidos, lo que facilita la interacción con el usuario.
2. **Detección de Comandos del Sistema:** La correcta detección de los comandos ps en ambos sistemas operativos (Slackware y Solaris) demuestra la capacidad del script para adaptarse a las diferencias entre los comandos estándar de POSIX y los específicos de BSD, asegurando que el listado de procesos sea correcto en ambos entornos.
3. **Gestión de Procesos:** La capacidad de listar, buscar y terminar procesos ha sido verificada satisfactoriamente. El script maneja correctamente los PIDs, y en caso de errores, muestra mensajes claros para indicar la causa del fallo, como en el caso de un PID inválido.
4. **Portabilidad:** La portabilidad del script entre los entornos bash (Slackware) y ksh (Solaris) se ha confirmado, lo que lo hace adecuado para su implementación en diferentes sistemas UNIX.

Script: files-script.sh

1. **Gestión de Archivos:** El script ha demostrado ser eficaz para listar los archivos más pequeños en un sistema de archivos, validando correctamente los parámetros de entrada, como el número de archivos y el tamaño máximo permitido. Los sufijos de tamaño no válidos son correctamente identificados y rechazados.

2. **Comportamiento Ante Errores:** El script maneja apropiadamente los errores de entrada, como parámetros incorrectos o insuficientes, y proporciona retroalimentación adecuada al usuario, indicando la causa del error y mostrando el mensaje de uso.
3. **Rendimiento y Presentación:** El listado de archivos más pequeños, ordenados por tamaño, ha sido exitoso tanto en entornos con archivos disponibles como en directorios vacíos. En escenarios sin resultados que mostrar, el script indica de manera clara que no se han encontrado archivos bajo el umbral especificado.
4. **Consistencia en Resultados:** En ambos sistemas operativos, el script ejecuta las pruebas de forma coherente, mostrando los resultados de manera ordenada y precisa.

Las pruebas realizadas aseguran que los scripts `schedule-task-script.sh`, `process-manager.sh` y `files-script.sh` son funcionales y robustos en los entornos de sistemas UNIX Solaris y Slackware GNU/Linux. Los scripts son capaces de gestionar tareas programadas, procesos del sistema y archivos con precisión, validando correctamente los parámetros de entrada y proporcionando respuestas claras ante errores. La compatibilidad entre los diferentes entornos de shell y la correcta adaptación a las diferencias entre Solaris y Slackware refuerzan su portabilidad y utilidad en sistemas de producción. Por lo tanto, estos scripts son herramientas confiables y eficaces para la administración de tareas y procesos en entornos UNIX.

Conclusiones

Durante el desarrollo de este laboratorio, se logró configurar exitosamente el servicio DNS en múltiples entornos virtualizados, incluyendo Solaris, Slackware y Windows Server. Se definieron y probaron registros NS, A, AAAA, MX y CNAME para dominios personalizados, lo que permitió validar la resolución de nombres tanto interna como externa mediante el uso de herramientas como nslookup. Además, se implementaron scripts en Shell para automatizar tareas administrativas, como la programación de procesos, la gestión de servicios y la exploración del sistema de archivos, fortaleciendo así las habilidades en administración de sistemas y automatización.

Este ejercicio permitió comprender de manera práctica el funcionamiento de los protocolos de capa de aplicación, la estructura de archivos de configuración DNS, y la interacción entre servidores y clientes en una infraestructura de red simulada. La experiencia adquirida es fundamental para el diseño y mantenimiento de servicios de red en entornos empresariales reales.

Bibliografia

Akamai Technologies. (2025). *A and AAAA records*. Akamai Cloud Computing Documentation.

<https://techdocs.akamai.com/cloud-computing/docs/a-and-aaaa-records>

Different Types of DNS Records: A, AAAA, CNAME, MX, and more. (n.d.).

<https://qrator.net/library/learning-center/DNS/What-Are-DNS-Records>

Linode. (2022). *Overview of DNS and DNS Records*. Linode Docs.

<https://www.linode.com/docs/guides/dns-overview/#a-and-aaaa>

Sentika, A. (2025, April 28). *What is nslookup command and how to use it*. Hostinger Tutorials.

<https://www.hostinger.com/tutorials/what-is-nslookup>