



**UNIVERSIDAD VERACRUZANA
DE ESTADÍSTICA E INFORMÁTICA**

Carrera: Redes y servicios de cómputo

Tema: Actividades inicio de sistema

Docente: Contreras Vega Gerardo

Alumno: Hernández Sánchez Mauricio

Materia: Administración de servidores

Fecha: 29/09/2023

Unidad de servicio SystemD miapache

Primero se colocó el archivo de apache en la carpeta etc.

```
mauricio2003@debian:~$ cd /etc/systemd/system/
mauricio2003@debian:/etc/systemd/system$ ls
basic.target.wants          getty.target.wants
bluetooth.target.wants      graphical.target.wants
dbus-fi.wl.wpa_supplicant1.service  multi-user.target.wants
dbus-org.bluez.service      network-online.target.wants
dbus-org.freedesktop.Avahi.service  printer.target.wants
dbus-org.freedesktop.ModemManager1.service  sockets.target.wants
dbus-org.freedesktop.nm-dispatcher.service  sshd.service
dbus-org.freedesktop.timesync1.service  sysinit.target.wants
display-manager.service          timers.target.wants
mauricio2003@debian:/etc/systemd/system$ sudo nano miapache.service
[sudo] contraseña para mauricio2003:
```

Después se deben de recargar los archivos de configuración de systemd.

```
mauricio2003@debian:/etc/systemd/system$ sudo systemctl daemon-reload
mauricio2003@debian:/etc/systemd/system$ sudo systemctl start miapache
```

Al realizar el paso anterior, debemos de iniciar el servicio.

```
mauricio2003@debian:/etc/systemd/system$ sudo systemctl start miapache.service
```

Una vez iniciado, se tendrá que habilitar el servicio para que se inicie el arranque.

Por último,
procedemos a verificar el estado del servicio.

```
mauricio2003@debian:/etc/systemd/system$ sudo systemctl status miapache.service
● miapache.service - MI Servidor Web Apache Personalizado
   Loaded: loaded (/etc/systemd/system/miapache.service; enabled; preset: enabled)
   Active: active (running) since Tue 2023-09-19 10:18:39 CST; 1min 45 ago
     Main PID: 3896 (httpd)
       Tasks: 62 (limit: 9002)
      Memory: 26.7M
         CPU: 49ms
    CGroup: /system.slice/miapache.service
            └─3896 /usr/local/miapache/bin/httpd -k start
              └─3897 /usr/local/miapache/bin/httpd -k start
                └─3898 /usr/local/miapache/bin/httpd -k start
                  └─3899 /usr/local/miapache/bin/httpd -k start

sep 19 10:18:39 debian systemd[1]: Starting miapache.service - MI Servidor Web Apache Personalizado...
sep 19 10:18:39 debian apache2ctl[3895]: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
sep 19 10:18:39 debian systemd[1]: miapache.service: Can't open PID file /usr/local/miapache/logs/httpd.pid (yet?) after start: No such file or directory
sep 19 10:18:39 debian systemd[1]: Started miapache.service - MI Servidor Web Apache Personalizado.
mauricio2003@debian:/etc/systemd/system$
```

0. Reto cambio de inicio por defecto:

- Descripción: Verifica cual es el modo en que inicia tu Linux por defecto, cambia el modo y comprueba el cambio. Deja por defecto el modo que prefieras.
- Objetivo: Aprender a cambiar el modo por defecto en SystemD.

Primero tenemos que checar como inicia nuestro Linux, que en este caso inicia con un entorno gráfico, al poner el comando `systemctl get-default` es el que nos indica como inicia nuestro Linux.

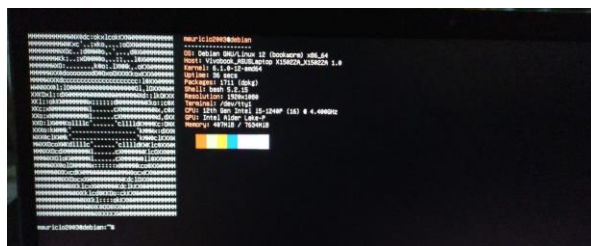
```
mauricio2003@debian:~$ systemctl get-default
graphical.target
```

Después tendremos que poner el comando `sudo systemctl set-default multi-user.target` para cambiar a que Linux inicie sin entorno gráfico.

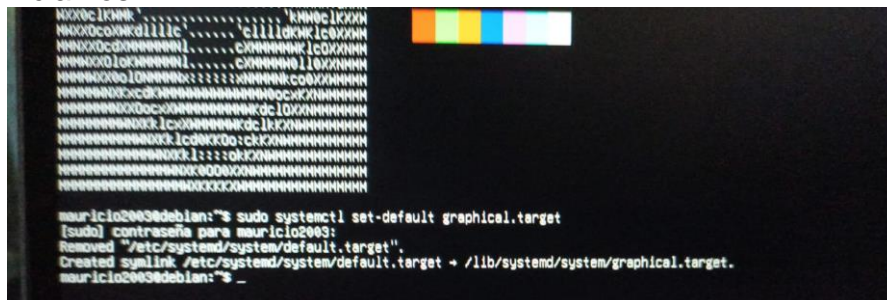
```
mauricio2003@debian:~$ sudo systemctl set-default multi-user.target
[sudo] contraseña para mauricio2003:
Created symlink /etc/systemd/system/default.target → /lib/systemd/system/multi-user.target.
```

Una vez realizado esto, reiniciamos la maquina para ver si funciona.

Al reiniciar nos damos cuenta que la maquina preno sin entorno gráfico.



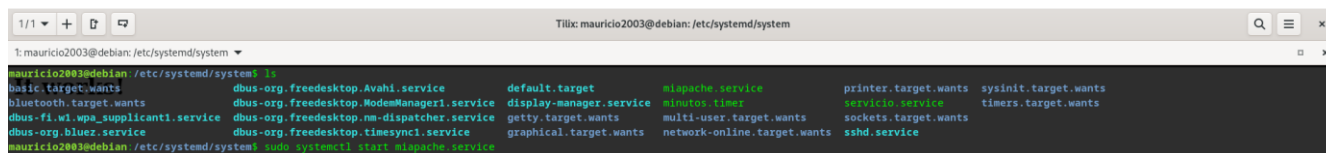
Para volver a establecer que la maquina inicia con entorno grafico ponemos los siguientes comandos y reiniciamos.



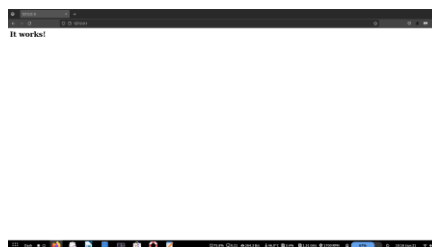
1. Reto del Servidor Web Personalizado:

- Descripción: Crea un servicio para el servidor web instalado desde código fuente.
- Objetivo: Aprender a configurar servicios personalizados en ubicaciones no estándar.

Para hacer esto, primero debo de iniciar algún servidor, en este caso utilice apache. Se debe de poner el código que se muestra continuación.



Después debemos de entrar a nuestro buscador y ponemos la local hosts que en este caso es 127.0.0.1 y tendremos que dar Enter y debe de mostrar lo siguiente.



Tome como ejemplo el script que vimos en clase de mi apache personalizado.

2. Reto del Temporizador:

- Descripción: Escribe un script que genere un archivo de registro con la fecha y hora actual en /var/log/mi-log.log. Luego, deben crear un servicio systemd para ejecutar el script y un temporizador (timer) que ejecute el servicio cada 5 minutos.
- Objetivo: Entender cómo funcionan los temporizadores y cómo pueden ser usados para programar la ejecución de servicios.

Lo primero que se debe de realizar es crear el script en el cual funcionara para que genere un archivo de registro.

```
GNU nano 7.2
#!/bin/bash

tiempo=$(date "+%Y-%m-%d %H:%M:%S")

log_file="/var/log/mi-log.log"

echo "Registro: $tiempo" >> $log_file
```

En segundo se crea el servicio

```
GNU nano 7.2
[Unit]

Description=Servicio para generar registros en /var/log/mi-log.log con temporizador

[Service]
Type=oneshot
ExecStart=/home/mauricio2003/tempo.sh

[Install]
WantedBy=multi-user.target
```

Se tiene que crear el temporizador

```
GNU nano 7.2
[Unit]
Description=Temporizador para ejecutar el servicio cada 5 minutos

[Timer]
OnUnitActiveSec=5m
Unit=servicio.service

[Install]
WantedBy=timers.target
```

Al tener todo creado, se tiene que activar los servicios, para hacer esto se ejecutan los siguientes comandos que se muestran a continuación.

```
mauricio2003@debian: /etc/systemd/system$ sudo systemctl daemon-reload
mauricio2003@debian: /etc/systemd/system$ sudo systemctl enable servicio.service
Removed "/etc/systemd/system/multi-user.target.wants/servicio.service".
Created symlink /etc/systemd/system/multi-user.target.wants/servicio.service -> /etc/systemd/system/servicio.service.
mauricio2003@debian: /etc/systemd/system$ sudo systemctl enable minutos.timer
Removed "/etc/systemd/system/timers.target.wants/minutos.timer".
Created symlink /etc/systemd/system/timers.target.wants/minutos.timer -> /etc/systemd/system/minutos.timer.
mauricio2003@debian: /etc/systemd/system$ systemctl start servicio.service
```

Después procedemos a ejecutar el archivo log para iniciar el registro

```
mauricio2003@debian: ~$ cat /var/log/mi-log.log
Registro: 2023-09-21 10:00:31
Registro: 2023-09-21 10:00:32
Registro: 2023-09-21 10:00:47
Registro: 2023-09-21 10:05:50
Registro: 2023-09-21 10:10:52
Registro: 2023-09-21 10:16:03
Registro: 2023-09-21 10:21:33
Registro: 2023-09-21 10:26:37
Registro: 2023-09-21 10:31:58
Registro: 2023-09-21 10:37:03
mauricio2003@debian: ~$
```

Una forma de ver cuanto tiempo falta para que se realice cada servicio es poniendo este comando, con este comando mostrara los minutos que faltan para que se haga el registro.

```
mauricio2003@debian: /etc/systemd/system$ sudo systemctl status minutos.timer
* minutos.timer - Temporizador para ejecutar el servicio cada 5 minutos
   Loaded: loaded (/etc/systemd/system/minutos.timer; enabled; preset: enabled)
   Active: active (waiting) since Thu 2023-09-21 09:59:18 CST; 2min 2s ago
     Trigger: Thu 2023-09-21 10:03:52 CST; 4min 10s left
    Triggers: * servicio.service

mauricio2003@debian: /etc/systemd/system$
```

3. Reto del Servicio Dependiente:

- Descripción: Configura dos servicios: servicioA y servicioB. servicioB no debe poder iniciar a menos que servicioA esté en funcionamiento.
- Objetivo: Aprender sobre las dependencias entre servicios y cómo configurarlas usando directivas como Requires, After, entre otras.

Primero se crea el archivo del servicioA, en este caso lo cree en la ruta de systemd y quedo así.

```
GNU nano 7.2
[Unit]
Description=Servicio A
After=network.target

[Service]
ExecStart=/etc/systemd/system/servicioA.service
Restart=always

[Install]
WantedBy=multi-user.target
```

Despues se crea el archivo del servicioB, así fue como quedo.

```
GNU nano 7.2
[Unit]
Description=Servicio B
Requires=servicioA.service
After=servicioA.service

[Service]
ExecStart=/etc/systemd/system/servicioB
Restart=always

[Install]
WantedBy=multi-user.target
```

Al tener los dos archivos procedemos a iniciarlos y habilitarlos para que empiecen los servicios, los siguientes comandos que se muestran a continuacion son para habilitar los servicios.

```
mauricio2003@debian: /etc/systemd/system$ sudo systemctl daemon-reload
mauricio2003@debian: /etc/systemd/system$ sudo systemctl enable ServicioA
Failed to enable unit: Unit file ServicioA.service does not exist.
mauricio2003@debian: /etc/systemd/system$ sudo systemctl enable servicioA
Created symlink /etc/systemd/system/multi-user.target.wants/servicioA.service - /etc/systemd/system/servicioA.service.
mauricio2003@debian: /etc/systemd/system$ sudo systemctl start servicioA
mauricio2003@debian: /etc/systemd/system$ sudo systemctl enable servicioB
sudo: systemctl: command not found
mauricio2003@debian: /etc/systemd/system$ sudo systemctl enable servicioB
The unit files have no installation config (WantedBy=, RequiredBy=, Also=,
Alias= settings in the [Install] section, and DefaultInstance= for template
units). This means they are not meant to be enabled using systemctl.

Possible reasons for having this kind of units are:
* A unit may be statically enabled by being symlinked from another unit's
  .wants/ or .requires/ directory.
* A unit's purpose may be to act as a helper for some other unit which has
  a requirement dependency on it.
* A unit may be started when needed via activation (socket, path, timer,
  D-Bus, udev, scripted systemctl call, ...).
* In case of template units, the unit is meant to be enabled with some
  instance name specified.
```

Paso 4: Habilitar y empezar a los servicios

Unidad que hayan creado ambos archivos de servicioB y servicioA.

```
sudo systemctl enable servicioA.serv
sudo systemctl enable servicioB.serv
sudo systemctl start servicioA.serv
sudo systemctl start servicioB.serv
```

Estos comandos habilitaran y comenzaran en hasta que "servicioA" este en funcionamiento.

Paso 4: Comprobar el estado de los servicios

4. Reto de la Restricción de Recursos:

- Descripción: Configurar un servicio que inicie un proceso que consuma una gran cantidad de CPU (por ejemplo, un bucle infinito en Python o Bash). Luego, debes limitar el uso de CPU de ese servicio al 10% utilizando systemd.
- Objetivo: Aprender a gestionar y restringir los recursos (como CPU, memoria) que un servicio puede usar.

Lo primero que realice fue crear un servicio, el cual quedo así:

```
[Unit]
Description=Limitar el uso de cpu a un servicio

[Service]

ExecStart=/bin/bash -c 'while true; do echo "Running..." && echo "$?" > /dev/null; done'
CPUQuota=10%
Restart=always

[Install]
WantedBy=multi-user.target
```

Después se tienen que activar todos los servicios para que inicie el programa.

```
mauricio2003@debian: /etc/systemd/system$ sudo systemctl daemon-reload
mauricio2003@debian: /etc/systemd/system$ sudo systemctl start servicioCpu.service
mauricio2003@debian: /etc/systemd/system$ sudo systemctl status servicioCpu.service
* servicioCpu.service
Loaded: loaded (/etc/systemd/system/servicioCpu.service; disabled; preset: enabled)
Active: active (running) since Mon 2021-09-27 12:19:40 CDT; 19s ago
Main PID: 11291 (bash)
Tasks: 1 (limit: 9802)
Memory: 400.0K
CPU: 1.962s
CGroup: /system.slice/servicioCpu.service
          └─11291 /bin/bash -c 'while true; do echo "Running..." && echo "$?" > /dev/null; done'

Sep 27 12:19:53 debian bash[11291]: Running...
Sep 27 12:19:53 debian bash[11291]: Running...
Sep 27 12:19:53 debian bash[11291]: Running...
Sep 27 12:19:53 debian bash[11291]: Running...
Sep 27 12:19:53 debian bash[11291]: Running...
Sep 27 12:20:06 debian systemd[1]: /etc/systemd/system/servicioCpu.service:1: Assignment outside of section: Ignored
Sep 27 12:20:06 debian systemd[1]: /etc/systemd/system/servicioCpu.service:2: Assignment outside of section: Ignored
Sep 27 12:20:06 debian systemd[1]: /etc/systemd/system/servicioCpu.service:1: Assignment outside of section: Ignored
Sep 27 12:20:06 debian systemd[1]: /etc/systemd/system/servicioCpu.service:2: Assignment outside of section: Ignored
```

Y para comprobar que el servicio este utilizando el 10% del systemd tenemos que poner un top.

```
mauricio2003@debian: /etc/systemd/system$ top

top - 12:21:21 up 30 min, 1 user, load average: 1.17, 0.96, 0.81
Tareas: 312 total, 2 running, 310 sleeping, 0 stopped, 0 zombie
%Cpu(s): 4.0 us, 1.7 sy, 0.0 ni, 94.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7634.1 total, 1126.3 free, 4627.5 used, 3094.1 buff/cache
MiB Intercambio: 7583.0 total, 7583.0 free, 0.0 used, 3006.6 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+  COMMAND
 2960 maurici+  20   0 3399292 567056 118180 S  29.9   7.3  10:21.34 Isolated Web Co
 2348 maurici+  20   0  27.9g 540496 224820 S  16.9   6.9   6:49.52 firefox-esr
   374 root       20   0  241412 116412 114788 S  11.6   1.5   0:08.31 systemd-journal
 1665 maurici+  20   0 6077572 278996 145644 S  10.0   3.6   2:55.83 gnome-shell
 11291 root       20   0  6932  1380  1224 R  10.0   0.0   0:09.28 bash
```

5. Reto del Análisis de Fallos:

- Descripción: Se tiene el siguiente archivo de servicio:

Archivo: enviaCorreo.service

[Unit]

Description=Servicio Fallido

[Service]

ExecStart=/usr/local/enviaCorreo.sh

[Install]

WantedBy=multi-user.target

Archivo enviarCorreo.sh

```
#!/bin/bash
# Intenta enviar el contenido de un archivo como cuerpo de un correo electrónico.
archivo="/etc/mensaje.txt"
destinatario="pato@localhost"
# Se lee el archivo.
contenido=$(cat $archivo)
# Se manda por correo.
echo "$contenido" | sendmail -s "Contenido del archivo" $destinatario
# Se genera una bitácora de la actividad
echo "Correo enviado a $destinatario con el contenido de $archivo"
/var/log/enviar-email.log
```

Identifica y corrige los errores para que el servicio funcione correctamente.

- Objetivo: Familiarizarse con las herramientas de diagnóstico y registro de systemd, como systemctl

status, journalctl, entre otras, y aprender a solucionar problemas comunes.

El error que tenía el script era la redirección del var log, el script y el servicio quedaron de esta manera.

```
GNU nano 7.2
[Unit]
Description=Servicio que envia correo

[Service]
ExecStart=/usr/local/enviarCorreo.sh

[Install]
WantedBy=multi-user.target
```

```
GNU nano 7.2
#!/bin/bash
# Intenta enviar el contenido de un archivo como cuerpo de un correo electrónico.
archivo="/etc/mensaje.txt"
destinatario="pato@localhost"
# Se lee el archivo.
contenido=$(cat "$archivo")
# Se manda por correo.
echo "$contenido" | sendmail -s "Contenido del archivo" "$destinatario"
echo "Correo enviado a $destinatario con el contenido de $archivo" >> /var/log/enviar-email.log
```

Al corregir el script, procedemos a dar inicio a los servicios.

```
mauricio2003@debian:~$ sudo systemctl daemon-reload
mauricio2003@debian:~$ sudo systemctl start enviarCorreo.service
mauricio2003@debian:~$ cat /var/log/enviar-email.log
Correo enviado a pato@localhost con el contenido de /etc/mensaje.txt
mauricio2003@debian:~$
```