Sesión 4

Procesos del Sistema

Son tareas que vienen llevándose a cabo en el servidor, dichas tareas consumen recursos del Sistema (Archivos, Directorios, memoria, disco, servicios, etc).

Un proceso en Linux no es más que un programa en ejecución. Es una instancia en ejecución de un programa. ***Cualquier comando que ejecute inicia un proceso.***

Para ver los procesos de mi usuario (en este caso el usuario **jeanleonardo**) ingrese el comando:

$ ps (ps viene de Process Status)

- El comando **ps** enumera todos los procesos que se están ejecutando actualmente junto con información adicional como **PID** (Identidad del proceso), **TTY**, **ID de usuario**, nombre del comando, cuánto espacio ocupa en la **CPU (en segundos) (TIME)** y muchas otras cosas según las opciones que usamos.
- El comando ps muestra una lista de todos los procesos activos.
- La columna TIME del "comando ps", se refiere al reporte del uso de CPU de un proceso en segundos.
- 00:00:00 se refiere a que el proceso en cuestión ha utilizado "menos de un segundo de CPU" hasta el momento.

Veamos el output del comando **"ps"**:

```
jeanleonardo@linux-oti-uni:~$ ps
    PID TTY          TIME CMD
   1467 pts/0    00:00:00 bash
   2809 pts/0    00:00:00 ps
jeanleonardo@linux-oti-uni:~$
```

- El proceso "bash" se refiere al PROCESO de LOGIN o INICIO DE SESION del usuario (root o común), Esto es porque apenas se loguea un usuario se carga su shell, en este caso el shell es bash.
- Aquí, el último proceso ejecutado fue "ps", el comando ps nos da desde el primer hasta el último proceso QUE SE ESTA EJECUTANDO, donde la primera fila es el primer proceso y la última fila es el último proceso

En este caso el proceso veamos el PID asignado a "ps" es 2804, veamos que ocurre si nuevamente colocamos el comando "ps":

```
jeanleonardo@linux-oti-uni:~$ ps
    PID TTY          TIME CMD
   1467 pts/0    00:00:00 bash
   2804 pts/0    00:00:00 ps
jeanleonardo@linux-oti-uni:~$
```

Vemos que se asignó un nuevo PID a “ps”, ahora su PID es 2804, esto significa que en realidad cuando ejecutamos la primera vez el comando “ps” inicio un proceso con un PID de 2809, luego termino y cuando por segunda vez se ejecuto el comando “ps” se inicio con un proceso con un PID de 2804.

Veamos una comparación entre el antes y después:

```
jeanleonardo@linux-oti-uni:~$ ps
    PID TTY          TIME CMD
   1467 pts/0    00:00:00 bash
   2804 pts/0    00:00:00 ps
jeanleonardo@linux-oti-uni:~$ ps
    PID TTY          TIME CMD
   1467 pts/0    00:00:00 bash
   2809 pts/0    00:00:00 ps
jeanleonardo@linux-oti-uni:~$
```

Donde:

- PID-ID de proceso (Identificador de proceso)
- TTY-Terminal que ejecuta el proceso
- TIME-Tiempo de CPU utilizado en segundos
- CMD-Nombre del proceso

Puede observar que “**bash**” es un proceso que sigue manteniendo su PID con el numero 1467, es decir aún no ha terminado desde que inicio el sistema. Esto es porque apenas se loguea un usuario se carga su shell, en este caso el shell es bash.

Nota: Cada proceso tiene su propio PID, nunca va a encontrar 2 procesos diferentes con el mismo PID

Ver los procesos de un usuario (común o root)

Si desea saber que procesos está ejecutando un usuario, en particular ingrese la orden:

ps -u jeanleonardo (que esta haciendo el usuario **jeanleonardo**)

Veamos una imagen con el output (salida) de este comando (en mi caso abrí una nueva terminal por eso se asignó al bash otro valor de PID)**:**

```
jeanleonardo@linux-oti-uni:~$ ps -u jeanleonardo
    PID TTY          TIME CMD
    895 ?        00:00:00 systemd
    896 ?        00:00:00 (sd-pam)
   1122 ?        00:00:00 sshd
   1123 ?        00:00:00 sshd
   1124 pts/0    00:00:00 bash
   1126 ?        00:00:00 sftp-server
   1141 pts/0    00:00:00 ps
jeanleonardo@linux-oti-uni:~$
```

Pregunta: ¿Que significa el “?” asociado al campo TTY?

Respuesta:

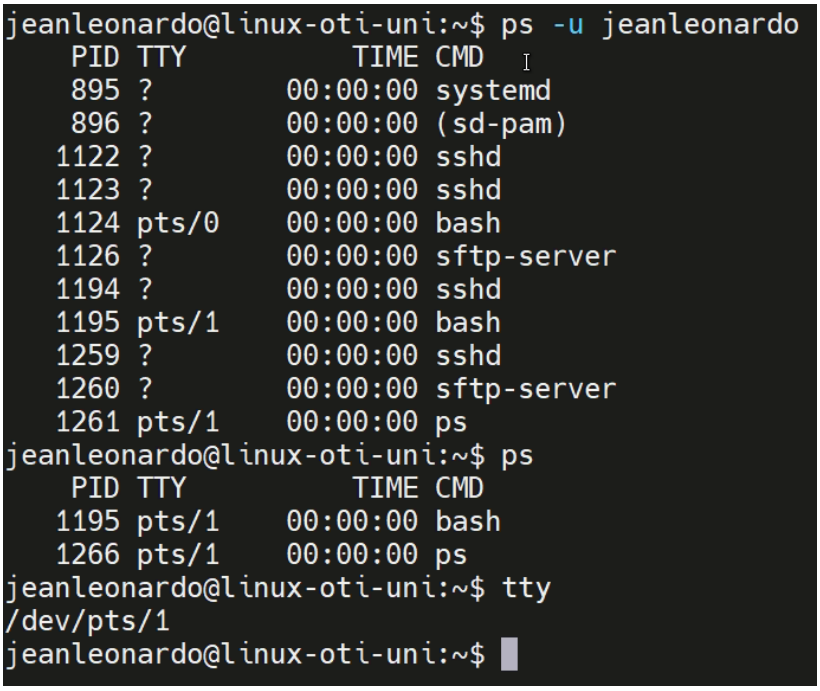
- Los procesos sin una terminal de control asignada muestran un signo de interrogación (?).
- El campo TTY se refiere a la terminal de control. Cuando aparece “?” en ese campo significa que el proceso no tiene terminal de control. Un proceso sin terminal de control es un demonio (daemon). Un demonio o daemon es un proceso es segundo plano, no puede abrir /dev/tty. Y no hay una terminal en la que pueda escribir Ctrl+C para interrumpirlo. El proceso puede haber sido iniciado por otro demonio como **cron**.

Por ejemplo, abra otra terminal, logueese con el usuario “**jeanleonardo**” y de nuevo ejecute el comando “**ps -u jeanleonardo**”:

```
jeanleonardo@linux-oti-uni:~$ ps -u jeanleonardo
    PID TTY          TIME CMD
    895 ?        00:00:00 systemd
    896 ?        00:00:00 (sd-pam)
   1122 ?        00:00:00 sshd
   1123 ?        00:00:00 sshd
   1124 pts/0    00:00:00 bash
   1126 ?        00:00:00 sftp-server
   1194 ?        00:00:00 sshd
   1195 pts/1    00:00:00 bash
   1259 ?        00:00:00 sshd
   1260 ?        00:00:00 sftp-server
   1261 pts/1    00:00:00 ps
jeanleonardo@linux-oti-uni:~$
```

Observe que aparece una nueva terminal (pts/1) esta se refiere a la nueva terminal que hemos abierto con el mismo usuario jeanleonardo.

Importante: ¿Cuál es la diferencia entre utilizar "ps" y "ps -u"?



ps: **muestra todos los procesos llevados a cabo desde la terminal** (del tipo que sea, puede ser pts, tty o incluso "?") **en la que me encuentro, para el usuario con el que estoy logueado actualmente.**

¿Como puedo ver que terminal estoy utilizando?

Tiene 2 formas de saber esto:

1. La 2da columna del comando "who am i" me muestra que terminal estoy utilizando
2. Con el comando "tty"

ps -u NOMBREUSUARIO: muestra todos los procesos llevados a cabo desde todas las terminales de cualquier tipo (tty, pts, ?, etc) del usuario (USUARIO) especificado

Ejercicio de aplicación:

Comando o Instrucción	Explicación o comentario
Abra una terminal e ingrese con el usuario "jeanleonardo" (Cierre las otras terminales que tiene abiertas con el usuario "jeanleonardo"	
$ vi file	Déjelo abierto no guarde ni salga
Abra una terminal e ingrese con el usuario "pizquierdo"	
$ cat >> file.txt (Recuerde con Ctrl+D guardamos o salimos)	Déjelo abierto no guarde ni salga

Abra una terminal e ingrese con el usuario "mgaray"	
$ nano	
Abra una terminal e ingrese con el usuario "root" Divida su pantalla en 4 (Split, 4 terminals mode)	
# ps -u jeanleonardo	Observe
# ps -u pizquierdo	Observe
# ps -u mgaray	Observe

Ver todos los procesos del sistema:

Comando o Instrucción	Explicación o comentario
Deje las terminales abiertas del ejemplo anterior, ubíquese en la terminal del usuario "root"	
# ps -ef	**Ver todos los procesos del sistema** Este comando me **da todos los procesos** llevados a cabo **desde todas las terminales** de cualquier tipo (tty,pts,?,etc) **de TODOS LOS USUARIOS** (root y comunes) que tengamos en nuestro sistema. Observe y ubique los procesos de los usuarios "jeanleonardo", "pizquierdo", "mgaray" y el usuario "root"
# clear	
# ps -ef |more	**Ver todos los procesos del sistema con paginación**

Vista dinámica de todos los procesos que se están ejecutando en tiempo real:

Comando o Instrucción	Explicación o comentario
# top Para salir presionamos la letra "q" del teclado.	Con el comando top obtienes una lista dinámica de todos los procesos activos.
Vaya al usuario jeanleonardo y ejecute el comando: $ vi file	¿Que observa en la ventana de top? Observe rápidamente.

Con el **comando top** **obtenemos una lista dinámica de todos los procesos activos** llevados a cabo **desde todas las terminales** de cualquier tipo (tty,pts,?,etc) **de TODOS LOS USUARIOS** (root y comunes) que tengamos en nuestro sistema.

Este comando nos el ultimo proceso ejecutado dinámicamente en la primera fila (aquí el orden es inverso respecto a los casos anteriores como en ps y ps -u)

El comando **top** se usa para mostrar los procesos activos de Linux. Proporciona una vista dinámica en tiempo real del sistema en ejecución. Por lo general, este comando muestra la información resumida del sistema y la lista de procesos o subprocesos que actualmente administra el kernel de Linux.

El comando top nos da también el uso en porcentaje de CPU en segundos, el uso de la memoria RAM y la memoria SWAP (memoria virtual). Aquí en estos 2 últimos apartados, cuando notemos que la memoria SWAP este siendo USADA (used), tenemos que preocuparnos, porque significa que ya se utilizó toda la memoria RAM (es decir se sobrecargo) y el sistema se está yendo a utilizar la memoria virtual, como la memoria RAM ya se usó completamente y se está recurriendo a la memoria virtual, puedo deducir que alguna de mis aplicaciones está consumiendo DEMASIADOS RECURSOS.

Filtrar Procesos:

Con esto podemos ubicar procesos que quisiéramos matar o detener.

Si desea buscar un proceso en particular ingrese el siguiente comando:

Comando o Instrucción	Explicación o comentario
Ingrese como usuario root	
# ps -ef| grep bash	Esto me indicaría quienes están conectados o tienen abierta su terminal.
# ps -ef| grep vi ¿Que es “grep --color=auto bash”? Esto solamente resalta las coincidencias con la palabra que buscamos a través de grep, en este caso resaltará de color rojo donde encuentre la palabra “vi” y nos lo mostrará.	Indica que usuarios están ejecutando “vi”
# ps -ef| grep cat	Indica que usuarios están ejecutando “cat”
# ps -ef| grep nano	Indica que usuarios están ejecutando “nano”

Vista dinámica de todos los procesos que se están ejecutando en tiempo real de un único USUARIO EN ESPECIFICO:

Comando o Instrucción	Explicación o comentario
Ingrese como usuario root	
# top -u jeanleonardo	Con esto **obtenemos una lista dinámica de todos los procesos activos** llevados a cabo **desde todas las terminales** de cualquier tipo (tty,pts,?,etc) **del usuario (USUARIO) especificado**, en este caso el usuario **jeanleonardo**
Vaya al usuario jeanleonardo y ejecute el comando: $ vi file	¿Que observa en la ventana de top? Observe rápidamente.

$ ping 8.8.8.8	

El comando "top-u USUARIO" nos da desde el primer hasta el último proceso QUE SE ESTA EJECUTANDO, donde la primera fila es el primer proceso y la última fila es el último proceso

Como matar un proceso desde "top -u" (también lo puede aplicar el comando top sin opciones ni argumentos):

Comando o Instrucción	Explicación o comentario
Esta con el usuario root en una terminal y el usuario jeanleonardo con el vi abierto.	Se sugiere que utilice el modo Split de 2 terminales.
Vaya a la terminal del usuario root: # top -u jeanleonardo	
Dentro de la ventana presione la letra "k" (kill) e ingresamos el PID (Process ID), luego damos ENTER 2 veces **En este caso ingrese el PID del proceso ejecutado por "vi"**	Observe que ocurre con el editor vi en la terminal de jeanleonardo. ¿Se cierra el editor vi?

Matar un proceso de la línea de comandos

Para "terminar" un proceso en ejecución (kill, matar) utilize el comando kill, con esto solo terminara la ejecución del programa (por ejm: cat, vi, ping, etc)

Nota: Para matar un proceso por lo general debe este logueado con el usuario ROOT o equivalente.

Comando o Instrucción	Explicación o comentario
Esta con el usuario root en una terminal y el usuario jeanleonardo con el vi abierto.	Se sugiere que utilice el modo Split de 2 terminales.
# ps -u jeanleonardo	
# kill -9 1898 (PID de vi en este caso)	Matamos el proceso de vi
# kill -9 1389 (PID de bash en este caso)	Si desea matar al usuario de su login, para eso tendra que hacer kill al proceso -bash (kill -9 bash)

Manejo de los registros del sistema:

Sirve para ver que han venido haciendo los usuarios o que acciones se han llevado a cabo en nuestro servidor como procesos de autenticación, por ejemplo.

Comando o Instrucción	Explicación o comentario
Ingrese con el usuario "root"	
# cd /var/log	
# ls -l	Ubique los archivos auth.log y syslog en esta lista. Ambos son los que contienen los eventos

	recientes, vea la fecha y hora de última modificación y compruébelo. Syslog es un archivo donde se colectan todos los log.
# cat auth.log	Este archivo almacena los eventos relacionados con mecanismos de autorización, por ejemplo, cuando un usuario inicia sesión en el sistema. Suele incluir detalles sobre el mecanismo utilizado y el resultado (si la autenticación ha sido correcta o no).
Pruebe abriendo otra terminal y logueese con un usuario no existente, por ejemplo: User: hacker Password: 123456	
# cat auth.log ¿Observe las últimas líneas del archivo que sucede? Se reporta el intento fallido de autenticación.	

Automatización de ejecución de tareas con cron:

¿Qué es cron?

El nombre cron viene del griego chronos que significa “tiempo”. En el sistema operativo Unix, cron es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab.

Cron administra procesos en segundo plano (demonios) a intervalos establecidos.

¿Como funciona cron?

El demonio cron inicia de /etc/rc.d/ o /etc/init.d dependiendo de la distribución. **Cron se ejecuta en el background (2do plano), revisa cada minuto la tabla de tareas crontab /etc/crontab o en /var/spool/cron en búsqueda de tareas que se deban cumplir.** Como usuario podemos agregar comandos o scripts con tareas a cron para automatizar algunos procesos. Esto es útil por ejemplo para automatizar la actualización de un sistema o un buen sistema de respaldos.

¿Qué es Crontab?

Crontab es un simple archivo de texto que guarda una lista de comandos a ejecutar en un tiempo especificado por el usuario. Crontab verificará la fecha y hora en que se debe ejecutar el script o el comando, los permisos de ejecución y lo realizará en el background (segundo plano). Cada usuario puede tener su propio archivo crontab, de hecho el /etc/crontab se asume que es el archivo crontab del usuario root, cuando los usuarios normales (e incluso root) desean generar su propio archivo de crontab, entonces utilizaremos el comando crontab.

Crontab es un archivo de texto que guarda la lista de comandos a ejecutar en el tiempo especificado por el usuario. Normalmente en /etc/crontab o /var/spool/cron.
Crontab es la forma mas fácil de manejar tareas con cron. Se pueden emplear como un usuario común o como superusuario (root).

¿Qué es un script?

- Un script no es más que un archivo que contiene un conjunto de órdenes para realizar una acción.
- Un script es un archivo que contiene comandos u ordenes que se van ejecutando de manera secuencial

¿Como creamos un script?

Hay varios métodos, pero este caso veremos el siguiente método.

Método 1: Usar el editor de texto predeterminado (vi es un editor de texto predeterminado, es decir ya vino instalado en nuestro sistema). Para crear el script nos basta con el editor de texto (vi por ejemplo) que venga en nuestra distro (distribución) de Linux.

Pasos de creación:

1.Cree un archivo de texto que tenga una extensión "**.sh**". Luego escriba un script simple.

vi copiasemanal.sh

2.Guardar el archivo (con **:wq!**)

3.Otorgue permisos de ejecución al archivo creado, ya sea al propietario (usuario dueño), grupo propietario u otros, usted elija a quien dar permisos de ejecución.

chmod u+x copiasemanal.sh

En este caso le está dando permiso de ejecución solo al propietario (usuario dueño)

3.Ejecutamos el script anteponiendo "./" y luego el nombre del archivo con extensión **.sh**, es decir ingresamos lo siguiente y damos ENTER:

./copiasemanal.sh

Nota: Para ejecutar un archivo anteponemos "./" seguido de su nombre.

Veamos un ejemplo práctico:

Comando o Instrucción	Explicación o comentario
Entramos como usuario ROOT	
# cd /root	

# **vi copiasemanal.sh** echo "Realizando COPIAS de los USUARIOS...!" cp -rf /home/* /root/backups (**OJO**: solo copiara si existe el directorio backups dentro de /root, si no existe el directorio backups dentro de /root BOTARA ERROR!!) echo "Copia realizada!" **:wq!** → Guardar y salir.	Procederemos a crear un script que copiará todo lo que esta dentro del directorio /home a un directorio backups de ruta /root/backups
# ls -l	Vemos que el archivo "copiasemanal.sh" no tiene permisos de ejecución, necesitamos permisos de ejecución para poder ejecutar el archivo (script)
Para el propietario (usuario dueño) hacemos: # chmod u+x copiasemanal.sh	Habilitamos este permiso para que cron (que lo usaremos más adelante) pueda ejecutar las tareas que ahí definimos, de lo contrario sin permisos no podrá ejecutar.
Verificamos: #ls -l	
#cd /root #mkdir backups	Creamos el directorio backups en /root si no existe, de existir omita este paso.
Si nosotros queremos ejecutar el script copiasemanal.sh y las tareas que tiene dentro Bastaría ubicarnos dentro de /root y realizar lo siguiente: # **./**copiasemanal.sh	Se realiza la copia **¿Profesor porque utiliza "./" para ejecutar el script?** Recuerde el significado especial del punto (.) en el sistema de archivos, hace referencia a la ruta donde actualmente nos encontramos, es decir en este caso " ./copiasemanal.sh = /root/copiasemanal.sh " como vemos hace referencia a la ruta absoluta. **Para ejecutar un script o archivo debemos colocar la ruta absoluta hacia donde se encuentra y dar Enter.**
# cd /root/backups #ls -l	Comprobemos y vemos que efectivamente se realizaron las tareas que especificamos en copiasemanal.sh

Observación:

Acá nos damos cuenta que MANUALMENTE estamos ejecutando "copiasemanal.sh" a través del comando "./copiasemanal.sh" para que se realicen las tareas especificadas en el script "copiasemanal.sh". La idea de usar CRON es automatizar esto y hacer que "copiasemanal.sh" se **ejecute AUTOMATICAMENTE** (en segundo plano) en una fecha y hora que especifiquemos, así automatizamos la ejecución de tareas.

Importante:

- Un proceso es un "comando" bajo ejecución
- Un comando es una solicitud para realizar una operación o ejecutar un programa. Un proceso es un programa o comando que realmente se está ejecutando en la computadora. Usted utiliza comandos para decirle al sistema operativo qué tarea desea que realice.

Editando el archivo crontab:

El comando empleado para editar el archivo crontab es el siguiente:

crontab -e

Seleccione su editor favorito. Se recomienda usar el vi. Si desea seleccionar otro editor puede utilizar el comando **export EDITOR=nano**

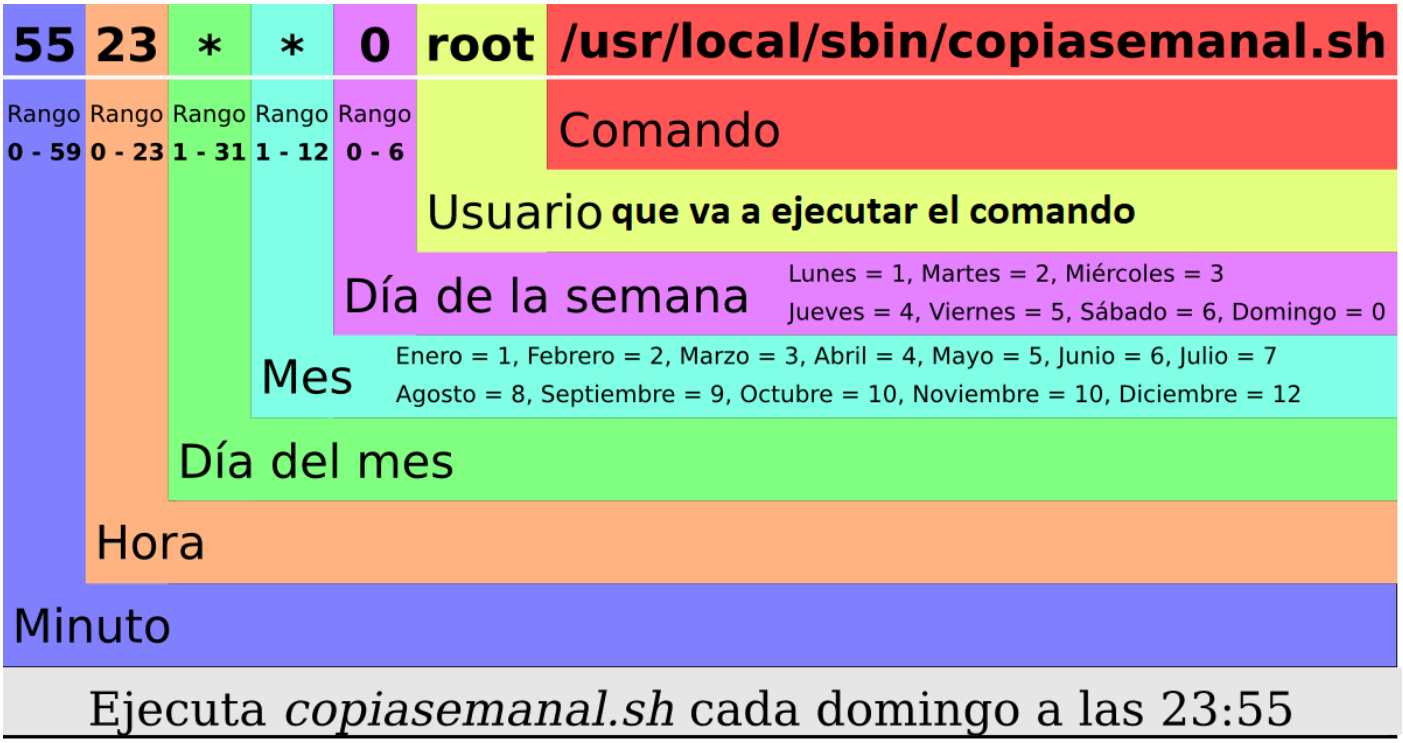
Listar contenido crontab:

El comando empleado para conocer que sentencias están en crontab es:

crontab -l

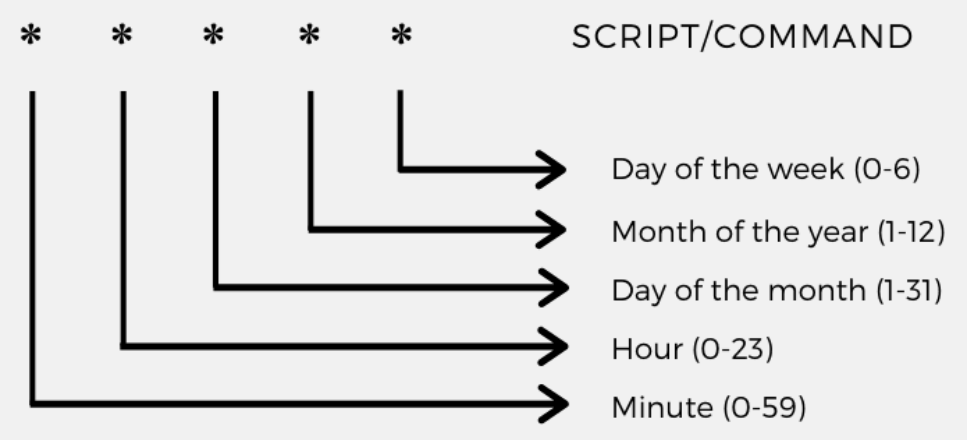
Si no existen comandos que ejecutar el listado indicara la no existencia de los mismos para el usuario en concreto.

Sintaxis de una sentencia dentro del archivo crontab:



Puede observar que el comando en este caso hace referencia a la ejecución de un script (terminación **.sh**)

La sintaxis que usaremos será la siguiente:



Con un asterisco * se indica todo o para todo. Es decir si observa un * en el campo de los minutos se refiere a todos los minutos, similarmente si ve el * en el campo de la hora, el día del mes, el mes del año, el día de la semana; se referirá a todas las horas, a todos los días del mes, a todos los meses del año, a todos los días de la semana; respectivamente.

También se admiten rangos en cada uno los campos:

- Un guion **(-)**
  Por ejemplo: 0-15 → indica un intervalo de valores: todos los valores de 0 hasta 15
- Una coma **(,)**
  Por ejemplo: 15,30,45 → indica unos valores determinados, solo esos tres valores: 15, 30 y 45
- Una barra **(/)**
  Por ejemplo: 1-10/2 → un incremento de valores: valores en incremento de 2 en 2 , empezando desde 1, es decir: 1,3,5,7,9

Ejemplos Prácticos:

Hagamos una prueba de crontab ejecutando línea de comandos sin necesidad de haber elaborado un script, sino defrente especifiquemos el comando a realizar.

¿Cuál es la diferencia de hacerlo con script y hacerlo comando por comando?

Simple, en un script yo puedo colocar varias tareas o comandos, luego voy a crontab -e, especifico la sentencia en el archivo crontab y ejecuto todas las tareas o comandos que están dentro del script en un solo paso

Haciéndolo comando por comando directamente, por ejemplo:

0-59 * * * * date >> /home/rrios/tiempo.txt

solo se ejecuta una tarea o comando que es "date"

Veamos un ejemplo:

Comando o Instrucción	Explicación o comentario
Entramos como usuario ROOT	
# cd /root	
# crontab -e Por defecto le abrirá el editor nano, si desea cambiarlo a vi, ejecute el siguiente comando: **export EDITOR=vi**	Sirve para editar las entradas crontab del usuario que esta logueado ACTUALMENTE
Dentro del archivo crontab ingrese la siguiente sentencia, note que se sigue la sintaxis especificada en pasos anteriores: **0-59 * * * * date >> /home/rrios/tiempo.txt** **:wq! → Guardar y salir (en vi)** **Ctrl + X → y → Enter (En nano)**	OJO cuando especifiques lo campos de la sentencia del archivo crontab, estos van separados por UN espacio, como se ve en: "0-59 * * * *" **Esta sentencia ejecutará el comando date cada minuto (esto viene especificado por 0-59) y redirigirá la salida agregándola (no sobrescribiéndola) al archivo tiempo.txt** Debido al >> se agrega cada output al archivo y no sobreescribe (como con >) el output a todo el archivo (estos son conceptos de redirección de salida)
# crontab -l	Lista todas las entradas o sentencias de crontab **del usuario que esta logueado ACTUALMENTE**
# crontab -u USERNAME -l	Lista todas las entradas de crontab del usuario especificado en **USERNAME**
# cd /home/rrios	
Abra una terminal para el usuario “rrios” y divida su pantalla en 2 terminales	
Con rrios ejecute el comando: $ cat tiempo.txt	Vemos que efectivamente se esta ejecutando el comando (un solo comando) en segundo plano, esto debido al Daemon o demonio cron, que ejecuta tareas especificadas en segundo plano.

Posible confusión a la hora de especificar una sentencia en el archivo crontab:

Sea:

*** * 13 * 5 command**

- 13 representa el "día del mes" (cada día 13 del mes)
- 5 representa el "día de la semana" (cada viernes)

¿Si ya especifico el día del mes, especificar el día de la semana no sería redundante?

La página de manual vinculada desde esa respuesta es bastante clara de que no se excluyen mutuamente: "Nota: **el día de ejecución de un comando se puede especificar en los dos campos siguientes: 'día del mes' y 'día de la semana'**. Si ambos campos están restringidos (es decir, no contienen el carácter *), el comando se ejecutará cuando cualquiera de los campos coincida con la hora actual. Por ejemplo:

30 4 1,15 * 5 command

Esta sentencia especificada en contrab haría que el comando se ejecutara a las **4:30** los días **1 y 15** de cada mes, además de **todos los viernes** (especificado por **5**) ". Acá por ejemplo se está definiendo el "**día del mes**" como una lista es decir **1,15**

La **especificación POSIX para crontab**, que está redactada en el lenguaje de un estándar, con el objetivo de minimizar la ambigüedad, tiene probablemente la explicación más clara (énfasis agregado, párrafo dividido para mayor claridad):

- La especificación de días se puede realizar mediante dos campos (día del mes y día de la semana).
- Si el mes, el día del mes y el día de la semana son todos caracteres <*>, todos los días coincidirán.
- Si se especifica el mes o el día del mes como elemento o lista, pero el día de la semana es un <*>, los campos de mes y día del mes especificarán los días que coinciden.
- Si tanto el mes como el día del mes se especifican como un <*>, pero el día de la semana es un elemento o una lista, solo coinciden los días de la semana especificados.
- Por último, si se especifica el mes o el día del mes como elemento o lista, y el día de la semana también se especifica como elemento o lista, cualquier día que coincida con el mes y el día del mes o el día de la semana, será emparejado.

Veamos una imagen ejemplificado lo anterior y centrándonos en los campos del “día del mes” y “día de la semana”:

Dia del mes	Dia de la semana	Accion
*	*	Se ejecutara TODOS los "dias del mes" y TODOS los "dias de la semana"
*	elemento o lista	Se ejecutara los "dias de la semana" especificados como elemento o lista
elemento o lista	*	Se ejecutara los "dias del mes" especificados como elemento o lista
elemento o lista	elemento o lista	Se ejecutara los "dias del mes" y "dias de la semana" especificados como elemento o lista

Ejemplos de sentencias crontab:

Recuerde:

* (asterisco) = siempre o para todo

Ejemplo 1:

*** * * * * comando/script**

Significa que el comando o script se ejecutará cada minuto de cada hora de cada día de cada mes y cada día de la semana.

Ejemplo 2:

0 * * * * comando/script

Esto significa que el comando o script se ejecutará siempre y cuando los minutos sean 0 (cada hora)

Ejemplo 3:

0 1 * * * comando/script

Esto significa que el comando o script se ejecutará siempre a la 1 en punto. (Recordar que la hora esta especificada en formato de 24 horas, es decir este 1 significa 1 de la mañana, ya que 13 seria 1 de la tarde)

Ejemplo 4:

*** 1 * * * comando/script**

Esto significa que el comando o script se ejecutará cada minuto cuando la hora sea la 1. Entonces, por ejemplo, el comando se ejecutará a las 1:00, 1:01, ... 1:59.

Ejemplo 5:

09 04 1 6 1 comando/script

El comando o script se ejecuta a las 04:09 am el primero de Junio y cada Lunes de Junio

Ejemplo 6:

***/1 * * * * eject**

***/1 * * * * eject -t**

Recordar que la barra representa un incremento de valores, en este caso valores en incremento de 1 en 1 (***/1**) , empezando desde 0, ósea está incrementando el campo de los minutos de 1 en 1.

En conclusión significa que el comando se ejecutará **cada 1 minuto** (***/1**).

- El comando eject expulsa la bandeja o lectora de CD del computador o laptop y la deja ahí.
- El comando eject -t recoje la bandeja o lectora de CD del computador o laptop y la guarda hacia adentro.

Ejemplo 7:

***/8** * * * * **comando/script**

Significa que el comando o script se ejecutará cada 8 minutos.

Ejemplo 8:

*** */8 * * * comando/script**

Significa que el comando o script se ejecutará cada 8 minutos.

Ejemplo 9:

30 22 * * 0 comando/script

Significa que el comando o script se ejecutará a las 10:30 pm el Domingo.

Ejemplo 10: Podemos ejecutar comandos directamente desde el crontab sin necesidad de un script.

0-59 * * * * date >> /home/rrios/tiempo.txt

Este ejemplo ya lo desarrollamos en un ejercicio práctico anteriormente.
Esta sentencia ejecutará el comando date cada minuto (esto viene especificado por 0-59) y redirigirá la salida agregándola (no sobrescribiéndola) al archivo tiempo.txt

Ejemplo 11:

05 20 * * * tar cvfz /home/rrios/**tiempo.tar.gz** /home/rrios/**tiempo.txt**

Esta sentencia significa que a las 20:05 pm de todos los días se empaqueta y comprime el archivo **tiempo.txt** y el archivo resultante se guarda con el nombre **tiempo.tar.gz**

Ejemplo 12: En el archivo contrab podemos especificar varias sentencias no necesariamente una sola, veamos el ejemplo a continuación:

0 0 1 5,10 * /home/jperez/respaldo.sh
0 0 1 mayo,octubre * /home/jperez/respaldo.sh

Esto significa que el script respaldo.sh se ejecutará dos veces al año, es decir, 01 de mayo a medianoche, y 01 de octubre a medianoche.

Ejemplo 13:

0 0 * * 6 /home/jperez/backup.sh

Esto significa que el script backup.sh se ejecutará todos los sábados a la medianoche.

Ahora probemos crontab para que realice las tareas del script que hicimos anteriormente pero que las ejecute automáticamente y no tengamos que ejecutar el script manualmente.

Comando o Instrucción	Explicación o comentario
# cd /root/backups # rm -rf *	Primero, borramos lo que hay dentro de /root/backups, ya que el directorio se llenó debido a que ejecutamos el script manualmente en pasos anteriores. Borra todo el contenido sin preguntar nada, a diferencia de rm -r que si le preguntará, rm -rf es muy efectivo
# crontab -e Comente (anteponiendo #) o borre la entrada anterior que hicimos y colocamos: 45 11 * * * /root/copiasemanal.sh :wq! → guardar y salir	Esto hará que a las **11:45 todos los días**, se ejecute el script **copiasemanal.sh** y por consecuencia se realicen las tareas especificadas en dicho SCRIPT Coloque una hora cercana a su hora actual, para que note que se ejecuto el script.
# ls -l	Compruebe, aun no hay nada,.
# ls -l	Esperamos y listo, se ejecuto el **script** en segundo plano.

Veamos otro ejemplo, en el cual utilizaremos el comando **tar**:

Comando o Instrucción	Explicación o comentario
# cd /root	
# crontab -e Comente (anteponiendo #) o borre la entrada anterior que hicimos y colocamos: **52 11 * * * tar cvfz /home/rrios/tiempo.tar.gz /home/rrios/tiempo.txt** :wq! → guardar y salir	Esta sentencia significa que a las 11:52 am, de todos los días se empaqueta y comprime el archivo **tiempo.txt** y el archivo resultante se guarda con el nombre **tiempo.tar.gz** Coloque una hora cercana a su hora actual, para que note que se ejecutó el script.
# cd /home/rrios # ls -l	Esperemos que llegue la hora que asignamos, verificamos y vemos que efectivamente se ejecuto el **comando** en segundo plano.

Enviar emails localmente desde la consola o terminal de Ubuntu

Ejercicio:

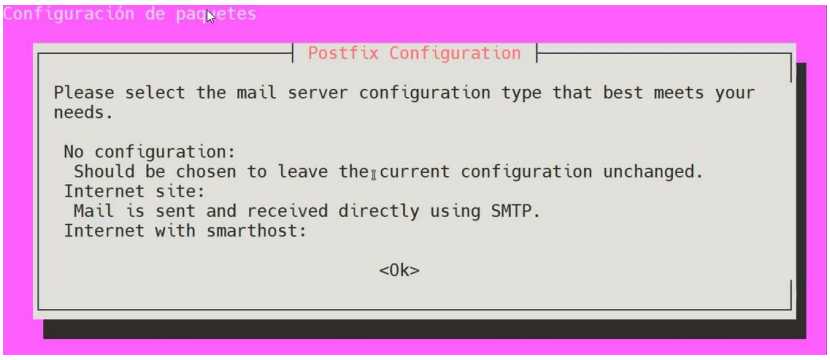
Crear un crontab que envié un correo con el mensaje "Sistema funcionando", usted programe la hora y compruebe de que efectivamente recibió el correo

Solución:

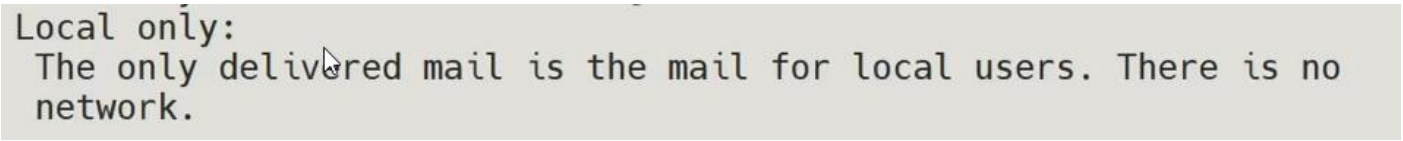
Ingrese como usuario **root** e instale el programa de envío de correo y sus utilidades

 # apt install mailutils

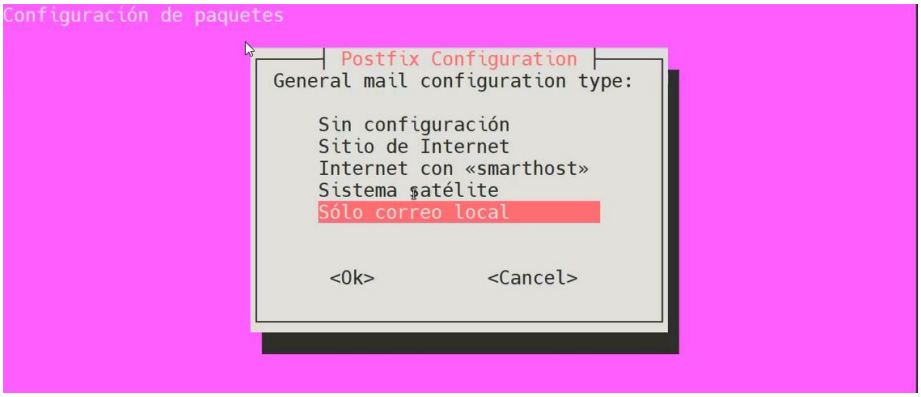
En el proceso de instalación le aparecerán las siguientes ventanas:



Desplácese con las flechas y lea los diferentes tipos de configuración, nosotros utilizaremos la configuración local (para los usuarios (común o root) locales que tenemos creados en el sistema):

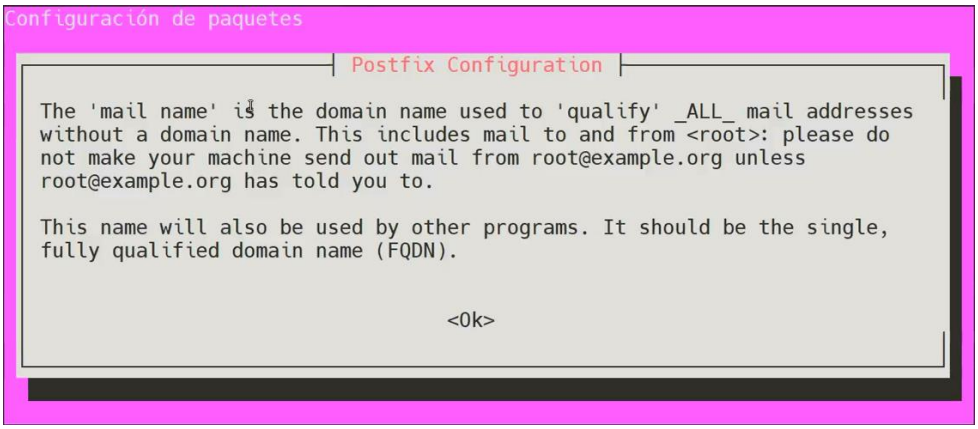


Sitúese sobre "Ok" y de "Enter", le aparecerá la siguiente ventana:

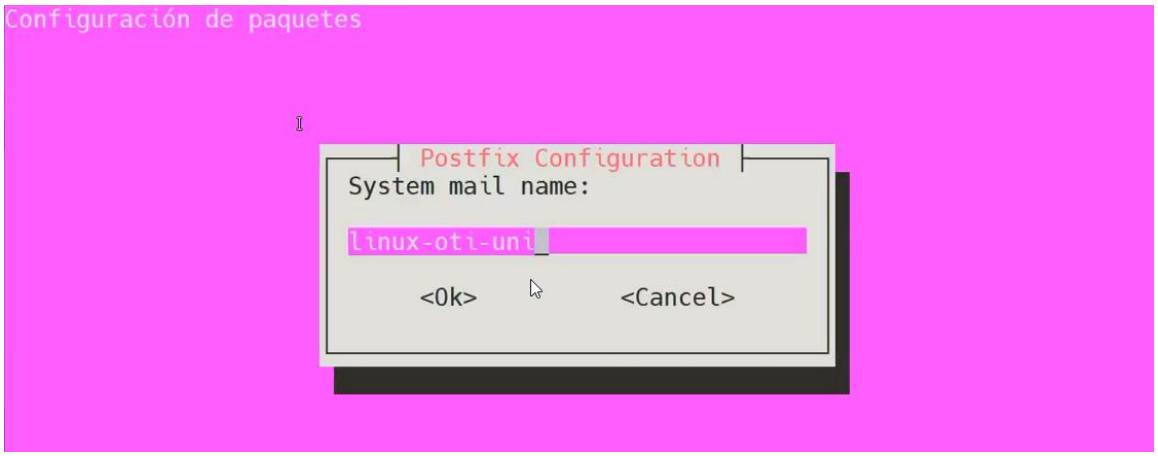


Seleccione "Solo correo local" y de "Enter" sobre "Ok"

De nuevo Enter sobre “Ok”



Le aparecerá la siguiente ventana:



Por defecto, su nombre de dominio será el nombre del host de su sistema (hostname), en este caso mi hostname es “linux-oti-uni”, de Enter sobre “Ok”

Y listo, ahora espere que termine la instalación y regrese al prompt (#).

¿Como sé que hostname tiene mi sistema?

Simple, ejecuta el comando "hostname"

 # hostname

linux-oti-uni

Tambien puede usar:

 # hostnamectl

El campo static hostname le indicara el hostname que tiene su sistema.

¿Como enviamos un correo a un usuario?

En este caso enviaremos un correo desde el usuario que estoy logueado actualmente (que es el usuario ROOT) al usuario "jeanleonardo"

Comando o Instrucción	Explicación o comentario
Entre como **root**	
Ingrese el siguiente comando: # mail NOMBREUSUARIO # mail jeanleonardo **Subject: Hola Jean (El asunto solo tiene una linea, usted da ENTER y ahora llenara el cuerpo del correo)** Presiona Enter y llene el cuerpo del correo (podemos escribir varias cosas y saltar de linea dando ENTER) **Estoy en clase de Linux** Para enviar el correo presione "Ctrl + D" y sale al prompt	
Como vemos si el correo llego a jeanleonardo? Nos logueamos con jeanleonardo $ mail Reconozca y observe el prompt de correo: **?**	Verifique que le llego el correo de parte del usuario “root”
? 1	Ingresamos y damos **Enter** para ver el primer correo
Para responder al que me envio el correo que vimos anteriormente, seguido a esto hacemos: ? r	Ingresamos y damos **Enter** para contestar el correo. Escriba su mensaje y presione “Ctrl+D” para finalizar. Para salir del prompt de correo ingrese “quit” y presione Enter
Vaya al usuario **root**	
# mail	Verifique que le llego el correo de parte del usuario “jeanleonardo”

Creación del crontab

0-59 * * * * mail -s "sistema funcionando" jeanleonardo@linux-oti-uni

Esta sentencia enviará un correo al usuario jeanleonardo con el mensaje “sistema funcionando” cada minuto (esto viene especificado por 0-59).

linux-oti-uni es mi nombre de host y por eso fue colocado en el **campo del dominio**

Comando o Instrucción	Explicación o comentario
Entre con el root	
# crontab -e Comente (anteponiendo #) o borre la entrada anterior que hicimos y colocamos: **0-59 * * * * mail -s "sistema funcionando" jeanleonardo@linux-oti-uni** **:wq! → Guardar y salir (en vi)** **Ctrl + X → y → Enter (En nano)**	Esta sentencia significa que a las 11:52 am, de todos los días se empaqueta y comprime el archivo **tiempo.txt** y el archivo resultante se guarda con el nombre **tiempo.tar.gz** Coloque una hora cercana a su hora actual, para que note que se ejecutó el script.
Logueese con el usuario jeanleonardo # mail	Esperemos que llegue la hora que asignamos, verificamos y vemos que efectivamente se están recibiendo los correos.

Instalación, configuración de Webmin

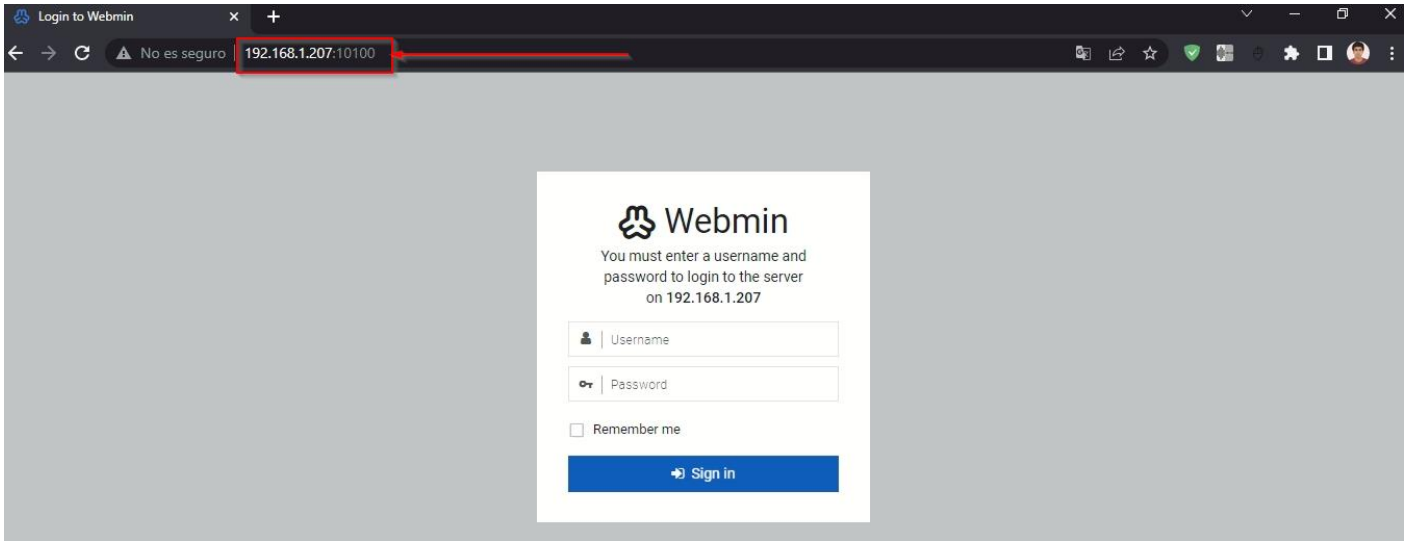
Webmin sirve para administrar el servidor Linux en modo gráfico.

Webmin es una herramienta de administración de sistemas basada en web para servidores y servicios similares a Unix con aproximadamente 1,000,000 de instalaciones anuales en todo el mundo.

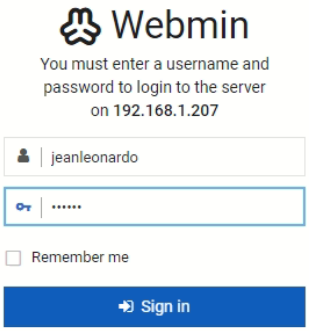
Comando o Instrucción	Explicación o comentario
Ingrese como el usuario root	
Recuerda que como utilizando SSH para controlar remotamente su servidor y MobaXterm, puede copiar y pegar texto en la terminal. Para esto abra su navegador, en el buscador de Google busque: Webmin , luego Downloading and Installing, vaya al apartado de manual, identifique “tar”, click derecho y “Copiar dirección de enlace” El enlace es: **https://www.webmin.com/download/webmin-current.tar.gz**	
# cd /root/Descargas/	
#wget https://www.webmin.com/download/webmin-current.tar.gz	Vemos que descargaremos un directorio que fue empaquetado y comprimido (esto debido a la terminación en .tar.gz) desde el URL. Para esto vemos que utilizamos "wget", **que es "wget"?** wget es una herramienta utilizada a través de la línea de comandos que permite descargar archivos mediante los protocolos web más utilizados (HTTP, HTTPS, FTP, FTPS).

# ls -l	Vemos que se descargó el directorio que fue empaquetado y comprimido. Recuerde los conceptos de la sesión 3.
# tar tfz webmin-current.tar.gz	Vemos el contenido del archivo **.tar.gz**
# tar xvfz webmin-current.tar.gz	Descomprimimos y desempaquetamos el contenido del archivo **.tar.gz**
# ls -l	Observe que se creó un directorio **webmin-2.013**
# cd webmin-2.013	
# ls -l | grep setup	Observe y ubique el archivo **setup.sh**, reconozca que es un script
# cat setup.sh	
# ./setup.sh Siga los siguientes pasos: Config file directory [/etc/webmin]: **Presione Enter** Log file directory [/var/webmin]: **Presione Enter** Full path to perl (default /usr/bin/perl): **Presione Enter** Web server port (default 10000): **10100** (generalmente es 10000) **Login name** (default admin): **jeanleonardo** **Login password**: **otiuni** Password again: **otiuni** Use SSL (y/n): **n** Start Webmin at boot time (y/n): **y** Listo, ahora espere que termine y lo regresará al prompt	Ejecutamos el script para inicial la instalación de Webmin.
Desde un navegador, ingrese en el campo de URL: http://**IPDESUSERVIDOR**:10100/ Recuerde el modo “Adaptador Puente” (Sesión 1) y para que servía. Para conocer la IP de su servidor ingrese el comando “**ip a s**” En mi caso sería lo siguiente: **http://192.168.1.207:10100/**	

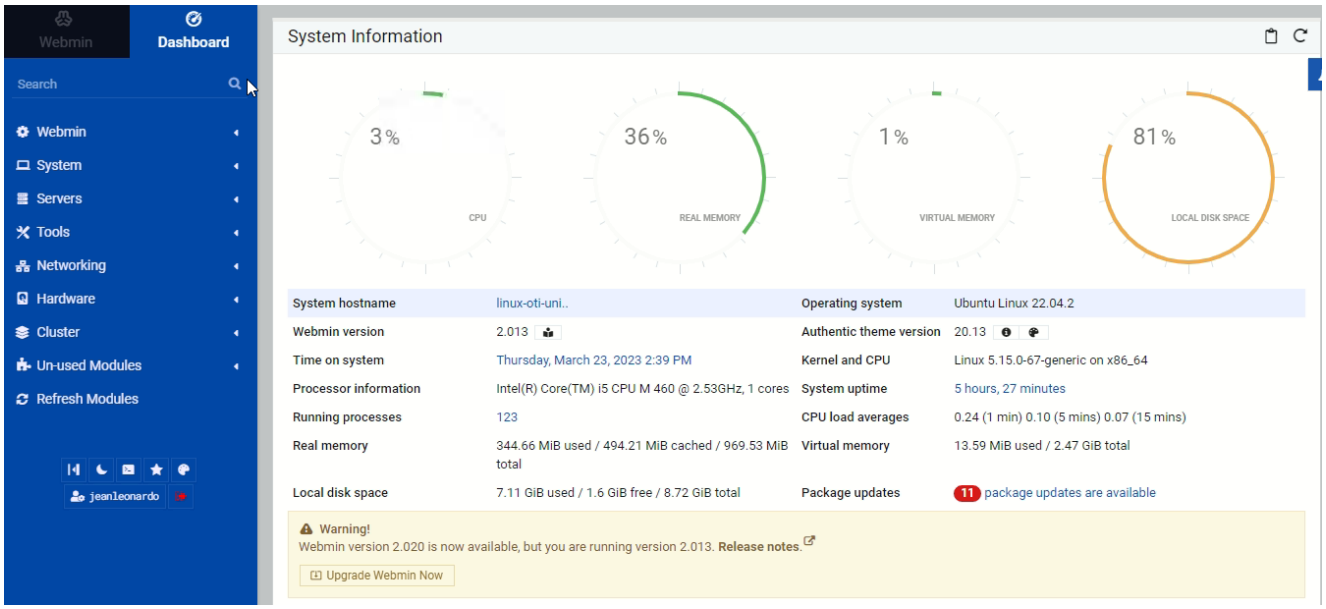
Le cargará la siguiente ventana:



Se loguea con sus credenciales:

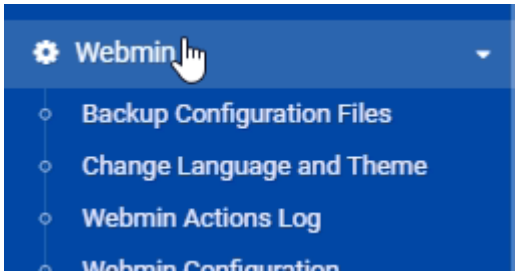


Le tendrá que aparecer el siguiente Dashboard:



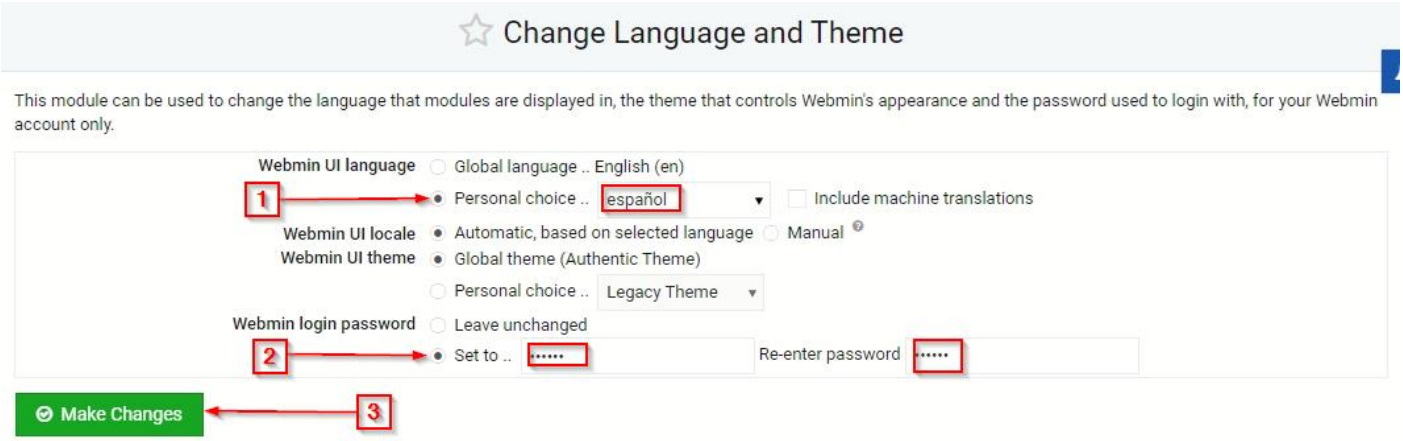
Por defecto le cargará la página en inglés, para cambiar el idioma, realicé lo siguiente:

Click en el apartado “Webmin”:

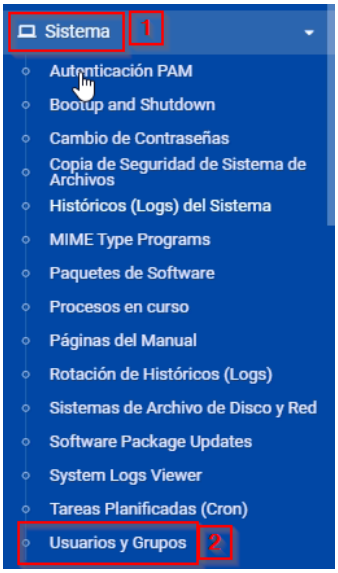


Click en “Change Language and Theme”:

Siga los pasos 1,2 y 3.

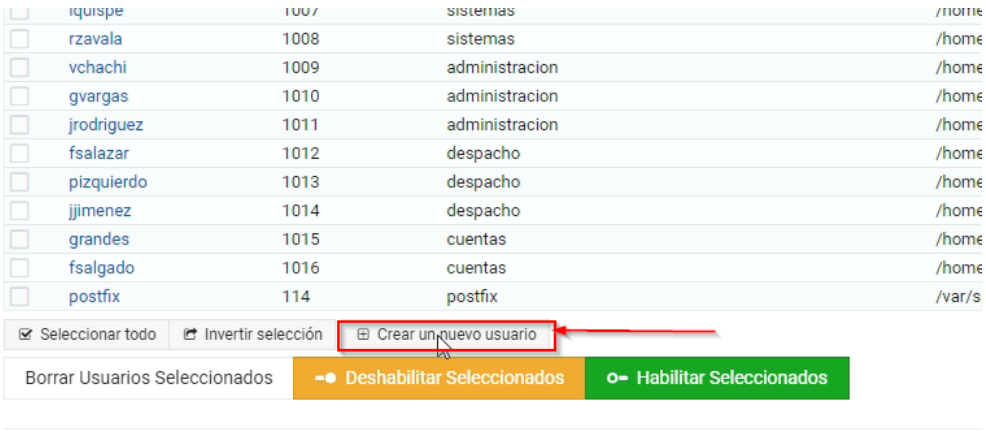


Ver los usuarios creados desde Webmin:

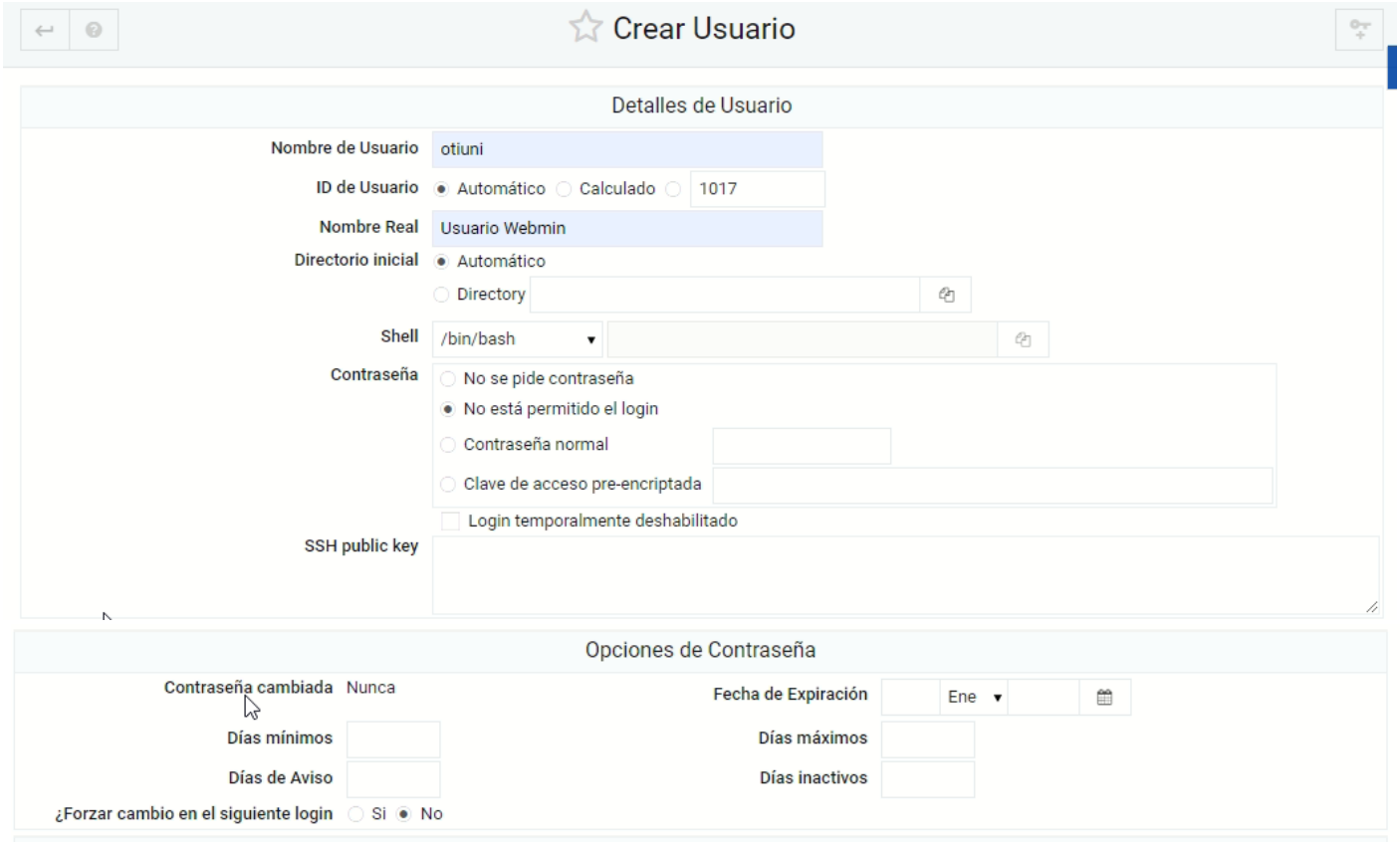


Crear un nuevo usuario con Webmin:

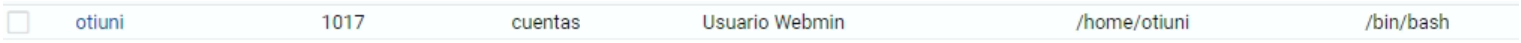
Dé click en “Crear un nuevo usuario”:



Luego realice las siguientes configuraciones, su ventana le debería quedar de la siguiente forma:

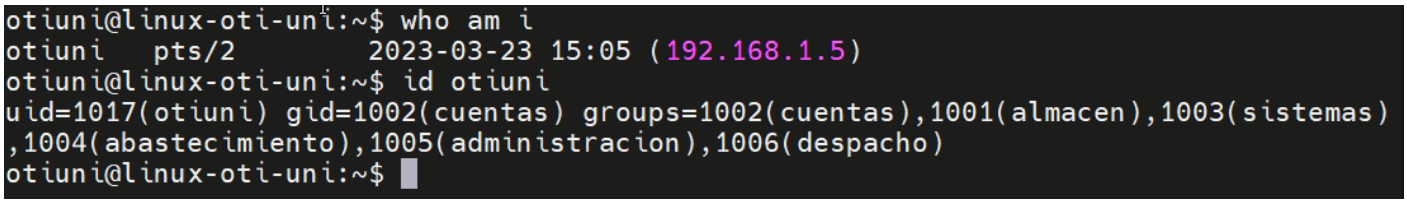


Verifique la creación de su usuario en Webmin:



A continuación, abra una terminal y logueese con este nuevo usuario:

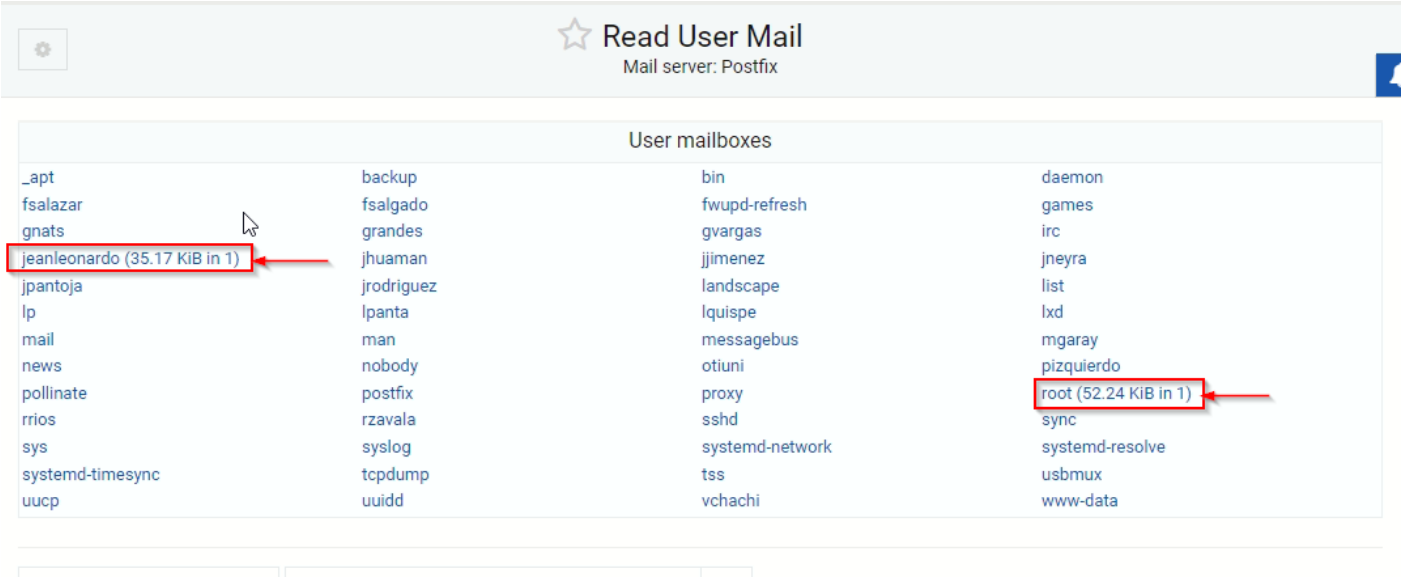
Verificamos:



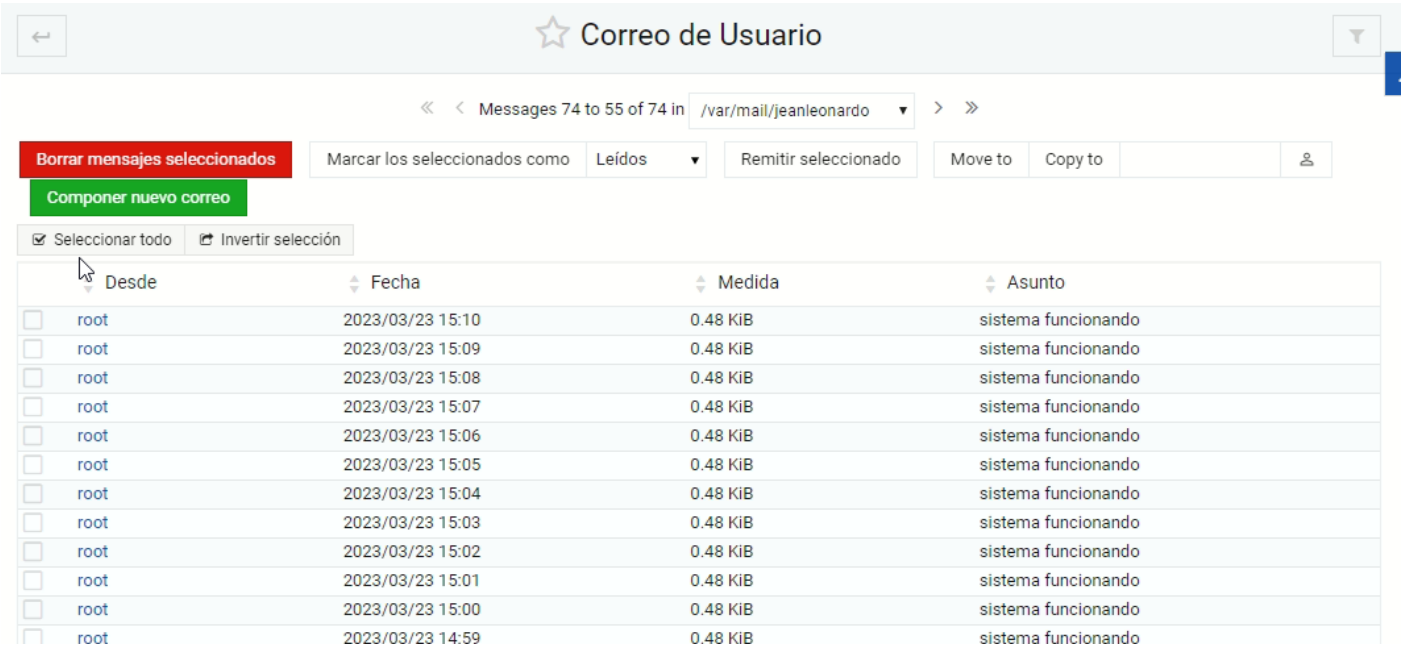
Pregunta: ¿Al crear un usuario, donde empleo más pasos?¿A través de la terminal o la interfaz gráfica de Webmin?

Desde Webmin también podemos ver los correos del programa de envío de correo que instalamos cuando hicimos la parte de cron, veamos:

Click en “Servidores” y luego en “Lectura de Correo de Usuarios”:

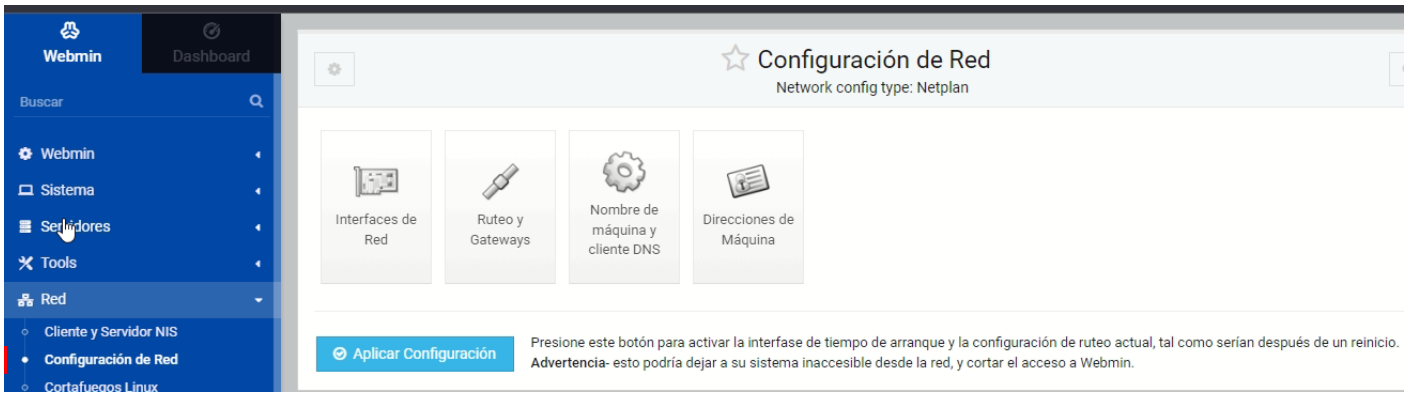


Vemos que hay espacio asignado a esos 2 usuarios, es decir hay correos que recibieron esos 2 usuarios, damos click por ejemplo en “jeanleonardo” y vemos los siguiente:



Podemos observar que se siguen enviando los correos con el mensaje “sistema funcionando” hasta la hora actual, esto debido a la sentencia en el archivo cron que configuramos.

En el apartado de “Red” observe que hay 4 campos:



Aquí puede configurar cada uno de los parámetros de red de su sistema. Abra todos los apartados y saque sus conclusiones.

Por ejemplo, dé click en el apartado “Direcciones de Maquinas” hace referencia al archivo “/etc /hosts”, puede ver el contenido del archivo con el comando cat, como se muestra a continuación

```
otiuni@linux-oti-uni:~$ cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 linux-oti-uni

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
otiuni@linux-oti-uni:~$
```

¿Qué hace el archivo /etc/hosts?

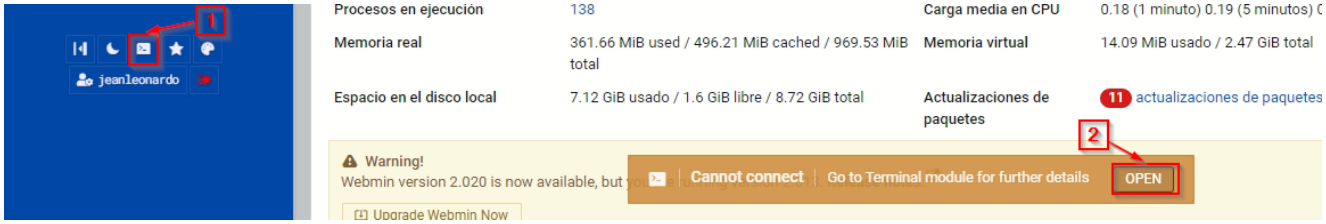
El archivo hosts de un ordenador se usa por el sistema operativo para guardar la correspondencia entre dominios de Internet y direcciones IP. Este es uno de los diferentes métodos que usa el sistema operativo para resolver nombres de dominio.

Abrir consola o terminal desde Webmin para administrar el servidor:

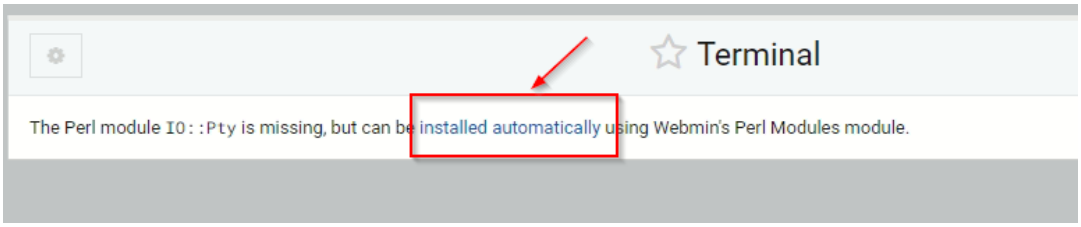
Siga los siguientes pasos:

Es posible que se le presente el siguiente inconveniente al comienzo, mostrado en la imagen. Lo solucionaremos a continuación

Dé click en 1, luego en 2:

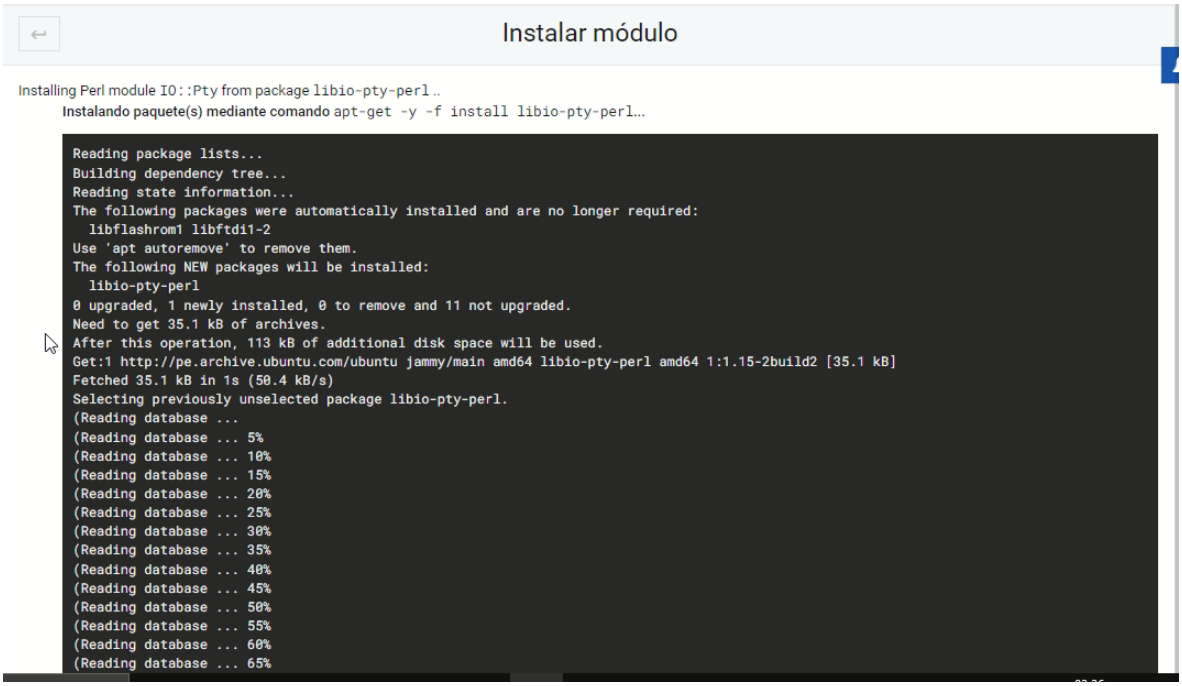


Ahora, click en:



Aquí tiene que esperar, ya que el proceso de instalar el módulo faltante suele demorar un poco.

Se recomienda que espere y no cierre la ventana, ya que podría originar problemas. Le saldrá la siguiente ventana:

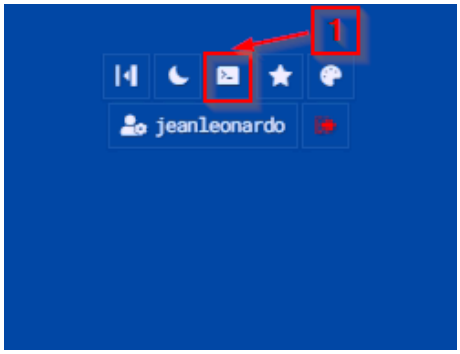


Al finalizar la instalación, en la parte inferior le saldrá el siguiente mensaje:

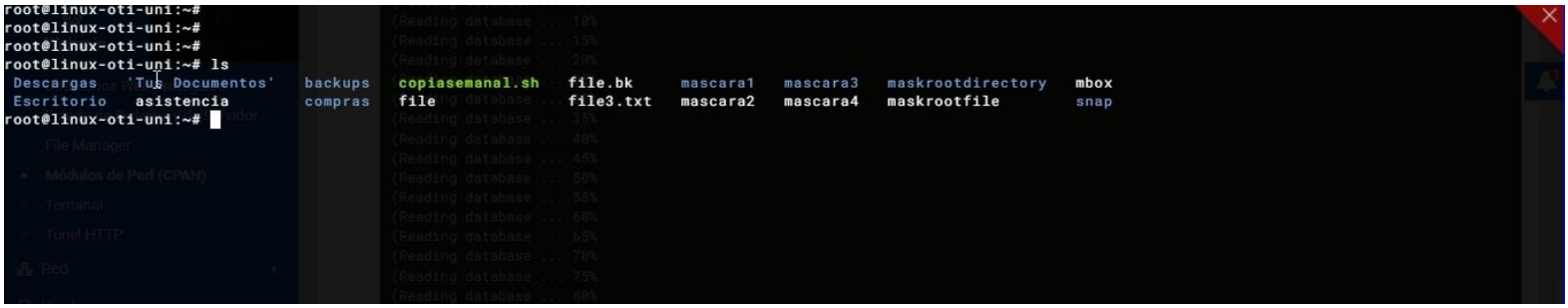
```
(Reading database ... 80%
(Reading database ... 85%
(Reading database ... 90%
(Reading database ... 95%
(Reading database ... 100%
(Reading database ... 90991 files and directories currently installed.)
Preparing to unpack .../libio-pty-perl_1%3a1.15-2build2_amd64.deb ...
Unpacking libio-pty-perl (1:1.15-2build2) ...
Setting up libio-pty-perl (1:1.15-2build2) ...
Processing triggers for man-db (2.10.2-1) ...
NEEDRESTART-VER: 3.5
NEEDRESTART-KCUR: 5.15.0-67-generic
NEEDRESTART-KEXP: 5.15.0-67-generic
NEEDRESTART-KSTA: 1
NEEDRESTART-SVC: webmin.service
```

... instalación completa.

Una vez realizado, todo lo anterior, realice lo siguiente:



Una vez que da click en el “Botón de Terminal”, le aparecerá lo siguiente:

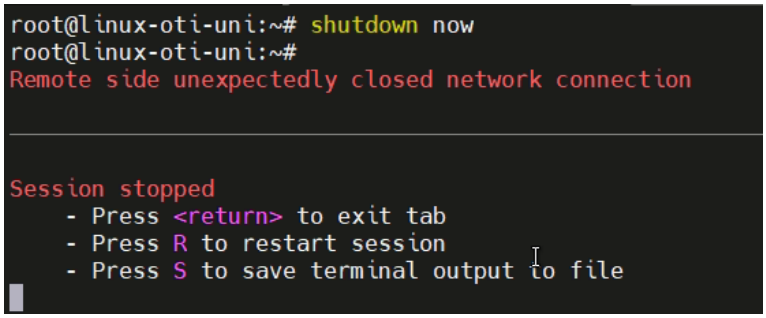


Su consola o terminal habrá aparecido desde Webmin, dándole como opción de que controle o maneje su sistema por medio de comandos si no lo quiere realizar con el entorno gráfico que le proporciona Webmin.

Montar un disco y crear partición en Linux:

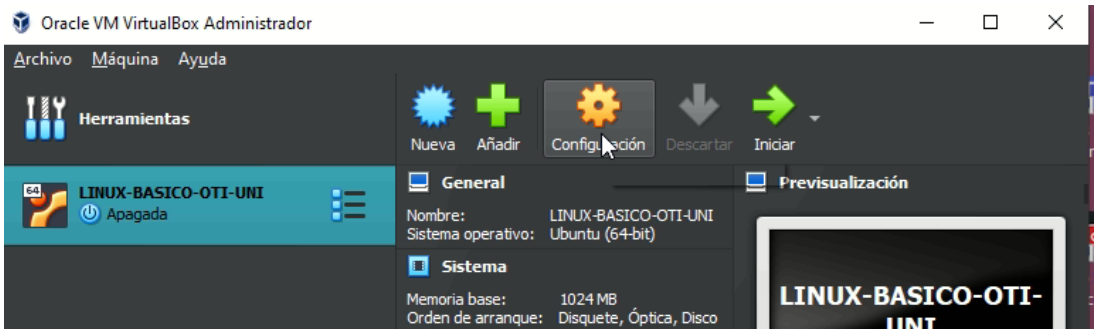
Supongamos que usted compra o en su organización le añaden un disco duro (HDD) o una unidad de estado sólido (SSD) a su servidor Linux, no solo basta con conectar el disco para poder usar el espacio disponible, tiene que realizar algunas configuraciones, en este apartado veremos cómo realizar dichas configuraciones.

Para nuestro entorno virtual utilizando VirtualBox tiene que primero apagar la máquina virtual, veamos:



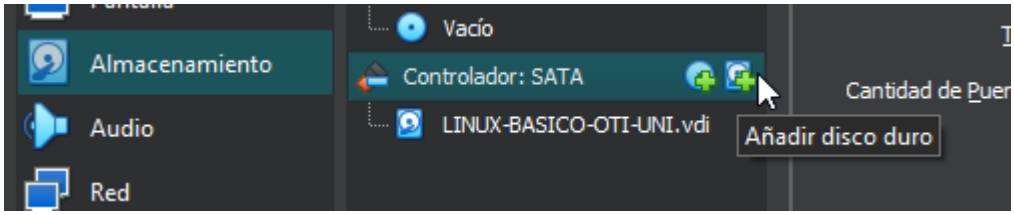
Luego de que termina de apagarse, realiza lo siguiente:

Click en configuración:

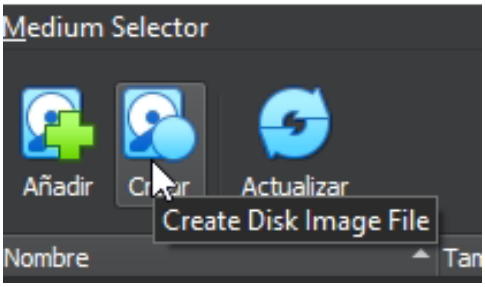


Click en almacenamiento:

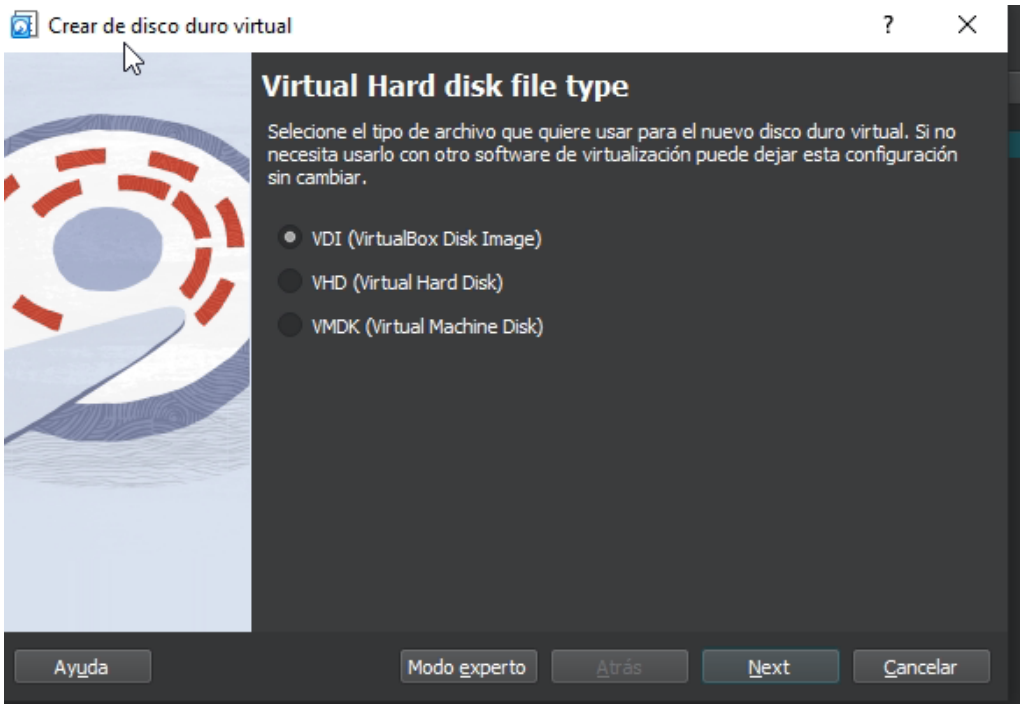
Click en “Añadir disco duro”



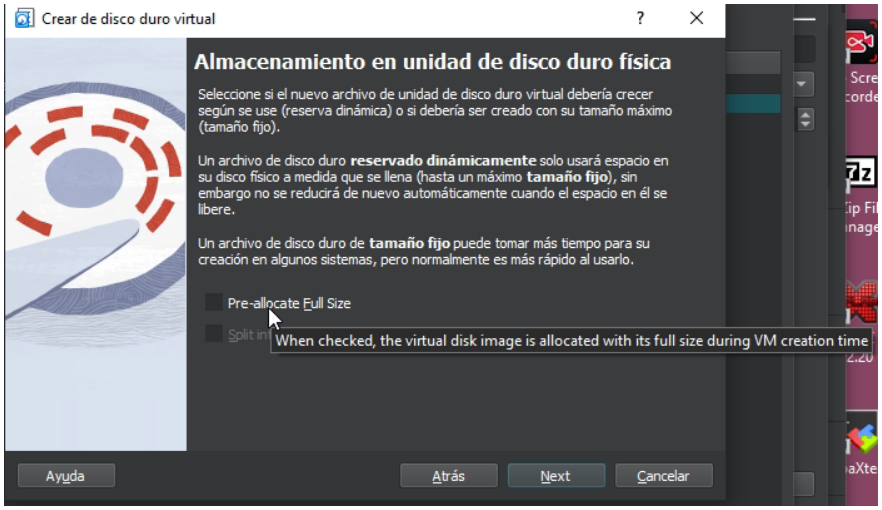
Click en “Crear”



Click en “Next”

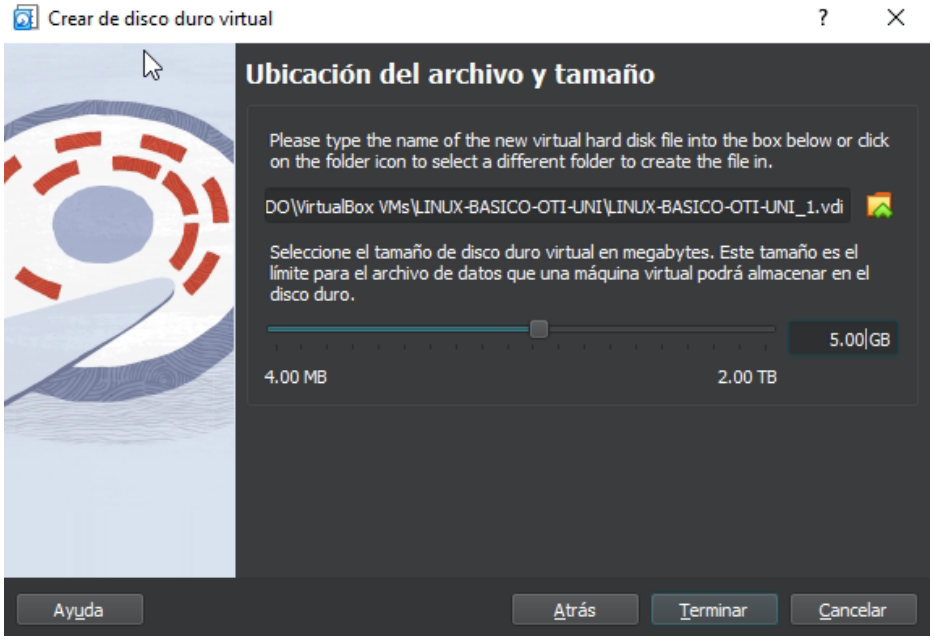


Le aparecerá la siguiente ventana:

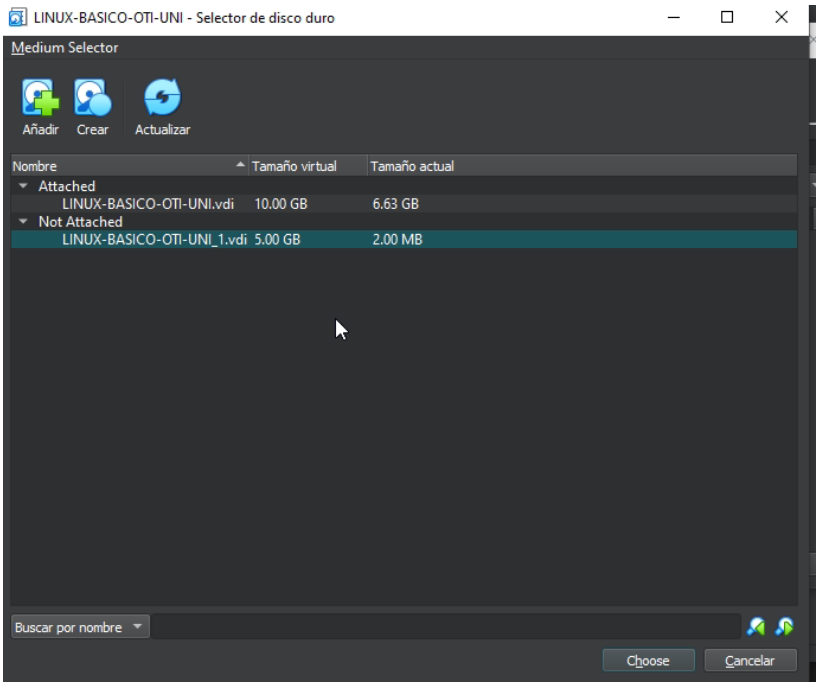


Un disco virtual preasignado asigna todo el almacenamiento necesario para una máquina virtual por adelantado. Por ejemplo, un volumen lógico preasignado de 20 GB creado para la partición de datos de una máquina virtual ocupará 20 GB de espacio de almacenamiento inmediatamente después de la creación. **En este caso "desmarque" esta opción, para que la asignación se haga dinámicamente a medida que vamos ocupando el espacio del disco y no ocupe todo al crearlo inicialmente.** Un disco virtual preasignado tiene el mismo tamaño que el tamaño asignado al disco duro de la máquina virtual. Por lo tanto, una máquina virtual asignada a un disco duro de 40 GB ocupa 40 GB de espacio en su Windows (más el espacio requerido para las instantáneas y otros archivos de la máquina virtual). Este espacio es el mismo que el "Tamaño total" del disco duro que muestra el sistema operativo invitado.

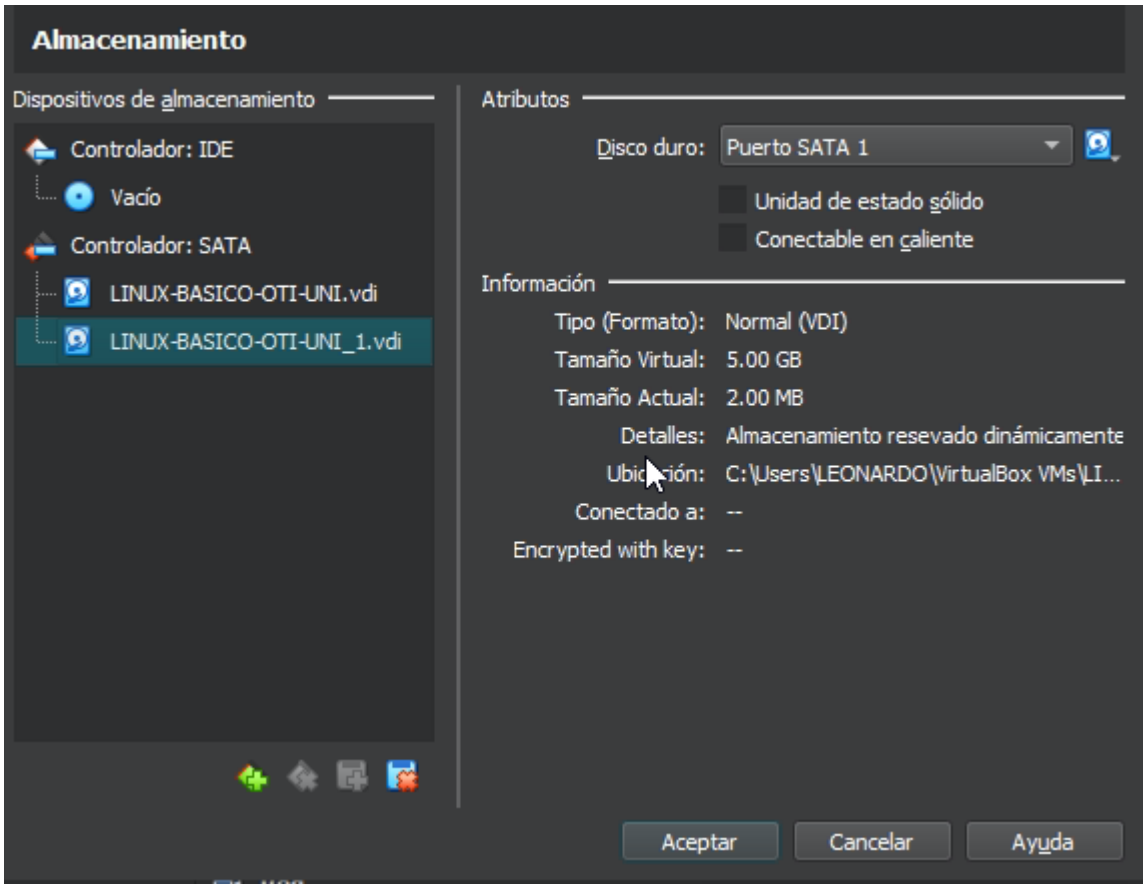
En este caso asignaremos 5GB de disco, solo para ejemplificar el proceso:



Luego, click en “Choose”:



Vea el resumen de su nuevo disco asignado:



Luego click en “Aceptar” e “Inicie” su máquina virtual

Logueese con el usuario “root” por SSH.

Montar un disco y crear partición en Linux:

Comando o Instrucción	Explicación o comentario
Entre como usuario root	
# lsblk sdb 8:16 0 5G 0 disk sr0 11:0 1 1024M 0 rom	Aquí vemos que nuestro disco fue reconocido con el nombre sdb
# ls -l /dev/sd* brw-rw---- 1 root disk 8, 16 mar 23 16:22 /dev/sdb	Vemos que nos sale nuestro disco con la ruta **/dev/sdb (esta ruta nos servirá para crear la partición más adelante)** Note que el tipo de archivo es tipo b es decir es un “dispositivo de bloques”
# mount	Nos aparecen las particiones montadas en un punto de montaje, como aun nuestro disco sdb no esta montado, no aparecerá aquí
# mount | grep sda	Vemos el punto de montaje de nuestra particion inicial
# mount | grep sdb	Vemos que aca no saldra nada, ya que es un disco nuevo
# df -h	Nos sale las particiones montadas, como aun nuestro disco "sdb" no está montado no aparecerá aquí

Creación de la partición

Comando o Instrucción	Explicación o comentario
# fdisk /dev/sdb p n p 1 Presione Enter Presione Enter p → **Aquí corroboramos que nuestra partición tomo el nombre de sdb1** w	El comando fdisk me permite inicializar un disco duro, la inicialización es el proceso mediante el cual yo puedo crear y asignar un partición dentro de este disco duro.

# ls -l /dev/sd*	Aquí corroboramos que nuestra partición tomo el nombre de **sdb1**, la ruta **/dev/sdb1** nos servirá para el darle el formato a continuación, tenga en cuenta este dato.

Formateo de la partición

Comando o Instrucción	Explicación o comentario
# mkfs.ext4 **/dev/sdb1**	En este caso estoy particionando con el formato ext4, porque con ese formato instalamos nuestro Ubuntu Server

Creación y asignación del punto de montaje:

Comando o Instrucción	Explicación o comentario
# cd /home/jeanleonardo	Crearemos el directorio que servirá como punto de montaje dentro del directorio “jeanleonardo”
# pwd	
# mkdir DiscoAdicional	La ruta seria **/home/jeanleonardo/DiscoAdicional**
# ls -l	
#mount /dev/sdb1 **/home/jeanleonardo/DiscoAdicional**	Asignamos el punto de montaje de la partición “sdb1” al directorio “DiscoAdicional”
# df -h	Verificamos. Observe que /dev/sdb1 esta montado en /home/jeanleonardo/DiscoAdicional Esto significa que si nosotros empezamos a llenar el directorio “DiscoAdicional” con archivos, directorios, etc; se llenará el disco de partición sdb1, que es el que agregamos y tiene 5GB.
# mount | grep sdb1	Verificamos

Después de realizar todo esto, recién podemos “utilizar” el espacio que tiene para ofrecer el “disco añadido”. El uso que le dé puede ser variado, por ejemplo lo puede utilizar como disco de backup.

Por ejemplo, podría utilizar este disco únicamente para copias de seguridad.

Veamos el consumo de espacio:

Comando o Instrucción	Explicación o comentario
Ingrese con el usuario root	
# cd /home/jeanleonardo/DiscoAdicional	
# df	

```
root@linux-oti-uni:/home/jeanleonardo# df
Filesystem     1K-blocks    Used Available Use% Mounted on
tmpfs              99284    1108     98176   2% /run
/dev/sda4        8881044 6864968   1543348  82% /
tmpfs             496404       0    496404   0% /dev/shm
tmpfs               5120       0      5120   0% /run/lock
/dev/sda2         271280  132764    117012  54% /boot
tmpfs              99280       4     99276   1% /run/user/0
/dev/sdb1        5073568      24   4795068   1% /home/jeanleonardo/DiscoAdicional
root@linux-oti-uni:/home/jeanleonardo#
```

ANTES

Particion Principal

Particion del Disco que añadimos

#wget https://www.webmin.com/download/webmin-current.tar.gz	Descarguemos este archivo que es un poco pesado.
# tar xvfz webmin-current.tar.gz	Descomprimimos y desempaquetamos el contenido del archivo **.tar.gz**
# ls -l	
# df	

```
root@linux-oti-uni:/home/jeanleonardo/DiscoAdicional# df
Filesystem     1K-blocks    Used Available Use% Mounted on
tmpfs              99284    1108     98176   2% /run
/dev/sda4        8881044 6864996   1543320  82% /
tmpfs             496404       0    496404   0% /dev/shm
tmpfs               5120       0      5120   0% /run/lock
/dev/sda2         271280  132764    117012  54% /boot
tmpfs              99280       4     99276   1% /run/user/0
/dev/sdb1        5073568  238704   4556388   5% /home/jeanleonardo/DiscoAdicional
root@linux-oti-uni:/home/jeanleonardo/DiscoAdicional#
```

DESPUES

Partición Principal

Partición del disco que añadimos

Observe que la partición principal sigue en 82% mientras que nueva partición paso de 1% a 5% en porcentaje de uso.