Comando date, sus usos y respaldo de archivos

Autor: Sergio González D. **(sergio.gonzalez.duran@gmail.com)**

Si ya has usado la línea de comandos o shell de Linux por un tiempo, seguramente entonces, el comando **`date`** ya te es familiar, lo ejecutas y te devuelve la fecha y hora, mmmm, nada especial ni muy útil. Peor atrás de esa aparente simplicidad se oculta un comando potente y sofisticado que bien utilizado te proporciona varias opciones muy útiles no solo de manejo de fechas sino también para una parte medular del control de respaldos automatizados, que es precisamente la del empaquetado de los archivos que se respaldan, bajo archivos que obtienen su nombre a partir de **`date`**.

`date` básico y cambiar el entorno local

En este manual, comenzaremos entonces desde lo básico, **`date`** sin argumentos:

```
#> date
Sun Jun 24 18:50:23 CDT 2007
```

Lo primero que notamos es la que la salida esta en inglés, ya que el primer campo representa el día de la semana "Sun" de "Sunday" domingo, y el siguiente campo no es tan evidente que esté en inglés ya que es Junio pero si fuera por ejemplo enero se mostraría "Jan" de "january" y como se trata de trabajar en español (claro, nada contra el inglés, solo es por si te gusta tener tu sistema operativo en español), será lo primero que cambiaremos entonces:

```
 #> echo $LC_TIME

(La variable de entorno LC_TIME no tiene ningún valor,
 por lo tanto toma el default que es el inglés.)

#>
#> export LC_TIME=es_MX
#> date
dom jun 24 18:55:18 CDT 2007
```

Ahora el resultado es en español al cambiar el "locale" sobre la manera de mostrar la fecha. Por cierto el código que utilizé corresponde a español México (es_MX) para establecerlo en tu país específico y no sabes cual es su código, puedes usar el comando **`locale -a`** para ver un listado completo de códigos de países. **`locale`** sin argumentos te devuelve el valor de las variables de entorno establecidas en tu sistema.

Esta variable "LC_TIME" se perderá en el siguiente inicio de sesión, si deseas conservarla edita **.bashrc** dentro de tu directorio de inicio y agrégalo al final, de este modo cada vez que entres a tu sesión de usario quedará establecida la variable de entorno.

Por cierto, después de la hora viene un campo "CDT" (Central Daylight Time) que representa la zona horaria o huso horario de referencia. En este caso en México es lo que conocemos como "Horario de Verano del Centro".

Estableciendo la fecha hora y del sistema

Para establecer la fecha y hora del sistema se usa **`date`** seguido del siguiente patrón de entrada de datos:

[MMDDhhmm[[CC]YY][.ss]]
MM = mes, DD = día, hh = hora, mm = minuto, CC = siglo (Century), YY = año, ss = segundos

Para establecer entonces la fecha al 20 de Julio del 2007 a las 8:05 de la noche:

```
#> date 0720200507
vie jul 20 20:05:00 CDT 2007
```

Obsérvese que siempre se usará el formato de 24 horas y dos dígitos en los campos. Si quisieramos fijar la fecha al 1 de enero de 1998 a las 12 del día con 45 segundos:

```
#> date 010112001998.45
jue ene  1 12:00:45 CST 1998
```

Como se puede observar en el patrón los campos obligatarios son el mes, el día, la hora y minutos, los demás son opcionales.

NOTA IMPORTANTE: **`date`**, como se ha mencionado, establece la fecha del sistema, que es diferente a la fecha de hardware o de bios. Esta fecha del reloj físico del sistema lo puedes consultar con el comando **`hwclock`**. Si deseas que la fecha del sistema sea igual a la de hardware, o la de hardware igual a la fecha del sistema, usa las siguientes opciones:

```
#> hwclock --hctosys reloj hardware a reloj sistema
#> hwclock --systohc reloj sistema a reloj hardware
```

Opción "-d" (trabajando con elementos relativos)

Pero regresemos a **`date`**, y ahora veamos como podemos obtener resultados de fechas en el pasado o presente, los más simples primero y todo con la opción "-d":

```
#> date -d "tomorrow"
lun jun 25 19:55:28 CDT 2007
#> date -d "yesterday"
sáb jun 23 19:55:36 CDT 2007
```

Pues interesante, dirás, pero sigue sin ser muy útil, pero espera, dentro de las comillas " ", es posible utilizar una combinación de las siguientes palabras: (todas en inglés)

- días de la semana, palabra completa o abreviación de tres letras: mon, tue, wen, etc
- meses, palabra completa o abreviación de tres letras: jan, feb, mar, apr, etc.
- next
- ago
- last
- year, month, week, day, hour, minute, second
- indicadores de zonas horarias, números
- el símbolo + y el símbolo -

Asi que si por ejemplo quisieras saber que día será dentro de 6 semanas:

```
#> date -d "6 weeks"
dom ago  5 20:25:08 CDT 2007
```

O lo contrario en el pasado con "ago", que día fue hace 6 semanas:

```
#> date -d "6 weeks ago"
dom may 13 20:26:46 CDT 2007
```

Con algunos ejemplos quedará más que claro como usar elementos relativos con la opción "-d" y asi conocer fechas pasadas o futuras, todas calculadas a partir de la hora y fecha en que se ejecute el comando: (en este caso el 24 jun 2007 20:40 aproximadamente:

```
#> date -d "next friday"
vie jun 29 00:00:00 CDT 2007
#> date -d "last friday"
vie jun 22 00:00:00 CDT 2007
#> date -d "2 months 1 week ago"
vie ago 17 20:33:19 CDT 2007
#> date -d "1 year 3 months 2 weeks 3 days 10 hours 25 minutes 5 seconds"
dom oct 12 06:59:09 CDT 2008
#> date -d "January 4"
jue ene  4 00:00:00 CST 2007
#> date -d "July 20 2020"
lun jul 20 00:00:00 CDT 2020
#> date -d "72 hours ago"
jue jun 21 20:47:01 CDT 2007
#> date -d "50 days ago"
sáb may  5 20:47:41 CDT 2007
#>date -d "+5 hours"
lun jun 25 01:58:07 CDT 2007
#> date -d "-5 days"
mar jun 19 20:58:56 CDT 2007
```

Los ejemplos son autoexplicativos, hay que tener en cuenta que si utilizas 'ago' o el símbolo "-" el resultado puede no ser el esperado, veamos:

```
#> date
dom jun 24 20:50:39 CDT 2007
#> date -d "3 years 1 month 10 days ago"
mié jul 14 20:51:06 CDT 2010
```

Como se puede observar, el resultado es una fecha en el 2010, y no 3 años atrás a partir del 2007, cuando se usa 'ago' es necesario indicarlo en cada elemento relativo de la instrucción:

```
#> date -d "3 years ago 1 month ago 10 days ago"
vie may 14 20:53:50 CDT 2004
```

Ahora si el resultado es el correcto.

Formateo de la salida

Una de las opciones más interesante y útiles de **`date`** es la opción de formato de la salida, que se indica con el signo "+" seguido de una o màs literales de formato, que se indican con "%". Por ejemplo: %a indica el día de la semana abreviado, %B el nombre completo del mes, etc. Con unos cuantos ejemplos se entenderá la idea:

```
#> date
dom jun 24 22:01:28 CDT 2007
#> date +%a
dom
#> date +%A
domingo
#> date +%B%A
juniodomingo
#> date +%B-%A
junio-domingo
#> date +%B-%A-%d
junio-domingo-24
#> date +%B-%A-%d-%Y
junio-domingo-24-2007
#> date +%B/%A/%d/%y
junio/domingo/24/07
#> date +%d%m%y
240607
#> date +%d %m %y
date: extra operand `%m'
Try `date --help' for more information.
#> date +%d\ %m\ %y
24 06 0
```

Cada literal, como se aprecia en los ejemplos, representa una parte de la fecha original y la salida es el formato que se indique con la combinación de literales que se haya usado.

Entre cada literal puede haber o no, uno de tres símbolos de separaciòn que son: "-","_","/" solo esos, y si se observa el penúltimo ejemplo, si queremos separar las literales con un espacio, el resultado será un error, entonces es posible usar el caracter de escape "\" que nos permite indicar cualquier caracter de separación entre literales.

Si se desea una salida mas elaborada como: "La fecha es: domingo 24 de junio del 2007" se indica entonces todo el formato entre comillas simples ':

```
#> date +'La fecha es: %A %d de %B del %Y'
La fecha es: domingo 24 de junio del 2007
```

De hecho la salida por defecto del comando es un formato, veamos:

```
#> date
dom jun 24 22:33:49 CDT 2007
#> date +'%a %b %e %H:%M:%S %Z %Y'
dom jun 24 22:33:46 CDT 2007
```

La siguiente tabla detalla los principales literales que se pueden usar para formatos de salida con **date**:

Literal	Descripción
%a	Nombre abreviado del día de la semana
%A	Nombre completo del día de la semana
%b	Nombre abreviado del mes
%E	Nombre compl- o del mes

%c	fecha y hora
%C	Dos primeros dígitos del año, ejemplo 20 de 2007
%d	Día del mes con dos dígitos, ejemplo 01
%D	Igual que indicar %m/%d/%y
%e	Día del mes con uno o dos dígitos, ejemplo 1, 10
%F	Fecha completa, igual que %Y-%m-%d
%h	igual que %b
%H	Hora en formato 24 horas con dos dígitos (00..23)
%I	Hora en formato 12 horas con dos dígitos(01..12)
%j	El día del año (001..366)
%k	Hora en formato 24 horas con uno o dos dígitos(0..23)
%l	Hora en formato 12 horas con uno o dos dígitos(1..12)
%m	Mes con dos dígitos(01..12)
%M	Minutos con dos dígitos (00..59)
%r	Hora completa en formato de 12 horas (ejemplo 01:23:45)
%R	Horas y minutos en formato de 24 horas, igual que %H:%M
%s	Segundos transcurridos desde 01/Ene/1970 00:00:00 (fecha epoch)
%S	Segundos con dos dígitos, (00..60)
%T	Hora completa en formato de 24 horas (ejemplo 13:23:45)
%u	Dia de la semana en número (1..7, 1 es lunes)
%U	Número de la semana en el año, domingo primer día de la semana (00..53)
%V	Número de la semana en el año, lunes primer día de la semana (01..53) Formato ISO
%w	Día de la semana en número (0..6, 0 es domingo)
%W	Número de la semana en el año, lunes primer día de la semana (00..53)
%y	Últimos dos dígitos del año
%Y	Año con cutro dígitos
%z	Huso o zona horaria numérica
%Z	Huso o zona horaria abreviación alfabética

`date` y respaldo de archivos

Respaldar el sistema, archivos de trabajo, configuración, etc. Es parte esencial de cualquier labor de administración de sistemas informáticos. **`date`** resulta muy conveniente para llevar un control automatizado de respaldos.

Supongamos que dos veces por día (a las 14:00 y 21:00), todos los días, se debe de respaldar el directorio /usr/archivos, y por simplicidad se respalda a un directorio montado en el mismo equipo por NFS que realmente esta ubicado en otro equipo, dicho directorio es /respaldos. Manualmente podría ser de la siguiente manera:

```
#> tar xvzf /usr/archivos/* /respaldos/respaldo20070624_1400.tar.gz
```

El administrador directamente está escribiendo el nombre del archivo donde se empaquetan y comprimen los archivos respaldados que es "respaldo20070624_1400", y su sintaxis para distinguir un respaldo del otro es la

fecha y la hora AÑOMESDIA_HHMM. Pero ¿que pasa si un día no puede estar presente para realizar el respaldo correpsondiente?. Ciertamente con cron (ver Manual básico de Cron (index.php?cont=info_admon_006)) es posible programar la tarea, pero el problema no es ese, sino como indicar autmáticamente otra fecha para el nombre del archivo, el siguiente script de shell resuelve el problema:

```
# respaldo de /usr/archivos
# se forma el nombre del archivo
DIA=`date +%d`
MES=`date +%m`
AÑO=`date +%Y`
HORA=`date +%H`
ARCHIVO=respaldo$AÑO$MES$DIA_$HORA00.tar.gz
# copia del archivo
tar xvzf /usr/archivos/* /respaldos/$ARCHIVO
```

Si este script lo nombramos como "respaldo.sh" ubicado dentro de /root con sus permisos adecuados de ejecucción, entonces la línea **cron** correspondiente en /etc/crontab sería la siguiente:

```
* 14,21 * * * root /root/respaldo.sh
```

Con lo anterior deberá bastar para que el respaldo se ejecute a las horas indicadas y automáticamente se forme el nombre del archivo con la ayuda de **`date`** y las literales correspondientes.

Esto por supuesto es un ejemplo muy sencillo de lo que puede lograrse, espero te sea de utilidad para tus proyectos, y **`date`** ofrece más de lo que se expuso en esta guía, para más información puedes usar cualquiera de las ayudas que viene por defecto en cualquier distribucción Linux:

```
$> date --help
$> man date
$> info date
```
