Computer programmgesteuert aufwecken

Weckdienst

Muss ein Computer nicht 24 Stunden am Tag laufen, lässt sich durch bedarfsgerechtes Ein- und Ausschalten eine Menge Strom sparen. Drei verschiedene Methoden programmieren den Aufwachzeitpunkt direkt unter Linux, ohne ihn von Hand im Bios einzustellen. Mirko Dölle

Server laufen 24 Stunden am Tag, die Raids natürlich auch und die Arbeits-PCs schaltet am Abend kaum jemand aus. Dabei sind die Rechner tatsächlich selten länger als zehn Stunden in Gebrauch. Für Spezialanwendungen, etwa einen Faxserver, der nachts um zwei Uhr alle Telefaxe versendet, reicht es aus, wenn das Gerät um zwei Uhr startet und sich nach getaner Arbeit bis zur nächsten Nacht wieder abschaltet.

Ganz ohne zusätzliche Hardware wie Zeitschaltuhren oder Remote Power Switches funktioniert das Aufwachen über das Bios des Mainboards (Abbildung 1). Praktisch alle aktuellen Geräte haben eine Funktion, den Rechner zeitgesteuert einzuschalten. Zusätzlich gibt es „Wake on LAN“, falls zwischendurch jemand den Fileserver braucht.

Das Problem ist, die Uhrzeiten für zeitgesteuertes Einschalten unter Linux zu verändern – je nach Rechner müssen unterschiedliche Methoden angewandt werden. So funktioniert ACPI-Wakeup nicht bei allen Mainboards, was an den unterschiedlichen Implementationen der Standards liegt. NVRAM-Wakeup hingegen verändert direkt die Zeiteinstellung im nicht-flüchtigen Speicher, in dem das Bios seine Werte ablegt. Die dritte Methode, »settime« genannt, arbeitet mit einem einfachen Trick: Das Bios wacht immer zum selben Zeitpunkt auf, beim Herunterfahren stellt aber ein Skript die Rechneruhr entsprechend zurück.

Wunderwaffe ACPI?

Mit ACPI lässt sich der Computer wohl am einfachsten zu einer bestimmten Uhrzeit wieder einschalten – vorausgesetzt Kernel und Mainboard arbeiten ausreichend gut zusammen. Die Aufwachzeit wird dazu einfach nach »/proc/acpi/alarm« geschrieben:

```
echo 2004-08-02 20:15:00 >/proc/acpi/alarm
```

Der Kernel überträgt die Uhrzeit direkt in die RTC (Real Time Clock) des Rechners, das Datum jedoch nicht. Das führt dazu, dass der Rechner jeden Tag zur eingestellten Uhrzeit aufwacht und nicht nur am gewünschten Datum.

Das Problem ist, dass zwar die Übergabe der Uhrzeit an die RTC überwiegend einheitlich ist, ein solcher Standard aber für das Aufwachdatum fehlt. Entsprechend berücksichtigt die Funktion »acpi_system_write_alarm« aus »drivers/acpi/system.c« von Kernel 2.4 respektive »drivers/acpi/sleep/proc.c« von Kernel 2.6 auch nur die Alarm-Uhrzeitfelder.

Kein Datum in Sicht

Die Kernelentwickler Andy Grover und Paul Diefenbaugh haben bereits den kompletten Funktionsumfang vorgesehen, benötigen dafür jedoch eine brauchbare Fixed-ACPI-Tabelle (FACP). Nach den Kommentaren im Kernel taugen die aktuellen FACP-Tabellen der Mainboards jedoch nichts, weshalb der entsprechende Codeblock permanent abgeschaltet ist.

Bei einigen Mainboards funktioniert das Übertragen in die Echtzeituhr des Rechners nicht, sodass statt des hineingeschriebenen Aufwachzeitpunkts die Zeit des letzten Rechnerstarts hinterlegt ist. Bei anderen Rechnern führt jeder Schreibzugriff auf »/proc/acpi/alarm« zu einem sofortigen Komplettabsturz. Inzwischen funktioniert das ACPI-Wakeup bei vielen aktuellen Mainboards, eine bei weitem nicht vollständige Kompatibilitätsliste gibt es auf der Homepage der Mini-Videorekorder-Distribution LinVDR im Bereich Dokumentation [1].

Verkehrte Welt

Damit das Aufwecken per ACPI funktionieren kann, müssen die Bios-Einstellungen stimmen. Seit Jahren hat praktisch jeder Rechner im Bios eine Funk-

tion wie »Wake on Timer«, »Resume on Alarm«, »RTC Alarm Resume« oder eine Variation des gleichen Themas – doch musste der Benutzer die entsprechende Zeit stets im Bios einstellen. Zwar nutzt das ACPI-Wakeup eine ähnliche Technik, bedient sich jedoch nicht der Bios-Funktionen. Deshalb muss die entsprechende Wakeup-Funktion im Bios, auch wenn es widersinnig klingt, bei praktisch allen Mainboards unbedingt ausgeschaltet (disabled) sein. Auch wird die Uhrzeit des nächsten Rechnerstarts nicht im Bios angezeigt.

Eine weitere Klippe: Einige wenige Mainboards wie das Asus A7V133 verkraften es nicht, wenn nach dem Eintragen der Aufwachzeit sich die Echtzeituhr per »hwclock -w« mit der Systemzeit synchronisiert – der Rechner wacht nicht wieder auf. Hier muss »hwclock« mit dem zusätzlichen Parameter »--directisa« aufgerufen werden oder man setzt den Aufwachzeitpunkt erst nach dem »hwclock«-Aufruf.

Und täglich grüßt ...

Wie eingangs schon beschrieben wachen Rechner bei ACPI-Wakeup jeden Tag zur eingestellten Uhrzeit auf, was den Einsatzbereich etwas einschränkt. So reicht ACPI-Wakeup für den Faxserver, der nachts die anstehenden Mailings automatisch verarbeitet und sich dann wieder abschaltet. Auch für PC-Videorekorder (PVR), zum Beispiel LinVDR, eignet sich ACPI-Wakeup: Zwischen Mitternacht und zwei Uhr aktualisieren die meisten Fernsehsender ihren elektronischen Programmführer (EPG), ein Start um zwei Uhr eignet sich also gut für den EPG-Scan und die automatische Timer-Programmierung.

Ein Printserver hingegen, der täglich um 9 Uhr startet und nach 19 Uhr abschaltet, würde auch am Wochenende stundenlang sinnlos laufen. Eine zusätzliche Routine in den Startskripten, die Wochenenden und Feiertage erkennt und den Rechner an solchen Tagen automatisch herunterfährt, löst das Problem nur auf den ersten Blick – soll der Printserver doch einmal am Samstag laufen, würde er auch herunterfahren, wenn man ihn manuell einschaltet.

Die bessere Methode ist ein Skript, das beim Herunterfahren Wochenenden und Feiertage berücksichtigt und den Startzeitpunkt samt Datum programmiert. Dies beherrschen NVRAM-Wakeup und die »settime«-Methode.

NVRAM-Wakeup

NVRAM-Wakeup arbeitet mit Einstellungen des Bios, die im Non-Volatile RAM (NVRAM) gespeichert sind. Für den Betrieb müssen das Kernelmodul »nvram« übersetzt und zudem die Character-Devices »/dev/nvram« mit Major 10 und Minor 144, »/dev/rtc« mit Major 10 und Minor 135 sowie »/dev/mem« mit Major 1 und Minor 1 angelegt sein. Zudem ist die Modulkonfiguration »/etc/modules.conf« mit einem Alias so anzupassen, dass der Rechner das NVRAM-Modul bei jedem Zugriff auf »/dev/nvram« oder direkt beim Systemstart lädt.

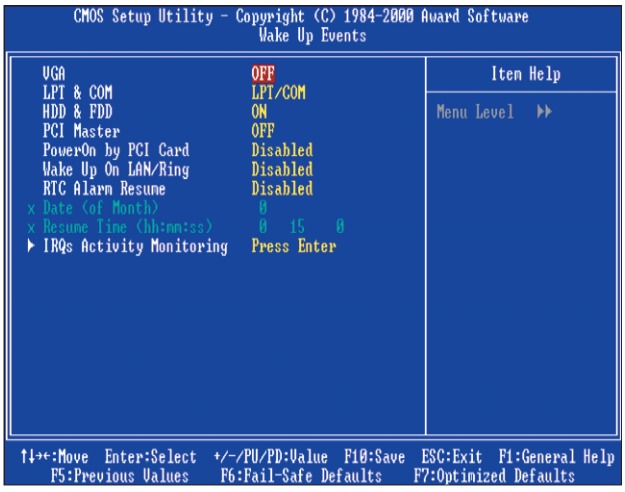

Abbildung 1: Das Bios fast aller Rechner enthält eine Aufwachfunktion, die »RTC Alarm Resume« oder ähnlich heißt.

Beim Übersetzen der Quellen von [2] gibt es keine Besonderheiten zu beachten, wie üblich landen die Binärdateien in »/usr/local«. Existieren die Devices »/dev/nvram«, »/dev/rtc« und »/dev/mem« noch nicht, legt »make devices« sie automatisch an.

In der Version 0.96 funktionieren die »tr«-Aufrufe in den Zeile 62 und 75 des Hilfsskripts »guess-helper.sh« nicht:

```
answers=`$echo $answers | ➥
tr [:lower:] [:upper:]`
```

Nach Einfügen von vier Hochkommas lief das Skript dann einwandfrei:

```
answers=`$echo $answers | ➥
tr '[:lower:]' '[:upper:]'`
```

Die meisten Hersteller betrachten die Aufteilung dieses Speicherbereichs als Betriebsgeheimnis – an welcher Stelle Datum und Uhrzeit wie gespeichert sind, ist nirgends beschrieben. Auch ändern sich die Positionen, etwa bei Bios-Updates, durchaus mal.

In den Quellen von NVRAM-Wakeup sind bereits die Daten einiger Mainboards enthalten, die Anwender mit Hilfe des Skripts »guess-helper.sh« ermittelt und an die Entwickler zurückgemeldet haben. Beim Aufruf von »nvram-wakeup« mit dem Parameter »-D« startet das Programm im Debug-Modus und zeigt an, ob das eingebaute Mainboard bereits bekannt ist. Bei der folgenden Meldung von »nvram-wakeup« kommt »guess-helper.sh« zum Zug:

```
nvram-wakeup: Your mainboard ➥
is currently not supported.
```

Das Skript »guess-helper.sh« versucht, die Positionen der einzelnen Felder im NVRAM herauszufinden. Dazu sind mindestens vier Neustarts des Rechners erforderlich: Das erste Mal muss der Benutzer von Hand im Bios die Aufwachfunktion aktivieren und auf den 31. des Monats um eine Sekunde vor Mitternacht stellen und dann »guess-helper.sh« aufrufen. Danach wird der Aufwachzeitpunkt auf den 11. des Monats um 12:13:14 Uhr gestellt, anschließend auf den 1. des Monats um Mitternacht und schließlich die Aufwachfunktion wieder deaktiviert.

Bei jedem Aufruf mit unterschiedlichem Aufwachzeitpunkt vergleicht »guess-helper.sh« den Inhalt des NVRAM mit den anderen Aufrufen, um die Speicherstellen für den Aufwachzeitpunkt zu bestimmen. Dabei verwendet das Skript zwei Methoden, und zwar den Zugriff über »/dev/nvram« und die direkte Ansteuerung über I/O-Adressen. Das Ergebnis sind zwei Konfigurationsdateien

»nvram-wakeup.conf« im Homeverzeichnis von Root in den Verzeichnissen »guess-nvram-module« beziehungsweise »guess-directisa«.

Ein Blick in die beiden Konfigurationsdateien zeigt, in welcher lediglich Kommentare zum Mainboard stehen und in welcher Adressinformationen gespeichert sind. Beim Mainboard Elito-Epox 8K5A2+ war die Datei »/root/guess-nvram-module« bis auf die Kommentare leer, während »/root/guess-directisa« die Adresseinträge aus Listing 1 enthielt. Sind beide Konfigurationsdateien leer, hat »guess-helper.sh« die Positionen nicht bestimmen können.

Der Pfad der Konfiguration muss, da es sich um kein bereits bekanntes Mainboard handelt, mit dem Parameter »-C« bei jedem Aufruf von »nvram-wakeup« angegeben sein. Zusätzlich ist in diesem Fall der Parameter »-A« erforderlich, weil der Zugriff nicht über das Device »/dev/nvram«, sondern direkt über die I/O-Adressen erfolgt. Bei einem »nvram-wakeup« bekannten Board würden beide Parameter entfallen.

Funktionsprobe

Für den ersten Test von »nvram-wakeup« ist diesmal ein beliebiger Aufwachzeitpunkt im Bios erforderlich, der dann mit »nvram-wakeup« verifiziert wird. Im folgenden Beispiel ist das Bios so eingestellt, dass der Rechner am 15. des Monats um 20:15 Uhr aufwacht. Beim Datum für den Aufruf von »nvram-wakeup« sind zusätzlich noch die Zeitdifferenz von zwei Stunden gegenüber UTC und der Vorlauf von fünf Minuten für den Start des Rechners zu berücksichtigen, also 2:05 Stunden zu addieren.

Abbildung 2 zeigt die Ausgabe von »nvram-wakeup«.

Listing 1: »nvram-wakeup.conf«

```
01 ################################################
02 ##  Mainboard autodetection information:
03 ##
04 ##    - Mainboard vendor:    ""
05 ##    - Mainboard type:      "VT8367-8235"
06 ##    - Mainboard revision: ""
07 ##    - BIOS vendor:         "Award Software
   International, Inc."
08 ##    - BIOS version:        "6.00 PG"
09 ##    - BIOS release:        "04/07/2003"
10
11 addr_stat         = 0xD2
12 shift_stat        = 5
13 addr_day          = 0xD8
14 addr_hour         = 0xD9
15 addr_min          = 0xDA
16 addr_sec          = 0xDB
17
18 upper_method = VT8235_37
```

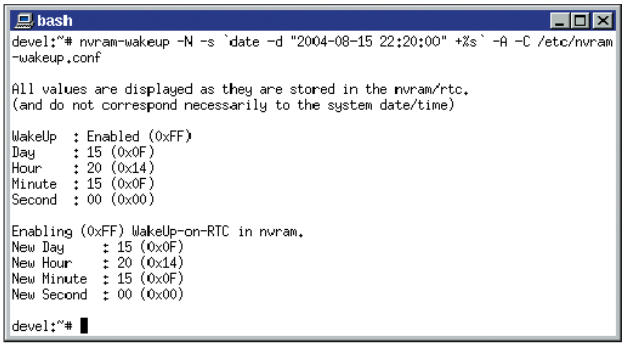

Abbildung 2: Beim Probeaufruf müssen die von »nvram-wakeup« angezeigten Zeiten übereinstimmen: Oben die gerade im Bios eingestellte Einschaltzeit, unten die vom Programm ermittelte.

Im oberen Abschnitt ist zu sehen, welche Aufwachdaten »nvram-wakeup« im NVRAM vorgefunden hat, unten dagegen, welche Daten ohne den Schreibschutz-Parameter »-N« eingetragen würden. Die Werte stimmen überein, somit ist davon auszugehen, dass »guess-helper.sh« die korrekten Speicherstellen ermittelt hat.

Im nächsten Schritt schreibt »nvram-wakeup« ein Datum etwa 15 Minuten in der Zukunft:

```
nvram-wakeup -s `date -d "+15 Minutes" ➥
+%s` -A -C /etc/nvram-wakeup.conf
```

Wacht der Rechner nach Aufruf von »poweroff« nicht wie vorgesehen auf, kann entweder ein grundsätzliches Problem mit APM und ACPI vorliegen oder der Rechner muss die Bios-Routine noch einmal abarbeiten – also neu starten –, um zur gewünschten Zeit aufzuwachen. Welche Änderungen dazu am NVRAM-Wakeup erforderlich sind, beschreibt die Anleitung von Hubertus Sandmann auf [3] sehr detailliert und in Deutsch.

Speicherbereich erweitern

Bei manchen Mainboards genügt die im »nvram«-Modul definierte Größe des über »/dev/nvram« auslesbaren Bereichs nicht, um den gesamten NVRAM zugänglich zu machen. Ein Zeichen dafür ist, dass »guess-helper.sh« die Positionen für Tag, Stunde, Minute und Sekunde gar nicht oder nur wenige Felder davon findet. In den Kernelquellen wird der Bereich in der Datei »drivers/char/nvram.c« definiert:

```
#define NVRAM_BYTES (128-NVRAM_FIRST_BYTE)
```

Hier muss der Bereich auf volle 128 Bytes erweitert werden:

```
#define NVRAM_BYTES  128
```

Möglicherweise findet »guess-helper.sh« nun die richtigen Felder.

Zurück in die Vergangenheit

Eine einfache Methode, die mit jedem Mainboard funktioniert, ist »settime«: Der Anwender programmiert einmalig im Bios einen fester Aufwach-Zeitpunkt, zum Beispiel den 31. des Monats um 23:59:59. Beim Herunterfahren berechnet ein Skript die Zeitdifferenz zwischen Herunterfahren und Aufwachen, zieht diesen Zeitraum vom 31. Mai 2004, 23:59:59 Uhr, ab und setzt System- und RTC-Zeit entsprechend auf einen Tag im Mai 2004. Beim nächsten Start muss ein Skript, hier »correcttime«, nur die Systemzeit wieder korrigieren.

In der Praxis ergibt diese Vorgehensweise einige Verwicklungen. So muss die Normalzeit bei jedem Systemstart unbedingt wieder hergestellt sein, noch bevor »fsck« läuft – andernfalls würden unter Umständen keine Plattenchecks mehr durchgeführt. Wird die Zeit jedoch vor dem »fsck« korrigiert, kann das nirgendwo vermerkt werden, weil ja alle Partitionen Read-only gemountet sind. Auch ein Dual-Bootsystem mit Windows oder einem anderen Betriebssystem ist problematisch, jene würden mit einer völlig falschen Systemzeit starten. Zudem eignet sich diese Methode nur, wenn der Rechner mindestens ein Mal in 60 Tagen starten soll.

Die Lösung sind zwei getrennte Skripte: »settime« berechnet die Zeitdifferenz

zwischen Normalzeit und Aufwachzeit, schreibt die Differenz in die Datei »/etc/timediff«, setzt die Systemzeit entsprechend in den Mai 2004 zurück und synchronisiert die RTC [4]. Beim Systemstart ruft das erste Init-Skript gleich am Anfang »correcttime« auf, das die Zeitdifferenz ausliest, die Systemzeit korrigiert und auch die RTC synchronisiert. So bekommen andere Startskripte nichts von der Zeitverschiebung von »settime« mit.

Delta-T

Die »settime«-Methode nutzt aus, dass die Uhrzeit in der RTC des Rechners normal weiterläuft. Somit ist es nicht nötig, nach jedem Start die Systemzeit mit einem anderen Rechner oder einer Uhr abzugleichen – es reicht zu wissen, wie viele Sekunden die Uhr gegenüber der Normalzeit nachgeht.

Auch komplizierte Datumsberechnungen mit Berücksichtigung von Schaltjahren ist nicht erforderlich, »date« stellt alle nötigen Funktionen zur Verfügung. So stellt der folgende Befehl die Systemzeit um eine Stunde vor:

```
date -s "+3600 seconds"
```

Zurück in die Vergangenheit geht es zum Beispiel mit dem Kommando:

```
date -s "3600 seconds ago"
```

Außerdem erlaubt »date«, das Datum in der international üblichen Schreibweise in Sekunden seit Epoch umzurechnen:

```
debian:~# date -d "2004-05-31 23:59:59" +%s
```

Alles Weitere ist einfach, Listing 2 zeigt einen Auszug des »settime«-Skripts: In Zeile 3 wird der Aufwachzeitpunkt in Sekunden seit Epoch umgerechnet, Zeile 4 bestimmt dann das aktuelle Datum in Sekunden seit Epoch. Daraus wird in Zeile 6 die Zeitdifferenz in Sekunden bestimmt. Zeile 5 rechnet die eingestellte Aufwachzeit des Bios, hier eine Sekunde vor Mitternacht am 31. Mai 2004, in Sekunden seit Epoch um, also 1086040799 Sekunden.

Daraus bestimmt Zeile 7 die Differenz zur Normalzeit und speichert sie in der Datei »/etc/timediff«. Diese Differenz muss »correcttime« beim nächsten Start zur dann aktuellen Systemzeit hinzuaddieren, um wieder die Normalzeit zu erreichen – das klappt allerdings nur, wenn die Systemzeit in der Vergangenheit liegt.

Der »date«-Aufruf in Zeile 8 schickt die Systemzeit in die Vergangenheit, »hwclock« synchronisiert dann in Zeile 9 die RTC mit der Systemzeit. Wie schon in Zeile 5 handelt es sich dabei um Zeitangaben in UTC, ohne die Parameter »-u« bei »date« und mit »--localtime« beim »hwclock«-Aufruf landet die Lokalzeit in der RTC.

Zeitkorrektur beim Systemstart

Das »settime«-Skript wird nur bei Bedarf aufgerufen, wenn der Rechner zeitgesteuert aufwachen soll – dann jedoch unmittelbar vor dem Shutdown. Das bedeutet, »correcttime« muss feststellen, ob es sich bei der RTC-Zeit um die Normalzeit handelt oder ob der Rechner in der Vergangenheit zu Hause ist.

Listing 3 zeigt einen Auszug aus »correcttime«. In Zeile 3 bestimmt der »date«-Aufruf das Datum der letzten Änderung der Datei »/etc/timediff« in Sekunden seit Epoch. Liegt dieses Änderungsdatum in Zeile 5 in der Zukunft, wurde die Systemzeit eindeutig in die Vergangenheit zurückversetzt. Liegt das Änderungsdatum hingegen in der Vergangenheit, ist die Systemzeit die aktuelle Normalzeit.

In den Zeilen 6 bis 8 liest »correcttime« die Zeitdifferenz zwischen Systemzeit und Normalzeit aus der Datei »/etc/timediff« und addiert sie auf die momentane Systemzeit, womit der Rechner wieder Normalzeit erreicht. Die Normalzeit wird noch in die RTC übertragen, da die meisten Distributionen beim Systemstart die Systemzeit noch einmal explizit mit der RTC synchronisieren.

Strom sparen – dem Geldbeutel zuliebe

Wie teuer Energie ist, sieht man spätestens bei der jährlichen Stromrechnung. Stromsparen ist möglich, mit etwas Überlegung sogar fast ohne Einbußen beim Komfort. So verbrauchen allein fünf normale Arbeitsplatz-PCs (je 150 Watt), die – wie in vielen Büros üblich – Tag und Nacht laufen, schon über 6500 Kilowattstunden Strom oder umgerechnet 1200 Euro pro Jahr. Ein einziger Server mit 400 Watt Leistung verbraucht allein schon 3500 Kilowattstunden oder 630 Euro pro Jahr.

Würden die Computer um 21 Uhr automatisch herunter- und werktags um 6 Uhr wieder hochfahren, verbrauchen die Arbeitsplatzrechner pro Jahr über 3600 und der Server knapp 2000 Kilowattstunden weniger Strom. Zusammen sind das fast 1000 Euro Ersparnis, von der entsprechend geringeren Umweltbelastung einmal ganz abgesehen.

Wird die EDV außerhalb der üblichen Arbeitszeiten benötigt, reicht bei den Arbeitsplätzen ein Druck auf den Einschalter – und der Server wird per Wake on LAN gestartet, etwa von den Init-Skripten der Workstations aus. ■

Infos

[1] Mainboard-Kompatibilitätsliste für ACPI-Wakeup: [http://linvdr.org/wiki/index.php?pagename=LinVDR-Mainboards]

[2] NVRAM-Wakeup: [http://sourceforge.net/projects/nvram-wakeup/]

[3] Anleitung zum NVRAM-Wakeup von Hubertus Sandmann: [http://home.t-online.de/home/hubertus.sandmann/vdr.htm]

[4] Die Skripte »settime« und »correcttime«: [http://www.linux-magazin.de/Service/Listings/2004/08/wakeup]

Listing 2: Auszug aus »settime«

```
01 #!/bin/bash
02 BiosWakeup="2004-05-31 23:59:59"
03 Wakeup=`date -d "$1" +%s`
04 Now=`date +%s`
05 Bios=`date -u -d "${BiosWakeup}" +%s`
06 Diff=$[${Wakeup}-${Now}]
07 echo "$[${Now}-${Bios}]" > /etc/timediff
08 date -u -s "${BiosWakeup} ${Diff} seconds ago" >/dev/null
09 hwclock -w --utc
```

Listing 3: Auszug aus »correcttime«

```
01 #!/bin/bash
02 if [ -r /etc/timediff ]; then
03   Timediff=`date -r /etc/timediff +%s`
04   Now=`date +%s`
05   if [ "${Timediff}" -gt "${Now}" ]; then
06     Diff=`cat /etc/timediff | head -n 1`
07     date -s "+${Diff} seconds" >/dev/null
08     hwclock -w --noadjfile --utc
09     exit 0
10   fi
11 fi
```