L'orologio del sistema non serve solo a fornire l'indicazione della data e dell'ora corrente; da esso dipende anche il buon funzionamento della pianificazione dei processi (*scheduling*). Nella pianificazione dei processi, poi, va considerato anche il livello di priorità di questi (nella sezione 36.10 viene trattato il problema della sincronizzazione dell'orologio attraverso la rete con il protocollo NTP). A fianco della gestione dell'orologio, il sistema Unix tipico prevede almeno un programma per la consultazione del calendario.

11.1 Orario locale

Generalmente, quando si considera la differenza di orario tra un paese e un altro si pensa ai fusi orari. In questa ottica, per stabilire l'orario basterebbe conoscere il fuso orario in cui ci si trova. Tuttavia, l'utilizzo dell'ora estiva in molti paesi, in forme differenti a seconda di

quanto stabilito dai vari governi, rende la determinazione dell'orario una cosa più complessa.

Per risolvere il problema si utilizza generalmente una sorta di base di dati contenente le regole con cui stabilire l'orario di ogni paese. A questo si abbina un orologio che mantiene un orario di riferimento, generalmente il tempo universale, dal quale il sistema è in grado di calcolare l'orario locale esatto.

Alcuni paesi utilizzano una notazione, generalmente di tre lettere, per indicare i vari fusi orari. Queste sigle non rappresentano uno standard per tutti, quindi vanno usate con prudenza. Il modo più sicuro per indicare un fuso orario è sempre quello di specificare il paese o una città precisa.

Alcune sigle sono particolarmente importanti, sia perché si usano spesso nella documentazione tecnica, sia perché appaiono nei messaggi dei programmi che si occupano di gestire l'orologio del sistema.

- UT (*Universal time*), UTC (*Universal Coordinated Time*)
  Il tempo universale, corrispondente all'ora solare di Greenwich.
- GMT (*Greenwich mean time*)
  Il modo tradizionale di indicare l'orario solare di Greenwich, corrispondente al tempo universale.
- DST (*Daylight savings time*)
  Non rappresenta un orario preciso, ma uno spostamento dell'orario per sfruttare meglio il periodo di illuminazione diurna durante la stagione estiva. In pratica si abbina questa sigla a quella del fuso orario, a indicare che l'ora solare corrispondente si ottiene sottraendo un'ora.
- CET (*Center Europe time*)
  L'orario dell'europa centrale.
- CEST (*Center Europe summer time*)
  L'orario dell'europa centrale durante il periodo estivo (in anticipo di un'ora sul tempo CET).

11.2 Distinzione tra hardware e software

Si distingue tra due orologi indipendenti: quello contenuto dell'hardware dell'elaboratore e quello gestito dal kernel del sistema. Per il sistema operativo, quello che conta è l'orario fornito dal kernel, ma l'orologio hardware è importante perché è in grado di funzionare anche quando l'elaboratore è spento. In pratica, all'avvio attraverso la procedura di inizializzazione del sistema, viene allineato l'orario del kernel con quello hardware.

11.2.1 Orologio hardware

L'orologio hardware è quello che appartiene alla parte fisica dell'elaboratore e normalmente è incorporato nella scheda madre. È alimentato attraverso una piccola batteria, in modo da poter funzionare anche quando l'elaboratore è spento.

Chi utilizza l'architettura x86 tende a chiamarlo «orologio del BIOS» oppure «orologio CMOS», dal momento che BIOS, CMOS e orologio, sono cose che tendono a confondersi. Si tratta comunque, sempre della stessa cosa.

Utilizzando un elaboratore x86, il modo migliore per regolare questo orologio è quello di utilizzare le funzioni del programma di configurazione della memoria CMOS, residente normalmente nella ROM e accessibile attraverso una combinazione di tasti al momento della verifica della memoria RAM.

È importante scegliere il tipo di orario su cui deve allinearsi l'orologio hardware. Generalmente, la scelta è tra la propria ora locale, o il tempo universale (UTC). Se non ci sono problemi di conflitti con altri sistemi operativi differenti dai sistemi GNU, è importante che si utilizzi come riferimento il tempo universale. In questo modo, è il sistema operativo a occuparsi di calcolare la differenza in base al fuso orario e all'eventuale ora estiva.

11.2.2 Orologio del kernel e orario locale

L'orologio del kernel viene impostato all'avvio, in base a quanto indicato dall'orologio hardware. Successivamente, finché il sistema resta in funzione, non viene più interpellato l'orologio hardware (a meno che il proprio sistema non abbia una configurazione particolare per qualche motivo).

Il sistema deve quindi sapere se l'orologio hardware è impostato sull'orario locale o sul tempo universale, per stabilire in che modo deve essere allineato l'orologio del kernel. Dal punto di vista dell'utilizzatore, nel primo caso occorre intervenire manualmente sull'orologio hardware quando inizia o termina il periodo dell'ora estiva, nel secondo caso no.

Il kernel tiene traccia esclusivamente del tempo universale, attraverso un numero che rappresenta il tempo trascorso in secondi dall'ora zero del primo gennaio 1970. Per conoscere l'orario locale si utilizza un file di configurazione, '`/etc/localtime`', contenente le informazioni necessarie a calcolarlo.

11.3 Modifica dell'orario

Se l'orologio del sistema è errato (intendendo in questo anche la data), è il caso di intervenire attraverso diverse azioni possibili. Il modo più semplice, nel senso che comporta meno complicazioni, è il riavvio del sistema impostando correttamente l'orologio hardware, eventualmente tenendo presente se si deve utilizzare il tempo universale.

Se il sistema non può essere riavviato, si deve intervenire attraverso il programma '**`date`**', come viene mostrato in seguito. Ma così facendo, dal momento che il sistema operativo è in funzione si provocano degli squilibri, sia nel sistema di pianificazione dei processi (Cron), sia in alte situazioni (anche il sistema grafico X può risentirne). Il minimo che può capitare è di osservare un'intensa attività del sistema dovuta all'avvio di processi da parte del demone '**`cron`**'.

Tuttavia, quando si interviene sull'orologio di un sistema in funzione, con un cambiamento importante, bisogna accettare il rischio di dover riavviare il sistema, se ci si accorge che tutto è diventato instabile.

In alternativa alla modifica dell'orologio del sistema, si può agire sull'orologio hardware attraverso il programma '**`clock`**'. Così, al prossimo riavvio l'orario dovrebbe risultare corretto, senza infastidire la sessione di lavoro attuale.

Quando si decide di modificare l'orario di sistema attraverso '**`date`**', dal momento che il più difficile è fatto, conviene anche aggiornare l'orologio hardware attraverso '**`clock`**'.

11.4 Strumenti per la gestione dell'orologio

Attraverso il programma '**`date`**' si può leggere o impostare la data e l'ora del sistema. Dal momento che il kernel gestisce l'orologio con riferimento al tempo universale, è necessaria un'opportuna conversione che avviene per mezzo di quanto indicato nel file '`/etc/localtime`', che generalmente è un collegamento simbolico al file adatto, contenuto nella directory '`/usr/share/zoneinfo/`' (queste collocazioni sono definite in base alla gerarchia standard dei sistemi GNU/Linux, a cui tutte le distribuzioni dovrebbero uniformarsi).

Il programma '**`clock`**' permette di leggere o impostare la data e l'ora dell'hardware. Utilizza il file '`/etc/adjtime`' per permettere un aggiustamento automatico del suo valore, quando si conosce esattamente di quanti secondi sbaglia ogni giorno.

11.4.1 Utilizzo di «date»

Il programma '**`date`**'[1] permette di conoscere o di modificare la data e l'ora del sistema, cioè quella gestita dal kernel. L'utente comune può utilizzare '**`date`**' per ottenere la data e l'ora attraverso

lo standard output, mentre solo l'utente '**root**' può intervenire per modificarne il valore.

`date` [*opzioni*] [*+formato*] [*data_orario*]

È importante tenere a mente che la modifica del valore contenuto nell'orologio del sistema può comportare instabilità.

La riga di comando di '**date**' si divide in tre parti principali: le opzioni, il formato di rappresentazione e la data. Il formato di rappresentazione è una stringa che descrive in che modo si vuole venga restituita la data o l'ora attuale. L'indicazione della data permette all'utente '**root**' di modificare la data e l'ora del sistema.

Tabella 11.1. Alcune opzioni.

Opzione	Descrizione
`-d` *data_orario* `--date=`*data_orario*	Emette la data e l'ora specificata attraverso l'argomento (composto da una stringa).
`-s` *data_orario* `--set=`*data_orario*	Modifica la data del sistema, secondo quanto indicato nell'argomento, costituito da una stringa. La data e l'ora possono essere espressi nello stesso modo in cui si può fare con l'opzione '**-d**'.
`-u` `--universal`	Emette o modifica la data riferita al tempo universale (UTC).

Se negli argomenti ne compare uno che inizia con un segno '**+**', questo viene interpretato come un formato di rappresentazione da utilizzare per emettere la data e l'ora. Si tratta di una stringa, dove tutti i caratteri vengono trattati per quello che sono, a eccezione delle direttive indicate nella tabella 11.2.

Tabella 11.2. Direttive per la rappresentazione delle informazioni data-orario di '**date**'.

Direttiva	Descrizione
`%%`	Rappresenta un simbolo di percentuale singolo.
`%n`	Rappresenta un codice di interruzione di riga.
`%t`	Rappresenta una tabulazione orizzontale.
`%s`	Numero di secondi trascorsi dall'epoca di riferimento (1970.01.01 00:00:00 UTC).
`%c`	Data e ora corrispondente alla stringa '**'%a %b %d %X %Z %Y'**'.
`%H`	L'ora secondo il formato «00..23».
`%I`	L'ora secondo il formato «01..12».
`%k`	L'ora secondo il formato « 0..23».
`%l`	L'ora secondo il formato « 0..12».
`%M`	Minuti secondo il formato «00..59».
`%S`	Secondi secondo il formato «00..59».
`%p`	AM o PM.
`%Z`	Sigla del fuso orario o nulla se non è determinabile.
`%r`	Orario in 12 ore, secondo il formato «hh:mm:ss AM/PM».
`%T`	Orario in 24 ore, secondo il formato «hh:mm:ss».
`%X`	Rappresentazione locale dell'orario, corrispondente generalmente alla stringa '**'%H:%M:%S'**'.
`%a`	Giorno della settimana abbreviato.
`%A`	Giorno della settimana esteso.

Direttiva	Descrizione
`%u`	Giorno della settimana, secondo il formato «1..7», dove uno corrisponde al lunedì.
`%U`	Settimana dell'anno, utilizzando la domenica come primo giorno, «00..53».
`%V`	Settimana dell'anno, utilizzando il lunedì come primo giorno, «01..53». La prima settimana (rappresentata come settimana «01»), per essere tale, deve includere almeno quattro giorni del nuovo anno.
`%w`	Giorno della settimana, con lo zero corrispondente alla domenica, «0..6».
`%b` `%h`	Mese abbreviato.
`%B`	Mese per esteso.
`%m`	Mese secondo il formato «01..12».
`%y`	Le ultime due cifre dell'anno, «00..99».
`%Y`	Anno per esteso.
`%C`	Secolo, pari al risultato intero di ***anno***/100.
`%d`	Giorno del mese secondo il formato «01..31».
`%e`	Giorno del mese secondo il formato « 1..31».
`%j`	Giorno dell'anno secondo il formato «001..366».
`%D`	Data secondo il formato «mm/gg/aa».
`%x`	Rappresentazione locale della data in forma numerica.

Quando non viene indicato un formato di rappresentazione della data, questa viene emessa secondo quanto si otterrebbe con la direttiva '**%c**'.

Se viene indicato un argomento che non appartiene alle opzioni e non inizia con il segno '**+**', allora viene inteso trattarsi dell'indicazione di una data ed eventualmente di un'ora, da utilizzare per modificare quella del sistema. La sintassi per indicare l'informazione data-orario, è la seguente:

MMGGhhmm[[*SS*]*AA*][*.ss*]

In pratica, si possono inserire otto cifre numeriche che rappresentano, rispettivamente a coppie: il mese, il giorno, le ore e i minuti. Di seguito si possono aggiungere altre due o quattro cifre che rappresentano l'anno («SS» sta per secolo). Infine, indipendentemente dal fatto che sia presente l'informazione dell'anno, possono essere aggiunte due cifre, separate da un punto, che rappresentano i secondi.

Segue la descrizione di alcuni esempi.

- `$ date -d '2 months 5 days'` [*Invio*]

  Restituisce la data corrispondente a due mesi e cinque giorni nel futuro.

- `$ date -d '1 month 3 hours ago'` [*Invio*]

  Restituisce la data corrispondente a un mese e tre ore fa.

- `$ date '+%d/%m/%Y'` [*Invio*]

  Emette la data nella forma ***giorno***/***mese***/***anno***, con l'anno per esteso.

- `# date 03151045` [*Invio*]

  Modifica la data del sistema in modo che corrisponda al 15 marzo dell'anno in corso, alle ore 10:45.

- `# date 03151045.10` [*Invio*]

  Modifica la data del sistema in modo che corrisponda al 15 marzo dell'anno in corso, alle ore 10:45:10.

- `# date 031510452012` [*Invio*]

  Modifica la data del sistema in modo che corrisponda al 15 marzo 2012 alle ore 10:45.

11.4.2 Utilizzo di «clock»

Il programma '**clock**' permette di accedere all'orologio hardware dell'elaboratore.

```
clock [-u] [-r | -w | -s | -a]
```

Tabella 11.3. Alcune opzioni.

Opzione	Descrizione
`-u`	Stabilisce che l'informazione data-orario contenuta nell'orologio hardware deve essere (oppure è) riferita al tempo universale (UTC).
`-r`	Legge la data e l'ora dell'orologio hardware e ne emette il contenuto attraverso lo standard output.
`-w`	Modifica la data e l'ora dell'orologio hardware, in base a quanto indicato dall'orologio di sistema. Quando il sistema fa affidamento sul fatto che l'orologio hardware contenga l'orario UTC, si utilizza questa opzione assieme a '**-u**'.
`-s`	Aggiorna la data e l'ora del sistema in base al contenuto dell'orologio hardware. Quando il sistema fa affidamento sul fatto che l'orologio hardware contenga l'orario UTC, si utilizza questa opzione assieme a '**-u**'.
`-a`	Aggiorna la data e l'ora del sistema in base al contenuto dell'orologio hardware, tenendo conto anche dell'errore sistematico indicato nel file '`/etc/adjtime`' e riaggiornando lo stesso orologio hardware.

Segue la descrizione di alcuni esempi.

- `# clock -r` [*Invio*]

  Legge e restituisce la data e l'orario contenuto nell'orologio hardware.

- `# clock -r -u` [*Invio*]

  La stessa cosa dell'esempio precedente, ma visualizza la data e l'ora locale, essendo l'orologio impostato sul tempo universale.

- `# clock -w -u` [*Invio*]

  Aggiorna l'orologio hardware, con riferimento al tempo universale, secondo l'orologio del sistema.

- `# clock -a -u` [*Invio*]

  Aggiorna l'orologio di sistema a partire da quello hardware, tenendo conto che l'orologio hardware è riferito al tempo universale, calcolando anche l'eventuale aggiustamento contenuto nel file '`/etc/adjtime`'.

11.4.3 Utilizzo di «hwclock»

Il programma '**hwclock**'[2] è una versione alternativa del programma '**clock**', specifica per i sistemi GNU/Linux, con il quale è comunque compatibile. In particolare, accetta anche quasi tutte le opzioni di quel programma.

```
hwclock [opzioni]
```

A differenza di '**clock**', '**hwclock**' è in grado di modificare direttamente l'orologio hardware, senza dover leggere l'orario fornito dal sistema operativo; inoltre, è in grado di gestire in modo autonomo il file di configurazione '`/etc/adjtime`' annotando l'errore dell'orologio hardware in base ai comandi di modifica dati dall'utente del sistema.

Tabella 11.4. Alcune opzioni.

Opzione	Descrizione
`-u` `--utc`	Stabilisce che l'informazione data-orario contenuta nell'orologio hardware deve essere (oppure è) riferita al tempo universale (UTC).
`-r` `--show`	Legge la data e l'ora dell'orologio hardware e ne emette il contenuto attraverso lo standard output.
`-w` `--systohc`	Modifica la data e l'ora dell'orologio hardware, in base a quanto indicato dall'orologio di sistema. Quando il sistema fa affidamento sul fatto che l'orologio hardware contenga l'orario UTC, si utilizza questa opzione assieme a '**-u**'.
`-s` `--hctosys`	Aggiorna la data e l'ora del sistema in base al contenuto dell'orologio hardware. Quando il sistema fa affidamento sul fatto che l'orologio hardware contenga l'orario UTC, si utilizza questa opzione assieme a '**-u**'.
`-a` `--adjust`	Aggiusta la data dell'orologio hardware in funzione del contenuto del file '`/etc/adjtime`'.
`--set`	Imposta l'orologio hardware in base all'indicazione data attraverso l'opzione '**--date**', modificando di conseguenza anche il file '`/etc/adjtime`'.
`--date=`***data_orario***	Definisce la data da attribuire all'orologio hardware. In pratica si usa solo assieme all'opzione '**--set**'.

Segue la descrizione di alcuni esempi.

- `# hwclock -r` [*Invio*]

  Mostra la data e l'ora dell'orologio hardware.

- `# hwclock -r -u` [*Invio*]

  Mostra la data e l'ora dell'orologio hardware, tenendo conto che quella è riferita al tempo universale, correggendola di conseguenza prima di visualizzarla.

- `# hwclock -u --set --date='04/01/2012 10:10:30'` [*Invio*]

  Imposta l'orologio hardware al 1 aprile 2012, alle ore 10:10 e 30 secondi. Contestualmente, '**hwclock**' modifica il file '`/etc/adjtime`' annotando l'errore sistematico dell'orologio hardware in base alla differenza riscontrata rispetto all'orario precedente.

- `# hwclock -a` [*Invio*]

  Corregge la data dell'orologio hardware in funzione delle informazioni contenute del file '`/etc/adjtime`'.

- `# hwclock -u -s` [*Invio*]

  Aggiorna l'orologio del sistema in base al valore riportato dall'orologio hardware, che risulta posizionato sul tempo universale.

11.4.4 File «/etc/adjtime»

Il file '`/etc/adjtime`' viene utilizzato da '**clock**' o da '**hwclock**' per tenere traccia dell'errore sistematico dell'orologio hardware. Contiene una sola riga di testo dove appaiono tre numeri, il cui significato è espresso dalla sintassi seguente:

```
[+|-] aggiustamento_giornaliero ultimo_utilizzo resto
```

Quando questo file non è configurato, appare la riga seguente:

```
0.0 0 0.0
```

Il primo valore rappresenta l'aggiustamento giornaliero in secondi che sarebbe necessario per fare sì che l'orologio hardware sia corretto. Per esempio, se vengono guadagnati sistematicamente cinque

secondi ogni giorno, si può modificare il primo valore indicando '**-5.0**'.

Il secondo numero viene gestito dai programmi che si occupano di aggiustare l'orologio e serve a memorizzare quando è stato fatto l'ultimo aggiustamento. Questo valore viene indicato con un numero che rappresenta quanti secondi sono trascorsi a partire dalla data di riferimento del sistema.

L'ultimo valore rappresenta la parte frazionaria di secondo che non ha potuto essere utilizzata nell'ultimo aggiustamento.

In pratica, l'amministratore del sistema deve occuparsi solo di modificare il primo valore, perché gli altri due sono gestiti direttamente dai programmi che si occupano di correggere l'orario. Eventualmente, utilizzando il programma '**hwclock**' con l'opzione '**--set**', non è nemmeno necessario preoccuparsi di questo.

Per fare in modo che l'orologio hardware venga corretto regolarmente attraverso le informazioni di questo file, è necessario che la procedura di inizializzazione del sistema sia stata predisposta in modo tale da provvedere ogni volta che viene avviato il sistema operativo. Di solito, le distribuzioni GNU sono già organizzate in questo modo; tuttavia potrebbe rimanere il problema di aggiornare l'orologio durante il funzionamento del sistema, in tutti i casi in cui l'elaboratore rimane acceso per tempi molto lunghi (si pensi a un nodo di Internet che offre dei servizi ininterrottamente). Evidentemente, occorre configurare il sistema Cron in modo da eseguire ogni giorno (a una certa ora) i comandi seguenti:

```
/sbin/clock -u -a
/sbin/clock -u -s
```

Oppure:

```
/sbin/hwclock -u -a
/sbin/hwclock -u -s
```

In pratica, prima si aggiorna l'orologio hardware e quindi si riallinea l'orologio del sistema operativo (negli esempi mostrati si presume che l'orologio hardware sia puntato sul tempo universale).

Il programma '**clock**' originale dovrebbe fare tutto utilizzando solo l'opzione '**-a**' (senza bisogno di essere riavviato con l'opzione '**-s**' per allineare il kernel). Tuttavia, se si tratta di un collegamento a '**hwclock**' che accetta la stessa opzione, l'aggiornamento dell'orologio del kernel deve essere richiesto in modo esplicito come è stato mostrato.

11.5 Cron

Nel bel mezzo della notte, mentre si sta lavorando isolati da qualunque rete, potrebbe capitare di notare un'intensa attività del sistema senza una giustificazione apparente. Di solito si tratta del demone '**cron**'. Cron è il sistema che si occupa di eseguire, attraverso il demone '**cron**', dei comandi in momenti determinati in base a quanto stabilito all'interno della sua configurazione, rappresentata dai file ***crontab***. Questi file possono essere diversi, solitamente uno per ogni utente che ha la necessità di pianificare l'esecuzione di alcuni comandi e uno generale per tutto il sistema.

I file crontab vengono creati attraverso il programma '**crontab**' e questo permette di non dovere sapere necessariamente dove devono essere collocati e in che modo vanno nominati. Oltre che per un fatto di relativa comodità, l'esistenza del programma '**crontab**' permette di evitare che i file crontab siano accessibili a utenti che non ne siano i proprietari. Inoltre, non è necessario preoccuparsi di avvisare il demone '**cron**' dell'avvenuto cambiamento nella situazione dei piani di esecuzione.[3]

L'output dei comandi che il sistema Cron mette in esecuzione, se non è stato ridiretto in qualche modo, per esempio a '`/dev/null`' o a un file, viene inviato con un messaggio di posta elettronica all'utente cui appartiene il file crontab.

Il demone '**cron**' viene avviato di norma durante la procedura di inizializzazione del sistema. Di questo demone ne esistono almeno due tipi diversi per i sistemi GNU: quello di Paul Vixie e quello di Dillon. Nelle sezioni seguenti si fa riferimento in particolare al sistema Cron di Paul Vixie.[4]

Vedere *cron(1)*, *crontab(1)* o *crontab(8)*, e *crontab(5)*.

11.5.1 Utilizzo di «cron»

Il programma '**cron**' è un demone funzionante sullo sfondo (*background*) che si occupa di interpretare i file crontab collocati in '`/var/spool/cron/crontabs/`' oltre a uno speciale, '`/etc/crontab`', il cui formato è leggermente diverso.

```
cron
```

Dal momento che la sostanza del funzionamento di questo programma sta nell'interpretazione dei file crontab, le altre notizie sul suo utilizzo sono riportate in occasione della presentazione di quei file.

11.5.2 Utilizzo di «crontab»

Il programma '**crontab**' permette di creare o modificare il file crontab di un utente determinato. In particolare, solo l'utente '**root**' può agire sul file crontab di un altro utente. Di solito, il programma '**crontab**' viene utilizzato con l'opzione '**-e**' per modificare o creare il file crontab.

```
crontab [opzioni]
```

I file crontab vengono poi utilizzati dal demone '**cron**' che si occupa di eseguire i comandi lì indicati.

Tabella 11.8. Alcune forme di utilizzo di '**crontab**'.

Sintassi	Descrizione
`crontab [-u utente] file`	Sostituisce il file crontab con il contenuto del file indicato come argomento.
`crontab -l [utente]`	Visualizza il file crontab dell'utente.
`crontab -e [utente]`	Crea o modifica il file crontab dell'utente.
`crontab -r [utente]`	Cancella il file crontab dell'utente.

Segue la descrizione di alcuni esempi.

- `$ crontab -e` [*Invio*]

  Inizia la modifica del file crontab dell'utente.

- `$ crontab -l` [*Invio*]

  Visualizza il contenuto del file crontab dell'utente. Il suo contenuto potrebbe apparire come nel listato seguente:

  ```
  # DO NOT EDIT THIS FILE - edit the master and reinstall.
  # (/tmp/crontab.1466 installed on Thu Aug 21 17:39:46 1997)
  10 6 * * * echo "ciao ciao"
  ```

- `$ crontab -r` [*Invio*]

  Elimina il file crontab dell'utente.

11.5.3 File «/var/spool/cron/crontabs/*»

I file contenuti nella directory '`/var/spool/cron/crontabs/`' sono i file crontab degli utenti comuni, creati generalmente attraverso il programma '**crontab**'. Ogni utente ha il proprio con il nome corrispondente all'utente stesso; i comandi contenuti al suo interno vengono eseguiti con i privilegi dell'utente proprietario del file crontab.

Le righe vuote sono ignorate e così anche quelle dove il primo carattere diverso da uno spazio lineare (sia spazi veri e propri, sia caratteri di tabulazione) è il simbolo '`#`', che serve così a introdurre dei commenti. Un record significativo può essere un assegnamento di una variabile di ambiente o un comando Cron.

L'assegnamento di una variabile può avvenire nel modo consueto, dove gli spazi attorno al segno di uguaglianza sono facoltativi e il valore assegnato può essere indicato eventualmente con virgolette (singole o doppie):

```
nome = valore
```

La possibilità di inserire degli assegnamenti di variabili di ambiente all'interno di un file crontab è una particolarità del sistema Cron di Paul Vixie.

Il demone '**cron**' utilizza delle variabili di ambiente per determinare il proprio comportamento. Alcune di queste ricevono un valore predefinito dal demone '**cron**' stesso, ma tutte, tranne ***LOGNAME***, possono essere modificate attraverso un assegnamento all'interno del file crontab.

Tabella 11.10. Utilizzo di alcune variabili di ambiente.

Variabile	Descrizione
SHELL	Il valore iniziale è '`/bin/sh`' stabilendo così che i comandi di Cron devono essere eseguiti facendo uso della shell standard. Normalmente, in un sistema GNU, '`/bin/sh`' è un collegamento alla shell predefinita, ovvero a Bash ('`/bin/bash`'), la quale, se avviata così, si comporta in modo compatibile con la shell standard.
LOGNAME	Il valore iniziale è il nome dell'utente e non può essere modificato.
HOME	Il valore iniziale è la directory personale dell'utente.
MAILTO	Non viene preassegnata dal demone '**cron**' e se risulta definita, ma non vuota, viene utilizzata per determinare il destinatario dei messaggi di posta elettronica che vengono generati. Se il contenuto di questa variabile è la stringa nulla ('**""**'), non viene inviato alcun messaggio. Se la variabile non esiste, il destinatario dei messaggi di posta elettronica è lo stesso utente a cui appartiene il file crontab.

Un file crontab tipico può contenere solo comandi di Cron. Il formato di questo può essere riassunto brevemente nel modo seguente:

```
data_orario comando
```

Il comando viene eseguito attraverso la shell indicata all'interno della variabile ***SHELL***, mentre l'indicazione data-orario si scompone in altri cinque campi.

```
minuti ore giorni_del_mese mesi giorni_della_settimana
```

I campi possono contenere un asterisco ('*****') e in tal caso rappresentano ogni valore possibile di quel campo. Per esempio, '*** * * * ***' rappresenta ogni minuto di ogni ora di ogni giorno del mese di ogni mese di ogni giorno della settimana.

A parte il caso degli asterischi, all'interno di questi campi si possono indicare dei valori numerici secondo gli intervalli seguenti:

Contesto	Intervallo possibile
minuti	da 0 a 59;
ore	da 0 a 23;
giorni del mese	da 1 a 31;
mesi	da 1 a 12;
giorni della settimana	da 0 a 7, dove sia zero, sia sette corrispondono alla domenica.

Per ognuno di questi campi, i valori possono essere indicati in vari modi con diversi significati.

- **Valori singoli**

  Un numero isolato all'interno di un campo indica che il comando deve essere eseguito quando l'orologio del sistema raggiunge quel valore. Per esempio, '**10 6 * * ***' rappresenta esattamente le ore 06:10 di ogni giorno.

- **Intervalli**

  Un intervallo, rappresentato da una coppia di numeri separati da un trattino, indica che il comando deve essere eseguito ogni volta che l'orologio del sistema raggiunge uno di quei valori possibili. Per esempio, '**10 6 1-5 * ***' rappresenta esattamente le ore 06:10 dei primi cinque giorni di ogni mese.

- **Elenchi**

  Un elenco, rappresentato da una sequenza di numeri separati da una virgola (senza spazi), indica che il comando deve essere eseguito ogni volta che l'orologio del sistema raggiunge uno di quei valori. Per esempio, '**10 6 1-5 1,3,5 ***' rappresenta esattamente le ore 06:10 dei primi cinque giorni di gennaio, marzo e maggio.

  Gli elenchi possono essere anche combinati con gli intervalli. Per esempio, '**10 6 1-5 1-3,5-7 ***' rappresenta esattamente le ore 06:10 dei primi cinque giorni di gennaio, febbraio, marzo, maggio, giugno e luglio.

- **Passo**

  Invece di indicare momenti precisi, è possibile richiedere una ripetizione o un passo. Questo può essere rappresentato con una barra obliqua seguita da un valore e indica che il comando deve essere eseguito ogni volta che è trascorsa quella unità di tempo. Per esempio, '***/10 6 * * ***' rappresenta le ore 06:10, 06:20, 06:30, 06:40, 06:50 e 06:00. In pratica, corrisponde a '**0,10,20,30,40,50 6 * * ***'.

  Il passo può essere combinato opportunamente con gli intervalli. Per esempio, '**0-30/10 6 * * ***' rappresenta le 06:00, le 06:10, le 06:20 e le 06:30. In pratica, corrisponde a '**0,10,20,30 6 * * ***'.

Quello che appare dopo i cinque campi dell'orario viene interpretato come un comando da eseguire. Più precisamente, viene considerato tale tutto quello che appare prima della conclusione della riga o di un segno di percentuale ('**%**'). Quello che eventualmente segue dopo il primo segno di percentuale viene interpretato come testo da inviare allo standard input del comando stesso. Se all'interno del testo da inviare appaiono altri segni di percentuale, questi vengono trasformati in codici di interruzione di riga.

Segue un esempio commentato di file crontab tratto da *crontab(5)*.

```
# Utilizza «/bin/sh» per eseguire i comandi,
# indipendentemente da quanto specificato all'interno di
# «/etc/passwd».
SHELL=/bin/sh

# Invia i messaggi di posta elettronica all'utente «tizio»,
# indipendentemente dal proprietario di questo file crontab.
MAILTO=tizio

# Esegue 5 minuti dopo la mezzanotte di ogni giorno.
5 0 * * *     $HOME/bin/giornaliero >> $HOME/tmp/out 2>&1

# Esegue alle ore 14:15 del primo giorno di ogni mese.
# L'output viene inviato tramite posta elettronica
# all'utente «tizio».
15 14 1 * *     $HOME/bin/mensile

# Esegue alle 22 di ogni giorno lavorativo (da lunedì al
# venerdì). In particolare viene inviato un messaggio di
# posta elettronica a «caio».
0 22 * * 1-5  mail -s "Sono le 22" caio%Caio,%%Vai a letto!%

# Esegue 23 minuti dopo mezzanotte, dopo le due, dopo le
# quattro,..., ogni giorno.
23 0-23/2 * * * echo "Ciao ciao"

# Esegue alle ore 04:05 di ogni domenica.
5 4 * * 0       echo "Buona domenica"
```

11.5.4 File «/etc/crontab»

Il file '`/etc/crontab`' ha un formato leggermente diverso da quello dei file crontab normali. In pratica, dopo l'indicazione dei cinque campi data-orario, si inserisce il nome dell'utente in nome del quale deve essere eseguito il comando indicato successivamente.

Nell'esempio seguente, tutti i comandi vengono eseguiti per conto dell'utente '**`root`**', ovvero, vengono eseguiti con i privilegi di questo utente.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# Run any at jobs every minute
* * * * * root [ -x /usr/sbin/atrun ] && /usr/sbin/atrun

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 1 * * * root run-parts /etc/cron.daily
02 2 * * 0 root run-parts /etc/cron.weekly
02 3 1 * * root run-parts /etc/cron.monthly

# Remove /tmp, /var/tmp files not accessed in 10 days
# (240 hours)
41 02 * * * root /usr/sbin/tmpwatch 240 /tmp /var/tmp

# Remove formatted man pages not accessed in 10 days
39 02 * * * root /usr/sbin/tmpwatch 240 /var/catman/cat?
```

Una parte dell'esempio mostrato è abbastanza comune nelle varie distribuzioni GNU e merita una spiegazione aggiuntiva. A metà dell'esempio appare l'avvio del comando '**`run-parts`**' a cadenza oraria, giornaliera, settimanale e mensile. Per esempio, la direttiva seguente avvia il comando '**`run-parts /etc/cron.hourly`**' ogni ora:

```
01 * * * * root run-parts /etc/cron.hourly
```

Il nome '**`run-parts`**' corrisponde a un programma (a volte realizzato in forma di script) che avvia tutti gli eseguibili contenuti nella directory indicata come argomento (salva la possibilità che alcuni file con nomi particolari siano esclusi); per cui, '**`run-parts /etc/cron.hourly`**' serve ad avviare tutto quello che c'è nella directory '`/etc/cron.hourly/`'.

Nella propria distribuzione GNU, il nome utilizzato per questo programma potrebbe essere diverso, così come i nomi delle directory, ma ciò che conta è comprendere che per inserire un'elaborazione nei momenti più comuni, basta mettere il programma o lo script relativo nella directory che rappresenta la cadenza desiderata. Questo, tra le altre cose, permette di realizzare dei pacchetti applicativi con tutto ciò che serve per fare in modo che il sistema Cron si preoccupi di loro nel modo corretto (senza dover intervenire manualmente nei file crontab).

Una realizzazione molto semplice di '**`run-parts`**' in forma di script potrebbe essere simile a quella seguente:

```
#!/bin/sh

for I in $1/* ; do
    if [ -x $I ]; then
        $I
    fi
done

exit 0
```

11.5.5 File «/etc/cron.d/*»

Alcune distribuzioni GNU introducono una variante, estendendo il crontab di sistema ai file contenuti nella directory '`/etc/cron.d/`'. Questi file sono altrettanti crontab con la stessa sintassi di '`/etc/crontab`', i quali vengono scanditi assieme a quello principale. L'utilità di questo sta nel fatto di evitare che i pacchetti che si installano debbano modificare il file crontab di sistema, limitandosi a gestire il proprio file particolare nella directory '`/etc/cron.d/`'.

11.6 Anacron

Cron è un sistema di pianificazione adatto principalmente per gli elaboratori che restano in funzione ininterrottamente per molto tempo; infatti, se non si accende mai l'elaboratore nell'intervallo di tempo in cui sarebbe previsto l'avvio di elaborazioni a cadenza giornaliera, settimanale o mensile, queste verrebbero automaticamente escluse. Per risolvere il problema e per garantire l'avvio di quelle elaborazioni, si può utilizzare Anacron.[5]

Anacron è un sistema di pianificazione che permette soltanto di programmare l'esecuzione di elaborazioni determinate a cadenza giornaliera, o a multipli di giorni. La sua logica è molto semplice: utilizza un file di configurazione, '`/etc/anacrontab`', concettualmente analogo al crontab di Cron, in cui si indica soltanto l'intervallo in giorni per l'esecuzione di processi determinati. Per mantenere memoria di ciò che è stato fatto, utilizza dei file nella directory '`/var/spool/anacron/`', annotando in che giorno ha eseguito un certo compito per l'ultima volta.

Questo sistema è gestito in pratica dall'eseguibile '**`anacron`**', il quale si comporta normalmente come un demone, restando in funzione solo per il tempo necessario a completare il suo lavoro: il giorno successivo, deve essere riavviato.

11.6.1 File «/etc/anacrontab»

Il file '`/etc/anacrontab`' si utilizza per configurare il comportamento di Anacron. Il file può contenere la definizione di alcune variabili di ambiente, così come si farebbe con una shell standard, quindi è composto da record (righe) che descrivono i vari compiti da eseguire. Come nel caso dei file crontab normali, le righe bianche e quelle vuote vengono ignorate; nello stesso modo sono ignorate le righe che iniziano con il simbolo '**`#`**'.

```
n_giorni  n_minuti_ritardo  nome_attribuito_al_job  comando
```

I record che definiscono i compiti di Anacron sono composti da campi separati da spazi bianchi di qualunque tipo:

1. il primo campo è un numero che esprime la cadenza in giorni con cui deve essere eseguito il comando;
2. il secondo campo è un altro numero che esprime un ritardo in minuti, che deve essere atteso prima di cominciare;
3. il terzo campo attribuisce un nome al compito;
4. l'ultimo campo è il comando corrispondente al compito e, in questo caso particolare, può contenere spazi.

Il significato dei campi dovrebbe essere abbastanza logico. In particolare, il ritardo viene stabilito per evitare che in un certo momento possano essere messi in funzione simultaneamente troppi processi, tenendo conto che è normale inserire l'avvio di Anacron all'interno della stessa procedura di inizializzazione del sistema.

È necessario attribuire un nome a ogni record (il compito, ovvero il *job*, secondo questa logica), per permettere a Anacron di annotarsi quando il comando relativo viene eseguito, in modo da determinare ogni volta se il tempo previsto è scaduto o meno.

La definizione di variabili di ambiente può essere necessaria, specialmente quando si prevede l'avvio di Anacron in modo automatico, attraverso la procedura di inizializzazione del sistema; in tal caso diventa fondamentale attribuire un valore alle variabili ***SHELL*** e ***PATH***. Si osservi l'esempio seguente:

```
# /etc/anacrontab

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

1       5       cron.daily      run-parts /etc/cron.daily
7       10      cron.weekly     run-parts /etc/cron.weekly
30      15      cron.monthly    run-parts /etc/cron.monthly
```

Oltre alla definizione delle variabili, si può vedere la dichiarazione di tre compiti che riguardano l'esecuzione di altrettanti comandi a cadenza giornaliera, settimanale e mensile. I tre compiti vengono avviati a distanza di cinque minuti uno dall'altro e anche il primo di questi attende cinque minuti per sicurezza. Si intuisce che questa pianificazione si affianchi a quella del crontab di sistema, in modo da garantire l'esecuzione degli script contenuti nelle directory '`/etc/cron.daily`', '`/etc/cron.weekly`' e '`/etc/cron.monthly`'.

11.6.2 Avvio di «anacron»

L'eseguibile '**anacron**' è quello che svolge in pratica il lavoro di gestione del sistema Anacron. Può essere avviato anche da utenti comuni, ma in generale questo non si fa, lasciando che sia la stessa procedura di inizializzazione del sistema a preoccuparsene.

```
anacron [opzioni]
```

Quando viene avviato, si mette normalmente a funzionare sullo sfondo, disimpegnandosi dal processo che lo ha avviato (diventando figlio del processo principale); quindi legge il file di configurazione e controlla nella directory '`/var/spool/anacron/`' quando è stata l'ultima volta che ha eseguito ogni compito previsto; di conseguenza avvia i comandi relativi ai compiti scaduti, aggiornando i file relativi nella stessa directory. L'avvio di questi comandi avviene di norma rispettando il ritardo indicato nel file di configurazione. Al termine, l'eseguibile '**anacron**' termina di funzionare.

Di solito, Anacron viene gestito direttamente dalla procedura di inizializzazione del sistema; in tal caso, è normale che l'eseguibile sia avviato con l'opzione '**-s**', che fa in modo di mettere in serie l'esecuzione dei compiti. In pratica, si evita che venga avviato un altro compito prima che sia terminata l'esecuzione di quello precedente.

Per evitare conflitti tra Anacron e Cron, quando potrebbe essere consentito a Cron di eseguire le stesse elaborazioni, conviene fare in modo che Cron «avvisi» Anacron nel momento in cui queste vengono svolte. In pratica, si utilizza l'opzione '**-u**' di '**anacron**', con la quale si ottiene proprio questo: aggiornare le date annotate nei file contenuti nella directory '`/var/spool/anacron/`'. Per comprendere meglio la cosa, si pensi alla situazione tipica, ovvero quella in cui si predispongono le solite directory '`/etc/cron.daily/`', '`/etc/cron.weekly/`' e '`/etc/cron.monthly/`', che devono contenere gli script da eseguire con cadenza giornaliera, settimanale e mensile, da parte di Cron o Anacron, indifferentemente. Per evitare che Anacron rifaccia quello che potrebbe avere già fatto Cron, si mette lo script '**0anacron**' in ognuna di queste directory (lo zero iniziale garantisce che questo sia il primo script a essere avviato). Nel caso si '`/etc/cron.daily/0anacron`', il contenuto potrebbe essere:

```
#!/bin/sh

anacron -u cron.daily
```

Questo dice a Anacron di aggiornare la data abbinata al compito '**cron.daily**', in modo da evitare di ripeterne l'esecuzione prima del necessario; inoltre, questo script non crea problemi a Anacron stesso, nel momento in cui dovesse essere avviato come parte del comando relativo a un compito.

11.7 At

L'esecuzione di un'elaborazione può essere necessaria una volta sola in una data e in un orario stabilito, oppure quando l'attività del sistema è ridotta. Anche se in generale la gestione della pianificazione dei processi è gestita dal sistema Cron, per questo scopo particolare gli si affianca il sistema At.[6]

A seconda dei casi, può trattarsi di un sottosistema di pianificazione dipendente da Cron, oppure può essere gestito da un demone indipendente: nel primo caso viene gestito dal programma '**atrun**', che a sua volta viene avviato periodicamente (di solito una volta al minuto) da Cron; nel secondo caso si fa affidamento sul demone '**atd**' che non interferisce in alcun modo con Cron.

Per verificare in che modo è organizzata la propria distribuzione GNU a proposito di At, si può osservare il file crontab di sistema, oppure quello dell'utente '**root**'; in alternativa si può osservare l'albero dei processi (con '**pstree**') per vedere se c'è in funzione il demone '**atd**'. Naturalmente, nel caso di gestione attraverso il demone '**atd**', dovrebbe esserci anche uno script che si occupa di avviarlo e di fermarlo nell'ambito della procedura di inizializzazione del sistema ('`/etc/init.d/atd`' o simile).

'**atrun**' o '**atd**' eseguono dei compiti accodati su diversi tipi di code dai programmi '**at**' e '**batch**'. Queste code sono classificate per importanza (priorità) attraverso una lettera alfabetica: le lettere minuscole si riferiscono a compiti accodati da '**at**', mentre quelle maiuscole rappresentano compiti accodati da '**batch**'.

La differenza tra questi due comandi sta nel fatto che il primo accoda comandi da eseguire in un momento determinato, mentre con il secondo, questi vengono eseguiti non appena l'attività del sistema raggiunge un livello sufficientemente basso da non disturbare l'esecuzione di processi più importanti.

Le code di questi compiti si trovano normalmente all'interno di '`/var/spool/cron/atjobs/`'.

L'utilizzo di '**at**' e '**batch**' può essere controllato attraverso due file: '`/etc/at.allow`' e '`/etc/at.deny`'. Se esiste '`/etc/at.allow`', solo gli utenti elencati al suo interno possono utilizzare '**at**' e '**batch**'. Se '`/etc/at.allow`' non esiste, viene preso in considerazione '`/etc/at.deny`' e gli utenti elencati al suo interno non possono utilizzare '**at**' e '**batch**'. Se questi due file non esistono, allora non si pongono limiti all'utilizzo di questi programmi.

11.7.1 Ambiente

Quando si decide di fare eseguire un comando in un momento successivo a quello attuale, si presenta il problema di definire l'ambiente in cui questo deve trovarsi. In linea di massima si può dire che si fa riferimento alla stessa situazione in cui ci si trova nel momento in cui si accoda il compito relativo. Si tratta dell'identità dell'utente (il numero UID), della directory corrente, della maschera dei permessi (per la creazione dei file) e delle variabili di ambiente. In particolare, le variabili ***TERM***, ***DISPLAY*** e il parametro '**$_**' non mantengono il loro valore originale.

11.7.2 Restituzione dell'output

Quando si pianifica l'esecuzione di un comando in un momento successivo, si ha il problema di stabilire dove debba essere diretto il suo output. Sarebbe buona norma indicarlo già nel comando, per esempio ridirigendo sia lo standard output che lo standard error in un file. Se qualcosa sfugge, l'output non ridiretto viene inviato all'utente che ha accodato il compito relativo attraverso un messaggio di posta elettronica (più precisamente attraverso '`/bin/mail`').

11.7.3 Utilizzo di «at», «batch», «atq» e «atrm»

Il programma '**at**' permette di pianificare, in un certo orario, l'esecuzione di un comando attraverso la shell predefinita, corrispondente a '`/bin/sh`'. Il comando o i comandi da eseguire vengono ricevuti dallo standard input, oppure da un file se ciò è specificato attraverso le opzioni.

```
at [opzioni] orario
```

L'orario di esecuzione può essere specificato in vari modi, anche combinando alcune parole chiave che in generale si riferiscono ai termini e alle abitudini dei paesi di lingua inglese.

Tabella 11.18. Orari: la tabella contiene un elenco di ciò che può essere utilizzato ragionevolmente in ambito internazionale.

Sintassi	Descrizione
hhmm [AM|PM] *hh* : *mm* [AM|PM]	Specifica un orario espresso in ore e minuti. Il simbolo '**:**' di separazione tra le due cifre che rappresentano le ore e quelle dei minuti, è facoltativo. L'utilizzo delle sigle '**AM**' e '**PM**' non è consigliabile: in generale, è preferibile esprimere gli orari utilizzando la notazione internazionale di 24 ore. A un orario si possono aggiungere due indicazioni ulteriori: '**today**' (oggi); '**tomorrow**' (domani).
mmggaa *mm* / *gg* / *aa* *gg* . *mm* . *aa*	Una data può essere espressa con un gruppo di sei cifre, separate eventualmente da una barra obliqua ('**/**'), con la sequenza mese-giorno-anno, oppure utilizzando il punto ('**.**') come separatore, con la sequenza giorno-mese-anno.
now	La sigla '**now**' viene usata per definire il momento attuale e si combina normalmente con una definizione di ritardo.
momento_di_partenza ↩ ↪+ *quantità_del_ritardo* ↩ ↪*unità_di_tempo*	Questa forma permette di stabilire un ritardo nell'avvio dei processi a partire dal momento indicato come riferimento. Il momento iniziale può essere '**now**' che si riferisce al momento presente, oppure un orario preciso. La durata del ritardo viene espressa da un numero che rappresenta una quantità definita subito dopo: '**minutes**' (minuti); '**hours**' (ore); '**days**' (giorni); '**weeks**' (settimane).

Tabella 11.19. Alcune opzioni.

Sintassi	Descrizione
-q *coda*	Permette di definire la coda. Si tratta di una lettera alfabetica minuscola, dalla '**a**' alla '**z**' e dalla '**A**' alla '**Z**'. La coda predefinita di '**at**' è quella corrispondente alla lettera '**a**', mentre quella di '**batch**' è quella della lettera '**b**'. Se si utilizzano lettere successive, i compiti associati ricevono un valore nice maggiore.
-m	Attraverso il sistema di posta elettronica, invia un messaggio all'utente che ha accodato la richiesta quando il compito è stato svolto.
-f *file*	Legge i comandi da eseguire da un file, invece che dallo standard input.
-l	Si comporta come '**atq**' e informa dei compiti in coda.
-d	Si comporta come '**atrm**'.
-v	Riguarda il funzionamento in modalità '**atq**' e permette di conoscere l'orario di esecuzione.
-c *job*…	Emette attraverso lo standard output il contenuto della coda associata al numero, o ai numeri dei compiti indicati. Viene in pratica emesso il contenuto del file che costituisce la coda associata al compito indicato.

Segue la descrizione di alcuni esempi.

- `$ at -f routine 13:30 + 3 days` [*Invio*]

  Esegue i comandi contenuti nel file '`routine`', fra tre giorni, alle ore 13:30.

- `$ at 15:00 tomorrow` [*Invio*]

  Esegue nel giorno successivo, alle ore 15:00, i comandi che vengono inseriti di seguito (si conclude con una combinazione [*Ctrl d*]).

- `$ at 10:00 10.11.2013` [*Invio*]

  Il 10 novembre 2013 alle ore 10:00 esegue i comandi da specificare successivamente.

La tabella successiva riassume l'utilizzo degli altri eseguibili che compongono At.

Tabella 11.20. '**batch**', '**atq**' e '**atrm**'.

Sintassi	Descrizione
batch [*opzioni*] [*orario*]	'**batch**' è normalmente un collegamento all'eseguibile '**at**'. Quando '**at**' viene avviato usando il nome '**batch**', i compiti associati vengono eseguiti non appena il livello di carico del sistema diventa ragionevolmente basso da permetterlo. In pratica, si può anche indicare un momento particolare (un orario), ma l'esecuzione può avvenire solo quando il sistema si trova ad avere un'attività ridotta.
atq [*opzioni*]	'**atq**' è normalmente un collegamento al programma '**at**'. Quando '**at**' viene avviato usando il nome '**atq**', emette l'elenco dei compiti in coda. Se viene specificata una coda attraverso l'opzione '**-q**', si limita a fare l'analisi di quella in particolare.
atrm *job*…	'**atrm**' è normalmente un collegamento al programma '**at**'. Quando '**at**' viene avviato usando il nome '**atrm**', elimina dalla coda i compiti specificati nella riga di comando.

11.7.4 Analisi di un esempio

L'esempio seguente dovrebbe permettere di comprendere il meccanismo attraverso cui viene registrata la situazione dell'istante in cui si accoda un compito. L'intenzione è quella di fare eseguire il programma '**ls**', alle ore 16:30, nella directory corrente nel momento in cui si accoda il compito, generando il file '`esempio`' (nella stessa directory) con l'output ottenuto. L'utente che accoda il comando è '**tizio**'.

```
$ at 16:30 [Invio]

  warning: commands will be executed using /bin/sh
at> ls > ./esempio [Invio]
at> [Ctrl d]

  Job 1 at 2012-09-22 16:30
```

A questo punto si può dare un'occhiata alla coda.

```
$ atq [Invio]

  1       2012-09-22 16:30 a tizio
```

Con i privilegi dell'utente '**root**', è possibile dare un'occhiata all'interno della directory '`/var/spool/cron/atjobs/`' per scoprire che è stato creato uno script, in questo caso denominato '**a0000400ee8c66**'. Segue il listato, molto semplificato, di quello che potrebbe essere il suo contenuto.

```
#!/bin/sh
# atrun uid=1001 gid=1001
# mail  tizio 0
umask 22
PS1=\\u@\\h:\\w\\\$\ ; export PS1
USER=tizio; export USER
MAIL=/home/tizio/mail/mbox; export MAIL
LANG=it_IT.UTF-8; export LANG
LOGNAME=tizio; export LOGNAME
SHLVL=1; export SHLVL
HOME=/home/tizio; export HOME
PATH=:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11; export PATH
cd /home/tizio || {
         echo 'Execution directory inaccessible' >&2
         exit 1
}
ls > ./esempio
```

Nella prima parte viene definita la maschera dei permessi attraverso il comando '**umask**'.

```
umask 22
```

Quindi seguono degli assegnamenti di variabili esportate che riproducono l'ambiente del momento in cui il compito è stato sottoposto.

```
PS1=\\u@\\h:\\w\\\$\ ; export PS1
USER=tizio; export USER
MAIL=/home/tizio/mail/mbox; export MAIL
LANG=it_IT.UTF-8; export LANG
…
```

Al termine, prima dell'esecuzione dei comandi richiesti, viene eseguito lo spostamento nella directory che, nel momento in cui è stato sottoposto il compito, risulta essere quella corrente. Se l'operazione fallisce viene interrotto lo script.

```
cd /home/tizio || {
         echo 'Execution directory inaccessible' >&2
         exit 1
}
```

11.8 Priorità

La priorità di esecuzione di un processo può essere modificata attraverso il valore nice che viene sommato, in modo algebrico, al valore di questa.

Quando si parla di priorità occorre però fare attenzione al contesto: di solito, un valore basso significa precedenza (quindi priorità) rispetto ai valori superiori. Spesso però si parla di priorità maggiore o minore in maniera impropria: quando si consulta della documentazione in cui si fa riferimento al concetto di priorità bisogna fare bene attenzione a non confondersi.

Dal momento che il valore nice viene sommato alla priorità, se nice è pari a zero non altera la priorità di esecuzione di un processo, se ha un valore positivo ne rallenta l'esecuzione, se invece ha un valore negativo ne accelera il funzionamento.

Alcuni programmi ricevono dal sistema un valore di priorità particolarmente basso per motivi fisiologici di funzionamento del sistema stesso. Il pretendere di portare agli stessi livelli di priorità altri programmi potrebbe comportare il blocco del sistema operativo. In pratica, anche se si tenta di dare a un processo un valore nice negativo, spesso il sistema non reagisce con un'eguale diminuzione del valore della priorità. Inoltre, solo l'utente '**root**' può attribuire ai processi valori nice inferiori a zero.

Di solito quindi, il valore nice viene usato per ridurre la velocità di esecuzione di un processo in modo da alleggerire l'impiego di risorse da parte dello stesso. Spesso si combina questa tecnica assieme all'utilizzo di elaborazioni sullo sfondo.

A fianco del problema della modifica della priorità di esecuzione di un programma c'è quello di mantenere in funzione un programma anche dopo la disconnessione del terminale dal quale questo viene avviato.

11.8.1 Utilizzo di «nice»

Il programma '**nice**' serve a eseguire un comando con un valore nice diverso dal normale. Minore è questo valore, maggiori sono le risorse (in termini di rapidità di esecuzione) che il sistema gli concede:

```
nice [-n variazione] [comando [argomenti]]
```

L'esecuzione senza alcun argomento visualizza il livello attuale del valore nice. Se viene specificato il nome di un comando, ma non viene indicato il livello di variazione (*adjustment*), il valore nice del comando indicato come argomento viene incrementato di 10 rispetto al valore attuale. Il livello di variazione può andare da un minimo di -20 a un massimo di +19, ma solo l'utente '**root**' può attribuire variazioni negative.[7]

11.8.2 Utilizzo di «renice»

Il programma '**renice**'[8] modifica il valore nice di uno o più processi:

```
renice -n priorità [-p pid…] [-g pid_di_gruppo…] [-u utente…]
```

È possibile indicare un processo singolo, un processo che faccia capo a un gruppo (un processo dal quale discendono altri processi), oppure tutti i processi che appartengono a un certo utente.

Tabella 11.28. Alcune opzioni.

Sintassi	Descrizione
`-p pid`	Indica esplicitamente che si fa riferimento a un processo singolo, indicato attraverso il numero PID.
`-g pid_di_gruppo`	Indica un processo, attraverso il numero PID, riferendosi anche a tutti i suoi processi discendenti.
`-u utente`	Indica che si fa riferimento a tutti i processi avviati con i privilegi dell'utente indicato per nome.

11.9 Calendario

La gestione dell'orologio di sistema serve alla vita del sistema operativo; pertanto, in questa logica, il tempo non esiste prima del 1970 e nemmeno esiste un futuro troppo lontano. Ci sono tuttavia dei programmi in grado di calcolare un calendario, più o meno «perpetuo», ma a questo proposito è bene tenere a mente il problema della riforma gregoriana che non è stata applicata in tutto il mondo nello stesso modo.

11.9.1 Utilizzo di «cal»

Il programma '**cal**'[9] serve a visualizzare un calendario molto semplice. Se non si indicano argomenti nella riga di comando, '**cal**' si limita a mostrare il mese attuale (in base alla data dell'orologio del sistema).

```
cal [opzioni] [mese [anno]]
```

È possibile indicare un mese particolare, di un certo anno, oppure solo un anno. Il mese si indica con un numero da 1 a 12, mentre l'anno si indica con un numero da 1 a 9999. A questo proposito, è bene precisare che se si indica un numero soltanto tra gli argomenti, questo viene inteso come l'anno e non il mese.

Il programma '**cal**' è sensibile alla configurazione locale, attraverso la variabile di ambiente ***LANG***, oppure le variabili ***LC_**** (si veda la sezione 16.11). Così facendo, si ottengono i nomi delle settimane e dei mesi nella lingua prescelta.

Tabella 11.29. Alcune opzioni.

Opzione	Descrizione
`-j`	Mostra la data giuliana; in pratica, si numerano i giorni a partire dall'inizio dell'anno, dove il primo giorno corrisponde al numero uno.
`-m`	Mostra il lunedì come il primo giorno della settimana.
`-y`	Mostra il calendario di tutto l'anno.

Segue la descrizione di alcuni esempi.

- `$` **`cal`** [*Invio*]

  Mostra il calendario del mese corrente.

- `$` **`cal -m`** [*Invio*]

  Come nell'esempio precedente, mettendo il lunedì all'inizio della settimana.

- `$` **`cal -m 1752`** [*Invio*]

  Mostra il calendario dell'anno 1752, che si considera essere quello dell'entrata in vigore della riforma gregoriana in Gran Bretagna e nelle sue colonie.

- `$` **`cal -m 9 1752`** [*Invio*]

  Mostra il calendario di settembre 1752. Il risultato che si ottiene è il seguente:

```
   settembre 1752
lu ma me gi ve sa do
    1  2 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

11.9.2 Utilizzo di «ccal»

Il programma '**`ccal`**'[10] serve a visualizzare un calendario molto semplice, utilizzando i colori se il terminale lo consente. Se non si indicano argomenti nella riga di comando, '**`ccal`**' si limita a mostrare il mese attuale (in base alla data dell'orologio del sistema).

`ccal` [*opzioni*] [*mese* [*anno*]]

A differenza del programma '**`cal`**' tradizionale, con questo è possibile annotare degli appuntamenti o delle scadenze in un file, per metterle in evidenza quando serve. Per approfondire l'uso di questo programma si deve consultare la pagina di manuale *ccal(1)*; qui vengono mostrati solo alcuni esempi elementari.

- `$` **`ccal`** [*Invio*]

  Mostra il calendario del mese in corso, dove appare evidenziato il giorno attuale (in questo caso si tratta del 23 settembre 2004).

```
   September 2004
Mo Tu We Th Fr Sa Su
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22<23>24 25 26
27 28 29 30
```

- `$` **`ccal 1 2010`** [*Invio*]

  Mostra il calendario del mese di gennaio 2010.

```
    October 2010
Mo Tu We Th Fr Sa Su
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

- `$` **`ccal 2010`** [*Invio*]

  Mostra il calendario di tutto l'anno 2010.


```
                                2010

        Jan                 Feb                 Mar
Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su
             1  2  3   1  2  3  4  5  6  7   1  2  3  4  5  6  7
 4  5  6  7  8  9 10   8  9 10 11 12 13 14   8  9 10 11 12 13 14
11 12 13 14 15 16 17  15 16 17 18 19 20 21  15 16 17 18 19 20 21
18 19 20 21 22 23 24  22 23 24 25 26 27 28  22 23 24 25 26 27 28
25 26 27 28 29 30 31                       29 30 31

        Apr                 May                 Jun
Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su
          1  2  3  4                1  2      1  2  3  4  5  6
 5  6  7  8  9 10 11   3  4  5  6  7  8  9   7  8  9 10 11 12 13
12 13 14 15 16 17 18  10 11 12 13 14 15 16  14 15 16 17 18 19 20
19 20 21 22 23 24 25  17 18 19 20 21 22 23  21 22 23 24 25 26 27
26 27 28 29 30        24 25 26 27 28 29 30  28 29 30
                      31
        Jul                 Aug                 Sep
Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su
          1  2  3  4                   1           1  2  3  4  5
 5  6  7  8  9 10 11   2  3  4  5  6  7  8   6  7  8  9 10 11 12
12 13 14 15 16 17 18   9 10 11 12 13 14 15  13 14 15 16 17 18 19
19 20 21 22 23 24 25  16 17 18 19 20 21 22  20 21 22 23 24 25 26
26 27 28 29 30 31     23 24 25 26 27 28 29  27 28 29 30
                      30 31
        Oct                 Nov                 Dec
Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su
             1  2  3   1  2  3  4  5  6  7           1  2  3  4  5
 4  5  6  7  8  9 10   8  9 10 11 12 13 14   6  7  8  9 10 11 12
11 12 13 14 15 16 17  15 16 17 18 19 20 21  13 14 15 16 17 18 19
18 19 20 21 22 23 24  22 23 24 25 26 27 28  20 21 22 23 24 25 26
25 26 27 28 29 30 31  29 30                 27 28 29 30 31
```

- `$` **`ccal 9 1752`** [*Invio*]

```
   September 1752
Mo Tu We Th Fr Sa Su
       1  2 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

  Mostra il calendario del mese di settembre dell'anno 1752, secondo la convenzione della Gran Bretagna, nel quale entra in vigore la riforma gregoriana.

11.9.3 Utilizzo di «ncal»

Il programma '**`ncal`**'[11] funziona in modo simile a '**`cal`**' (spesso si tratta dello stesso programma che si comporta in modo diverso a seconda del nome con cui lo si chiama), ma in più consente di calcolare il giorno di pasqua e di tenere conto della località per considerare l'entrata in vigore della riforma gregoriana.

`ncal` [*opzioni*] [`-s` *codice_paese*] [[*mese*] *anno*]

Il modello sintattico mostrato è generalizzato, ma in pratica, le opzioni che servono a calcolare il giorno di pasqua di un certo anno, sono in conflitto con quelle che servono a mostrare un calendario.

Tabella 11.35. Alcune opzioni.

Opzione	Descrizione

`-j`
`-m`
`-y`
	Queste opzioni funzionano come per '**`cal`**'.

`-J`	Mostra il calendario giuliano.
`-p`	Questa opzione, usata da sola, serve a mostrare l'elenco dei codici di identificazione dei paesi, da usare eventualmente con l'opzione '**`-s`**'. A fianco di ogni codice, appare la data presunta dell'entrata in vigore della riforma gregoriana.
`-s` *paese*	Richiede espressamente di calcolare la data, tenendo conto dell'entrata in vigore della riforma gregoriana, secondo il codice che deve indicare il paese a cui si vuole fare riferimento. In mancanza di questa opzione, il programma tenta di determinare il paese dalla configurazione locale.

Opzione	Descrizione
`-o` [*anno*]	Questa opzione, usata da sola, mostra la data della pasqua delle chiese ortodosse.
`-e` [*anno*]	Questa opzione, usata da sola, mostra la data della pasqua delle chiese occidentali (non ortodosse).

Segue la descrizione di alcuni esempi.

- `$` **`ncal -p`** [*Invio*]

```
AL Albania          1912-11-30    IT Italy          1582-10-04
AT Austria          1583-10-05    JP Japan          1918-12-18
AU Australia        1752-09-02    LI Lithuania      1918-02-01
BE Belgium          1582-12-14    LN Latin          9999-05-31
BG Bulgaria         1916-03-18    LU Luxembourg     1582-12-14
CA Canada           1752-09-02    LV Latvia         1918-02-01
CH Switzerland      1655-02-28    NL Netherlands    1582-12-14
CN China            1911-12-18    NO Norway         1700-02-18
CZ Czech Republic   1584-01-06    PL Poland         1582-10-04
DE Germany          1700-02-18    PT Portugal       1582-10-04
DK Denmark          1700-02-18    RO Romania        1919-03-31
ES Spain            1582-10-04    RU Russia         1918-01-31
FI Finland          1753-02-17    SI Slovenia       1919-03-04
FR France           1582-12-09    SE Sweden         1753-02-17
GB United Kingdom   1752-09-02    TR Turkey         1926-12-18
GR Greece           1924-03-09    US United States  1752-09-02
HU Hungary          1587-10-21    YU Yugoslavia     1919-03-04
IS Iceland          1700-11-16
```

- `$` **`ncal -s IT 10 1582`** [*Invio*]

  Mostra il mese di ottobre 1582 italiano, tenendo conto che la riforma gregoriana dovrebbe essere entrata in vigore il giorno 4 di quel mese:

```
    Ottobre 1582
lu  1 18 25
ma  2 19 26
me  3 20 27
gi  4 21 28
ve 15 22 29
sa 16 23 30
do 17 24 31
```

- `$` **`ncal -e 2005`** [*Invio*]

```
27 marzo 2005
```

  Mostra il giorno di pasqua dell'anno 2005.

11.9.4 Utilizzo di «gcal»

Il programma '**`gcal`**'[12] consente di ottenere un calendario, secondo vari formati, in grado di riconoscere il momento dell'introduzione della riforma gregoriana in base alla configurazione locale.

```
gcal [opzioni] [comando]
```

Il modello sintattico mostrato è semplificato rispetto alla documentazione originale, ma occorre tenere presente che questo programma è ricco di opzioni e di argomenti particolari, che qui vengono omessi. Probabilmente '**`gcal`**' è la soluzione «definitiva» al problema della gestione di un calendario, ma occorre leggere la documentazione originale, costituita dal documento *info gcal*. Qui vengono mostrati solo alcuni esempi banali, ignorando completamente tutte le potenzialità del programma.

- `$` **`gcal`** [*Invio*]

  Mostra il calendario del mese e dell'anno in corso, evidenziando il giorno corrente (in questo caso si tratta del 23 settembre 2004).

```
September 2004

Monday          6 13 20 27
Tuesday         7 14 21 28
Wednesday    1  8 15 22 29
Thursday     2  9 16<23>30
Friday       3 10 17 24
Saturday     4 11 18 25
Sunday       5 12 19 26
```


- `$` **`gcal 11 2004`** [*Invio*]

  Mostra il calendario del mese di novembre 2004.

```
November 2004

Monday       1  8 15 22 29
Tuesday      2  9 16 23 30
Wednesday    3 10 17 24
Thursday     4 11 18 25
Friday       5 12 19 26
Saturday     6 13 20 27
Sunday       7 14 21 28
```

- `$` **`gcal 2004`** [*Invio*]

  Mostra il calendario dell'anno 2004.

```
                                2004

     January               February              March                 April

Mo       5 12 19 26        2  9 16 23     1  8 15 22 29        5 12 19 26
Tu       6 13 20 27        3 10 17 24     2  9 16 23 30        6 13 20 27
We       7 14 21 28        4 11 18 25     3 10 17 24 31        7 14 21 28
Th    1  8 15 22 29        5 12 19 26     4 11 18 25        1  8 15 22 29
Fr    2  9 16 23 30        6 13 20 27     5 12 19 26        2  9 16 23 30
Sa    3 10 17 24 31        7 14 21 28     6 13 20 27        3 10 17 24
Su    4 11 18 25        1  8 15 22 29     7 14 21 28        4 11 18 25

      May                   June                  July                August

Mo       3 10 17 24 31     7 14 21 28        5 12 19 26        2  9 16 23 30
Tu       4 11 18 25     1  8 15 22 29        6 13 20 27        3 10 17 24 31
We       5 12 19 26     2  9 16 23 30        7 14 21 28        4 11 18 25
Th       6 13 20 27     3 10 17 24        1  8 15 22 29        5 12 19 26
Fr       7 14 21 28     4 11 18 25        2  9 16 23 30        6 13 20 27
Sa    1  8 15 22 29     5 12 19 26        3 10 17 24 31        7 14 21 28
Su    2  9 16 23 30     6 13 20 27        4 11 18 25        1  8 15 22 29

    September             October              November              December

Mo       6 13 20 27        4 11 18 25     1  8 15 22 29        6 13 20 27
Tu       7 14 21 28        5 12 19 26     2  9 16 23 30        7 14 21 28
We    1  8 15 22 29        6 13 20 27     3 10 17 24        1  8 15 22 29
Th    2  9 16<23>30        7 14 21 28     4 11 18 25        2  9 16 23 30
Fr    3 10 17 24        1  8 15 22 29     5 12 19 26        3 10 17 24 31
Sa    4 11 18 25        2  9 16 23 30     6 13 20 27        4 11 18 25
Su    5 12 19 26        3 10 17 24 31     7 14 21 28        5 12 19 26
```

- `$` **`echo $LANG`** [*Invio*]

```
it_IT
```

  `$` **`gcal 10 1582`** [*Invio*]

```
October 1582

Monday       1 18 25
Tuesday      2 19 26
Wednesday    3 20 27
Thursday     4 21 28
Friday      15 22 29
Saturday    16 23 30
Sunday      17 24 31
```

  Mostra il calendario del mese di ottobre 1582, durante il quale entra in vigore la riforma gregoriana. Si osservi che la configurazione della variabile di ambiente ***LANG*** fa riferimento all'Italia.

11.9.5 Utilizzo di «calendar»

Il programma '**`calendar`**'[13] serve a ricordare delle date importanti, attraverso un meccanismo molto semplice, il quale prevede almeno un file contenente delle righe, ognuna con una data e l'annotazione da ricordare:

```
calendar [opzioni]
```

Normalmente il programma viene usato senza argomenti, per ottenere l'elenco delle cose da ricordare per il giorno attuale e quello successivo (se ci si trova nel finesettimana, le informazioni possono prolungarsi fino al lunedì successivo).

Le date o le scadenze da ricordare possono essere memorizzate in uno o più file. Volendo semplificare, può trattarsi del file '`calendar`'

collocato nella directory corrente, oppure del file '`~/.calendar`', o anche di '`~/.calendar/calendar`'; in mancanza di uno di questi file personali, il programma va a cercare qualcosa che può essere generalizzato per tutto il sistema (nella directory '`/etc/calendar/`' o all'interno di '`/usr/share/calendar/`'). La gestione di questi file può essere anche molto complicata, pertanto, se serve, si deve consultare la pagina di manuale *calendar(1)*.

In condizioni normali, quando si installa il pacchetto che contiene il programma '**`calendar`**', si ottiene anche una configurazione predefinita con tante ricorrenze utili o curiose:

```
$ calendar [Invio]

set 15  Agatha Christie born in Torquay, England, 1890
set 15  ACM founded, 1947
set 15  Soviet Premier Nikita Khrushchev begins his 13 day
        tour of the US, 1959
set 15  The US Foreign Affairs Dept. becomes the US State
        Department, 1789
set 15  Foundation of Panama in Panama
set 15  Aujourd'hui, c'est la St(e) Roland.
set 15  La rosée de saint Albin
        est, dit-on, rosée de vin.
set 15  Wahl von Konrad Adenauer (CDU) zum Bundeskanzler,
        1949
set 15  Nürnberger Gesetze, 1935
set 16  Allen Funt born in Brooklyn, NY, 1914
set 16  The village of Shawmut, Massachusetts, becomes the
        city of Boston, 1630
set 16  Malaya, Sabah, Sarawak and Singapore unite to become
        Malaysia, 1963
set 16  Cherokee Strip Day in Oklahoma
set 16  Independence Days in Mexico and Papua New Guinea
set 16  B.B. King is born in Itta Bena, Mississippi, 1925
set 16  N'oubliez pas les Édith !
```

Quando invece l'utente predispone il proprio file '`~/calendar`', o '`~/.calendar/calendar`', tutto questo sparisce, a meno di richiamarlo in modo esplicito.

Il file che contiene le ricorrenze o le scadenze da ricordare, può anche essere piuttosto complesso. In generale conviene limitarsi a scrivere righe secondo la sintassi seguente, senza commenti e lasciando eventualmente solo righe vuote o bianche:

data*<HT>*descrizione

Tra la data e la descrizione ci deve essere un codice di tabulazione orizzontale (<*HT*>), che diventa l'unico modo per distinguere dove finisce un'informazione e dove incomincia l'altra. Tuttavia, nonostante la semplicità estrema di questa notazione, occorre fare attenzione a scrivere la data in modo che il programma sia in grado di riconoscerla. Sono ammissibili diverse modalità; per esempio si può mettere la sigla del mese secondo le convenzioni della propria nazionalità, ma in generale, è meglio usare delle sigle numeriche dove possibile e limitarsi alla lingua inglese per le denominazioni dei giorni della settimana o della festa pasquale. Segue un esempio di questo file che mostra diversi casi, dove in particolare le date che si riferiscono a una festività vanno segnate alla fine con un asterisco:

```
09.15   Ricordati di fare gli auguri a Tizio che è nato il 15 settembre 1970.
09.17   Ricordati di fare gli auguri a Caio che è nato il 17 settembre 1980.
Sun     Buona domenica!
.1      Inizio del mese.
.15     Seconda metà del mese.
Easter* Buona pasqua!
Easter-2*       Buona venerdì di pasqua, fra due giorni è la domenica di pasqua.
09 Wed+1        Primo mercoledì di settembre.
09 Wed+2        Secondo mercoledì di settembre.
09 Wed+3        Terzo mercoledì di settembre.
09 Wed-3        Terzultimo mercoledì di settembre.
09 Wed-2        Penultimo mercoledì di settembre.
09 Wed-1        Ultimo mercoledì di settembre.
Tue     "De venare e de marte, no se se sposa e no se parte"
Fri     "De venare e de marte, no se se sposa e no se parte"
```

Ecco cosa si potrebbe ottenere; si osservi anche il giorno previsto dall'esempio:

```
$ date [Invio]
Wed Sep 15 21:50:03 CEST 2004
$ cal [Invio]

   September 2004
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30

$ calendar [Invio]

Sep 15  Ricordati di fare gli auguri a Tizio che è nato il
        15 settembre 1970.
Sep 15  Seconda metà del mese.
Sep 15  Terzo mercoledì di settembre.
Sep 15  Terzultimo mercoledì di settembre.
```

Eventualmente, si può utilizzare il file seguente per le ricorrenze italiane comuni:

```
easter* domenica di pasqua
easter+1*       lunedì dell'angelo
easter-47*      martedì grasso
easter-46*      mercoledì delle ceneri
easter-7*       domenica delle palme

1.1*    capodanno
1.6*    epifania
4.1     pesce di aprile
4.25*   liberazione
5.1*    festa del lavoro
8.15*   ferragosto
11.1*   tutti i santi
12.8*   immacolata
12.25*  natale
12.26*  S. Stefano
12.31   ultimo dell'anno
```

Per la descrizione delle opzioni eventuali e degli altri modi possibili per scrivere le date delle ricorrenze, si può consultare la pagina di manuale *calendar(1)*.

11.9.6 Utilizzo di «pcal»

Il programma '**`pcal`**'[14] consente di ottenere un calendario in forma di file PostScript o di file HTML. Ciò consente di produrre un risultato stampato di qualità ottima oppure un file adatto alla pubblicazione elettronica. Questo programma consente anche di predisporre un file di ricorrenze o di scadenze da utilizzare per inserire delle note nel calendario che si ottiene. Questo file di ricorrenze e di scadenze può essere costruito anche cercando di ottenere una compatibilità minima con il programma '**`calendar`**'.

`pcal` [*opzioni*] [*anno* | {*mese anno*} | {*mese_inizio anno n_mesi*}]

Tralasciando inizialmente le opzioni, si può osservare dal modello sintattico che è possibile fare a meno di qualunque argomento; in tal caso si ottiene dallo standard output un file PostScript con il calendario del mese in corso. Se si indica un numero, si intende che si vuole ottenere tutto il calendario di quell'anno; se si mettono due numeri, si intende ottenere il calendario di un certo mese e di un certo anno; infine, se si indicano tre numeri, si vuole il calendario a partire da un certo mese, di un certo anno, per un totale di un certo numero di mesi.

Il programma '**`pcal`**' cerca un file contenente le ricorrenze o le scadenze da ricordare, tra diverse possibilità. In generale conviene usare per questo il file '`~/.calendar`', oppure è anche meglio usare l'opzione '**`-f`**' per indicare direttamente quale file usare per questo. All'interno di questo file si possono annotare dei commenti, preceduti dal cancelletto ('`#`'), si possono anche lasciare righe vuote o bianche e il resto delle righe deve essere costituito da direttive nella forma seguente:

data*<HT>…*descrizione

La data potrebbe essere separata dalla descrizione anche da più di uno spazio normale, ma conviene usare la forma mostrata (con il carattere di tabulazione, singolo o ripetuto), per avere più possibilità di usare questo file anche con altri programmi.

La data, o comunque il riferimento alla ricorrenza può essere indicato con una varietà di possibilità, che possono essere approfondite eventualmente leggendo la documentazione originale, ovvero la pagina di manuale *pcal(1)*; qui viene mostrato un solo esempio che intende rappresentare le festività e le ricorrenze principali secondo la tradizione italiana:

```
# Ricorrenze legate in qualche modo al periodo pasquale.
easter*                 domenica di pasqua
1st day after easter*   lunedì dell'angelo
47th day before easter* martedì grasso
46th day before easter* mercoledì delle ceneri
7th day before easter*  domenica delle palme

# Ora estiva.
last sunday in march    passaggio all'ora estiva (+1 ora)
last sunday in october  fine dell'ora estiva (-1 ora)

# Ricorrenze fisse.
1.1*                    capodanno
1.6*                    epifania
4.1                     pesce di aprile
4.25*                   liberazione
5.1*                    festa del lavoro
8.15*                   ferragosto
11.1*                   tutti i santi
12.8*                   immacolata
12.25*                  Natale
12.26*                  S. Stefano
12.31                   ultimo dell'anno
```

Nell'esempio si può osservare l'asterisco alla fine delle date che rappresentano una festività, da segnalare in modo speciale nel calendario. Si osservi anche che le date composte da giorno e mese, sono formate mettendo il mese per primo, secondo la convenzione che si usa nella lingua inglese, per non dover usare opzioni particolari nella riga di comando del programma.

Tabella 11.51. Alcune opzioni.

Opzione	Significato mnemonico	Descrizione
`-e`	*empty*	Genera un calendario «vuoto», nel senso che si ignorano le annotazioni delle ricorrenze o delle scadenze che possono essere disponibili nel file appropriato.
`-f` ***file***	*file*	Consente di specificare esplicitamente quale file usare per acquisire le ricorrenze o le scadenze.
`-o` ***file***	*output*	Consente di specificare il nome del file da generare, evitando così di ottenere il risultato dallo standard output
`-l`	*landscape*	Fa in modo che il risultato da stampare sia orientato in modo orizzontale. Questo è comunque il comportamento predefinito.
`-p`	*portrait*	Fa in modo che il risultato da stampare sia orientato in modo verticale.
`-j`	*julian*	Fa in modo che venga stampata anche la data giuliana, ovvero il numero del giorno dell'anno.
`-J`	*julian*	Fa in modo che venga stampata anche la data giuliana e la quantità di giorni rimanenti nell'anno.
`-m`	*moon*	Fa sì che vengano inserite le icone delle fasi lunari principali

Opzione	Significato mnemonico	Descrizione
`-M`	*moon*	Fa sì che per ogni giorno ci sia un'icona con la fase lunare.
`-b` ***giorno*** [`-`***giorno***]	*black*	Fa sì che il giorno o l'intervallo di giorni della settimana indicati siano rappresentati normalmente; per esempio si usa questa opzione per fare in modo che il sabato appaia come un giorno lavorativo comune.
`-F` ***giorno***	*first*	Fa in modo che il giorno della settimana indicato appaia per primo nella sequenza della settimana.
`-A`	*american*	Con questa opzione si dice al programma di interpretare le date come si fa nei paesi di lingua inglese, dove il mese precede il giorno. Questo comportamento è predefinito.
`-E`	*european*	Con questa opzione si dice al programma di interpretare le date come si fa nei paesi europei nei quali si usa mettere il giorno prima del mese.
`-H`	*html*	Genera un file in formato HTML.
`-a` ***linguaggio***		Consente di specificare la lingua da usare per i nomi dei giorni e dei mesi. La sigla del linguaggio è di due caratteri.

Segue la descrizione di alcuni esempi, dove si utilizza il file '`.calendar`' con il contenuto mostrato in precedenza.

- `$ pcal > calendario.ps` [*Invio*]

  Genera il file '`calendario.ps`' con i dati del mese in corso. Il risultato potrebbe essere simile a quello seguente:

- `$ pcal -o calendario.ps` [*Invio*]

  Fa esattamente la stessa cosa del comando precedente.

- `$ pcal -f .calendar -o calendario.ps` [*Invio*]

  Dichiara espressamente l'uso del file '`.calendar`' (contenuto nella directory corrente) per inserire nel calendario ricorrenze o scadenze.

- `$ pcal -m -o calendario.ps` [*Invio*]

  Fa in modo che vengano mostrate le fasi lunari principali.

- `$ pcal -M -o calendario.ps` [*Invio*]

  Fa in modo che vengano mostrate le fasi lunari in ogni giorno.

- `$ pcal -b sat -M -o calendario.ps` [*Invio*]

  Rispetto all'esempio precedente, fa in modo che il sabato appaia come un giorno lavorativo normale.

- `$ pcal -F mon -b sat -M -o calendario.ps` [*Invio*]

  Rispetto all'esempio precedente, fa in modo che le settimane inizino dal lunedì.

- `$ pcal -F mon -b sat -M -a it -o calendario.ps` [*Invio*]

  Rispetto all'esempio precedente, fa in modo che la denominazione dei mesi e delle settimane appaia in lingua italiana.

- `$ pcal -F mon -b sat -M -a it -o calendario.ps 3 2005` [*Invio*]

  Rispetto all'esempio precedente, fa in modo di generare la pagina del mese di marzo dell'anno 2005. Il suo aspetto dovrebbe essere quello seguente:

- `$ pcal -F mon -b sat -M -a it -o calendario.ps 2005` [*Invio*]

  Rispetto all'esempio precedente, genera il calendario di tutto l'anno 2005.

- `$ pcal -F mon -b sat -M -a it -H ↩`
  `↪ -o calendario.html 2005` [*Invio*]

  Rispetto all'esempio precedente, genera il calendario in formato HTML. Le fasi lunari potrebbero mancare.

11.9.7 File di ricorrenze e scadenze compatibile

I programmi '**calendar**' e '**pcal**' usano un file per le ricorrenze o le scadenze che ha caratteristiche simili, anche se non identiche. Tuttavia, se non si hanno grosse pretese, si può riuscire a realizzare un file unico, rispettando delle limitazioni: non si devono usare commenti, le date devono essere separate dalla descrizione attraverso stop di tabulazione, l'ultima direttiva deve essere conclusa da un codice di interruzione di riga.

Purtroppo il modo di rappresentare le date non è perfettamente compatibile, ma si può riuscire a scrivere qualcosa anche in modo ridondante, quando necessario. L'esempio seguente può funzionare sia con '**calendar**', sia con '**pcal**', anche se si possono osservare delle segnalazioni di errore a proposito di alcune righe che non vengono interpretate. Si osservi la necessità di usare direttive doppie per i calcoli relativi ai giorni collegati in qualche modo al periodo pasquale.


```
easter*                 domenica di pasqua
easter+1*               lunedì dell'angelo
1st day after easter*   lunedì dell'angelo
easter-47*              martedì grasso
47th day before easter* martedì grasso
easter-46*              mercoledì delle ceneri
46th day before easter* mercoledì delle ceneri

easter-7*               domenica delle palme
7th day before easter*  domenica delle palme

last sunday in march    passaggio all'ora estiva (+1 ora)
last sunday in october  fine dell'ora estiva (-1 ora)

1.1*                    capodanno
1.6*                    epifania
4.1                     pesce di aprile
4.25*                   liberazione
5.1*                    festa del lavoro
8.15*                   ferragosto
11.1*                   tutti i santi
12.8*                   immacolata
12.25*                  natale
12.26*                  S. Stefano
12.31                   ultimo dell'anno
```

Si osservi in particolare che le direttive che si riferiscono al cambio dell'ora estiva non vengono prese in considerazione da '**calendar**'.

[1] **GNU core utilities** GNU GPL

[2] **util-linux: hwclock** GNU GPL

[3] Indipendentemente dal fatto che il demone '**cron**' necessiti o meno di essere avvisato.

[4] **Vixie Cron** software libero con licenza speciale

[5] **Anacron** GNU GPL

[6] **At** GNU GPL

[7] **GNU core utilities** GNU GPL

[8] **Renice** UCB BSD

[9] **BSD utils** UCB BSD

[10] **Ccal** dominio pubblico

[11] **BSD utils** UCB BSD

[12] **Gcal** GNU GPL

[13] **BSD utils** UCB BSD

[14] **Pcal** software libero con licenza speciale