Cybersecurity con sistemi GNU/Linux

Dispensa del corso

Andrea Fazzi

Indice

1 Utilizzo avanzato del terminale

In questa sezione verranno discussi gli utilizzi più avanzati del terminale. In particolare, ci si soffermerà sui cosidetti *terminal multiplexer*.

1.1 I *terminal multiplexer*

I terminal multiplexer in Linux sono strumenti potenti che consentono agli utenti di creare più sessioni di terminale all'interno di una singola finestra del terminale. Questi strumenti sono particolarmente utili per gestire più processi contemporaneamente, mantenendo aperte le sessioni anche dopo la disconnessione, e per organizzare le finestre del terminale in modo efficiente.

Uno dei terminal multiplexer più popolari e ampiamente utilizzati è `tmux`. `tmux` permette agli utenti di creare, gestire e navigare tra più sessioni di terminale, facilitando l'organizzazione dei processi e la gestione delle finestre. Con `tmux`, è possibile dividere la finestra del terminale in più riquadri, ciascuno dei quali può eseguire un processo separato, e passare facilmente da una sessione all'altra.

Un altro esempio di terminal multiplexer è `screen`, che offre funzionalità simili a `tmux` ma con una sintassi leggermente diversa. Entrambi gli strumenti sono disponibili per la maggior parte delle distribuzioni Linux e possono essere installati tramite il gestore di pacchetti della distribuzione.

Utilizzare un terminal multiplexer può migliorare notevolmente l'efficienza e la produttività degli sviluppatori e degli amministratori di sistema, consentendo di lavorare con più processi contemporaneamente senza dover aprire molteplici finestre del terminale.

1.1.1 Riferimenti

1. https://opensource.com/article/21/5/linux-terminal-multiplexer

1.2 Introduzione a `tmux`

`tmux` è un potente strumento di gestione delle sessioni di terminale in Linux. Permette agli utenti di creare, gestire e navigare tra più sessioni di terminale all'interno di una singola finestra del terminale. Questa funzionalità è particolarmente utile per l'esecuzione di più programmi con una singola connessione, come quando si effettua una connessione remota a una macchina utilizzando Secure Shell (SSH) [1].

Per iniziare ad utilizzare `tmux`, basta digitare `tmux` nel terminale. Questo comando avvia un server `tmux`, crea una sessione predefinita (numero 0) con una singola finestra e si collega ad essa. Una volta connessi a `tmux`, è possibile eseguire qualsiasi comando o programma come si farebbe normalmente.

1.3 Installazione di `tmux`

`tmux` è un'applicazione disponibile nel repository di `Arch Linux`. Per installarla basterà utilizzare il comando `pacman`.

```
sudo pacman -S tmux
```

Per verificare che l'installazione sia andata a buon fine:

```
tmux -V
```

Il comando restituirà la versione installata.

1.3.1 Riferimenti

1. https://www.redhat.com/sysadmin/introduction-tmux-linux
2. https://linuxhandbook.com/tmux/
3. https://hamvocke.com/blog/a-quick-and-easy-guide-to-tmux/

4. https://linuxize.com/post/getting-started-with-tmux/
5. https://linuxconfig.org/introduction-to-terminal-multiplexer-tmux
6. https://github.com/tmux/tmux/wiki/Getting-Started
7. https://www.howtogeek.com/671422/how-to-use-tmux-on-linux-and-why-its-better-than-screen/
8. https://opensource.com/article/17/2/quick-introduction-tmux
9. https://wiki.archlinux.org/title/Tmux
10. https://github.com/tmux/tmux/wiki

1.4 Primi passi con `tmux`

Per eseguire `tmux` si dovrà semplicemente invocare il comando all'interno di una sessione di terminale.

```
tmux
```

È possibile staccare la sessione `tmux` premendo `Ctrl+B` seguito da `D`. `tmux` opera utilizzando una serie di scorciatoie da tastiera (keybindings) attivate premendo la combinazione "prefisso". Di default, il prefisso è `Ctrl+B`. Dopo di che, premere `D` per staccare dalla sessione corrente. La sessione continua ad eseguire in background anche dopo la disconnessione, permettendo di riprendere dove si è lasciati quando si è pronti a riconnettersi al server e riattaccarsi alla sessione esistente.

`tmux` fornisce una serie di scorciatoie da tastiera per eseguire comandi rapidamente all'interno di una sessione `tmux`. Alcune delle più utili includono:

- `Ctrl+B D` — Stacca dalla sessione corrente.
- `Ctrl+B %` — Suddividi la finestra in due pannelli orizzontalmente.
- `Ctrl+B "` — Suddividi la finestra in due pannelli verticalmente.
- `Ctrl+B` seguito da una freccia (sinistra, destra, su, giù) — Sposta tra i pannelli.
- `Ctrl+B X` — Chiudi il pannello.
- `Ctrl+B C` — Crea una nuova finestra.
- `Ctrl+B N` o `P` — Sposta alla finestra successiva o precedente.
- `Ctrl+B 0` (1,2...) — Sposta a una finestra specifica per numero.
- `Ctrl+B :` — Entra nella riga di comando per digitare comandi. L'auto-completamento tramite tab è disponibile.
- `Ctrl+B ?` — Visualizza tutte le scorciatoie da tastiera. Premere `Q` per uscire.
- `Ctrl+B W` — Apre un pannello per navigare tra le finestre in più sessioni.

1.4.1 Esercizio

Utilizza `tmux` per suddividere il terminale in due pannelli verticali. Nel pannello di sinistra apri uno script in Python utilizzando l'editor `nano`. Nel pannello di destra esegui il comando `htop`. Se `htop` non è presente nel sistema, procedi con la sua installazione attraverso il package manager `pacman`.

1.5 Configurazione di `tmux`

Il file di configurazione di `tmux`, noto come `tmux.conf`, è un file di testo che permette agli utenti di personalizzare l'ambiente di lavoro di `tmux` secondo le proprie preferenze. Questo file può essere posizionato in due luoghi principali:

- `~/.tmux.conf` per una configurazione specifica dell'utente corrente.
- `/etc/tmux.conf` per una configurazione globale, applicabile a tutti gli utenti del sistema.

Se il file `~/.tmux.conf` non esiste, può essere creato semplicemente eseguendo il comando `touch ~/.tmux.conf` nel terminale. Questo creerà un file di configurazione vuoto che può essere modificato per aggiungere le impostazioni desiderate.

La configurazione di `tmux` può includere una vasta gamma di opzioni, tra cui:

- Cambio del prefisso di comando predefinito.
- Abilitazione della modalità mouse.
- Impostazione di due prefissi.
- Cambio del comportamento predefinito del server.
- Inizio del conteggio dei numeri delle finestre e dei pannelli (Base-Index) a 1.
- Modifica dello sfondo del pannello corrente.
- …

Per esempio, per cambiare il prefisso di comando predefinito da `Ctrl+B` a `Ctrl+A`, si potrebbe aggiungere la seguente riga al file `tmux.conf`:

```
set-option -g prefix C-a
```

Dopo aver apportato modifiche al file di configurazione, è necessario ricaricarlo per applicare le nuove impostazioni. Questo può essere fatto eseguendo il comando `tmux source-file ~/.tmux.conf` dal terminale o utilizzando il comando `source-file ~/.tmux.conf` dalla modalità di comando di `tmux`. Per facilitare il processo, è possibile aggiungere un collegamento rapido nel file `tmux.conf` per ricaricare facilmente la configurazione:

```
bind r source-file ~/.tmux.conf \; display "Reloaded!"
```

Questo permette di ricaricare la configurazione premendo il prefisso seguito da `r`, visualizzando un messaggio di conferma.

1.5.1 Riferimenti

1. https://hamvocke.com/blog/a-guide-to-customizing-your-tmux-conf/
2. https://www.hostinger.com/tutorials/tmux-config
3. https://dev.to/iggredible/useful-tmux-configuration-examples-k3g
4. https://github.com/gpakosz/.tmux
5. https://arcolinux.com/everthing-you-need-to-know-about-tmux-configuration/
6. https://thevaluable.dev/tmux-config-mouseless/
7. https://github.com/samoshkin/tmux-config
8. https://wiki.archlinux.org/title/tmux
9. https://medium.com/@bhavik.n/customize-tmux-to-use-it-effectively-28b262c8b692

1.6 Creazione di layout personalizzati

Si supponga di voler creare un file di configurazione specifico per un progetto in cui il terminale viene diviso in tre parti: due colonne verticali di cui una a sua volta suddivisa orizzontalmente. In questo caso è possibile utilizzare uno script di configurazione personalizzato per `tmux`. Questo script può essere salvato in un file separato, ad esempio in `~/.config/tmux/split.conf`.

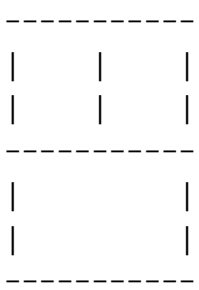
```
new -s splitted_session # crea una nuova sessione

selectp -t 0 # seleziona il primo pannello
splitw -h # divide il pannello corrente orizzontalmente in due parti

selectp -t 1 # seleziona il nuovo secondo pannello
splitw -v # divide il pannello corrente verticalmente in due parti
selectp -t 0 # torna al primo pannello
```

Una volta creato il file di configurazione, occorrerà “eseguirlo” con

```
tmux source-file ~/.config/tmux/split.conf
```

1.6.1 Esercizio

Crea un file di configurazione `tmux` per produrre un layout con due righe di cui la prima suddivisa in due colonne secondo uno schema simile a quello riportato sotto.



2 Secure Shell

Il protocollo SSH (Secure Shell) è un protocollo di rete che fornisce una connessione sicura e crittografata tra due sistemi informatici su una rete non sicura. OpenSSH è una delle implementazioni più comuni di SSH, che offre una serie di strumenti per la gestione delle chiavi SSH, l'autenticazione e la sicurezza delle connessioni.

2.1 Comando `ssh`

Il comando `ssh` è utilizzato per stabilire una connessione sicura con un server remoto. Ad esempio, per connettersi a un server remoto con l'indirizzo IP `192.168.1.100` come utente `utente`, il comando è:

```
ssh utente@192.168.1.100
```

Questo comando avvia una sessione SSH con il server specificato, richiedendo l'autenticazione dell'utente.

2.2 Comando `ssh-keygen`

`ssh-keygen` è uno strumento per generare una coppia di chiavi pubbliche/private per l'autenticazione SSH. Per generare una nuova chiave SSH, si può eseguire il comando `ssh-keygen` come nell'esempio sotto

```
ssh-keygen -t rsa
```

Agendo sulle opzioni del comando, si può specificare un percorso per salvare la chiave impostare una passphrase per la chiave privata per una maggioreq sicurezza.

```
ssh-keygen -t rsa -b 2048 -f ~/.ssh/mykey
```

Questo comando genera una chiave RSA di 2048 bit e la salva nel file `~/.ssh/mykey`. La chiave pubblica corrispondente sarà salvata in `~/.ssh/mykey.pub`.

2.3 Comando `ssh-copy-id`

`ssh-copy-id` è uno strumento che copia la chiave pubblica SSH di un utente in un server remoto, consentendo l'accesso senza password. Per copiare la chiave pubblica `mykey.pub` dell'utente corrente al server `192.168.1.100`, il comando è:

```
ssh-copy-id -i ~/.ssh/mykey.pub utente@192.168.1.100
```

Questo comando copia la chiave pubblica nel file `~/.ssh/authorized_keys` dell'utente remoto, consentendo l'accesso senza password. Dopo aver copiato la chiave, è possibile testare l'accesso senza password con:

```
ssh -i ~/.ssh/mykey utente@192.168.1.100
```

I nomi di default per le chiavi pubbliche e private sono rispettivamente `id_rsa.pub` e `id_rsa`. Utilizzando queste chiavi, l'accesso al server senza password si semplifica. Basterà infatti eseguire

```
ssh utente@192.168.1.100
```

2.4 Mobile Shell (mosh)

Mosh, acronimo di Mobile Shell, è un'applicazione di terminale remoto che risolve i problemi di connettività tipici di SSH, specialmente su reti mobili o instabili. A differenza di SSH, che utilizza TCP e richiede una connessione stabile, Mosh utilizza UDP, che è un protocollo senza connessione, permettendo una connessione più stabile e reattiva anche in presenza di interruzioni temporanee della connessione. Mosh mantiene una connessione attiva attraverso cambiamenti di indirizzo IP e sospensioni del dispositivo, rendendolo ideale per l'uso su dispositivi mobili.

2.4.1 Come funziona Mosh

Mosh inizia stabilendo una connessione SSH per l'autenticazione, utilizzando le stesse credenziali di SSH (ad esempio, password o chiavi pubbliche). Successivamente, avvia un server Mosh sul dispositivo remoto e stabilisce una connessione UDP per la comunicazione. Questo approccio consente a Mosh di gestire meglio la perdita di pacchetti e di mantenere una connessione attiva anche in presenza di interruzioni temporanee della rete.

2.4.2 Vantaggi di Mosh rispetto a SSH

- **Robustezza e Responsività**: Mosh è più robusto e reattivo rispetto a SSH, soprattutto su connessioni Wi-Fi, cellulari e lunghe distanze. Questo è dovuto al suo protocollo basato su UDP che gestisce meglio la perdita di pacchetti e imposta il tasso di fotogrammi in base alle condizioni della rete.
- **Eco Locale Intelligente**: Mosh fornisce un eco locale speculativo delle pressioni dei tasti, permettendo all'utente di vedere le proprie pressioni dei tasti quasi istantaneamente, senza attendere il round trip di rete. Questo migliora l'esperienza utente, specialmente su connessioni ad alta latenza.
- **Supporto per Roaming**: Mosh supporta il roaming, permettendo all'utente di cambiare la propria posizione fisica senza interrompere la sessione. Questo è particolarmente utile per gli utenti mobili che si spostano tra diversi luoghi.

2.4.3 Esempi

Per connettersi a un server remoto utilizzando Mosh, il comando è simile a quello di SSH, ma utilizzando `mosh` al posto di `ssh`. Ad esempio:

```
mosh utente@host
```

E' possibile utilizzare `mosh` e `tmux` insieme. Ad esempio, con la seguente linea di comando è possibile connettersi al server remoto ed eseguire tmux.

```
mosh utente@host -- tmux
```

2.5 Riferimenti

- [1] https://www.ssh.com/academy/ssh/copy-id
- [2] https://www.techtarget.com/searchsecurity/tutorial/Use-ssh-keygen-to-create-SSH-key-pairs-and-more
- [3] https://alexhost.com/faq/using-ssh-copy-id-ssh-keygen-commands-in-linux/
- [4] https://www.ssh.com/academy/ssh/keygen
- [5] https://www.digitalocean.com/community/tutorials/ssh-essentials-working-with-ssh-servers-clients-and-keys
- [6] https://docs.oracle.com/en/operating-systems/oracle-linux/openssh/openssh-WorkingwithSSHKeyPairs.html
- [7] https://www.redhat.com/sysadmin/configure-ssh-keygen
- [8] https://www.digitalocean.com/community/tutorials/how-to-configure-ssh-key-based-authentication-on-a-linux-server

- [9] https://www.thegeekstuff.com/2008/11/3-steps-to-perform-ssh-login-without-password-using-ssh-keygen-ssh-copy-id/
- [10] https://mosh.org/

2.5.1 Esercizio

2.5.2 Prerequisiti

Installa un server OpenSSH e assicurati di poter effettuare l'accesso. Crea un file layout.conf che funga da configurazione per una sessione tmux. La configurazione deve prevedere una qualche suddivisione della finestra. L'accesso via ssh dev'essere senza password.

2.5.3 Obiettivo

Effettuando un ssh da locale verso locale, una volta effettuato l'acccesso, la sessione tmux dev'essere automaticamente avviata.

3 Introduzione a `systemd`

Systemd è un sistema di inizializzazione e gestione dei servizi per Linux, che sostituisce il tradizionale SysVinit. È progettato per essere il punto di partenza per il sistema operativo, gestendo l'avvio dei servizi, la gestione delle sessioni utente, la configurazione di rete e molto altro. In Arch Linux, systemd è l'impostazione predefinita e viene utilizzato per gestire l'avvio del sistema e i servizi.

Systemd utilizza i "target" per raggruppare le unità insieme tramite dipendenze e come punti di sincronizzazione standardizzati. Questi servono uno scopo simile ai runlevel di SysVinit ma agiscono leggermente diversamente. Ogni target è nominato invece di essere numerato e serve uno scopo specifico con la possibilità di avere più attivi allo stesso tempo.

3.1 Riferimenti

- [1] https://wiki.archlinux.org/title/systemd
- [2] https://bbs.archlinux.org/viewtopic.php?id=214188
- [3] https://www.reddit.com/r/archlinux/comments/4lzxs3/why_did_archlinux_embrace_systemd/
- [4] https://stackoverflow.com/questions/10297969/systemd-on-arch-linux
- [5] https://wiki.archlinux.org/title/Systemd/User
- [6] https://superuser.com/questions/1025091/start-a-systemd-user-service-at-boot

- [7] https://halestrom.net/darksleep/blog/005_distrohop_p1/
- [8] https://www.youtube.com/watch?v=M51mbTRyL6U

3.2 Gestione di `systemd`

`systemctl` è il comando principale utilizzato per esaminare e controllare lo stato di `systemd` in un sistema Arch Linux. Questo strumento è fondamentale per la gestione dei servizi e delle unità del sistema. Ecco alcuni esempi di utilizzo di `systemctl`:

3.2.1 Avvio e Arresto di Servizi

Per avviare un servizio, ad esempio `sshd`, si utilizza:

```
sudo systemctl start sshd
```

Per fermare lo stesso servizio:

```
sudo systemctl stop sshd
```

3.2.2 Abilitazione e Disabilitazione di Servizi all'Avvio

Per abilitare un servizio all'avvio del sistema:

```
sudo systemctl enable sshd
```

Per disabilitare l'avvio automatico del servizio:

```
sudo systemctl disable sshd
```

3.2.3 Controllo dello Stato dei Servizi

Per controllare lo stato corrente di un servizio:

```
systemctl status sshd
```

3.2.4 Gestione dei Target

I target in systemd rappresentano stati di sistema o configurazioni specifiche. Per vedere il target corrente:

```
systemctl get-default
```

Per cambiare il target di default al prossimo avvio:

```
sudo systemctl set-default multi-user.target
```

3.2.5 Utilizzo di systemctl su una macchina remota

È possibile utilizzare `systemctl` per controllare un'istanza di `systemd` su una macchina remota tramite SSH:

```
systemctl -H user@host status sshd
```

3.2.6 Diagnostica dei servizi

Per visualizzare i servizi che hanno manifestato problemi alla partenza:

```
systemctl --state=failed
```

3.2.7 Riferimenti

- [1] https://wiki.archlinux.org/title/systemd
- [2] https://man.archlinux.org/man/systemctl.1.en
- [3] https://wiki.archlinux.org/title/Systemd/User
- [4] https://bbs.archlinux.org/viewtopic.php?id=260255
- [5] https://gist.github.com/bugyt/c149de4ba5b1ea79a077
- [6] https://superuser.com/questions/1025091/start-a-systemd-user-service-at-boot
- [7] https://wiki.archlinux.org/title/systemd/Timers
- [8] https://www.reddit.com/r/linux4noobs/comments/18fklyi/using_systemd_to_automatically_run_commands
- [9] https://stackoverflow.com/questions/10297969/systemd-on-arch-linux
- [10] https://man.archlinux.org/man/systemd.unit.5.en

3.3 Utilizzo di `journalctl`

`journalctl` è uno strumento potente per la gestione dei log in sistemi che utilizzano `systemd`, come Arch Linux. Questo strumento consente di visualizzare, filtrare e analizzare i messaggi di log generati dal sistema e dai servizi. Ecco alcuni esempi notevoli di utilizzo di `journalctl`:

3.3.1 Visualizzazione dei Log

Per visualizzare tutti i messaggi di log:

```
journalctl
```

3.3.2 Filtraggio dei Messaggi di Log

- Per visualizzare tutti i messaggi che corrispondono a un pattern specifico:

```
journalctl --grep=PATTERN
```

- Per visualizzare tutti i messaggi dal boot corrente:

```
journalctl -b
```

- Per visualizzare i messaggi dal boot precedente:

```
journalctl -b -1
```

- Per visualizzare i messaggi da una data specifica:

```
journalctl --since="2022-01-01"
```

- Per visualizzare i messaggi degli ultimi 20 minuti:

```
journalctl --since "20 min ago"
```

3.3.3 Seguire i nuovi messaggi

Per seguire i nuovi messaggi di log in tempo reale:

```
journalctl -f
```

3.3.4 Visualizzazione per Unità di Servizio

- Per visualizzare tutti i messaggi per un servizio specifico, ad esempio `httpd`:

```
journalctl -u httpd.service
```

- Per visualizzare i messaggi di log per un'unità utente specifica, ad esempio `dbus`:

```
journalctl --user -u dbus
```

3.3.5 Visualizzazione dei Messaggi di Errore

Per visualizzare solo i messaggi di errore, critico e allarme:

```
journalctl -p err..alert
```

3.3.6 Visualizzazione del Buffer del Kernel

Per visualizzare i messaggi del buffer del kernel:

```
journalctl -k
```

3.3.7 Visualizzazione dei Log in un File Specifico

Se il directory dei log contiene una grande quantità di dati di log, `journalctl` può richiedere diversi minuti per filtrare l'output. Può essere significativamente velocizzato utilizzando l'opzione `--file` per forzare `journalctl` a cercare solo nel journal più recente:

```
journalctl --file /var/log/journal/*/system.journal -f
```

3.3.8 Esercizio

Utilizzando il terminal multiplexer `tmux` suddividi il terminale in due pannelli. Nel pannello di sinistra esegui `journalctl` in modo da visualizzare i log del servizio `sshd`. Nel pannello di destra connettiti al server ssh in esecuzione sulla tua macchina. Qualora il servizio non fosse in esecuzione, fallo partire utilizzando `systemctl`. Salva una porzione dei log in un file di testo come prova di aver svolto l'esercizio.

3.3.9 Riferimenti

- [1] https://wiki.archlinux.org/title/Systemd/Journal
- [2] https://man.archlinux.org/man/journalctl.1.en
- [3] https://www.debugpoint.com/systemd-journalctl/
- [4] https://serverfault.com/questions/721963/how-do-you-use-systemds-journalctl-patterns
- [5] https://www.loggly.com/ultimate-guide/using-journalctl/
- [6] https://bbs.archlinux.org/viewtopic.php?id=208480
- [7] https://unix.stackexchange.com/questions/199988/how-to-inspect-systemd-journal-files-directly
- [8] https://kb.adamsdesk.com/operating_system/arch_linux_quick_reference_commands/

- [9] https://www.reddit.com/r/archlinux/comments/lskkdt/always_check_your_journalctl_when_weird_problem
- [10] https://wiki.archlinux.org/title/systemd

3.4 Timer di `systemd`

`systemd/Timers` in Arch Linux sono unità di sistema che permettono di pianificare l'esecuzione di servizi o eventi in base a specifici intervalli di tempo o eventi di sistema. Questi timer possono essere utilizzati come alternativa a `cron` per la pianificazione di attività. I timer di `systemd` supportano eventi di tempo calendario e monotonici, e possono essere eseguiti in modo asincrono.

3.4.1 Tipi di Timer

- **Timer Realtime (o Wallclock Timers)**: Attivano un evento in base a un calendario, similmente a come funzionano i cronjobs. Utilizzano l'opzione `OnCalendar=` per definire l'evento.
- **Timer Monotonici**: Attivano dopo un intervallo di tempo relativo a un punto di partenza variabile. Si fermano se il computer viene sospeso temporaneamente o spento. Esempi comuni includono `OnBootSec` e `OnUnitActiveSec` [1].

3.4.2 Creazione di un Timer

Per creare un timer, è necessario creare un file di unità con estensione `.timer` che descriva quando e come il timer deve attivare un'unità di servizio corrispondente. Ad esempio, per creare un timer che esegue un servizio ogni giorno alle 12:00, il file del timer potrebbe essere simile a questo:

```
[Unit]
Description=Esegui il mio servizio giornaliero

[Timer]
OnCalendar=daily
Persistent=true

[Install]
WantedBy=timers.target
```

E il file del servizio corrispondente potrebbe essere:

```
[Unit]
Description=Il mio servizio giornaliero

[Service]
ExecStart=/path/to/my-service
```

La sintassi `OnCalendar` in `systemd` è utilizzata per specificare quando un timer deve essere attivato, offrendo una maggiore flessibilità rispetto al formato cron tradizionale. La sintassi `OnCalendar` segue il formato `DOW YYYY-MM-DD HH:MM:SS`, dove `DOW` è il giorno della settimana (opzionale), e gli altri campi possono utilizzare un asterisco (`*`) per corrispondere a qualsiasi valore per quella posizione. Se l'ora non è specificata, viene assunta come `00:00:00`. Se la data non è specificata ma l'ora è, il prossimo match potrebbe essere oggi o domani, a seconda dell'ora corrente [5].

Ecco alcuni esempi di come utilizzare `OnCalendar`:

- Per eseguire un evento ogni giorno alle 10:00, puoi usare:

```
OnCalendar=*-*-* 10:00:00
```

- Per eseguire un evento ogni anno il 15 ottobre alle 17:48:00, puoi usare:

```
OnCalendar=*-10-15 17:48:00
```

- Per eseguire un evento ogni anno il 1 gennaio alle 00:00:00, puoi usare:

```
OnCalendar=*-01-01 00:00:00
```

- Puoi anche combinare questi campi per specificare eventi più precisi. Ad esempio, per eseguire un evento ogni mercoledì alle 17:48:00, puoi usare:

```
OnCalendar=Wed *-*-* 17:48:00
```

Questo può anche essere scritto come: `ini  OnCalendar=3Wed, 17:48`

Sì, `OnCalendar` in `systemd` ammette alias come "daily" per specificare intervalli di tempo più comuni. Questo alias può essere utilizzato per semplificare la pianificazione di eventi ricorrenti. Ad esempio, per eseguire un evento ogni giorno, puoi semplicemente utilizzare:

```
OnCalendar=daily
```

Questo è equivalente a specificare l'evento per ogni giorno alle 00:00:00. Altri alias comuni includono "weekly", "monthly", "yearly", e specifici giorni della settimana come "Mon", "Tue", "Wed", ecc.

Per esempio, per eseguire un evento ogni lunedì alle 10:00, potresti usare:

```
OnCalendar=Mon 10:00:00
```

O per eseguire un evento ogni settimana, potresti usare:

```
OnCalendar=weekly
```

Questi alias rendono più semplice e leggibile la configurazione dei timer in `systemd`, specialmente per intervalli di tempo ricorrenti comuni [4].

`systemd` offre strumenti come `systemd-analyze calendar` per validare e esaminare gli eventi del calendario utilizzati in un'espressione. Questo strumento analizza un'espressione di evento del calendario e fornisce la forma normalizzata e altre informazioni, come la data e l'ora del prossimo "elapso" (cioè, corrispondenza) e la quantità approssimativa di tempo prima che raggiunga il tempo di attivazione [5].

3.4.3 Gestione dei Timer

Per abilitare e avviare un timer, utilizza i comandi `systemctl enable` e `systemctl start`, ricordando di aggiungere il suffisso `.timer` al nome dell'unità. Per visualizzare tutti i timer attivi, esegui:

```
systemctl list-timers
```

Per elencare tutti i timer (inclusi quelli inattivi), usa:

```
systemctl list-timers --all
```

Se un timer si sincronizza, potrebbe essere utile eliminare il suo file `stamp-*` in `/var/lib/systemd/timers` (o `~/.local/share/systemd/` nel caso dei timer utente). Questi file di lunghezza zero contrassegnano l'ultima volta in cui ogni timer è stato eseguito. Se vengono eliminati, verranno ricostruiti al prossimo avvio del loro timer [1].

3.4.4 Esempi di Utilizzo

Per impostare un timer che effettui il backup della cartella home ogni mattina alle 10:00, e che si assicuri che il backup venga effettuato anche se il PC è spento, segui questi passaggi. Questo esempio utilizza un timer e un servizio di `systemd` per eseguire il backup.

3.4.4.1 Creazione del Servizio Prima di tutto, crea un file di servizio che eseguirà il backup. Puoi chiamarlo `home-backup.service` e metterlo in `/etc/systemd/system/` per un backup a livello di sistema, o in `~/.config/systemd/user/` per un backup a livello utente.

```
[Unit]
Description=Backup della cartella home

[Service]
Type=oneshot
ExecStart=/path/to/your/backup-script.sh
```

Assicurati che lo script di backup (`/path/to/your/backup-script.sh`) sia eseguibile e che contenga i comandi necessari per il backup.

3.4.4.2 Creazione del Timer Successivamente, crea un file di timer che attiverà il servizio sopra ogni mattina alle 10:00. Anche questo file va in `/etc/systemd/system/` per un backup a livello di sistema, o in `~/.config/systemd/user/` per un backup a livello utente. Chiamalo `home-backup.timer`.

```
[Unit]
Description=Timer per il backup della cartella home

[Timer]
OnCalendar=*-*-* 10:00:00
Persistent=true

[Install]
WantedBy=timers.target
```

L'opzione `Persistent=true` assicura che il timer si attivi non appena il sistema è avviato, anche se il tempo previsto per l'esecuzione è passato.

3.4.4.3 Abilitazione e Avvio del Timer Per abilitare il timer, esegui:

```
sudo systemctl enable home-backup.timer
```

E per avviare il timer immediatamente, senza doverlo abilitare:

```
sudo systemctl start home-backup.timer
```

Se stai creando un timer a livello utente, sostituisci `sudo` con `--user` nei comandi sopra.

3.4.4.4 Verifica Per verificare che il timer sia attivo e pianificato correttamente, puoi usare:

```
systemctl list-timers
```

Questo mostrerà tutti i timer attivi, inclusi quelli pianificati per il futuro.

3.4.4.5 Considerazioni

- Assicurati che lo script di backup sia configurato correttamente e che abbia i permessi necessari per eseguire il backup.
- Se il tuo sistema è spento quando il timer dovrebbe attivarsi, il timer si attiverà al prossimo avvio del sistema, grazie all'opzione `Persistent=true`.
- Ricorda che i timer a livello utente richiedono che l'utente sia loggato per funzionare, a meno che non sia configurato per "linger" con `loginctl enable-linger $USER`.

Questo metodo ti permette di pianificare il backup della cartella home ogni mattina alle 10:00, garantendo che il backup venga effettuato anche se il PC è spento al momento previsto per il backup [1].

3.4.5 Riferimenti

- [1] https://wiki.archlinux.org/title/Systemd/Timers
- [2] https://linuxconfig.org/how-to-schedule-tasks-with-systemd-timers-in-linux
- [3] https://www.airplane.dev/blog/systemd-timer-how-to-schedule-tasks-with-systemd
- [4] https://man.archlinux.org/man/systemd.timer.5
- [5] https://opensource.com/article/20/7/systemd-timers
- [6] https://wiki.archlinux.org/title/Systemd
- [7] https://lloydrochester.com/post/unix/systemd-timer-example/
- [8] https://unix.stackexchange.com/questions/704109/configure-systemd-timer-to-run-every-hour-after-first-run
- [9] https://linuxman.co/linux-desktop/easily-automate-maintenance-in-arch-linux-with-the-power-of-systemd-timers/
- [10] https://www.fosslinux.com/48317/scheduling-tasks-systemd-timers-linux.htm
- [11] https://www.binaryte.com/blog/systemd-timers-tutorial-for-scheduling-tasks
- [12] https://www.learnlinux.tv/automate-your-tasks-with-systemd-timers-a-step-by-step-guide/
- [13] https://blog.devgenius.io/devops-in-linux-systemd-timer-4e95f57b6d71
- [14] https://dashdash.io/5/systemd.timer
- [15] https://techviewleo.com/configuring-cron-jobs-in-linux-using-systemd-timers/