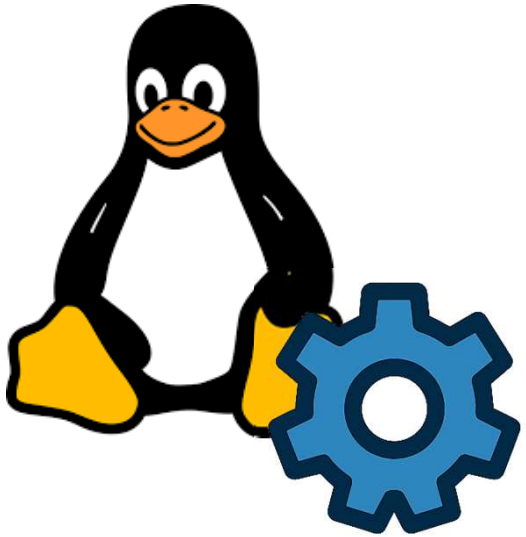
ZUCCHETTI
Academy

LINUX:
I processi e i servizi

Relazione a cura di:
Nicolas Centofanti – Cristina Tinti

Argomenti

- I processi di Linux
- Schedulazione processi
- Processo di inizializzazione



ZUCCHETTI
Academy

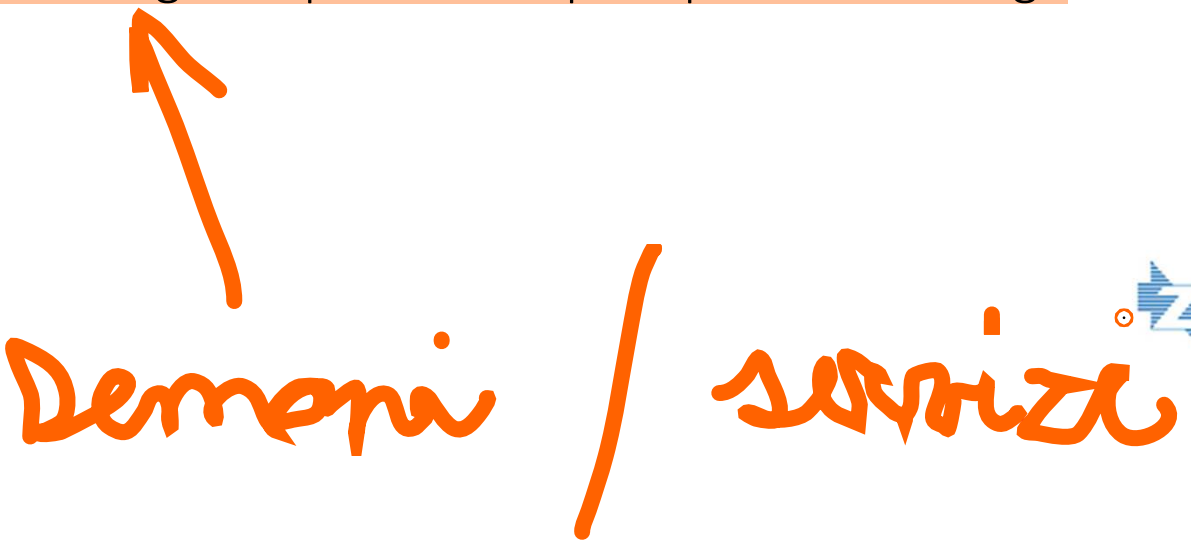
I processi di Linux

- In Linux, un **processo** è un'istanza in esecuzione di un programma.
- Ogni volta che avvii un comando, un'applicazione o uno script, il sistema crea un processo per gestirlo.
- I processi sono il cuore del multitasking in Linux: il sistema li gestisce, li pianifica e li monitora costantemente.
- Un processo rappresenta una risorsa fondamentale che il kernel gestisce e pianifica per l'esecuzione sulla CPU.
- Ogni processo ha un **ID di processo (PID)** univoco e uno **stato** che ne descrive l'attività corrente.

ZUCCHETTI Academy

I processi di Linux: tipi e stati

- I processi possono essere classificati in due categorie principali:
  - Processi foreground (o interattivi): Sono quelli che l'utente sta eseguendo attivamente e che interagiscono direttamente con un terminale.
  - Processi background (o in background): vengono eseguiti indipendentemente da un terminale e non richiedono l'interazione diretta dell'utente. Vengono spesso usati per operazioni a lungo termine.



ZUCCHETTI Academy

I processi di Linux: tipi e stati

- Gli stati del ciclo di vita di un processo sono:
  - In esecuzione (Running): Il processo sta usando la CPU o è pronto per farlo.
  - In attesa (Sleeping/Waiting): Il processo è in attesa che si verifichi un evento, come il completamento di un'operazione di input/output.
  - Terminato (Zombie): Il processo è stato terminato, ma la sua voce nella tabella dei processi non è ancora stata rimossa. Questo stato dura finché il processo padre non raccoglie il suo stato di uscita.
  - Interrotto (Stopped): Il processo è stato sospeso, solitamente da un segnale, come *SIGSTOP*.

I processi di Linux: relazione padre-figlio

- I processi in Linux sono organizzati in una **gerarchia genitore-figlio:**
  - Processo padre: È il processo che crea un nuovo processo.
  - Processo figlio: È il nuovo processo creato dal processo padre.
- **Ogni processo, ad eccezione di init (che ha PID 1), ha un processo padre.**
- Quando un processo padre termina, il kernel riassegna i processi figli rimasti al processo init, che diventa il loro nuovo genitore.
- Questo meccanismo previene la creazione di "orfani" nel sistema.

ZUCCHETTI Academy

I processi di Linux: relazione padre-figlio

- I processi in Linux sono organizzati in una **gerarchia genitore-figlio:**
  - Processo padre: È il processo che crea un nuovo processo.
  - Processo figlio: È il nuovo processo creato dal processo padre.
- **Ogni processo, ad eccezione di** init **(che ha** PID 1**), ha un processo padre**
- Quando un processo padre
  rimasti al processo init, ch
- Questo meccanismo previ

PID:
*è un **numero intero univoco** che il kernel assegna a ogni processo in esecuzione.*
In parole pratiche:
viene usato per distinguere un processo da un altro.

ZUCCHETTI Academy

I processi di Linux: attributi

- Gli attributi definisco le caratteristiche di un processo:
  - PID (Process ID).
  - PPID (Parent Process ID): PID del processo padre.
  - UID (User ID) e GID (Group ID)
  - Stato del processo: running, sleeping, zombie, stopped
  - Priorità: indica al kernel se eseguire prima o più spesso un processo
  - Tempo di CPU (CPU Time): tempo CPU usato dal processo
  - Memoria: memoria usata dal processo
  - Comando (CMD): comando che ha avviato il processo
  - Terminale (TTY): terminale associato al processo (se in background non ha tty)

I processi di Linux: ps

- Il comando ps sta per «process status» fornisce una vista statica dei processi in esecuzione.
- Viene usato per visualizzare un'istantanea dei processi in esecuzione sul sistema.

```
root@zaRPROXY:/tmp# ps -auxw
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.3 168072 12452 ?        Ss   lug23   1:09 /lib/systemd/systemd --system --deserialize=33
root           2  0.0  0.0      0     0 ?        S    lug23   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   lug23   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   lug23   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I<   lug23   0:00 [slub_flushwq]
root           6  0.0  0.0      0     0 ?        I<   lug23   0:00 [netns]
root           8  0.0  0.0      0     0 ?        I<   lug23   0:00 [kworker/0:0H-events_highpri]
root          10  0.0  0.0      0     0 ?        I<   lug23   0:00 [mm_percpu_wq]
root          11  0.0  0.0      0     0 ?        I    lug23   0:00 [rcu_tasks_kthread]
root          12  0.0  0.0      0     0 ?        I    lug23   0:00 [rcu_tasks_rude_kthread]
root          13  0.0  0.0      0     0 ?        I    lug23   0:00 [rcu_tasks_trace_kthread]
root          14  0.0  0.0      0     0 ?        S    lug23   0:21 [ksoftirqd/0]
root          15  0.0  0.0      0     0 ?        I    lug23  17:41 [rcu_preempt]
root          16  0.0  0.0      0     0 ?        S    lug23   0:12 [migration/0]
```

Academy

I processi di Linux: ps

- Di tutte le colonne dell'output del comando ps quelle che ci interessano sono:
  - USER
  - PID
  - %CPU: utilizzo, in percentuale, della CPU da parte del processo.
  - %MEM: utilizzo, in percentuale, della memoria fisica da parte del processo.
  - COMMAND: comando che ha avviato il processo.

ZUCCHETTI Academy

I processi di Linux: ps

- Esempi:

`ps -u nome_utente`

Visualizza i processi di un utente specifico.

`ps -aux`

Visualizza tutti i processi.

`ps -ef --forest`

Visualizza una gerarchia di processi ad albero.

Estratto di un output:

```
root      957861  413937  0 10:27 ?        00:00:00 \_ sshd: root@pts/1
root      957867  957861  0 10:27 pts/1    00:00:00 |   \_ -bash
root      957902  957867  0 10:30 pts/1    00:00:00 |       \_ more /etc/profile
root      958819  413937  0 11:26 ?        00:00:00 \_ sshd: root@pts/0
root      958825  958819  0 11:26 pts/0    00:00:00 |   \_ -bash
root      963161  958825  0 15:34 pts/0    00:00:00 |       \_ ps -ef --forest
```

ZUCCHETTI Academy

I processi di Linux: top

- Il comando top fornisce una **vista dinamica** in tempo reale dei processi in esecuzione sul sistema.
- Mostra un elenco aggiornato dei processi che consumano più risorse, in particolare CPU e memoria, e viene spesso utilizzato dagli amministratori di sistema per monitorare le prestazioni e identificare i processi che causano un carico elevato.

```
top - 14:58:28 up 44 days,  2:37,  3 users,  load average: 0,00, 0,00, 0,00
Tasks: 207 total,   1 running, 206 sleeping,   0 stopped,   0 zombie
%Cpu(s):100,0 us,  0,0 sy,  0,0 ni,  0,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   3915,2 total,    385,8 free,   1097,0 used,   2701,5 buff/cache
MiB Swap:    975,0 total,    975,0 free,      0,0 used.   2818,2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
      1 root      20   0  168072  12452   9132 S   0,0   0,3   1:09.89 systemd
      2 root      20   0       0      0      0 S   0,0   0,0   0:00.78 kthreadd
      3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
      5 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 slub_flushwq
      6 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 netns
      8 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworker/0:0H-events_highpri
     10 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 mm_percpu_wq
     11 root      20   0       0      0      0 I   0,0   0,0   0:00.00 rcu_tasks_kthread
     12 root      20   0       0      0      0 I   0,0   0,0   0:00.00 rcu_tasks_rude_kthread
     13 root      20   0       0      0      0 I   0,0   0,0   0:00.63 rcu_tasks_trace_kthread
     14 root      20   0       0      0      0 S   0,0   0,0   0:21.81 ksoftirqd/0
     15 root      20   0       0      0      0 I   0,0   0,0  17:41.60 rcu_preempt
     16 root      rt   0       0      0      0 S   0,0   0,0   0:12.66 migration/0
```

Ha molte opzioni per cambiare la visualizzazione.

ZUCCHETTI Academy

I processi di Linux: top

- La parte superiore dell'interfaccia di top dà una panoramica del sistema:
  - Prima riga: Mostra l'ora corrente, il tempo di attività del sistema (uptime), il numero di utenti connessi e il carico medio del sistema (load average) su 1, 5 e 15 minuti.

```
top - 15:04:02 up 44 days,  2:42,  3 users,  load average: 0,04, 0,01, 0,00
```

  - Seconda riga: Riassume il numero totale di processi e il numero di processi in ogni stato.

```
Tasks: 206 total,   1 running, 205 sleeping,   0 stopped,   0 zombie
```

ZUCCHETTI Academy

I processi di Linux: top

- Terza riga: visualizza l'utilizzo della CPU in percentuale, suddiviso per tipo di utilizzo (ad esempio, user, system, nice, idle, wait).

```
%Cpu(s):  1,1 us,  0,6 sy,  0,0 ni, 98,1 id,  0,0 wa,  0,0 hi,  0,2 si,  0,0 st
```

- Quarta e quinta riga: visualizzano l'utilizzo della memoria fisica e della memoria Swap, mostrando la memoria totale, libera, utilizzata e in cache.

```
MiB Mem :   3915,2 total,    372,2 free,   1110,3 used,   2701,9 buff/cache
MiB Swap:    975,0 total,    975,0 free,      0,0 used.   2804,9 avail Mem
```

ZUCCHETTI Academy

I processi di Linux: top

- Terza riga: visualizza l'utilizzo della CPU in percentuale, suddiviso per tipo di utilizzo (ad esempio, user, system, nice, idle, wait).

```
%Cpu(s):  1,1 us,  0,6 sy,  0,0 ni, 98,1 id,  0,0 wa,  0,0 hi,  0,2 si,  0,0 st
```

- Quarta e quinta riga: visualizzano l'utilizzo della memoria fisica e della memoria Swap, mostrando la memoria totale, libera, utilizzata e in cache.

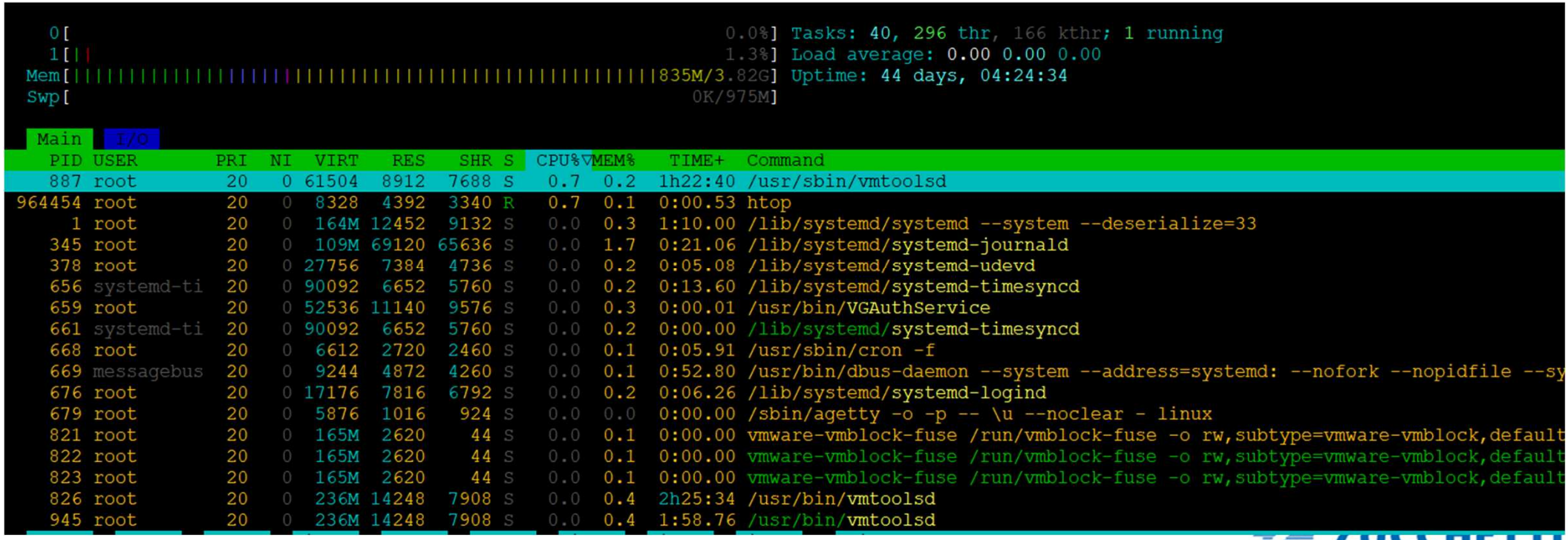
```
MiB Mem :                                                    /cache
MiB Swap:                                                    l Mem
```

Memoria Swap:
è una parte del disco rigido (HDD o SSD) che il sistema operativo Linux utilizza come estensione della memoria RAM fisica. Può essere implementata come una partizione dedicata o come un file di swap all'interno di un filesystem esistente.

ZUCCHETTI Academy

I processi di Linux: htop

- Dai repository è possibile installare un'evoluzione del comando top chiamata htop.



Academy

I processi di Linux: i segnali

- I segnali in Linux sono un metodo di comunicazione asincrona tra i processi o tra il kernel e i processi.
- I processi possono reagire in modo standard ai segnali, oppure essere internamente programmati per reagire in modo diverso (handler del segnale)

I processi di Linux: i segnali

- I segnali più comuni sono:
  - SIGTERM (15): Segnale di **terminazione «gentile»**.
    Chiede al processo di chiudersi in modo pulito.
    Il processo può scegliere di ignorare questo segnale, ma solitamente lo usa per salvare lo stato prima di terminare.

  - SIGKILL (9): Segnale di **terminazione «forzato»**.
    Non può essere intercettato o ignorato dal processo.
    Il kernel termina il processo immediatamente, senza dargli la possibilità di eseguire operazioni di pulizia.
    È l'ultima risorsa per terminare un processo che non risponde.

I processi di Linux: i segnali

- SIGHUP (1): segnale di **«hang up»**.
  Originariamente usato per indicare la disconnessione di un terminale, oggi è spesso usato per chiedere a un processo (in particolare a un demone o un servizio) di ricaricare la sua configurazione senza terminare.

- SIGINT (2): segnale di **interruzione**.
  Viene generato quando l'utente preme Ctrl+C in un terminale.
  È un segnale «amichevole» che chiede al processo di terminare.

I processi di Linux: i segnali

- Sintassi

```
kill [opzioni] <PID>
```

- Esempi:

  `kill 1234`

  Equivalente a kill –SIGTERM 1234 o kill -15 1234.

  Termina il processo con PID 1234 in modo «gentile»

  `kill -9 1234`

  Termina il processo con PID 1234 in modo «forzato»

Un utente normale può «killare» solo i suoi processi.
L'utente root può «killare» i processi di tutti gli utenti.

ZUCCHETTI Academy

I processi di Linux: priorità

- nice e renice sono due comandi che permettono di controllare la **priorità di esecuzione** dei processi.
- Il termine **«nice value»** si riferisce a un parametro che influenza la priorità di un processo assegnata dal kernel Linux.
  - Il valore è compreso tra **-20** (priorità più alta) e **+19** (priorità più bassa).
  - Un valore di **0** è la priorità predefinita.
  - Un valore più basso (più vicino a -20) significa che il processo è meno «gentile» e richiede più tempo di CPU.
  - Un valore più alto (più vicino a +19) significa che il processo è più «gentile» e lascia la CPU ad altri processi.

I processi di Linux: priorità

- Il comando nice consente di **avviare un programma** con priorità di esecuzione modificata.
- Il comando renice consente di **modificare la priorità di processi in esecuzione**, intervenendo su un singolo processo, o su tutti i processi di un utente o su tutti i processi di un gruppo (usando EUID eUGID)

- Sintassi:

```
nice –n [valore] [comando]

renice [valore] [opzioni] [ID_processo]
```

ZUCCHETTI Academy

I processi e la rete: netstat

- netstat mostra le **connessioni di rete** (in entrata e in uscita), le tabelle di routing, le statistiche dell'interfaccia, le connessioni multicast e i socket aperti.
- Principali utilizzi:
  - Visualizzare le connessioni attive: quali porte sono in ascolto sul sistema e quali connessioni sono state stabilite.
  - Diagnosi di problemi: aiuta a capire perché una connessione non funziona o a identificare servizi che non sono in ascolto sulla porta corretta
  - Sicurezza: permette di individuare connessioni sospette da e verso il server.
  - Statistiche di rete

ZUCCHETTI Academy

I processi e la rete: netstat

- Opzioni comuni del comando netstat:
- Principali utilizzi:
  - -p (program) PID e nome programma
  - -l (listening) solo i socket in ascolto
  - -a (all)
  - -n (numeric)
  - -t (tcp)

```
root@zaRPROXY:/tmp# netstat -antpl
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:40343         0.0.0.0:*               LISTEN      413602/arcproxy
tcp        0      0 127.0.0.1:40342         0.0.0.0:*               LISTEN      413569/himds
tcp        0      0 127.0.0.1:40341         0.0.0.0:*               LISTEN      413569/himds
tcp        0      0 127.0.0.1:40344         0.0.0.0:*               LISTEN      413602/arcproxy
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      413937/sshd: /usr/s
tcp        1      0 172.19.2.99:60210       172.26.22.9:8004        CLOSE_WAIT  962652/apache2
tcp        0      0 172.19.2.99:56248       172.26.22.230:8021      ESTABLISHED 962652/apache2
tcp        1      0 172.19.2.99:58694       172.26.22.9:8004        CLOSE_WAIT  962653/apache2
tcp        0      0 172.19.2.99:46834       172.26.22.230:8016      ESTABLISHED 962652/apache2
tcp        1      0 172.19.2.99:58682       172.26.22.9:8004        CLOSE_WAIT  962653/apache2
tcp        1      0 172.19.2.99:38358       172.26.22.230:8024      CLOSE_WAIT  962652/apache2
tcp        1      0 172.19.2.99:45302       172.26.22.214:8009      CLOSE_WAIT  962652/apache2
tcp        0      0 172.19.2.99:33202       172.26.22.230:8016      ESTABLISHED 962653/apache2
```

I processi e la rete: ntop

- ntop (non standard) applicazione open source progettata per il monitoraggio del traffico di rete.
- Il suo nome deriva da «network top», indicando la sua funzione di «top» per la rete.
- Analizza il traffico di rete e offre una visione dettagliata in tempo reale dei flussi di dati. Le sue funzioni principali includono:
  - Monitoraggio in tempo reale: quali host stanno usando la rete e il traffico.
  - Analisi dei protocolli: protocolli di rete in uso.
  - Visualizzazione dei flussi: connessioni attive tra gli host.
  - Statistiche sull'uso della banda

ZUCCHETTI Academy

Schedulazione processi: at e cron

- at e cron sono entrambi utilizzati per pianificare l'esecuzione di comandi o script in un momento futuro.
- La differenza principale tra i due è la loro finalità:
  - at viene usato per eseguire un comando o uno script una sola volta in un momento specifico nel futuro.
    È l'ideale per compiti che devono essere eseguiti solo una volta e non in modo ripetitivo.
  - cron il demone (daemon) di pianificazione che esegue comandi e script a intervalli regolari.
    È l'ideale per compiti che devono essere eseguiti periodicamente, come backup giornalieri, aggiornamenti settimanali o pulizia dei file di log mensile.

ZUCCHETTI Academy

Schedulazione processi: at e cron

- I permessi per usare i comandi at e cron sono impostati dai file:
  - /etc/at.allow : utenti abilitati a usare at
  - /etc/at.deny : utenti espressamente negati (se vuoto, sono tutti abilitati)
  - /etc/cron.allow : utenti abilitati a usare cron
  - /etc/cron.deny : utenti espressamente negati (se vuoto, sono tutti abilitati)

Schedulazione processi: at

- La sintassi del comando at è:

  ```
  at [opzioni] TEMPO
  ```

- Dopo aver digitato at e aver specificato l'orario, l'utente viene portato a un prompt speciale (at>) dove può inserire il comando da eseguire.
- Per terminare l'inserimento e salvare il job, si preme Ctrl+D.
- Altri comandi:
  - atq: visualizza la coda dei job pianificati.
  - atrm: rimuove un job specifico dalla coda.

ZUCCHETTI Academy

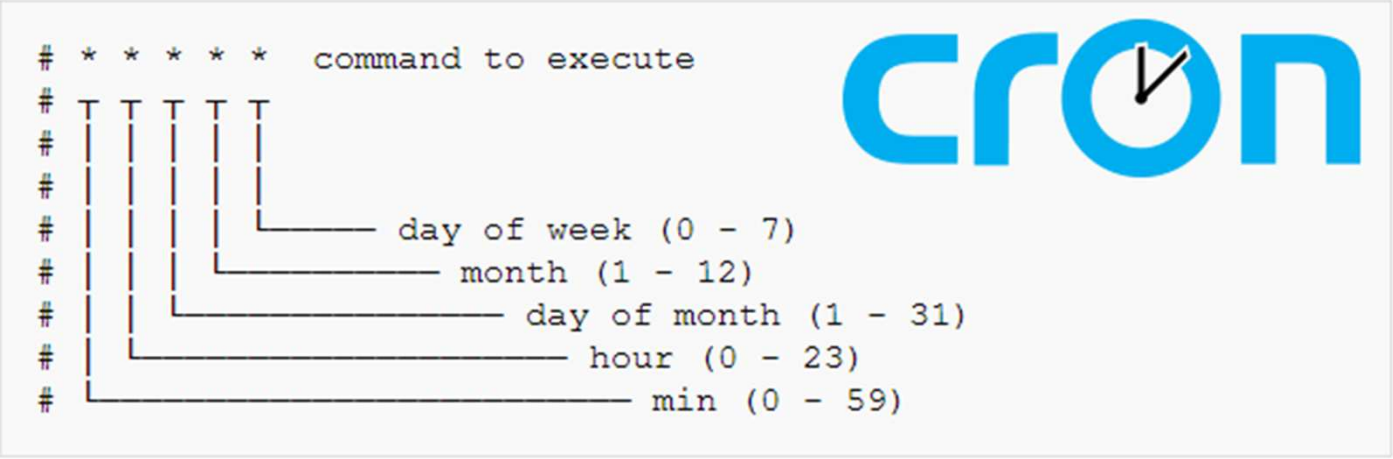
Schedulazione processi: at

- Esempi:
  - Orario specifico: `at 14:30`
    esegue alle 14:30 di oggi o domani se l'orario è già passato.
  - Orario e data: `at 14:30 2025-12-25`
    esegue alle 14:30 del 25 dicembre 2025
  - Relativo: `at now + 5 minutes`
    esegue tra 5 minuti
    `at now + 1 day`
    esegue tra un giorno
  - Parole chiave: `at noon`, `at midnight`, `at tomorrow`.

ZUCCHETTI Academy

Schedulazione processi: cron

- I comandi da eseguire e i relativi orari di esecuzione vengono salvati in un file chiamato **crontab** (dall'unione di "cron" e "table").
- Ogni riga di questo file rappresenta un'attività e segue una sintassi precisa composta da **cinque campi temporali** seguiti dal comando da eseguire.



ZUCCHETTI Academy

Schedulazione processi: cron

- Per gestire i cron jobs del proprio utente si usa il comando crontab con varie opzioni:
- crontab -e: Apre il file crontab dell'utente per l'editing. È il modo più comune per aggiungere, modificare o rimuovere job.
- crontab -l: Mostra tutti i cron jobs attivi per l'utente corrente.
- crontab -r: Rimuove il crontab dell'utente corrente, cancellando tutti i job pianificati.

*È importante notare che **ogni utente ha il proprio crontab**. Inoltre, esiste un crontab di sistema (solitamente in /etc/crontab) che può essere gestito solo da root.*

ZUCCHETTI Academy

Schedulazione processi: anacron

- anacron è un programma che serve a eseguire i job di cron che non sono stati eseguiti a causa dello spegnimento del computer.
- A differenza di cron, che presuppone che il sistema sia sempre acceso, anacron è pensato specificamente per computer che non sono in funzione 24/7, come i laptop o i desktop di casa.
- Controlla, a intervalli regolari, se i job pianificati siano stati saltati. Se un job non è stato eseguito, lo lancia non appena il sistema si riaccende.

Schedulazione processi: anacron

- Utilizza un file di configurazione (/etc/anacrontab) che contiene la lista dei job da eseguire e la loro frequenza.
- Sintassi del file:

```
periodo_in_giorni   ritardo_in_minuti
identificativo_job   comando
```

ZUCCHETTI Academy

Inizializzazione di Linux: il processo di init

- Il processo init ha diverse funzioni critiche:
  - Avvio dei processi: quando viene creato un nuovo processo, questo è un figlio di init o un suo discendente. Questo crea un albero dei processi con init alla radice.
  - Gestione dei demoni: è responsabile dell'avvio dei servizi di sistema e dei demoni (processi in background).
  - Adozione degli orfani: se un processo padre termina prima del suo figlio, il processo figlio «orfano» viene automaticamente adottato da init garantendo che nessun processo diventi uno «zombie» consumando risorse di sistema senza fare nulla.

Inizializzazione di Linux: il boot loader

- Il **boot loader** in Linux è un piccolo programma che viene eseguito dal BIOS (o UEFI) del computer dopo il Power-On Self-Test (POST).
- È **il primo software a essere eseguito** all'accensione del PC, a meno che non ci sia un hardware malfunzionante che interrompa il processo di avvio.
- Il suo scopo principale è quello di **individuare, caricare e avviare il kernel** del sistema operativo nella memoria RAM del computer.

ZUCCHETTI Academy

Inizializzazione di Linux: il boot loader

- Il **boot loader** :
  - riceve il controllo dal BIOS/UEFI (**si interfaccia con il firmware**).
  - **localizza il kernel** del sistema operativo sul disco rigido, che può trovarsi in diverse posizioni a seconda della configurazione.
  - **carica l'immagine del kernel** e l'initramfs (un disco virtuale iniziale) nella memoria.
  - **avvia il kernel**, passandogli i parametri necessari per il corretto funzionamento.

ZUCCHETTI Academy

Inizializzazione di Linux: LILO

- LILO (LInux LOader) è stato uno dei primi e più diffusi boot loader per i sistemi Linux.
- Fa riferimento al dile /etc/lilo.conf
- Per molti anni, è stato il modo standard per avviare il sistema operativo.
- Oggi, pur essendo ancora disponibile, è considerato un'opzione «legacy»: è un pezzo di storia del software Linux, affidabile ma meno flessibile e più esigente in termini di manutenzione rispetto alle soluzioni moderne.

ZUCCHETTI Academy

Inizializzazione di Linux: LILO

- **LILO** (LInux LOader) è stato uno dei primi e più diffusi boot loader per i sistemi Linux.
- Per molti anni, è stato il modo standard per avviare il sistema operativo.
- Oggi, pur essendo ancora disponibile, è considerato un'**opzione «legacy»**: è un pezzo di storia [illegible] esigente in termini

Opzione «legacy»:
si riferisce a una scelta o modalità di funzionamento che permette a un sistema moderno di supportare tecnologie, standard o hardware obsoleti.
*Il termine «legacy» significa letteralmente «**eredità**» e, in informatica, indica ciò che è stato ereditato dal passato.*

ZUCCHETTI Academy

Inizializzazione di Linux: LILO

- Caratteristiche:
  - **Mappatura diretta**: LILO non «conosce» i filesystem, si basa su un file di mappa (/boot/map) che contiene la posizione fisica, settore per settore, del kernel sul disco.
  - **Aggiornamento manuale**: ogni volta che si aggiorna il kernel o si modifica il file di configurazione (/etc/lilo.conf), è necessario eseguire manualmente il comando `lilo` per aggiornare il file di mappa. Dimenticare questo passaggio può rendere il sistema non avviabile al successivo riavvio.
  - **Installazione**: può essere installato nel **Master Boot Record** (MBR) del disco o nel settore di avvio di una singola partizione.

ZUCCHETTI Academy

Inizializzazione di Linux: GRUB

- Il **boot loader** più diffuso nelle moderne distribuzioni Linux è **GRUB** (**GRand Unified Bootloader**).
- Fa riferimento al file /boot/grub/grub.conf
- Il processo di avvio di GRUB si articola in diverse fasi, o «stage», che lavorano insieme per caricare il kernel e avviare il sistema.
- È noto per la sua versatilità.

ZUCCHETTI Academy

Inizializzazione di Linux: GRUB

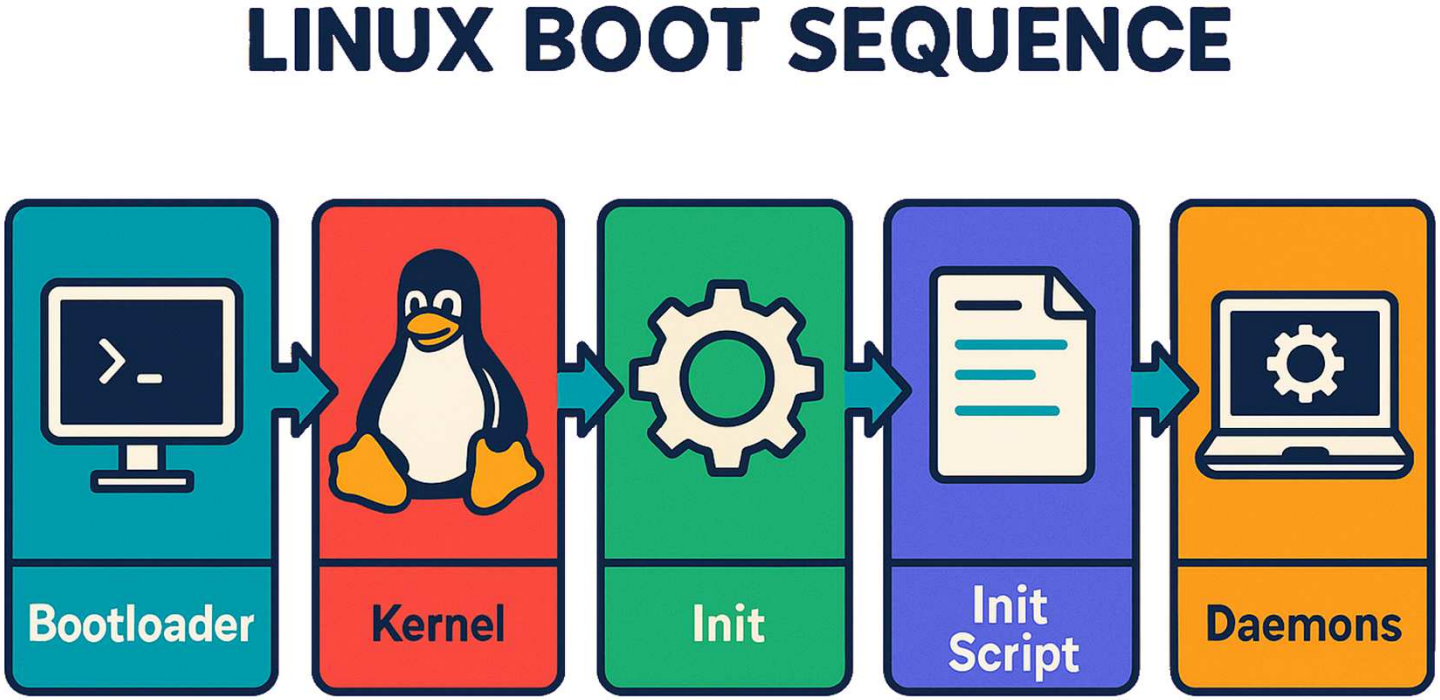
- Funzionalità avanzate:
  - **Supporto multi-sistema**: GRUB è in grado di avviare diversi sistemi operativi (Linux, Windows e macOS) installati sullo stesso computer.
  - **Configurazione dinamica**: permette di modificare i parametri di avvio (come i parametri del kernel) direttamente da un menu prima di avviare il sistema.
  - **Riconoscimento dei filesystem**: a differenza dei boot loader più vecchi, GRUB può leggere direttamente i file di configurazione e le immagini del kernel da vari filesystem, come ext4, NTFS e FAT32.

ZUCCHETTI Academy

Inizializzazione di Linux: LILO vs GRUB

- L'avvento di GRUB ha segnato il **declino di LILO** per diversi motivi:
  - **Flessibilità**: GRUB è più flessibile.
  - **Aggiornamento automatico**: GRUB si integra meglio con i moderni sistemi di gestione dei pacchetti. Le distribuzioni Linux sono in grado di aggiornare automaticamente la sua configurazione all'installazione di un nuovo kernel, eliminando il rischio di problemi di avvio.
  - **Supporto UEFI/GPT**: LILO è progettato per i sistemi BIOS e per la tabella di partizionamento MBR. GRUB, invece, supporta pienamente i sistemi UEFI e le tabelle di partizionamento GPT, che sono lo standard sui computer più recenti.
  - **Interfaccia utente**: GRUB offre un'interfaccia a riga di comando più potente e un menu di avvio più versatile.

Inizializzazione di Linux: la sequenza di boot



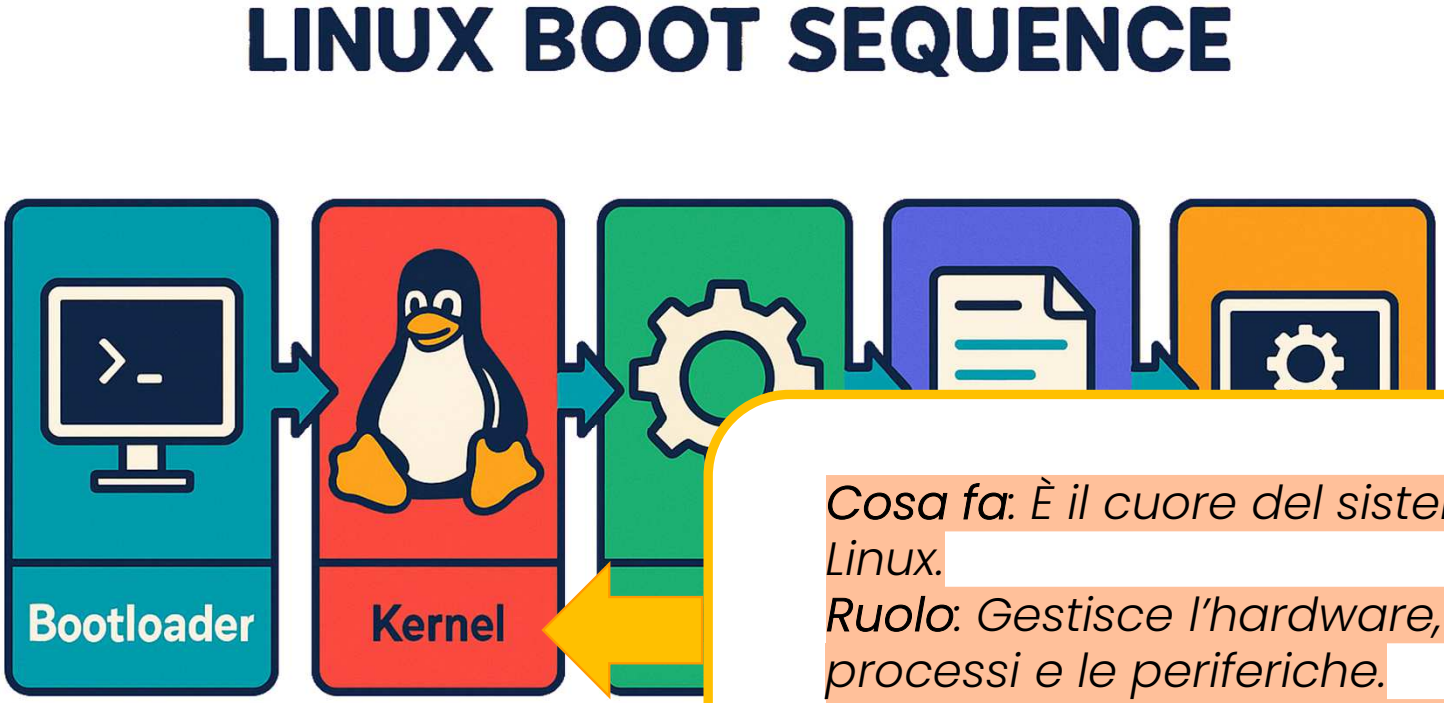
ZUCCHETTI Academy

Inizializzazione di Linux: la sequenza di boot

Cosa fa: *È il primo software che viene eseguito all'accensione del computer.*
Ruolo: *Carica il kernel in memoria e avvia il sistema operativo.*
Curiosità: *Puoi scegliere tra più sistemi operativi se ne hai installati diversi.*

ZUCCHETTI Academy

Inizializzazione di Linux: la sequenza di boot



Cosa fa: *È il cuore del sistema operativo Linux.*
Ruolo: *Gestisce l'hardware, la memoria, i processi e le periferiche.*
Curiosità: *Dopo il bootloader, il kernel prende il controllo totale.*

Academy

Inizializzazione di Linux: la sequenza di boot

Cosa fa: *È il primo processo utente avviato dal kernel (PID 1).*
Ruolo: *Coordina l'avvio dei servizi e delle configurazioni iniziali.*
Curiosità: *Se Init fallisce, il sistema non può continuare il boot.*

ZUCCHETTI Academy

Inizializzazione di Linux: la sequenza di boot

Cosa fanno: *Sono script che definiscono quali servizi e demoni avviare.*
Ruolo: *Configurano rete, interfaccia grafica, logging, ecc.*
Curiosità: *Puoi modificarli per avviare o disattivare servizi.*

ZUCCHETTI Academy

Inizializzazione di Linux: la sequenza di boot

Cosa fanno: *Sono processi in background che gestiscono funzioni continue.*
Ruolo: *Mantengono il sistema operativo e reattivo.*
Curiosità: *Il nome deriva dal greco «δαίμων», spiriti invisibili che eseguono compiti..*

ZUCCHETTI Academy

Inizializzazione di Linux: evoluzione di init

- Il concetto di «init» è rimasto costante, ma la sua implementazione è cambiata nel tempo:
  - SysVinit: era il sistema di init tradizionale, in stile Unix. Usava una serie di script organizzati in «runlevel» per decidere quali servizi avviare o terminare.
    Era affidabile, ma lento perché avviava i servizi in modo sequenziale.

ZUCCHETTI Academy

Inizializzazione di Linux: evoluzione di init

- Il concetto di «init» è rimasto costante, ma la sua implementazione è cambiata nel tempo:
  - **SysVinit**: era il sistema di init tradizionale, in stile Unix. Usava una serie di script organizzati in «**runlevel**» per decidere quali servizi avviare o

Runlevel 0: *Arresta il sistema (halt).*

Runlevel 1: *Modalità utente singolo. Per manutenzione e riparazioni, non avvia i servizi di rete e permette l'accesso solo all'amministratore (root).*

Runlevel 2: *Modalità multi-utente con servizi di base. Predefinito su sistemi senza GUI.*

Runlevel 3: *Modalità multi-utente completa con rete. Predefinito per i server.*

Runlevel 4: *Non utilizzato di default, è personalizzabile dall'utente.*

Runlevel 5: *Modalità multi-utente completa. Predefinito per i desktop Linux.*

Runlevel 6: *Riavvia il sistema (reboot).*

Inizializzazione di Linux: evoluzione di init

- Systemd: è l'attuale standard per la maggior parte delle principali distribuzioni Linux (come Debian, Fedora e Ubuntu).
  Systemd è un gestore di sistema e di servizi molto più completo e veloce, gestisce:
  - l'avvio parallelo dei servizi;
  - il logging;
  - altre attività di gestione del sistema;
  - usa le cosiddette «unit» per descrivere come devono essere gestiti i servizi, i dispositivi e altre risorse.

ZUCCHETTI Academy

Inizializzazione di Linux: /etc/inittab

- Il file /etc/inittab è stato storicamente fondamentale nel sistema di inizializzazione SysVinit di Linux.
- Anche se oggi è spesso sostituito da sistemi più moderni come systemd.
- È un file di configurazione che:
  - Definisce il runlevel di default (cioè lo stato operativo del sistema).
  - Specifica quali processi avviare, monitorare e riavviare.
  - Determina le azioni da eseguire quando il sistema cambia runlevel.

ZUCCHETTI Academy

Inizializzazione di Linux: /etc/inittab

- Ogni riga ha quattro campi separati da due punti:

`id:runlevels:action:process`

  - id: Identificatore univoco (1-4 caratteri).
  - runlevels: Livelli a cui si applica (es. 3 per multiutente con rete).
  - action: Cosa fare (es. respawn, wait, initdefault).
  - process: Comando o script da eseguire.

ZUCCHETTI Academy

Inizializzazione di Linux: SysVinit vs systemd

Aspetto	SysVinit	systemd
Anno di introduzione	Anni '80	2010
Filosofia	Semplice, modulare, basato su script	Integrato, parallelo, orientato ai servizi
Tipo di avvio	Sequenziale	Parallelo e basato su dipendenze

Funzione	SysVinit	systemd
Script di avvio	`/etc/init.d/`	Unit file in `/etc/systemd/system/`
Comando di gestione	`service`, `chkconfig`	`systemctl`
Monitoraggio dei servizi	Limitato	Avanzato, con logging e controllo dei processi
Riavvio automatico	Manuale	Integrato (`Restart=always`)

Inizializzazione di Linux: /etc/init.d

- La directory /etc/init.d è stata storicamente il cuore del sistema di avvio SysVinit su molte distribuzioni Linux.
- Contiene gli script di inizializzazione che gestiscono l'avvio, l'arresto e il controllo dei servizi di sistema.
- È una raccolta di script Bash che rappresentano i servizi del sistema (es. sshd, network, cron, httpd).
- Ogni script può essere eseguito manualmente con comandi come:

```
/etc/init.d/sshd start
/etc/init.d/network restart
```

ZUCCHETTI Academy

Inizializzazione di Linux: /etc/init.d

- I servizi vengono avviati o arrestati in base al runlevel del sistema.

- I link simbolici a questi script si trovano in /etc/rc[0-6].d/, dove ogni directory rappresenta un runlevel.

- Esempio:

  `/etc/rc3.d/S20sshd`

  avvia sshd nel runlevel 3.

ZUCCHETTI Academy

Inizializzazione di Linux: /etc/init.d

```
#!/bin/bash
# chkconfig: 2345 20 80
# description: Avvia il servizio SSH

start() {
  echo "Avvio SSH..."
  /usr/sbin/sshd
}

stop() {
  echo "Arresto SSH..."
  killall sshd
}

case "$1" in
  start) start ;;
  stop) stop ;;
  restart) stop; start ;;
  *) echo "Uso: $0 {start|stop|restart}" ;;
esac
exit 0
```

- Il campo chkconfig indica i runlevel e le priorità di avvio/arresto.
- I comandi start, stop, restart sono gestiti tramite case.

Script bash:
*Le righe commentate (e quindi non prese in considerazione durante l'esecuzione dello script) iniziano con il carattere «**#**»*
Fatta eccezione di:
#!/bin/bash
Indica la shell con cui deve essere interpretato lo script.

Inizializzazione di Linux: /etc/init.d

```
#!/bin/bash
# chkconfig: 2345 20 80
# description: Avvia il servizio SSH

start() {
  echo "Avvio SS
  /usr/sbin/sshd
}

stop() {
  echo "Arresto
  killall sshd
}

case "$1" in
  start) start ;
  stop) stop ;;
  restart) stop;
  *) echo "Uso: $
esac
exit 0
```

- Il campo chkconfig indica i runlevel e le priorità di avvio/arresto.

chkconfig:

è uno strumento classico usato per gestire i servizi e il loro comportamento nei diversi runlevel.

```
chkconfig --list                    # Elenca tutti i servizi e i loro runlevel
chkconfig servizio --list           # Stato di un singolo servizio
chkconfig servizio on               # Abilita il servizio all'avvio
chkconfig servizio off              # Disabilita il servizio all'avvio
chkconfig --add servizio            # Aggiunge un servizio alla gestione
chkconfig --del servizio            # Rimuove un servizio dalla gestione
chkconfig --level 3 servizio on     # Abilita il servizio solo nel runlevel 3
```

Inizializzazione di Linux: /etc/systemd/system

- La directory /etc/init.d con l'arrivo di systemd è stata in gran parte deprecata.
- I servizi sono ora gestiti tramite unit file in /etc/systemd/system/.
- I comandi sono sostituiti da systemctl.
- Esempi:

```
systemctl status sshd
systemctl start sshd
systemctl stop sshd
```

ZUCCHETTI Academy

Inizializzazione di Linux: /etc/systemd/system

```
[Unit]
Description=OpenSSH server daemon
After=network.target auditd.service
ConditionPathExists=/etc/ssh/sshd_config

[Service]
Type=notify
ExecStart=/usr/sbin/sshd -D
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s

[Install]
WantedBy=multi-user.target
```

- After: indica che sshd deve partire dopo la rete e il demone audit.
- ConditionPathExists: controlla che il file di configurazione esista.
- ExecStart: comando per avviare il demone SSH.
- ExecReload: comando per ricaricare la configurazione.
- Restart: riavvia il servizio in caso di errore.
- WantedBy: specifica il target di avvio (multi-user è il runlevel 3 moderno).

ZUCCHETTI Academy

Inizializzazione di Linux: le «unit» di systemd

- Principali tipi di «unit»:

Tipo di unità	Estensione	Descrizione
Servizio	`.service`	Avvia e gestisce demoni (es. `sshd.service`)
Socket	`.socket`	Attiva servizi al momento dell'accesso al socket
Target	`.target`	Raggruppa unità per stati del sistema
Mount	`.mount`	Gestisce punti di montaggio
Timer	`.timer`	Avvia unità in base al tempo
Path	`.path`	Attiva unità al cambiamento di file o directory
Device	`.device`	Rappresenta dispositivi hardware
Automount	`.automount`	Montaggio automatico al momento dell'accesso
Swap	`.swap`	Gestisce partizioni di swap
Slice	`.slice`	Gruppi di risorse per il controllo tramite cgroups
Scope	`.scope`	Processi esterni non avviati da systemd

ZUCCHETTI Academy

Inizializzazione di Linux: le «unit» di systemd

- Le «unit» si possono trovare nei seguenti percorsi:
  - /etc/systemd/system/
    «*unit» personalizzate dall'amministratore*
  - /lib/systemd/system/
    «*unit» fornite dal sistema*
  - ~/.config/systemd/user/
    «*unit» per sessioni utente*

ZUCCHETTI Academy

ZUCCHETTI
Academy

Il software che crea successo