Linux

Comandi utilizzati

Watch -> https://www.man7.org/linux/man-pages/man1/watch.1.html

Interrupt

Segnale che invia alla CPU la richiesta di interrompere la normale elaborazione per richiedere l'immediata attenzione della CPU.
Immagina una situazione in cui una periferica, come una tastiera o un mouse, deve comunicare con il sistema. Queste periferiche possono generare interrupt per notificare alla CPU che è necessario intraprendere un'azione immediata.
Ad esempio nella maggior parte dei computer moderni, l'input dalla tastiera è gestito attraverso interrupt hardware generati quando si premono i tasti.

Su Linux tutto è un file

Su Linux, il concetto fondamentale è che "tutto è un file". Questo significa che sia i dati che i dispositivi hardware sono rappresentati come file nel sistema operativo. Ad esempio, i file tradizionali su disco, le cartelle, i dispositivi come dischi rigidi e stampanti, così come le comunicazioni con le periferiche hardware, sono tutti trattati come file. Questo approccio semplifica notevolmente l'interazione con il sistema, poiché puoi utilizzare operazioni di lettura/scrittura standard per accedere e gestire diverse risorse.
L'alberatura delle cartelle di sistema su Linux è organizzata in modo gerarchico, seguendo il modello del "File System Hierarchy Standard" (FHS). Le cartelle di sistema sono organizzate in una struttura ad albero che parte dalla radice del sistema ("/") e si estende in sotto-cartelle.

La directory /proc

Una delle directory di sistema è **/proc ,** che contiene informazioni riguardanti i processi di sistema.

Per esempio se digitiamo **cat /proc/interrupts** potremmo vedere stampato sul terminale il file relativo agli interrupts. Il file è strutturato in colonne :

- Una colonna per ogni core della cpu
- La colonna "IRQ line" elenca il numero dell'interrupt associato a un dispositivo hardware specifico.
- La colonna "Interrupt type" indica il tipo di interrupt, che può essere "edge" (bordo) o "fasteoi" (first available).

```
           CPU0       CPU1       CPU2       CPU3
  1:          0          0          0       5502  IR-IO-APIC    1-edge      i8042
  8:          0          0          0          0  IR-IO-APIC    8-edge      rtc0
  9:          0     100000          0          0  IR-IO-APIC    9-fasteoi   acpi
 14:          0          0          0          0  IR-IO-APIC   14-fasteoi   INT3455:00
 16:          0          0          0          0  IR-IO-APIC   16-fasteoi   i801_smbus, idma64.0, i2c_designware.0
 17:     156430    1387392          0          0  IR-IO-APIC   17-fasteoi   idma64.1, i2c_designware.1
 48:          0     200133          0          0  IR-IO-APIC   48-fasteoi   MSFT0001:01
120:          0          0          0          0  DMAR-MSI    0-edge      dmar0
121:          0          0          0          0  DMAR-MSI    1-edge      dmar1
122:          0          0          0          0  IR-PCI-MSI-0000:00:1d.0    0-edge      PCIe PME, aerdrv, pcie-dpc
123:          0          0          0          0  IR-PCI-MSI-0000:00:0d.0    0-edge      xhci_hcd
124:          0          0       6794          0  IR-PCI-MSI-0000:00:14.0    0-edge      xhci_hcd
125:          0          0          0          0  IR-PCI-MSI-0000:00:17.0    0-edge      ahci[0000:00:17.0]
126:         15          0          0          0  IR-PCI-MSIX-0000:01:00.0    0-edge      nvme0q0
127:      25047          0          0          0  IR-PCI-MSIX-0000:01:00.0    1-edge      nvme0q1
128:          0      16294          0          0  IR-PCI-MSIX-0000:01:00.0    2-edge      nvme0q2
129:          0          0      23647          0  IR-PCI-MSIX-0000:01:00.0    3-edge      nvme0q3
130:          0          0          0      22961  IR-PCI-MSIX-0000:01:00.0    4-edge      nvme0q4
131:          0          0         96          0  IR-PCI-MSI-0000:00:16.0    0-edge      mei_me
132:       6701       3978       1514      15051  IR-PCI-MSIX-0000:00:14.3    0-edge      iwlwifi:default_queue
133:       1262       1125       1162       2350  IR-PCI-MSIX-0000:00:14.3    1-edge      iwlwifi:queue_1
134:       1921       1948        396       3278  IR-PCI-MSIX-0000:00:14.3    2-edge      iwlwifi:queue_2
135:       4426       1842        452        903  IR-PCI-MSIX-0000:00:14.3    3-edge      iwlwifi:queue_3
136:       2364       1689        182       2458  IR-PCI-MSIX-0000:00:14.3    4-edge      iwlwifi:queue_4
137:          3          0          0          0  IR-PCI-MSIX-0000:00:14.3    5-edge      iwlwifi:exception
138:          0      90202     511384          0  IR-PCI-MSI-0000:00:02.0    0-edge      i915
```

Se invece usiamo il comando **watch -n 1 cat /proc/interrupts** possiamo visualizzare il file aggiornato ogni secondo e digitando dentro ad un file di testo possiamo visualizzare come il numero di interrupts inviati dalla periferica della tastiera aumenti.

Invece il file **/proc/ioports** mostra gli indirizzi di memoria assegnati alle porte I/O e alle periferiche.

```
0000-0cf7 : PCI Bus 0000:00
  0000-001f : dma1
  0020-0021 : pic1
  0040-0043 : timer0
  0050-0053 : timer1
  0060-0060 : keyboard
  0062-0062 : PNP0C09:00
    0062-0062 : EC data
  0064-0064 : keyboard
  0066-0066 : PNP0C09:00
    0066-0066 : EC cmd
  0070-0071 : rtc_cmos
    0070-0071 : rtc0
  0080-008f : dma page reg
  00a0-00a1 : pic2
  00c0-00df : dma2
  00f0-00ff : fpu
  0400-041f : iTCO_wdt
```

Riassumendo , la CPU comunica con le periferiche grazie ai bus di sistema e quando una periferica viene inserita o esegue un azione (come la digitazione della tastiera) invia un interrupt alla CPU cosi che questa si occupi di elaborare il segnale e ,comunicando con il kernel ,restituisca il risultato (come stampare a video la lettera che abbiamo premuto sulla tastiera).Quindi dentro alla directory /proc l'amministratore di sistema può consultare tutti i dati riguardanti i processi.

La directory /dev

La directory **/dev** invece contiene file speciali che rappresentano le periferiche del sistema o pseudo dispositivi .

```
lanza@lenovo-lanza:~$ ls /dev
acpi_thermal_rel  drm_dp_aux0  i2c-1   initctl  loop6         ng0n1       rtc
autofs            drm_dp_aux1  i2c-10  input    loop7         null        rtc0
block             drm_dp_aux2  i2c-11  kmsg     loop8         nvme0       sgx_enclave
btrfs-control     ecryptfs     i2c-12  kvm      loop9         nvme0n1     sgx_provision
bus               fb0          i2c-13  log      loop-control  nvme0n1p1   sgx_vepc
char              fd           i2c-14  loop0    mapper        nvme0n1p2   shm
console           full         i2c-2   loop1    mcelog        nvram       snapshot
core              fuse         i2c-3   loop10   media0        port        snd
cpu               gpiochip0    i2c-4   loop11   mei0          ppp         stderr
cpu_dma_latency   hidraw0      i2c-5   loop12   mem           psaux       stdin
cuse              hpet         i2c-6   loop2    mqueue        ptmx        stdout
disk              hugepages    i2c-7   loop3    mtd0          pts         tpm0
dma_heap          hwrng        i2c-8   loop4    mtd0ro        random      tpmrm0
dri               i2c-0        i2c-9   loop5    net           rfkill      tty
```

Ad esempio **/dev/zero** rappresenta un disco virtuale che contiene un infinita quantità di zero e viene usato per sovrascrivere un file di zeri eliminandone non rendendo possibile risalire al suo contenuto , cosa che invece succede quando eliminiamo un file con il comando **rm** .

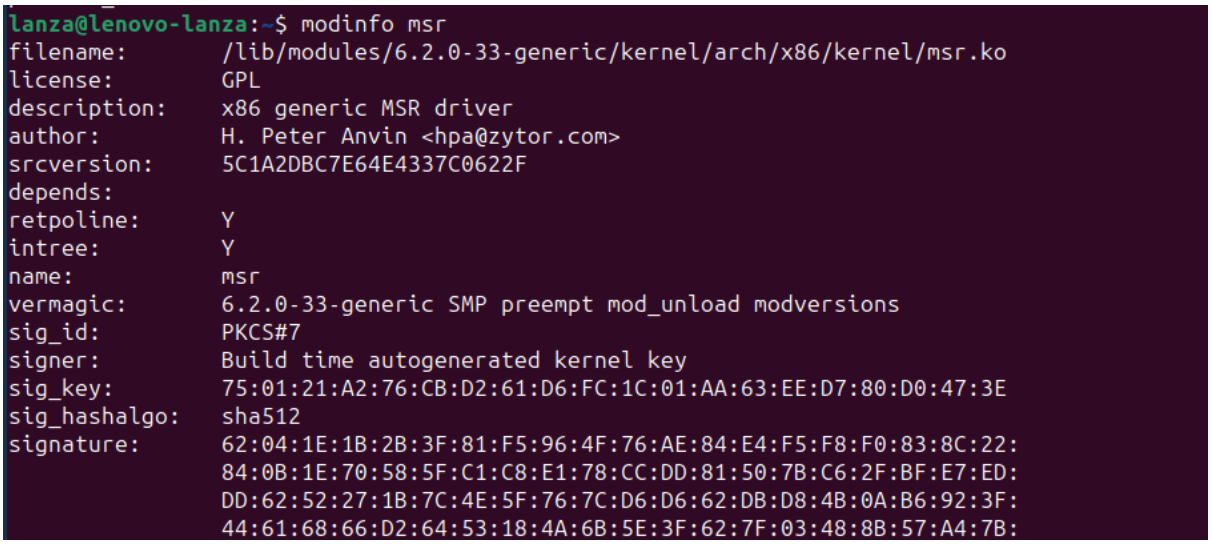
Normalmente quando una periferica viene connessa alla macchina , la CPU riceve il segnale interrupt , grazie al kernel che si occupa di caricare automaticamente il driver per la periferica , indispensabile per far comunicare la periferica con la cpu, e il daemon **udev** crea il file per la periferica dentro alla directory **/dev,** che funge da interfaccia per l'utente per poter consultare e interagire con le periferiche.

Gestione dei moduli

LSMOD - lista i moduli del kernel integrati , l'output del comando si presenta in 3 colonne , nome modulo , dimensione e n° di istanze(a volte in questa colonna è mostrato il nome del processo che richiama quel modulo).

```
lanza@lenovo-lanza:~$ lsmod
Module                  Size  Used by
xt_conntrack           16384  2
nft_chain_nat          16384  3
xt_MASQUERADE          20480  2
nf_nat                 61440  2 nft_chain_nat,xt_MASQUERADE
nf_conntrack_netlink    61440  0
nf_conntrack          200704  4 xt_conntrack,nf_nat,nf_conntrack_netlink,xt_MASQUERADE
nf_defrag_ipv6         24576  1 nf_conntrack
nf_defrag_ipv4         16384  1 nf_conntrack
xfrm_user              61440  1
xfrm_algo              20480  1 xfrm_user
xt_addrtype            16384  2
nft_compat             20480  6
nf_tables             348160  96 nft_compat,nft_chain_nat
libcrc32c              16384  3 nf_conntrack,nf_nat,nf_tables
nfnetlink              24576  4 nft_compat,nf_conntrack_netlink,nf_tables
br_netfilter           32768  0
```

MODINFO - Comando che , dato il nome di un modulo , ci permette di visualizzare le informazioni riguardanti quest'ultimo .



RMMOD - Dato il nome di un modulo lo elimina
MODPROBE - Inserisce un modulo al kernel utilizzando il nome del modulo
INSMOD - Inserisce un modulo al kernel utilizzando il path del modulo

/sys - È la cartella che contiene i file che servono a visualizzare o configurare i bus e device di sistema.
/sys/module - qui sono contenuti i file di configurazione dei moduli

Gestione dei Bus

D-Bus è un sistema di comunicazione interprocessuale (IPC) utilizzato in ambienti basati su Unix e Linux per consentire la comunicazione tra processi all'interno del sistema.
D-Bus funziona utilizzando un sistema di autobus (bus system), che agisce come canale di comunicazione per tutti i processi che vogliono scambiare messaggi.
In particolare abbiamo guardato il funzionamento di **system bus** e **session bus** .

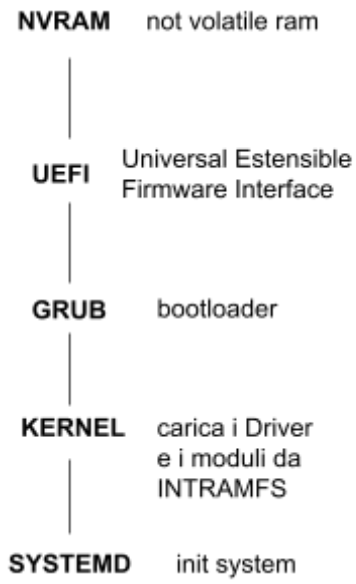
- **system-bus** Il System Bus è un bus centrale che collega processi a livello di sistema e servizi centrali del sistema operativo.
  Fa comunicare il kernel con le applicazioni
- **session-bus** si occupa della comunicazione tra le applicazioni ed è specifico per le sessioni dell'utente

Busctl - comando per visualizzare informazioni sul D-bus , permette diverse opzioni tra cui **busctl list** per poter listare gli oggetti Dbus presenti sul Dbus.

NAME	PID	PROCESS	USER	CONNECTION	UNIT	SESSION	DESCRIPTION
:1.0	570	systemd-resolve	systemd-resolve	:1.0	systemd-resolved.service	-	-
:1.1	569	systemd-oomd	systemd-oom	:1.1	systemd-oomd.service	-	-
:1.10	740	wpa_supplicant	root	:1.10	wpa_supplicant.service	-	-
:1.100	2822	fwupd	root	:1.100	fwupd.service	-	-
:1.101	3082	gnome-calendar	lanza	:1.101	user@1000.service	-	-
:1.102	3425	gsd-xsettings	lanza	:1.102	user@1000.service	-	-
:1.103	3328	firefox	lanza	:1.103	user@1000.service	-	-
:1.11	730	switcheroo-cont	root	:1.11	switcheroo-control.service	-	-
:1.115	5023	Discord	lanza	:1.115	user@1000.service	-	-
:1.12	728	accounts-daemon	root	:1.12	accounts-daemon.service	-	-
:1.124	5973	busctl	lanza	:1.124	user@1000.service	-	-
:1.13	737	NetworkManager	root	:1.13	NetworkManager.service	-	-
:1.14	742	ModemManager	root	:1.14	ModemManager.service	-	-
:1.15	732	systemd-logind	root	:1.15	systemd-logind.service	-	-
:1.16	817	cupsd	root	:1.16	cups.service	-	-
:1.17	842	dbus	lp	:1.17	cups.service	-	-
:1.2	571	systemd-timesyn	systemd-timesync	:1.2	systemd-timesyncd.service	-	-
:1.20	857	gdm3	root	:1.20	gdm.service	-	-
:1.23	834	unattended-upgr	root	:1.23	unattended-upgrades.service	-	-
:1.26	733	thermald	root	:1.26	thermald.service	-	-
:1.27	951	upowerd	root	:1.27	upower.service	-	-
:1.3	686	bluetoothd	root	:1.3	bluetooth.service	-	-
:1.32	984	rtkit-daemon	root	:1.32	rtkit-daemon.service	-	-

busctl status <nome del bus> è invece la funzione che mi da informazioni più dettagliate sullo stato di un singolo bus .

PROCESSO DI AVVIO

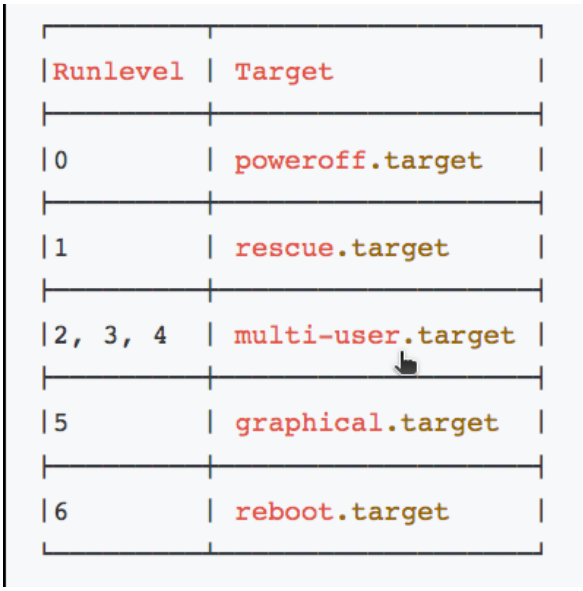


NVRAM (Not Volatile Ram) è una Ram che conserva dati sulle nostre preferenze di boot , all'avvio si occupa di caricare i firmware (BIOS o UEFI)
BIOS (Basic Input Output System) si occupa di inizializzare le periferiche e carica il bootloader, che si aspetta di trovare nei primi 512 byte del primo disco (partizione MBR)
UEFI (Universal Estensible Firmware Interface) assomiglia più ad un vero e proprio sistema operativo in grado di eseguire gli eseguibili , con UEFI non si ha più bisogno di avere la partizione MBR nei primi 512 byte del disco ma si basa sulla partizione EFI (FAT32) che viene cercata da dal firmware grazie alla tabella delle partizioni GPT e individuata , la avvia.

Inoltre rende più semplice la configurazione e l'aggiornamento del firmware grazie all'interfaccia fornita.

SYSTEMD è un sistema di Init inizializzazione dei servizi , ampiamente utilizzato , tra le sue caratteristiche principali ci sono:

- La possibilità di avviare più processi in parallelo , il che migliora l'efficienza nella fase di avvio
- La gestione automatica delle dipendenze tra processi
- Il monitoraggio costante dei servizi e il riavvio automatico in caso di erroe
- Systemd Journal un registro avanzato di log che permette di visualizzare i log di sistema.
- La gestione delle Units , systemd non gestisce solamente i servizi ma anche altri tipi di unità come Target , Device:mount , timer, socket .
  Il suo predecessore SystemV init , si basava sul concetto di RunLevel , mentre al loro posto SystemD utilizza i Target , una delle Units.



Runlevel	Target
0	poweroff.target
1	rescue.target
2, 3, 4	multi-user.target
5	graphical.target
6	reboot.target

SystemD si occupa quindi dell'avvio vero e proprio del nostro sistema operativo , ma anche una volta avviato il pc possiamo comunicare con SystemD grazie ad alcuni comandi :

- **systemctl**: Questo comando è utilizzato per avviare, fermare, riavviare, abilitare o disabilitare servizi e unità. Ad esempio il comando **systemctl set-default multi-user.target** selezionera come scelta di default per l'avvio il target multi-user , che non comprende l'interfaccia grafica.
- **journalctl**: Usato per visualizzare i log del sistema registrati dal sistema di logging di systemd. È molto potente e offre varie opzioni per filtrare e cercare i log.