Certificazione LPI 101-102

La certificazione Linux LPI è un attestato professionale internazionalmente riconosciuto, ideato da Linux Professional Institute, una organizzazione no profit dal 1999 con lo scopo di creare uno standard professionale di formazione Linux.

● Gli interrupts in linux [Certificazione LPI - 101.1 ~ 1]

Da terminale digitare:

cat /proc/interrupts

per visualizzare gli interrupts che ci sono stati durante la sessione del sistema operativo. Questo file non è persistente e viene riscritto ad ogni nuova sessione.

● ioports e DMA in Linux [Certificazione LPI - 101.1 ~ 2]

da terminale digitare:

cat /proc/ioports

al suo interno visualizziamo ogni dispositivo e il corrispondente univico spazio in memoria Una volta che un dispositivo ha mandato l'interrupts alla cpu deve mediare il trasferimento dei dati tra la periferica e la memoria e questo è lo scopo delle ioports cioè fornire ad un device un range di memoria in cui possa essere assegnato e poi la cpu Trasferisce cioè gestisce il trasferimento/mediazione dei dati che si passono in memoria.

cat /proc/dma

Mentre ioports assegna ad ogni dispositivo la sua zona in memoria e lascia gestore alla cpu le mediazione tra memoria e dispositivo, DMA -Direct Memory Access- permette di assegnare la memoria direttamente al dispositivo e bypassare la cpu nel trasferimento/mediazione con la memoria.questo dovrebbe migliorare significativamente le prestazioni per devices che fanno largo uso di memoria.

● Dispositivi e Drivers in Linux [Certificazione LPI - 101.1 ~ 3]

I dispositivi **cold-plug** che si devono sostituire necessariamente a macchina spenta al contrario i dispositivi **hot-plug** si possono sostituire a macchina accesa.

con il comando: **lspci**

possiamo visualizzare un elenco e gli indirizzi dei dispositivi pci collegati.

con **lspci -t** possiamo visualizzare una struttura ad albero.

con:

ls /dev/sd? (? è un espressione per indicare tutti i dispositivi che cominciano con sd e che hanno un (1) carattere soltanto successivamente; a differenza di * che avrebbe indicato qualsiasi cosa dopo sd.

Con il comando **lsmod** possiamo visualizzare tutti i driver che sono stati caricati.
Con il comando **rmmod** (più il nome del modulo ad es. **rmmod pcspkr** si può rimuovere un driver (modulo) dalla memoria (per usare questo comando bisogna essere root).

per filtrare lsmod possiamo utilizzare una **pipe** tipo:
lsmod | grep pcs*

Per caricare un modulo si usa invece il comando:
insmod (seguito dal nome) questo comando richiede il **full-path** cioè il percorso completo nel file system. Per rintracciare il percorso completo: /lib/modules

find /lib/modules/$(uname -r)/ -iname "*pcspkr*.ko*"

(**$** aperta parentesi sta per: esegui il comando)
(**ko** sarebbero i moduli da caricare/spenti).

Un altro comando più pratico è: **modprobe -r pcspkr** per rimuovere un modulo.

modprobe pcspkr si usa invece per caricare il modulo.

● Spiegazione di D-Bus, HAL, udev [Certificazione LPI - 101.1 ~ 4]

In questo capitolo vediamo a che servono, come funzionano e dove si trovano **D-Bus, HAL Hardware Abstraction Layer - udev**, e perchè sono così importanti.

"D-Bus, fornisce un canale di comunicazione per permettere alle applicazioni di parlarsi l'un l'altra fornisce un demone di sistema per eventi come "un nuovo hardware è stato collegato" o "la coda di stampa è cambiata" ecc.. e un demone per-user-login-session per far comunicare le applicazioni in userspace"

"**HAL**, deprecato dal 2008/2010 --integrato in **udev** si collega a dbus e fornisce un database aggiornato in tempo reale sulle periferiche collegate al computer il suo scopo è di fornire alle applicazioni una API semplice, portabile e astratta. indipendentemente dall' hardware di appoggio "

● BIOS, EFI, Bootloader, runlevel [Certificazione LPI - 101.2 ~ 1]

Breve teoria sulla sequenza di boot.

Nel caso di un computer che utilizza (il vecchio) sistema **BIOS** (che riesce ad indirizzare **16 bit**) viene prima di tutto eseguito il **POST** (Power On Self Test) il quale controlla lo stato della memoria, dell'hardware e del sistema in generale. Successivamente carica lo **STAGE 1 boot loader**, lo carica dal **MBR** (Master Boot Record) MBR è contenuto nei primi **512 byte** del disco rigido (designato per il boot) il quale al suo interno contiene un codice che a sua volta carica lo **STAGE 2 boot loader** che a sua volta carica il sistema operativo (**SO**). Il boot loader (nel caso di Linux) può caricare anche insieme al Kernel un

initrd (initial ram disk), lo si può trovare sia nella notazione **initramfs** (init ram filesystem) ed è un file che contiene I moduli kernel caricabili compilati al di fuori del kernel; solitamente è un file compresso e viene caricato in ram dal boot loader (che può essere GRUB o LILO). Il kernel vi accede come se fosse un filesystem montato quindi senza alcun problema. Su alcuni sistemi come **FEDORA12** può essere creato con una utility chiamata "**dracut**" (questa utility crea l'initrd). Se vogliamo vedere cosa c'è dentro possiamo farlo con il comando: **lsinitrd .**

Un boot loader può caricare un altro bootloader, un esempio classico può essere quando abbiamo installato **UBUNTU** e GRUB è il boot loader predefinito, quindi dopo il **POST** il **BIOS** cerca di caricare lo **STAGE 1 boot loader** (leggendo dai primi 512 byte del MBR) il quale trova il codice di **GRUB** quindi carica GRUB che mostra a sua volta una lista di opzioni ed eventualmente la lista di altri sistemi operativi installati, se selezioniamo Linux allora si passa allo **STAGE 2 boot loader** che a sua volta avvia il sistema operativo; se invece selezioniamo (ad esempio) **Windows**, GRUB carica un altro boot loader, il **Windows boot loader** che poi effettua il boot di Windows (perchè GRUB non può fare il boot di Windows). Questa tecnica viene chiamata **Chain Loading** (caricamento a catena).

Recentemente in sostituzione al vecchio BIOS è stata introdotta una nuova tecnologia chiamata **EFI / UEFI** , EFI è un progetto di **INTEL** abbandonato nel 2005 in favore di **UEFI** (Unified Extensible Firmware Interface).
UEFI può caricare sistemi **UEFI** e sistemi **legacy**. UEFI è un firmware complesso che richiede una partizione UEFI dedicata sul disco per poter effettuare il boot.
UEFI può leggere il contenuto delle partizioni.

I boot loader per sistemi **BIOS** in Linux sono: **LILO** (Linux Loader) e **GRUB** (**Legacy)** (Gnu Grand Unified Bootloader). Per i sistemi UEFI sono **ELILO** (Efi LILO) e **GRUB 2.0** comunemente chiamato solo **GRUB**.

Una volta che il boot loader carica il **kernel**, l'**initrd** e viene completato il caricamento del kernel normalmente quest'ultimo avvia il processo **/sbin/init** ed è un processo che viene avviato per primo e il suo **PID** sarà sempre **1,** il suo scopo è quello di far partire tutti gli altri processi, rimane attivo fino allo spegnimento e ha come figli tutti gli altri processi.

Su i sistemi che fanno uso del tradizionale **systemv init** o su i sistemi più nuovi che utilizzano l'inizializzazione **Upstart** il processo sarà sempre chiamato **init**.
Nei sistemi più recenti che utilizzano **systemd** invece **/sbin/init** è un collegamento simbolico a **/lib/systemd/systemd**.

Nei sistemi **Upstart** si ha il concetto dei **JOBS**.
Nei sistemi con **systemd** si hanno le **UNIT.** Esempi di UNIT sono: **.service, .socket, .device, .mount, .automount, .target, .time, .path, .snapshot**, **.slice, .scope**.
Se il sistema carica il kernel ma fallisce nel caricare l'**init** si può tentare il ripristino avviando un programma differente all'avvio ad esempio specificando come init **/sbin/sh**.

Approfondimento su UEFI.

I primi **512 byte** di un disco fisso (primo settore del disco o meglio **MBR**) sono utilizzati per poter avviare il sistema operativo; i primi **446 byte** servono per memorizzare il **bootloader**, un programma già compilato che ha il compito di caricare il sistema operativo. Gli altri byte contengono **4 indirizzi** che indicano dove iniziano le partizioni (e infatti si possono creare non più di **4 partizioni** su un disco). Gli ultimi **2 byte** sono

utilizzati per effettuare dei controlli (una specie di **checksum**). Questo sistema per creare le partizioni su un disco si chiama **MS-DOS**.Per mantenere questo sistema ma allo stesso tempo superare il limite di 4 partizioni si è creato un metodo: una partizione “estesa” con le partizioni “logiche”. I 4 indirizzi creano 4 partizioni dette primarie e in queste si possono creare altre dette **logiche**.
Oltre a MS-DOS esistono altri sistemi di partizionamento e uno di questi è **GPT** (Guid Partition Table) il quale ci permette di creare fino a **128 partizioni** primarie e rispetto a MS-DOS non ha un limite di giga che può leggere; ad esempio se abbiamo un disco più grande di **2 terabyte** MS-DOS lo calcolerà sempre come uno da 2 terabyte che è il suo massimo. GPT ha all’inizio 512 byte che si chiamano **protectiv master boot record**) praticamente questa parte viene “saltata” poi, **17 kb** di cui un primo tot. dedicato al primary gpt header e questo è uno spazio riservato a GPT. Successivamente ci stanno scritti gli indirizzi per poter creare le partizioni e al termine c’è il secondary gpt header che una copia del primo (a parte il protectiv master boot record) quindi è anche un sistema più sicuro rispetto a MS-DOS.
In una scheda madre bisogna utilizzare un solo sistema di partizionamento e per poter utilizzare UEFI bisogna adottare il sistema di partizionamento GPT. Creando una partizione dedicata a UEFI ci si possono installare dei programmi come anche GRUB (o altro); infatti non si è più obbligati ad inserire il bootloader all’interno dei primi **446 byte** ma può risiedere all’interno della partizione UEFI come un normale programma.
Il Flag boot in GPT determina dove si trova la partizione UEFI. **EFI** è l'acronimo di **Extensible Firmware Interface**, una tecnologia annunciata, inizialmente solo da Intel al momento della presentazione della propria achitettura IA-64 del processore Itanium, e poi ripresentata in maniera decisamente più consistente insieme a Microsoft a fine 2003, che sostituirà gradualmente gli attuali BIOS delle schede madri, e arrivata per la prima volta sul mercato agli inizi del 2006 grazie ai primi iMac Intel Core Duo di Apple, affiancandosi a un'altra tecnologia Intel arrivata a fine 2005, iAMT.

Caratteristiche principali

EFI consentirà ai produttori di integrare nel firmware del computer applicazioni e nuove funzionalità, fra cui tool per la diagnostica e il ripristino dei dati, servizi di crittografia dei dati, e anche estensioni per la gestione dei consumi.
Secondo Intel, EFI renderà anche più facile gestire PC e server da remoto, aiutando così le aziende a ridurre i costi di manutenzione e supporto, e potrà gestire direttamente le connessioni di rete per connettersi ad una LAN o a Internet. A tal proposito i BIOS basati sullo standard EFI saranno dotati di alcune utility di rete e, eventualmente, anche di un browser Web. L'altra miglioria promessa da EFI è la capacità di ridurre anche drasticamente i tempi di caricamento del sistema operativo e quello di supportare, similmente a quanto succede con i computer palmari, forme di avvio istantaneo. L'EFI ha anche il compito di dotare il firmware del PC di un'interfaccia grafica più amichevole, facile da usare e in grado di supportare le risoluzioni video permesse dalle moderne schede grafiche. In aggiunta a questo, l'EFI fornirà un ambiente per il boot multipiattaforma capace di fornire i servizi base richiesti dai sistemi operativi. In un certo senso, EFI si può considerare un piccolo sistema operativo dedicato a presiedere tutte quelle operazioni che intercorrono fra l'accensione fisica della macchina e l'avvio del sistema operativo vero e proprio, superando però tutte le problematiche emerse negli anni con gli attuali BIOS. Come tale infatti, sarà in grado di far girare applicazioni di alto livello scritte attraverso tool di programmazione standard. Tutto questo verrà reso possibile dal fatto che le interfacce di EFI saranno totalmente scritte in linguaggio C++, mettendo così definitivamente al bando l'ostico codice assembler degli attuali BIOS. Sebbene EFI sia già arrivato sul mercato nei nuovi iMac, questi si appoggiano a un chipset proprietario sviluppato da

Apple. Il primo chipset di Intel a supportare i BIOS EFI arriverà all'inizio del 2007 grazie alla piattaforma mobile Santa Rosa basata sul chipset Crestine e il processore Merom. Windows Vista x64 supporta EFI in maniera nativa.

Interfaccia EFI = interfaccia tra il firmware e il sistema operativo. L'interfaccia EFI (Extensible Firmare Interface) definisce un nuovo stile di partizione denominato GPT (Guid Partition Table). Nei computer Itanium l'interfaccia EFI svolge le stesse funzioni del Bios dei computer x86. Offre tuttavia funzioni aggiuntive che rendono disponibile un metodo uniforme di avvio in qualsiasi sistema operativo compatibile e un metodo semplice per l'aggiunta di driver EFI per le nuove periferiche avviabili senza che sia necessario aggiornare il firmware del computer.

Tabella di partizionamento GUID = schema di partizionamento del disco utilizzato dall'interfaccia EFI. La tabella GPT offre maggior vantaggi rispetto a MBR (Master Boot Record) in quanto supporta fino a 128 partizioni per disco, volumi fino a 18 esabyte, tabelle di partizioni primarie di backup per la ridondanza, nonché ID univoci (GUID) per disci e partizioni.

MBR = Record di avvio principale. Il primo settore di un disco rigido che inizializza il processo di avvio del computer. IL record MBR contiene la tabella di partizione per il disco e una parte limitata di codice eseguibile denominato codice di avvio principale.

Bios = sistema di input/output di base. Nei computer x86, il set di routine software essenziali che verifica i componenti hardware in fase di avvio, avvia il sistema operativo e supporta il trasferimento di dati tra periferiche ed hardware. Il bios è archiviato nella memoria di sola lettura (Rom) in modo che possa essere eseguito all'accensione del computer.

Partizione = parte di un disco fisico che funziona come se fosse un'unità disco separata fisicamente. Dopo la creazione della partizione (su Windows) è necessario formattarla e assegnarle una lettera di unità prima di poter procedere all'archiviazione dei dati.

Runlevel

In informatica il runlevel è un concetto tipico dei sistemi UNIX e Unix-like e rappresenta lo stato di attività di una macchina, relativamente ai programmi in esecuzione e ai servizi offerti. Ogni runlevel è identificato da un numero, solitamente compreso tra 0 e 6, e ad ogni stato associato la macchina esegue script di sistema relativi alle funzionalità assegnate a quel livello. Non esiste uno standard su cosa rappresenti ogni singolo runlevel, anche se alcuni sono riservati:

- **Il runlevel 0** serve ad arrestare (halt) il sistema. In sostanza corrisponde ad uno stato in cui nessun programma è in esecuzione (e la macchina è spenta).

- **Il runlevel 1** è detto Single user mode ("modalità ad utente singolo"): in questo stato non vi sono programmi in esecuzione in background, e nessun utente oltre all'amministratore di sistema ha accesso al sistema, per cui esso è sotto il suo pieno controllo. È utile, ad esempio, per eseguire attività di manutenzione come controlli sui dischi o aggiornare file di sistema, dato che vi è la certezza che nessun processo li stia utilizzando e che nessun utente possa intervenire all'insaputa dell'amministratore.

• **Il runlevel 2** avvia la macchina in modalità multiutente con il networking abilitato ma senza servizi di rete.

• **Il runlevel 3** avvia in modalità multiutente con tutte le funzionalità di networking abilitate, e con tutti i relativi servizi attivi.

• **Il runlevel 4** generalmente non è utilizzato;

• **Il runlevel 5** avvia in modalità multiutente con tutte le funzionalità di networking abilitate come il runlevel 3, e con un server grafico (quasi sempre X Window System) abilitato;

• **Il runlevel 6** serve a riavviare (reboot) il sistema. È simile al runlevel 0, con la differenza che al termine dell'inizializzazione, la macchina viene riavviata piuttosto che spenta.

I runlevel rimanenti sono personalizzabili dall'amministratore del sistema. Sui sistemi GNU/Linux è pratica comune avere un runlevel corrispondente alla modalità grafica e uno corrispondente alla modalità testuale. Un altro possibile utilizzo è quello di creare un runlevel in cui vengono lanciati i servizi di rete, e un altro in cui questi non sono attivi, in modo da poter facilmente cambiare tra di essi a seconda della connettività disponibile.

Conoscere il runlevel attuale in esecuzione.

Per conoscere il runlevel attuale e quello precedente e si può usare il comando **runlevel**.

Cambiare il runlevel in esecuzione.

Per cambiare runlevel si utilizza il comando **init** (o **telinit**), seguito dal numero del runlevel a cui si desidera passare. Esso prenderà tutti i provvedimenti necessari per eseguire il cambio di runlevel.

Runlevel e diagnostica al Boot [Certificazione LPI - 101.2 ~ 2]

Per cambiare Runlevel all'avvio del sistema dal menu di GRUB premere "**e**" (che sta per **edit**) e posizionarsi sull'ultima riga del kernel e inserire la modalità di runlevel desiderata ad esempio 1 , 2 , 3 , ecc. Questo dirà all'init in che modalità effettuare il boot.

Per vedere qual'è il runlevel di default digitare: **grep initdefault /etc/initab**
Se però ci troviamo su una distro che utilizza **systemd** ci verrà mostrato un avviso che ci informa che systemd utilizza i **Target** anziché i runlevel e di default troveremo i principali target appunto, utilizzati. Per scoprire qual'è quello predefinito digitare il comando: **systemctl get-default** .
Per cambiare il target di default digitiamo: **systemctl set-default *multi-user.target*** (oppure ***grafical.target***).

Per vedere se il target è stato sostituito possiamo spostarci direttamente nella cartella con: **cd /systemd/system** e fare un **ls -l runlevel * .target** (su tutti i file che si chiamano runlevel e che hanno estensione .target).