LPI Certification

Exam 101-500
LPI Level 1 Certification Part 1

Exam Questions (2026-07-15 Updated)

Linux Professional Institute
LPIC-1

101-500 Linux Professional Institute Level 1 (Jul 2026)

QUESTION 1

Which of the following statements are true about the boot sequence of a PC using a BIOS? (Choose two.)

A. Some parts of the boot process can be configured from the BIOS

B. Linux does not require the assistance of the BIOS to boot a computer

C. The BIOS boot process starts only if secondary storage, such as the hard disk, is functional

D. The BIOS initiates the boot process after turning the computer on

E. The BIOS is started by loading hardware drivers from secondary storage, such as the hard disk

Answer: A, D

A 正確核心原因：開機流程的特定部分可以在 BIOS 中進行設定

管理員可以進入 BIOS 設定畫面，調整與開機直接相關的參數。

例如修改開機裝置順序（Boot Device Priority，如優先從硬碟、USB 或光碟開機）。

也可以設定硬體參數、啟用或停用特定內建裝置，進而影響後續的開機引導流程。

D 正確核心原因：BIOS 會在電腦開啟電源後立刻啟動開機流程

當按下電腦主機的電源鍵後，主機板會供電並觸發中央處理器（CPU）執行固化在唯讀記憶體（ROM）中的 BIOS 程式。

BIOS 啟動後會先執行開機自檢（POST，Power-On Self-Test）來確認基本硬體正常。

隨後便會去尋找啟動裝置，將控制權交給引導程式，正式初始化整個開機引導流程。

錯誤選項詳細除錯 ：

B. 錯誤：Linux 核心引導初期依然需要 BIOS 的協助

- 在使用傳統 BIOS 的 PC 上，Linux 無法完全脫離 BIOS 單獨開機。
- Linux 的引導程式（如 GRUB）必須依賴 BIOS 提供的實體中斷服務（如 INT 13h 磁碟讀取功能），才能將 Linux 核心與 initramfs 從硬碟載入到記憶體中。
- 只有在核心成功接管硬體、進入保護模式並載入專屬的硬體驅動程式後，Linux 才會停止依賴 BIOS。

C. 錯誤：即使次級儲存媒體（如硬碟）故障，BIOS 開機流程依然會啟動

- BIOS 程式是存在主機板上的快閃記憶體中，它不需要任何硬碟就能運作。
- 即使電腦上沒有安裝任何硬碟，或是硬碟完全損壞，電腦開機後 BIOS 依然會正常執行。
- 它會完成 POST 檢查，並在畫面上提示找不到開機裝置的錯誤訊息（例如 No bootable device found）。

E. 錯誤：BIOS 啟動不需要從硬碟等外部儲存裝置載入硬體驅動

- BIOS 是電腦系統中「最底層」的韌體。
- 它的初始化與執行完全依靠主機板上內建的 ROM/Flash 晶片內容。
- 它不需要、也無法在啟動初期從硬碟去載入任何硬體驅動程式；相反地，是 BIOS 先行啟動後，才賦予電腦去讀取硬碟資料的能力。

QUESTION 2

What is true regarding UEFI firmware? (Choose two.)

A. It can read and interpret partition tables

B. It can use and read certain file systems

C. It stores its entire configuration on the **`/boot/`** partition

D. It is stored in a special area within the GPT metadata

E. It is loaded from a fixed boot disk position

Answer: A, B

A 正確核心原因：UEFI 韌體能夠讀取並解析分割區表

- 相較於傳統 BIOS 只能盲目讀取磁碟第 0 軌的 MBR 啟動碼，UEFI 韌體具備更高級的邏輯分析能力。
- 它能直接原生讀取、解析並識別 GPT（GUID Partition Table）與傳統 MBR 的分割區表結構，進而尋找存放引導檔案的特定分割區。

B 正確核心原因：UEFI 韌體能夠識別並讀取特定的檔案系統

- 這是 UEFI 技術最重大的革新之一。
- UEFI 韌體內建了特定檔案系統的驅動程式（依據官方規範，必須強制支援 FAT12、FAT16 與 FAT32 格式）。
- 這使得 UEFI 能夠直接進入 EFI 系統分割區（ESP，EFI System Partition）去瀏覽目錄，並直接讀取、載入內部的二進位引導檔案（如 .efi 檔案），而不需要依賴硬碟引導扇區的低階組合語言代碼。

錯誤選項詳細除錯

C. 錯誤：UEFI 的配置儲存在主機板晶片中，而非硬碟的 /boot/ 分割區

- UEFI 的底層硬體配置、安全開機金鑰（Secure Boot Keys）以及開機順序選單等資訊，是儲存在主機板上的非揮發性隨機存取記憶體（NVRAM）晶片中。
- 硬碟上的 /boot/ 或者是 ESP 分割區只用來儲存作業系統的引導檔案（如核心或 GRUB 執行檔），並不儲存 UEFI 韌體本身的硬體設定。

D. 錯誤：UEFI 韌體儲存在主機板的快閃記憶體晶片中

- UEFI 是一種硬體韌體（Firmware），它固化在主機板上的唯讀記憶體（ROM/Flash 晶片）裡。
- GPT（GUID 分割表）中只包含磁碟分割的元資料，韌體絕對不可能儲存在硬碟的 GPT 元資料欄位中。

E. 錯誤：UEFI 是透過檔案路徑引導，而非固定磁碟位置

- 傳統 BIOS 必須從硬碟特定的固定扇區（如主引導記錄 MBR 所在的第 0 軌扇區）載入開機碼。
- 而 UEFI 則是透過指定「磁碟分割區與標準檔案路徑」（例如 \EFI\BOOT\BOOTX64.EFI）的方式來彈性載入引導程序，這徹底擺脫了「固定磁碟位置」的物理限制。

QUESTION 3

Where is the bootloader stored on the hard disk of a UEFI system?

A. In the EFI Boot Record (EBR).
B. In the Master Boot Record (MBR).
C. On the EFI System Partition (ESP).
D. On the partition labeled **`boot`**.
E. On the partition number **`127`**.

Answer: C

C 正確核心原因：UEFI 引導程式儲存在專屬的 EFI 系統分割區中

- 在採用 UEFI 韌體的系統中，開機引導程式（Bootloader，如 grubx64.efi 或者是 Windows Boot Manager）不再儲存在硬碟的隱藏扇區中。
- UEFI 規範要求在 GPT 磁碟上建立一個格式為 FAT32（或 FAT16/12）的專屬分割區，稱為 EFI 系統分割區（ESP，EFI System Partition）。
- 所有作業系統的引導檔案都以標準檔案的形式，分門別類存放在該分割區的 \EFI\ 目錄結構下，供 UEFI 韌體直接讀取並引導開機。

錯誤選項詳細除錯

A. 錯誤：EFI 檔案系統結構中不存在 EBR 這種代號

EBR（Extended Boot Record，擴展引導記錄）是傳統 MBR 分割架構下，用來定義和引導「邏輯分割區（Logical Partitions）」的鏈結結構，與 UEFI 開機完全無關。

B. 錯誤：UEFI 開機不再將引導程式寫入 MBR

MBR（主引導記錄）位於磁碟的第 0 個扇區。雖然在 GPT 磁碟上為了維持舊硬體相容性會保留一個「保護性 MBR（Protective MBR）」，但 UEFI 韌體在開機時並不會去讀取或執行 MBR 內部的啟動代碼。

D. 錯誤：標籤名稱與 UEFI 官方標準規範無關

boot 只是檔案系統的一個自訂文字標籤（LABEL）或者是傳統分區的旗標（Flag）。UEFI 韌體識別 ESP 分割區是透過專屬的 GUID 編號（即 C12A7328-F81F-11D2-BA4B-00A0C93EC93B），而非透過字串標籤。

E. 錯誤：分割區編號 127 只是 GPT 的最大分區上限

GPT 分割表預設最多支援 128 個分區（編號從 1 到 128）。編號 127 只是其中一個常規分區位置，並不是官方標準定義存放引導程式的特殊專用編號。

QUESTION 4

What is contained on the EFI System Partition?

A. The Linux root file system
B. The first stage boot loader
C. The default swap space file
D. The Linux default shell binaries
E. The user home directories

Answer: B

B 正確核心原因：EFI 系統分割區儲存了作業系統的第一階段開機引導程式

- EFI 系統分割區（ESP，EFI System Partition）是採用 UEFI 韌體架構的電腦開機時，最先被讀取的關鍵分割區。
- 它的核心任務就是用來儲存各個作業系統的第一階段開機引導程式（First-stage Boot Loader）與相關驅動程式（例如 Linux 的 grubx64.efi 或者是 Windows 的 bootmgfw.efi）。
- UEFI 韌體會直接進入這個格式化為 FAT32 的分割區，執行該 EFI 執行檔，進而將開機主控權傳遞給後續的系統核心。

錯誤選項詳細除錯

A. 錯誤：Linux 根檔案系統儲存在獨立的常規分區中

Linux 根檔案系統（Root Filesystem，包含 /usr、/etc、/var 等全系統目錄）儲存在獨立的常規 Linux 分割區中（例如格式化為 ext4 或 XFS 的主分割區），絕對不會放在容量極小的 ESP 分割區內。

C. 錯誤：預設置換空間（Swap）擁有專屬的獨立分割區或檔案

置換空間（Swap Space）在 Linux 中有其專屬的分區類型（Linux swap）或者是存放在根目錄下的 swap 檔案，用作虛擬記憶體擴充，與提供引導程式的 ESP 分割區無關。

D. 錯誤：Linux 預設 Shell 執行檔儲存在根檔案系統中

標準的 Shell 執行檔（如 Bash 或者是 Sh）是屬於使用者空間的二進位程式，依照 FHS 標準規範固定存放在根檔案系統下的 /bin/ 或 /usr/bin/ 之中。

E. 錯誤：使用者家目錄儲存在根檔案系統或獨立的 /home 分割區中

使用者的個人資料與設定檔目錄（User Home Directories）在 FHS 規範中被嚴格定義存放在 /home/ 底下，此路徑通常包含在根分割區內或獨立劃分出來，不屬於開機引導資料。

QUESTION 5

What are the main differences between GPT and MBR partition tables regarding maximum number and size of partitions? (Choose two.)

A. MBR can handle partition sizes up to 4 TB, whereas GPT supports partition sizes up to 128 ZB.

B. By default, GPT can manage up to 128 partitions while MBR only supports four primary partitions.

C. By default, GPT can manage up to 64 partitions while MBR only supports 16 primary partitions.

D. MBR can handle partition sizes up to 2.2 TB, whereas GPT supports sizes up to 9.4 ZB.

E. Both GPT and MBR support up to four primary partitions, each with up to 4096 TB.

Answer: B, D

B 正確核心原因：GPT 預設支援最多 128 個分割區，而 MBR 僅支援 4 個主分割區

- 在傳統的 MBR（Master Boot Record）架構中，分割表空間極度有限，原生最多只能定義 4 個主分割區（Primary Partitions）。若需要更多分割區，必須將其中一個犧牲改為擴充分割區來包容邏輯分割區。
- 而現代的 GPT（GUID Partition Table）徹底解決了這個限制，在標準實作（如 Windows 與多數 Linux 預設）下，GPT 允許直接建立多達 128 個獨立的分割區，不再需要傳統的擴充/邏輯分割區盲腸設計。

D 正確核心原因：MBR 的容量上限為 2.2 TB，而 GPT 理論上可支援高達 9.4 ZB

- MBR 在記錄磁碟磁區（Sector）數量時使用的是 32 位元（32-bit）的定址數值。在常見的每磁區 512 位元組（512 bytes）定義下，其數學極限就是 2 的 32 次方乘以 512 位元組，大約等於 2.199 TB（即通常所說的 2.2 TB）。
- 而 GPT 則改用 64 位元（64-bit）的邏輯區塊定址（LBA），這使得它的最大定址容量上限暴增至 2 的 64 次方乘以 512 位元組，大約等於 9.4 ZB（Zettabytes，1 ZB 等於 10 億 TB），徹底擺脫了現代大容量硬碟的硬體瓶頸。

錯誤選項詳細除錯

A. 錯誤：MBR 的極限是 2.2 TB 而非 4 TB

選項中提到 MBR 能處理高達 4 TB 的容量是錯誤的，MBR 因 32 位元限制，容量超過 2.2 TB 以上的空間將完全無法被定址與辨識。

C. 錯誤：數值完全捏造顛倒

選項中所稱的「GPT 支援 64 個、MBR 支援 16 個主分割區」是完全錯誤的虛構數據，與兩者的官方技術規範均不相符。

E. 錯誤：兩者的架構限制與容量完全不同

GPT 與 MBR 在主分割區數量和單一分割區容量上限上有著本質上的巨大差異，並非如選項所說的「兩者都一樣支援 4 個主分區及 4096 TB」。

QUESTION 6

What is the first program the Linux kernel starts at boot time when using System V init?

A. **`/lib/init.so`**

B. **`/proc/sys/kernel/init`**

C. **`/etc/rc.d/rcinit`**

D. **`/sbin/init`**

E. **`/boot/init`**

Answer: D

D 正確核心原因：/sbin/init 是 Linux 核心啟動的第一個使用者空間程序

- 在傳統的 System V init 系統中，Linux 核心完成硬體初始化並掛載根檔案系統（Root Filesystem）後，在使用者空間（User Space）所拉起的第一個常駐程式就是 /sbin/init。
- 它的行程識別碼（PID，Process ID）固定為 1。
- 它負責讀取設定檔 /etc/inittab，並依據定義的執行級別（Runlevels）去依序觸發對應的系統初始化腳本與服務，是所有後續行程的父行程。

錯誤選項詳細除錯

A. 錯誤：/lib/init.so 不是執行檔，系統中也沒有此標準共用庫

以 .so 結尾的通常是動態共享函式庫（Shared Object）。Linux 核心的第一個啟動標的必須是一個可以獨立執行的二進位執行檔或腳本，並非共享庫。

B. 錯誤：/proc/sys/kernel/init 不是啟動程序本身

雖然核心有一部分關於 init 路徑的變數可能與此類似，但在 /proc/ 虛擬檔案系統中的結構是用於核心參數的動態讀寫與檢視，並非存放初始執行檔的實體路徑。

C. 錯誤：/etc/rc.d/rcinit 是錯誤或特定發行版自訂的後續腳本名稱

在 System V init 中，/etc/rc.d/ 或 /etc/init.d/ 目錄下包含了許多服務控制腳本（如 rc 腳本），但這些都是由 /sbin/init 啟動後才被調用執行的子程序，並不是 Linux 核心直接載入的第一個程式。

E. 錯誤：/boot/ 專門存放開機引導檔案，不存放 init 程式

/boot/ 目錄在 FHS 規範中用來存放核心映像（vmlinuz）與開機引導程式（GRUB）相關檔案。在開機控制權移交給使用者空間後，核心不會去 /boot/ 目錄尋找 init 程式。

QUESTION 7

Given the following two symbolic links in a System V init configuration:

`/etc/rc1.d/K01apache2`

`/etc/rc2.d/S02apache2`

When are the scripts executed that are referenced by these links? (Choose two.)

A. **`S02apache2`** is run when runlevel 2 is entered.

B. **`S02apache2`** is run when runlevel 2 is left.

C. **`K01apache2`** is never run because **`K`** indicates a deactivated service.

D. Both **`S02apache2`** and **`K01apache2`** are run during a system shutdown.

E. **`K01apache2`** is run when runlevel 1 is entered.

Answer: A, E

A 正確核心原因：S 開頭的腳本會在進入該執行級別時執行

- 在 System V init 架構中，以大寫字母 S（Start）開頭的符號連結，代表「啟動服務」。
- 緊接著的數字 02 代表執行順序的優先級（數字越小越先執行）。
- 因此，當系統切換並進入執行級別 2（Runlevel 2）時，系統會自動呼叫並執行 /etc/rc2.d/S02apache2 所指向的腳本，並帶入 start 參數以啟動 apache2 服務。

E 正確核心原因：K 開頭的腳本會在進入該執行級別時執行

- 以大寫字母 K（Kill）開頭的符號連結，代表「終止服務」。
- 許多人常誤以為 K 代表不執行，但實際上，當系統切換並進入執行級別 1（Runlevel 1，通常為單使用者救援模式）時，系統會主動執行該目錄下的 K 開頭腳本（例如 /etc/rc1.d/K01apache2），並帶入 stop 參數，以便在進入該層級時將 apache2 服務強行關閉。

錯誤選項詳細除錯

B. 錯誤：腳本是在進入（Entered）新層級時觸發，而非離開（Left）時

System V init 的運作邏輯是：當管理員下達切換層級指令後，系統會先轉移到目標 Runlevel，然後讀取「目標 Runlevel 目錄」內的所有連結。它不會在離開原本層級的當下回頭去觸發離開層級的特殊腳本。

C. 錯誤：K 腳本會被執行以進行關閉動作

K 並不代表服務被停用而不執行。相反地，它是用來確保系統在進入該層級時，該服務是處於被「殺死/終止（Kill）」的狀態。因此它一定會被調用執行並傳入 stop 參數。

D. 錯誤：兩者不會同時在系統關機時被呼叫

系統關機對應的是特定的執行級別 0（Runlevel 0）。當關機時，系統只會去讀取並執行 /etc/rc0.d/ 目錄下的軟連結，並不會跨目錄去執行 rc1.d 或 rc2.d 裡面的特定編號連結。

QUESTION 8

What is the process ID number of the **`init`** process on a System V init based system?

A. **`-1`**

B. **`0`**

C. **`1`**

D. It is different with each reboot

E. It is set to the current run level

Answer: C

C 正確核心原因：init 行程的 PID 在所有類 Unix 系統中固定且永遠為 1

- 在 System V init（以及任何其他 Linux 初始化系統如 systemd）中，init 是核心（Kernel）在引導期間拉起的第一個「使用者空間（User-mode）」行程。
- 因為它是系統第一個啟動的常規行程，作業系統核心會固定為其指派行程識別碼（PID，Process ID）為 1。
- PID 1 的 init 行程是系統中所有其他使用者行程的共同始祖（Parent/Ancestor），並負責接管所有孤兒行程（Orphaned Processes），在系統關機前它會一直保持執行狀態。

錯誤選項詳細除錯

A. 錯誤：-1 在 Linux 中通常被用作錯誤代碼或特定系統呼叫的廣播標記

在 Linux 行程管理中，-1 絕不是一個有效的行程 PID 號碼。例如在 kill(-1, SIGKILL) 系統呼叫中，-1 代表向當前用戶有權發送訊號的所有行程進行群體廣播。

B. 錯誤：PID 0 是屬於內核交換/排程行程（Idle 任務）

PID 0 是在系統最早期、核心剛被載入時由核心本身所建立的第一個內部執行緒（通常被稱為 swapper、sched 或 idle 任務）。

它完全運作在核心空間中，不屬於任何使用者空間的實體程式（如 init），且無法在常規的 ps 指令清單中查到。

D. 錯誤：init 的 PID 是硬性保留的，重開機絕對不會改變

雖然其他常規應用程式（如 Apache、SSH）在每次重新開機（Reboot）時會被分配到不同的隨機 PID 號碼，但 init（PID 1）和核心執行緒（如 kthreadd PID 2）的號碼是 Linux 核心底層技術硬性保留並寫死的，每一次開機都必定完全相同。

E. 錯誤：行程 ID 與執行級別（Runlevel）是完全不同的系統概念

執行級別（Runlevel，如 0 代表關機、3 代表文字文字模式、5 代表圖形介面模式）是 init 程式用來決定要啟動哪些服務的「狀態代號」。

init 的 PID 永遠是 1，它絕對不會隨著系統切換到 Runlevel 3 或 Runlevel 5 而把自己的行程識別碼改成 3 或 5。

QUESTION 9

Instead of supplying an explicit device in **`/etc/fstab`** for mounting, what other options may be used to identify the intended partition? (Choose two.)

A. LABEL
B. ID
C. FIND
D. NAME
E. UUID

Answer: A, E

A 正確核心原因：可以使用檔案系統標籤來識別分割區

LABEL 是使用者為特定檔案系統自行定義的文字名稱（標籤）。

在 /etc/fstab 中，可以使用格式如 `LABEL=BackupDrive` 來指定要掛載的分割區。

這能避免因為硬碟插槽順序改變導致 /dev/sdX 裝置代號變動的風險。

E 正確核心原因：可以使用通用唯一識別碼來識別分割區

UUID（Universally Unique Identifier）是在檔案系統格式化時，由系統自動產生的 128 位元唯一字串。

在 /etc/fstab 中，這是目前最推薦且標準的指定方式，格式如 `UUID=a1b2c3d4-e5f6-...`。

由於 UUID 具有絕對的唯一性，不論硬碟如何更換介面、代號或順序，系統都能精準鎖定並正確掛載。

錯誤選項詳細除錯

B. 錯誤：fstab 設定檔中不支援直接使用 ID 語法

雖然在 Linux 的 `/dev/disk/by-id/` 路徑下存在依據硬體序號產生的識別檔，但在 `/etc/fstab` 的標準語法中，**不支援**直接寫成 `ID=...` 的格式。

C. 錯誤：Linux fstab 語法中根本不存在 FIND 關鍵字

`FIND=...` 是虛構的選項，檔案系統掛載設定中並沒有這種用來搜尋或動態查找分割區的參數。

D. 錯誤：fstab 設定檔中不支援直接使用 NAME 語法

雖然 GPT 分割區表格式支援 PARTLABEL（分割區名稱），且系統中存在 `/dev/disk/by-partlabel/`，但在 `/etc/fstab` 的第一欄位中，**不支援**直接寫成 `NAME=...`。

QUESTION 10

How many fields are in a syntactically correct line of **`/etc/fstab`**?

A. 3
B. 4
C. 5
D. 6
E. 7

Answer: D

D 正確核心原因：標準的 /etc/fstab 每一行必須包含 6 個欄位

在 Linux 系統中，一個語法正確的 `/etc/fstab` 設定檔行，必須由空白或 Tab 鍵分隔成 6 個固定的欄位。

這 6 個欄位依序代表的意義如下：

1. **第一欄：儲存裝置（Device）**
   可以用 UUID、LABEL（標籤）或傳統的設備路徑（如 /dev/sda1），指定要掛載的來源。

2. **第二欄：掛載點（Mount Point）**
   指定該裝置要在 Linux 目錄樹中的哪個路徑被掛載（例如 /、/home、/mnt/data）。

3. **第三欄：檔案系統類型（File System Type）**
   指定該分割區的格式類型（例如 ext4、xfs、vfat、btrfs、swap）。

4. **第四欄：掛載選項（Mount Options）**
   設定掛載時的行為與權限（例如 defaults、ro、rw、noexec），多個選項之間用逗號隔開。

5. **第五欄：傾印檢查（Dump Request）**
   設定是否要讓 dump 指令進行備份備份。通常設定為 0（不備份）或 1（需要備份）。

6. **第六欄：檔案系統檢查順序（FSCK Order）**
   設定系統開機時，fsck 指令檢查檔案系統損壞的順序。0 代表不檢查，1 代表根目錄（/）優先檢查，2 代表其他需要檢查的分割區。

錯誤選項詳細除錯

A、B、C、E 皆為錯誤數值。如果少於或多於 6 個欄位，系統在開機解析或執行 'mount –a' 時將會發生語法錯誤，進而導致掛載失敗或開機卡死。

QUESTION 11
Which of the following Linux filesystems preallocate a fixed number of inodes when creating a new filesystem instead of generating them as needed? (Choose two.)

A. JFS
B. ext3
C. XFS
D. ext2
E. procfs

Answer: B, D

B 正確核心原因：ext3 採用靜態 Inode 配置架構

- ext3 檔案系統承襲了傳統 UNIX 檔案系統的設計。
- 在執行 mkfs 建立檔案系統時，系統就會根據磁碟大小，預先計算並規劃出一個「固定數量」的 Inode 區塊。
- 建立完成後，Inode 的總數就固定下來，無法隨檔案增加而動態擴充。
- 因此，若大量小檔案耗盡了預先配額的 Inode，即使磁碟仍有剩餘空間，也無法再建立新檔案。

D 正確核心原因：ext2 採用靜態 Inode 配置架構

- ext2 作為 Linux 最早期的標準實體檔案系統，其底層設計結構與後來的 ext3 幾乎完全一致。
- 它同樣是在格式化磁碟（make/creation time）的當下，就已經固定了整個分割區的 Inode 數量上限。
- 不支援在後續運作中根據檔案增長的需求來動態產生新的 Inode。

錯誤選項詳細除錯

A. 錯誤：JFS 採用動態 Inode 配置

- IBM 開發的 JFS（Journaled File System）檔案系統具備動態 Inode 分配（Dynamic Inode Allocation）功能。
- 它不需要在格式化時預先保留固定空間給 Inode，而是隨着新檔案的建立，視需要動態地在磁碟中產生並配置 Inode，因此不會發生「硬碟空間還夠卻把 Inode 用完」的現象。

C. 錯誤：XFS 採用動態 Inode 配置

- SGI 開發的 XFS 檔案系統同樣採用動態 Inode 機制。
- 它會將磁碟劃分為多個分配組（Allocation Groups），在這些組別內根據檔案數量的成長隨時動態地產生與回收 Inode。
- 使用者只會在整個磁碟空間被填滿時，才會無法建立檔案。

E. 錯誤：procfs 是虛擬記憶體檔案系統，不涉及傳統磁碟的 Inode 預配

- procfs（即掛載在 /proc 的檔案系統）是一個暫時存在於記憶體中的虛擬檔案系統（Virtual Filesystem）。
- 它裡面的所有「檔案」與「目錄」都是 Linux 核心在使用者存取的當下，即時、動態從記憶體中虛擬產生出來的，根本不需要、也不可能在磁碟上預先格式化或配置任何固定數量的實體 Inode。

QUESTION 12

What information can the **`lspci`** command display about the system hardware? (Choose three.)

A. System battery type

B. Device IRQ settings

C. PCI bus speed

D. Ethernet MAC address

E. Device vendor identification

Answer: B, C, E

B 正確核心原因：可以顯示裝置的 IRQ 設定

當使用詳細檢視參數（例如 lspci -v 或 lspci -vv）時，指令會從 PCI 配置空間（Configuration Space）中讀取系統分配給該硬體裝置的分配中斷請求（IRQ，Interrupt Request）編號。

這對於排查硬體衝突或硬體中斷設定非常有用。

C 正確核心原因：可以顯示 PCI 匯流排速度

透過進階的詳細模式（如 lspci -vv），該指令能夠解析並輸出特定 PCI / PCIe 裝置當前運作的匯流排速度與頻寬。

例如會顯示裝置運作在 PCIe 3.0 或 4.0 速率、以及使用的是 x1、x4 還是 x16 通道（LnkSta: Speed 8GT/s, Width x16）。

E. 正確核心原因：可以顯示裝置廠商識別資訊

這是 lspci 的核心本質功能。

預設執行 lspci 時，它會去讀取硬體內建的廠商識別碼（Vendor ID）與裝置識別碼（Device ID）。

隨後透過對照系統內建的資料庫（通常是 pci.ids 檔案），轉換成直觀的廠商名稱（例如 Intel Corporation, NVIDIA Corporation, Realtek Semiconductor 等）。

錯誤選項詳細除錯

A. 錯誤：系統電池型態不屬於 PCI 匯流排裝置資訊

- 筆記型電腦的系統電池與電源管理資訊通常是透過 ACPI 介面進行通訊。
- 要查詢電池型態、健康度與容量，應該使用 upower、acpi 指令，或是檢查 /sys/class/power_supply/ 目錄，lspci 偵測不到非 PCI 的電池硬體。

D. 錯誤：lspci 不會直接顯示乙太網路的實體 MAC 位址

- 雖然 'lspci' 可以偵測並列出網路卡硬體本身（例如 Ethernet controller: Realtek...），但它只負責顯示 PCI 介面層級的硬體資訊與暫存器狀態，並不會去讀取或解析網路卡內建的網路層 MAC 位址（Hardware Address）。
- 要查看 MAC 位址，必須使用 ip link、ip addr 或 ifconfig 等網路管理指令。

QUESTION 114
What directory contains configuration files for additional yum repositories? (Specify the **full path** to the directory.)

Answer: **`/etc/yum.repos.d/`**

該目錄是 Linux（如 RHEL、CentOS）系統中高階套件管理器 YUM/DNF 存放第三方與附加軟體源（Additional repositories）定義檔的核心路徑。該目錄下的檔案均以 .repo 作為副檔名，系統在執行套件查詢或安裝時會自動動態掃描並載入這些檔案中的網路位域配置。

QUESTION 115
Which file from the **`/proc/`** file system contains a list of all currently mounted devices? (Specify the **full name** of the file, including path.)

Answer: **`/proc/mounts`**

該核心虛擬檔案（或等同於現行程序命名空間內的 `/proc/self/mounts`）是由 Linux 核心（Kernel）即時動態維護的虛擬文字檔，裡面精準地記錄了當前全系統所有已經成功掛載（Currently mounted）的實體硬碟設備、虛擬檔案系統、對應的掛載點以及詳細的掛載參數，是高階查詢指令（如不帶參數的 `mount`）在讀取系統狀態時的核心資料來源。

QUESTION 116
Following the Filesystem Hierarchy Standard (FHS), where should binaries that have been compiled by the system administrator be placed in order to be made available to all users on the system? (Specify the **full path** to the directory.)

Answer: **`/usr/local/bin/`**

根據 Filesystem Hierarchy Standard (FHS)（檔案系統階層標準）的嚴格定義，由系統管理員在本機上手動編譯、安裝且要開放給全系統所有一般使用者（all users）使用的本地端二進位執行檔（Binaries），必須精準放置於此目錄下。這能確保本機自訂軟體與發行版官方套件管理器所維護的標準目錄（如 /usr/bin/）安全隔離，避免在系統升級時被覆蓋抹除。

QUESTION 117
Which command is used to change the priority of an already running process? (Specify **ONLY** the command without any path or parameters.)

Answer: **`renice`**

該指令（Renice）是 Linux 系統中專門用來針對一個「已經在執行中（already running）」的行程動態修改其 CPU 排程優先權（Nice Level）的核心指令。管理員可以透過傳入行程識別碼（PID）、使用者名稱（User）或群組（Group）來即時調校其執行權重。