Chapter 1

a.

當 CPU 想要與設備進行數據傳輸時，它會將數據的傳輸請求發送給 DMA 控制器。然後，DMA 控制器將直接與該設備通訊，而不需要 CPU 的參與。

b.

當 DMA 控制器完成數據傳輸時，它會向 CPU 發送一個中斷信號，通知 CPU 數據傳輸已經完成。

c.

是的，DMA 控制器在傳輸數據時，CPU 是允許執行其他程序的。然而，這可能會導致一些干擾，特別是當 DMA controller 訪問和修改了與用戶程序相同的記憶體區域。這可能會導致用戶程序的執行時間延遲，或者一致性問題。

Chapter 2

a.

- shared memory：
  - 優點：
    - 效率高：shared memory通常比Message Passing更高效，因為行程可以直接訪問共享的記憶體區域，而無需進行額外的數據複製。
    - 方便快捷：行程之間可以直接通過共享的記憶體區域進行通訊，這使得實現起來比較簡單且快速。
    -
  - 缺點：
    - 安全性較差：由於多個行程共享同一塊記憶體區域，可能會出現競爭條件和死鎖等問題，需要使用同步機制來確保安全性。
    - 難以跨越不同裝置：shared memory通常僅適用於同一台裝置上的行程通訊，跨越不同裝置的情況下則較為困難。
- Message Passing：
  - 優點：
    - 安全性較高：每個行程擁有自己的記憶體空間，通訊通常是通過消息傳遞，這降低了競爭條件和死鎖的發生概率。

        - 跨平台性：Message Passing通常可以在不同裝置之間進行通訊，因此更適合於分布式系統。
    - 缺點：
        - 效率較低：消息傳遞通常涉及數據的複製和通訊開銷，因此在效率上可能不如共享記憶體模型。
        - 實現複雜：Message Passing需要實現消息的發送、接收、排隊和管理等功能，因此實現起來相對複雜一些。

b.

- 模組化：微內核將操作系統的基本功能劃分為核心功能和外部功能模塊，使得系統結構更加清晰和易於擴展。這使得系統更容易維護和修改，並且使得定制化更加容易。

- 靈活性：微內核使得系統服務可以以獨立的行程運行，這使得系統服務之間的隔離更加明確，並且可以提高系統的可靠性和安全性。此外，微內核還可以支持動態加載和卸載模塊，使得系統的運行時行為更加靈活。

Chapter 3

a.

1. 保存當前行程的context：當內核決定要切換到另一個行程時，它首先需要保存當前行程的context，包括 CPU 的狀態（例如寄存器內容）、堆棧指針以及其他相關的行程狀態信息。這樣可以確保當再次切換回該行程時，能夠恢復到正確的執行狀態。

2. 調度新的行程：內核根據調度算法選擇下一個要執行的行程，通常是根據優先級、時間片等信息來進行選擇。一旦選擇了下一個行程，內核就會準備好切換到該行程。

3. 恢復新行程的context：一旦新行程被選中，內核將從保存的該行程的context中恢復相應的 CPU 狀態、堆棧指針等信息。這樣，CPU 就可以從新行程的狀態繼續執行。

4. 更新行程狀態：在完成切換之後，內核需要更新相關的PCB和調度信息，以反映當前行程的狀態和位置。這包括更新行程的就緒狀態、時間片等。

b.

- named pipes :

    - 用於監聽來自其他行程的請求（類似於 TCP IP 端口）。如果調用行程知道該名稱，就可以向其發送請求

- ordinary pipes :
    - 用於只需在事先已知的兩個指定行程之間進行通訊的情況。在這種情況下，named pipes的開銷太大。