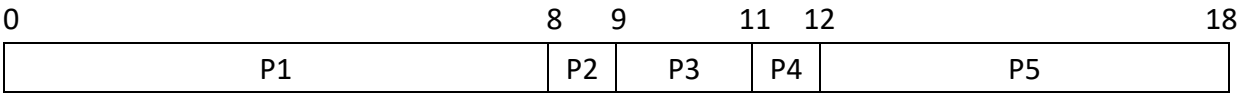
1. memory-mapped I/O 是一種 CPU 對 I/O 設備的通信方法，將 I/O 的 port 或 memory 映射到記憶體位址上（I/O 和 memory 共用記憶體空間），不需要特殊指令即可處理 I/O，但 I/O 占用的記憶體位址就不能當作記憶體使用。

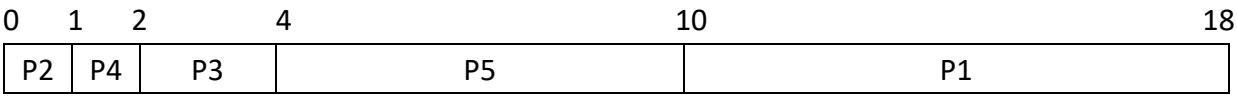
2. DMA 是一種記憶體存取技術， DMA 控制器允許 I/O 設備直接讀寫記憶體而不需要經過 CPU。DMA 的流程：
   (1) CPU 告訴 DMA 控制器要轉移的資料、要轉移到哪裡、資料的長度，並傳一個指令給硬體控制器，從 I/O 設備讀取資料並確認是否正確讀入（正確讀入的話才會進入 DMA）
   (2) DMA 透過 bus 向硬體控制器傳送讀取要求（附上轉移的目的地位址，bus 會將資料寫入該位址）
   (3) 如果 bus 寫入成功，硬體控制器透過 bus 發送 ACK 訊號給 DMA 控制器
   (4) 重複以上步驟直到資料全數轉移，DMA 終止

3. (a)

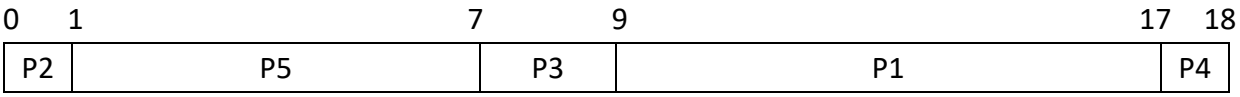
FCFS:



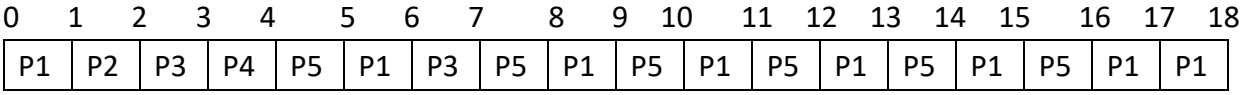
SJF:



priority:



RR:



(b) turnaround time

	FCFS	SJF	priority	RR
P1	8	18	17	18
P2	9	1	1	2
P3	11	4	9	7
P4	12	2	18	4
P5	18	10	7	16
AVG	11.6	7	10.4	9.4

(c) waiting time

	FCFS	SJF	priority	RR
P1	0	10	9	9
P2	8	0	0	1
P3	9	2	7	5
P4	11	1	17	3
P5	12	4	1	10
AVG	8	3.4	6.8	5.6

(d) 根據 3.(c)的結果，SJF 可以使程序的平均等待時間最小。

4. subroutine 像是副函式，執行到呼叫副函式的地方時，主程式會先暫停並執行副函式，副函式執行完就會跳回主程式；coroutine 像是 thread，允許函式執行到一半就中斷（此時的內部狀態會被保留），呼叫端可以隨時恢復這個函式。
寫程式時我們通常是在 user part，需要執行高權限的指令時才會進入 kernel part 去呼叫 system call，執行完該指令就會回到 user part，比較像 subroutine。