	Mô tả	Ưu điểm	Nhược điểm
FCFS	Cơ chế thực thi : Tiến trình nào yêu cầu CPU trước sẽ được cấp phát trước. Tiến trình sẽ thực thi đến khi kết thúc hoặc bị blocked do I/O. Chế độ quyết định : Non-Preemptive. Hiện thực : Sử dụng hàng đợi FIFO. Tiến trình đi vào được thêm vào cuối hàng đợi. Tiến trình được lựa chọn để xử lý được lấy từ đầu của queue.	Sẽ không bị starvation. Thuật toán này dễ cài đặt. Code đơn giản.	Thời gian chờ trung bình của FCFS thường khá dài (VD : Một process có burst-time rất dài đến trước, khi đó các process có burst-time nhỏ sẽ phải chờ 1 khoảng thời gian rất lâu mới đến lượt thực thi). Lãng phí thời gian do thời gian phần cứng trống khá nhiều (convoy effect). Non-preemptive. Sẽ không hoạt động tốt trong các hệ thống chia sẻ thời gian (time-sharing system) khi các user đều mong muốn được sử dụng CPU trong một khoảng thời gian và không muốn delay quá lâu.
SJF	Cơ chế thực thi: Định thời công việc ngắn nhất trước (Burst-time nhỏ nhất). Khi CPU được tự do, nó sẽ cấp phát cho tiến trình nào yêu cầu ít thời gian nhất để kết thúc (burst-time nhỏ nhất). Burst-time có được từ việc dự đoán, dựa vào các lần chạy trước của tiến trình. Nếu có 2 tiến trình cùng Burst-time, tiến trình nào vào hàng đợi trước sẽ được chạy trước (không xét độ ưu tiên). Chế độ quyết định: Non-Preemptive.	Tối ưu cho thời gian chờ đợi trung bình tối thiểu với một tập tiến trình cho trước.	Cần phải ước lượng thời gian cần CPU tiếp theo của process (Burst time). Có thể xảy ra starvation nếu số lượng process có burst time nhỏ cần được thực thi quá nhiều.
SRTF	Cơ chế thực thi : (Tương tự SJF). Nếu một tiến trình mới được đưa vào danh sách với chiều dài sử dụng CPU cho lần tiếp theo nhỏ hơn (lưu ý, chỉ nhỏ hơn, nếu burst-time bằng	Preemptive. Thời gian đáp ứng nhanh cho các tác vụ nhỏ. Tránh việc một tác vụ lớn độc chiếm CPU.	(Các nhược điểm của SJF). Tăng thời gian hoàn thành trung bình.

	thì không preempt) thời gian còn lại của tiến trình đang xử lý, nó sẽ dừng hoạt động tiến trình hiện hành (preempt). Chế độ quyết định : Preemptive.	Thời gian chờ đợi trung bình thường sẽ nhỏ hơn SJF	
Priority Scheduling	Cơ chế hoạt động : Mỗi tiến trình sẽ được gán 1 độ ưu tiên. CPU sẽ được cấp cho tiến trình có độ ưu tiên cao nhất. Định thời sử dụng độ ưu tiên có thể là : Preemptive : Khi một tiến trình mới xuất hiện có độ ưu tiên cao hơn, nó sẽ preempt tiến trình đang chạy. Non-Preemptive : Tiến trình đang chạy sẽ tiếp tục chạy. Nếu có 2 tiến trình cùng độ ưu tiên, thì tiến trình nào đến trước sẽ được chạy trước. Burst-time không được áp dụng để so sánh ở đây. Chế độ quyết định : Non-Preemptive hoặc Preemptive.	Các tác vụ quan trọng sẽ được thực thi trước.	Có thể xảy ra starvation : Các process có độ ưu tiên thấp có thể không bao giờ được thực thi (giải pháp : aging – Độ ưu tiên của process sẽ tăng theo thời gian).

Round Robin	Cơ chế hoạt động : Mỗi tiến trình nhận được một đơn vị nhỏ thời gian CPU (time-slice, quantum time), thông thường từ 10-100msec để thực thi. CPU Schedulers sẽ chọn 1 tiến trình từ ready queue và "lên dây cót" một quantum cho tiến trình, sau đó cho tiến trình chạy. Lúc này, sẽ có 2 khả năng có thể xảy ra : Thời gian chạy > Quantum : Khi đó, tiến trình sẽ bị interrupt và CPU Schedulers sẽ chọn tiếp tiến trình tiếp theo. Thời gian chạy < Quantum : Tiến trình tiếp theo sẽ ngay lập tức được thực thi tiếp (không cần chờ hết quantum time của tiến trình trước), và tiến trình tiếp theo đó cũng được gán 1 quantum time. Phụ thuộc nhiều vào quantum time : Quantum time ngắn thì đáp ứng nhanh, tuy nhiên overhead lớn do chuyển ngữ cảnh nhiều.	Thời gian đáp ứng trung bình thường thấp -> Thích hợp cho các hệ thống time-sharing. Không xảy ra tình trạng starvation.	Thời gian chờ đợi trung bình thường khá lớn. Chuyển ngữ cảnh nhiều -> Hao phí cao. Hiệu suất thuật toán phụ thuộc nhiều vào việc chọn quantum time. Không thể sử dụng thuật toán nếu muốn các ứng dụng có độ ưu tiên khác nhau.

	Quantum time phải > thời gian chuyển ngữ cảnh (context switch). Quantum time dài thì đáp ứng chậm, tuy nhiên thông lượng (throughput) sẽ cao. Và khi quantum time quá lớn RR->FCFS (Quantum time lớn -> Không bao giờ bị ngắt -> Ai vào trước làm trước -> FCFS). Khi cả tiến trình vừa thực thi xong và tiến trình mới cũng arrive vào cùng một thời điểm, thì tiến trình mới sẽ vào hàng đợi trước rồi mới đến tiến trình cũ. Các tiến trình đều có độ ưu tiên giống nhau. Chế độ quyết định : Preemptive.		
HRRN	Cơ chế hoạt động : Chọn process tiếp có giá trị RR (Response Ratio) lớn nhất. Các process ngắn được ưu tiên hơn vì service time (hay burst time) nhỏ.	Không xảy ra starvation. Tự động cân bằng giữa việc ưu tiên một tiến trình có thời gian thực thi nhỏ và một tiến trình đã ở quá lâu trong hệ thống (aging).	Non-Preemptive.

Multilevel Queue Scheduling	Cơ chế hoạt động : Hàng đợi ready được chia thành nhiều hàng đợi riêng biệt theo một số tiêu chuẩn như : Đặc điểm và yêu cầu định thời của process. Foreground (interactive) và background process. Process được gán cố định vào một hàng đợi, mỗi hàng đợi sẽ sử dụng một giải thuật riêng. Có 2TH hệ điều hành định thời cho các hàng đợi : Có một độ ưu tiên cố định cho từng hàng đợi (fixed priority scheduling). Hàng đợi có độ ưu tiên cao hơn phải được chạy xong (empty) trước khi hàng đợi có độ ưu tiên thấp hơn được phép chạy. Nếu có 1 tiến trình đi vào hàng đợi có độ ưu tiên cao hơn trong khi hàng đợi có độ ưu tiên thấp hơn đang được thực thi, hàng đợi có độ ưu tiên thấp hơn đó sẽ bị preempt.	Áp dụng nhiều giải thuật định thời cho nhiều loại tiến trình có độ ưu tiên khác nhau. Cho phép các CPU-Bound process được ưu tiên hơn trong việc thực thi -> Thời gian hệ thống thực thi tác vụ được cải thiện. Có thể hoạt động trong cả 2 chế độ : Preemptive và Non-Preemptive.	Các hàng đợi đa cấp này cần được giám sát -> Hao phí tài nguyên hệ thống. Process không thể di chuyển từ hàng đợi này sang hàng đợi khác -> Không linh động.

	Time-slice : Mỗi hàng đợi nhận được một khoảng thời gian chiếm CPU và phân phối cho các process trong hàng đợi khoảng thời gian đó. Chế độ quyết định : Non-Preemptive hoặc Preemptive.		
Multilevel Feedback Queue	Cơ chế hoạt động : (Tương tự Multilevel Feedback Queue). Điểm khác biệt : Cho phép process nhảy từ queue này đến queue khác.	Thích nghi với các tiến trình. VD : Một tiến trình nếu sử dụng quá nhiều CPU time thì sẽ xếp nó vào queue có độ ưu tiên thấp hơn. Aging. VD : Một process đã xuất hiện lâu mà không được thực thi, sẽ được đưa lên 1 queue có độ ưu tiên cao hơn. Thuật toán chung nhất, có thể được thiết kế để phù hợp với các hệ thống khác biệt.	Tốn tài nguyên hệ thống để duy trì các queue -> Có thể không thích hợp đối với các hệ thống nhỏ. Thiết kế phức tạp.