خوارزميات الجدولة Scheduling Algorithms :

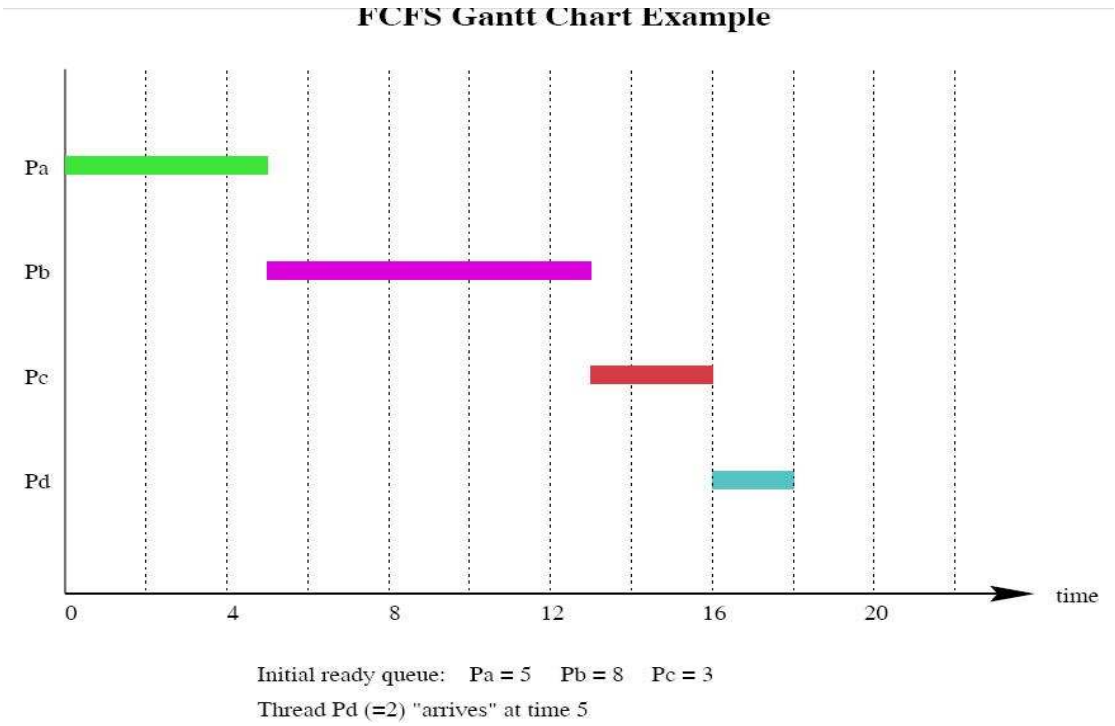
من يأتي أولاً يخدم أولاً First-come First-served (FCFS-FIFO) :

فلسفة هذه الطريقة تعتمد على من يصل أولاً من العمليات هو من يدخل أولاً إلى وحدة المعالجة المركزية (CPU) ،وهي تعتبر غير قابلة للإجهاض (Non-preemptive) أي أن هذه العملية لا تخرج من وحدة المعالجة المركزية إلا بعد انتهائها من التنفيذ ولا يتدخل لب النظام (kernel) في إجهاض العملية .

خصائص الخوارزمية :

1. أبسط خوارزميات جدولة المهام على الإطلاق .
2. معدل وقت الانتظار فيها ليس بالضرورة أن يكون الأقصر .
3. الأداء (performance) هنا يعيبه عدم الاستغلال الأمثل للمعالج وهذا سببه (convoy effect)ونعني به أن هناك عمليات قصيرة ويمكن إنجازها بسرعة ولكنها رغم ذلك تضطر للانتظار بسبب وجود عمليات أطول منها في طور التنفيذ .
4. يعتبر غير ملائم أبداً للاستخدام في الأنظمة التفاعلية (interactive system) وهذا العيب ناشئ وبشكل واضح عن كون هذه الخوارزمية غير قابلة للإجهاض (non preemptive scheduling)

مثال 1:



مثال 2:

(Process) العملية	*(Burst time) وقت التنفيذ*
P1	24
P2	3
P3	3

زمن الوصول لجميع العمليات هو t0 و ترتيب وصول العمليات هو P1,P2,P3

Gantt chart:

0-23	24-26	27-30
P1	P2	P3

معدل وقت الانتظار(Average waiting time) = (0+24+27)/3= 17

لو كان ترتيب وصول العمليات P2,P3,P1 :

0-2	3-5	6-30
P2	P3	P1

معدل وقت الانتظار(Average waiting time) = (0+3+6)/3= 3

العملية الأقصر أولاً Shortest-Job-First (SJR) :

تأتي كل عملية مصاحبة للوقت الذي تحتاجه للتنفيذ ويتم اختيار العملية ذات الوقت القصير.

وينقسم إلى قسمان:

1. **غير القابلة للإجهاض (Non-preemptive)** : يتم اختيار العملية ذات الوقت الأقصر من مصفوفة الانتظار و لا تخرج من وحده المعالجة المركزية إلا بعد انتهاء وقت تنفيذها.

2. **القابلة للإجهاض (Preemptive)**: أي انه عند وصول عملية جديدة ووقت تنفيذها اقصر من الوقت المتبقي لتنفيذ العملية الحالية فيتم إجهاض العملية الحالية وإدخال العملية الجديدة إلى وحدة المعالجة المركزية، تعتبر هذه الطريقة الأمثل لأنها تعطي اقل قيمة لمتوسط وقت الانتظار لمجموعة من العمليات ولكن لا تعطي تقديرات دقيقة للوقت الذي تنفذ فيه العملية .

كيف يتنبأ بوقت العملية قبل حدوثها؟

يحاول الجدول الزمني التنبؤ بوقت تنفيذ العملية استنادا إلى تاريخ تنفيذ العملية السابق وإذا كانت العملية جديدة فيسجل أول وقت لها في التنفيذ .

ويستخدم هذه المعادلة لتحديد زمن تنفيذ العملية:

$$t(n+1) = w * t(n) + (1 - w) * T(n)$$

حيث:

t(n+1) وقت العملية القادم

t(n) وقت العملية الحالي

T(n) متوسط وقت العمليات السابقة

W وقت الانتظار وغالبا ما يتراوح قيمته بين 0 <= w <= 10

أولوية الجدولة:

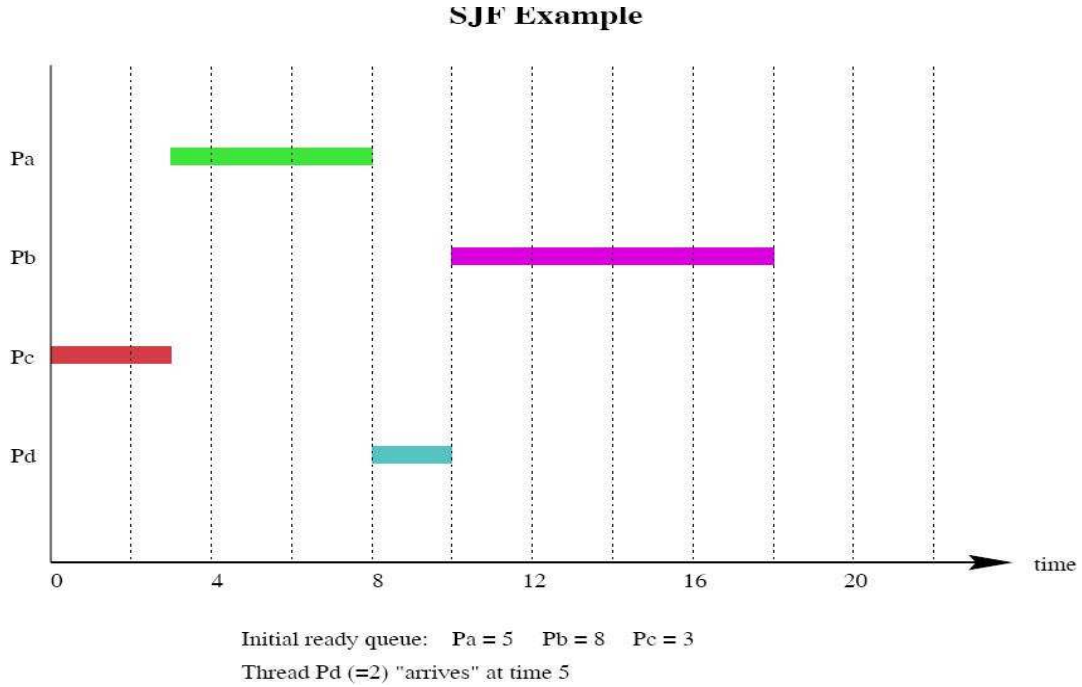
الأولوية هي عدد صحيح يرتبط مع كل عملية وإذا كان العدد مرتفع فان الأولوية تكون منخفضة والعكس صحيح (الأعداد الصغيرة تكون مرتفعة الأولوية في اليونيكس ولكن منخفضة في الجافا) .

ولكن هناك مشكلة :وهي إن العمليات ذات الأولوية المنخفضة لا تنفذ أبدا (مجاعة starvation) ، والحل: إن العمليات كلما زاد وقتها في الانتظار كلما زادت أولويتها حتى لا تهمل مع الوقت ولا يتم تنفيذها.

خصائص الخوارزمية :

1. صعبة البرمجة .
2. الخوارزمية التي تعطي أقل معدل انتظار على الإطلاق.

مثال 1:



مثال 2 (غير القابلة للإجهاض (Non-preemptive)):

(Process) العملية	(Burst time) وقت التنفيذ
P1	6
P2	8
P3	7
P4	3

Gantt chart:

0-2	3-8	9-15	16-24
P4	P1	P3	P2

معدل وقت الانتظار(Average waiting time) = (0+3+9+16)/4= 7

مثال 3 (القابلة للإجهاض (preemptive)):

العمليـــــــــــــة (Process)	زمن الوصول (Arrival time)	وقـــت التنفيـــذ (Burst time)
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Gantt chart:

0	1-4	5-9	10-16	17-25
P1	P2	P4	P1	P3

معدل وقت الانتظار(Average waiting time) = ((10-1)+(1-1)+(17-2)) / 4 = 6.5

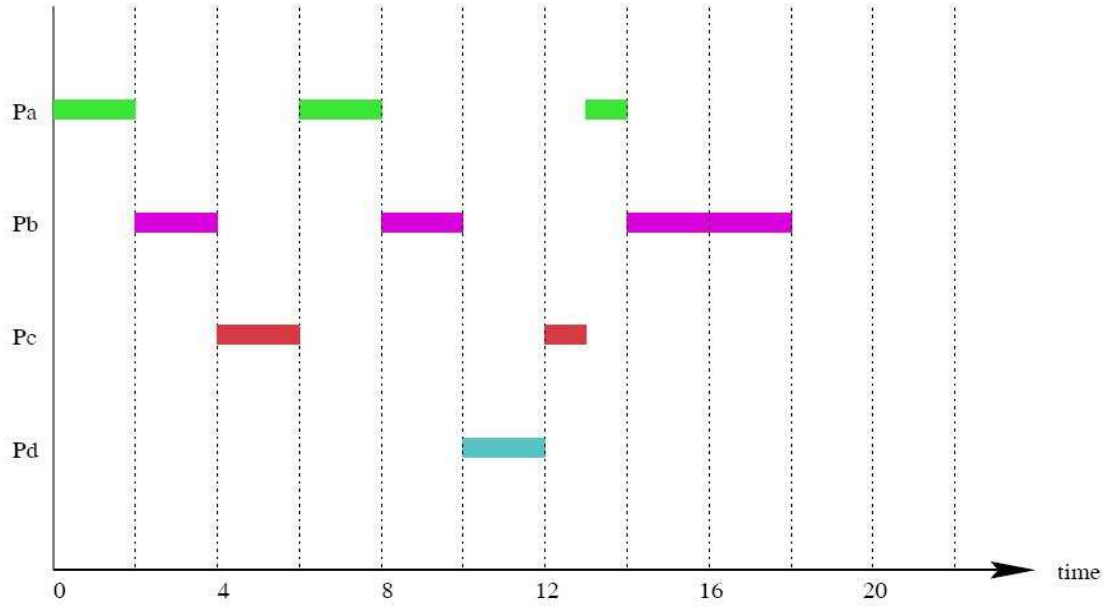
راوند روبن **Round Robin (RR)** :

- هذه الخاصية تعطي لكل عملية وقت محدد من الـــزمن للتنفيـــذ داخـــل وحـــده المعالجـــة المركزية (CPU) ويسمى هذا الوقت (time quantum).
- يتراوح الوقت المحدد (time quantum) بـــين 10- 100 milliseconds وبعد انتهاء هذا الوقت تجهض العملية وتدخل في نهاية مصفوفة الانتظار .
- إذا كان هناك عدد من العمليـــات في مـــصفوفة الانتظـــار والوقـــت المحـــدد (time quantum) هو q فان كل عملية تأخذ 1\عدد العمليات
- لا يوجد عملية تأخذ أكثر من متوسط الوقت (1 – عدد العمليات) * الوقت المحدد.
- هذه الخاصية تشبه القادم أولاً يخدم أولاً (FCFS) ولكن تختلف عنها إن هنـــاك وقـــت محدد لتنفيذ العملية داخل وحدة المعالجة المركزية (CPU) .

ملاحظات:

- إذا كانت قيمة وقت تنفيذ العملية اقل من الوقت المحدد فانه يأخذ وقت العملية
- في حالة إذا ضاعفنا الوقت المحدد (time quantum) فإننا سنـــصل إلى خوارزميـــة القادم أولاً يخدم أولاً (FCFS)
- وإذا قللنا الوقت المحدد (time quantum) فسيضيع وقت وحده المعالجة المركزية في التحويل (context switch)

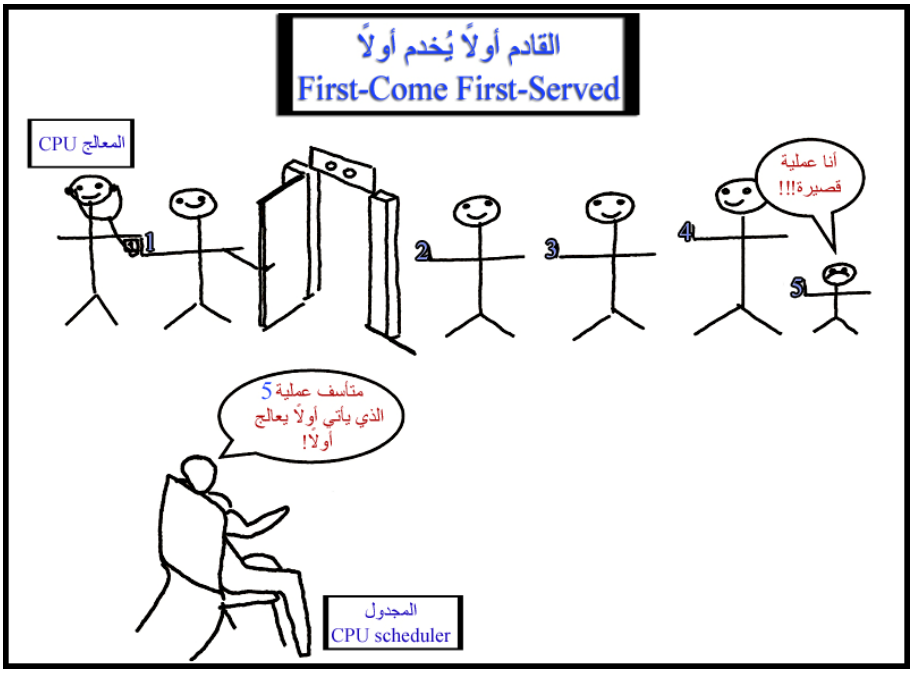
مثال1:

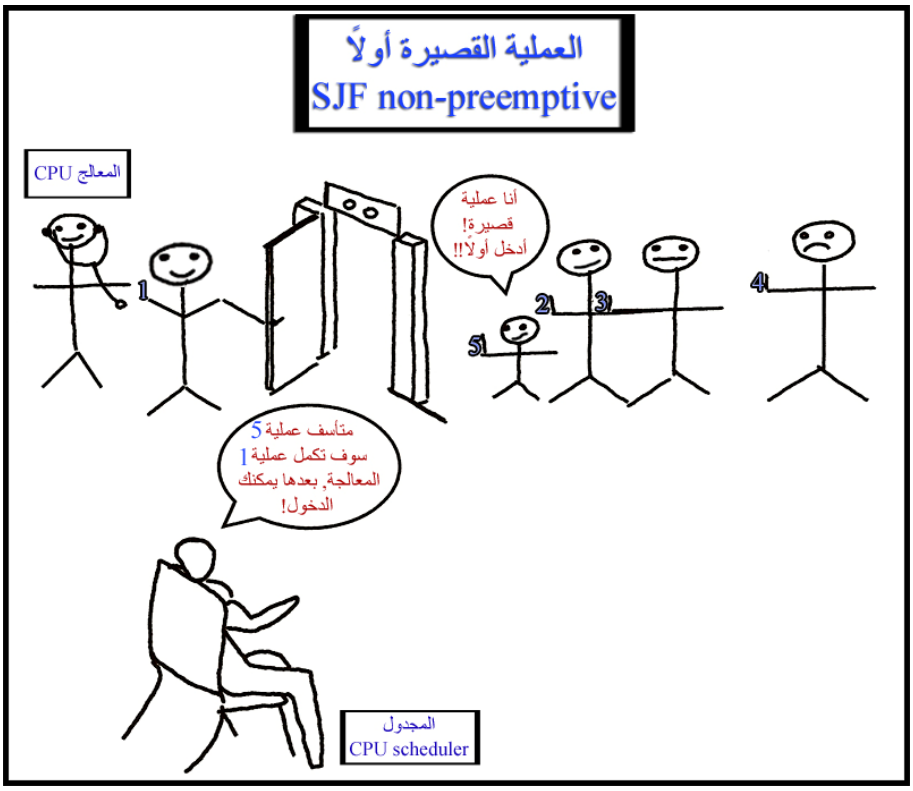


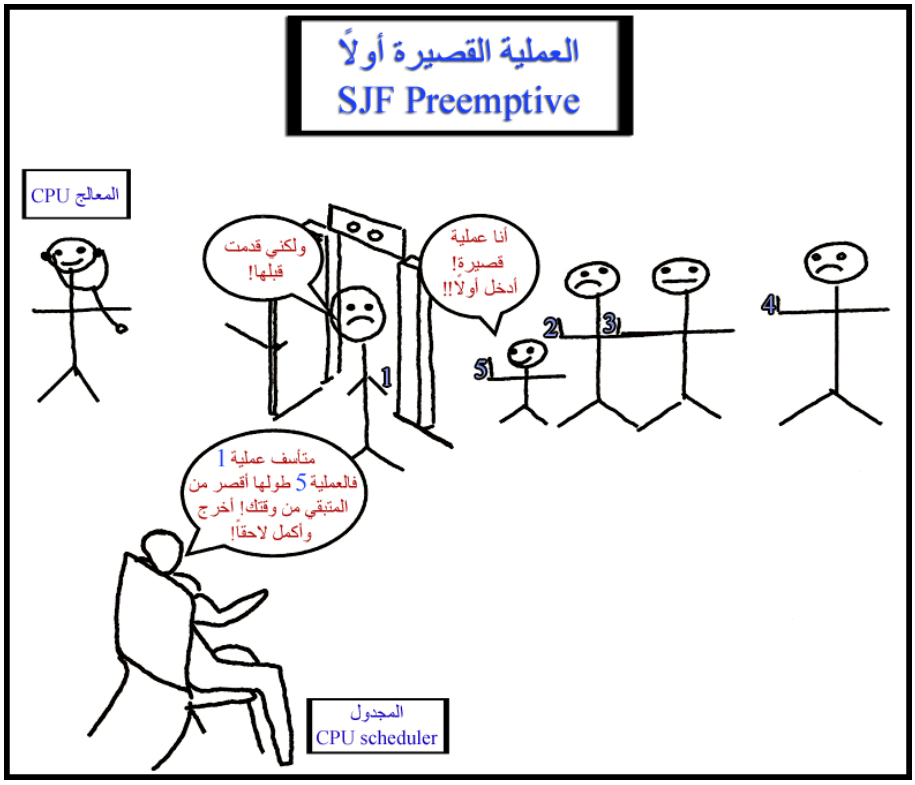
Initial ready queue: Pa = 5 Pb = 8 Pc = 3 Quantum = 2

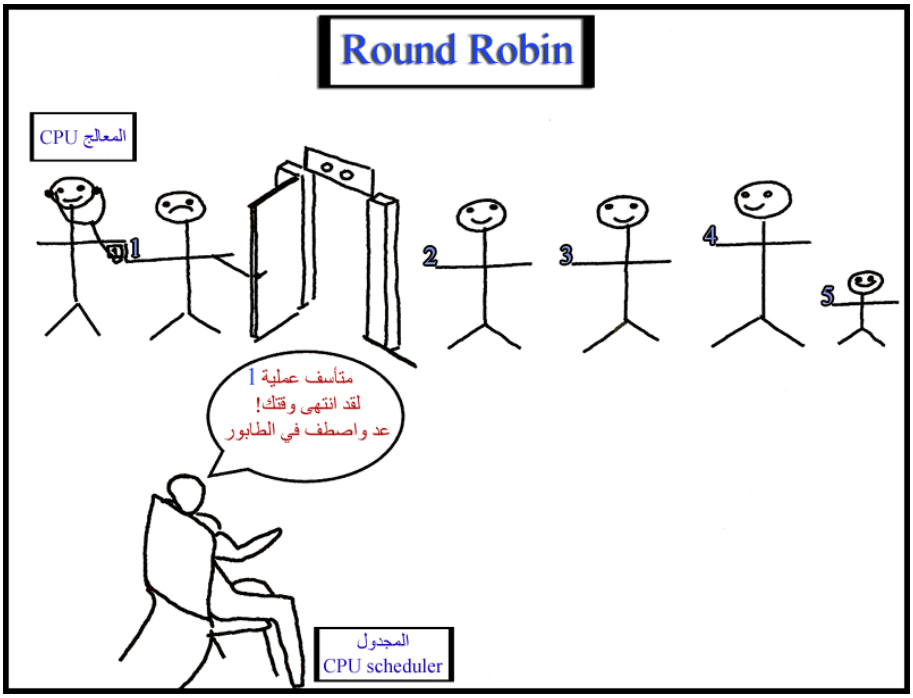
Thread Pd (=2) "arrives" at time 5

رسومات تبين طريقة كل خوارزمية:




العملية القصيرة أولاً
SJF non-preemptive
CPU المعالج
أنا عملية
قصيرة!
أدخل أولاً!!
1
2
3
4
5
متأسف عملية 5
سوف تكمل عملية 1
المعالجة, بعدها يمكنك
الدخول!
المجدول
CPU scheduler


العملية القصيرة أولاً
SJF Preemptive
CPU المعالج
ولكني قدمت قبلها!
أنا عملية قصيرة! أدخل أولاً!!
1
2
3
4
5
متأسف عملية 1 فالعملية 5 طولها أقصر من المتبقي من وقتك! أخرج وأكمل لاحقاً!
المجدول
CPU scheduler


Round Robin
CPU المعالج
1
2
3
4
5
متأسف عملية 1 لقد انتهى وقتك! عد واصطف في الطابور
المجدول
CPU scheduler