

## CPU Scheduling

إن جدولة وحدة المعالجة (CPU scheduling) هي أساس نظم التشغيل المتعددة البرمجة بالتأني للحصول على حاسوب ذو فعالية عالية لخواص المبرمج  $\rightarrow$  multiprogramming systems

خوارزمية جيدة تستطيع جدولة وحدة المعالجة بشكل يؤمن الهدف من تعدد البرمجة وهو

((استخدام وحدة المعالجة على أعلى حد ممكن maximize CPU Utilization))

نعلم أن :

}}}} في النظم ذات المعالج الواحد فإنه يوجد إجرائية واحدة فقط قيد التنفيذ في لحظة ما

- الجدولة وظيفة أساسية من وظائف نظم التشغيل ، بحيث أن كل الموارد الحاسوبية تخضع للجدولة على اعتبار أن الـ CPU هو مورد أيضاً من هذه الموارد.
- يوجد ما يسمى بالجدول Scheduler الذي يقوم بترتيب تنفيذ الإجراءات في وحدة المعالجة المركزية

## CPU Scheduler

ما إن تصبح وحدة المعالجة في حالة لا عمل idle ( الخمول ) حتى يجري الجدول نظام التشغيل على انتقاء ( اختيار أو انتخاب ) إحدى الإجراءات الموجودة في رتل الباهرة ( ready queue ) ليقوم بتنفيذها

تم عملية الانتقاء هذه بواسطة الجدول قصير الأمد Short-Term Scheduler ( جدول وحدة

المعالجة CPU Scheduler ) بحيث ينتهي إجرائية موجودة في الذاكرة حالتها ready ويقوم بتحصيص وحدة المعالجة لها ( تصبح في الـ CPU تنفذ ).

((عملية الانتقاء هذه تشبه عملية الانتخاب ))

## نظم تشغيل المحاضرة الخامسة (خوارزميات الجدولة 1)

يتم اتخاذ قرار للجدولة في أحد الحالات الأربع التالية :

1\_ عندما تبدل الإجرائية حالتها من **running** (تُنفذ) إلى **waiting** (تنظر)

مثال : طلب إدخال/إخراج I/O request أو انتظار انتهاء إحدى الإجرائيات الأولاد.

2\_ عندما تبدل الإجرائية حالتها من **running** (تُنفذ) إلى **ready** (جاهزة)

مثال : عند حصول طلب مقاطعة interrupt request

3\_ عندما تبدل الإجرائية حالتها من **waiting** (تنظر) إلى **ready** (جاهزة).

مثال : عند انتهاء عملية إدخال / إخراج.

4\_ عندما تنتهي الإجرائية من التنفيذ terminates

مثال : إما أن تنتهي تلقائياً أو أن يقوم نظام التشغيل بإنهائها قسرياً.

ملاحظة :

في الحالتين 2 و 3 فإن الجدول عليه الاختيار أي الإجرائيات سيتم اختيارها لكي تدخل إلى وحدة المعالجة

بما فيها الإجرائية التي أصبحت ready (جاهزة).

← نسمى الجدولة في هذه الحالة جدوله شفعية preemptive

ملاحظة :

في الحالتين 1 و 4 فإن الجدول ليس عليه إلا أن يتلقى إجرائية جديدة من رتل الجاهزية (إذا توافرت أصلاً)

لكي يتم تنفيذها في وحدة المعالجة.

← نسمى هذه الجدولة في هذه الحالة جدوله لا شفعية non-preemptive

الآن أصبح لدينا مفهومين جديدين :

جدولة شفعية ... preemptive ... جدوله لا شفعية non-preemptive

ما الفرق بينهما ???

## نظم تشغيل المحاضرة الخامسة (خوارزميات الجدولة 1)

### preemptive scheduling جدولة شفعة

نقصد بالشفعة (الاستباقية) أي أن الإجرائية لا تريد أن تخرج من وحدة المعالجة ، ولكن نظام التشغيل يخرجها بشكل قسري لإدخال إجرائية جديدة.

(الإجرائية كانت تعمل ولا تريد أن تخرج لكنها أجرت على الخروج)

### non-preemptive scheduling جدولة لا شفعة

وهي غير استباقية بحيث تبقى الإجرائية محتفظة بوحدة المعالجة CPU

إلى أن تطلقها (تحررها) ، إما أن تنتهي terminates

أو أن تنتقل إلى حالة الانتظار waiting

(الإجرائية هي التي خرجت من وحدة المعالجة بطلب منها أي أنه لا يوجد قسر في هذه الحالة)

بالتالي لا تحتاج إلى عتاديات خاصة مثل مؤقت timer .

ملاحظة: في الجدولة الشفعة preemptive يجب الانتباه إلى آلية تنفيذ للبيانات المشتركة shared data

((هذا الملاحظة لن تفهم المقصود منها إلا في اخضرات القادمة)))

## نظم تشغيل المحاضرة الخامسة (خوارزميات الجدولة 1)

### مسند المهام Dispatcher

وهو المنفذ العملي لجدول وحدة المعالجة ، فهو الذي يقوم بتنفيذ القرارات التي قام الجدول CPU Scheduler (جدول الأمد القصير) بالتخاذلها أي هو الذي يسلم الإجرائية لوحدة المعالجة ليقوم بتنفيذها .  
نعتبر أن المجدول هو مغفر الشرطة

ونعتبر أن مسند المهام هو الشرطي الذي يقبض على مجرمين ويسلمهم للعدالة ...



### مهام ال Dispatcher

1\_ تبديل المحتوى switching context

2\_ التبديل إلى نمط المستخدم user mode

3\_ القفز jumping (الانتقال go to) إلى المكان الصحيح لإعادة تنفيذ البرنامج

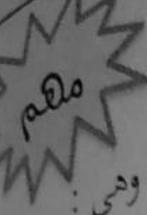
يجب أن نعلم أنه عند كل تبديل بين إجرائية وأخرى يتم استدعاء ال Dispatcher وبالتالي

يجب أن يكون ال Dispatcher سريع جداً

### زمن استجابة مسند المهام Dispatch Latency

هو الوقت اللازم الذي يأخذة ال Dispatcher لإيقاف إجرائية ما والبدء بإجرائية أخرى .

## Scheduling Criteria معايير الجدولة



يوجد العديد من المعايير التي يتم على أساسها مقارنة خوارزميات جدولة ال CPU مع بعضها وهي :

- استخدام وحدة المعالجة (CPU Utilization) : القدرة على إبقاء ال CPU مشغولاً أقصى وقت ممكن وهي تأخذ قيمها بين 0-100 %

- الإنتاجية (Throughput) : وهي تكافئ عدد (#=number) الإجرائيات التي تنهي بشكل كامل تنفيذها في وحدة الزمن .

- الفترة الزمنية الكلية (Turnaround time) : كمية الوقت اللازمة لإنجاز الإجرائية من وقت بدء التنفيذ حتى انتهاء التنفيذ .

$$\text{Turn Around Time (TAT)} = \text{Completion Time} - \text{Arrival Time}$$

- زمن الانتظار (Waiting Time) : كمية الوقت التي قضتها الإجرائية منتظرة في رتل الجاهزية



وهو مجموع مدد الانتظار في رتل الجاهزية .

$$\text{Waiting Time (WT)} = \text{TAT} - \text{Burst Time}$$

- زمن الاستجابة (Response Time) : كمية الوقت اللازم منذ تقديم الإجرائية لطلب ما

submit request first response حتى تم الحصول على أول استجابة

ملاحظة : نقصد بالاستجابة هنا الوقت اللازم لبدء الاستجابة وليس الوقت اللازم لإنتاج تلك الاستجابة .

# معيار زمن الاستجابة Response Time مهم في أنظمة اقتسام الزمن Time-Sharing

# نظم تشغيل المحاضرة الخامسة (خوارزميات الجدولة 1)

هناك العديد من خوارزميات الجدولة ، بحيث أن كل خوارزمية تتمتع بخصائص مختلفة عن الأخرى ...

سؤال : ما هي المعايير المثلث لخوارزمية جدولة وحدة المعالجة ؟

- Max CPU Utilization
- Max Throughput
- Min Turnaround time
- Min Waiting Time
- Min Response Time

MAX  
MIN

مهام

إن المطلوب هو التوازن بين هذه المعايير ... حيث أنه من الصعب وجود خوارزمية تعطي تحسين optimize في المعايير السابقة جميعها ، لكن سنركز في تقسيم الخوارزميات على معدل زمن الانتظار

{Average Waiting Time}

زمن الانتظار الوسطي Average Waiting Time : هو مجموع أزمنة الانتظار للإجراءات

الموجودة في رتل الجاهزية مقسوماً على عددها وغالباً ما يقاس بواحدة الميللي ثانية ms .

مثلاً : لدينا أزمنة الانتظار للإجراءات هي  $p_1 = 2ms, p_2 = 8ms, p_3 = 5ms$  وبالتالي :

$$\text{average} = \frac{2+8+5}{3} = \frac{15}{3} = 5ms$$

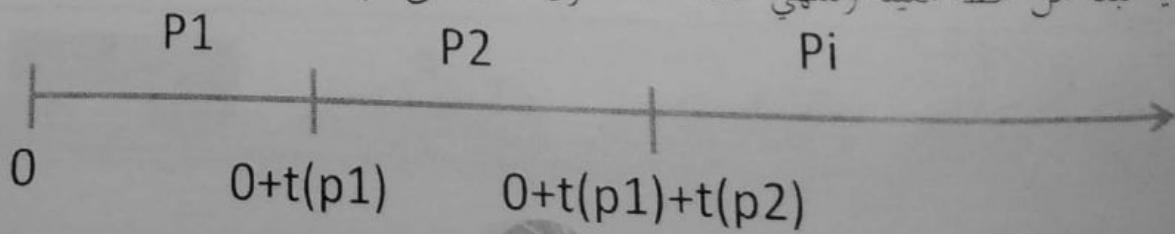
زمن الانتظار الوسطي يساوي 5ms

مخطط غانت لتمثيل جدولة الإجراءات :

اقرخ العالم Henry Gantt طريقة رسومية لتمثيل تسلسل تنفيذ الإجراءات من أجل مقارنة الخوارزميات مع بعضها من أجل عدة إجراءات محددة.

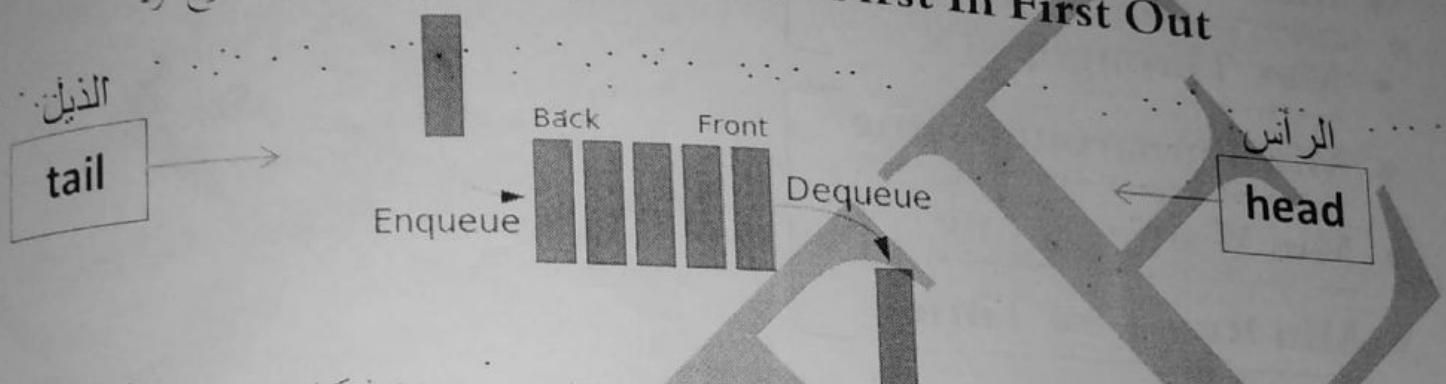
مخطط غانت gantt chart : هو عبارة عن خور زمني مقسم بالميلى ثانية يبدأ من الصفر ، بحيث أن

الإجراء تبدأ من لحظة معينة وتنتهي عند لحظة أخرى لتدل على الوقت الذي تقضيه في ال CPU .



**First-Come First-Served (FCFS)**

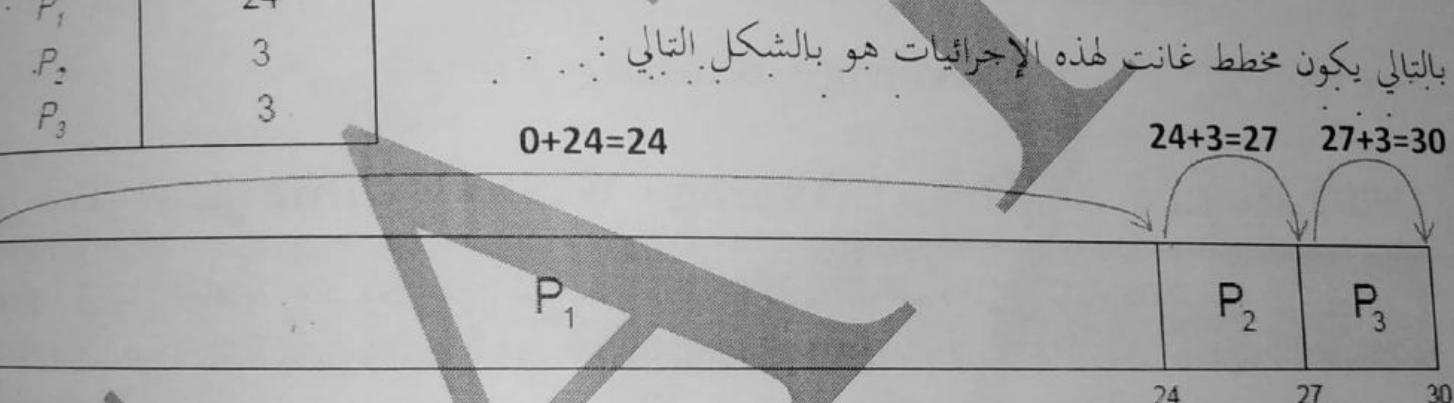
1\_ خوارزمية القادر أولاً يخدم أولاً وهي أبسط خوارزميات الجدولة على الإطلاق ، في هذا الأسلوب فإن الإجرائية التي تطلب وحدة المعالجة أولاً سيتم تخصيصها للمعالج أولاً ، بحيث أنها يمكن تنفيذ خوارزمية FCFS باستخدام رتل FIFO بحيث : **FIFO=First In First Out**



مثال : ليكن لدينا الجدول التالي الذي يحتوي على اسماء الإجرائيات و زمن تنفيذ كل منها في ال CPU

Process	Burst Time
$P_1$	24
$P_2$	3
$P_3$	3

حيث أن الإجرائيات تأتي بالترتيب  $P_1 \rightarrow P_2 \rightarrow P_3$



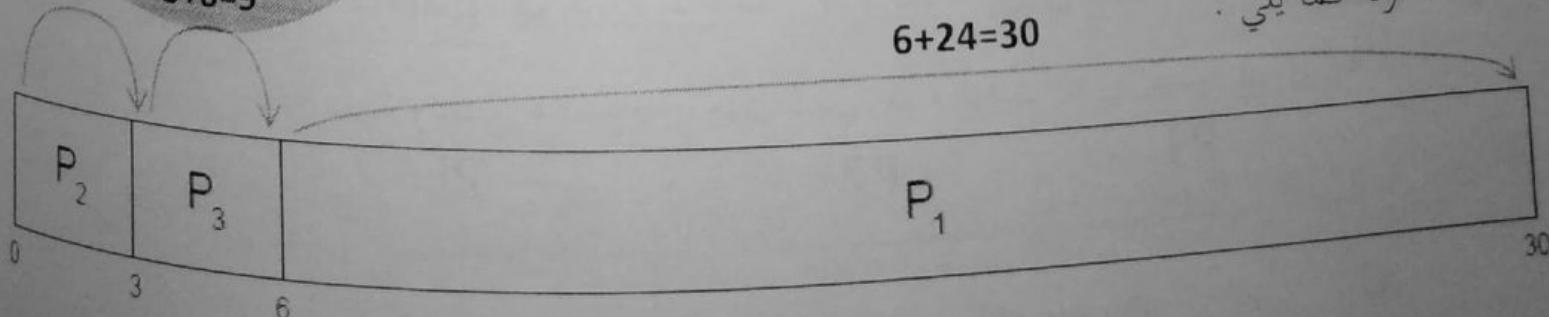
ويمكن زمن الانتظار waiting time للإجراءات هو :

$$P_1 = 0 \text{ ms} , P_2 = 24 \text{ ms} , P_3 = 27 \text{ ms}$$

بالتالي يكون متوسط زمن الانتظار يساوي :

$$\text{avg} = \frac{0 + 24 + 27}{3} = \frac{51}{3} = 17 \text{ ms}$$

لكن ، في حال كان ترتيب وصول الإجرائيات مختلفاً ، ولتكن  $P_2 \rightarrow P_3 \rightarrow P_1$  بالعالي ، سيكون مخطط غانت للجدولة كما يلي :



## نظم تشغيل المحاضرة الخامسة (خوارزميات الجدوله 1)



ويكون زمن الانتظار للإجراءات هو :

$$P_1 = 6 \text{ ms}, P_2 = 0 \text{ ms}, P_3 = 3 \text{ ms}$$

بالتالي يكون متوسط زمن الانتظار يساوي :

$$\text{avg.} = \frac{6+0+3}{3} = \frac{9}{3} = 3 \text{ ms}$$

ونلاحظ بأن ترتيب الإجرائيات في الحالة هذه هو أفضل بكثير ( $\text{avg}=3$ ) من الحالة السابقة ( $\text{avg}=17$ )

نستنتج أن خوارزمية FCFS

1) زمن الانتظار متعلق بترتيب ورود الإجرائيات إلى رتل الحافظة وهذا ما يسمى بـ :

### Convoy effect

2) جدوله لا شفعة non-preemptive لأن الإجرائية التي ستدخل إلى وحدة المعالجة تتبقى فيها

إلى أن تطلب I/O أو أن تنتهي تلقائياً (لا يمكن طرد الإجرائية بشكل قسري (استباقي)).

3) غير ملائمة للأنظمة التفاعلية interactive systems (أنظمة اقتسام الزمن) لأنها لا تسمح بحدوث المقاطعات ((وهذا سيكون كارثي بأن نترك إجرائية تحتكر المعالج لوقت غير محدود)).

تحدثنا سابقاً عن نوعين من الإجرائيات :

1\_ إجرائيات مقيدة بوحدة المعالجة CPU-bound process

2\_ إجرائيات مقيدة بوحدات الإدخال والإخراج I/O-bound process

بال التالي عند وجود إجرائية CPU-bound والعديد من الإجرائيات I/O-bound هنا سوف يتعاظم تأثير القافلة convoy effect في حال كانت الإجرائية المقيدة بوحدة المعالجة هي الإجرائية التي ستتفقد أولاً.



## نظم سين المتعدد النسبي (حوارميات الجدولة 1)

المثالين السابقين في FCFS نعتمد على أن زمن الوصول Arrival Time للإجراءات كلها يساوي الصفر أي أنه عند بدء الزمن  $t=0$  فإن كل الإجرائيات موجودة في رتل الماجazine.

وبحسب ترتيب ورود هذه الإجرائيات فإنها تنفذ داخل المعالج ...

الآن على اعتبار أن أزمنة الوصول مختلفة (ليست صفر) كيف سنتعامل مع الموضوع؟

نقوم بترتيب الإجرائيات (ترتيب الورود) اعتماداً على زمن وصول الإجرائيات Arrival Time

بحيث أن الإجرائية  $P_i$  لا تستطيع التنفيذ إن لم تصل (اللحظة الحالية  $\leq$  زمن وصولها).

لناخذ المثال التالي :

Processes	Burst time	Arrival Time
P <sub>0</sub>	5	0
P <sub>1</sub>	3	1
P <sub>2</sub>	8	2
P <sub>3</sub>	6	3

لاحظ أن ترتيب الورود هو  $P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_3$  هنا لن يغير شيء ...

لكن لو كان لدينا ما يلي :

Processes	Burst time	Arrival Time
P <sub>0</sub>	5	3
P <sub>1</sub>	3	0
P <sub>2</sub>	8	2
P <sub>3</sub>	6	4

هنا سيكون ترتيب الورود  $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_0$  وسيتغير كل شيء ☺

احسب زمن الانتظار الوسطي

## نظم تشغيل المحاضرة الخامسة (خوارزميات الجدولة 1)

### 2\_ خوارزمية أقصر عمل أولاً (Shortest Job First (SJF))

يتم ربط كل إجرائية بطول رشقة CPU التالية بحيث يتم استخدام هذه الأطوال لتحديد أي إجرائية سيتم تنفيذها ، يمكننا تسميتها بـ (خوارزمية أقصر رشقة CPU) تالية (shortest next CPU Burst) CPU . لأن عملية الجدولة تعتمد على طول رشقة الـ CPU . ولأن طول رشقات الإجرائية .

ملاحظة : الإجرائية تتألف من عدة رشقات I/O و رشقات CPU بحيث أن :

طول رشقات الإجرائية <> طول رشقة CPU أو طول رشقة I/O واحدة

#### : SJF ميزة الـ

هي الخوارزمية الأمثل Optimal فهي التي تعطي أقل معدل زمن انتظار average waiting time لمجموعة من الإجرائيات .

#### : SJF سلبيات الـ

- في نظم Multi-Processing من الصعب معرفة طول رشقة الـ CPU التالية للإجرائية ما ، وإنما يتم التنبؤ بها فقط من خلال طول رشقة CPU السابقة للإجرائية نفسها .
- في نظم Batch Systems قد نسأل المستخدم عن الزمن اللازم لهذه الإجرائية (تسمى هنا عمل job) وبالتالي قد يطلب زمناً أكبر (وهذا ضياع للوقت) وقد يطلب زمناً أقل مما يستدعي لأن يطلب مدة إضافية مما يؤدي إلى حدوث المشاكل .



## [[ خوارزمية الأقصر عمل أولاً غير الشفعة non preemptive ]]

تتألف من الخطوات التالية :

- 1\_ ربط كل إجرائية بطول رشقة CPU التالية التي سيتم تنفيذها (إعطاء كل إجرائية رقم يدل على الزمن).
- 2\_ يتم اختيار الإجرائية ذات طول الرشقة التالية الأقصر ، وفي حال وجدت إجرائيتين بنفس طول الرشقة التالية يتم استخدام خوارزمية FCFS (أي يتم الاختيار حسب الإجرائية التي أتت أولاً).
- 3\_ طالما يوجد في رتل الجاهزية إجرائيات وبالتالي يتم العودة إلى الخطوة 1 مجدداً.

مثال 1 : لتكن لدينا إجرائيات التالية مع زمن التنفيذ لكل إجرائية مبينة في الجدول التالي :

Process	Burst Time
P <sub>1</sub>	6
P <sub>2</sub>	8
P <sub>3</sub>	7
P <sub>4</sub>	3

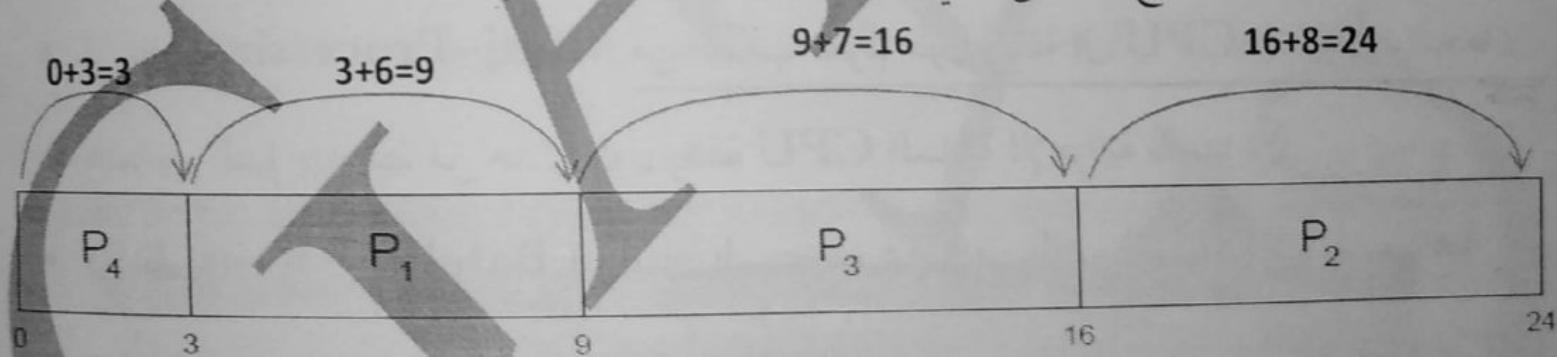
نلاحظ أنه لدينا أربع إجرائيات في ready queue

(رتل الجاهزية) ، وبالتالي نبحث عن الإجرائية ذات طول الرشقة

الأقل ونلاحظ أنها الإجرائية P<sub>4</sub> وبالتالي نضعها أولاً على مخطط

غانث ، ثم سنضع الإجرائية P<sub>1</sub> ثم P<sub>3</sub> ثم الإجرائية P<sub>2</sub>

أي أن مخطط غانث سيصبح بالشكل التالي :



ويكون زمن الانتظار waiting time للإجراءات هو :

$$P_1 = 3 \text{ ms} , \quad P_2 = 16 \text{ ms} , \quad P_3 = 9 \text{ ms} , \quad P_4 = 0 \text{ ms}$$

بالتالي يكون متوسط زمن الانتظار يساوي :

$$\text{avg} = \frac{3 + 16 + 9 + 0}{4} = \frac{28}{4} = 7 \text{ ms}$$

## : Preemptive SJF

[ب] - خوارزمية الأقصر عمل أولاً الشفعة

يتم اختيار هذه الخوارزمية عندما تأتي إجرائية جديدة إلى رتل الانتظار بينما هناك إجرائية أخرى يتم تنفيذها في CPU. بحيث أنه قد تكون رشقة الـ CPU التالية للإجرائية الجديدة أقصر من رشقة الـ CPU التالية للإجرائية التي تنفذ حالياً، وبالتالي هنا إن استخدمنا خوارزمية SJF الشفعة فسوف تقوم باستباق الإجرائية (وضعها في رتل الانتظار) ووضع الإجرائية الجديدة ذات طول الرشقة الأقصر في الـ CPU حتى تنتهي لتعود الإجرائية ذات الرشقة الأقصر إلى التنفيذ.

يمكننا تسمية خوارزمية SJF الشفعة بـ خوارزمية أقصر وقت متبقي أولاً

Preemptive SJF  $\Rightarrow$  Shortest-Remaining-Time-First(SRTF)

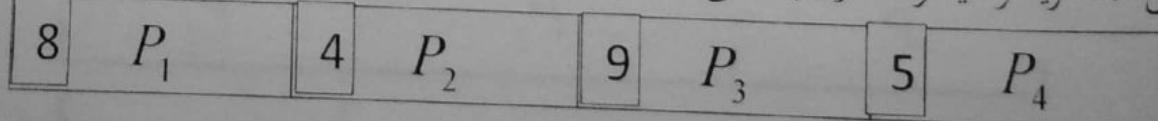
### : preemtive SJF

- عند تطبيق هذه الخوارزمية يجب أن نعلم زمن الرشقة CPU Burst وزمن الوصول arrive time.
- يتم البدء بالإجرائية ذات زمن الوصول الأقل Min Arrival Time.
- إن وجد إجرائيتين بنفس زمن الوصول وبالتالي يتم اختيار الإجرائية ذات زمن الرشقة الأقصر ومن ثم يتم وضع هذه الإجرائية على خط غانت.
- إن تقاطع زمن تنفيذ الإجرائية التي وضعتها مع زمن وصول إجرائية أخرى وكان زمن رشقة الإجرائية التي وصلت مؤخراً أقصر من الوقت المتبقى للإجرائية التي تتفق وبالتالي يتم التبديل بين الإجرائيتين مباشرةً.

مثال 2 : لتكن لدينا الإجرائيات التالية المبينة في الجدول التالي :

Process	Arrival Time	Burst Time
$P_1$	0	8
$P_2$	1	4
$P_3$	2	9
$P_4$	3	5

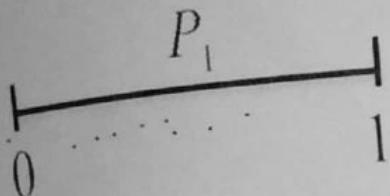
أولاً لرسم رتل الجاهزية وسيكون الترتيب على أساس زمن الوصول إليه أي :



# نظم سعيل المعاصره الخامسة (خوارزميات الجدولة ١)

حسب خوارزمية SJF فإنه في هذه الحالة لا يوجد خيار أمامنا إلا الإجرائية P1

وذلك لأننا الآن على مخطط غانت في اللحظة 0 والإجرائية ذات زمن الوصول 0 هي الإجرائية P1 فقط أي أنه لا وجود تفاضل بين الإجرائيات في هذه الحالة

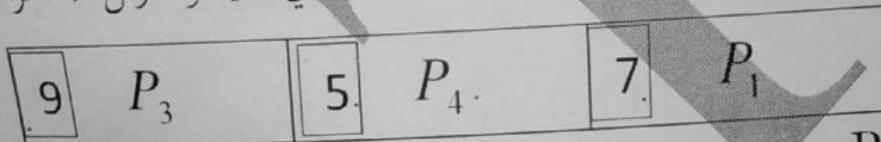


في اللحظة 1 جاءت الإجرائية P2

هنا نظام التشغيل (وفق هذه الخوارزمية) سيختبر الوقت المتبقى لتنفيذ الإجرائية P1 هنا سيكون 7 ms

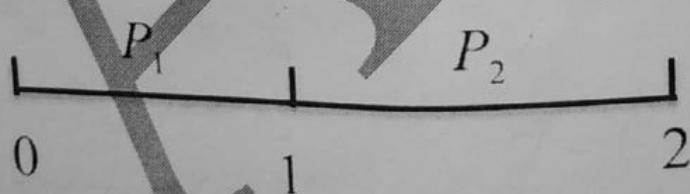
وبنقارنه مع طول رشقة P2 وهو 4ms وبما أن  $7 > 4$  وبالتالي فإن نظام التشغيل سيطرد الإجرائية P1 من CPU وسيدخل P2 إلى CPU لتبدأ العمل.

ملاحظة: P1 عندما تطرد تذهب إلى رتل الجاهزات  $\Leftarrow$  وبالتالي سيكون رتل الجاهزات بعد الطرد :



بحيث أن زمن وصول P1 هو زمن وصول آخر إجرائية ويساوي 3 مضافاً له الزمن الذي استغرقه الإجرائية P1 في الـ CPU ويساوي 1 وبالتالي زمن وصول P1 يساوي 4 وطول رشقتها المتبقية يساوي 7.

$\Leftarrow$  مخطط غانت سيصبح بالشكل :



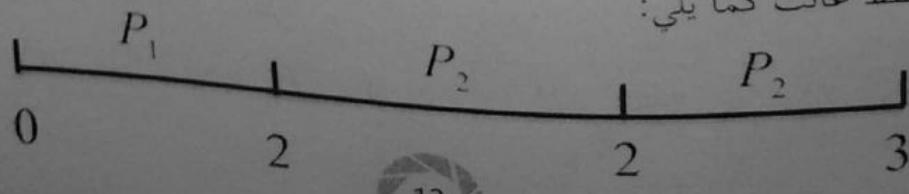
في اللحظة 2 جاءت الإجرائية P3  $\Leftarrow$  سيختبر نظام التشغيل الأمرين السابقين ويقارنهم كما يلي :

أ - ما هو الوقت المتبقى لـ P2 ؟ 3 ms .

ب - ما هو طول رشقة لـ P3 ما هو الوقت الكلي اللازم لتنفيذ P3 ؟ 9 ms .

بما أن  $3 < 9 \Leftarrow$  لن تطرد P2 من CPU بل ستتابع عملها .

وبالتالي يصبح مخطط غانت كما يلي :



## نظم تشغيل المحاضرة الخامسة (خوارزميات الجدوله 1)

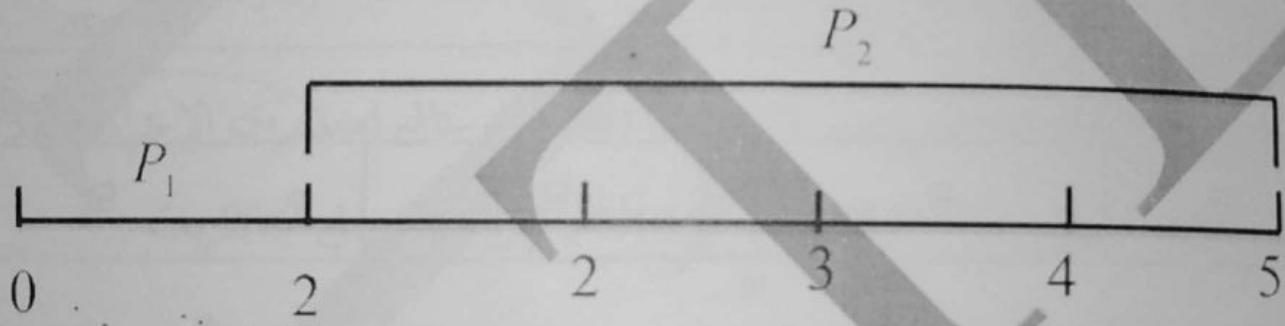


- في اللحظة 3 ستأتي  $P_4 \leftarrow$  سيختبر نظام التشغيل كلا الأمرتين السابقتين ويقارنهما كما يلي :
- ما هو الوقت المتبقى لـ  $P_2$  ؟  $2 \text{ ms}$ .
  - ما هو طول رشقة الإجرائية  $P_4$  ؟  $5 \text{ ms}$ .

$P_2 \leftarrow 5 > 2$  لن تطرد من وحدة المعالجة بل ستتابع عملها.

وستستمر بذلك لمدة  $2 \text{ ms}$  أي حتى اللحظة 5 لأنها خلال هذه المدة لم تأتي أي إجرائية جديدة إلى رتل المستعديات حتى يقوم نظام التشغيل باختبار الشرطين السابقتين.

وبالتالي يصبح مخطط غانت كما يلي :



بالتالي سيكون رتل الجاهزات بالشكل :

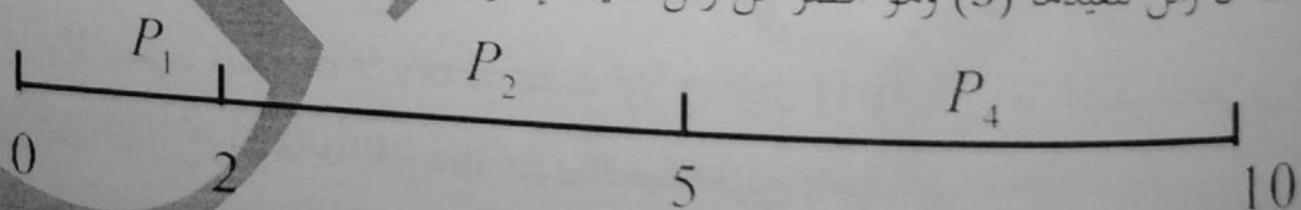
9	$P_3$	5	$P_4$	7	$P_1$
---	-------	---	-------	---	-------

الآن عند اللحظة 5 ، الإجرائية  $P_2$  ستكون قد انتهت من تنفيذها  $\leftarrow$

المجدول عليه اختيار إجرائية جديدة من رتل الجاهزية ، ولكن أي إجرائية سيختار ؟

الجواب : سيختار الإجرائية  $P_4$  لأن طول رشقتها هو أقل طول رشقة بين الإجرائيات في رتل الجاهزية.

ونلاحظ أن زمن تنفيذها (5) وهو أقصر من زمن تنفيذ الإجرائيات المتبقية أي أنه لن يتم مقاطعتها أبداً

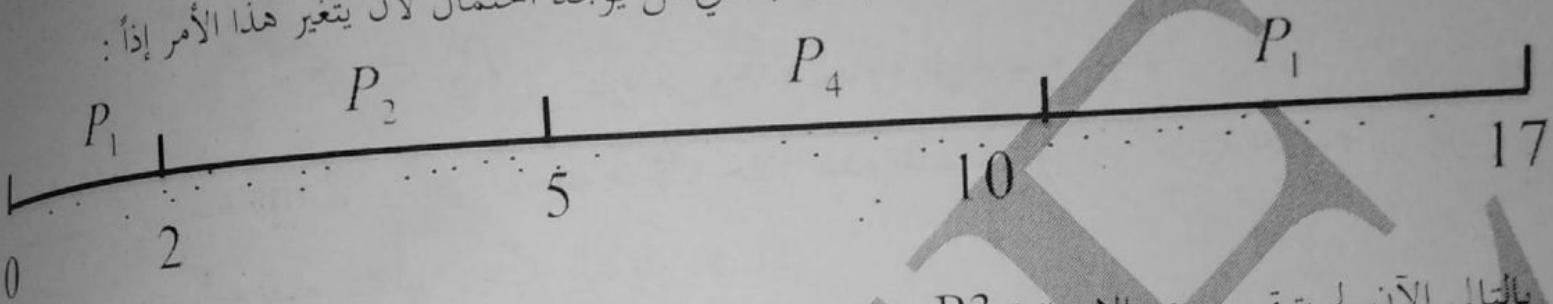


بالتالي يصبح رتل الجاهزية بالشكل :

9	$P_3$	7	$P_1$
---	-------	---	-------

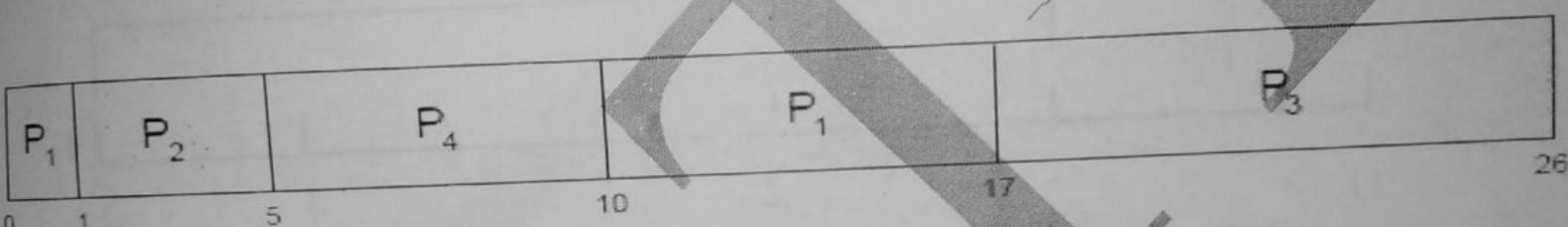
# السبعين استاذة الخامسة (خوارزميات الجدولية)

الآن عند اللحظة 10 ستكون الإجرائية  $P_4$  قد انتهت عملها تماماً وبالتالي علينا الاختيار بين  $P_1$  و  $P_3$  وبالطبع سوف نختار الإجرائية  $P_1$  لأنها ذات طول الرشقة الأقصر ونلاحظ أنه لن يتم مقاطعتها من قبل الإجرائية  $P_3$  لأنه بالأصل أقصر منها منذ البداية وبالتالي لن يوجد احتمال لأن يتغير هذا الأمر إذاً:



بالتالي الآن لم يتبقى سوى الإجرائية  $P_3$  ذات طول الرشقة 9 أي أن مخطط غانت النهائي يصبح:

$$17+9=26$$



هام جداً

بالتالي يكون زمن انتظار كل إجرائية يساوي

وقت بدء التنفيذ للإجرائية - زمن وصول الإجرائية - الزمن الذي قضته هذه الإجرائية في التنفيذ سابقاً

$$\text{Wait}(P_i) = \text{start}(P_i) - \text{arrive}(P_i) - \text{ExecutingTime}(P_i)$$

$$P_1 \Rightarrow 10 - 0 - 1 = 9$$

$$P_3 \Rightarrow 17 - 2 - 0 = 15$$

$$P_2 \Rightarrow 1 - 1 - 0 = 0$$

$$P_4 \Rightarrow 5 - 3 - 0 = 2$$

ملاحظة: إن  $P_1$  انتظرت فقط 9 ms لأنها قد نفذت أولاً في المجال [0,1] أي أنها لم تكن تنتظر في هذا الوقت وبالتالي تعتبر أن زمن وصولها هو 1 (نقطة انتهاء التنفيذ الحرئي للإجرائية).

بالتالي يكون متوسط زمن الانتظار يساوي :

$$\text{avg} = \frac{9 + 0 + 15 + 2}{4} = \frac{26}{4} = 6.5$$

تنوية: لو كنا استخدمنا خوارزمية SJF غير الشفعة لكان زمن الانتظار الوسطي يساوي 7.5 ms

ملخص عمل خوارزمية SJF

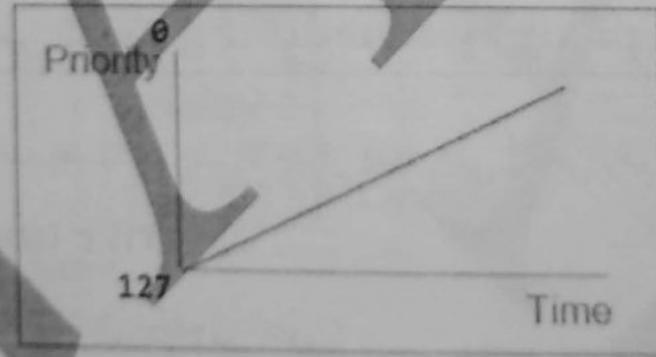
• ترتيب كل الإجرائيات بالنسبة ل زمن الوصول Arrival Time

• اختيار الإجرائية التي يمتلك أصغر minimum زمن وصول وأصغر طول رسالة Burst Time.

• بعد انتهاء تنفيذ هذه الإجرائية ، نقوم بإنشاء لائحة pool من الإجرائيات التي مستنفذة بعد انتهاء الإجرائية الأخيرة ونختار الإجرائية الموجودة في اللائحة التي تملك أقصر طول رسالة .

إن خوارزمية SJF ستتعاني من Starvation في حال استمرت الإجرائيات ذات زمن الرسالة القصير short burst time بالورود إلى رتل أباهازية ...

يتم حل هذه المشكلة باستبدال مبدأ Aging (التعمير) ، أي أنه كلما يقضى الإجرائية في الرتل سيرداد معامل آخر وهو الأولوية Priority بحيث بعد انقضاء مدة محددة سيتم تنفيذ الإجرائية ذات زمن الرسالة الطويلة .



☺ سفهـ ما المقصود بالأولوية عند التعرف على خوارزمية Priority Scheduling

كلما ازداد الوقت يتناقص الرقم الصحيح  
بالتالي تزداد الأولوية وتستطيع الإجرائية السعيدة أن  
تنفذ وتصبح أكثر سعادة ☺

### 3\_ خوارزمية الجدولة وفق الأولوية Priority Scheduling

يتم إعطاء رقم أولوية (رقم صحيح) لكل إجرائية بحيث أن الرقم الأصغر يعني أولوية أكبر.

يتم تخصيص الأولوية ذات الأولوية الأعلى إلى المعالج ليتم معالجتها أولاً.

خوارزمية SJF السابقة هي حالة خاصة من خوارزمية Priority أي أنه :

يوجد تناوب عكسي inverse بين زمن الرشقة التالية CPU Burst وأولوية الإجرائية Priority

مثال : لدينا إجرائية P1 زمن الرشقة = 5 وإجرائية P2 زمن الرشقة = 3 .

← الإجرائية P2 لها أولوية أعلى من الإجرائية P1 .

لكن !! رقم الأولوية المعطى ل P2 هو أصغر من رقم الأولوية المعطى ل P1 .

تنويه : يجب التمييز بين الأولوية ورقم الأولوية حيث أن الأولوية العليا تعني رقم أولوية صغير ☺

يوجد نوعان من Priority Scheduling

#### [أ]\_ الجدولة وفق الأولوية غير الشفعة non-preemptive priority scheduling

هنا لا يوجد استباقيّة للإجراءات بحيث أنه يتم الاختيار فقط على أساس رقم الأولوية أي أننا لا نتعامل في هذا النوع مع زمن الوصول arrive time أبداً .

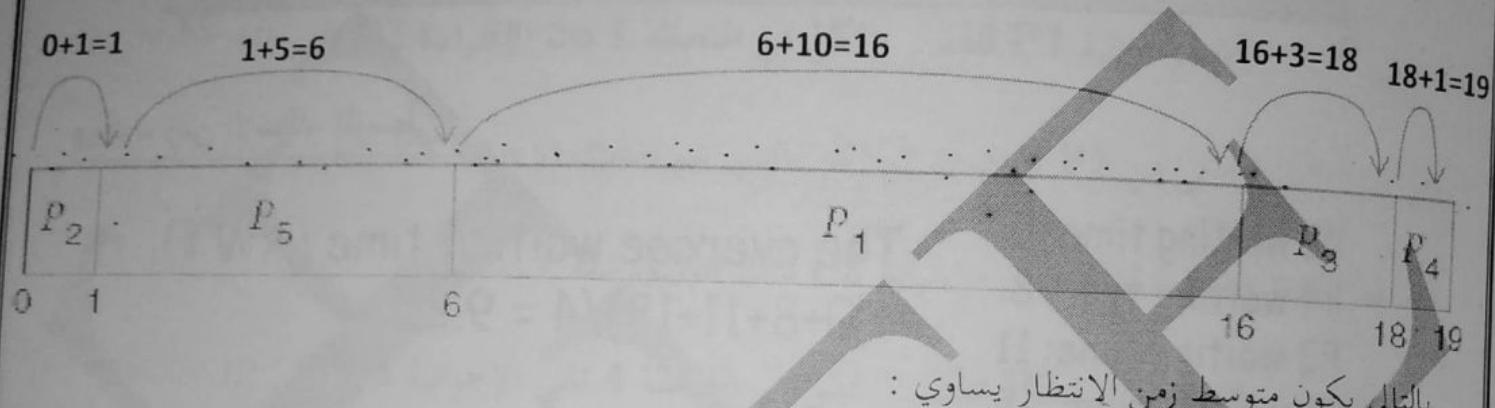
مثال : لتكن لدينا الإجراءات المبينة في الجدول التالي :

Process	Burst Time	Priority
$P_1$	10	3
$P_2$	1	1
$P_3$	2	4
$P_4$	1	5
$P_5$	5	2

## نظم تشغيل المحاضرة الخامسة (خوارزميات الجدولة 1)



بالناتي سيتم اختيار الإجرائية ذات رقم الأولوية الأقل وهي الإجرائية P2 ثم الإجرائية P5 ثم P1 ثم الإجرائية P4 ثم P3 أي أن مخطط غانت سيكون بالشكل التالي :

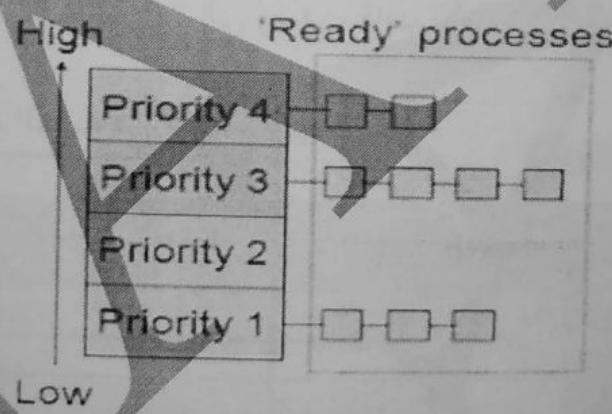


بالناتي يكون متوسط زمن الانتظار يساوي :

$$avg = \frac{0+1+6+16+18}{5} = \frac{41}{5} = 8.2$$

يمكنا فهم ذلك من منظور آخر بحيث يكون لدينا مصفوفة من اللوائح المرتبطة

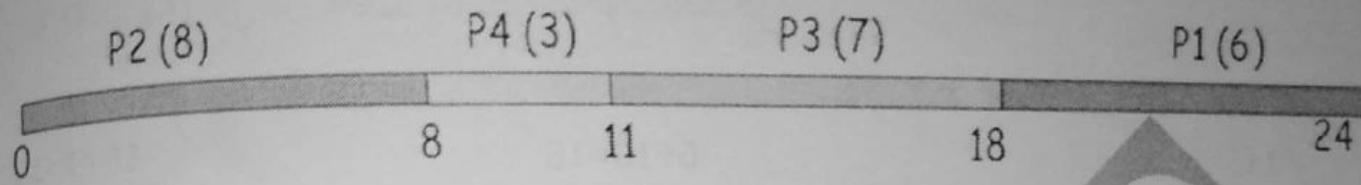
كل عنصر (لائحة متراقبة) يعبر عن أولوية معينة ويحتوي على الإجرائيات التي تمتلكها هذه الأولوية :



مثال 2 :

Process	Duration	Priority #	Arrival Time
P1	6	4	0
P2	8	1	0
P3	7	3	0
P4	3	2	0

ويكون مخطط غانت بالشكل :



ويكون زمن الانتظار الوسطي :

P2 waiting time: 0  
 P4 waiting time: 8  
 P3 waiting time: 11  
 P1 waiting time: 18

The average waiting time (AWT): ~  
 $(0+8+11+18)/4 = 9.25$

### [أ] الجدولة وفق الأولوية الشفعة : preemptive priority scheduling

يتم اختيار الإجرائية ذات الأولوية الأعلى بحسب زمن الوصول أي أنه عند لحظة معينة فإنه يجب الاختيار من بين عدة إجرائيات وبالتالي يتم اختيار الإجرائية التي وصلت ذات الأولوية الأعلى (أصغر رقم أولوية)

مثال : لتكن لدينا الإجرائيات المبينة في الجدول التالي :

Process	Burst Time	Priority	Arrive Time
P1	4	3	0
P2	7	2	1
P3	6	1	4

الآن يجب التعامل مع الإجرائيات بالنسبة لعاملين :

الأول هو زمن الوصول Arrive Time (لا يمكننا تنفيذ إجرائية لم تصل بعد) ...

الثاني هو الأولوية Priority (لا يمكننا تنفيذ إجرائية لها أولوية أقل من إجرائية أخرى موجودة)

## نظم تشغيل المحاضرة الخامسة (خوارزميات الجدولة 1)

أولاً سبباً من اللحظة 0 وتنظر إلى أزمنة الوصول للإجراءات كلها ... فنلاحظ أن الإجرائية P1 قد وصلت في اللحظة 0 (التي نحن فيها الآن) ذات طول الرشقة 4 والأولوية 3 (بعض النظر عن أولويتها).

الآن الإجرائية P1 تنفذ ... لكن في اللحظة 1 فإن الإجرائية P2 تأتي !!!

وهي تملك أولوية أعلى من P1 حيث أن  $3 < 2$  وبالتالي يتم استياب الإجرائية P1.

ووضعها في رتل الجاهزية ثم تخصيص الإجرائية P2 ذات طول الرشقة 7.

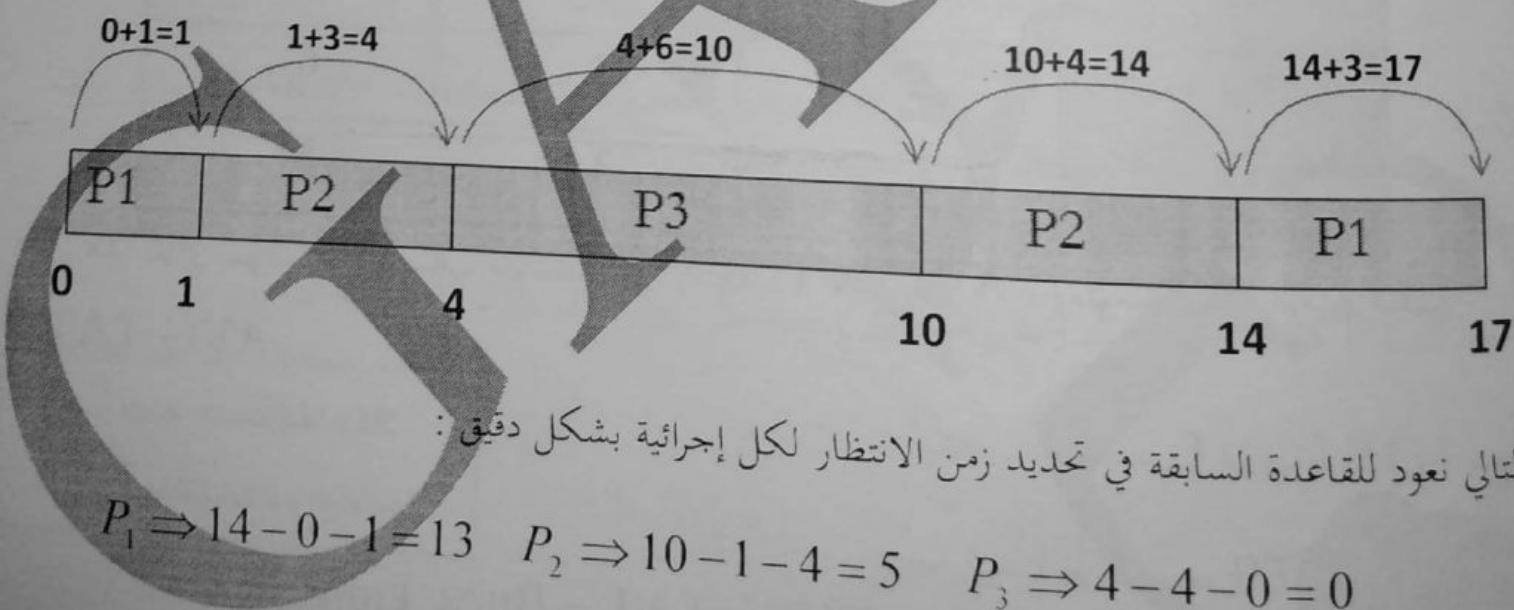
الآن الإجرائية P2 تنفذ ... لكن في اللحظة 4 تأتي الإجرائية P3 تأتي !!!

وهي تملك أولوية أعلى من P2 حيث أن  $2 < 1$  وبالتالي يتم استياب الإجرائية P2.

ووضعها في رتل الجاهزية ثم تخصيص الإجرائية P3 ذات طول الرشقة 6.

بالتالي بعد انتهاء P3 سوف يتم إرجاع P2 أولاً ثم P1 ثانياً (بحسب الأولوية).

ويكون مخطط غانت بالشكل :



بالتالي نعود للقاعدة السابقة في تحديد زمن الانتظار لكل إجرائية بشكل دقيق :

$$P_1 \Rightarrow 14 - 0 - 1 = 13 \quad P_2 \Rightarrow 10 - 1 - 4 = 5 \quad P_3 \Rightarrow 4 - 4 - 0 = 0$$

بالتالي يكون متوسط زمن الانتظار يساوي :

$$\text{avg} = \frac{13 + 5 + 0}{3} = \frac{18}{3} = 6 \text{ ms}$$

## ملاحظات حول Priority Scheduling (هام جداً) :

\* هناك مشكلة تسمى بال **Starvation** (الحرمان) (Infinity Blocking) وهي تعني أن

الإجراءات ذات الأولويات المنخفضة **low priority** ربما لن تنفذ أبداً بسبب وجود إجراءات

أخرى أولويتها أعلى منها مما يعني تأجيل تنفيذها في كل مرة وهكذا إلى أن تصبح مناسبة تماماً ⑥

\* تم إيجاد حل لهذه المشكلة ويسمى بال **Aging** ويعني التعديل أي أنه كلما ازداد الزمن الذي تنتظر

فيها الإجرائية (كلما ازداد عمرها) كلما ازدادت أولويتها وبالتالي إتاحة فرصة لتنفيذها في المستقبل.

((نقصد هنا بزيادة الأولوية إننا نقص رقم الأولوية الخاص بالإجرائية))

مثال هام : لتكن لدينا الإجراءات المبينة في الجدول التالي :

Process	Burst Time	Priority	Arrive Time
P1	5	2	0
P2	8	1	4
P3	2	4	6
P4	6	3	8

في هذا المثال سوف نقوم بحساب متوسط زمن الانتظار **AVG WT** ، ومتوسط الفترة الزمنية الكلية ... **AVG TAT**

نعلم أن :

$$\text{Waiting Time (WT)} = \text{TAT} - \text{Burst Time}$$

بالتالي نستطيع أن نكتب :

$$\text{TAT} = \text{WT} + \text{Burst Time}$$

## نظم تشغيل المحاضرة الخامسة (خوارزميات الجدولة 1)

الآن نحن في اللحظة 0 ولدينا الإجرائية P1 قد وصلت ولا يوجد أي إجرائية أخرى موجودة لكي تنفذ وبالتالي يوف تدخل الإجرائية P1 ويتم تخصيصها ...

الآن الإجرائية P1 تنفذ ... في اللحظة 1 تنفذ أيضاً ... في اللحظة 2 تنفذ أيضاً ...

في اللحظة 3 تنفذ أيضاً ... لكن !!!

في اللحظة 4 تأتي الإجرائية P2 وتملك أولوية 1 أعلى من أولوية الإجرائية P1

← وبالتالي يتم استباق الإجرائية P1 وضع P2 مكانها ☺

الآن الإجرائية P2 تنفذ ... في اللحظة 5 أيضاً تنفذ ... في اللحظة 6 أيضاً تنفذ ...

في اللحظة 6 تأتي الإجرائية P3 لكن لا يوجد مشكلة لأن أولويتها 4 أقل من أولوية P2

الآن الإجرائية P2 تنفذ ... في اللحظة 7 أيضاً تنفذ ...

في اللحظة 8 تأتي الإجرائية P4 لكن لا يوجد مشكلة لأن أولويتها 3 أقل من أولوية P2

☺ وبالتالي ستستمر الإجرائية بالتنفيذ إلى أن تنتهي التنفيذ بشكل تام ☺

الآن بما أن كل الإجرائيات وصلت ، سننصح للإجرائيات بالتنفيذ دون مقاطعة بعد الآن ولكن يجب أن نرتب

شكل تنازلي حسب الأولوية (الأعلى أولوية أولاً ثم الأقل أولوية) .

لاحظ أن P1 كانت أولويتها 2 وتبقي لها ثانية واحدة لكي تنتهي تنفيذها ، وبالتالي نضعها ...

هذه المنشآت يجب أن  
تسنطط مناقشتها بمفردها

لاحظ أن P4 أولويتها 3 نضعها قيد التنفيذ إلى أن تنتهي

ثم يتبقى الإجرائية P3 نضعها أيضاً حتى تنتهي .

P1	P2	P1	P4	P3
0	4	12	13	19
				21

# نظم تشغيل المحاضرة الخامسة (خوارزميات الجدولة 1)

الآن سنقوم بحساب زمن الانتظار لكل إجرائية :

من الواضح أن الإجرائية P1 أتت في اللحظة 0 وبدأت التنفيذ في نفس اللحظة ... لم تنتظر هنا لكنها انتظرت من اللحظة 4 إلى 12 أي  $12-4=8$ .

الإجرائية P2 أتت في اللحظة 4 ونفدت في نفس اللحظة أي  $0-4=4$ .

الإجرائية P3 أتت في اللحظة 6 وبدأت التنفيذ في اللحظة 19 أي  $19-6=13$ .

الإجرائية P4 أتت في اللحظة 8 وبدأت التنفيذ في اللحظة 13 أي  $13-8=5$ .

$$\text{Average } WT = \frac{8+0+13+5}{4} = \frac{26}{4} = 6.5 \text{ ms}$$

بحساب بسيط نستطيع إيجاد ال TAT لكل إجرائية كما يلي :

$$TAT(P_i) = WT(P_i) + \text{Burst}(P_i)$$

$$TAT(P1) = 8 + 5 = 13$$

$$TAT(P2) = 0 + 8 = 8$$

$$TAT(P3) = 13 + 2 = 15$$

$$TAT(P4) = 5 + 6 = 11$$

$$\text{Average TAT} = \frac{13+8+15+11}{4} = \frac{47}{4} = 11.75 \text{ ms}$$

*Special thanks to Ahmad Mokayes*

انتهت المحاضرة

بالتوفيق ☺

M#

Mohammed Moulla