الوحدة الثالثة: إدارة المعالجة

م. غنام الجعبري

مقدمة

- مدير المعالجة هو المسؤول عن تخصيص او اشغال المعالج لتنفيذ المهام او البرامج الواردة وإدارة العمليات في وحدة المعالجة المركزية (CPU)
- في نظم التشغيل التي تتكون من مستخدم واحد ومعالج واحد، يكون المعالج مشغول فقط في تنفيذ مهمة للمستخدم او برنامج للنظام
- في نظم التشغيل التي تتكون من عدة مستخدمين او عدة برامج تتنافس على معالج واحد، يجب تخصيص وقت على المعالج لكل مهمة او برنامج بطريقة عادلة وفعالة
- تزداد وظيفة مدير المعالجة تعقيدا عند توفر اكثر من معالج (multiprocessing)، او عندما يحتوي المعالج على اكثر من نواة (multi-core) أي اكثر من وحدة معالجة في نفس الشريحة

مصطلحات

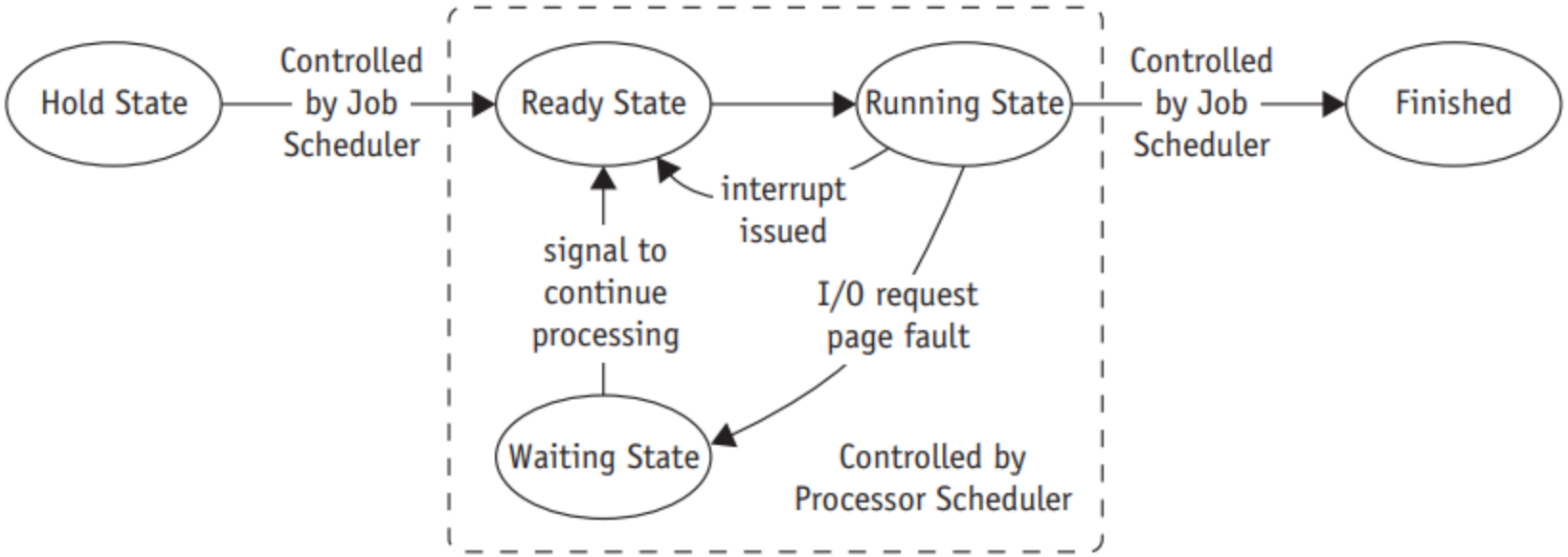
- **المعالج (processor)** او وحدة المعالجة المركزية (CPU) هي وحدة في جهاز الحاسوب تقوم بإجراء الحسابات وتنفيذ البرامج
- **البرنامج (program)** او المهمة (Job) هي وحدة غير نشطة اي ملف على القرص
- **العملية (process)** هي وحدة نشطة تتطلب مجموعة من الموارد حتى تقوم بأداء المهمة أي برنامج قيد التنفيذ (program in execution)
- **سلسلة (thread)** هي وحدة تنفيذية داخل العملية يتم جدولتها وتنفيذها بشكل مستقل عن العملية التابعة لها، ويمكن ان تتألف العملية من مجموعة من السلاسل او الأجزاء التي تستغرق وقتا اقل في التنفيذ من العمليات لكن اكثر تعقيدا
- **تعدد السلاسل (multithreading)** يتيح للتطبيقات ادارة العملية باستخدام عدة سلاسل مثل متصفحات الويب، قد تستخدم سلسلة لجلب الصور وسلسلة اخرى لجلب النص

مديرو الجدولة (Scheduling Submanagers)

- يتضمن مدير المعالجة مديرين فرعيين على الأقل: احدهما مسؤول عن جدولة المهام ويدعى **Job Scheduler** والآخر عن جدولة العمليات ويدعى **Process Scheduler**
- يهتم مدير جدولة المهام باختيار المهام من قائمة المهام الواردة ووضعها في قائمة العمليات بناء على خصائص كل مهمة مثل حجم الذاكرة المطلوبة ومدة المعالجة المتوقعة والاولوية
- بعض المهام تحتوي على العديد من طلبات الادخال والإخراج (I/O) وتدعى **I/O-bound** والبعض الآخر يحتوي على العديد من العمليات الحسابية وتدعى **CPU-bound**، لذا يسعى مدير المهام الى الموازنة في الاختيار بينهما لتجنب اشغال وحدات الادخال والإخراج لوحدها او وحدة المعالجة المركزية، ومحاولة اشغال معظم مكونات الحاسوب اغلب الوقت
- بعد قبول المهام من قبل مدير جدولة المهام، يتولى مدير جدولة العمليات المسؤولية عنها
- يحدد مدير جدولة العمليات المهام التي سوف تشغل وحدة المعالجة المركزية والوقت والمدة

حالات العملية (Process States)

- تتغير حالة المهمة او العملية بعد تسليمها الى النظام كما يلي: إيقاف (HOLD)، استعداد (READY)، تشغيل (RUNNING)، انتظار (WAITING)، وأخيرا انتهاء (FINISHED)

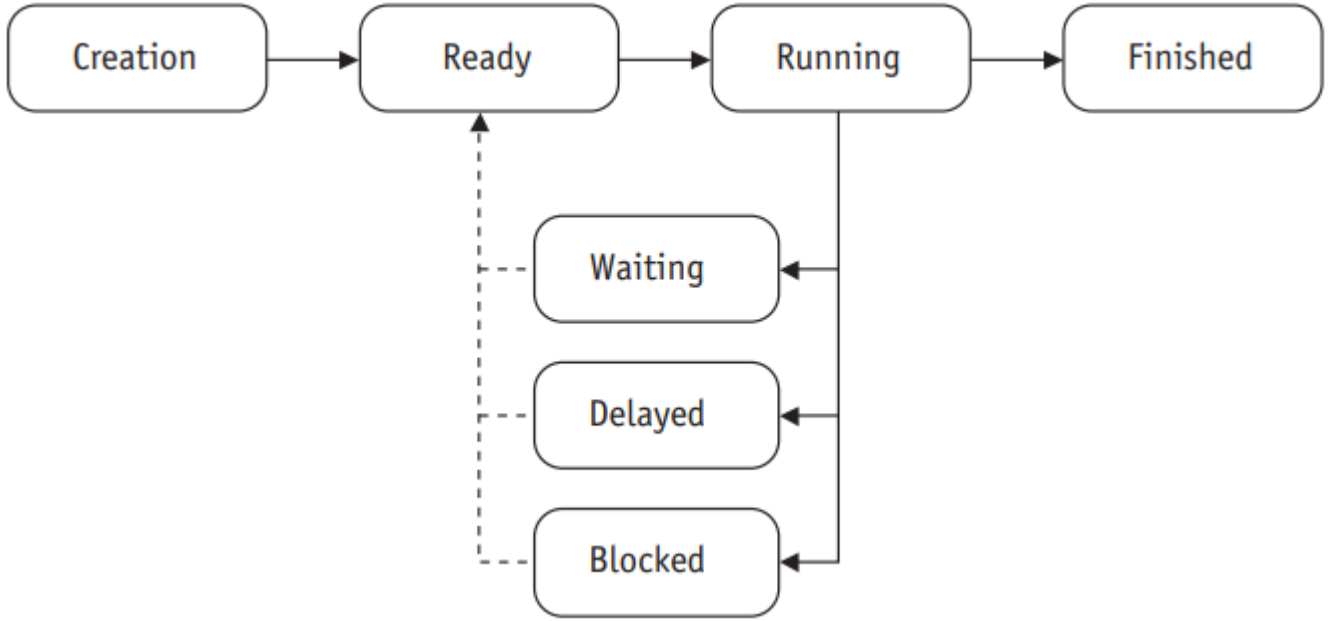


حالات العملية (Process States)

- تنتقل المهمة من حالة الى أخرى بواسطة مدير جدولة المهام، وتنتقل العملية من حالة الى أخرى من خلال مدير جدولة العمليات كما يلي:
  - الانتقال من حالة HOLD الى حالة READY يتم بواسطة مدير جدولة المهام
  - الانتقال من حالة READY الى حالة RUNNING يتم بواسطة مدير جدولة العمليات
  - الانتقال مرة أخرى من حالة RUNNING الى حالة READY يتم بواسطة مدير جدولة العمليات بناء على نهاية فترة المعالجة او معيار آخر مثل الأولوية في المقاطعة
  - الانتقال من حالة RUNNING الى حالة WAITING يتم من قبل مدير جدولة العمليات استجابة لاحد التعليمات في العملية مثل امر قراءة (READ)، او كتابة (WRITE)، او طلب ادخال وإخراج (I/O)
  - الانتقال من حالة WAITING الى حالة READY يتم من قبل مدير جدولة العمليات بعد تحقيق مدير الأجهزة لطلب الادخال والإخراج او امر القراءة او الكتابة على القرص وإمكانية استئناف العملية
  - الانتقال من حالة RUNNING الى حالة FINISHED من قبل مدير جدولة العمليات او جدولة المهام بعد انتهاء تنفيذ العملية بنجاح

حالات السلسلة (Thread States)

- عندما يقوم التطبيق بإنشاء سلسلة، يتم تخصيص الموارد لها ووضعها في قائمة READY
- عندما يقوم مدير جدولة العمليات باختيار السلسلة للمعالجة تتغير حالتها من READY الى RUNNING

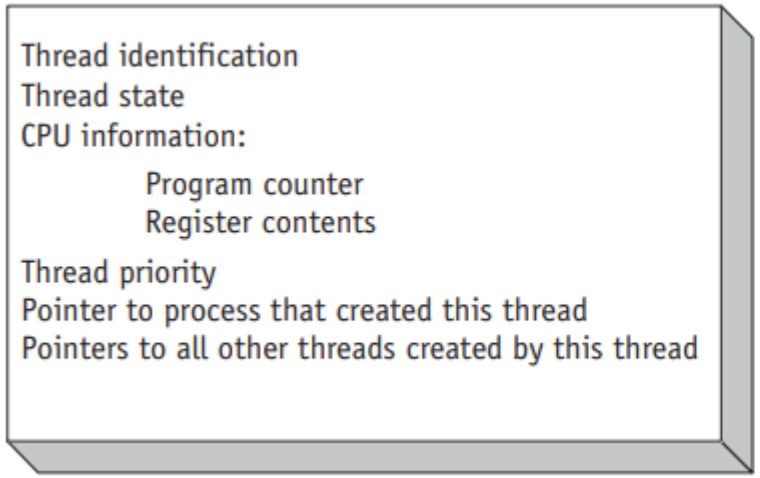


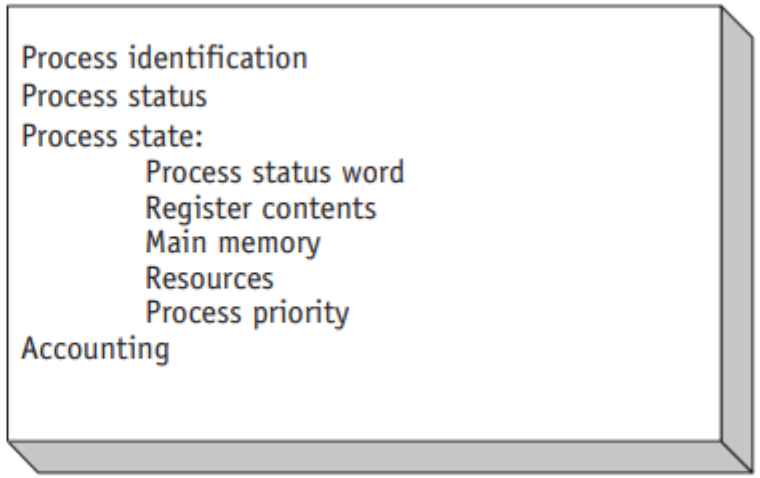
حالات السلسلة (Thread States)

- تنتقل السلسلة من حالة RUNNING الى WAITING عندما تنتظر وقوع حدث خارج عن سيطرتها مثل نقرة بزر الماوس، وبعد الحدث تعود الحالة من WAITING الى READY
- عندما يقوم التطبيق بتأخير معالجة السلسلة لفترة معينة من الوقت، تنتقل حالة السلسلة من RUNNING الى DELAYED، وبعد انقضاء الوقت تعود الحالة الى READY مثل تأخير السلسلة التي تقوم بحفظ المستند الحالي بشكل دوري لفترة من الوقت بعد انتهاء الحفظ
- تنتقل حالة السلسلة من RUNNING الى BLOCKED عند اصدار طلب ادخال او إخراج (I/O)، وبعد الانتهاء من عملية الادخال او الإخراج تعود الحالة الى READY
- عندما تنتقل حالة السلسلة من RUNNING الى FINISHED، يتم ارجاع كافة الموارد التي تم تخصيصها والخروج من النظام

كتل التحكم (Control Blocks)

- يمكن تمثيل كل عملية في النظام من خلال هيكل بيانات يدعى كتلة التحكم بالعملية (PCB)، ويمكن تمثيل كل سلسلة من خلال نفس هيكل البيانات ويدعى كتلة التحكم بالسلسلة (TCB)
- كتل التحكم تحتوي على المعلومات الأساسية حول العملية او السلسلة مثل الرقم والحالة
- القوائم (Queues) تستخدم كتل التحكم في متابعة المهام اثناء المعالجة مثل جواز السفر





سياسة الجدولة (Scheduling Policy)

- في بيئة البرمجة المتعددة غالبا هنالك مهام اكثر مما يمكن معالجتها في نفس الوقت، وحتى يتمكن نظام التشغيل من جدولتها ينبغي مراعاة القيود التالية في سياسة الجدولة:
  - عدد الموارد محدود
  - بعض الموارد عند تخصيصها او حجزها لا يمكن مشاركتها مع مهام أخرى
  - بعض الموارد تتطلب تدخل المشغل أي لا يمكن إعادة تعيينها تلقائيا من مهمة الى اخرى
- ما هي افضل سياسة في جدولة المهام والعمليات؟ او الأهداف التي يسعى نظام التشغيل الى تحقيقها مع ملاحظة ان بعض الأهداف قد تتعارض مع بعضها:
  - **زيادة الإنتاجية** (Maximize throughput) تعني تشغيل اكبر عدد من المهام في فترة زمنية محددة ويمكن تحقيق ذلك بسهولة عبر تشغيل مهام قصيرة او تشغيل مهام بدون مقاطعة
  - **تقليل وقت الاستجابة** (Minimize response time) يعني الاستجابة السريعة للطلبات التفاعلية ويمكن تحقيق ذلك عبر تشغيل المهام التفاعلية فقط وتأخير المهام الدفعية

سياسة الجدولة (Scheduling Policy)

- **تقليل وقت الإنجاز** (Minimize turnaround time) يعني ادخال وإخراج المهام بسرعة من النظام ويمكن تحقيق ذلك عبر تشغيل المهام الدفعية أولا ثم المهام التفاعلية
- **تقليل وقت الانتظار** (Minimize waiting time) يعني اخراج المهام من قائمة الانتظار (READY) في اقرب وقت ممكن ويمكن تحقيق ذلك فقط عبر تقليل عدد المستخدمين على النظام حتى تصبح وحدة المعالجة المركزية متاحة فورا عند دخول مهمة الى قائمة الانتظار (READY)
- **زيادة استغلال وحدة المعالجة المركزية** (Maximize CPU utilization) تعني إبقاء CPU مشغولة بنسبة 100% ويمكن تحقيق ذلك عبر تشغيل المهام المرتبطة فقط بوحدة المعالجة المركزية (CPU-bound) وليس المهام المرتبطة بالإدخال والاخراج (I/O-bound)
- **ضمان العدالة بين كافة المهام** (Ensure fairness for all jobs) تعني منح كل مهمة وقت متساوي على وحدة المعالجة المركزية (CPU) ووحدات الادخال والإخراج (I/O) ويمكن تحقيق ذلك من خلال عدم منح معاملة خاصة لأي مهمة بعض النظر عن خصائصها في المعالجة او الاولوية

سياسة الجدولة (Scheduling Policy)

- سياسة الجدولة نوعان:
  - **سياسة جدولة تداخلية (preemptive)** تسمح بمقاطعة المهمة وانتقال المعالجة الى مهمة أخرى وتستخدم هذه السياسة بشكل واسع في انظمة المشاركة الزمنية (time-sharing)
  - **سياسة جدولة غير تداخلية (nonpreemptive)** لا تسمح بالمقاطعات الخارجية اثناء المعالجة وبمجرد ان يبدأ تنفيذ المهمة لا يمكن مقاطعتها حتى تصدر طلب ادخال وإخراج او ينتهي التنفيذ
- في حالة دخول المهمة في حلقة لا نهائية (infinite loop) يتم إيقاف المهمة ونقلها الى قائمة FINISHED سواء كانت سياسة الجدولة تداخلية او غير تداخلية

خوارزميات الجدولة (Scheduling Algorithms)

- يعتمد مدير جدولة العمليات على خوارزمية الجدولة في تخصيص او اشغال وحدة المعالجة المركزية بناء على المعايير الأكثر أهمية في سياسة الجدولة، واغلب نظم التشغيل تركز على وقت الاستجابة
- خوارزميات الجدولة:
  - الأولوية لمن يصل أولا First-Come, First-Served (FCFS)
  - الأولوية للمهمة الاقصر Shortest Job Next (SJN)
  - الأولوية للمهمة الأقرب الى الاكتمال Shortest Remaining Time (SRT)
  - التخصيص الدوري Round Robin
  - الأولوية للمهمة الأقرب الى الموعد النهائي Earliest Deadline First (EDF)

خوارزمية FCFS

- خوارزمية FCFS هي خوارزمية جدولة غير تداخلية تعتمد على وقت وصول المهمة او العملية في المعالجة، من يصل أولا يحصل على الخدمة أولا
- تستخدم هذه الخوارزمية هيكل بيانات يدعى الطابور (queue) او من يدخل أولا يخرج أولا (FIFO)
- يمكن تطبيق خوارزمية FCFS في نظم التشغيل الدفعية لكنها غير مقبولة في النظم التفاعلية لان المستخدمين في النظم التفاعلية يفضلون السرعة في الاستجابة
- وقت الإنجاز في خوارزمية FCFS متغير بشكل كبير، لهذا السبب تعد غير مفضلة حتى في النظم التي لا تدعم البرمجة المتعددة

مثال على خوارزمية FCFS

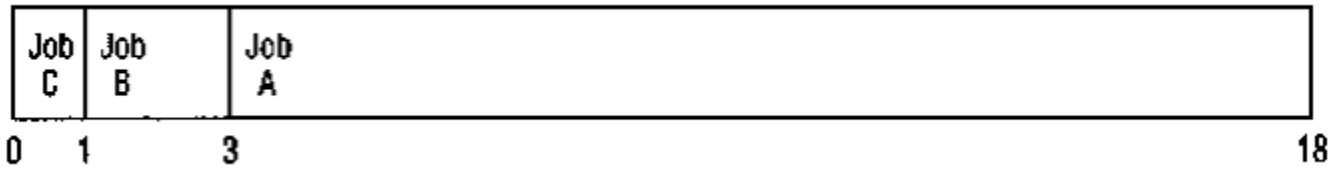
- اوجد معدل وقت الإنجاز (turnaround time) للمهام التالية (دون برمجة متعددة):
  - Job A تحتاج الى 15 ميلي ثانية في CPU
  - Job B تحتاج الى 2 ميلي ثانية في CPU
  - Job C تحتاج الى 1 ميلي ثانية في CPU
- اذا وصلت المهام الثلاثة في نفس الوقت

- وقت الإنجاز للمهمة = وقت الانتهاء – وقت الوصول
- معدل وقت الإنجاز

$$\frac{(15 - 0) + (17 - 0) + (18 - 0)}{3} = 16.67$$

مثال على خوارزمية FCFS

- اوجد معدل وقت الإنجاز (turnaround time) للمهام التالية:
  - Job A تحتاج الى 15 ميلي ثانية في CPU
  - Job B تحتاج الى 2 ميلي ثانية في CPU
  - Job C تحتاج الى 1 ميلي ثانية في CPU
- اذا كان ترتيب وصول المهام الثلاثة كالتالي: C، B، A



- وقت الإنجاز للمهمة = وقت الانتهاء – وقت الوصول
- معدل وقت الإنجاز

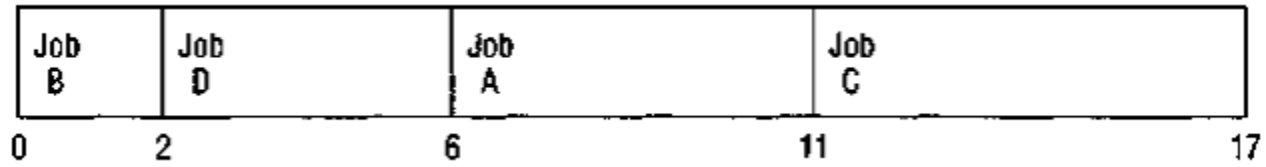
$$\frac{(1 - 0) + (3 - 0) + (18 - 0)}{3} = 7.3$$

خوارزمية SJN

- خوارزمية SJN هي خوارزمية جدولة غير متداخلة تعتمد على وقت المعالجة للمهمة، المهمة الأقصر أولا (shortest job first)
- يمكن تطبيق خوارزمية SJN في النظم الدفعية حيث يمكن تقدير الوقت قبل بدأ كل مهمة، ولا تطبق في النظم التفاعلية لان المستخدمين لا يستطيعون تقدير الوقت المطلوب لمهامهم
- توفر خوارزمية SJN ادنى معدل لوقت الانجاز اذا كانت كافة المهام متوفرة في نفس الوقت وتمكنت وحدة المعالجة المركزية من تقدير الوقت المطلوب لكل مهمة بدقة

مثال على خوارزمية SJN

- اوجد معدل وقت الإنجاز للمهام التالية علما ان وقت وصولها الى قائمة READY هو 0:
  - Job A تحتاج الى 5 ميلي ثانية في CPU
  - Job B تحتاج الى 2 ميلي ثانية في CPU
  - Job C تحتاج الى 6 ميلي ثانية في CPU
  - Job D تحتاج الى 4 ميلي ثانية في CPU
- وفقا لخوارزمية الأولوية للمهمة الأقصر سوف يصبح ترتيب المهام كالتالي: B، D، A، C



- معدل وقت الإنجاز

$$\frac{(2-0)+(6-0)+(11-0)+(17-0)}{4} = 9.0$$

الجدولة بالأولوية

- خوارزمية الجدولة بالأولوية (Priority Scheduling) هي خوارزمية غير تداخلية، تتيح معالجة البرامج ذات الأولوية الأعلى أولا (highest priority first)
- تعد خوارزمية الجدولة بالأولوية من اكثر خوارزميات الجدولة شيوعا في النظم الدفعية
- عند تساوي اكثر من مهمة في الأولوية يتم تخصيص وحدة المعالجة المركزية (CPU) للمهمة التي وصلت أولا (FCFS)
- مدير العمليات يعمل مع اكثر من طابور اعتمادا على أولوية المهمة بدلا من طابور واحد
- يمكن تعيين الأولوية للمهام من قبل مشرف النظام باستخدام بعض الخصائص مثل نوع المستخدم، وفي الأنظمة التجارية يمكن منح أولوية اعلى لمن يدفع اكثر
- يمكن تعيين الأولوية ايضا من قبل مدير المعالجة اعتمادا على خصائص المهمة مثل كمية الذاكرة المطلوبة واجمالي وقت المعالجة ونوع الأجهزة الطرفية وعددها ومدة الانتظار

خوارزمية SRT

- خوارزمية SRT هي خوارزمية جدولة تداخلية تعتمد على تخصيص المعالج للمهمة الأقرب الى الاكتمال أي تسمح بمقاطعة المهمة الحالية اذا كانت المهمة الجديدة وقتها المتبقي اقل
- يمكن تطبيق خوارزمية SRT في النظم الدفعية، ولا تطبق في النظم التفاعلية حيث لا يمكن تقدير الوقت المطلوب لإنهاء كل مهمة
- من عيوب خوارزمية SRT انها تزيد من أعباء نظام التشغيل لأنها تتطلب مراقبة وقت المعالجة لكافة المهام في قائمة الانتظار وتبديل سياق التنفيذ (context switching) بين المهام عند المقاطعة
- تبديل سياق التنفيذ هو حفظ معلومات المعالجة في كتلة التحكم بالعملية (PCB) قبل خروجها من المعالج وتحميل معلومات المعالجة من كتلة التحكم بالعملية الاخرى حتى يتمكن المعالج من تنفيذها

مثال على خوارزمية SRT

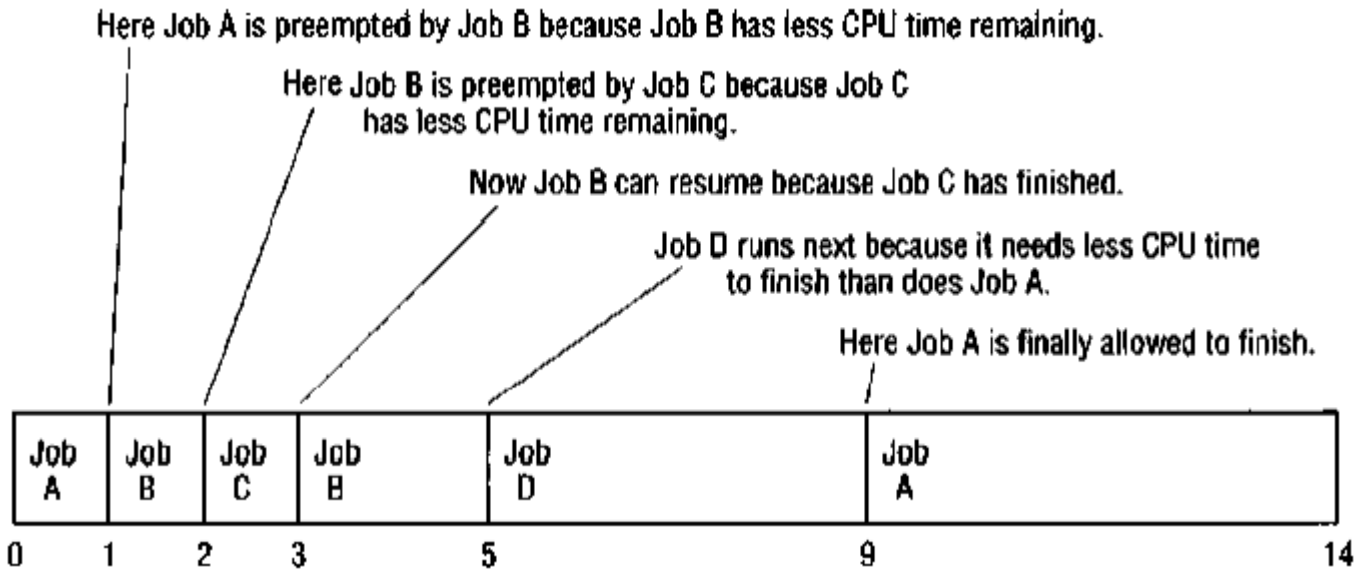
- اوجد معدل وقت الإنجاز للمهام التالية:

وقت الوصول (Arrival time)	المهمة (Job)	وقت المعالجة (CPU time)
0	A	6
1	B	3
2	C	1
3	D	4

مثال على خوارزمية SRT

- وقت الإنجاز (turnaround time) = وقت الانتهاء – وقت الوصول
- معدل وقت الإنجاز

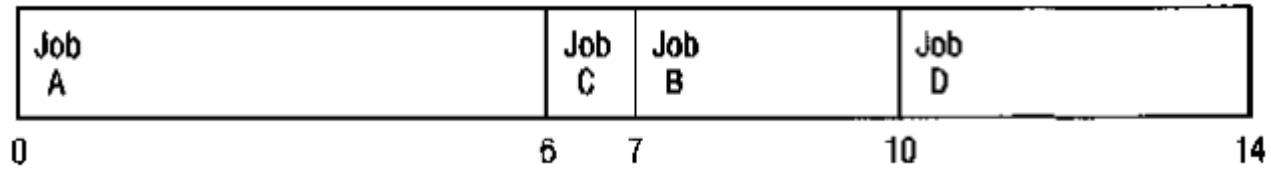
$$\frac{(14 - 0) + (5 - 1) + (3 - 2) + (9 - 3)}{4} = 6.25$$



مثال على خوارزمية SRT

- اوجد معدل وقت الإنجاز باستخدام سياسة جدولة غير تداخلية (nonpreemptive SJN)؟
- معدل وقت الإنجاز

$$\frac{6+9+5+11}{4} = 7.75$$



- اوجد معدل وقت الانتظار (waiting time)؟
- وقت الانتظار = وقت البدء – وقت الوصول
- وقت الانتظار = وقت الإنجاز – وقت المعالجة
- معدل وقت الانتظار = 4 / ((6-6) + (9-3) + (5-1) + (11-4)) = 4.25

خوارزمية Round Robin

- خوارزمية Round Robin هي خوارزمية جدولة تداخلية تعتمد على تخصيص فترة زمنية لكل مهمة بشكل دوري، ويطلق على هذه الفترة الزمنية time quantum وتتراوح قيمتها من 100 ميلي ثانية وحتى 1 او 2 ثانية
- يمكن تطبيق خوارزمية Round Robin في النظم التفاعلية بسهولة لأنها لا تعتمد على خصائص المهمة وانما تجزئة الوقت بشكل دوري
- يتم وضع المهام في قائمة انتظار تعمل وفق خوارزمية الأولوية لمن يصل أولا (FCFS)، وبعد اختيار المهمة من القائمة وتخصيص المعالج لها، يعمل مؤقت على تحديد الفترة الزمنية للمهمة، واذا انتهت الفترة ولم يكتمل تنفيذ المهمة يتم اخراج المهمة ووضعها في نهاية القائمة وحفظ معلومات المعالجة في كتلة التحكم بالعملية (PCB)

مثال على خوارزمية Round Robin

- اوجد معدل وقت الإنجاز للمهام التالية مع فترة زمنية مقدارها 4 ميلي ثانية:

وقت المعالجة (CPU time)	المهمة (Job)	وقت الوصول (Arrival time)
8	A	0
4	B	1
9	C	2
5	D	3

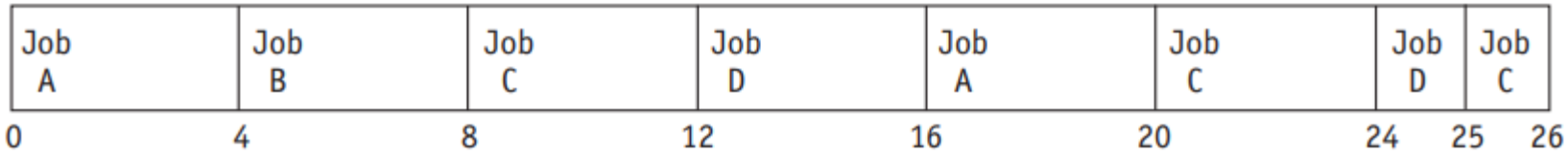
مثال على خوارزمية Round Robin

- وقت الإنجاز لكل مهمة

A	B	C	D
20-0	8-1	26-2	25-3
20	7	24	22

- معدل وقت الإنجاز

$$\frac{20 + 7 + 24 + 22}{4} = 18.25$$



خوارزمية Round Robin

- يعتمد اداء خوارزمية Round Robin على طول الفترة الزمنية ومعدل وقت المعالجة:
  - اذا كانت الفترة الزمنية أطول من وقت المعالجة لأغلب المهام تتحول الخوارزمية الى FCFS
  - اذا كانت الفترة الزمنية قصيرة جدا يؤدي تبديل سياق التنفيذ الى ابطاء تنفيذ كافة المهام وزيادة العبء
- ما هي افضل قيمة لطول الفترة الزمنية في خوارزمية Round Robin؟
- تعتمد الإجابة على نوع النظام، اذا كانت البيئة حرجة للوقت يتوقع من النظام ان يرد فورا لكن هنالك قاعدتين في اختيار طول الفترة الزمنية هما:
  - ان تكون الفترة كافية لانهاء 80% من وقت المعالجة للمهام
  - أن تكون الفترة أطول بمقدار 100 ضعف الوقت المطلوب في تبديل سياق التنفيذ

تمرين

- اوجد معدل وقت الإنجاز للمهام التالية لكل خوارزمية جدولة:

وقت الوصول (Arrival time)	المهمة (Job)	وقت المعالجة (CPU time)
0	A	15
2	B	2
3	C	14
6	D	10
9	E	1

- الأولوية لمن يصل أولا (FCFS)
- المهمة الأقصر أولا (SJN)
- المهمة الأقرب الى الاكتمال (SRT)
- التخصيص الدوري (Round Robin) مع فترة زمنية مقدارها 5 ميلي ثانية

قوائم الانتظار متعددة المستويات

- تستخدم قوائم الانتظار متعددة المستويات (multiple-level queues) مع خوارزميات الجدولة المختلفة للاستفادة من ميزات كل خوارزمية في بعض النظم مثل:
  - النظم القائمة على الأولوية التي لديها قائمة انتظار لكل مستوى من الأولوية
  - النظم التي تجمع كافة المهام المرتبطة بالمعالج (CPU-bound) في قائمة انتظار والمهام المرتبطة بالإدخال والإخراج (I/O-bound) في قائمة انتظار أخرى
  - النظم الهجينة التي تدعم المهام التفاعلية والدفعية، المهام الدفعية توضع في قائمة انتظار تدعى الخلفية (background queue) والمهام التفاعلية توضع في قائمة انتظار أخرى تدعى المقدمة (foreground queue) وعادة تفضل هذه النظم المهام في المقدمة على المهام في الخلفية
- تعتمد سياسة الجدولة في قوائم الانتظار متعددة المستويات على التعامل مع المهام في كل قائمة انتظار بشكل مختلف بناء على خصائصها

قوائم الانتظار متعددة المستويات

- ما هي السياسة التي يتبعها النظام في نقل المهام بين قوائم الانتظار؟
- هنالك أربعة سياسات:
  - عدم نقل المهام من قائمة الى أخرى قبل الانتهاء من كافة المهام في القائمة ذات الأولوية الأعلى
  - نقل المهام من قائمة الى أخرى عبر تعديل الأولوية التي تم تعيينها للمهام مسبقا
  - تعيين فترة زمنية مختلفة لمعالجة المهام في كل قائمة (variable time quantum)
  - التقادم (aging) اي تعديل أولوية المهام بناء على اقدميتها او مدة انتظارها في القائمة

خوارزمية EDF

- خوارزمية EDF هي خوارزمية جدولة تداخلية تعتمد على تخصيص المعالج للمهمة الأقرب الى الموعد النهائي او الأقرب الى انتهاء المهلة المحددة لها (closer to deadline)
- يطلق عليها أيضا خوارزمية الأولوية الديناميكية لان اولوية المهام يمكن تعديلها اثناء التنفيذ
- يمكن تطبيق خوارزمية EDF في أنظمة الوقت الحالي او الفعلي التي لديها مواعيد نهائية (deadlines) في معالجة المهام او البرامج
- الهدف من خوارزمية EDF هو معالجة كافة المهام بترتيب يؤدي الى انهاء تنفيذها قبل وصول كل مهمة الى موعدها النهائي
- عندما تشترك اكثر من مهمة بنفس الموعد النهائي، يمكن اللجوء الى سياسة جدولة أخرى مثل الأولوية للمهمة التي تصل أولا (FCFS)

مثال على خوارزمية EDF

المهمة (Job)	وقت الوصول (Arrival time)	وقت المعالجة (CPU time)	الموعد النهائي (Deadline)
A	0	3	6
B	1	1	2
C	2	5	10
D	3	2	8
E	3	2	8
F	5	5	15