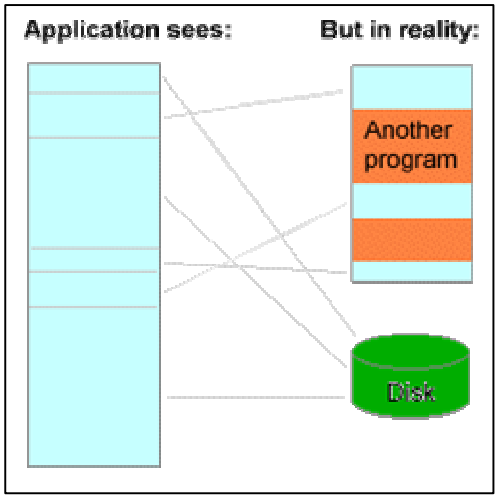
أهداف الفصل

أولاً: خلفية

إن مبدأ تشغيل البرامج بدون تحميلها كاملةً إلى الذاكرة العشوائية يحمل في طيّه الكثير من الفوائد سواء على مستوى النظام أم على مستوى المستخدم، ففي هذه الحالة يمكن للمبرمجين برمجة العديد من البرامج الطويلة دون الأخذ بعين الاعتبار محدودية الذاكرة العشوائية للأجهزة المستهدفة، بالتالي تسهيل عملية صنع البرامج، كما أن المستخدمون لأنظمة تطبّق مبادئ الذاكرة الافتراضية يمكنهم العمل على عدة برامج في نفس الوقت، وإن كانت الذاكرة العشوائية لا تتحمل حجم جميع هذه البرامج مجتمعة.



إن تقنية الذاكرة الافتراضية تطبق مبدأ الفصل بين مفهومين للمساحة التي تمثّل البرنامج (الأوامر والبيانات): مساحة البرنامج الافتراضية التي يراها المبرمج أو المستخدم (virtual address space)، ومساحته الفعلية (actual address space) التي يطبّقها الجهاز عند تحميل وتشغيل البرنامج، إذ أن المبرمج يبرمج برنامجه واضعاً بعين الاعتبار مساحة افتراضية كبيرة إلى حد ما لبرنامجه، بينما في الواقع عند تشغيل هذا البرنامج فإن مساحته الفعلية في الذاكرة لا تطابق المساحة الافتراضية بل تغايرها من نواحي عدّة، فالمساحة الافتراضية –كما يتضح من اسمها– تفترض أن البرنامج مخزّن في الذاكرة ابتداءً من عنوان افتراضي (logical address) كصفر مثلاً، ثم تمتدّ مساحة البرنامج بشكل مستمرّ ومتصل، لكنّ مبدأ تقسيم البرامج إلى صفحات (pages) في فصل الذاكرة الرئيسية، يدلّنا على أن المساحة الواقعية للبرنامج مقسّمة إلى صفحات كما أن تخزين هذه الصفحات ليس متصلاً بل بحسب ما يتوفر لنا من أماكن شاغرة في الذاكرة الرئيسية، كما أن عنوان الصفحات الافتراضي سيتم ترجمته إلى عنوان حقيقي من قبل وحدة إدارة الذاكرة (Memory Management Unit).

(أفنان السبيهين)

وبذلك نستنتج أن هذا النوع من الذاكرة يعطي صورة للمستخدم بأن البرنامج تم تحميله بالكامل على الذاكرة الرئيسية خلال فترة المعالجة، و تعطي البرنامج التطبيقي الانطباع بأن هناك ذاكرة متصلة يعمل عليها، بينما في الحقيقة البرنامج مجزّأ إلى أجزاء وعند تحميل البرنامج لمعالجته يتم تحميل بعض هذه الأجزاء

للذاكرة والبعض الآخر يتم وضعه في مساحة في القرص الصلب (مساحة داعمة – backing store).

الذاكرة التخيلية ليست عبارة عن استخدام مساحات في القرص الصلب لتوسعة مساحة الذاكرة الحقيقة فقط, ولكنها تهدف إلى مخادعة البرامج لتظن أنها تستخدم مساحات كبيرة ومتجاورة من الذاكرة استعانةً بالقرص الصلب، بالتالي فإن القرص الصلب يوصف بأنه في هذه الحالة يتظاهر بأنه جزء من الذاكرة العشوائية، ويوهم البرامج بأنه امتدادٌ لها.[53]

(سارة الشثري)

يقوم نظام التشغيل بالبحث عن الأجزاء الغير مستعملة باستمرار من الذاكرة العشوائية ثم يقوم بنسخها في القرص الصلب وبذلك فانه يتيح مساحه أكبر في الذاكرة العشوائية حتى يتمكن نظام التشغيل من استخدام جزءاً أكبر من الذاكرة العشوائية في تشغيل تطبيقاته.

عملية التبديل التي تتم بين الذاكرة العشوائية و القرص الصلب تتم في شفافية تامة فمستخدم الجهاز يكاد أن لا يشعر بهذه العملية، وبذلك نكون قد حظينا بجزء أكبر من الذاكرة العشوائية وأيضا زيادة في عدد التطبيقات التي يعمل عليها المستخدم في الجهاز.

(نوال باعبد الله)

هذه التقنية تعطي عدة تسهيلات، فالنظام الذي يطبق هذا المبدأ يعمل بشكل جيد مع بيئة البرامج المتعددة، كما أنه يتم تقليل وقت الانتظار بشكل كبير في ظل وجود هذه الذاكرة، كما أن من مميزات تقنية الذاكرة الافتراضية أن حجم البرنامج لا يؤثر كثيراً في عدد البرامج التي يمكن تحميلها للذاكرة، بالتالي إعطاء لا محدودية في عدد البرامج المتزامنة، مما يعطينا استخداماً أكثر كفاءة للذاكرة، و لا يمكننا تجاهل ما تقدمه هذه التقنية من تسهيلٍ لعملية مشاركة الأوامر أو البيانات بين البرامج التي تقع في الذاكرة.

بعدد ما تقدمه هذه التقنية من فوائد ومميزات فإنها تحتوي عدداً من العيوب، فهذه التقنية قد تكون مكلفة من نواحٍ عدة، إما من ناحية المساحة الداعمة (backing store) أو من ناحية الوقت، فهي تزيد بشكل لا يمكن تجاهله عدد مرات مقاطعة البرامج، كما أنها من الممكن أن تزيد تعقيد مهمة البرمجة.[54]

---

[1] en.Wikipedia.org

1 Microcomputer Operating Systems book, Operating System Concept book.

(أمينة العبيد)

ثانياً: طلب الصفحات

عند القيام باستخدام برنامج ما فان هذا البرنامج يتم تقسيمه لصفحات Paging وهذه الصفحات Pages يتم تحميلها للذاكرة لكي يتم تنفيذ هذا البرنامج، في آلية طلب الصفحات في الذاكرة التخيلية Demand Paging in Virtual Memory لا يتم تحميل جميع صفحات البرنامج المشغل إلى الذاكرة الفعلية، إنما فقط الصفحات المطلوبة لتنفيذ العمليات التي يقوم بها المستخدم لهذا البرنامج، وهذه الآلية تعتبر أفضل بكثير من تحميل كامل البرنامج إلى الذاكرة (جميع صفحات البرنامج)، وذلك لأن أجزاء من البرنامج قد لا تستخدم خلال تشغيله، مما يعني حجز جزء من الذاكرة لصفحات من البرنامج غير مستخدمه، هذه الآلية تمكننا من تحميل عدة برامج كبيرة في آن واحد نظرا لأن كل برنامج لن يتم تحميله بالكامل، مما يوفر مساحة في الذاكرة لبرامج أخرى.

وحين يريد المستخدم استخدام جزئية جديدة من البرنامج موجودة في صفحات لم يتم تحميلها إلى الذاكرة، سيتم تحميل الصفحات المطلوبة عن طريق المبدل الكسول Lazy Swapper أو بالأصح Pager حيث إن لفظ Swapper يستخدم حينما يتم التعامل مع كل البرنامج دون تقسيمه إلى صفحات، ولفظ Pager يستخدم حين يتم التعامل مع الصفحات Pages كل على حدة.

و حينما يحاول برنامج ما الدخول على صفحة غير موجودة في الذاكرة الفعلية سينتج عن ذلك ما يعرف بخطأ الصفحة (Page-fault) مما يجعل الـ Pager يجلب الصفحات المطلوبة من القرص الصلب إلى الذاكرة لكي يتم استخدامها.

عند تطبيق هذه الآلية يمكننا أن نبدأ باستخدام برنامجٍ ما دون أن تكون أي من صفحاته قد تم تحميلها للذاكرة الفعلية، وخلال تنفيذ البرنامج يتم تحميل ما يحتاجه المستخدم من صفحات للذاكرة لتنفيذ العمليات المطلوبة.[55]

(نورة سلمان بن سعيد – حليمة حكمي)

مبادئ رئيسية

[55] En.wikipedia.org and www.science.unitn.it

نستخلص مما سبق أن المبدأ الأساسي لتقنية طلب الصفحات هو عدم تحميل جميع صفحات البرنامج من الذاكرة الثانوية (Secondary Storage) أو القرص إلى الذاكرة الحقيقية (Physical Memory) مباشرة عند بداية تنفيذ البرنامج، بل تحميلها حسب الحاجة **أثناء** التنفيذ.

عندما يتم تحميل (تبديل) البرنامج إلى الذاكرة الحقيقية للمرة الأولى فإن المصفح يحاول التنبّؤ بالصفحات التي سيتم استخدامها قبل إرجاعه إلى الذاكرة الثانوية مرة أخرى، هذه الصفحات هي التي سيتم إحضارها إلى الذاكرة الحقيقية بدلاً من إحضار كامل البرنامج، وبذلك تم اختصار الوقت الذي كان سيقضيه لو أنه اضطر إلى جلب عدد أكبر من الصفحات، فضلاً عن اختصار المساحة التي كانت ستملؤها تلك الصفحات.

هنا تظهر لنا الحاجة لمعرفة كافّة الصفحات التي تخص البرنامج: أيّها في الذاكرة الحقيقية وأيّها في الذاكرة الثانوية (لم يتم جلبها بعد)، ولحل هذه المشكلة نحتاج إلى دعم الجهاز لإضافة bit إضافي في الجدول الذي يربط بين كل صفحة و مكانها في الذاكرة الحقيقية (Page Table). هذه الخانة الإضافية تعطي إحدى قيمتين:

1- صحيح (valid) : تعني أن هذه الصفحة موجودة حاليّاً في الذاكرة الحقيقية.
2- غير صحيح (invalid) : تحتمل إحدى معنيين:
   أ. الصفحة خاطئة (لا يحتويها البرنامج أساساً).
   ب. الصفحة موجودة لكن لم يتم تحميلها إلى الذاكرة الحقيقية بعد. (لا تزال في القرص)

ويعرف النظام ما إذا كانت الصفحة المطلوبة موجودة على القرص (لم تجلب بعد) بإحدى طريقتين:

1- مكان الصفحة في الجدول محدد بعلامة (invalid).
2- مكان الصفحة في الجدول مربوط بعنوان لا يوجد في الذاكرة الحقيقية بل يوجد في القرص.

خطأ الصفحة

ذكرنا أنه عند بداية تنفيذ البرنامج للمرة الأولى فإن المُصفّح (pager) يحاول التنبؤ بالصفحات التي سيتم استخدامها، المُصفح يحاول جاهداً أن يكون هذا التنبؤ صحيحاً، إذ لن يحتاج إلى جلب صفحات إضافية من القرص أثناء التنفيذ. لكن لو أن النظام أثناء التنفيذ أراد صفحة لم تكن ضمن الصفحات التي تنبأ بها،

أي أنها لا تزال في القرص، فإنه عند الاستعانة بالجدول سيجد أن هذه الصفحة "غير صحيحة" بالتالي سيتسبب في حصول تعثّر (Trap) خاص يدعى **بخطأ الصفحة** (Page-Fault Trap) ، هذا التعثر حاصل نتيجة تقصير النظام في إحضار صفحات البرنامج إلى الذاكرة.

حينها سيضطر النظام للقيام بعدة خطوات للتعامل مع هذا التعثر: سيضطر المصفح إلى إحضار هذه الصفحة من القرص إلى الذاكرة،ثم تسكينها في مكان (Frame) خالٍ في الذاكرة، ثم سيعدل الجدول ليعكس وجود هذه الصفحة، ثم أخيراً يتم إعادة العملية التي طلبت هذه الصفحة.

نأخذ في الاعتبار حالة خاصة، وهي أن النظام لن يجلب أي صفحة تخص البرنامج عند بداية تنفيذه، أي أن كل أمر يتم تنفيذه في هذا البرنامج سيتسبب في خطأ الصفحة (Page-Fault) ، إلى أن يتم جلب جميع الصفحات التي يحتاجها ثم سيكمل تنفيذه بشكل طبيعي، تدعى هذه الحالة **بطلب الصفحات النقي** (Pure Demand Paging).[56]

(أفنان السبيهين)

أنواع خطأ الصفحة

- **خطأ الصفحة الطفيف**: وهو أن تكون الصفحة المطلوبة موجودة في الذاكرة ولكن لم يتم تحديث الجدول ليعكس وجودها، وتحدث هذه الحالة إذا كان هناك ذاكرة مشتركة بين برنامجين أو أكثر فتكون الصفحة المطلوبة جاءت إلى الذاكرة عن طريق البرنامج الآخر.

- **خطأ الصفحة العظيم**: وهو أن تكون الصفحة المطلوبة غير محملة في الذاكرة وهذا الخطأ مكلف لأننا نأخذ بالاعتبار الوقت الذي يستغرقه جلب الصفحة من القرص.

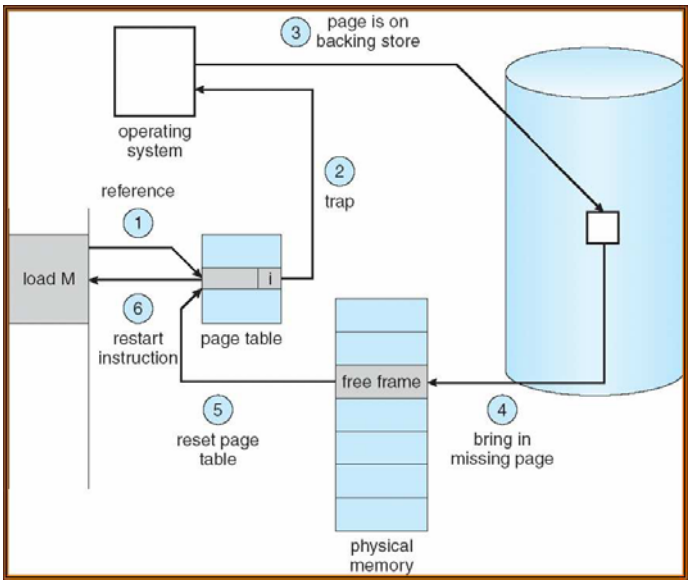
- **خطأ الصفحة الباطل**: ويحدث هذا الخطأ عند محاولة القراءة من مكان فارغ في الذاكرة.

- **خطأ الحماية**: ويحدث هذا الخطأ عندما لا يستطيع البرنامج القراءة أو الكتابة في الصفحة المطلوبة ويحل نظام التشغيل مشكلة الكتابة بواسطة آلية النسخ عند الكتابة (copy-on-write)، وهو مشاركة نفس النسخة بين أكثر من برنامج ,وعند ما يريد أحد البرامج التحديث يعمل نظام التشغيل نسخة خاصة به .[57]

---

[56] Operating System Concepts by Silberschatz, Galin and Gange.

[57] En.wikipedia.org



خطوات التعامل مع خطأ الصفحة:

أولاً: يتم التأكد من الجدول الداخلي (عادة يكون محفوظ مع الـــPCB) للعملية ؛ ليتم تحديد ما إذا كانت الصفحة المطلوبة صحيحة (valid) أم غير صحيحة (invalid)، اعتماداً على ما إذا كانت تم تحميلها للذاكرة أم لا.

ثانياً: إذا كانت الصفحة غير صحيحة لكونها محمية من ذلك البرنامج (لا تنتمي له) إذن سيتم إنهاء العملية، أما إذا كانت الصفحة صحيحة لكن غير موجودة في الذاكرة إذن يتم إحضارها.

ثالثاً: يتم البحث عن إطار (frame) فارغ، لتوضع به الصفحة التي سيتم جلبها، وذلك بأخذه من قائمة الإطارات الفارغة (free-frame).

رابعاً: تتم جدولة عمليات القرص ليقرأ الصفحة المطلوبة إلى الإطار الجديد.

خامساً: عندما ينهي القرص عملية القراءة، يقوم النظام بتعديل الجدول الداخلي للعملية و جدول الصفحة، دليلاً على وجود الصفحة في الذاكرة.

سادساً: يتم إعادة تنفيذ الأمر الذي قُطع بتعثر نظام التشغيل.

الآن العملية تستطيع الوصول إلى الصفحة بما أن الصفحة جلبت إلى الذاكرة.[58]

(ليلى البيشي – نهلة محمد)

أداء تقنية طلب الصفحات

ثالثاً: النسخ عند الكتابة Copy-on-Write

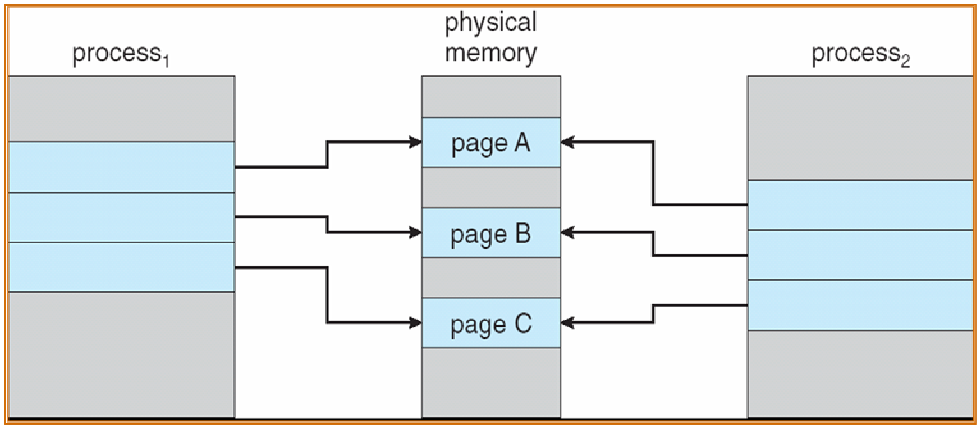
نعرف من فصول سابقة أنه عندما يستخدم برنامج ما الـــ ()Fork فإنه ينتج برنامجاً آخراً، هذا البرنامج الجديد (الابن) يكون نسخة طبق الأصل من البرنامج الذي أنتجه (الأب)، أي أن صفحات الأب يتم

[58] Operating System Concepts by Silberschatz, Galin and Gange

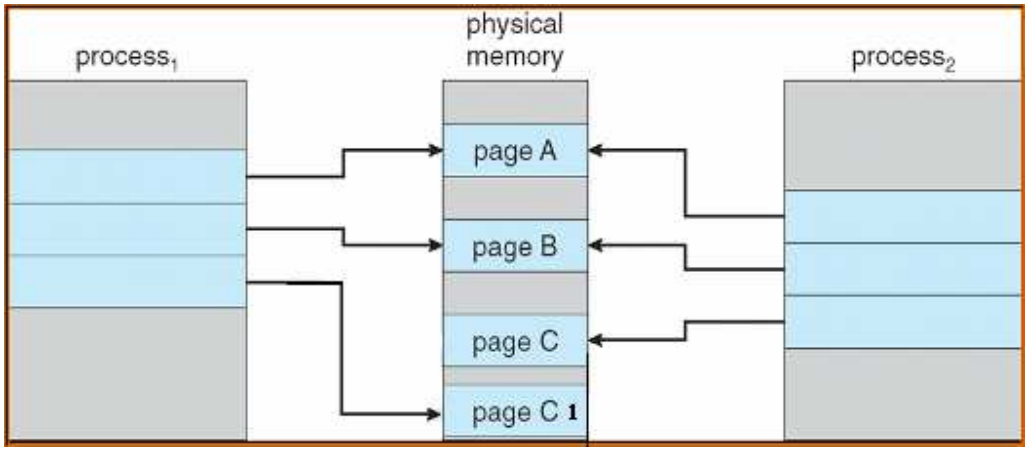
نسخها كما هي لتكون صفحات الابن، يتضح لنا ما في هذه العملية من إهدار للمصادر إذ أن الابن نادراً ما يُترك ليكون نسخة عن أبيه بل سيستخدم الـــ exec() فلا تكون له نفس صفحات أبيه.
تقنية النسخ عند الكتابة (COW)، تسمح لبرنامجين الاشتراك في نفس الصفحات دون أن تسند نسخة لكل برنامج، إلى إن يحاول أحد هذين البرنامجين التعديل على إحدى الصفحات المشتركة، حينها يتم نسخ تلك الصفحة له، ليعدّل عليها كيفما شاء.

مثال:

لنفرض أن كل من (process1 - child) و (process2- parent) يؤشران على الصفحات A و B و C



عند محاولة (process1) التعديل على الصفحة (C) ، ننتج نسخة جديدة من هذه الصفحة لتعكس هذه التعديلات، ونجعلها ملكاً لـــ(process1)، فيؤشر عليها.



(خلود الرومي)

رابعاً: استبدال الصفحات

علمنا أنه عند تطبيق تقنية طلب الصفحات، قد تحتاج أحد البرامج التي يعمل عليها المعالج حاليّاً إلى صفحة لم يتم تحميلها إلى الذاكرة، مما ينتج لنا تعثراً أسميناه بخطأ الصفحة، ونعرف أيضاً أن خطأ الصفحة يمكن معالجته بأن يتم إحضار هذه الصفحة من القرص إلى الذاكرة، لا يمكننا تجاهل المشكلة الأساسية التي قد يواجهها النظام عند هذه النقطة، ألا وهي عدم وجود مكان فارغ في الذاكرة، لذلك يطبق النظام تقنيّات على أساسها يختار صفحة (الضحيّة) ليستبدلها بالصفحة المطلوبة.

(أفنان السبيهين)

1-أساس استبدال الصفحات

إذاً فخوارزميات تبديل الصفحات هي التي تختار وتقرر الصفحة التي تخرج من الذاكرة عندما لا يكون هناك إطاراً (frame) فارغاً في الذاكرة، وتوضع الصفحة الخارجة (victim) في القرص (back store)، وتهدف هذه الخوارزميات إلى تقليل نسبة حدوث خطأ الصفحة، فكلما زادت عدد الإطارات بالذاكرة كلما قلت نسبة خطأ الصفحة.

يوجد في الحقيقة عدة خوارزميّات للتعامل مع هذه الحالة، الاختلاف الوحيد بين هذه الخوارزميات هي طريقة اختيار الضحية، وبينما تختلف فيما بينها في هذه الخطوة إلا أنها تتفق في بقية الخطوات المتبعة للتعامل مع الاستبدال هذه الخطوات تعرف بأساس استبدال الصفحات (basic replacement).

(حليمة حكمي)

هذه الخطوات هي:

1- البحث عن موقع الصفحة المطلوب جلبها من القرص.

2- البحث عن إطار فارغ في الذاكرة:

a. إذا وجد إطار فارغ في الذاكرة، يتم استخدامه.

b. إذا لم يجد إطاراً فارغاً يتبّع إحدى خوارزميات تبديل الصفحات لاختيار الإطار الضحية.

c. يتم نقل محتويات الإطار الضحية إلى القرص، ويتم تعديل جدول الصفحات لتعكس عدم وجود هذه الصفحة في الذاكرة.

3- يتم إحضار الصفحة المطلوبة إلى الإطار الذي تمّ تفريغه في الذاكرة.

4- إعادة تنفيذ الأمر الذي طلب هذه الصفحة.

(صفا البلاع)

نلاحظ أنه في حالة عدم وجود مكان فارغ في الذاكرة، فإن النظام سيضطرّ أن ينقل صفحتين (صفحة خارجة للقرص وصفحة داخلة للذاكرة)، بالتالي فإن معالجة خطأ الصفحة يستهلك ضعف الوقت في حال عدم وجود إطار خالٍ في الذاكرة.

لكن لو أخذنا بعين الاعتبار أن الصفحة المطلوبة يتم **نسخها** من القرص ثم وضعها في الذاكرة، أي أن القرص يحتوي نسخة منها، حينها فلن نضطر لإرجاعها للقرص لو تم اختيارها لاحقاً لتكون ضحية، إلا إذا كان قد تم تعديلها، بالتالي يجب نقلها للقرص لتعكس الصفحة المخزنة فيه هذه التعديلات، لتطبيق هذه الطريقة نضيف خانة في الصفحة ونسمّيها خانة التعديل (modify bit or dirty bit)، بحيث أنه إذا تم التعديل على هذه الصفحة خلال تنفيذ البرنامج فإن هذه الخانة تحمل القيمة 1، وإذا لم يتم التعديل فإن الخانة قيمتها صفر، وهكذا إذا اختيرت صفحة لتكون ضحية فإن النظام يختبر خانة التعديل، إذا كانت الصفحة غير معدّلة (قيمة الخانة صفر) فإنه لا يجب نقلها إلى القرص، بل يتم استبدالها بالصفحة المطلوبة مباشرة، أما إذا كانت الصفحة الضحية معدّلة (قيمة الخانة واحد) فإنه يتم نقلها للقرص قبل أن يتم استبدال إطارها بالصفحة المطلوبة، وبهذا استطعنا أن نقلل من الوقت المطلوب لحلّ خطأ الصفحة إلى النصف **لو** كانت الصفحة لم يتم التعديل عليها من قبل.

وبهذا لإكمال تطبيق تقنية طلب الصفحات، نحتاج إلى **خوارزمية إسناد الإطارات (frame-allocation algorithm)** لتحديد كميّة الإطارات المسندة إلى برنامج ما في الذاكرة بحيث أن لا نسند للبرنامج أكثر مما يحتاج (over allocation)، أو أقل مما يحتاج، كما نحتاج أيضاً إلى **خوارزمية لاستبدال الصفحات (page replacement algorithm)**.

كما قلنا سابقاً يوجد العديد من خوارزميات استبدال الصفحات، كل خوارزمية لها طريقتها الخاصة في اختيار الضحية في حال وجود خطأ الصفحة، ويتم تقييم واختيار واحدة منها على أساس الأقل تسببا في خطأ الصفحة، نقوم باختبار كل خوارزمية بتطبيقها على سلسلة من طلبات الصفحات (reference string)، كل صفحة تُمثّل برقم، إذاً فهي سلسة من الأرقام، ومن ثم نحسب عدد مرات حدوث خطأ الصفحة (رقم الصفحة المطلوبة غير موجود بالذاكرة) لكل خوارزمية.

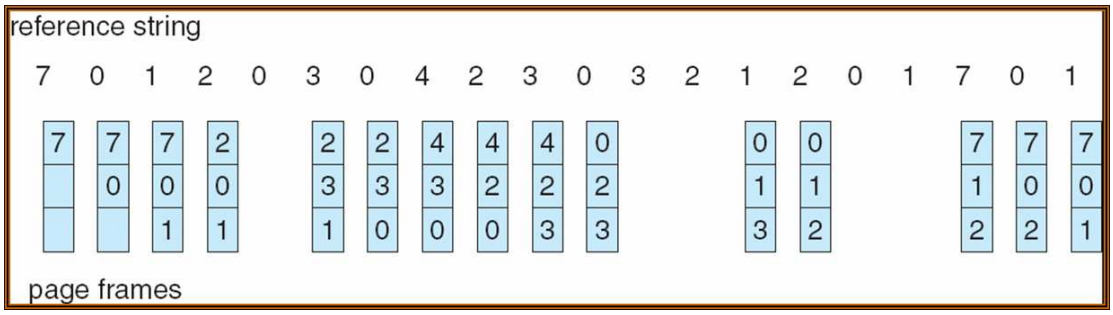
(أفنان السبيهين)

2-خوارزمية الداخل أولاً يخرج أولاً FIFO

تعتبر هذه الخوارزمية من أبسط خوارزميات تبديل الصفحة، إذ أن هذه الخوارزمية تربط كل صفحة مع الوقت الذي أُحضرت فيه للذاكرة، وعندما نختار صفحة لنستبدلها فإننا نختار أقدم صفحة دخلت الذاكرة، كما أنه ليس بالضرورة أن نسجل وقت دخول الصفحة للذاكرة، بل يكفي أن ننشئ صفّاً (FIFO queue) يمثّل كل الصفحات بالذاكرة على ترتيب قدومها، وعند إحضار صفحة للذاكرة تكون في آخر الصف (لأنها صفحة حديثة)، بالتالي فإن الصفحة التي في مقدمة الصف هي أقدم صفحة.

(خديجة أخرفي – ليلى البيشي)

مثال:



- أولاً نبدأ بثلاث إطارات فارغة في الذاكرة، سيحدث خطأ صفحة لأول ثلاث طلبات وهي 7 و 0 و 1، لكنه سيجد إطاراً فارغاً لكل واحدة منها فلا نضطر لاستخدام تقنيات استبدال الصفحات.
- عند طلب الصفحة رقم اثنين، لن يجد لها مكاناً فارغاً بالتالي سيبحث عن أقدم صفحة تم جلبها للذاكرة ليستبدلها، هذه الصفحة هي الصفحة رقم 7، بالتي ستحل صفحة 2 بدل صفحة رقم 7.
- طلب صفحة رقم صفر لن يسبب خطأ صفحة، لأن الصفحة صفر موجودة بالذاكرة أساساً.
- طلب الصفحة 3 يسبب خطأ صفحة لأنها غير موجودة بالذاكرة، ولأنه لا يوجد مكان فارغ لنجلبها له، سنضطر لتطبيق تقنية استبدال الصفحات لتحديد الضحية، الضحية في هذه الخوارزمية هي أقد صفحة تم جلبها للذاكرة، في هذه الحالة تحل الصفحة 3 بدلاً من الصفحة صفر.
- وهكذا نطبق هذه الخطوات لكل رقم في سلسلة الطلبات، في النهاية سنجد أنه عند تطبيق تقنية الداخل أولاً خارج أولاً لاستبدال الصفحات على هذه السلسلة، ينتج 15 خطأ صفحة.

(نورة الحماد)

نعرف بديهيّاً أن زيادة عدد الإطارات الفارغة في الذاكرة المعطى لكل برنامج، يمكن أن يقلل من احتمال العدد الكلي لأخطاء الصفحة التي تتسبب بها تقنية استبدال الصفحات، ولأن لكل قاعدة شذوذ، فإن الشذوذ عن هذه القاعدة يدعى **شذوذ بيلادي** (Belady's Anomaly).

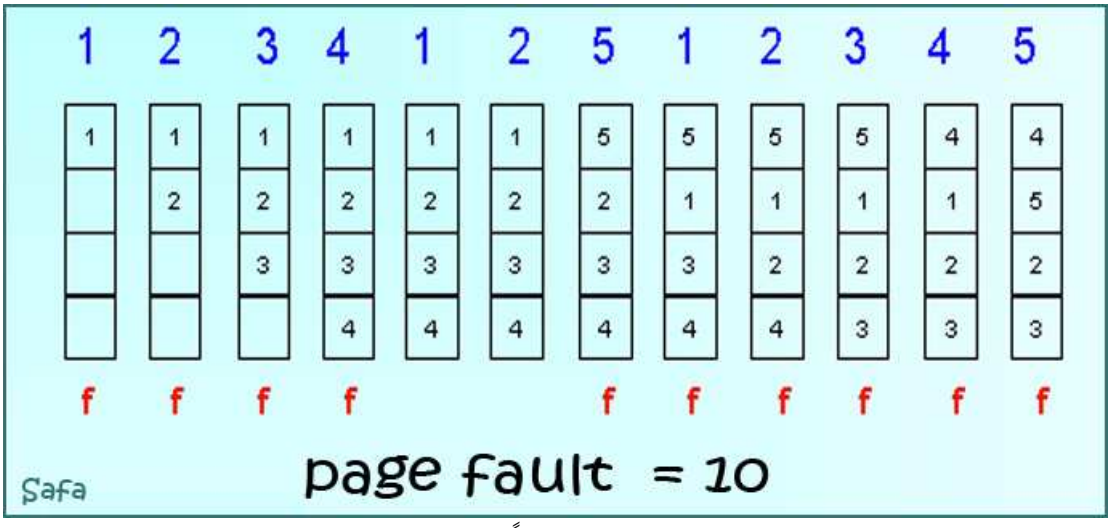
(أفنان السبيهين)

تعاني خوارزمية الداخل أولاً خارج أولاً من هذا الشذوذ، فإن زيادة عدد الإطارات المعطى لكل برنامج، في نظامٍ تطبق فيه خوارزمية الداخل أولاً خارج أولاً، لا يعني بالضرورة أن العدد الكلي لأخطاء الصفحات سيقل.

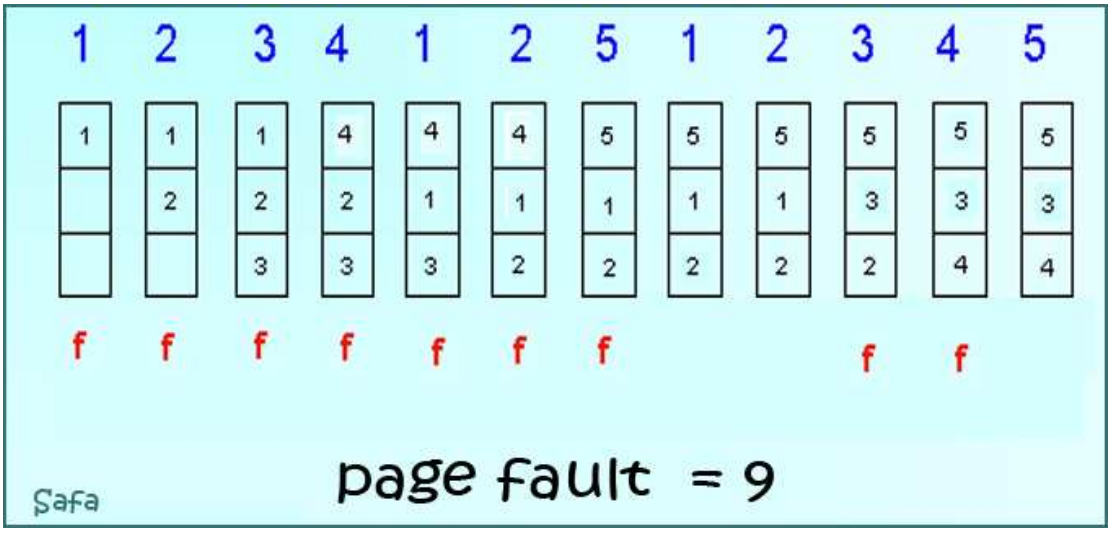
(خديجة أخرفي)

مثال:

نرى مثالاً يوضح شذوذ بيلادي، أولاً بتطبيق سلسلة الطلبات على أربعة إطارات ينتج لنا عشرة أخطاء (f).



والآن نطبق نفس السلسلة لكن على ثلاث إطارات بدلاً من أربعة، فينتج لنا 9 أخطاء فقط.



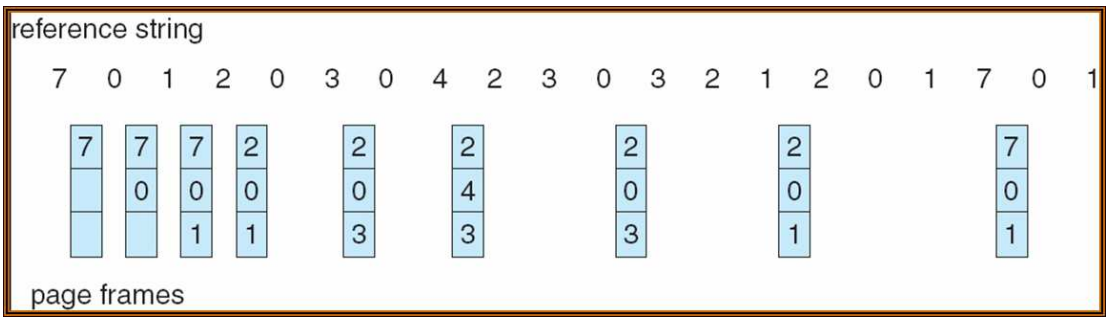
(صفا البلاع)

3- الخوارزمية المثاليّة

نتج الكشف عن شذوذ بيلادي أن أراد الباحثون الحصول على طريقة مثالية لاستبدال الصفحات، هذه الطريقة هي أقل الخوارزميات تسبباً في خطأ الصفحة كما أنها لا تعاني من شذوذ بيلادي، تقوم هذه الطريقة على اختيار الصفحة التي لن يتم استخدامها قريباً، أو سيتم استخدامها بعد فترة طويلة بالنسبة لبقية الصفحات الموجودة بالذاكرة، أي يجب أن نعرف بقية سلسلة الطلبات والذي لا يُعطى للنظام جاهزاً وإنما تحدث هذه الطلبات في أوقات زمنية متفرقة، يعني هذا أنه لنعرف أيّ الصفحات لن يتم طلبها لوقت طويل في المستقبل يجب أن نعرف ماذا سنحتاج في المستقبل، لهذا يستحيل تطبيق هذه الخوارزمية لأنها تتطلب معرفة بالمستقبل الذي لا يمكن التنبؤ به، الفائدة من هذه الخوارزمية هو استخدامها كمعيار مقارنة مع بقية الخوارزميات لمعرفة مدى فعاليتها، فكلما كانت الخوارزمية نتائجها قريبة من نتائج الخوارزمية المثاليّة كلما كانت أفضل.

(صفا البلاع – ليلى البيشي – خديجة أخرفي)

مثال:



– أولاً نبدأ بثلاث إطارات فارغة في الذاكرة، أول ثلاث طلبات ستتسبب في خطأ الصفحة، لكنها ستجد إطاراً فارغاً.

– ثانياً، يتم طلب الصفحة رقم 2، فتتسبب بخطأ الصفحة لعدم وجودها في الذاكرة، ولأنه لا يوجد إطار فارغ لتوضع فيه، سنستخدم تقنية استبدال الصفحات، بالخوارزمية المثالية وبالنظر في بقية سلسلة الطلبات، نجد أن الصفحة صفر سنحتاجها قريباً جداً، كما أننا سنحتاج الصفحة 1 قبل أن نحتاج الصفحة 7، فنستنتج أن الصفحة 7 هي التي لن نستخدمها قريباً مقارنةً بصفر وواحد، فنضع 2 بدلاً من 7.

– عند طلب الصفحة صفر لن تتسبب بخطأ صفحة لوجودها في الذاكرة.

– عند طلب الصفحة 3، سنحتاج لاختيار الصفحة الضحية، وهي الصفحة التي لن تستخدم لأطول فترة من الزمن بين الصفحات الموجودة بالذاكرة، وهي في هذه الحالة الصفحة رقم 1 لأننا سنحتاج كل من صفر و 2 قبل أن نحتاجها، فنستبدل 1 بـــ 3.

– وهكذا إلى نهاية سلسلة الطلبات نجد أن المجموع الكلّي لعدد مرات حدوث خطأ الصفحة هو 9 أخطاء.

(نورة الحماد)

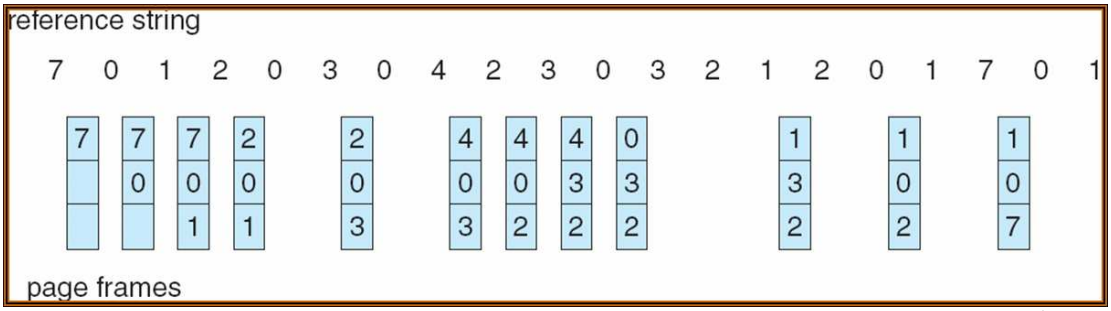
ولأننا قلنا أن الخوارزمية المثالية تضمن حدوث خطأ الصفحة بأقل نسبة ممكنة، فإنه لا يمكن أن نطبق أي خوارزمية على سلسلة الطلبات هذه وباستخدام ثلاث إطارات، إلا بحدوث 9 أو أكثر من الأخطاء.

4- خوارزمية الأقل استخداماً مؤخراً

هي خوارزمية تحاكي الخوارزمية المثالية، فبينما خوارزمية الداخل أولاً خارج أولاً تأخذ بعين الاعتبار وقت إحضار الصفحة للذاكرة، والخوارزمية المثالية تأخذ بعين الاعتبار وقت الاستخدام القادم لكل من الصفحات الموجودة بالذاكرة، فإن خوارزمية الأقل استخداماً مؤخراً تستخدم الماضي القريب بدلاً عن المستقبل، فتستبدل الصفحة التي لم يتم استخدامها لأطول مدة، أي الأقل استخداماً.

لتطبيق هذه النظرية تربط الخوارزمية كل صفحة بوقت آخر استخدام لها، وبذلك تعتبر هذه الخوارزمية تمثّل الخوارزمية المثاليّة لكن بالنظر للماضي بدلاً من المستقبل، مما يجعلها معقولة التطبيق.

مثال:



- أولاً: سيتسبب طلب كل من الصفحات 7 و صفر و 1 بخطأ صفحة، لكن سيتوفر لهذه الصفحات إطارات فارغة، فلن نستخدم تقنية استبدال الصفحات.

- ثانياً: عند طلب الصفحة 2، ولعدم وجود إطار فارغ لوضعها به سنضطر لاختيار إحدى صفحات الذاكرة لتكون الضحية، باستخدام طريقة الأقل استخداماً مؤخراً نجد أن الصفحة رقم 7 هي أقدم صفحة تم طلبها في سلسلة طلب الصفحات، فيتم استبدالها بـــ2.

- ثالثاً: عند طلب الصفحة صفر، لن يتسبب ذلك في خطأ صفحة لوجودها في الذاكرة.

- رابعاً: عند طلب الصفحة 3، وبعد تطبيق خوارزمية الأقل استخداماً مؤخراً نجد أن الصفحة 3 يجب أن تأخذ مكان الصفحة 1، لأنها كل من صفحة صفر وصفحة 2 تم استخدامهما قبل استخدام الصفحة 1، فهي الأقل استخداماً.

- وهكذا لنجد أن عدد أخطاء الصفحة الناتج عن تطبيق هذه الخوارزمية هو اثنا عشر خطأً.

(نورة الحماد)

هذه الخوارزمية هي الأكثر استخداماً بين بقية الخوارزميات، نظراً لأدائها الجيّد، وإمكانية تطبيقها، ولتطبيق هذه النظرية يمكن استخدام طريقتين لمعرفة أي الصفحات الموجودة بالذاكرة هي الأحدث استخداماً وأيها الأقدم استخداماً، هذين الطريقتين هما:

- **طريقة العداد**: في هذه الطريقة نسند لكل صفحة عداداً يحمل عدد المرات التي طُلبت فيه هذه الصفحة، بحيث أنه كلما استخدمنا هذه الصفحة كلما زادت قيمة العداد، فعندما يتم طلب هذه الصفحة، تُنسخ قيمة الساعة الحالية إلى العدّاد، فيعكس العداد وقت آخر استخدام، وعندما يحتاج النظام إلى اختيار صفحة لإخراجها من الذاكرة فإنه يختار الصفحة ذات العداد الأقل قيمة، فالعداد الأقل قيمة يعني أنها الأقدم استخداماً، تتميز هذه الطريقة بسهولة تطبيقها.

- طريقة الـــstack: في هذه الطريقة، يتم حفظ أرقام الصفحات في شكل متسلسلة بحيث أن الصفحة الأقدم تكون في الأسفل، أما الصفحة الأحدث استخداماً فتكون في الأعلى، بحيث أنه

عندما نستخدم صفحة ما، فإننا نخرجها من هذه المتسلسلة ثم نضعها في الأعلى، وهكذا تكون الصفحة الأخيرة هي الأقدم استخداماً، للقيام بهذه العملية يحتاج النظام إلى تغيير ستّ مؤشرات للحفاظ على ترتيب المتسلسلة، لكنها تتميز بأن النظام لا يحتاج للقيام بعملية بحث عن الصفحة الأقدم استخدام لأنها حتماً ستكون في الأسفل، فهذه الطريقة سريعة لكن يصعب تطبيقها نسبيّاً.
(حليمة حكمي – منى حكمي – صفا البلاع)

5- الخوارزمية التقريبية لخوارزمية الأقل استخداماً مؤخراً

تضيف بعض أنظمة الحاسب لكل مدخل في جدول الصفحات (أي لكل صفحة) خانة إضافية تدعى بخانة الطلب (reference bit)، بحيث أنه عندما تُطلب هذه الصفحة فإن هذه الخانة تحمل القيمة 1 ، هذه الخانة تساعد النظام لاحقاً في التعرف على الصفحات التي تم طلبها، والصفحات التي لم تُستخدم مطلقاً، رغم أن النظام لا يستطيع أن يعرف ترتيب طلب الصفحات التي تحمل القيمة 1 ، إلا أن هذه الخانة يمكن أن تساعد كثيراً في عملية استبدال الصفحات، فالصفحة ذات القيمة 1 هي بالتأكيد صفحة مستخدمة ومرغوبة من قبل النظام، لذا تميل أغلب الخوارزميات إلى عدم التردد في استبدال الصفحة التي تحمل القيمة صفراً في خانة الطلب، سنستعرض الآن عدّة خوارزميات مبنية على إضافة خانة الطلب في الجدول الذي يمثّل الصفحات الموجودة بالذاكرة.

أ- خوارزمية الفرصة الثانية

هذه الخوارزمية تبدأ بتطبيق خوارزمية الداخل أولاً خارج أولاً، بالإضافة إلى خانة الطلب، في البداية تكون خانة الطلب لكل الصفحات صفراً، ويتم تعديلها إلى 1 عندما يتم استخدام هذه الصفحة، وعندما يختار النظام صفحة لاستبدالها (بنظام الداخل أولاً خارج أولاً)، فإنه يفحص خانة الطلب، إن كانت مساوية لصفر تستبدل هذه الصفحة مباشرة، أما إن كانت تحمل القيمة 1 ، فإن النظام يعطيها فرصةً ثانية، أي لا يقوم بإخراجها من الذاكرة مباشرة، وعندما تنال إحدى الصفحات فرصةً ثانية فإن قيمة خانة الطلب التابعة لها تقلب إلى صفر، لنتأكد أنها لن تظل تنال فرصاً عديدة، حينها سينتقل النظام إلى الصفحة التي تليها في ترتيب الداخل أولاً خارج أولاً، وتُعامل هذه الصفحة بالمثل.

لتطبيق هذه الخوارزمية يتم ترتيب الصفحات على شكل حلقة، بحيث يبحث النظام في هذه الحلقة على الصفحة التي تحمل القيمة صفر في خانة الطلب، مغيراً في طريقة جميع الصفحات ذات القيمة 1 إلى

صفحات تحمل القيمة صفر، وهكذا فإن الصفحة التي تستخدم بشكل مستمر ستحافظ على القيمة 1 في خانة الطلب، أما الصفحات التي لم تستخدم منذ آخر تصفير (آخر فرصة أعطيت لها)، فإنها عند اختيارها مرة أخرى ستخرج من الذاكرة دون أن تعطى فرصة أخرى.

(صفا البلاع – أفنان السبيهين)

6- استبدال الصفحات المبني على العد

تبقي هذه الطريقة عداداً لكل صفحة، يحمل هذا العداد المرات التي تم طلب هذه الصفحة، بحيث أنه كلما طُلبت صفحةٌ ما، فإن عدادها يزيد بواحد.

هناك خوارزميتين تستخدم هذه الطريقة لتختار الصفحة التي سيتم إخراجها من الذاكرة، اختلفت هاتين الطريقتين في دلالة هذا العداد، هل يجب إخراج هذه الصفحة لو كان عدادها أقل، أم لو كان عدادها أكبر؟

أ- خوارزمية الأقل استخداماً :

تستبدل هذه الخوارزمية الصفحة الأقل استخداماً، أي أن عدّادها يحمل الأقل قيمة، تستند هذه الخوارزمية على أن الصفحة ذات الاستخدام الأكثر، لها الحق في البقاء مدةً أطول في الذاكرة، لحاجة النظام لها، وتظهر لنا مشكلة الصفحة التي تستخدم بشكل مكثّف في بداية تنفيذ البرنامج، ثم لا تستخدم أبداً بعد ذلك، هذه الصفحة ستحمل الأعلى قيمة، لكنها في الحقيقة لا تستحق البقاء في الذاكرة.

ب- خوارزمية الأكثر استخداماً:

هذه الخوارزمية تعاكس الخوارزمية السابقة لحلّ المشكلة المطروحة، أي أنها تستبدل الصفحة الأكثر استخداماً.

يندر استخدام هاتين الطريقتين، لأن نتائجهما لا تمثّل نتائج الخوارزمية المثالية، كما أنهما تستهلكان جزءاً من المصادر لا يُستهان به.

(صفا البلاع – أفنان السبيهين)