Betriebssysteme

Speicherverwaltung, Prozessverwaltung

G. Richter

9. April 2017

Zuteilung des Speichers - Demand-Paging

- Wenn Prozesse in den Hauptspeicher kommen, benötigen sie Speicherplatz

Zuteilung des Speichers - Demand-Paging

- Wenn Prozesse in den Hauptspeicher kommen, benötigen sie Speicherplatz
- Der Speicher kann dem Prozess sofort oder während der Laufzeit zugeteilt werden

Zuteilung des Speichers - Demand-Paging

- Wenn Prozesse in den Hauptspeicher kommen, benötigen sie Speicherplatz
- Der Speicher kann dem Prozess sofort oder während der Laufzeit zugeteilt werden
- Durch die virtuelle Adressierung können Seiten einzeln den Prozessen zur Verfügung gestellt werden (und auch weggenommen werden - Seitenverdrängung)

Zuteilung des Speichers - Demand-Paging

- Wenn Prozesse in den Hauptspeicher kommen, benötigen sie Speicherplatz
- Der Speicher kann dem Prozess sofort oder während der Laufzeit zugeteilt werden
- Durch die virtuelle Adressierung können Seiten einzeln den Prozessen zur Verfügung gestellt werden (und auch weggenommen werden - Seitenverdrängung)
- Die Seiten werden erst zur Verfügung gestellt, wenn sie auch angesprochen werden

Zuteilung des Speichers - Demand-Paging

- Wenn Prozesse in den Hauptspeicher kommen, benötigen sie Speicherplatz
- Der Speicher kann dem Prozess sofort oder während der Laufzeit zugeteilt werden
- Durch die virtuelle Adressierung können Seiten einzeln den Prozessen zur Verfügung gestellt werden (und auch weggenommen werden - Seitenverdrängung)
- Die Seiten werden erst zur Verfügung gestellt, wenn sie auch angesprochen werden
- Dabei wird ausgenutzt, dass bei einem Zugriff auf eine nicht geladene Seite eine Page-Fault-Exception ausgelöst wird

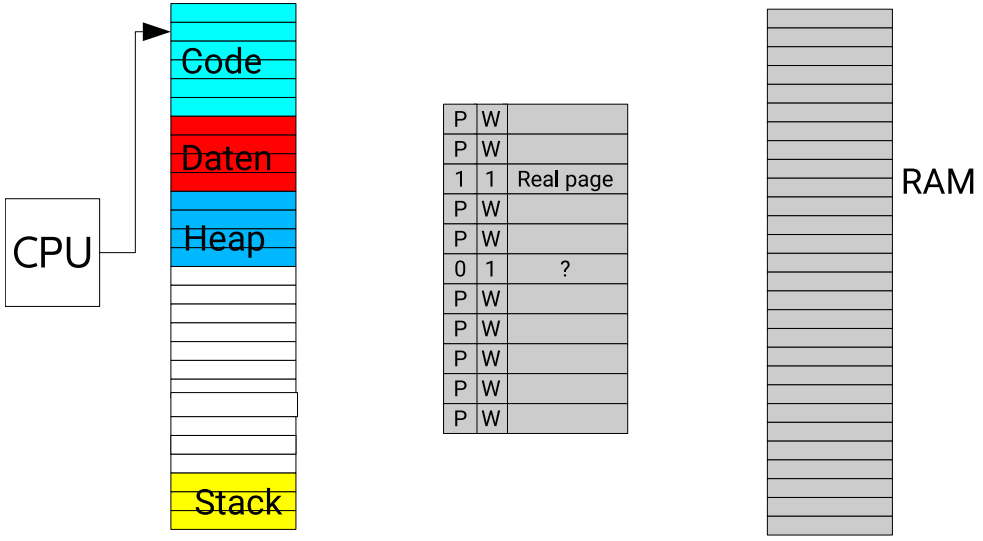
Zuteilung des Speichers - Demand-Paging

- Wenn Prozesse in den Hauptspeicher kommen, benötigen sie Speicherplatz
- Der Speicher kann dem Prozess sofort oder während der Laufzeit zugeteilt werden
- Durch die virtuelle Adressierung können Seiten einzeln den Prozessen zur Verfügung gestellt werden (und auch weggenommen werden - Seitenverdrängung)
- Die Seiten werden erst zur Verfügung gestellt, wenn sie auch angesprochen werden
- Dabei wird ausgenutzt, dass bei einem Zugriff auf eine nicht geladene Seite eine Page-Fault-Exception ausgelöst wird
- Nicht benötigte Seiten werden nicht geladen - Speichereinsparung

Zuteilung des Speichers - Demand-Paging

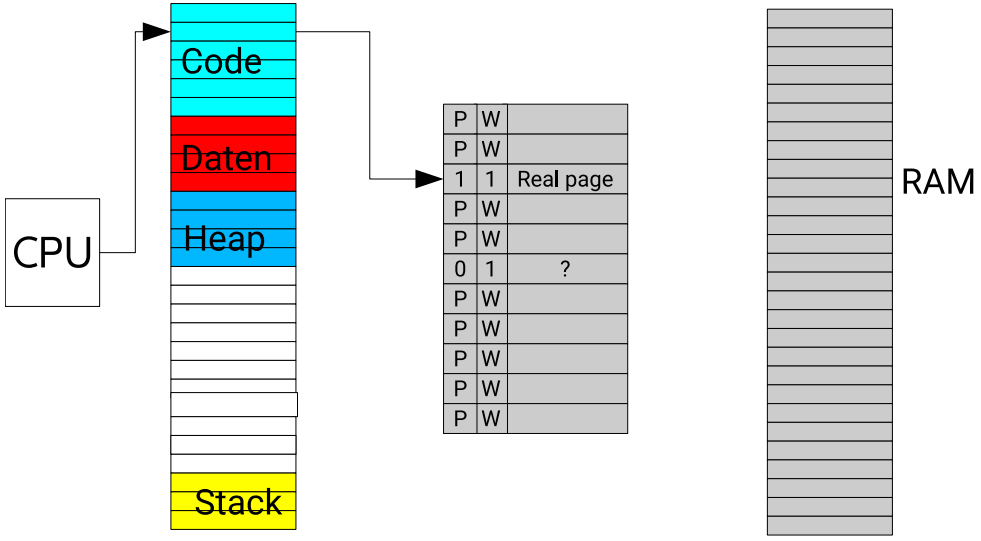
- Wenn Prozesse in den Hauptspeicher kommen, benötigen sie Speicherplatz
- Der Speicher kann dem Prozess sofort oder während der Laufzeit zugeteilt werden
- Durch die virtuelle Adressierung können Seiten einzeln den Prozessen zur Verfügung gestellt werden (und auch weggenommen werden - Seitenverdrängung)
- Die Seiten werden erst zur Verfügung gestellt, wenn sie auch angesprochen werden
- Dabei wird ausgenutzt, dass bei einem Zugriff auf eine nicht geladene Seite eine Page-Fault-Exception ausgelöst wird
- Nicht benötigte Seiten werden nicht geladen - Speichereinsparung
- Anwendung startet schneller, da wenige, zuerst benötigte, Seiten als erste geladen werden

Demand Paging



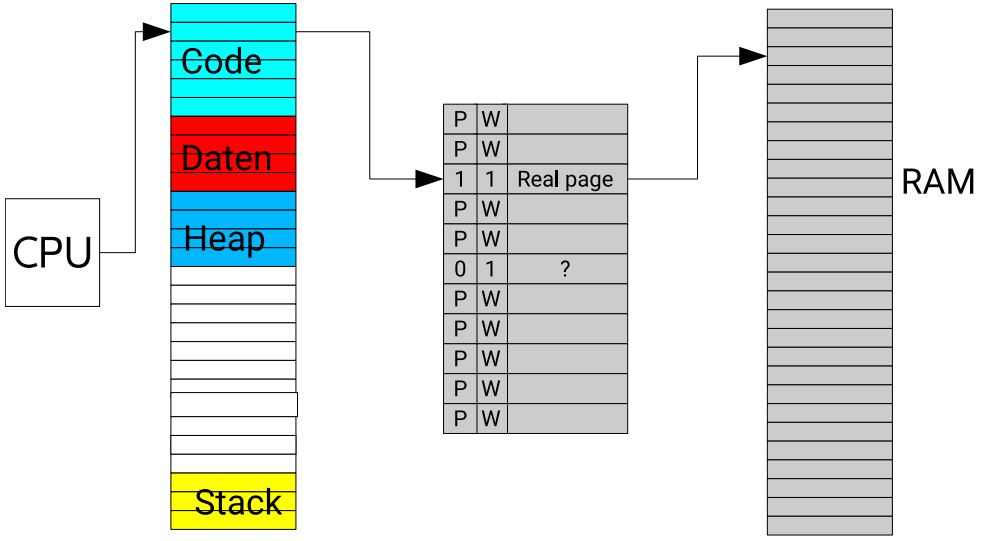
CPU greift auf Code im virtuellen Adressraum zu

Demand Paging



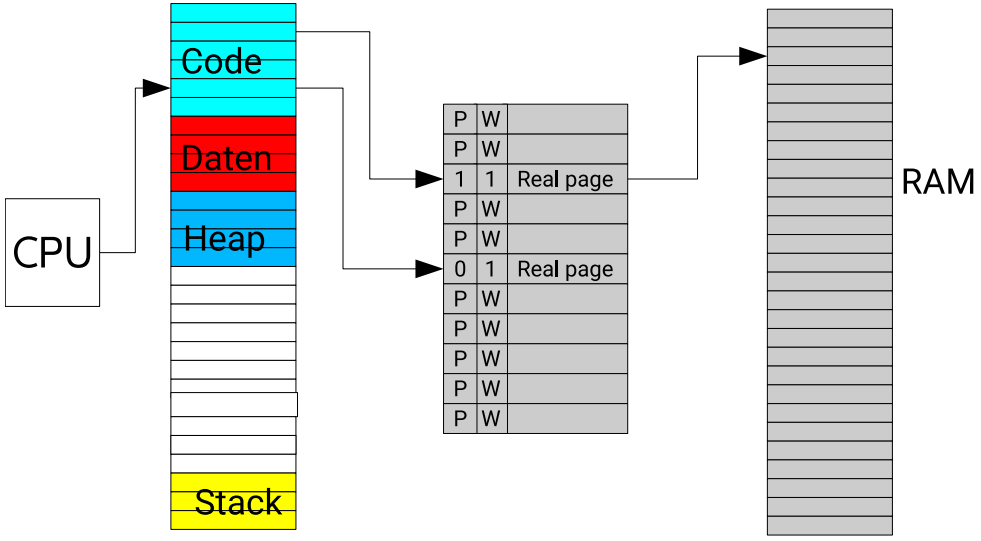
MMU übersetzt virtuelle Adresse

Demand Paging



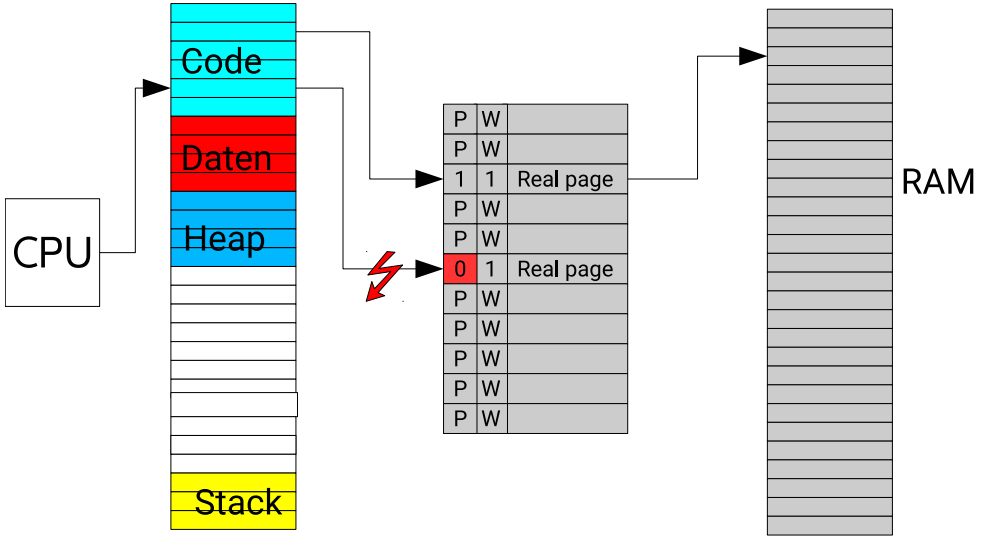
Mit der gefundenen reellen Adresse wird im RAM zugegriffen

Demand Paging



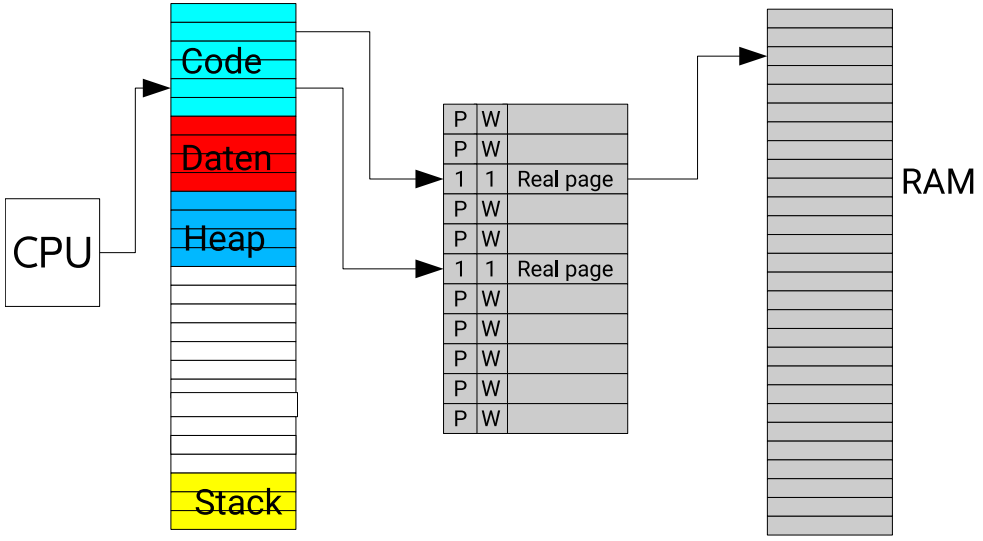
CPU springt auf Adresse in einer anderen Seite

Demand Paging



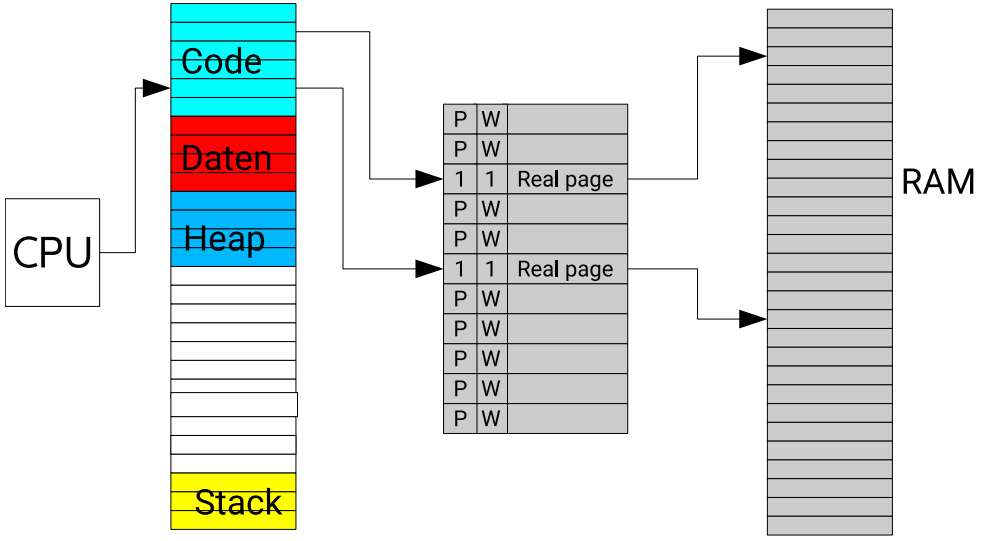
Seite nicht vorhanden (P=0) -> Page-Fault Exception

Demand Paging



Seite wird vom BS geladen und P=1 gesetzt

Demand Paging



Seite kann gelesen werden, Befehl wird erneut ausgeführt

Copy on Write

- Bei „fork()“ entsteht ein neuer Prozess als Kopie des aufrufenden Prozesses

Copy on Write

- Bei „fork()“ entsteht ein neuer Prozess als Kopie des aufrufenden Prozesses
- Da die Inhalte beider Adressräume direkt nach dem Aufruf gleich sind, wird keine wirklich Kopie angelegt, sondern nur eine neue Pagetable mit denselben Inhalten

Copy on Write

- Bei „fork()“ entsteht ein neuer Prozess als Kopie des aufrufenden Prozesses
- Da die Inhalte beider Adressräume direkt nach dem Aufruf gleich sind, wird keine wirklich Kopie angelegt, sondern nur eine neue Pagetable mit denselben Inhalten
- Alle Seiten beider PT werden mit Schreibschutz versehen

Copy on Write

- Bei „fork()“ entsteht ein neuer Prozess als Kopie des aufrufenden Prozesses
- Da die Inhalte beider Adressräume direkt nach dem Aufruf gleich sind, wird keine wirklich Kopie angelegt, sondern nur eine neue Pagetable mit denselben Inhalten
- Alle Seiten beider PT werden mit Schreibschutz versehen
- Beim schreibenden Zugriff auf eine Seite wird diese auf eine freie Seite kopiert, der PT-Eintrag aktualisiert und beide Schreibschutzeinträge aufgehoben

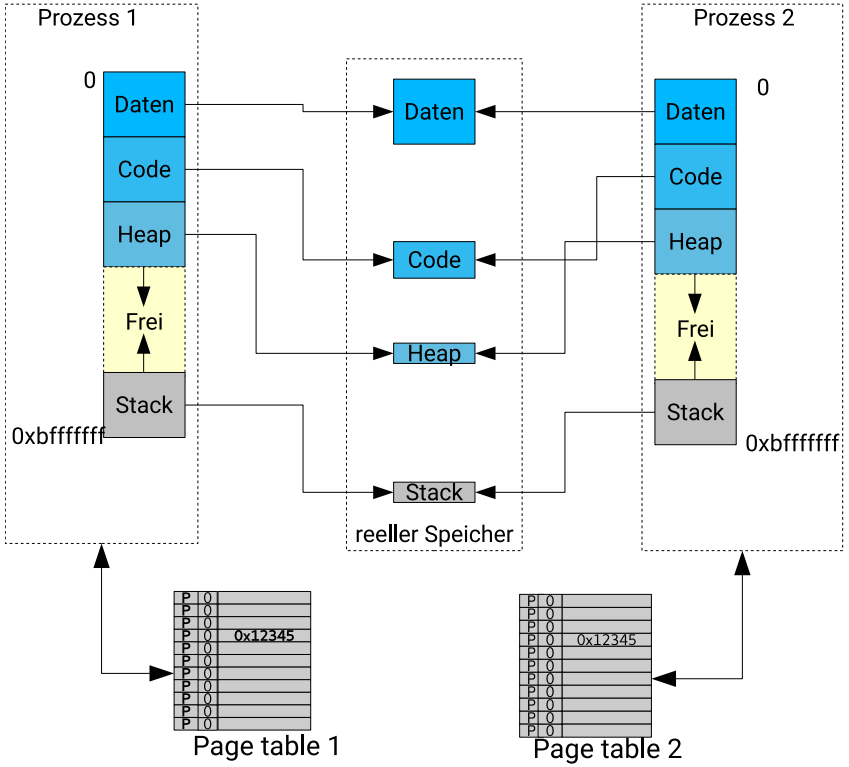
Copy on Write

- Bei „fork()“ entsteht ein neuer Prozess als Kopie des aufrufenden Prozesses
- Da die Inhalte beider Adressräume direkt nach dem Aufruf gleich sind, wird keine wirklich Kopie angelegt, sondern nur eine neue Pagetable mit denselben Inhalten
- Alle Seiten beider PT werden mit Schreibschutz versehen
- Beim schreibenden Zugriff auf eine Seite wird diese auf eine freie Seite kopiert, der PT-Eintrag aktualisiert und beide Schreibschutzeinträge aufgehoben
- So werden alle beschriebenen Seiten mit der Zeit kopiert

Copy on Write

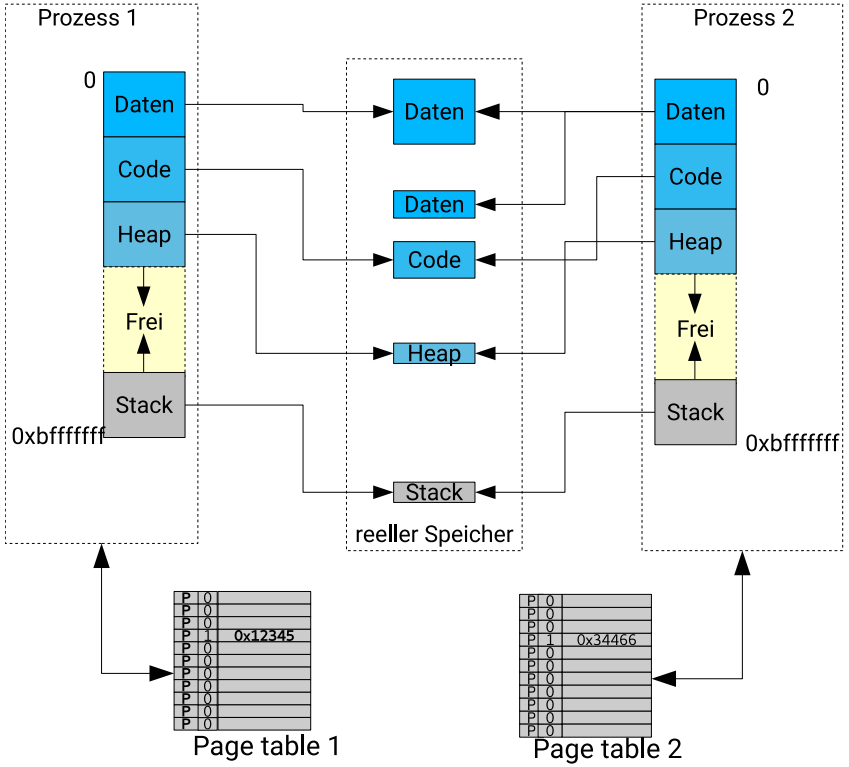
- Bei „fork()“ entsteht ein neuer Prozess als Kopie des aufrufenden Prozesses
- Da die Inhalte beider Adressräume direkt nach dem Aufruf gleich sind, wird keine wirklich Kopie angelegt, sondern nur eine neue Pagetable mit denselben Inhalten
- Alle Seiten beider PT werden mit Schreibschutz versehen
- Beim schreibenden Zugriff auf eine Seite wird diese auf eine freie Seite kopiert, der PT-Eintrag aktualisiert und beide Schreibschutzeinträge aufgehoben
- So werden alle beschriebenen Seiten mit der Zeit kopiert
- Da der Code nie beschrieben wird, wird zumindest dieser Speicherplatz eingespart

Copy on Write



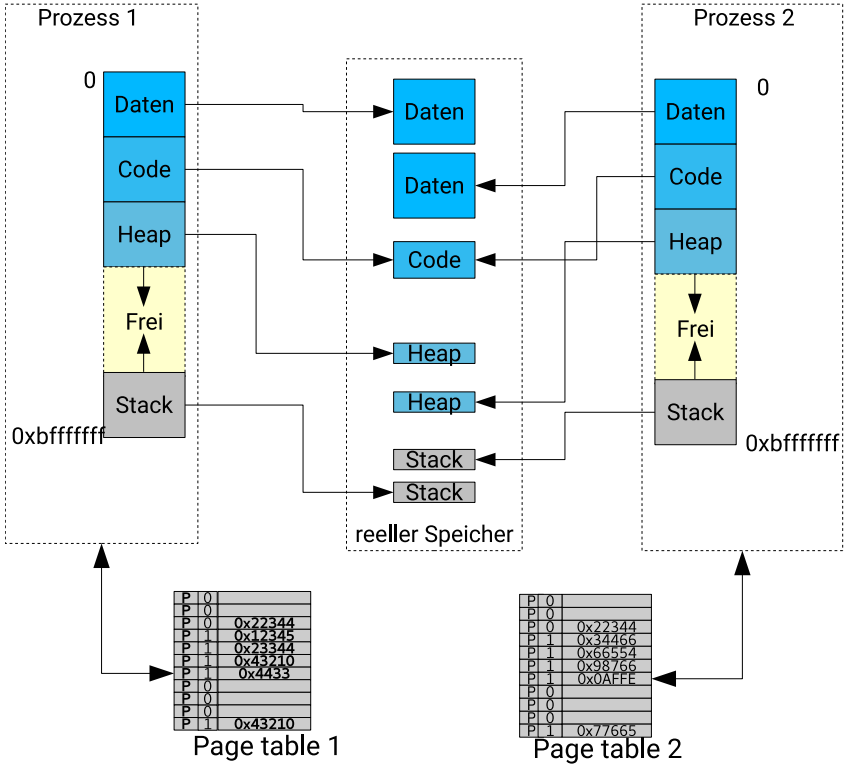
- Anfänglich zeigen alle PT-Einträge auf dieselben Seiten
- alle sind schreibgeschützt

Copy on Write



- Auf eine Datenseite wird geschrieben - Schreibschutzverstoß
- Seite wird kopiert , Schreibschutz der Seite wird aufgehoben

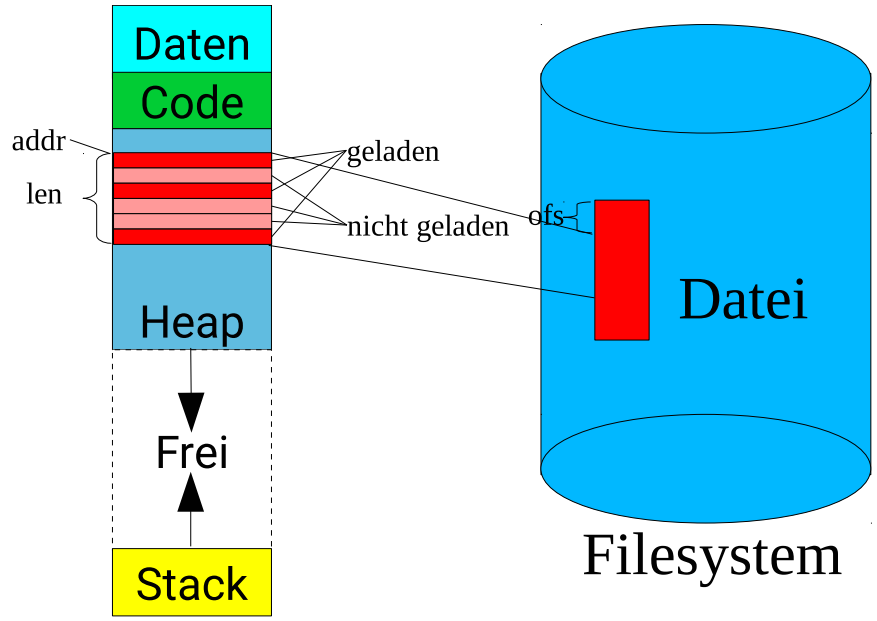
Copy on Write



- Alle Seiten sind kopiert außer den Codeseiten

Memory-mapped files

Zugriff auf eine Datei:
Öffnen der Datei: fd = open(Fname,RW);
Lesen ret = read(fd,buf1,len);
Positionieren: ret = lseek(fd, offset, MODE);
Wieder Lesen: ret = read(fd,buf2,len);
Alternative Methode
mmap(addr,length, , ,fd,ofs);



Seitenverdrängung

- Während des Betriebs werden immer mehr Seiten von Prozessen im Hauptspeicher belegt
- Wird der freie Speicher knapp, müssen - wenn möglich - Seiten freigemacht, also den Prozessen entzogen werden
- Ist der besitzende Prozess inaktiv (bereit/wartend) kann in der PT das P-Bit gelöscht werden und die Seite freigegeben werden
- Erfolgt dennoch ein Zugriff des wieder aktiven Prozesses, kann die Seite wieder einkopiert werden und wie beim Demand-Paging verfahren werden. Das nennt man Seitenfehler (Page-Fault).
- Die optimale Strategie ist es, die Seiten auszulagern, die nie wieder oder am spätesten wieder benötigt werden um den o.g. Fall zu vermeiden. Diese Strategie ist natürlich unmöglich, liefert aber gute Vergleichszahlen für die Bewertung aller anderen Strategien

Seitenverdrängung

Die Wiedereinlagerung von ausgelagerten Seiten hat Auswirkungen auf die Systemleistung:
Der Zugriff auf einen Befehl im Hauptspeicher sei 40 ns ($4 \bullet 10^{-8}$s) lang.
Dauer für die Wiedereinlagerung der Seite von einer Festplatte: 20 ms ($20 \bullet 10^{-3}$s).
Wenn nun ein Seitenfehler nur alle 100000 (10^5) Befehle auftritt, dann berechnet sich die Gesamtdauer für diese Befehlssequenz zu:

$$T = 10^5 \bullet 40 \bullet 10^{-9} + 20 \bullet 10^{-3} = 4 \bullet 10^{-3} + 20 \bullet 10^{-3} = 24 \bullet 10^{-3} s$$

Die Laufzeit wird also fast nur von der Dauer des Festplattenzugriffs bestimmt. Ohne Seitenfehler wären es

$$T = 10^5 \bullet 40 \bullet 10^{-9} = 0,4 \bullet 10^{-3} s.$$

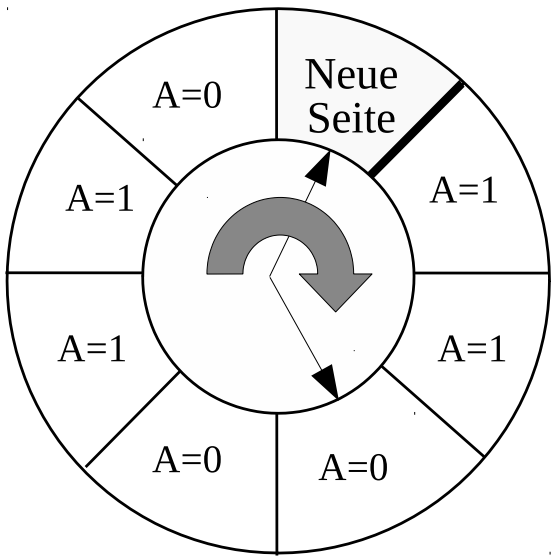
Es ist also von großem Vorteil, möglichst wenige Seitenfehler zu haben.

Seitenverdrängung FIFO

- Einfach und unaufwändig
- Die Seiten werden in der Reihenfolge, in der sie eingelagert wurden, in einer verlinkten Liste verwaltet
- Wird eine Seite benötigt, wird die erste aus der Liste ausgewählt
- Der Nachteil ist, dass auch Seiten verdrängt werden, die auch zu diesem Zeitpunkt noch häufig genutzt werden, da die Nutzung nicht in das Verdrängungskriterium eingeht

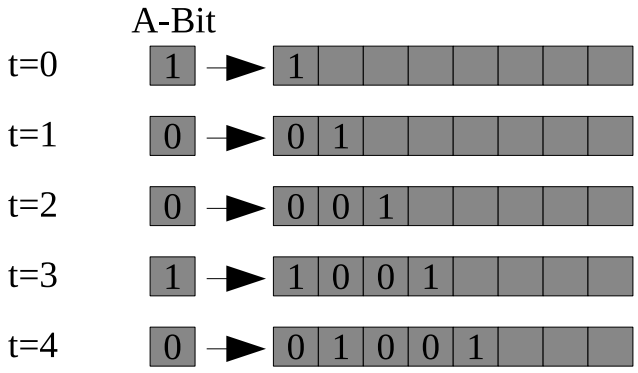
Seitenverdrängung Clock-Algorithmus

- Die Seiten werden in der Reihenfolge der Einlagerung in einem Ringpuffer verwaltet. Ein Daemon löscht im Durchlauf durch den Ring das Accessed-Bit. Ein zweiter Daemon läuft dem ersten in einem kleinen zeitlichen Abstand hinterher. Findet er A=0, wird die Seite zur Auslagerung freigegeben.
- Falls zwischen der Löschung des ersten und der Überprüfung durch den zweiten auf die Seite zugegriffen wird, wird bekanntlich wieder von der MMU das Accessed-Bit gesetzt und damit kann die Seite vom zweiten Daemon nicht zur Auslagerung freigegeben werden.



Seitenverdrängung Least Recently Used

- Die Seiten erhalten einen inversen Alterungs-/Häufigkeitszähler.
- Beim Überprüfen der Seiten wird die Maßzahl um eine Stelle nach rechts geschoben und das Accessed-Bit in die höchste Stelle der Maßzahl transferiert. Anschließend wird es wieder in der PT gelöscht.
- Die Maßzahl ist also umso höher, je öfter in den letzten 8 Überprüfungszyklen auf die Seite zugegriffen wurde.
- Die Seiten mit den kleinsten Werten werden freigegeben.



Thrashing

- Wächst durch viele laufende Prozesse der Speicherbedarf immer mehr, wird die mittlere Verweildauer einer Seite immer kleiner.
- Bei gleicher Nutzung bedeutet das, dass sehr oft Seiten ausgelagert werden müssen, die noch relativ häufig benutzt werden. Dadurch treten immer mehr Seitenfehler auf und Seiten müssen ausgelagert werden, weil andere Seiten wieder eingelagert werden müssen.
- Dies bedeutet, dass die Anzahl der Befehle in Prozessen zwischen zwei Seitenfehlern immer kleiner wird und die Systemgeschwindigkeit immer mehr abnimmt bei gleichzeitig stark steigender Plattenaktivität.
- Da die Plattenzugriffe nicht parallel, sondern seriell ablaufen, kann es sein, dass sogar die CPU-Last sinkt, weil fast nur noch auf Plattentransfers gewartet wird.
- Gegenmaßnahmen: ganze Prozesse auslagern -> keine neuen Prozesse mehr aufnehmen -> Prozesse abbrechen.