TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute for System Architecture, Operating Systems Group

Betriebssysteme und Sicherheit, WS 2014/15

5. Aufgabenblatt – Speicherverwaltung

Geplante Bearbeitungszeit: drei Wochen

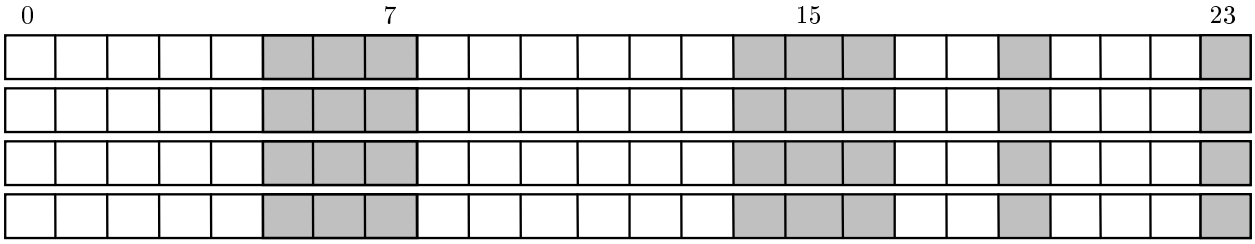
Aufgabe 5.1

Eine Möglichkeit, mehrere Programme gleichzeitig laufen zu lassen, ist die Einteilung des Speichers in feste Partitionen. Erklären Sie die damit verbundenen Probleme und ihre Lösung.

Aufgabe 5.2

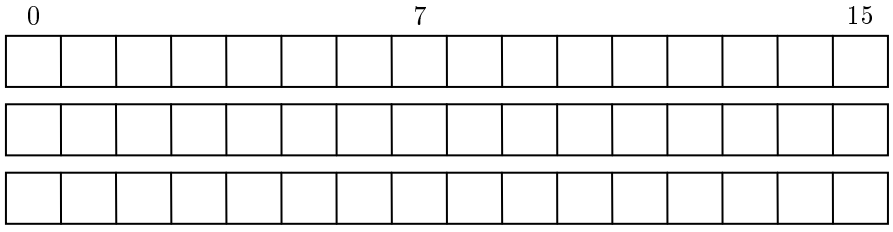
Der Übergang von festen zu variablen Partitionen (Segmenten) erfordert eine entsprechende Verwaltung. Dazu werden im wesentlichen – unter Nutzung von Listen oder Bitmaps – verschiedene Einlagerungsstrategien oder das Buddy-Verfahren verwendet (s TANENBAUM, Kap. 3.2). Erläutern Sie die generellen Vor- und Nachteile dieser Vorgehensweise. Erklären und bewerten Sie die Einlagerungsstrategien an folgendem Beispiel:

- HS-Gesamtkapazität: 24 KByte
- Anfangsbelegung (belegte Blöcke in KByte): 5–7 / 14–16 / 19 / 23
- nacheinander sollen Segmente der Größe 3 – 5 – 1 – 5 (in KByte) eingelagert werden.



Aufgabe 5.3

Diskutieren Sie das Buddy-Verfahren (Vorgehen, Vorteile und Probleme) anhand des folgenden Beispiels. Ein Speicher kann 16 Seiten aufnehmen. Nacheinander sind Segmente A,. . . ,F einzulagern, die aus 5 – 1 – 3 – 4 – 2 – 3 Seiten bestehen (kann ein Segment bei seiner Anforderung nicht eingelagert werden, so wird es für den nächstmöglichen Zeitpunkt vorgemerkt). Danach wird Segment E und schließlich Segment B freigegeben. Geben Sie den Ablauf der Einlagerungen und die Lage der Segmente im Speicher an.



Aufgabe 5.4

Geben Sie einen Überblick über den Begriff „virtueller Speicher“ (Anliegen, Aufgaben, Begriffe, Vorgehensweise, Vor- und Nachteile). Verdeutlichen Sie das Problem von Platzbedarf und Tabellenauslastung bei der Verwendung einfacher Seitentabellen anhand des nachfolgenden Beispiels, und diskutieren Sie Lösungen für dieses Problem.

- zugrundeliegende Architektur: 64 Bit
- Größe des physischen Speichers: 256 MByte
- Seitengröße: 8 KByte
- Größe eines Eintrags in der Seitentabelle: 8 Byte.

Aufgabe 5.5

Betrachtet werde ein System mit folgenden Eigenschaften:

- Seitengröße: 4 KByte
- Größe des virtuellen Speichers: 64 KByte
- Größe des realen Speichers: 32 KByte.

Geben Sie sowohl die einfachen Seitentabellen als auch die invertierte Seitentabelle für folgenden Systemzustand an:

- In Prozess P_1 bestehen folgende Abbildungen von virtuellen auf physische Adressen: `0x13AF` auf `0x53AF`, `0x2745` auf `0x3745`, `0xF30D` auf `0x30D`.
- In Prozess P_2 gelten folgende Abbildungen: `0x1ABD` auf `0x4ABD`, `0x2007` auf `0x6007`, `0x3FFE` auf `0x7FFE`.

Was ändert sich, wenn in Prozess P_2 die virtuelle Adresse `0x1ABD` auf `0x5ABD` statt auf `0x4ABD` abgebildet werden soll?

Aufgabe 5.6

Erläutern und bewerten Sie die verschiedenen Strategien zur Verdrängung (Ersetzung) von Seiten beim virtuellen Speicher anhand der folgenden Beispiele, wobei jeweils ein Speicher mit vier Rahmen (Kacheln) zugrunde gelegt wird.

(a) Für die Seitenreferenzfolge 1 – 2 – 3 – 4 – 1 – 2 – 5 – 1 – 2 – 3 – 4 – 5 sind die optimale Seitenersetzung sowie die Strategien FIFO und LRU zu betrachten. Bestimmen Sie zusätzlich für den Algorithmus FIFO die Anzahl der Seitenfehler in dem Fall, dass nur drei Rahmen vorhanden sind. Welches „unnormale" Verhalten bei FIFO zeigt das Beispiel, wenn die Anzahl der Rahmen vergrößert wird?

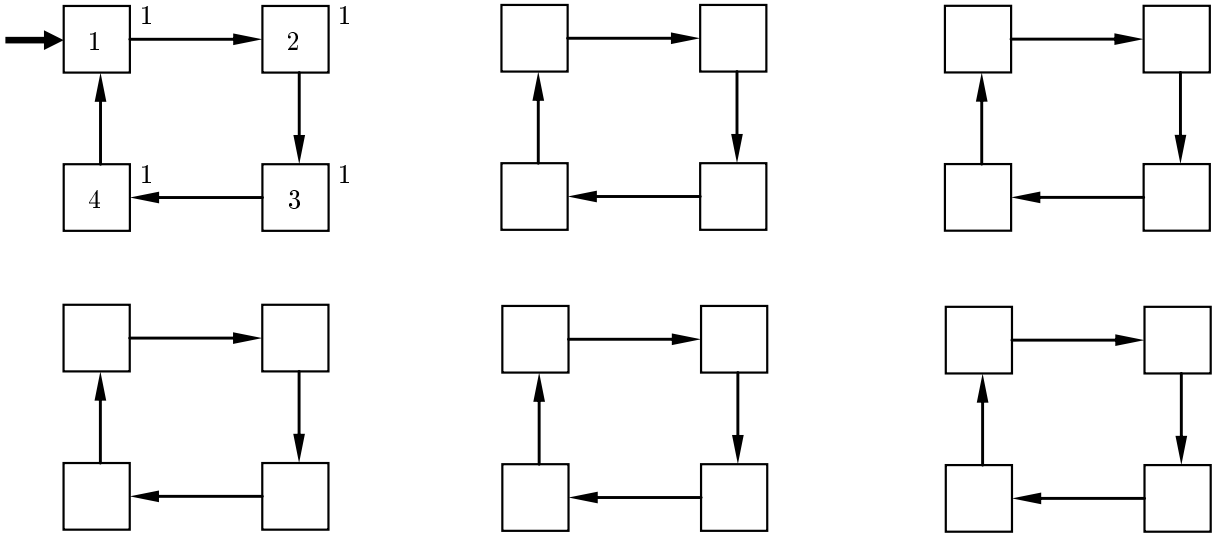
OPT	1	2	3	4	1	2	5	1	2	3	4	5
Rahmen 1												
2												
3												
4												
Seitenfehler												

FIFO	1	2	3	4	1	2	5	1	2	3	4	5
Rahmen 1												
2												
3												
4												
Seitenfehler												

FIFO	1	2	3	4	1	2	5	1	2	3	4	5
Rahmen 1												
2												
3												
Seitenfehler												

LRU	1	2	3	4	1	2	5	1	2	3	4	5
Rahmen 1												
2												
3												
4												
Seitenfehler												

(b) Eine Variante von FIFO, die auch das Referenzverhalten der Prozesse berücksichtigt, ist der Second-Chance- bzw. Clock-Algorithmus. Diskutieren Sie diesen Algorithmus für die Referenzfolge 1 – 2 – 3 – 4 – 2 – 3 – 5 – 2 – 3 – 1 – 2 – 4.



(c) Das Aging-Verfahren ist eine näherungsweise Implementierung von LRU. Um das Alter einer Seite zu bestimmen, werden dabei die Referenzbits zyklisch ausgewertet und anschließend wieder zurückgesetzt. Untersuchen Sie dieses Verfahren für die Referenzfolge

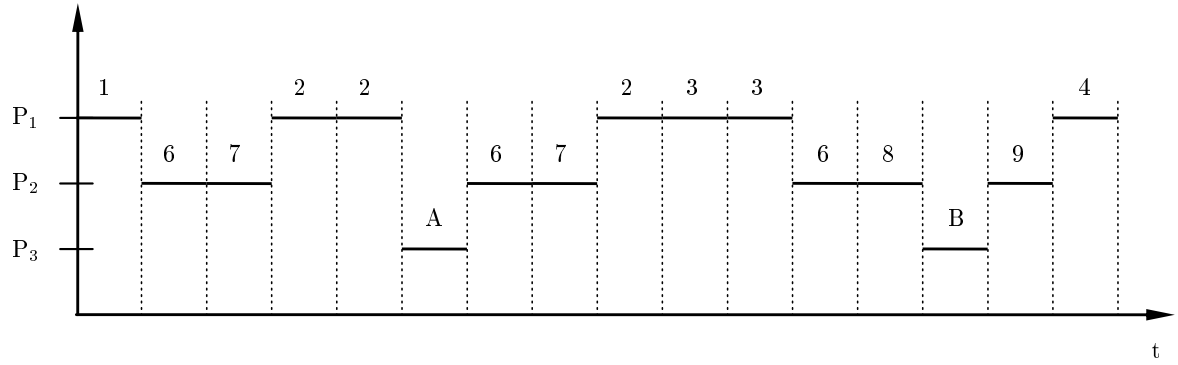
1 2 3 <tick> 4 3 2 3 2 4 <tick> 2 5 3 <tick> 3 1 5 <tick>

wobei <tick> für den Zeitpunkt der Auswertung der Referenzbits steht!

Seite	Initialisierung	Tick 1	Tick 2	Tick 3	Tick 4
1					
2					
3					
4					
5					
Rahmen					
R_1					
R_2					
R_3					
R_4					

Aufgabe 5.7

Aufgrund der Seitenreferenzen dreier Prozesse möge sich die in unten stehender Abbildung dargestellte resultierende Referenzfolge ergeben. Erläutern Sie den Begriff „Arbeitsmenge“ an diesem Beispiel. Tragen Sie in der Tabelle die entstehende Speicherbelegung ein unter der Annahme, dass fünf Rahmen verfügbar sind und dass jeder Prozess einen Arbeitsmengenparameter (Fenstergröße) von 2 Seiten besitzt. Erklären Sie ferner den Thrashing-Effekt.

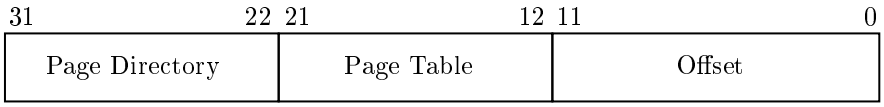


Rahmen	1	6	7	2	2	A	6	7	2	3	3	6	8	B	9	4
1																
2																
3																
4																
5																

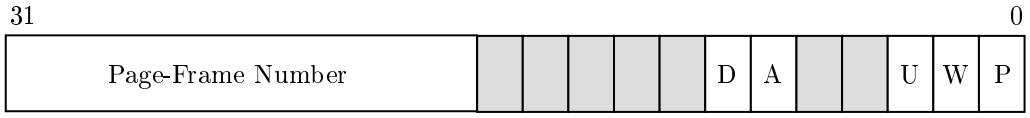
Hinweis:

Die restlichen Aufgaben basieren auf der x86-Prozessor-Familie und gehen daher von folgenden Voraussetzungen aus:

- 32-Bit-Adressen
- Größe des virtuellen Adressraums: 4 GByte
- Seitengröße: 4 KByte
- zweistufige Seitentabellen: die ersten 10 Bit der Adresse dienen als Index für die erste Stufe (dem Seitenverzeichnis, Page Directory), die nächsten 10 Bit als Index für die zweite Stufe (den Seitentabellen, Page Table) und der Rest als Offset innerhalb der Seite:



- ein Eintrag in einer Seitentabelle hat die folgende Struktur:



Bedeutung der Einträge (jeweils bei gesetztem Bit):

P	Present Bit – die zum Eintrag gehörende Seite ist im Speicher
W	Write Bit – die Seite ist auch schreibbar
U	User Bit – die Seite ist für den Nutzer zugreifbar
A	Accessed Bit – auf die Seite wurde zugegriffen
D	Dirty Bit – die Seite wurde verändert

Die anderen Bits werden hier nicht betrachtet.

Aufgabe 5.8

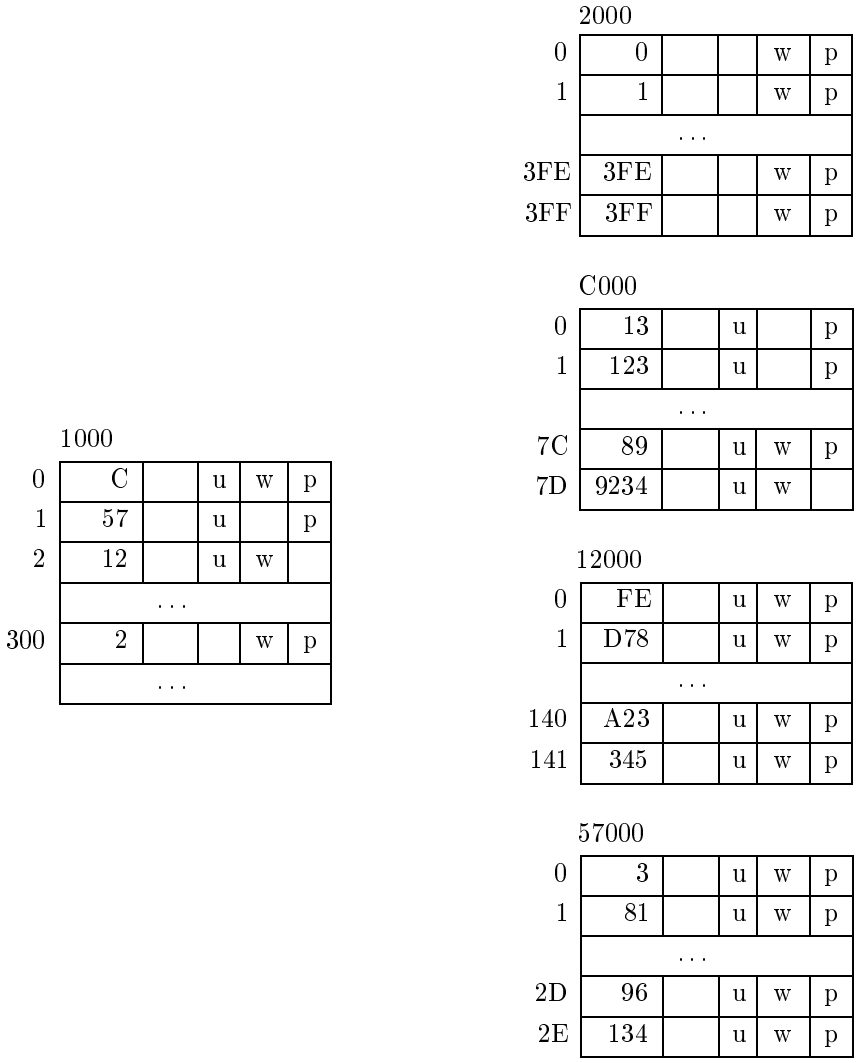
Machen Sie sich mit Hilfe der auf der Webseite zur Verfügung gestellten Web-App noch einmal mit dem Prinzip der Adressumsetzung in mehrstufigen Seitentabellen vertraut. Wählen Sie dazu folgende Adressen (wie üblich sind führende Nullen weggelassen) und Zugriffsmodi aus, entscheiden Sie vorab über die resultierende physische Adresse bzw. den resultierenden Fehler und überprüfen Sie anschließend mit Hilfe der App Ihr Ergebnis.

virtuelle Adresse	Zugriffsart	Modus	PD	PT	physische Adresse	PF	Grund für PF
0x000267	lesend	user					
0x001DED	schreibend	user					
0x40026C	schreibend	user					
0x4012AD	lesend	user					
0x8012B2	lesend	user					
0xC00276	lesend	user					
0xC012B7	schreibend	kernel					

PD Page Directory, 1. Stufe der Seitentabellen-Hierarchie
PT Page Table, 2. Stufe der Seitentabellen-Hierarchie
PF Page Fault (Seitenfehler)

Aufgabe 5.9

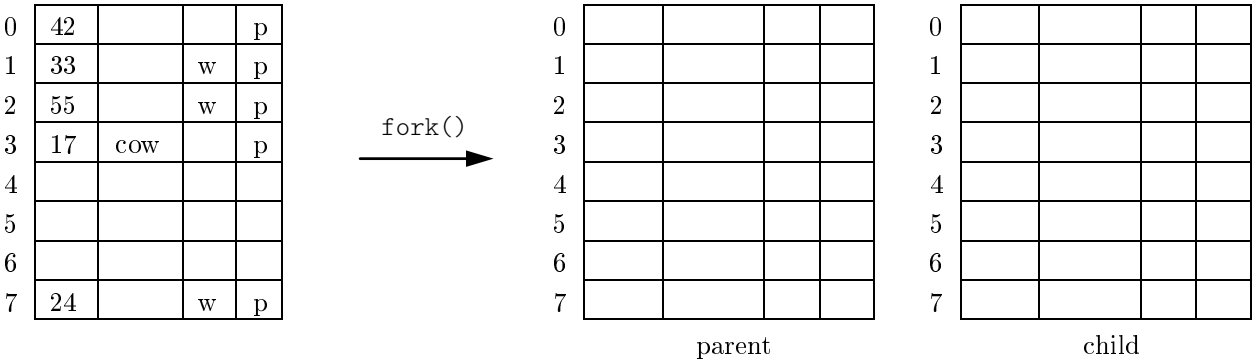
Gegeben sei der durch die abgebildete zweistufige Seitentabelle beschriebene Adressraum, wobei `0x1000` die Eintrittsadresse für das Page Directory ist. Alle angegebenen Zahlen sind Hexadezimal notiert, ein freies Feld in den letzten drei Spalten bedeutet, dass das entsprechende Bit nicht gesetzt ist. Betrachtet werde eine Menge von Speicherzugriffen aus dem normalen „user mode“ heraus (siehe Tabelle, Bedeutung der Abkürzungen siehe Aufgabe 8). Entscheiden Sie für jeden Speicherzugriff, ob der Zugriff gestattet ist und geben Sie in diesem Fall die physische Adresse an. Bestimmen Sie andernfalls den Grund für auftretende Seitenfehler und beschreiben Sie die Reaktion des Betriebssystems.



virtuelle Adresse	Zugriffsart	PD	PT	physische Adresse	PF	Grund für PF
`0x1234`	lesend					
`0x42D8E4`	lesend					
`0x1256`	schreibend					
`0x400985`	schreibend					
`0x7DF87`	schreibend					
`0x940AFE`	lesend					
`0xC02A4B81`	lesend					

Aufgabe 5.10

Zum Erzeugen eines neuen Prozesses wird in Unix der Systemruf `fork()` benutzt. Der Adressraum des dadurch erzeugten Prozesses ist eine identische Kopie des Adressraums des erzeugenden Prozesses. Da in der Regel der neue Prozess sofort ein `execve()` ausführt und dabei den Inhalt seines Adressraums ersetzt, versucht man das tatsächliche Kopieren zu vermeiden. Die zugehörige Technik heißt copy on write. Erläutern Sie, wie copy on write prinzipiell funktioniert, und führen Sie es für den folgenden, durch eine einfache Seitentabelle beschriebenen Adressraum exemplarisch durch (die 2. Spalte der Tabelle werde dabei für das COW-Bit genutzt). Was geschieht, wenn nach dem Kopieren zunächst der Kind-Prozess und danach der Vater-Prozess schreibend auf Seite 7 zugreifen?



Aufgabe 5.11

Im Unix-Teil der Vorlesung wurde die logische Struktur des Adressraums eines Unix-Prozesses vorgestellt. Die Bereiche innerhalb dieses Adressraums werden durch das Betriebssystem mittels geeigneter Datenstrukturen – analog zum Prozesssteuerblock – verwaltet.

(a) Geben Sie einen möglichen Aufbau einer solchen Datenstruktur in Form einer Tabelle an, die die Informationen enthält, die zur Beschreibung der einzelnen Bereiche erforderlich sind.

(b) Bilden Sie die folgende Situation mit Hilfe dieser Datenstruktur ab: Die Größe des Text-, Daten- und BSS-Segments eines Prozesses betrage 20 KByte, 6 KByte bzw. 11 KByte (jeweils aufgerundet). Ferner habe der Prozess während seiner Laufzeit ab Adresse `0x5009000` einen 24 KByte großen Ausschnitt aus einer Datei eingeblendet, beginnend ab dem Datei-Offset `0x7000`. Der Stack (Keller) beansprucht aktuell eine Seite.

(c) In diesem Adressraum sei auf folgende Adressen lesend zugegriffen worden, und es gelte die unten stehende Rahmenzuordnung. Schlagen Sie für die fehlenden Einträge geeignete Rahmen vor und konstruieren Sie die entsprechende einfache Seitentabelle.

virtuelle Adresse	physische Adresse
0x1000	0x340000
0x7000	0x32000
0x8000	
0xA000	
0x5009000	0x13000
0xBFFFF000	0x42000
0xC0000000	0
0xC0001000	0x1000

(d) Was geschieht bei einem Lesezugriff auf die Adressen `0x3400`, `0x500D100`, `0xC0000040` und einem Schreibzugriff auf `0xA780`, `0x3B060`, `0xBFFFEDCB`, wenn der Zugriff von einem normalen Nutzerprogramm aus erfolgt?