TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Betriebssysteme

Betriebssysteme und Sicherheit, WS 2021/22

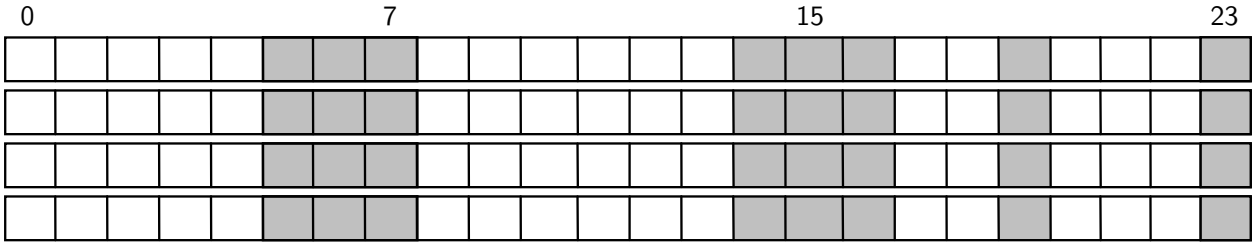
9. Aufgabenblatt – Speicherverwaltung

Geplante Bearbeitungszeit: 13.12.2021 – 17.12.2021

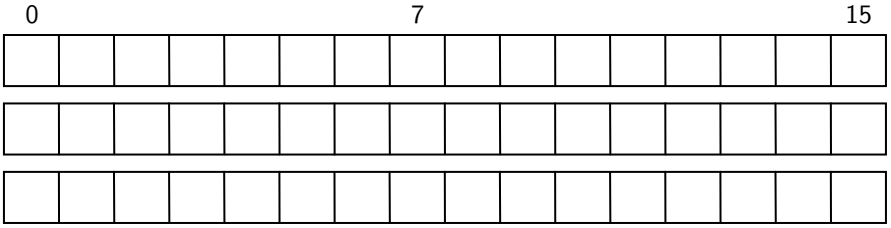
Aufgabe 9.1 Bei der Verwendung von malloc() oder new müssen dynamische Speicherbereiche im Heap eines Programmes verwaltet werden. Veranschaulichen Sie die Platzierungsstrategien *First-Fit*, *Next-Fit*, *Best-Fit* und *Worst-Fit* anhand des folgenden Beispiels:

- Gesamtkapazität des Heap-Segments: 24 KiByte,
- bereits belegte Bereiche (in KiByte) durch zurückliegende Anfragen: 5–7 / 14–16 / 19 / 23,
- nacheinander werden Bereiche der Größen 3 – 5 – 1 – 5 (in KiByte) angefragt und müssen entsprechend platziert werden.

Beurteilen Sie die verwendeten Platzierungsverfahren bezüglich ihrer Komplexität und der Speicherauslastung.



Aufgabe 9.2 Ein weiteres Platzierungsverfahren, das in die Kategorie *Quick-Fit* fällt, ist das Buddy-Verfahren. Diskutieren Sie das Buddy-Verfahren (Vorgehen, Vorteile und Probleme) anhand des folgenden Beispiels. Das Heap-Segment ist 16 KiByte groß. Nacheinander sind die Bereiche A,. . . ,F zu platzieren, die aus 5 – 1 – 3 – 4 – 2 – 3 KiByte bestehen (kann ein Bereich bei seiner Anforderung nicht platziert werden, so wird er für den nächstmöglichen Zeitpunkt vorgemerkt). Danach wird der Bereich E und schließlich der Bereich B freigegeben. Geben Sie den Ablauf der Platzierungen und die Lage der Bereiche im Heap-Segment an.



Aufgabe 9.3 Mit der parallelen Ausführung vieler Programme auf einem System, braucht das Betriebssystem eine Technik den verfügbaren physischen Speicher den einzelnen Programmen zuzuordnen. Eine häufig verwendete Technik dafür ist die *virtuelle Adressierung* mittels *Seitentabellen*. Dafür unterscheidet das Betriebssystem zwischen dem *virtuellen Speicher* eines Prozesses und dem *physischen Speicher* der Hardware. Mit Hilfe von Hardwareunterstützung und den Seitentabellen werden Bereiche des virtuellen Speichers (auch genannt *Seiten*) auf Bereiche des physischen Speichers (*Kacheln*) zur Laufzeit des Programms übersetzt.

Betrachtet werde ein System mit folgenden Eigenschaften:

- Seitengröße: 4 KiByte
- Größe des virtuellen Speichers: 64 KiByte
- Größe des physischen Speichers: 32 KiByte.

Geben Sie die für die Übersetzung nötigen *einfachen* Seitentabellen 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 P_2 die virtuelle Adresse `0x1ABD` auf `0x5ABD` statt auf `0x4ABD` abgebildet werden soll?

Aufgabe 9.4 Welches Problem entsteht, wenn für folgendes System virtuelle Adressierung verwendet wird?

- Seitengröße: 4 KiByte
- Architektur (virtuelle Adressbreite): 36 Bit
- physischer Speicher: 128 MiByte
- Seitentabelleneintrag: 4 Byte

(a) Diskutieren Sie den Ansatz der *mehrstufigen Seitentabellen* zur Lösung des beschriebenen Problems. Erstellen Sie eine 2-stufige Seitentabelle, unter der Verwendung von 12 Bit für den Index einer Stufe, für folgende Abbildungen. Zeigen Sie, dass die entstandene 2-stufige Seitentabelle das Problem einfacher Seitentabellen löst. Gibt es ansonsten noch weitere Vorteile mehrstufiger Seitentabellen gegenüber einfacher?

Hinweis: Führende Nullen werden bei den Adressen weggelassen.

- `0x13AF` auf `0xF3AF`
- `0x3FFA` auf `0x5FFA`
- `0x1001BEA` auf `0x1BEA`
- `0x1234567` auf `0xFF567`
- `0xDEADBEEF` auf `0xC7EEF`

(b) Ein anderer Ansatz zur Lösung des Problems, ist die Verwendung von *invertierten Seitentabellen*. Erstellen Sie (auszugsweise) eine invertierte Seitentabelle für die Abbildungen beschrieben in Aufgabe (a).