Lehrstuhl für Connected Mobility
School of Computation, Information and Technology
Technische Universität München

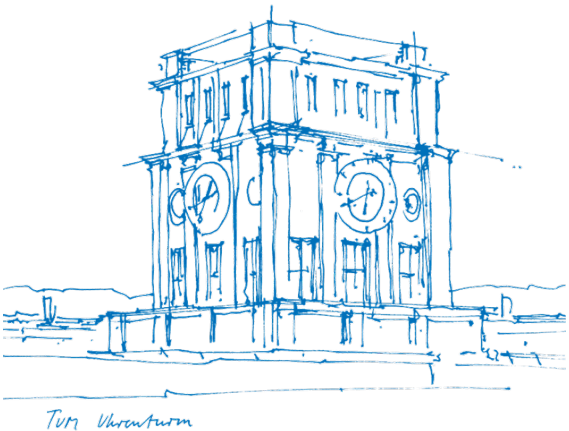
Grundlagen: Betriebssysteme und Systemsoftware

Tutorübung

Mario Delic

Lehrstuhl für Connected Mobility
School of Computation, Information and Technology
Technische Universität München

Übungswoche 11



Dateien

• **Genaue Implementierung von Dateimechanismen kann variieren**,z.B.:
↪ Windows: name.endung; interpretierte Endungen (.exe, .txt)
↪ UNIX: freie Benennung; Endungen per default nicht interpretiert

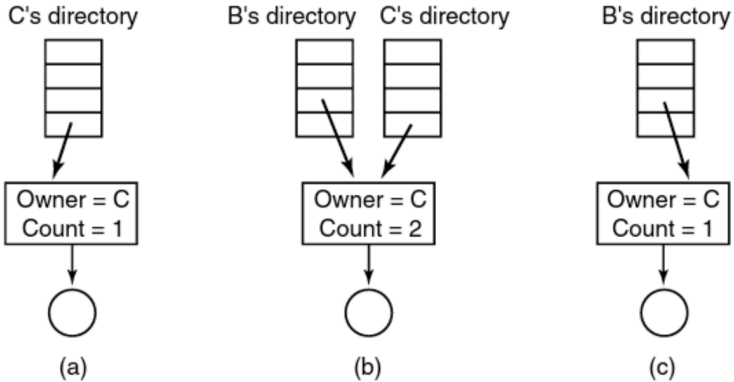
• **Strukturierung**:
- Unstrukturiert: Folge von Bytes (Windows, UNIX)
- Sequenzen von Einträgen: Einträge fester Größe und Struktur (i.d.R. nicht mehr genutzt)
- sortierte Baumstruktur: Einträge variabler Größe; für Verwaltung großer Datenmengen (BS in Großrechnern)

• **Einige unterstütze Typen:**
Directories (d), Files (-: files, hard links)(l: symbolic links), Character Special Files (c), Block Special Files (b)

Dateien

•Hard & symbolic links:

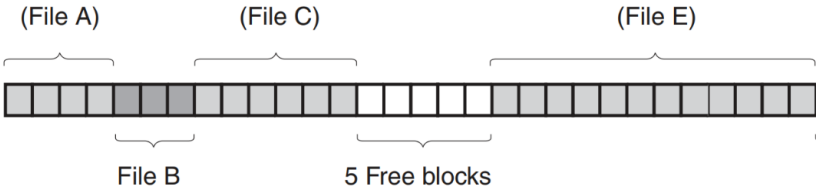
- Hard link: verweist direkt auf die selbe i-node wie link target
- Symbolic link: verweist per Dateipfad auf einen Dateieintrag im FS



Implementierung

• **Contiguous Allocation**:

- Datei wird als zusammenhängende Folge von Blöcken auf der Festplatte verwaltet



- Vorteil: Sehr simpel; Nachteil: Fragmentierung (Löcher im Speicher)
- zu finden bei ROMs (CDs, DVDs, etc.)

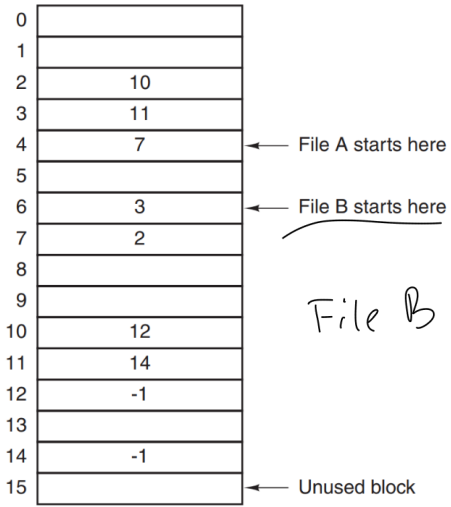
Implementierung

• **Linked List Allocation**:
- Die durch die Datei belegten Blöcke werden in einer verketteten Liste verwaltet
- Das erste Wort innerhalb des Blocks wird als Zeiger auf den nächsten Block verwendet

- Vorteil: Nahezu keine Fragmentierung (lediglich letzer Block); Nachteil: langsam, Blöcklgröße =|= 2er-Potenz (wegen Anfangspointer))
- Lösung: Seperate Tabelle speichert die Allokation der Blöcke

↪ FAT (File Allocation Table)

Implementierung



Quelle: Tanenbaum/Bos

Implementierung

• **index-nodes (i-node)**:
- Jede Datei wird repräsentiert durch eine i-node
- Enthält Attribute und Belegte Blöcke

- Vorteil: Muss nur für geöffnete Dateien geladen werden; Nachteil: theoretisch begrenzte Dateigröße

- Lösung: Indirect Verweise auf weitere Blockadressen bei bedarf

i-nodes

user group other

111 111 111

0777

0611

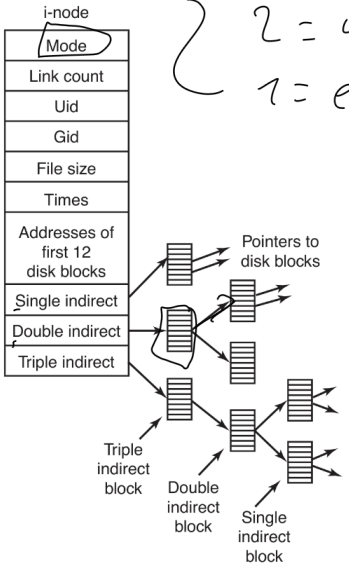
110 001 001

0b 111

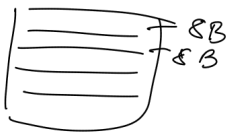
4 = read

2 = write

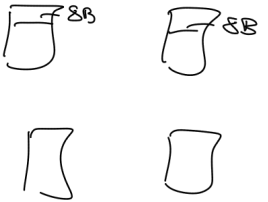
1 = exec



PTO (9-bits)



PTIs (2^9 PTIs)



Paging Kompakt

• **Speichergröße** $\longleftrightarrow$ **Adressbits** & **Seiten-/Kachelgröße** $\longleftrightarrow$ **Offsetbits**:
Bei 2^X (adressierbaren) Bytes $\rightarrow$ X Bits für die Adresse/das Offset.

• **Virtuelle/Physische Adresse**:
virt./phys. Adresse = (Seiten/Kachel Nr. | offset)
$\hookrightarrow$ Bits für virt./phys. Adresse = Bits für Seiten/Kachel Nr. + Bits für offset
$\hookrightarrow$ Offset bei beiden identisch da Seitengröße = Kachelgröße

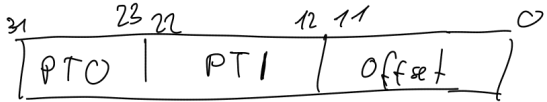
• **Seiten-/Kachelanzahl** $\longleftrightarrow$ **Seiten-/Kachelbits**:
Bei 2^Y Seiten/Kacheln $\rightarrow$ Y Bits für die Seiten/Kachel Nr..

• Speichergröße $\div$ Seiten-/Kachelgröße = Seiten-/Kachelanzahl ($\longleftrightarrow$ Seiten-/Kachelbits)
• Speichergröße $\div$ Seiten-/Kachelanzahl = Seiten-/Kachelgröße ($\longleftrightarrow$ offsetbits)

X-bit Architektur = X-virt. Adr.

Aufgabe 2

a) Ein Computer mit **32-bit breiten virtuellen Adressen** benutzt eine zweistufige Seitentabelle zur Adressübersetzung. (PTO) (PT1)
Eine virtuelle Adresse bestehe aus **9 Bits für die erste Stufe**, **11 Bits für die zweite Stufe** sowie einem Offset. Wie sieht die Adresse aus?



32 − (9 + 11) = 12

b) Wie groß sind die Seiten?

12-bit → 2^{12} Byte = 4 KiB

c) Aus wie vielen Seiten besteht der virtuelle Adressraum?

$2^{20} = 2^{9} \cdot 2^{11} = 2^{20}$ Seiten

Aufgabe 2

d) Wie groß sind die Seitentabellen jeweils, wenn für die Größe eines Eintrags vereinfachend 8 Byte angenommen werden?

PT0 = $2^9 \cdot 2^3 = 2^{12}$ = 4096 Byte

PT1 = $2^{11} \cdot 2^3 = 2^{14}$ = 16384 Byte

Insg. 4 KiB

$2^9 \cdot 2^{14} = 2^{23}$

$\Rightarrow$ 8 MiB + 4 KiB

e) Nehmen wir an, wir verwenden statt der zweistufigen Übersetzung eine einstufige, bei der die erste Stufe 20-bit breite Seitennummern verwendet. Wie viel Speicher kann adressiert werden? Kann mittels der zweistufigen Übersetzung mehr Speicher adressiert werden?

Aufgabe 2

d) Wie groß sind die Seitentabellen jeweils, wenn für die Größe eines Eintrags vereinfachend 8 (2^3) Byte angenommen werden?

Tabelle der Ersten Stufe: 2^9 Einträge $* 2^3 \frac{\text{Bytes}}{\text{Eintrag}} = 2^{12}$ Bytes = 4 KiB

Tabelle der Zweiten Stufe: 2^{11} Einträge $* 2^3 \frac{\text{Bytes}}{\text{Eintrag}} = 2^{14}$ Bytes = 16 KiB (pro Tabelle!)

Summer der Tabellen der Zweiten Stufe: $2^{14} \frac{\text{Bytes}}{\text{Tabelle}} * 2^9$ Tabellen = 2^{23} Bytes = 8 MiB

Insgesamt also: 2^{23} Bytes $+ 2^{13}$ Bytes = 4 KiB $+$ 8 MiB

e) Nehmen wir eine einstufige Tabelle, die 20-bit breite Seitennummern verwendet. Kann mittels der zweistufigen (9+11) Übersetzung mehr Speicher adressiert werden?

Aufgabe 2

d) Wie groß sind die Seitentabellen jeweils, wenn für die Größe eines Eintrags vereinfachend 8 (2^3) Byte angenommen werden?

Tabelle der Ersten Stufe: 2^9 Einträge $* 2^3 \frac{\text{Bytes}}{\text{Eintrag}} = 2^{12}$ Bytes = 4 KiB

Tabelle der Zweiten Stufe: 2^{11} Einträge $* 2^3 \frac{\text{Bytes}}{\text{Eintrag}} = 2^{14}$ Bytes = 16 KiB (pro Tabelle!)

Summer der Tabellen der Zweiten Stufe: $2^{14} \frac{\text{Bytes}}{\text{Tabelle}} * 2^9$ Tabellen = 2^{23} Bytes = 8 MiB

Insgesamt also: 2^{23} Bytes $+ 2^{13}$ Bytes = 4 KiB $+$ 8 MiB

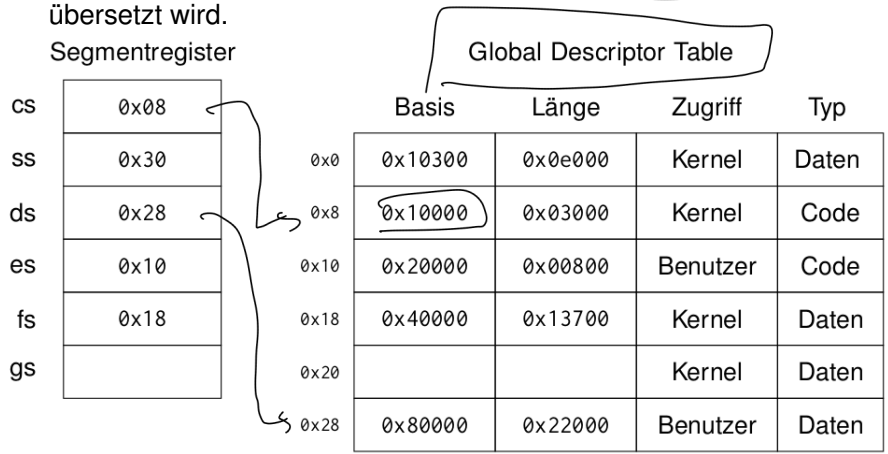
e) Nehmen wir eine einstufige Tabelle, die 20-bit breite Seitennummern verwendet. Kann mittels der zweistufigen (9+11) Übersetzung mehr Speicher adressiert werden?

- Einstufig = $2^{20} * 2^{12} = 2^{32}$
- Zweistufig = $2^9 * 2^{11} * 2^{12} = 2^{32}$

Es kann gleich viel Speicher adressiert werden!

Aufgabe 3
Segmentierung

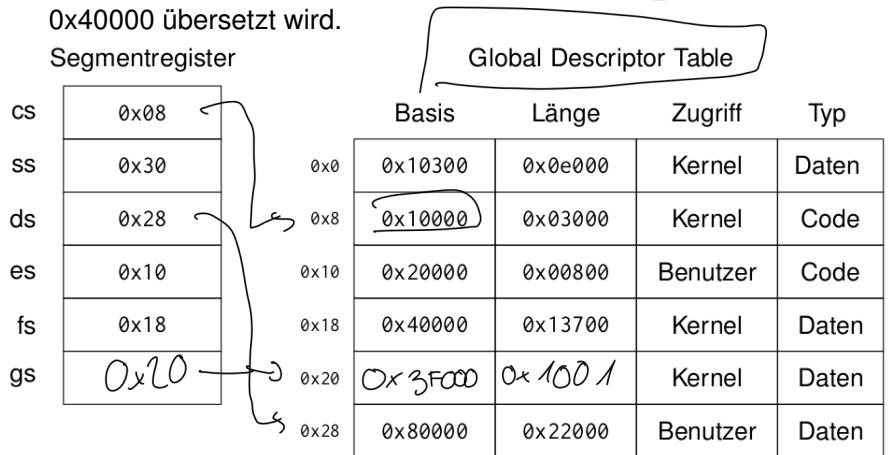
a) Erweitern Sie das Bild, sodass ein Zugriff auf gs:0x1000 auf einen Zugriff auf 0x40000 übersetzt wird.



Aufgabe 3
Segmentierung

CS: 0x10 → 0x10000 + 0x10
= 0x100010

a) Erweitern Sie das Bild oben, sodass ein Zugriff auf gs:0x1000 auf einen Zugriff auf 0x40000 übersetzt wird.



0x40000
- 0x1000
= 0x3F000

Aufgabe 3
Segmentierung

b) Lösen Sie die folgenden Speicherzugriffe auf.
Falls nicht anders angegeben erfolgen die Zugriffe lediglich mit Nutzerrechten.

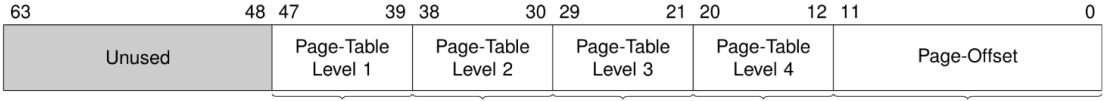
↓ addr.

Lesezugriff auf `ss:0`: SS → 0x30 ↯ nicht in GDT → SEGFAULT

Lesezugriff mit Kernelrechten auf `cs:0x101`:
↳ 0x10000 + 0x101 → 0x10101

Schreibzugriff auf `es:0x1111`:
es: 0x1111 überschreitet Länge von es! → SEGFAULT

Aufgabe 4
Altklausuraufgabe




```
$ cat /proc/cpuinfo
[...]
model name      : Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz
[...]
address sizes   : 39 bits physical, 48 bits virtual
[...]
```

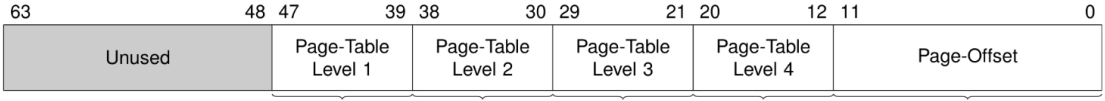
a) Wie viele Bits der physischen Adresse werden für den Frame-Offset verwendet?

12 bit Page-Offset → 12-bit Frame-Offset

b) Wie viele Bits der physischen Adresse werden für die Frame-Nummer verwendet?

39 bit p.a. ; 12-bit offset → 39 − 12 = 27

Aufgabe 4
Altklausuraufgabe




```
$ cat /proc/cpuinfo
[...]
model name      : Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz
[...]
address sizes   : 39 bits physical, 48 bits virtual
[...]
```

c) In wie viele Frames wird der physische Speicher eingeteilt?

39 - 12 = 27-bit für FrameNr. → 2^{27} Frames

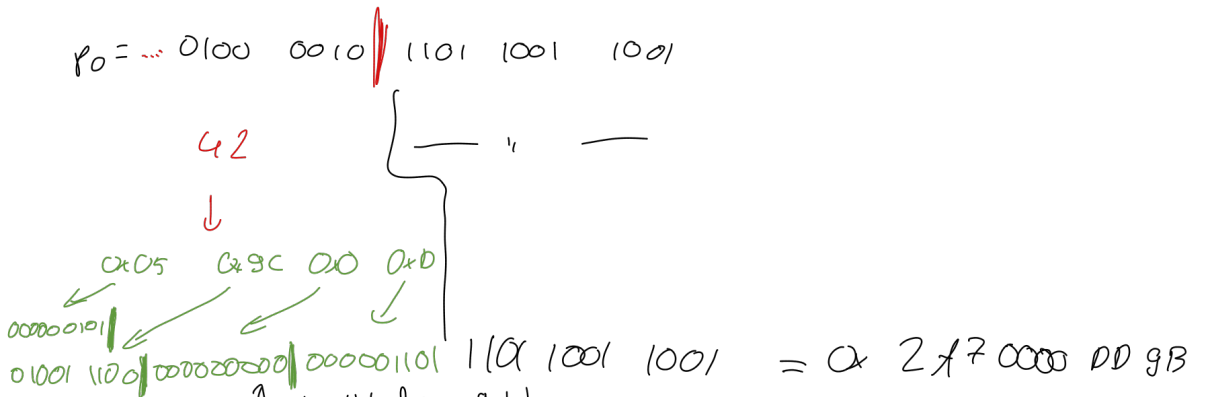
d) Wie viele Pages kann Linux mit diesem Verfahren adressieren?

48 - 12 = 36 ⇒ 2^{36} | $2^9 \cdot 2^9 \cdot 2^9 \cdot 2^9 \rightarrow 2^{36}$

Aufgabe 4
Altklausuraufgabe

39-bit phys; 12-bit offset

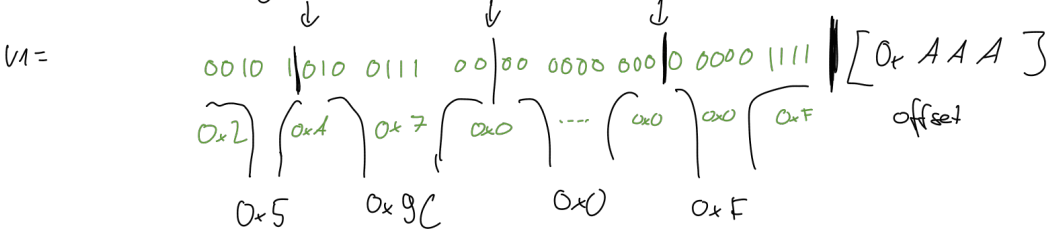
e) Übersetzen Sie die folgende physische Adresse p0 = 0x42D9B in eine virtuelle Adresse v0 und geben Sie diese in hexadezimaler Notation an.



Aufgabe 4
Altklausuraufgabe

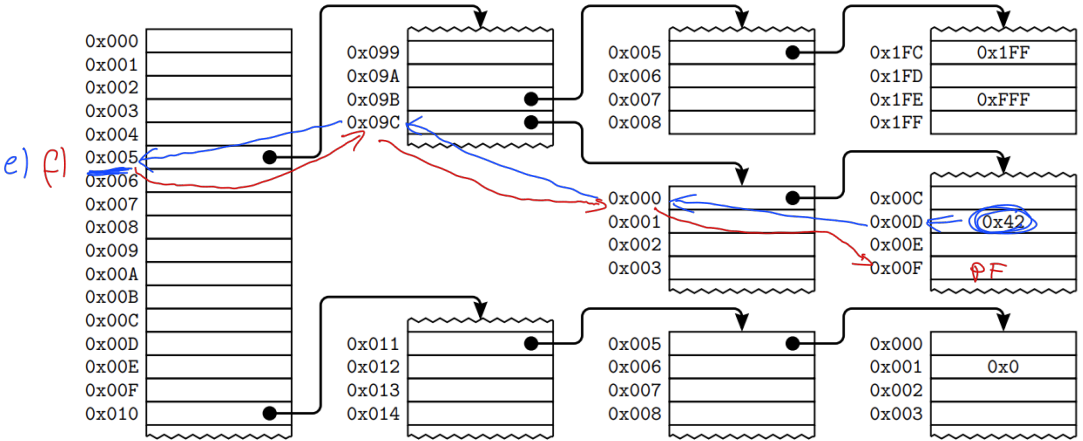
f) Übersetzen Sie die folgende virtuelle Adresse v1 = 0x2A70000FAAA in eine physische Adresse p1.

Bitnotation wichtig, da man Page Nr. nicht direkt ablesen kann ($9 \bmod 4 \neq 0$)



↳ Jetzt Tabelle abgehen: 0x5 → 0x9C → 0x0 → 0xF ↯ 0xF hat keinen Eintrag, Seite nicht eingelagert, also Pagefault!

Aufgabe 4
Altklausuraufgabe



Aufgabe 4
Altklausuraufgabe

g) Gegeben sei die physische Adresse p = 0xD19B. Wie lautet die höchste physische Adresse des Frames, auf das p zeigt, in Hexadezimaldarstellung?

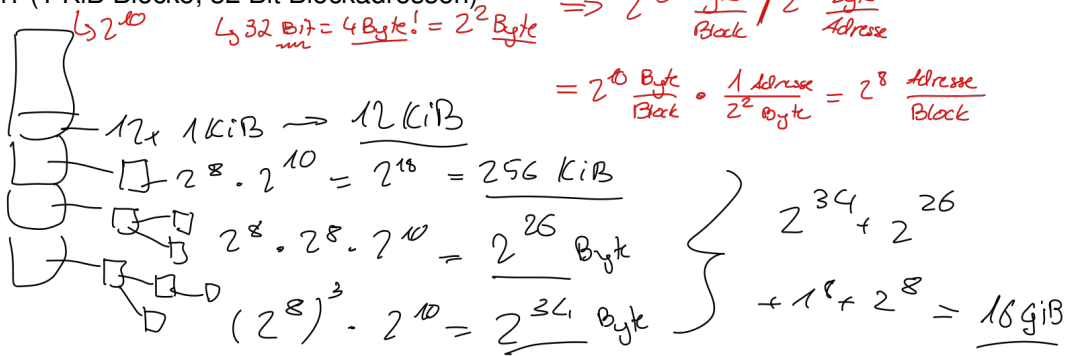
höchste Adresse innerh. Seite = Offset

↳ 12-bit offset → 0b 1111 1111 1111 = 0xFFF ist höchstes offset

Adresse p = 0xD198; 12-bit davon ist offset (0x198), also ist die physische Adresse ohne Offset = 0xD198 (offset auf 0 setzen; das ist der Anfangspunkt des Frames); jetzt Anfangsadresse + max. offset = 0xD000 + 0xFFF = 0xDFFF

Aufgabe 5
Maximale Dateigröße unter UNIX

Welche Größe (in Byte) kann eine Datei maximal haben, wenn im i-node der Datei neben den single-, double- und triple-indirect-Verweisen noch 12 Datenblöcke direkt referenziert werden? (1 KiB Blöcke; 32 Bit Blockadressen)



Aufgabe 5
Maximale Dateigröße unter UNIX

Welche Größe (in Byte) kann eine Datei maximal haben, wenn im i-node der Datei neben den single-, double- und triple-indirect-Verweisen noch 12 Datenblöcke direkt referenziert werden? (1 KiB Blöcke; 32 Bit Blockadressen)

Maximalgröße in Bytes = Anzahl an Blöcken $*$ Bytes pro Block (Blockgröße)
Direkte Blöcke: $12 \rightarrow 12 * 1\ \text{KiB} = 12\ \text{KiB}$

Wie viele Elemente referenziert ein Indirect Block?

Aufgabe 5
Maximale Dateigröße unter UNIX

Welche Größe (in Byte) kann eine Datei maximal haben, wenn im i-node der Datei neben den single-, double- und triple-indirect-Verweisen noch 12 Datenblöcke direkt referenziert werden? (1 KiB Blöcke; 32 Bit Blockadressen)

Maximalgröße in Bytes = Anzahl an Blöcken $*$ Bytes pro Block (Blockgröße)
Direkte Blöcke: $12 \rightarrow 12 * 1$ KiB $= 12$ KiB

Wie viele Elemente referenziert ein Indirect Block?
Blockgröße $\div$ Adresslänge = $2^{10} \div 2^2 = 2^8$, also jeweils 256 andere Blöcke.

Aufgabe 5
Maximale Dateigröße unter UNIX

Welche Größe (in Byte) kann eine Datei maximal haben, wenn im i-node der Datei neben den single-, double- und triple-indirect-Verweisen noch 12 Datenblöcke direkt referenziert werden? (1 KiB Blöcke; 32 Bit Blockadressen)

Maximalgröße in Bytes = Anzahl an Blöcken $*$ Bytes pro Block (Blockgröße)
Direkte Blöcke: $12 \rightarrow 12 * 1$ KiB $= 12$ KiB

Wie viele Elemente referenziert ein Indirect Block?
Blockgröße $\div$ Adresslänge = $2^{10} \div 2^2 = 2^8$, also jeweils 256 andere Blöcke.

Single-Indirect: $256 = 2^8$ Blöcke
Double-Indirect: $256 * 256 = 2^{16}$ Blöcke
Triple-Indirect: $256 * 256 * 256 = 2^{24}$ Blöcke
$\hookrightarrow 2^{24}$ Blöcke $+ 2^{16}$ Blöcke $+ 2^8$ Blöcke $+ 12$ Blöcke $= 16843020$ Blöcke
$\hookrightarrow 16843020$ Blöcke $* 1$ KiB = **16843020 KiB ($\approx$ 16 GiB)**

Aufgabe 2
Maximale Dateigröße unter UNIX

Welche Größe (in Byte) kann eine Datei maximal haben, wenn im i-node der Datei neben den single-, double- und triple-indirect-Verweisen noch 12 Datenblöcke direkt referenziert werden? (1 KiB Blöcke; 32 Bit Blockadressen)

In der Realität meistens 4 KiB Blöcke (2^{12} Bytes), also:
$2^{12} \div 2^2 = 2^{10}$, also jeweils 1024 andere Blöcke referenzierbar.
$\hookrightarrow$ (2^{36} Blöcke $+$ 2^{24} Blöcke $+$ 2^{12} Blöcke $+$ 12 Blöcke) $*$ 4 KiB $\approx$ 4 TiB)