

```

Sep 19 14:20:18 amd64 sshd[20494]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61557
Sep 19 14:27:41 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 01:00:01 amd64 /usr/sbin/cron[29278]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 20 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 02:00:01 amd64 /usr/sbin/cron[30103]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 20 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:46:44 amd64 sshd[6016]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62004
Sep 20 12:46:44 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:48:41 amd64 sshd[6097]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62105
Sep 20 16:04:44 amd64 sshd[6594]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62514
Sep 20 15:27:39 amd64 sshd[6594]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62514
Sep 20 15:27:39 amd64 sshd[6594]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62514
Sep 20 16:37:11 amd64 sshd[10130]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63852
Sep 20 16:37:11 amd64 sshd[10130]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63852
Sep 20 16:38:10 amd64 sshd[10130]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63852
Sep 21 01:00:01 amd64 /usr/sbin/cron[3111]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 21 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 02:00:01 amd64 /usr/sbin/cron[31978]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 21 17:43:26 amd64 sshd[11088]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63397
Sep 21 17:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:53:39 amd64 sshd[31269]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64391
Sep 21 18:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 18:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 01:00:01 amd64 /usr/sbin/cron[4674]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 02:00:01 amd64 /usr/sbin/cron[5493]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 22 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 20:23:21 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 01:00:01 amd64 /usr/sbin/cron[2479]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 02:00:01 amd64 /usr/sbin/cron[2555]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 23 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:05 amd64 sshd[16554]: Accepted publickey for esser from ::ffff:192.168.1.15
Sep 23 18:04:05 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:34 amd64 sshd[6606]: Accepted rsa for esser from ::ffff:87.234.201.207 port 6209
Sep 24 01:00:01 amd64 /usr/sbin/cron[12436]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 24 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 02:00:01 amd64 /usr/sbin/cron[13253]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 24 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 11:15:48 amd64 sshd[2098]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61310
Sep 24 11:15:48 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 13:49:08 amd64 sshd[13197]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61310
Sep 24 13:49:08 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: snd_seq_oss: unsupported module, tainting kernel.
Sep 24 15:42:07 amd64 kernel: snd_seq_oss: unsupported module, tainting kernel.
Sep 24 20:25:31 amd64 sshd[29399]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62566
Sep 24 20:25:31 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 01:00:02 amd64 /usr/sbin/cron[662]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 01:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 02:00:01 amd64 /usr/sbin/cron[1154]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 25 02:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:25 amd64 sshd[1888]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62951
Sep 25 10:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:47 amd64 sshd[1892]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63392
Sep 25 11:30:02 amd64 sshd[1970]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63709
Sep 25 11:30:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:05:37 amd64 sshd[1154]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62951
Sep 25 14:05:37 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:06:10 amd64 sshd[11586]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62951
Sep 25 14:07:17 amd64 sshd[11608]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63392
Sep 25 14:08:23 amd64 sshd[11601]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63709
Sep 25 15:25:33 amd64 sshd[12930]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62778

```

6. Speicherverwaltung

- 6. Speicherverwaltung
 - 6.1 Zusammenhängende Speicherzuteilung
 - 6.2 Nicht-zusammenhängende Speicherzuteilung.
 - 6.3 Demand Paging
 - 6.4 Swapping

Die Folien zu diesem Thema basieren auf den Ausarbeitungen von Prof. Christian Vogt, <http://www.cs.hm.edu/~vogt> und Prof. Claudius Schnorr, <http://www.cs.hm.edu/~schnorr>

Motivation

- Wie setzt sich eine Speicheradresse zusammen?
- Was geschieht bei einem Adresszugriff?
- Wie kann man als Administrator oder Software-Entwickler Nutzen aus Kenntnissen der Speicher-verwaltung ziehen?
 - wie funktioniert Shared Memory?
 - was sind Memory-mapped Files?
- Wie entsteht ein Segmentation Fault?
- Wie erreicht man einen virtuellen Adressraum?

Speicherverwaltung: Gliederung

- 6.1 Zusammenhängende Speicherzuteilung
 - Partitionen fester / variabler Größe
 - Methoden zur Verwaltung des freien Speichers
- 6.2 Nicht zusammenhängende Speicherzuteilung
 - Virtuelle Speicherverwaltung (Paging)
 - Mehrstufiges Paging
- 6.3 Demand Paging
 - Page Faults und deren Behandlung
 - Strategien für die Seitenersetzung
- 6.4 Swapping

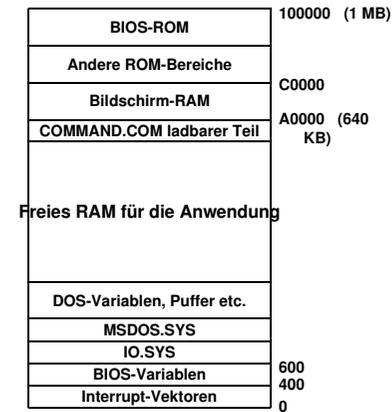
Arten der Speicherverwaltung

- Zwei prinzipielle Arten der Speicherverwaltung:
- Zusammenhängende Speicherzuteilung
 - Jede Anforderung eines Prozesses nach einer bestimmten Menge Speicher muss das BS durch zusammenhängenden (contiguous) Speicher erfüllen.
 - Nicht zusammenhängende Speicherzuteilung
 - BS kann Speicheranforderung durch Zuweisung mehrerer kleiner Speicherbereiche erfüllen, die zusammen die geforderte Menge Speicher ergeben.
 - Wiederauffinden der verstreuten Speicherbereiche ist eine Aufgabe der Speicherverwaltung.

Arten der Speicherverwaltung

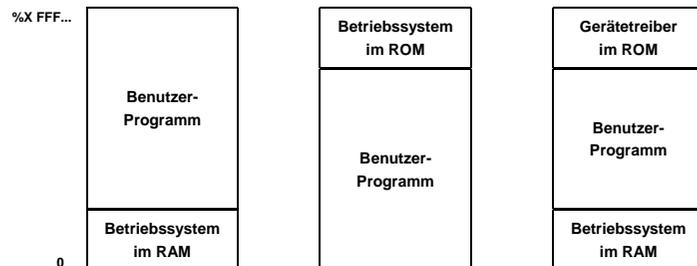
- Heute: Hauptspeicher fast immer **nicht-zusammenhängend** (virtuelle Speicherverwaltung).
- Zusammenhängende Speicherverwaltung bei
 - Verwaltung von Plattenplatz,
 - Verwaltung des Platzes in Page- und Swap-Dateien.

Speicherbelegung unter MS-DOS



Single-Tasking ohne Swapping und Paging

- Aufteilung des Speichers (RAM und ROM) in Bereiche für
 - das Betriebssystem
 - ein Benutzerprogramm
- Beim Beenden eines Programmes: Überlagern des Programmbereichs durch ein neues Programm.

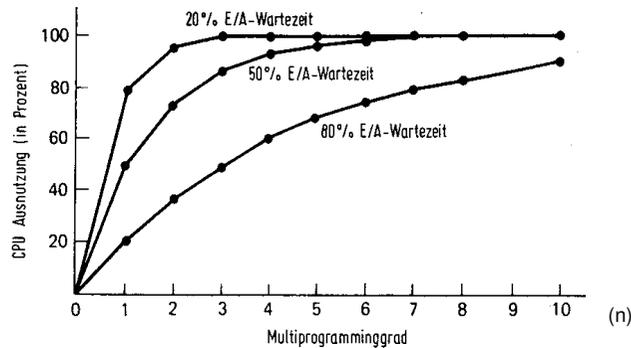


Multiprogramming

- Programme verbringen einen großen Teil ihrer Ausführungszeit mit Warten (auf I/O etc.).
- **Auslagern (Swapping)** auf Platte bei jedem Warten ist ineffizient.
- Lösung: mehrere Programme gleichzeitig im Speicher.
- Voraussetzung: **verschiebbarer Code** und **Speicherschutz**.

Multiprogramming

- Wenn n Programme den Bruchteil p ihrer Zeit mit Warten verbringen, ist die CPU-Ausnutzung = $1 - p^n$.



Relocator für Z80, 20 Jahre alt



für 464-664-6128

(Nicht ganz ernst gemeinter) Literaturhinweis:

H.-G. Eßer: *MC-Relocator – Programmverschiebung leicht gemacht*, Schneider CPC International 03/1988, DMV-Verlag, S. 67 f.

```

1000 ' ***** [1164]
1010 ' *** RELOCATOR V1.0 *** [918]
1030 ' *** [51]
1040 ' *** Geschrieben in *** [2088]
1050 ' *** Oktober 1987 von *** [900]
1070 ' *** Hans-Georg Esser *** [51]
1080 ' ***** [1164]
1100 ' ***** [51]
1120 ' ***** [1164]
1130 ' ***** [117]
1140 MODE 2:ZONE 3 [1209]
1150 WINDOW #1,48,76,7,10 [1139]
1160 WINDOW #2,48,76,12,18 [1223]
1170 FOR N#1 TO 2:PRINT #N,D:PAPER #N,1:NEXT [2164]
1180 PRINT CHR$(24);STRING$(80,216); [8496]
Relocator V1.0 (c) 1987 by Hans-Geor
g Esser Relocator V1.0 ;STRINGS
(80,216);CHR$(24) [1164]
1190 LOCATE 1,22 [730]
1200 PRINT CHR$(24);STRING$(80,216);" [8529]
Relocator V1.0 (c) 1987 by Hans-Geor
g Esser Relocator v1.0 ;STRINGS
(80,216);CHR$(24) [1040]
1210 WINDOW 1,48,8,20 [576]
1220 CLR #1:CLR #2 [2624]
1230 LOCATE #1,2,2:PRINT #1,"Aktuelle Adr [2624]
ese:" [3236]
1240 LOCATE #2,2,2:PRINT #2,"Geänderte Ad [3236]
ressen:";PRINT #2 [4640]
1250 LOCATE 5,3:LINE INPUT "Name des Bin [4640]
nr-Files" > ,file$ [4581]
1260 LOCATE 5,5:INPUT "Ursprungsadresse [4581]
" > ,adr1 [3793]
1270 LOCATE 5,7:INPUT "Programm-Länge (in [3793]
Bytes) > ,length [3384]
1280 LOCATE 5,9:INPUT "Neue Adresse [3384]
" > ,adr2 [2949]
1290 LOCATE 5,11:LINE INPUT "Name des neu [2949]
n Files" > ,file2$ [5399]
1300 LOCATE 5,14:PRINT "Relokation wird du [5399]
rchgeführt." [1453]
1310 ' Einlesen [1453]
1320 MEMORY adr1=1 [108]
1330 LOAD file$,adr1 [518]
1340 diff=adr2-adr1 [566]
1350 ' Alle Adressen messen um diff nach [4774]
oben verschoben werden. [2180]
1360 FOR n=adr1 TO adr1+length [2013]
1370 LOCATE #1,20,2:PRINT #1,"&":HEX$(n,4) [2013]
;32,2,XX [2409]
1380 GOSUB 1480 ' Überprüfen und ggf. ae [2409]
ndern [1815]
1390 NEXT n [1815]
1400 dum$="dummy dum" [1815]
1410 BYTE dum$,b,adr1,length [1815]
1420 MEMORY adr2=1 [1815]
1430 LOAD dum$,adr2 [1815]
1440 LERA,80um$ [988]
1450 SAVE #1:um$,b,adr1,length [1561]
1460 LOCATE 5,14:PRINT "Die Relokation wur [1540]
de durchgeführt." [882]
1470 END ' Programm-Ende [933]
1480 ' Aktuelle Adresse überprüfen [933]
1490 pr=HEX$(PEEK(n),2) [1242]
1500 byte=VAL("&"+HEX$(PEEK(n+2),2)+HEX$(P [2760]
EEK(n+1),2)) [920]
1510 RESTORE 1830 [1830]
1520 rdb$="" [1830]
1530 WHILE rdb$<"XX" [1542]
1540 READ rdb$ [483]
1550 IF pr#rdb$ THEN 1640 [763]
1560 WEND [390]
1570 ' Untersuchung auf 2-Byte-Befehl [2626]
1580 dr=HEX$(PEEK(n),2)+HEX$(PEEK(n+1),2) [1841]
1590 byte=VAL("&"+HEX$(PEEK(n+3),2)+HEX$(P [1647]
EEK(n+2),2)) [815]
1600 RESTORE 1840 [1840]
1610 rdb$="" [2416]
1620 WHILE rdb$<"XXXX":READ rdb$:IF [2416]
rdb$#rdb$ THEN 1730 [390]
1630 RETURN [390]
1640 ' Bezug auf Adresse innerhalb des Pro [3010]
gramms? [2412]
1650 IF byte<adr1 OR byte>adr1+length THEN [2412]
RETURN [815]
1660 ' Verschieben [584]
1670 byte=byte+diff [1060]
1680 b=HEX$(byte,4) [1198]
1690 POKE #2,VAL("&"+LEFT$(b$,2)) [1323]
1700 POKE #1,VAL("&"+RIGHT$(b$,2)) [2014]
1710 PRINT #2, " &":HEX$(n,4) [553]
1720 RETURN [1842]
1730 ' 2-Byte Befehl auswerten [2412]
1740 IF byte<adr1 OR byte>adr1+length THEN [2412]
RETURN [815]
1750 ' Verschieben [584]
1760 byte=byte+diff [1060]
1770 b=HEX$(byte,4) [1198]
1780 POKE #3,VAL("&"+LEFT$(b$,2)) [992]
1790 POKE #2,VAL("&"+RIGHT$(b$,2)) [764]
1800 PRINT #2, " &":HEX$(n,4) [2014]
1810 RETURN [553]
1820 ' Daten [442]
1830 DATA CD,CC,FC,D4,C4,F4,EC,E4,CC,C3,DA [6617]
,F4,C2,C2,F4,E4,E2,CA,E9,DA,01,11,21,2A,31 [32,2,2,XX]
1840 DATA ED48,ED58,DD21,DD2A,FD21,FD2A,ED [3306]
7B,ED43,ED53,DD22,FD22,ED73,XXXX [1842]
Listing Relocator
    
```

Code-Verschiebung: Z80-Assembler

```

#base 0xA000 ; Programmstart bei 0xA000

ld hl,data A000 ld hl,0xA011 ; data
ld a,(hl) A003 ld a,(hl)
call print A004 call 0xA008 ; print
ret A007 ret

;print ; Funktion print
cmp a,' ' A008 cmp a,' '
jne ende A00A jne 0xA010 ; ende
call OS_PRINT A00D call OS_PRINT
:ende ; Label ende
ret A010 ret

;data ; Daten
char 'x' A011 char 'x'
    
```

Code-Verschiebung: Z80-Assembler

Probleme mit nachträglicher Relokation:

- Was sind Adressen?
 - Sprung- und Funktionsaufruf-Adressen
 - Zugriff auf Datenbereiche
- Was sieht nur zufällig wie eine Adresse aus?
- Mehrfach indirekte Adressierung („lade den Wert, der in der Speicherstelle steht, deren Adresse hier steht“)

Besser: Lade-System plant Relokation von vornherein ein, merkt sich Stellen mit Adressen

Code-Verschiebung und Speicherschutz (1/2)

- Ein Programm muss an verschiedenen Stellen im Speicher laufen können.

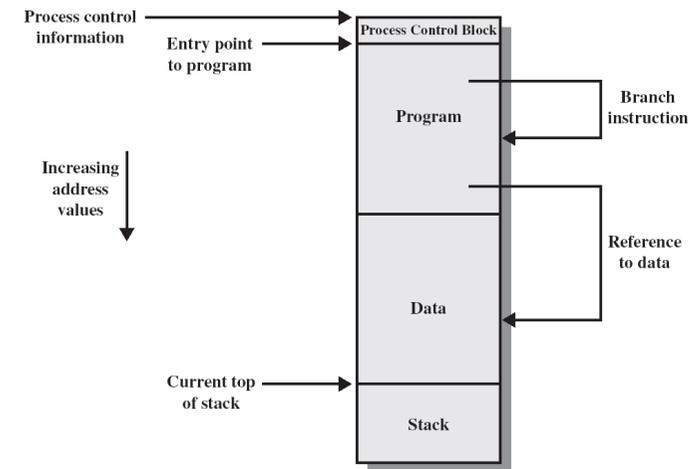
Zwei Möglichkeiten:

- Linker vermerkt, welche Code-Stellen absolute Adressen sind. Beim Laden des Programms werden diese Stellen entsprechend abgeändert.

- Der Rechner hat ein spezielles Hardware-Register, ein sog. **Basisregister**

Bei jeder Adressberechnung (zur Laufzeit) wird die Adresse im Basisregister zu der Adresse im Programm addiert.

Code-Verschiebung: Prozess - Adressierungsanforderungen



Code-Verschiebung und Speicherschutz (2/2)

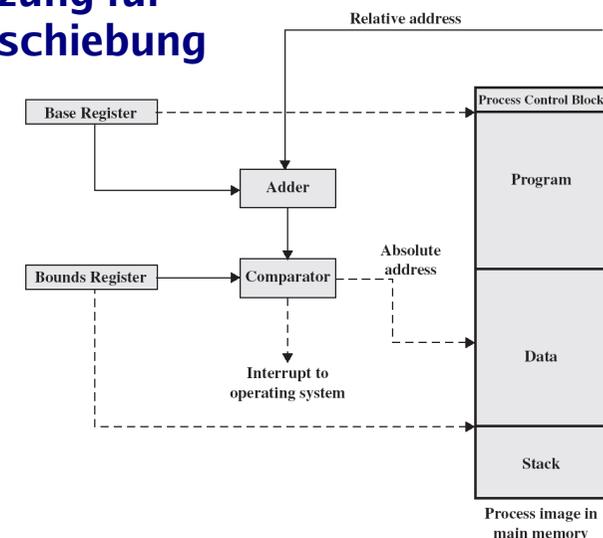
- Ein Programm darf nicht auf den Speicherbereich eines anderen Programms zugreifen.

Zwei Möglichkeiten, dies zu erreichen:

- Schutzcode
- Der Rechner hat ein spezielles Hardware-Register, ein sog. **Längenregister**

Durch Überprüfen des Längenregisters wird festgestellt, ob die zugriffene Adresse in der dem Programm zugewiesenen Partition liegt.

Unterstützung für Code-Verschiebung und Speicherschutz



```

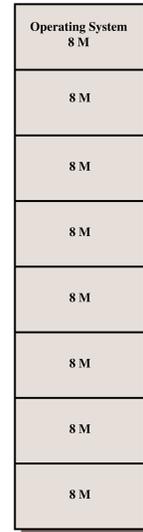
Sep 19 14:07:43 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 01:00:01 amd64 /usr/sbin/cron[29278]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 20 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 02:00:01 amd64 /usr/sbin/cron[30103]: (root) CMD (/sbin/evlogmgr -c "age > *30d**")
Sep 20 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:46:44 amd64 sshd[6516]: Accepted rsa for esser from :ffff:87.234.201.207 port 62004
Sep 20 12:46:44 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:48:43 amd64 sshd[6609]: Accepted rsa for esser from :ffff:87.234.201.207 port 62105
Sep 20 12:54:44 amd64 sshd[6943]: Accepted rsa for esser from :ffff:87.234.201.207 port 62314
Sep 20 15:27:35 amd64 sshd[9077]: Accepted rsa for esser from :ffff:87.234.201.207 port 64242
Sep 20 15:27:35 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:37:11 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:37:11 amd64 sshd[10140]: Accepted rsa for esser from :ffff:87.234.201.207 port 63379
Sep 20 16:38:10 amd64 sshd[10140]: Accepted rsa for esser from :ffff:87.234.201.207 port 63546
Sep 21 01:00:01 amd64 /usr/sbin/cron[17055]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 21 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 02:00:01 amd64 /usr/sbin/cron[17878]: (root) CMD (/sbin/evlogmgr -c "age > *30d**")
Sep 21 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:43:26 amd64 sshd[31088]: Accepted rsa for esser from :ffff:87.234.201.207 port 63397
Sep 21 17:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:53:19 amd64 sshd[31088]: Accepted rsa for esser from :ffff:87.234.201.207 port 64391
Sep 21 18:43:26 amd64 sshd[7653]: STATS: dropped 0
Sep 21 18:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 01:00:01 amd64 /usr/sbin/cron[1491]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 02:00:01 amd64 /usr/sbin/cron[1491]: (root) CMD (/sbin/evlogmgr -c "age > *30d**")
Sep 22 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 20:23:21 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 01:00:01 amd64 /usr/sbin/cron[54739]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 02:00:01 amd64 /usr/sbin/cron[25555]: (root) CMD (/sbin/evlogmgr -c "age > *30d**")
Sep 23 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:05 amd64 sshd[6554]: Accepted rsa for esser from :ffff:87.234.201.207 port 62093
Sep 23 18:04:05 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:14 amd64 sshd[6606]: Accepted rsa for esser from :ffff:87.234.201.207 port 62093
Sep 24 01:00:01 amd64 /usr/sbin/cron[12436]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 24 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 02:00:01 amd64 /usr/sbin/cron[12436]: (root) CMD (/sbin/evlogmgr -c "age > *30d**")
Sep 24 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 11:15:48 amd64 sshd[20998]: Accepted rsa for esser from :ffff:87.234.201.207 port 64456
Sep 24 11:15:48 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 13:49:08 amd64 sshd[21319]: Accepted rsa for esser from :ffff:87.234.201.207 port 61330
Sep 24 13:49:08 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: snd_seq_oss: unsupported module, tainting kernel.
Sep 24 15:42:07 amd64 kernel: snd_seq_oss: unsupported module, tainting kernel.
Sep 24 20:25:31 amd64 sshd[29399]: Accepted rsa for esser from :ffff:87.234.201.207 port 62566
Sep 24 20:25:31 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 01:00:02 amd64 /usr/sbin/cron[162]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 01:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 02:00:02 amd64 /usr/sbin/cron[162]: (root) CMD (/sbin/evlogmgr -c "age > *30d**")
Sep 25 02:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:25 amd64 sshd[8889]: Accepted rsa for esser from :ffff:87.234.201.207 port 64183
Sep 25 10:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:47 amd64 sshd[9321]: Accepted rsa for esser from :ffff:87.234.201.207 port 64253
Sep 25 11:30:02 amd64 sshd[9372]: Accepted rsa for esser from :ffff:87.234.201.207 port 62029
Sep 25 11:30:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 11:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:05:17 amd64 sshd[11564]: Accepted rsa for esser from :ffff:87.234.201.207 port 62822
Sep 25 14:05:17 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:05:19 amd64 sshd[11588]: Accepted rsa for esser from :ffff:87.234.201.207 port 62951
Sep 25 14:07:17 amd64 sshd[11608]: Accepted rsa for esser from :ffff:87.234.201.207 port 63392
Sep 25 14:08:33 amd64 sshd[11630]: Accepted rsa for esser from :ffff:87.234.201.207 port 63709
Sep 25 15:25:33 amd64 sshd[12930]: Accepted rsa for esser from :ffff:87.234.201.207 port 62778

```

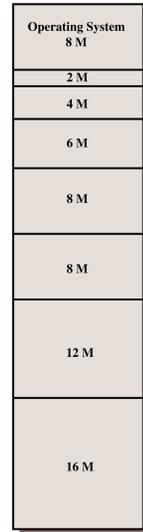
6.1 Zusammenhängende Speicheraufteilung

Aufteilung in feste Partitionen

- **Gleiche Größe**
 - Verschwendung bei kleinen Programmen
 - Programme passen in jede freie Partition
- **Ungleiche Größe**
 - Bessere Speichernutzung
 - Evtl. ungeschickte Belegung



(a) Equal-size partitions



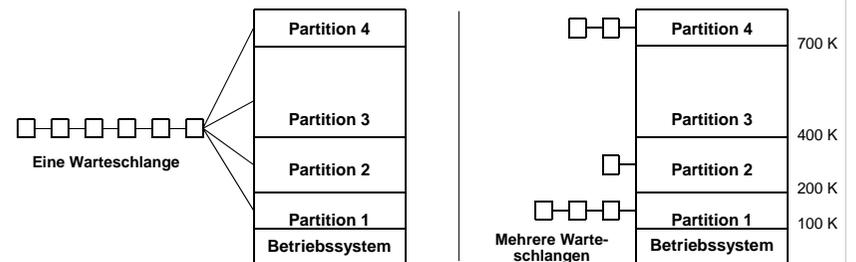
(b) Unequal-size partitions

Aufteilung in feste Partitionen

- Aufteilung des Speichers in Partitionen fester (gleicher oder ungleicher) Größe.
- Zuweisung eines Programms zu einer freien Partition. Alternativen:
 - Erstes Programm, das in die freie Partition passt (eine Warteschlange)
 - FIFO für jede einzelne Partition (mehrere Warteschlangen)
 - Größtes Programm, das in die freie Partition passt
 - Nachteil: kleine Programme werden zurückgestellt
 - Lösung: kleines Programm nur k-mal überspringen

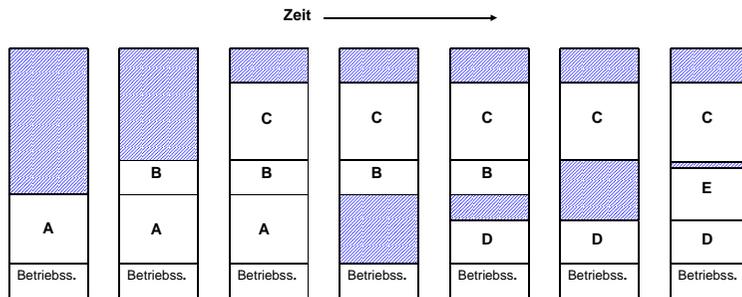
Aufteilung in feste Partitionen

- Evtl. große freie Bereiche in der Partition: **interne Fragmentierung**.



Aufteilung in variable Partitionen (1)

- Anzahl und Größe der Partitionen werden dynamisch festgelegt.



Aufteilung in variable Partitionen (3)

Was ist, wenn der Speicherbedarf eines Prozesses wächst?

- Wenn neben der Partition ein freier Speicherbereich ist, kann die Partition vergrößert werden.
- Es kann eine neue, ausreichend große Partition reserviert und der Inhalt dorthin verschoben werden.

Aufteilung in variable Partitionen (2)

- Es bleiben u. U. viele kleine freie Bereiche im Hauptspeicher (Löcher). Dies wird als **externe Fragmentierung** bezeichnet.
- Evtl. müssen diese Löcher durch Verschieben der Partitionen entfernt werden (**memory compaction**).

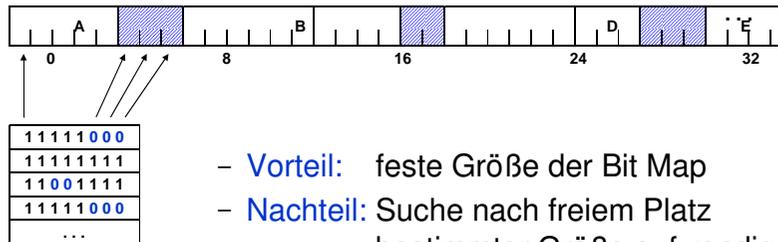
Aufteilung in variable Partitionen (4)

- Wenn keine ausreichend große Partition zur Verfügung steht, müssen ein oder mehrere Prozesse auf Platte ausgelagert werden (**Swapping**).
- Es kann auch dem Prozess von vornherein ein größerer Speicherbereich zugewiesen werden, als angefordert.
 - interne Fragmentierung.
 - Wenn auch dieser größere Bereich nicht ausreicht, muss wieder eines der vorher beschriebenen Verfahren angewandt werden.

Welche Bereiche sind frei?

a) Bit Maps

- Aufteilung des Speichers in Einheiten (einige Byte bis einige KByte)
- Ein Bit pro Einheit: 0 wenn die Einheit frei ist
1 wenn die Einheit belegt ist



Zuteilung freien Speichers

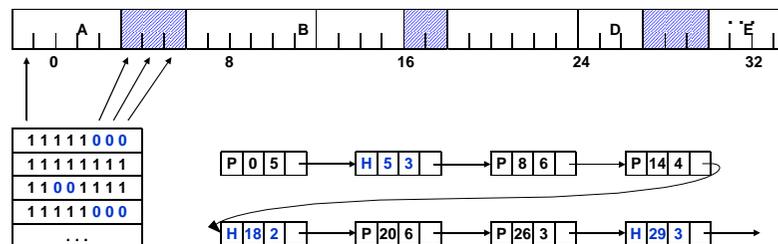
Angeforderter Speicherbereich einer bestimmten Größe kann wie folgt zugeteilt werden:

- **First-Fit-Methode**
 - Der erste genügend große freie Speicherbereich wird zugeordnet.
- **Best-Fit-Methode**
 - Der kleinste ausreichende freie Speicherbereich wird zugeordnet.
 - Aufwendiger, da die gesamte Liste bzw. Bitmap durchsucht werden muss.
 - Starke Fragmentierung des freien Speichers in viele kleine Bereiche.

Welche Bereiche sind frei?

b) Linked Lists

- Verkettete Liste von Beschreibungen der Speicherbereiche:
 - Von Prozess belegt (P) oder frei (H=hole)
 - Anfangsadresse und Länge des Bereichs
 - Zeiger auf nächsten Eintrag



Zuteilung freien Speichers

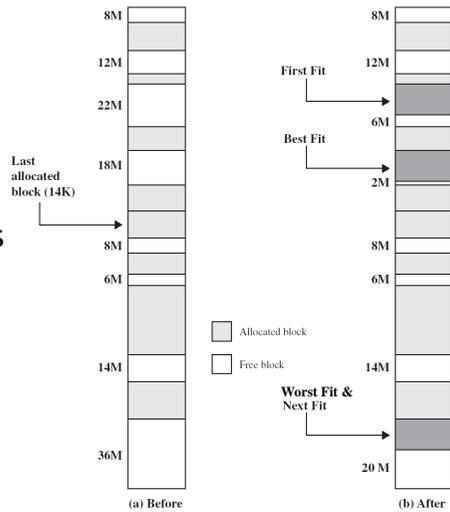
- **Worst-Fit-Methode**
 - Der größte freie Speicherbereich wird zugeteilt.
 - Es bleiben verhältnismäßig große freie Bereiche übrig.
- **Quick-Fit-Methode**
 - Unterhalten von mehreren Listen freier Speicherbereiche für bestimmte Standardgrößen (z. B. 4 KB, 8 KB, 12 KB etc.) führt zu einer schnellen Zuteilung.
 - Bei Freigabe eines Bereiches muss dieser mit evtl. benachbarten freien Bereichen zusammengelegt werden. Dies bedeutet bei mehreren Listen einen erhöhten Aufwand.
 - Spezielle Variante: **Buddy System**

Zuteilung freien Speichers: Beispiel

Beispiel für eine Speicherbelegung

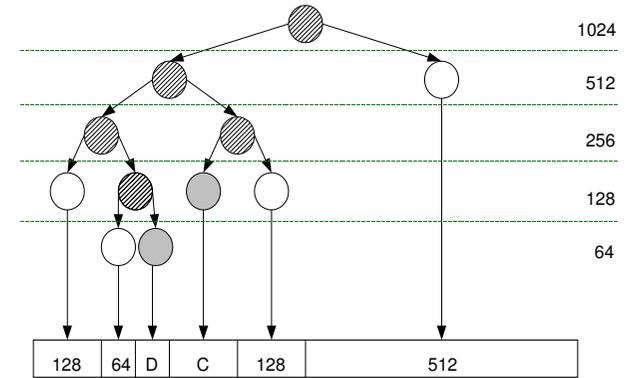
- vor und
- nach

der Allokierung eines 16-MByte-Blocks



Buddy-System: Baum-Repräsentation

Zustand vor der Freigabe von Prozess D



Buddy-System

- Separate Listen freier Bereiche der Größen 1, 2, 4, 8, 16 etc. KByte bis zur Speichergröße.
- Bei Freigabe eines Speicherbereichs muss nur eine der Listen durchsucht werden, um festzustellen, ob der Bereich mit einem anderen freien Bereich zusammengefasst werden kann.
- Blockgröße immer Zweierpotenz ist nicht sehr speichereffizient.

	0	128 K	256K	384 K	512K	640K	768K	896K	1M
Anfang	1024								
Anfrage 70	A	128	256	512					
Anfrage 35	A	B	64	256	512				
Anfrage 80	A	B	64	C	128	512			
Freigabe A	128	B	64	C	128	512			
Anfrage 60	128	B	D	C	128	512			
Freigabe B	128	64	D	C	128	512			
Freigabe D	256		C	128	512				
Freigabe C	1024								

6.2 Nicht zusammenhängende (virtuelle) Speicheraufteilung

```

Sep 19 14:20:18 amd64 sshd[20494]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61557
Sep 19 14:27:41 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 01:00:01 amd64 /usr/sbin/cron[29278]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 20 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 02:00:01 amd64 /usr/sbin/cron[30103]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 20 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:44:44 amd64 sshd[6581]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62004
Sep 20 12:44:44 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:48:41 amd64 sshd[6609]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62105
Sep 20 12:54:44 amd64 sshd[6694]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62354
Sep 20 15:27:35 amd64 sshd[9077]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62442
Sep 20 15:27:35 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:17:11 amd64 sshd[10102]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63375
Sep 20 16:17:11 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:38:10 amd64 sshd[10140]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63546
Sep 21 01:00:01 amd64 /usr/sbin/cron[17055]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 21 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 02:00:01 amd64 /usr/sbin/cron[17878]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 21 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:43:26 amd64 sshd[31088]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63397
Sep 21 17:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:53:39 amd64 sshd[31100]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64191
Sep 21 18:43:26 amd64 sshd[3786]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62105
Sep 21 19:43:26 amd64 sshd[3786]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62105
Sep 22 01:00:01 amd64 /usr/sbin/cron[1935]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 22 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 20:23:21 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 01:00:01 amd64 /usr/sbin/cron[17653]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 01:00:01 amd64 /usr/sbin/cron[17653]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 23 02:00:01 amd64 /usr/sbin/cron[17653]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 23 18:04:05 amd64 sshd[6584]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62105
Sep 23 18:04:05 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:14 amd64 sshd[6606]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62105
Sep 24 01:00:01 amd64 /usr/sbin/cron[1244]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 24 01:00:01 amd64 /usr/sbin/cron[1244]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 24 02:00:01 amd64 /usr/sbin/cron[1244]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 24 11:15:48 amd64 sshd[10999]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64456
Sep 24 11:15:48 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 11:49:08 amd64 syslog-ng[7653]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61330
Sep 24 11:49:08 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: snd_seq_mid_event: unsupported module, tainting kernel.
Sep 24 15:42:07 amd64 kernel: snd_seq_oss: unsupported module, tainting kernel.
Sep 24 20:25:31 amd64 sshd[29399]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62566
Sep 24 20:25:31 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 01:00:02 amd64 /usr/sbin/cron[662]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 01:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 02:00:01 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 25 02:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:25 amd64 sshd[8889]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64183
Sep 25 10:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:47 amd64 sshd[8921]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64253
Sep 25 11:30:02 amd64 sshd[9972]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62029
Sep 25 11:30:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:05:37 amd64 sshd[11554]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62822
Sep 25 14:05:37 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:05:10 amd64 sshd[11586]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62951
Sep 25 14:07:17 amd64 sshd[11608]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63392
Sep 25 14:08:33 amd64 sshd[11630]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63709
Sep 25 15:25:33 amd64 sshd[12930]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62776
    
```

Ausgangslage

- Speicher zu knapp für große Programme
-> Overlay-Programmierung
- Programmteile dynamisch nachladen, wenn sie benötigt werden
- Programmierer muss sich um Aufteilung in Overlays kümmern

Lösung des Problems

- Virtueller Speicher, der das gesamte Programm aufnehmen kann
- Programm sieht Speicherbereich, der ihm zur Verfügung gestellt wurde – wie viel wirklich vorhanden ist, spielt (für das Programm) keine Rolle

Overlay-Programmierung

Turbo Pascal, um 1985-90:

```
program grossesprojekt;  
  
overlay procedure kundendaten;  
...  
  
overlay procedure lagerbestand;  
...  
  
{ Hauptprogramm }  
begin  
  while input <> "exit" do begin  
    case input of  
      "kunden": kundendaten;  
      "lager": lagerbestand;  
    end;  
  end;  
end.  
end.
```



Virtuelle Speicherverwaltung (Paging)

- Aufteilung des Adressraums in **Seiten (pages)** fester Größe und des Hauptspeichers in **Seitenrahmen (page frames)** gleicher Größe.
 - Typische Seitengrößen: 512 Byte bis 8192 Byte (immer Zweierpotenz).
- Der lineare, zusammenhängende Adressraum eines Prozesses („virtueller Adressraum“) wird auf beliebige, nicht zusammenhängende Seitenrahmen abgebildet.
- Eine einzige Liste freier Seitenrahmen wird vom Betriebssystem verwaltet.

Virtuelle Speicherverwaltung (Paging)

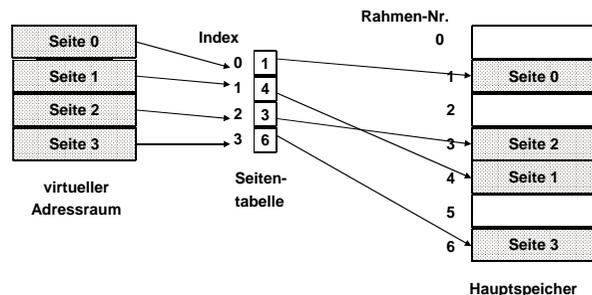
- Die Berechnung der **physikalischen** Speicheradresse aus der vom Programm angegebenen **virtuellen** Adresse
 - geschieht zur Laufzeit des Programms,
 - ist transparent für das Programm,
 - muss von der Hardware unterstützt werden.
- Vorteile** der virtuellen Speicherverwaltung:
 - Einfache Zuteilung von Hauptspeicher.
 - Keine externe Fragmentierung, geringe interne Fragmentierung.
 - Kein Aufwand für den Programmierer.

Virtueller Adressraum (2)

- Die vom Programm verwendeten Adressen werden deshalb auch **virtuelle Adressen** genannt.
- Der **virtuelle Adressraum** eines Programms ist der **lineare, zusammenhängende Adressraum**, der dem Programm zur Verfügung steht.

Virtueller Adressraum (1)

- Beim Paging wird der Zusammenhang zwischen Programmadresse und physikalischer Hauptspeicheradresse erst zur Laufzeit mit Hilfe der Seitentabellen hergestellt.



Adressübersetzung beim Paging (1)

- Die Programmadresse wird in zwei Teile aufgeteilt:
 - eine Seitennummer
 - eine relative Adresse (offset) in der Seite

Beispiel: 32-bit-Adresse bei einer Seitengröße von 4096 ($=2^{12}$) Byte:



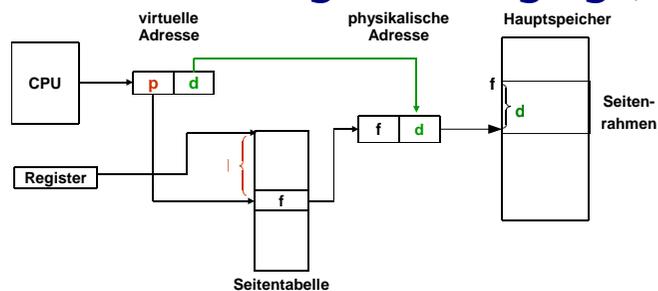
Adressübersetzung beim Paging (2)

- Für jeden Prozess gibt es eine **Seitentabelle** (**page table**). Diese enthält für jede Prozessseite
 - eine Angabe, ob die Seite im Speicher ist,
 - die Nummer des Seitenrahmens im Hauptspeicher, der die Seite enthält.
- Ein spezielles Register enthält die Anfangsadresse der Seitentabelle für den aktuellen Prozess.
- Die Seitennummer wird als Index in die Seitentabelle verwendet.

Virtueller Speicher allgemein (1)

- Mehr Prozesse können effektiv im Speicher gehalten werden
--> **bessere Systemauslastung**
- Ein Prozess kann viel **mehr Speicher** anfordern **als physikalisch verfügbar**

Adressübersetzung beim Paging (3)



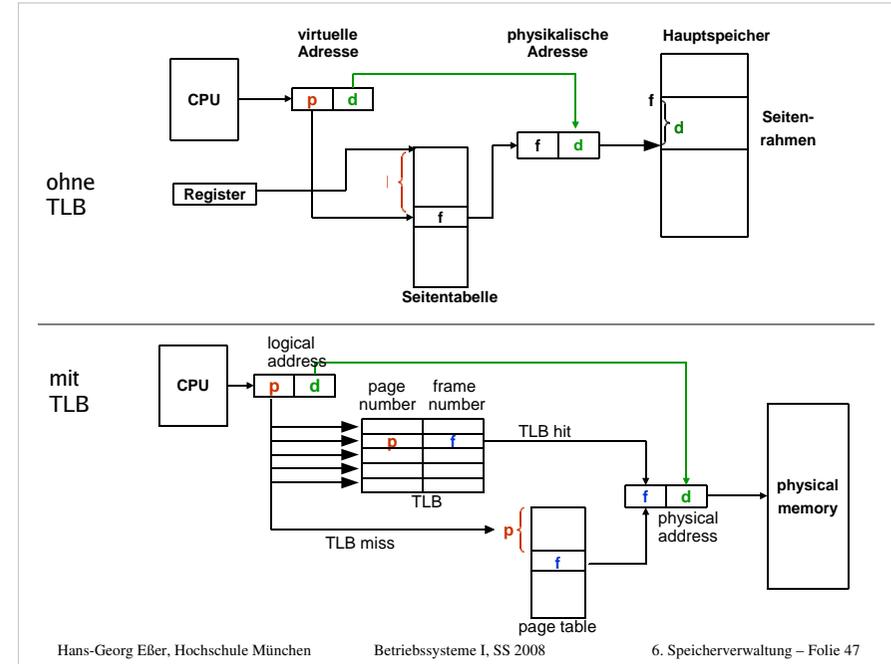
- Für jeden Hauptspeicherzugriff wird ein zusätzlicher Hauptspeicherzugriff auf die Seitentabelle benötigt. Dies muss durch Caches in der Hardware beschleunigt werden!
- Seite nicht im Speicher -> spezielle Exception, einen sog. **page fault (Seitenfehler)** auslösen.

Virtueller Speicher allgemein (2)

- allgemeiner Vorgang:
 - Nur Teile des Prozesses befinden sich im physikalischen Speicher
 - falls Zugriff auf eine Adresse, die ausgelagert ist:
 - BS setzt den Prozess auf blockiert
 - BS setzt eine Disk-I/O-Leseanfrage ab
 - Nach Laden des fehlenden Stücks (Seite oder Segment) wird ein I/O-Interrupt abgesetzt
 - das BS setzt Prozess zuletzt wieder in den Bereit- (Ready-) Zustand

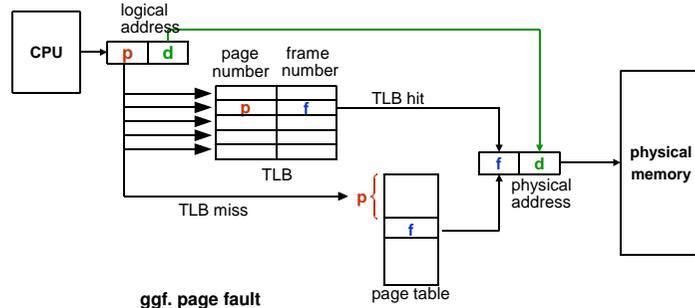
Virtueller Speicher allgemein (3)

- „*thrashing*“ (siehe später): Prozessor verbringt die meiste Zeit mit Ein- und Auslagern von Prozessteilen statt mit der Ausführung von Prozessanweisungen
- **Lokalitätsprinzip:**
 - Zugriffe auf Daten und Programmcode häufig lokal gruppiert;
 - > Annahme gerechtfertigt, dass nur wenige Prozessteile während einer kurzen zeitlichen Periode gleichzeitig vorgehalten werden müssen



Translation Look-Aside Buffer (1)

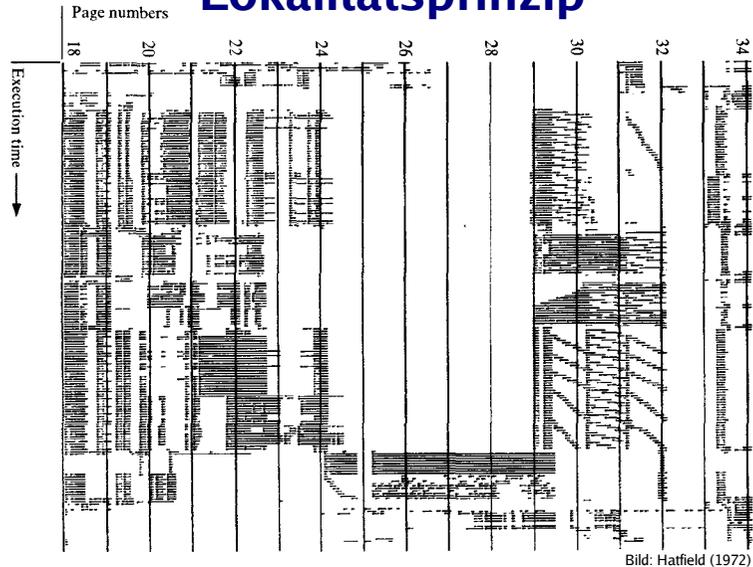
- **Translation Lookaside Buffer (TLB):** schneller Hardware-Cache, mit den zuletzt benutzten Seitentabelleneinträgen
- **Assoziativ-Speicher:** bei Übersetzung einer Adresse wird deren Seitennummer gleichzeitig mit allen Einträgen des TLB verglichen.



Translation Look-Aside Buffer (2)

- **Treffer im TLB**
 - > Speicherzugriff auf Seitentabelle unnötig
- **Fehlertreffer**
 - > Zugriff auf die Seitentabelle
 - Alten Eintrag im TLB durch neuen ersetzen
- **Trefferquote (hit ratio) beeinflusst die durchschnittliche Zeit einer Adressübersetzung.**
- **Lokalitätsprinzip:** Programme greifen meist auf benachbarte Adressen zu
 - > auch bei kleinen TLBs hohe Trefferquoten (typisch: 80-98%).

Lokalitätsprinzip



Hans-Georg Eßer, Hochschule München

Betriebssysteme I, SS 2008

Bild: Hatfield (1972)
6. Speicherverwaltung – Folie 49

Translation Look-Aside Buffer (4)

Was macht hier eigentlich das Betriebssystem?

- Page-Table-Register laden
- Im Falle eines Page Fault: Fehlende Seite aus dem Swap holen und Seitentabelle aktualisieren
- Evtl. vorher: Seitenverdrängung – welche Seite aus dem Hauptspeicher entfernen? (-> später)

Alles andere: Hardware

- Zugriff auf TLB und ggf. auf Seitentabelle
- Wenn Seite im Speicher: Berechnung der phys. Adresse
- Inhalt aus Cache oder ggf. aus Hauptspeicher holen

Hans-Georg Eßer, Hochschule München

Betriebssysteme I, SS 2008

6. Speicherverwaltung – Folie 51

Translation Look-Aside Buffer (3)

- Inhalt des TLB ist prozessspezifisch!
Zwei Möglichkeiten:
 - Jeder Eintrag im TLB enthält ein „valid bit“.
Bei Prozesswechsel (Context Switch) wird der gesamte Inhalt des TLB invalidiert.
 - Jeder Eintrag im TLB enthält Prozessidentifikation (PID), die mit der PID des zugreifenden Prozesses verglichen wird.
- Beispiele für TLB-Größen:
 - Intel 80486: 32 Einträge.
 - Pentium-4, PowerPC-604: 128 Einträge für jeweils Code und Daten.

Hans-Georg Eßer, Hochschule München

Betriebssysteme I, SS 2008

6. Speicherverwaltung – Folie 50

Invertierte Seitentabellen (1)

- Bei großem virtuellen Speicher sehr viele Einträge in der Seitentabelle nötig, z.B. 2^{32} Byte Adressraum, 4 KByte/Seite -> über 1 Millionen Seiteneinträge, also Seitentabelle > 4 MByte (pro Prozess!)
- Platz sparen durch invertierte Seitentabellen:
 - normal: ein Eintrag pro (virtueller) Seite mit Verweis auf den Seitenrahmen (im Hauptspeicher)
 - invertiert: ein Eintrag pro Seitenrahmen mit Verweis auf Tupel (Prozess-ID, virtuelle Seite)

Hans-Georg Eßer, Hochschule München

Betriebssysteme I, SS 2008

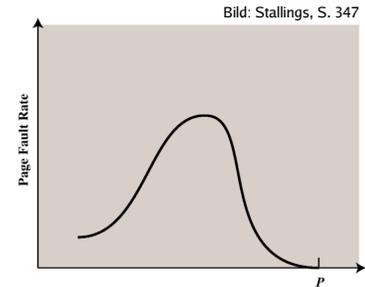
6. Speicherverwaltung – Folie 52

Invertierte Seitentabellen (2)

- Problem: Suche zu Prozess p und seiner Seite n nach dem Eintrag (p,n) in der invertierten Tabelle -> langwierig
- Auch hier TLB einsetzen, um auf „meist genutzte“ Seiten schnell zugreifen zu können
- Bei TLB-Miss hilft aber nichts: Suchen...
- Andere Lösung für Problem der großen Seitentabellen: Mehrstufiges Paging (-> gleich)

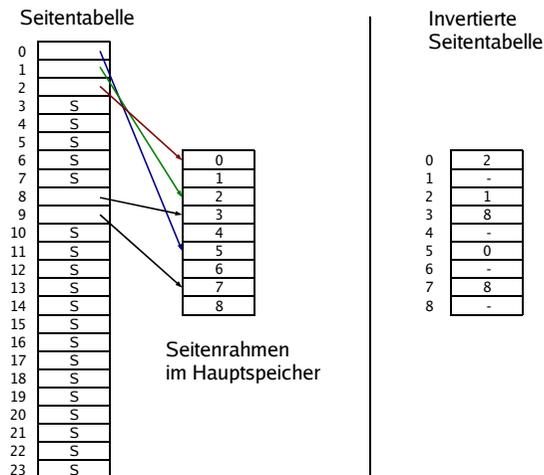
Auswirkungen der Seitengröße

- Interne Fragmentierung: Je kleiner die Seiten, desto geringer die Fragmentierung
- Kleine Seiten -> große Tabellen evtl. Teil der Tabelle ausgelagert -> doppelter Page Fault beim Zugriff auf eine Seite, deren Tabelleneintrag ausgelagert ist
- Lokalitätsprinzip: Kleine Seiten: lokal, wenig Faults. Größere Seiten, nicht mehr lokal. Annäherung der Seitengröße an Gesamtgröße P des Prozessspeichers



(a) Page Size

Invertierte Seitentabellen (3)



Mehrstufiges Paging (1)

Die Seitentabelle kann sehr groß werden.

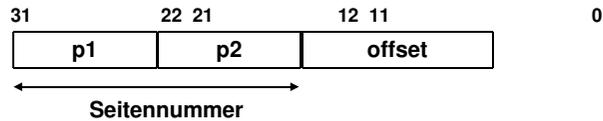
- Beispiel:
- 32-Bit-Adressen,
 - 4 KByte Seitengröße,
 - 4 Byte pro Eintrag

Seitentabelle:
>1 Million Einträge,
4 MByte Größe (pro Prozess!)

Mehrstufiges Paging (2)

- **Zweistufiges Paging:**

- Seitennummer noch einmal unterteilen, z. B.:



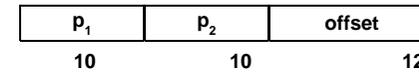
- p_1 : Index in **äußere Seitentabelle**, deren Einträge jeweils auf eine **innere Seitentabelle** zeigen
- p_2 : Index in eine der inneren Seitentabellen, deren Einträge auf Seitenrahmen im Speicher zeigen
- Die **inneren Seitentabellen** müssen **nicht** alle **speicherresident** sein

- Analog dreistufiges Paging etc. implementieren

Mehrstufiges Paging (4)

- Größe der Seitentabellen:

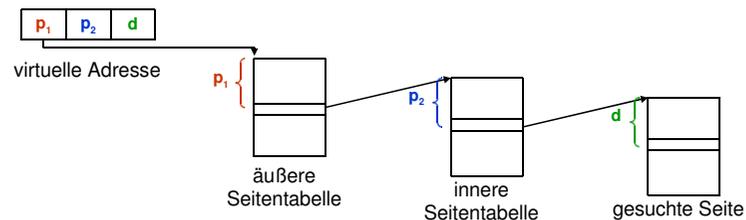
Beispiel:



- Die äußere Seitentabelle hat 1024 Einträge, die auf (potentiell) 1024 innere Seitentabellen zeigen, die wiederum je 1024 Einträge enthalten.
- Bei einer Länge von 4 Byte pro Seitentabelleneintrag ist also jede Seitentabelle genau eine 4-KByte-Seite groß.
- Es werden nur so viele innere Seitentabellen verwendet, wie nötig.

Mehrstufiges Paging (3)

Adressübersetzung bei zweistufigem Paging:



Mehrstufiges Paging (5)

- Jede Adressübersetzung benötigt noch mehr Speicherzugriffe, deshalb ist der Einsatz von **TLBs noch wichtiger**.
- Als Schlüssel für den TLB werden alle Teile der Seitennummer zusammen verwendet (p_1, p_2, \dots).

Speicherschutz beim Paging (1)

- **Schutz vor Zugriff durch andere Prozesse:**
 - Da jeder Prozess eine **eigene** Seitentabelle hat, ist Zugriff auf Speicherbereiche anderer Prozesse nicht möglich. (Dies macht andererseits die Implementierung von gemeinsam benutzten Speicherbereichen aufwendiger.)
- **Schutz vor (z. B.) unberechtigtem Schreiben:**
 - Die Einträge der Seitentabellen enthalten zusätzlich einen **Schutzcode**, der z. B. angibt, ob die Seite gelesen und/oder geschrieben werden darf (evtl. auch noch abhängig davon, ob der Zugriff im User- oder im Kernel-Mode erfolgt).

Seiten-Sharing beim Paging (1)

- **Theoretisch** könnten Einträge verschiedener Seitentabellen auf den gleichen Seitenrahmen zeigen.

Probleme:

- Wie stellt man fest, ob eine Seite bereits von einem anderen Prozess benutzt wird, und in welchem Seitenrahmen sich diese befindet?
- Bei Änderungen (z. B. des verwendeten Seitenrahmens) **wären viele Seitentabellen anzupassen.**

Speicherschutz beim Paging (2)

- Die Seiteneinteilung ist **transparent** für Programmierer !
- Festlegen des Schutzcodes durch Compiler und/oder Linker:
 - Das Programm wird in Abschnitte eingeteilt, deren Größe ein Vielfaches der Seitengröße ist.
 - Pro Abschnitt wird ein Schutzcode für alle Seiten dieses Abschnitts festgelegt und im Kopf der Programmdatei vermerkt.
 - Der Loader setzt die Schutzcodes in den Seitentableneinträgen.

Seiten-Sharing beim Paging (2)

- **Praktisch** wird der gemeinsam zu benutzende Teil des Adressraums
 - entweder als **gemeinsam benutzbares Segment** mit eigener Seitentabelle implementiert (Kombination von Segmentierung und Paging, z. B. bei Unix) oder
 - es werden die gemeinsam zu nutzenden Teile als eine Art Pseudo-Prozess-Adressbereich implementiert, für den es eine eigene (**globale**) Seitentabelle gibt (z. B. bei Windows).

```

Sep 19 14:20:18 amd64 sshd[20494]: Accepted rsa for esser from :ffff:87.234.201.207 port 61557
Sep 19 14:27:41 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 01:00:01 amd64 /usr/sbin/cron[19278]: (root) CMD (/sbin/evlogmgr -c *severity=DEBUC*)
Sep 20 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 02:00:01 amd64 /usr/sbin/cron[10103]: (root) CMD (/sbin/evlogmgr -c 'age > *30d*')
Sep 20 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:46:44 amd64 sshd[6516]: Accepted rsa for esser from :ffff:87.234.201.207 port 62004
Sep 20 12:46:44 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:48:41 amd64 sshd[6609]: Accepted rsa for esser from :ffff:87.234.201.207 port 62105
Sep 20 12:54:44 amd64 sshd[6694]: Accepted rsa for esser from :ffff:87.234.201.207 port 62514
Sep 20 15:27:15 amd64 sshd[9077]: Accepted rsa for esser from :ffff:87.234.201.207 port 64242
Sep 20 15:27:15 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:17:11 amd64 sshd[10102]: Accepted rsa for esser from :ffff:87.234.201.207 port 63375
Sep 20 16:17:11 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:38:10 amd64 sshd[10140]: Accepted rsa for esser from :ffff:87.234.201.207 port 63546
Sep 21 01:00:01 amd64 /usr/sbin/cron[17055]: (root) CMD (/sbin/evlogmgr -c *severity=DEBUC*)
Sep 21 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 02:00:01 amd64 /usr/sbin/cron[17878]: (root) CMD (/sbin/evlogmgr -c 'age > *30d*')
Sep 21 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:43:26 amd64 sshd[31088]: Accepted rsa for esser from :ffff:87.234.201.207 port 63397
Sep 21 17:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:53:39 amd64 sshd[31269]: Accepted rsa for esser from :ffff:87.234.201.207 port 64391
Sep 21 18:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 19:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 01:00:01 amd64 /usr/sbin/cron[4674]: (root) CMD (/sbin/evlogmgr -c *severity=DEBUC*)
Sep 22 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 02:00:01 amd64 /usr/sbin/cron[5495]: (root) CMD (/sbin/evlogmgr -c 'age > *30d*')
Sep 22 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 02:23:21 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 01:00:01 amd64 /usr/sbin/cron[2473]: (root) CMD (/sbin/evlogmgr -c *severity=DEBUC*)
Sep 23 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 02:00:01 amd64 /usr/sbin/cron[25555]: (root) CMD (/sbin/evlogmgr -c 'age > *30d*')
Sep 23 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:05 amd64 sshd[6364]: Accepted publickey for esser from :ffff:192.168.1.5 port 59771 ssh2
Sep 23 18:04:05 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:34 amd64 sshd[6606]: Accepted rsa for esser from :ffff:87.234.201.207 port 62093
Sep 24 01:00:01 amd64 /usr/sbin/cron[12461]: (root) CMD (/sbin/evlogmgr -c *severity=DEBUC*)
Sep 24 01:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 02:00:01 amd64 /usr/sbin/cron[13293]: (root) CMD (/sbin/evlogmgr -c 'age > *30d*')
Sep 24 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 11:15:48 amd64 sshd[20998]: Accepted rsa for esser from :ffff:87.234.201.207 port 64456
Sep 24 11:15:48 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 13:49:08 amd64 sshd[13197]: Accepted rsa for esser from :ffff:87.234.201.207 port 61330
Sep 24 13:49:08 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: snd_seq_pci_avevt: unsupported module, tainting kernel.
Sep 24 15:42:07 amd64 kernel: snd_seq_oss: unsupported module, tainting kernel.
Sep 24 20:25:31 amd64 sshd[29397]: Accepted rsa for esser from :ffff:87.234.201.207 port 62566
Sep 24 20:25:31 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 01:00:02 amd64 /usr/sbin/cron[462]: (root) CMD (/sbin/evlogmgr -c *severity=DEBUC*)
Sep 25 01:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 02:00:01 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c 'age > *30d*')
Sep 25 02:00:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:25 amd64 sshd[8889]: Accepted rsa for esser from :ffff:87.234.201.207 port 64183
Sep 25 10:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:47 amd64 sshd[8921]: Accepted rsa for esser from :ffff:87.234.201.207 port 64253
Sep 25 11:00:02 amd64 sshd[9372]: Accepted rsa for esser from :ffff:87.234.201.207 port 62038
Sep 25 11:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:05:37 amd64 sshd[11554]: Accepted rsa for esser from :ffff:87.234.201.207 port 62822
Sep 25 14:05:37 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 14:06:10 amd64 sshd[11586]: Accepted rsa for esser from :ffff:87.234.201.207 port 62951
Sep 25 14:07:17 amd64 sshd[11600]: Accepted rsa for esser from :ffff:87.234.201.207 port 63392
Sep 25 14:08:13 amd64 sshd[12931]: Accepted rsa for esser from :ffff:87.234.201.207 port 63705
Sep 25 15:25:33 amd64 sshd[12930]: Accepted rsa for esser from :ffff:87.234.201.207 port 62778

```

6.3 Demand Paging

Demand Paging (1)

- Der Adressbereich eines Prozesses muss nicht vollständig im Hauptspeicher sein.
 - Das Lokalitätsprinzip besagt, dass ein Prozess in einer Zeitspanne nur relativ wenige, nahe beieinanderliegende Adressen anspricht.
 - Teile des Programms werden bei einem bestimmten Ablauf möglicherweise gar nicht benötigt (Spezialfälle, Fehlerbehandlungs-routinen etc.).

Demand Paging (2)

- **Demand Paging** bedeutet
 - dass eine Seite nur dann in den Speicher geladen wird, wenn der Prozess sie anspricht,
 - dass eine Seite auch wieder aus dem Speicher entfernt werden kann.
- Vorteile von Demand Paging:
 - Der Adressbereich eines Prozesses kann größer sein als der physikalische Hauptspeicher.
 - Prozesse belegen weniger Platz im Hauptspeicher, somit können mehr Prozesse gleichzeitig aktiv sein.

Voraussetzungen für Demand Paging (1)

- Jeder Eintrag in der Seitentabelle enthält ein **valid bit**, das angibt, ob die Seite im Speicher ist oder nicht.
- Wenn ein Prozess eine Seite anspricht, die nicht im Speicher ist, wird eine spezielle Exception ausgelöst, ein sog. **page fault**.
- Eine Betriebssystem-Routine, der **page fault handler**, lädt bei einem page fault die benötigte Seite in den Speicher.

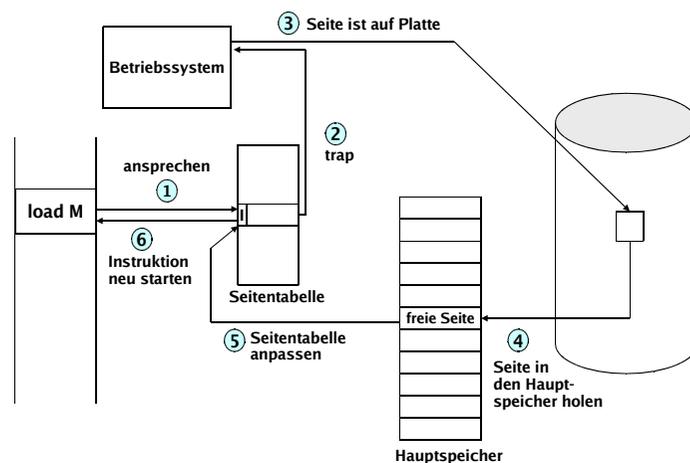
Voraussetzungen für Demand Paging (2)

- Falls **kein freier Seitenrahmen** im Speicher **vorhanden** ist, muss eine andere Seite ersetzt werden. Für die Auswahl der zu ersetzenden Seite muss eine Strategie implementiert werden.
- Die durch den page fault unterbrochene Instruktion muss erneut ausgeführt werden (können).

Seitenersetzung (1)

- Wenn bei einem Page Fault **kein freier Seitenrahmen** zur Verfügung steht, muss das Betriebssystem einen frei machen.
- Ein Algorithmus wählt nach einer bestimmten Strategie diesen Seitenrahmen aus.

Page-Fault-Behandlung



Seitenersetzung (2)

- Falls die zu ersetzende Seite, seit sie zuletzt in den Speicher geholt wurde, verändert wurde, muss ihr aktueller Inhalt gesichert werden:
 - Ein **modify bit** (oder **dirty bit**) im Seitentableneintrag vermerkt, ob die Seite verändert wurde.
 - Eine veränderte Seite wird auf Platte gesichert (im sog. **Page- oder Swap-Bereich**).

Seitenersetzung (3)

- Eine unveränderte Seite kann später - bei Bedarf - wieder von der alten Stelle auf der Platte geladen werden.
- Im Seitentabelleneintrag für die ersetzte Seite wird
 - das **valid bit** gelöscht,
 - vermerkt, von wo die Seite wieder geladen werden kann.

Seiteneretzungsstrategien (2)

Lokale Ersetzung:

Es wird immer eine Seite desjenigen Prozesses ersetzt, der eine neue Seite anfordert.

- Die Zahl der Seiten, die ein Prozess im Speicher belegen kann, ist nach oben beschränkt. Die maximale Anzahl wird pro Prozess festgelegt (z. B. vom Systemverwalter) und kann die Laufzeit eines Prozesses stark beeinflussen.
- Ein Prozess, der viele Page Faults verursacht, z. B. weil er sich nicht an das Lokalitätsprinzip hält, beeinträchtigt nur sich selbst, nicht aber das Gesamtsystem.

Seiteneretzungsstrategien (1)

- Ziel: Es sollen so wenig Page Faults wie möglich auftreten.
- Zwei prinzipielle Arten von Seiteneretzungsstrategien:
 - lokale Ersetzung
 - globale Ersetzung

Seiteneretzungsstrategien (3)

Globale Ersetzung:

Es wird eine beliebige Seite im Speicher ersetzt.

- Prozesse nehmen sich gegenseitig Seiten weg.
- Ein Prozess, der viele Page Faults verursacht, erhält automatisch mehr Speicher. (Dies kann sowohl ein Vorteil als auch ein Nachteil sein.)

Optimale Strategie

Diejenige Seite ersetzen, auf die in Zukunft am längsten nicht zugegriffen wird.

- **Vorteil:** Diese Strategie verursacht die kleinste Zahl an Page Faults.
- **Nachteil:** Diese Strategie ist nicht implementierbar.

Die optimale Strategie kann modellhaft zur Bewertung anderer Strategien benutzt werden.

Least Recently Used (LRU) (1)

Die Seite ersetzen, die **am längsten nicht benutzt worden ist**.

- **Vorteil:** In der Regel weniger Page Faults als FIFO.
- **Nachteil:** Aufwändige Implementierung.

Zwei mögliche Implementierungen:

- mit Zähler
- mit verketteter Liste

First In First Out (FIFO)

Die Seite ersetzen, die schon am längsten im Speicher ist.

- **Vorteil:** Sehr einfach zu implementieren:
 - Es wird eine verkettete Liste der Seiten im Speicher (globale Strategie) bzw. der Seiten eines Prozesses (lokale Strategie) unterhalten.
 - Bei einem Page Fault wird die erste Seite der Liste ersetzt und die neue Seite ans Ende der Liste angefügt.
- **Nachteil:** Die ersetzte Seite kann in dauernder Benutzung sein und gleich wieder angefordert werden.

Least Recently Used (LRU) (2)

- Implementierung mit **Zähler:**
 - Systemweiten Zähler bei jedem Speicherzugriff inkrementieren.
 - Aktuellen Wert des Zählers in einem Feld in der Datenstruktur vermerken, welche die angesprochene Seite beschreibt.
 - Seite mit dem kleinsten Zählerwert ersetzen.

Least Recently Used (LRU) (3)

- Implementierung mit **verketteter Liste**:
 - Eine verkettete Liste enthält alle Seiten.
 - Bei jedem Speicherzugriff wird die angesprochene Seite an den Anfang der Liste gebracht.
(Liste durchsuchen und Reihenfolge ändern, also Zeiger umsetzen!)
 - Die Seite am Ende der Liste wird ersetzt.

Benutzen eines Referenz-Bits (2)

- Mit Referenz-Bits kann man weitere Seitenersetzungsstrategien implementieren, z. B.
 - Modifikationen von LRU, die weniger aufwändig sind.
 - Second-Chance-Algorithmus, eine Verbesserung der FIFO-Strategie.

Benutzen eines Referenz-Bits (1)

- Jeder Seitentableneintrag kann ein **Referenz-Bit** enthalten
 - das bei einem Zugriff auf die Seite gesetzt wird (Hardware),
 - das nach bestimmten Kriterien wieder gelöscht wird (Software).
- Ein Referenz-Bit
 - liefert die Information, ob auf eine Seite seit dem letzten Löschen des Bits zugegriffen wurde,
 - sagt nichts über den Zeitpunkt des Zugriffs auf eine Seite aus,
 - sagt nichts über die Reihenfolge der Zugriffe auf mehrere Seiten aus.

Modifikation von LRU (1)

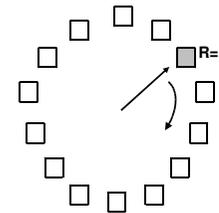
- Es wird ein **binärer Zähler** für jede Seite unterhalten.
- In regelmäßigen Abständen wird
 - jeder Zähler eine Position nach rechts geschoben („**Aging**“),
 - das Referenzbit in das höchste Bit des Zählers kopiert,
 - das Referenzbit gelöscht.

Modifikation von LRU (2)

- Es wird eine der Seiten ersetzt, die den kleinsten Zählerwert enthalten.
 - Bei gleichem Zählerwert ist nicht bekannt, auf welche Seite zuletzt zugegriffen wurde.
 - Länger zurückliegende Zugriffe werden zunächst weniger stark gewichtet und schließlich „vergessen“ (aus dem Zähler hinausgeschoben).

Second-Chance-Algorithmus (2)

- Einfachere Implementierung: „Uhrzeiger“
 - Anordnung der Seiten in einer Ringliste, und Verschieben eines Zeigers statt Umpositionieren eines Listenelements.



Überprüfen der Seite, auf die der Zeiger zeigt:

- Ist $R=0$, wird die Seite ersetzt und der Zeiger weiterbewegt.
- Ist $R=1$, wird R gelöscht, der Zeiger weiterbewegt und die nächste Seite überprüft.

Second-Chance-Algorithmus (1)

- Modifikation des FIFO-Algorithmus: Ist bei der Seitenersetzung das Referenz-Bit der ältesten Seite gesetzt, so wird
 - das Referenz-Bit gelöscht und die Seite am Ende der Liste eingereiht,
 - die gleiche Prüfung für die nächstälteste Seite durchgeführt.
- Es wird also
 - die älteste Seite ersetzt, deren Referenz-Bit gelöscht ist,
 - einer kürzlich benutzten Seite zunächst eine „zweite Chance“ gegeben.

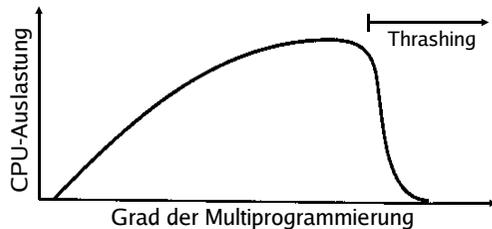
Seitenersetzung, Beispiele

Page address stream	2	3	2	1	5	2	4	5	3	2	5	2																																																																																																																																																																																																																																																																																																																																																												
OPT	<table border="1"><tr><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>4</td><td>4</td><td>4</td><td>2</td><td>2</td><td>2</td><td>2</td></tr><tr><td></td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td></tr><tr><td></td><td></td><td></td><td>1</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td></tr></table>	2	2	2	2	2	2	4	4	4	2	2	2	2		3	3	3	3	3	3	3	3	3	3	3	3				1	5	5	5	5	5	5	5	5	5	<table border="1"><tr><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>4</td><td>4</td><td>4</td><td>2</td><td>2</td><td>2</td><td>2</td></tr><tr><td></td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td></tr><tr><td></td><td></td><td></td><td>1</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td></tr></table>	2	2	2	2	2	2	4	4	4	2	2	2	2		3	3	3	3	3	3	3	3	3	3	3	3				1	5	5	5	5	5	5	5	5	5	<table border="1"><tr><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>4</td><td>4</td><td>4</td><td>2</td><td>2</td><td>2</td><td>2</td></tr><tr><td></td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td></tr><tr><td></td><td></td><td></td><td>1</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td></tr></table>	2	2	2	2	2	2	4	4	4	2	2	2	2		3	3	3	3	3	3	3	3	3	3	3	3				1	5	5	5	5	5	5	5	5	5	<table border="1"><tr><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>4</td><td>4</td><td>4</td><td>2</td><td>2</td><td>2</td><td>2</td></tr><tr><td></td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td></tr><tr><td></td><td></td><td></td><td>1</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td></tr></table>	2	2	2	2	2	2	4	4	4	2	2	2	2		3	3	3	3	3	3	3	3	3	3	3	3				1	5	5	5	5	5	5	5	5	5	<table border="1"><tr><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>4</td><td>4</td><td>4</td><td>2</td><td>2</td><td>2</td><td>2</td></tr><tr><td></td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td></tr><tr><td></td><td></td><td></td><td>1</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td></tr></table>	2	2	2	2	2	2	4	4	4	2	2	2	2		3	3	3	3	3	3	3	3	3	3	3	3				1	5	5	5	5	5	5	5	5	5	<table border="1"><tr><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>4</td><td>4</td><td>4</td><td>2</td><td>2</td><td>2</td><td>2</td></tr><tr><td></td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td></tr><tr><td></td><td></td><td></td><td>1</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td></tr></table>	2	2	2	2	2	2	4	4	4	2	2	2	2		3	3	3	3	3	3	3	3	3	3	3	3				1	5	5	5	5	5	5	5	5	5	<table border="1"><tr><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>4</td><td>4</td><td>4</td><td>2</td><td>2</td><td>2</td><td>2</td></tr><tr><td></td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td></tr><tr><td></td><td></td><td></td><td>1</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td></tr></table>	2	2	2	2	2	2	4	4	4	2	2	2	2		3	3	3	3	3	3	3	3	3	3	3	3				1	5	5	5	5	5	5	5	5	5	<table border="1"><tr><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>4</td><td>4</td><td>4</td><td>2</td><td>2</td><td>2</td><td>2</td></tr><tr><td></td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td></tr><tr><td></td><td></td><td></td><td>1</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td><td>5</td></tr></table>	2	2	2	2	2	2	4	4	4	2	2	2	2		3	3	3	3	3	3	3	3	3	3	3	3				1	5	5	5	5	5	5	5	5	5	<table border="1"><tr><td>2⁰</td><td>2⁰</td><td>2⁰</td><td>2⁰</td><td>5⁰</td><td>5⁰</td><td>5⁰</td><td>5⁰</td><td>3⁰</td><td>3⁰</td><td>3⁰</td><td>3⁰</td><td>3⁰</td></tr><tr><td></td><td>3⁰</td><td>3⁰</td><td>3⁰</td><td>3⁰</td><td>2⁰</td><td>2⁰</td><td>2⁰</td><td>2⁰</td><td>2⁰</td><td>2⁰</td><td>2⁰</td><td>2⁰</td></tr><tr><td></td><td></td><td></td><td>1⁰</td><td>1⁰</td><td>1⁰</td><td>4⁰</td><td>4⁰</td><td>4⁰</td><td>4⁰</td><td>4⁰</td><td>4⁰</td><td>4⁰</td></tr></table>	2 ⁰	2 ⁰	2 ⁰	2 ⁰	5 ⁰	5 ⁰	5 ⁰	5 ⁰	3 ⁰		3 ⁰	3 ⁰	3 ⁰	3 ⁰	2 ⁰				1 ⁰	1 ⁰	1 ⁰	4 ⁰																	
2	2	2	2	2	2	4	4	4	2	2	2	2																																																																																																																																																																																																																																																																																																																																																												
	3	3	3	3	3	3	3	3	3	3	3	3																																																																																																																																																																																																																																																																																																																																																												
			1	5	5	5	5	5	5	5	5	5																																																																																																																																																																																																																																																																																																																																																												
2	2	2	2	2	2	4	4	4	2	2	2	2																																																																																																																																																																																																																																																																																																																																																												
	3	3	3	3	3	3	3	3	3	3	3	3																																																																																																																																																																																																																																																																																																																																																												
			1	5	5	5	5	5	5	5	5	5																																																																																																																																																																																																																																																																																																																																																												
2	2	2	2	2	2	4	4	4	2	2	2	2																																																																																																																																																																																																																																																																																																																																																												
	3	3	3	3	3	3	3	3	3	3	3	3																																																																																																																																																																																																																																																																																																																																																												
			1	5	5	5	5	5	5	5	5	5																																																																																																																																																																																																																																																																																																																																																												
2	2	2	2	2	2	4	4	4	2	2	2	2																																																																																																																																																																																																																																																																																																																																																												
	3	3	3	3	3	3	3	3	3	3	3	3																																																																																																																																																																																																																																																																																																																																																												
			1	5	5	5	5	5	5	5	5	5																																																																																																																																																																																																																																																																																																																																																												
2	2	2	2	2	2	4	4	4	2	2	2	2																																																																																																																																																																																																																																																																																																																																																												
	3	3	3	3	3	3	3	3	3	3	3	3																																																																																																																																																																																																																																																																																																																																																												
			1	5	5	5	5	5	5	5	5	5																																																																																																																																																																																																																																																																																																																																																												
2	2	2	2	2	2	4	4	4	2	2	2	2																																																																																																																																																																																																																																																																																																																																																												
	3	3	3	3	3	3	3	3	3	3	3	3																																																																																																																																																																																																																																																																																																																																																												
			1	5	5	5	5	5	5	5	5	5																																																																																																																																																																																																																																																																																																																																																												
2	2	2	2	2	2	4	4	4	2	2	2	2																																																																																																																																																																																																																																																																																																																																																												
	3	3	3	3	3	3	3	3	3	3	3	3																																																																																																																																																																																																																																																																																																																																																												
			1	5	5	5	5	5	5	5	5	5																																																																																																																																																																																																																																																																																																																																																												
2	2	2	2	2	2	4	4	4	2	2	2	2																																																																																																																																																																																																																																																																																																																																																												
	3	3	3	3	3	3	3	3	3	3	3	3																																																																																																																																																																																																																																																																																																																																																												
			1	5	5	5	5	5	5	5	5	5																																																																																																																																																																																																																																																																																																																																																												
2 ⁰	2 ⁰	2 ⁰	2 ⁰	5 ⁰	5 ⁰	5 ⁰	5 ⁰	3 ⁰																																																																																																																																																																																																																																																																																																																																																																
	3 ⁰	3 ⁰	3 ⁰	3 ⁰	2 ⁰																																																																																																																																																																																																																																																																																																																																																																			
			1 ⁰	1 ⁰	1 ⁰	4 ⁰																																																																																																																																																																																																																																																																																																																																																																		

Thrashing (1)

- **Thrashing** bedeutet, dass ein Prozess **exzessiv viele Page Faults** macht (alle paar tausend Instruktionen).
- Thrashing entsteht, wenn ein Prozess mehr Seiten aktiv benutzt, als ihm Seitenrahmen zur Verfügung stehen.

Thrashing mehrerer Prozesse führt zu niedriger CPU-Auslastung:



```

Sep 19 14:27:41 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 01:00:01 amd64 /usr/sbin/cron[29278]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 20 01:00:01 amd64 /usr/sbin/cron[30103]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 20 02:00:01 amd64 /usr/sbin/cron[30103]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 20 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:44:44 amd64 sshd[6534]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62004
Sep 20 12:44:44 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 12:48:41 amd64 sshd[6609]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62105
Sep 20 12:54:44 amd64 sshd[6694]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62534
Sep 20 15:27:35 amd64 sshd[9077]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64242
Sep 20 15:27:35 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:17:11 amd64 sshd[13024]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63375
Sep 20 16:17:11 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 20 16:38:10 amd64 sshd[10440]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63546
Sep 21 01:00:01 amd64 /usr/sbin/cron[19555]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 21 01:00:01 amd64 /usr/sbin/cron[19555]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 21 01:00:01 amd64 /usr/sbin/cron[19555]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 21 02:00:01 amd64 /usr/sbin/cron[19555]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 21 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:43:26 amd64 sshd[31088]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63397
Sep 21 17:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 21 17:53:39 amd64 sshd[31269]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64391
Sep 21 18:43:26 amd64 /usr/sbin/cron[4674]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 21 18:43:26 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 01:00:01 amd64 /usr/sbin/cron[5493]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 22 01:00:01 amd64 /usr/sbin/cron[5493]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 01:00:01 amd64 /usr/sbin/cron[5493]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 22 02:00:01 amd64 /usr/sbin/cron[5493]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 22 02:00:01 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 22 20:23:21 amd64 /usr/sbin/cron[25555]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 01:00:01 amd64 /usr/sbin/cron[25555]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 01:00:01 amd64 /usr/sbin/cron[25555]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 23 02:00:01 amd64 /usr/sbin/cron[25555]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 23 18:04:05 amd64 sshd[6584]: Accepted publickey for esser from ::ffff:192.168.1.5 port 59771 sshd
Sep 23 18:04:05 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 23 18:04:34 amd64 sshd[6606]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62093
Sep 24 01:00:01 amd64 /usr/sbin/cron[12436]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 24 01:00:01 amd64 /usr/sbin/cron[12436]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 24 02:00:01 amd64 /usr/sbin/cron[13253]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 24 02:00:01 amd64 /usr/sbin/cron[13253]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 24 02:00:01 amd64 /usr/sbin/cron[13253]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 24 11:15:48 amd64 /usr/sbin/cron[6456]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 24 11:15:48 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 11:49:06 amd64 sshd[23197]: Accepted rsa for esser from ::ffff:87.234.201.207 port 61330
Sep 24 11:49:06 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 24 15:42:07 amd64 kernel: snd_seq_mid_event: unsupported module, tainting kernel.
Sep 24 15:42:07 amd64 kernel: snd_seq_oss: unsupported module, tainting kernel.
Sep 24 15:42:07 amd64 kernel: snd_seq_oss: unsupported module, tainting kernel.
Sep 24 20:25:31 amd64 sshd[29399]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62566
Sep 24 20:25:31 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 01:00:02 amd64 /usr/sbin/cron[621]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 01:00:02 amd64 /usr/sbin/cron[621]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 01:00:02 amd64 /usr/sbin/cron[621]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 25 02:00:01 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 02:00:01 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 25 02:00:01 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 10:59:25 amd64 sshd[8889]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64183
Sep 25 10:59:25 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 10:59:47 amd64 sshd[8921]: Accepted rsa for esser from ::ffff:87.234.201.207 port 64253
Sep 25 11:30:02 amd64 sshd[9372]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62029
Sep 25 11:30:02 amd64 syslog-ng[7653]: STATS: dropped 0
Sep 25 11:59:25 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 14:05:37 amd64 sshd[11554]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62822
Sep 25 14:05:37 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 14:05:37 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c "age > *30d*")
Sep 25 14:05:37 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 14:07:17 amd64 sshd[11608]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63392
Sep 25 14:07:17 amd64 /usr/sbin/cron[1484]: (root) CMD (/sbin/evlogmgr -c "severity=DEBUG")
Sep 25 14:08:33 amd64 sshd[11630]: Accepted rsa for esser from ::ffff:87.234.201.207 port 63709
Sep 25 15:25:35 amd64 sshd[12830]: Accepted rsa for esser from ::ffff:87.234.201.207 port 62778
    
```

6.4 Swapping

Thrashing (2)

Lösungen

- Falls noch freier Speicher vorhanden: Zuteilung weiterer Seitenrahmen an den betreffenden Prozess (z. B. durch **globale Ersetzungsstrategie**).
- Falls kein freier Speicher mehr: Auslagern (Swapping) von Prozessen.

Swapping (1)

- **Wichtig:** Es gibt zwei Bedeutungen von Swapping:
 - **Eigentlich:** komplettes Auslagern eines Prozesses (oder bei Segmentierung: eines ganzen Segments) auf die Festplatte (to swap: vertauschen; ein Prozess geht aus dem Hauptspeicher, damit ein anderer herein kommen kann)
 - **Allgemeiner:** Auslagern von Teilen eines Prozesses (Paging: einzelne Seiten) auf die Platte

Swapping (2)

- **Swapping** ist die **zeitweise Auslagerung aller** von einem Prozess benutzten Speicherseiten (oder zumindest kompletter Segmente) auf einen Hintergrundspeicher (Platte), um z. B. bei zu wenig freiem Hauptspeicher Platz zu schaffen.
- Bei zusammenhängender Speicherzuteilung
 - ist Swapping die einzige Möglichkeit, mehr Programme gleichzeitig auszuführen, als im Hauptspeicher Platz haben,
 - muss der Speicher eines Prozesses, der dynamisch wächst, u. U. ausgelagert werden.

Swapping (4)

- Swapping ist ein Vorläufer von Paging
- Da Prozesse immer „ganz oder gar nicht“ im Speicher sind, lassen Swapping-basierte BS keine Prozesse zu, deren Speicherbedarf größer als der Hauptspeicher ist
- Linux und Windows: kein Swapping, sondern Paging (nur aus Traditionsgründen heißt es oft „Swap-Partition“, „Swap-Datei“ etc.)
- sehr alte Unix-Versionen: Swapping

Swapping (3)

- **Kriterien** für die Auslagerung können z. B. sein:
 - Prozesszustand
 - Prozesspriorität
 - Prozessgröße (im Hauptspeicher)
 - Zeit, die der Prozess im Hauptspeicher war
- Die Zuteilung und Verwaltung des Platzes im Swapbereich auf Platte geschieht mit einem der Verfahren der zusammenhängenden Speicherverwaltung.