

LUENTO 10

Muistinhallinnan esimerkit UNIX, Solaris, Linux, W2000

Ch 8.3-8.6 [Stal 05]
Ch 10-11 [Tane 01]

1

UNIX / Solaris (+4BSD) MUISTINHALLINTA

2

UNIX / Solaris (SVR4)

- Vanhoissa UNIXeissa ei virtuaalimuistia
 - heitettävissä aina kokonaisia prosesseja
- Sivutus
 - tarvenotto
 - osittain ytimessä
 - osittain pagedaemon-prosessissa (pid=2)
 - käynnistyy aika-ajoin (esim. 250 ms:n välein) tarkistamaan, onko riittävästi vapaita sivutiloja (25%?)
- Heittävähti
 - swapper-prosessi (pid=0)
 - pagedaemon käynnistää tarvittaessa
 - vain PCB (user structure) ja sivutaulu jää muistiin

3

UNIX/Solaris: Tietorakenteita

Page table - jokaisella prosessilla oma Tbl 8.5 [Stal05]
- alkio per virt. muistin looginen sivu

Missä sivu keskusmuistissa?

Page frame number	Age	Copy on write	Mod-ify	Refer-ence	Valid	Pro-ject
-------------------	-----	---------------	---------	------------	-------	----------

(a) Page table entry

4

UNIX/Solaris: Tietorakenteita

Disk block descriptor - alkio per virt. muistin looginen sivu

Missä sivu tukimuistissa (eli levyllä)?

Swap device number	Device block number	Type of storage
--------------------	---------------------	-----------------

(b) Disk block descriptor

Tbl 8.5 [Stal05]

5

UNIX/Solaris: Tietorakenteita

Tbl 8.5 [Stal05]

Page frame data table - alkio per fyysinen sivukehys, vapaat sivutilat lisäksi free-listassa

Muistin sivukehysten tila?

Page state	Reference count	Logical device	Block number	Pfdata pointer
------------	-----------------	----------------	--------------	----------------

(c) Page frame data table entry

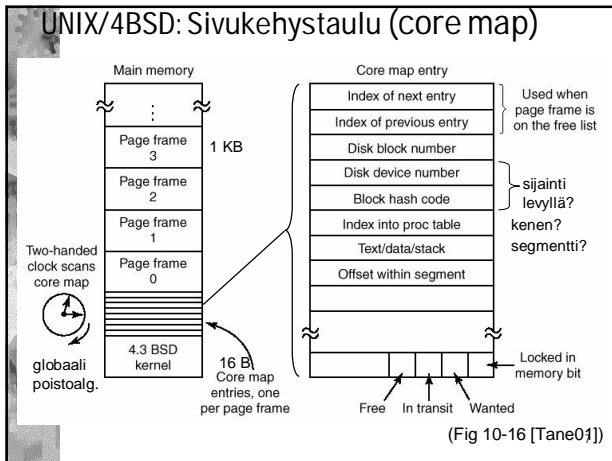
Swap-use table - alkio per levyllä oleva sivu

Tukimuistissa olevan sivun tiedot?

Reference count	Page/storage unit number
-----------------	--------------------------

(d) Swap-use table entry

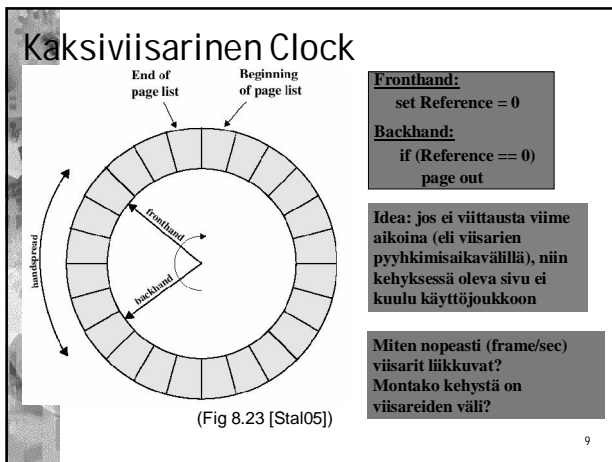
6



UNIX SVR4: Korvausalgoritmi

- Kaksiviisarinen globaali Clock poistoalgoritmi
 - tutki sivukehystaulua
 - prosessille annettujen sivukehysten (sivutilojen) lkm vaihtelee
 - etummainen viisari asettaa *Reference count* = 0
 - perässä tuleva viisari siirtää vapautettavaksi ne, joiden *Reference count* == 0 (edelleen?)

8



Unix SVR4: korvausalgoritmi

- Kaksiviisarinen clock
 - viisarien nopeus (scanrate) välillä [slowscan, fastscan]
 - molemmat viisaarit etenevät samaa tahtia kehys kerrallaan, kunnes tarpeeksi vapaita tilaa
 - reset (fronthead): jos U(backhand)==0, vapautta se ; etene
 - gap (handspread) viisarien välissä
 - jos spread on pieni, vain hyvin usein viitattuihin sivuihin ehtii tulla viitebitti päälle
 - Jos gap = 359°, niin sama kuin 1-viisarinen clock (Fig. 8.16)
 - sopii paremmin suurille muisteille, nopeampi kuin tavallinen clock
 - Ylläpida: minfree < vapaiden sivutilojen lkm < maxfree
 - Jos lkm < minfree, käynnistä swapper heittovaihtamaan prosesseja, jolloin sivutiloja vapautuu äkkiä paljon
 - pisimmän aikaa vähintään 20 sek. idle'nä ollut prosessi?
 - 4 suurimmasta prosessista kauttaen muistissa ollut prosessi?
 - Jos lkm lähellä minfree-arvoa
 - kasvata scanrate-arvoa (vapautta herkemmin, käynnistä algoritmi useammin)
 - Jos lkm lähellä maxfree-arvoa,
 - pienennä scanrate-arvoa (viitebittiin asetukselle aikaa lisää)

(UNIX 4BSD)

10

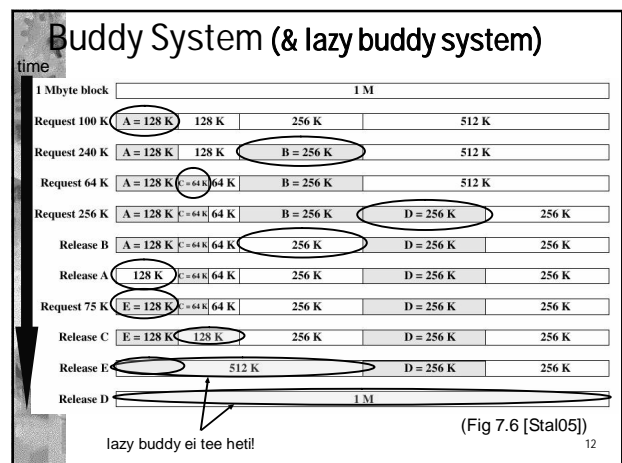
UNIX/Solaris: ytimen (kernel) muisti

- Perusvaraus sivutusta käyttäen
- Sivuja voi lukita muistiin
- Ydin varaa ja vapauttaa paljon pieniä alueita ja puskureita
 - tiedoston polkunimen selvitys
 - zombie-prosessista jälkeenjäävät tiedot
 - dynaaminen varaus: proc, vnode, tiedostokuvaaja
- Ydin varaa ja vapauttaa myös sivun osia
 - sivun sisällä tehtäviä varauksia hallitaan dynaamista tilanvarausta käyttäen (Ch 7 [Stal05])
 - varaus ja vapautus oltava tehokkaita

=> Laiska Buddy System (Lazy Buddy)

- lohkojen yhdistelyä viivästetään, kunnes vapaita lohkoja on "liikaa"

11



Muistiinkuvatut tiedostot

- Memory-mapped files
- Tiedostovälimuisti on muistissa
- Mapataan (kuvataan) virtuaalimuistialue tiedostovälimuistin päälle
- Tiedoston luku/kirjoitus muistin luku/kirjoitus-operaatioilla
 - nopeat!
- Monta prosessia voi mapata helposti saman alueen
 - yhteiskäyttöiset tiedostot (koodi ja/tai data)
 - ks. kuva seuraavalla sivulla
- VM hallinnoi tiedostoja, ei File System!
 - VM voi siirtää sivun levyllä, kun FS pitäisi sen muistissa
 - FS suunniteltu tiedostojen käyttöä varten, VM ei

13

Muistiinkuvattu tiedosto

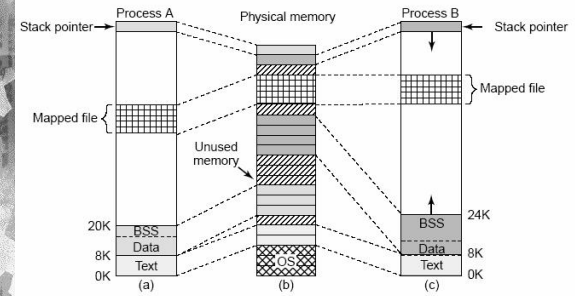


Fig. 10-14. Two processes can share a mapped file. [Tane01]

14

Linux MUISTINHALLINTA



Ch 8.4 [Stal 05]

15

Linux: muistinhallinta



- 32-bittisissä arkkitehtuureissa
 - 3 GB:n virtuaaliavaruus prosesseille
 - 1 GB ylimelle (mm. sivutaulut)
 - fyysisesti todellisen muistin alussa
 - etuoikeutetussa tilassa viitattavissa 4 GB
 - prosessin oma 3 GB
 - ylimen 1 GB
- Virtuaalinen osoiteavaruus jaettu yhtenäisiin alueisiin, joilla joka sivulla samat suojaukset ja ominaisuudet
 - virt. segmentti
- Ytimen pienin allokoinnin yksikkö: slab (<< sivu)
 - ei ole tehokasta varata aina kokonaisia sivuja
 - vain ytimen käyttöön, joka tarvitsee paljon pieniä muistialueita

16

Linux: sivutus

Laitteistoriippumaton toteutus

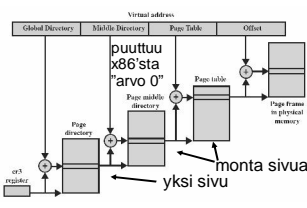
- Alpha: 64b osoitteet, tuki 3:lle tasolle, sivu 8KB
 - offset 13 bittia
- x86: 32b osoitteet, käyttää 2-tasoa, sivu 4KB
 - offset 12 bittia

3-tasoinen sivutaulu

- page directory (PD), 1 sivu
- page middle directory (PMD), monta sivua
- page table (PT), monta sivua

Ylin taso aina muistissa

- säikeen vaihdossa fyysinen osoite MMU:hun (PD)
- muut voivat olla tukimuistissa!



(Fig 8.25 [Stal05])

17

Linux: osoitemuunnos



Jaa osoite 4:ään osaan

- indeksi ylimmän tason hakemistoon (PD-offset)
- indeksi välitason hakemistoon (PMD-offset)
- indeksi sivutauluun (PT-offset)
- siirtymä sivun sisällä (offset)

$$\begin{aligned} \text{PMD_base} &= \text{PD_base} + \text{PD}[\text{PD_offset}] \\ \text{PT_base} &= \text{PMD_base} + \text{PMD}[\text{PMD_offset}] \\ \text{Page_base} &= \text{PT_base} + \text{PT}[\text{PT_offset}] \\ \text{fyys.os} &= \text{Page_base} + \text{offset} \end{aligned}$$

X86-jippo

- välitason hakemiston (PMD) koko vain yksi alkio!
- laitteisto tulkitsee ylimmän tason (PD) alkion osoittavan suoraan alimman tason sivutauluun (PT)
- sopii muillekin laitteistoille, joissa vain 2 tasoa

18

Linux: varausalgoritmi

- Varaa yhtenäinen lohko (area, region) peräkkäisille sivuille
 - tehostaa levyllä noutoa ja takaisin kirjoitusta
 - optimoi siis levyn käyttöaikaa (tilan kustannuksella)
- Buddy System: 1, 2, 4, 8, 16 tai 32 sivutilaa
 - toteutus: taulukko, jossa osoittimet koon mukaisesti vapaiden listoihin
 - paljon sisäistä pirstoutumista
 - tarvitsit 18 sivua, mutta varaat 32 sivua!



19

Linux: noutoalgoritmi

- Tarvenouto
 - ei ennaltanoutoa, ei käyttöjoukon algoritmeja
- Toisaalta, lohkon usea sivu voidaan ehkä hakea kerralla (buddy system)
 - ennakova nouto?
 - ei viitepaikallisuuden vaan levypaikallisuuden mukaan (alueellinen paikallisuus)
 - toivon mukaan lähellä toisiaan

20

Linux: korvausalgoritmi

- Globaali Clock M-bitillä + eräänlainen LRU
 - prosessilla dynaaminen varattujen sivutilojen lkm
- 8 bitin ikälaskuri (age) vrt. U-bitti
 - MMU kasvattaa, kun sivuun viitataan
 - taustaprosessi (kswapd) tutkii sekunnin välein
 - vähentää 1 ikälaskurista jokaisella tutkintakerralla
 - jos vapaana liian vähän, vapauttaa sivutiloja
- Mitä suurempi ikälaskuri, sitä tiuhempaan sivuun viitattu
- Jos ikälaskuri = 0, sivuun ei ole viitattu
- Korvaa se, jonka laskuri pienin
- Käyttää sivujen puskurointia (pagebuffer)



21

Linux memory manager

260 * For choosing which pages to swap out, inode pages carry a
261 * PG_referenced bit, which is set any time the system accesses
262 * that page through the (mapping,index) hash table. This referenced
263 * bit, together with the referenced bit in the page tables, is used
264 * to manipulate page->age and move the page across the active,
265 * inactive_dirty and inactive_clean lists.
266 *
267 * Note that the referenced bit, the page->lru_list_head and the
268 * active, inactive_dirty and inactive_clean lists are protected by
269 * the pagemap_lru_lock, and *NOT* by the usual PG_locked bit!

(include/linux/mm.h)

22

Linux daemon *kswapd* [Stal 05]

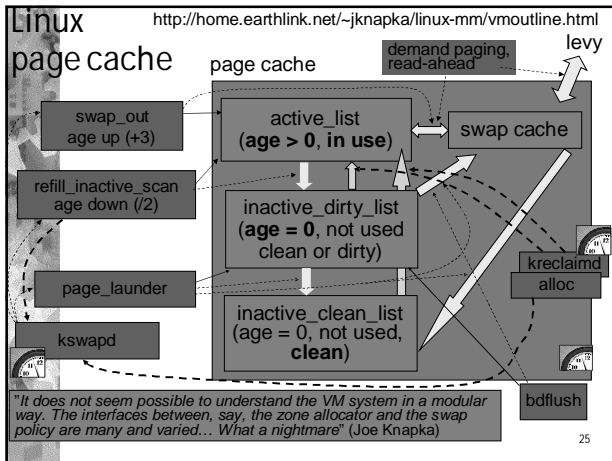
- Suorittaa kerran sekunnissa. Jos tarvitaan lisää vapaita sivukehyksiä, niin käy kaikki käytössä olevat sivut läpi, max 6 kierrosta, kunnes tarpeeksi vapaita
 - etsi käyttämättömiä sivuja modifoidulla clock'illa sivupuskurista (page buffer) ja tiedostovalimuistista
 - etsi yhteiskäytössä olevia käyttämättömiä sivuja
 - etsi tavallisista käytössä olevista sivuista
 - käy sivut läpi 1 prosessi kerrallaan
 - poista käyttämättömät puhtaat hetki
 - pistä käyttämättömät likaiset levyjonoon
 - pistä käytössä olevat likaiset sivupuskuriin

23

Linux daemon *bdflush*

- Herää henkiin aika ajoin, tai eksplisiittisesti
- Jos "liikaa" likaisia sivuja, rupea kirjoittamaan niitä levyille

24



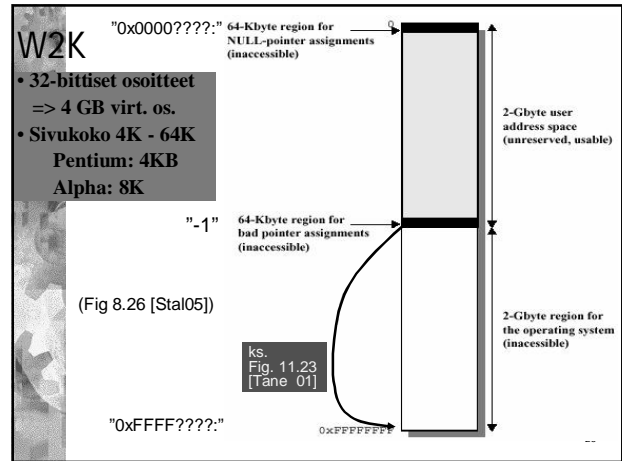
Linux: tilanvaraus ytimessä

- Ydin lukittu pysyvästi muistiin
- Dynaaminen varaus sivuittain
- Dynaamisesti ladattavat moduulit kokonaisina ytimen muistialueelle
 - buddy system sivutilatasolla `kmalloc()`
- Ytimessä usein tarvetta myös pienille lyhytaikaisille varauksille, slab'eille
 - yksittäinen sivu leikattavissa pienempiin osiin (slab)
 - buddy system sivun sisäisesti
 - x86: 32, 64, 128, 252, 508, 2040, 4080 tavua (sivu 4KB)

Slab: laatta, leikata laatoiksi, kattaa laatoilla

Windows 2000 MUISTINHALLINTA

Ch 8.5 [Stal 05]
Ch 11.5 [Tane 01]



W2K: Sivutus

- Ei segmentointia
- Vaihtuva määrä sivukehyksiä / prosessi
 - aluksi kiinteä määrä (muistin määrän mukaan)
 - minimi 20-50, maksimi 45-345 (muistin koosta riippuen)
 - jos vapaata paljon, prosessi voi saada lisää
 - ahneen pitää jättää viimeiset 512 kehystä muille prosesseille
 - jos vapaan muistin määrä vähenee, prosessikohtaista sivutilamäärää vähennetään
- Tarvenouto
 - ei ennallanoutoa
- Levy I/O isompina paloina
 - 1-8 sivua kerrallaan
 - eli siis myös jonkin asteen ennallanouto!

W2K: Sivutus

(Fig 11.26 [Tane01])

- Sivutaulun alkioissa eroja eri järjestelmissä

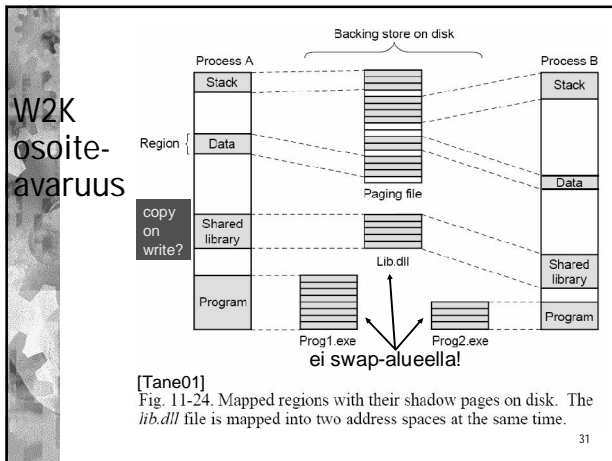
Bits	20	3	1	1	1	1	1	1	1	1
	Page frame		G	L	D	A	C	W	U	W
			t							

G: Page is global to all processes
 L: Large (4-MB) page
 D: Page is dirty
 A: Page has been accessed
 C: Caching enabled/disabled
 Wt: Write through (no caching)
 U: Page is accessible in user mode
 W: Writing to the page permitted
 V: Valid page table entry

Pentium

- Sivun tilat
 - Available
 - Reserved:
 - varaus, jota ei vielä laskettu prosessin osuuteen
 - nopea allokointi, kun tarvitaan (esim. pino)
 - Committed: käytössä, tila myös tukimuistissa

Fig 11.24 [Tane01] (seur. kalvo)

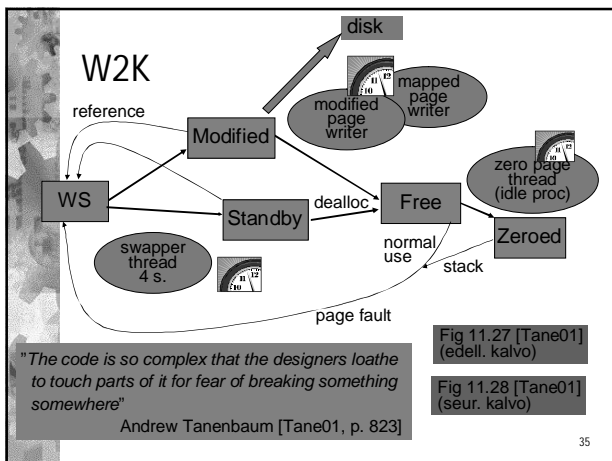
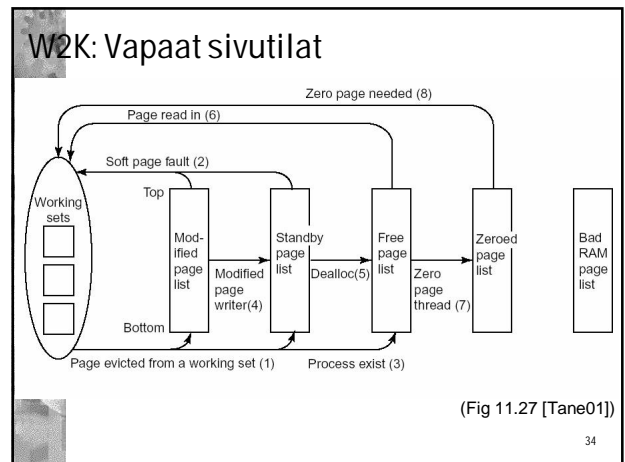


W2K: Korvausalgoritmi

- Käyttöjoukkoalgoritmi, lokaali korvaus
 - käyttöjoukko per prosessi, ei per säie
- Balance set manager -daemon
 - tutkii 1 sek välein, onko riittävästi vapaita sivutiloja
- Working set manager
 - käynnistyy tarvittaessa
 - käyttöjoukon koko määräytyy dynaamisesti
 - vapauttaa prosessikohtaisia sivutiloja
 - aloittaa isoista idle-prosesseista
- Swapper-daemon
 - ajetaan 4 sek välein: etsi prosesseja, joiden kaikki säikeet olleet passiivisia kauan aikaa
- Osa KJ:sta ja puskuriallas lukittu muistiin
- Vapautettujen sivujen puskurointi

W2K Poistettavan sivun valinta

- Joka sivulla laskuri – kauanko sivu käyttämättä
- Yksi suoritin vrt. Linux Age
 - jos ei viitettä, lisää laskuria
 - jos viite, nolaa laskuri
 - modifioitu Clock?
- Monta suorittinta
 - viitebitit eivät toimi, koska lokaaleja suorittimille!
 - FIFO-järjestys (onko hyvä? nyt yleinen tapaus!)
- Neljä "vapaiden sivujen" listaa
 - dirty, clean, free, zeroed free (ja bad)
- Kukin fyysinen sivu Fig 11.27 [Tane01] (seur. kalvo)
 - jossakin käyttöjoukossa
 - jossakin vapaiden sivujen joukossa
 - silti prosessin käytettävissä?



W2K: Sivutilataulu

Page frame database

	State	Cnt	WS	Other	PT	Next
14	Clean					X
13	Dirty					X
12	Clean					X
11	Active		20			
10	Clean					
9	Dirty					
8	Active		4			
7	Dirty					
6	Free					X
5	Free					X
4	Zeroed					X
3	Active		6			
2	Zeroed					
1	Active		14			
0	Zeroed					

Page tables

List headers: Standby, Modified, Free, Zeroed

(Fig 11.28 [Tane01])