TI 3 Übung 8

Tobias Lohse & Luisa Castano
Tutor: Thomas Tegethoff, Do 8-10

Aufgabe 1

(a)

Beim Paging tritt gegenüber Segmenting weniger Fragmentierung auf. Beim Segmenting kann die Sicherheit jedoch besser sein, da den verschiedene festen Segmenten leicht verschiedene Rechte gegeben werden können. In der Praxis wird häufig beider kombiniert.

(b)

FIFO

hit/miss	0	0	0	0	1	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0
Page 1	0				5									0				2		
Page 2	-	1				4									6				3	
Page 3	-		2								1					7				1
Page 4	-			3									2				5			
total hits: 5														total misses: 15						

LRU

hit/miss	0	0	0	0	1	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0
Page 1	0				*0*						1	*1*				7				1
Page 2	-	1				5				*5*					6				3	
Page 3	-		2				4	*4*					2				5			
Page 4	-			3					*3*					0				2		
total hits: 5														total misses: 15						

SC

hit/miss	0	0	0	0	1	0	0	1	1	1	0	1	0	0	0	0	0	0	0	1
Page 1	0				0u						0	1	1u							1u
Page 2	-	1				5					5			2			7		3	
Page 3	-		2				4	4u			4				0			5		
Page 4	-			3					3u		3					6			2	
total hits: 6														total misses: 14						

LFU

hit/miss	0	0	0	0	1	0	0	1	1	1	0	1	0	0	0	0	1	0	1	1
Page 1	00				01						10	11								12
Page 2	-	10				50				51								52		
Page 3	-		20				40	41					20	00	60	70		20		
Page 4	-			30					31										32	
total hits: 8														total misses: 12						

CLIMB

hit/miss	0	0	0	0	1	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0
Page 1	0	*0*				4									6			3		
Page 2	-	1			*5*								0				2			
Page 3	-		2								1					7				1
Page 4	-			3	*3*								2				5			
total hits: 5														total misses: 15						

OPT

hit/miss	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0	1	1	1	0
Page 1	0				5															
Page 2	-	1											0	6	7					1
Page 3	-		2			4						2								
Page 4	-			3																
total hits: 9														total misses: 11						

(c)

Zusammengehörende Programme und Daten, liegen häufig hintereinander im Speicher. Es kann also Sinn machen diese schon zu laden, bevor sie angefragt werden, um page-miss-load routinen zu sparen.

Aufgabe 2:

(a+d)

Testergebnisse:

- Identische Dateien werden als identisch erkannt.
- Bsp1 und Bsp2 werden als unterschiedlich erkannt. Anzahl unterschiedlicher Zeilen wird als 13 ausgegeben, sollte 22 sein.
- Bsp1 und Bsp2 werden nicht als unterschiedlich erkannt.
- Bsp2 und Bsp3 werden als unterschiedlich erkannt. Anzahl unterschiedlicher Zeilen wird als 123 ausgegeben, sollte 244 sein.

Fehler:

- Ein `int8_t` kann maximal die Zahl 127 halten, bei Dateien mir mehr unterschiedlichen Zeilen tritt ein Integer Overflow auf.
  - Zeile: 61
- Es wird angenommen, dass die letzte Zeile ein `'\n'` enthällt. Beziehungsweise der Buffer wird nicht nach jedem schreiben geleert.
  - Zeile: 101, 117
- Dadurch wird die letzte Zeile auch immer als verschieden von einer nicht letzten Zeile angesehen, da sie selbst wenn der Inhalt gleich ist, kein `\n` am Ende hat.
  - Zeile: 90
- Lange Zeilen, die sich erst nach dem 8. Buchstaben unterscheiden werden nicht richtig geprinted, da der Buffer nur geprinted wird, wenn ein Unterschied vorliegt, es werden also nur die letzten Zeichen, die sich unterscheiden geprinted.
  - Zeile: 128
- Im Fall, dass eine Datei früher komplett gelesen wurde, wird in der While Schleife nicht mehr zwischen Bufferinhalt mit und ohne '\n' unterschieden, wie zufor. In allen Fällen wird `printLine` verwendet, was dazu führt, dass zwei Zeilen als eine gezählt werden, wenn sie nicht grade in den Buffer passen.
  - Zeile: 154-159

(b+d)

Weitere Probleme:

- Dateinamenlänge auf `sizeof(fname) = 32` Byte beschränkt.
  - Zeile: 60
- `gets` ist anfällig für Buffer Overflow Angriffe, da es keine Längenbeschränkung des Inputs gibt. (siehe [1])
  - Zeile: 68, 70
- `printf` sollte nicht ohne String-Format auf einerm vom Nutzer angegebenen String eingegeben werden. Format-String Angriffe können Format tokens wie `"%x"`, `"%s"` und `"%n"` nutzen, um Inhalte vom Stack auszugeben und zu manipulieren. (siehe [2])
  - Zeile: 17, 24, 32, 98, 111, 139, 146, 157

(e)

Die geänderten Code Zeilen sind mit `//!` markiert und kommentiert. Es wurden zusätzliche Printstatements zur besseren Leserlichkeit hinzugefügt.

Quellen:

[1] C reference from cppreference.com [2] stackoverflow.com/questions/5428325/issue-with-code-format-string-is-not-a-string-literal