Page Replacement Algorithms

Introduction

This handout shows how the various page replacement algorithms work. We shall call the pages of the program a, b, c, … to distinguish them from the time (1, 2, 3, …).

Fixed Number of Frames

We shall demonstrate these algorithms by running them on the reference string ω = cadbebabcd and assume that, initialy, pages a, b, c, and d occupy frames 0, 1, 2, and 3 respectively. When appropriate, the little arrow → indicates the location of the "pointer" which indicates where the search for the next victim will begin.

First In/First Out (FIFO)

This policy replaces pages in the order of arrival in memory.

time	0	1	2	3	4	5	6	7	8	9	10
ω		c	a	d	b	e	b	a	b	c	d
frame 0	→a	→a	→a	→a	→a	e	e	e	e	→e	d
frame 1	b	b	b	b	b	→b	→b	a	a	a	→a
frame 2	c	c	c	c	c	c	c	→c	b	b	b
frame 3	d	d	d	d	d	d	d	d	→d	c	c
page fault						1		2	3	4	5
page(s) loaded						e		a	b	c	d
page(s) removed						a		b	c	d	e

Optimal (OPT, MIN)

This policy selects for replacement the page that will not be referenced for the longest tile in the future.

time	0	1	2	3	4	5	6	7	8	9	10
ω		c	a	d	b	e	b	a	b	c	d
frame 0	a	a	a	a	a	a	a	a	a	a	d
frame 1	b	b	b	b	b	b	b	b	b	b	b
frame 2	c	c	c	c	c	c	c	c	c	c	c
frame 3	d	d	d	d	d	e	e	e	e	e	e
page fault						1					2
page(s) loaded						e					d
page(s) removed						d					a

Least Recently Used (LRU)

This policy selects for replacement the page that has not been used for the longest period of time.

time	0	1	2	3	4	5	6	7	8	9	10
ω		c	a	d	b	e	b	a	b	c	d
frame 0	a	a	a	a	a	a	a	a	a	a	a
frame 1	b	b	b	b	b	b	b	b	b	b	b
frame 2	c	c	c	c	c	e	e	e	e	e	d
frame 3	d	d	d	d	d	d	d	d	d	c	c
page fault						1				2	3
page(s) loaded					e				c	d	
page(s) removed					c				d	e	
stack (top)		c	a	d	b	e	b	a	b	c	d
		—	c	a	d	b	e	b	a	b	c
		—	—	c	a	d	d	e	e	a	b
stack (bottom)		—	—	—	c	a	a	d	d	e	a

Not-Recently-Used or Not Used Recently (NRU, NUR)

This policy selects for replacement a random page from the following classes (in the order given): not used or modified, not used but modified, used and not modified, used and modified. In the following, assume references 2, 4, and 7 are writes. The two numbers written after each page are the use and modified bits, respectively.)

time	0	1	2	3	4	5	6	7	8	9	10
ω		c	a*	d	b*	e	b	a*	b	c	d
frame 0	a	a/00	a/11	a/11	a/11	a/01	a/01	a/11	a/11	a/01	a/01
frame 1	b	b/00	b/00	b/00	b/11	b/01	b/11	b/11	b/11	b/01	b/01
frame 2	c	c/10	c/10	c/10	c/10	e/10	e/10	e/10	e/10	e/00	d/10
frame 3	d	d/00	d/00	d/10	d/10	d/00	d/00	d/00	d/00	c/10	c/10
page fault						1				2	3
page(s) loaded						e				c	d
page(s) removed						c				d	e

Clock

This policy is similar to LRU and FIFO. Whenever a page is referenced, the use bit is set. When a page must be replaced, the algorithm begins with the page frame pointed to. If the frame's use bit is set, it is cleared and the pointer advanced. If not, the page in that frame is replaced. Here the number after the page is the use bit; we'll assume all pages have been referenced initially.

time	0	1	2	3	4	5	6	7	8	9	10
ω		c	a	d	b	e	b	a	b	c	d
frame 0	a	→a/1	→a/1	→a/1	→a/1	e/1	e/1	e/1	e/1	→e/1	d/1
frame 1	b	b/1	b/1	b/1	b/1	→b/0	→b/1	b/0	b/1	b/1	b/0
frame 2	c	c/1	c/1	c/1	c/1	c/0	c/0	a/1	a/1	a/1	a/0
frame 3	d	d/1	d/1	d/1	d/1	d/0	d/0	→d/0	→d/0	c/1	c/0
page fault						1		2		3	4
page(s) loaded						e		a		c	d
page(s) removed						a		c		d	e

Second-chance Cyclic

This policy merges the clock algorithm and the NRU algorithm. Each page frame has a use and a modified bit. Whenever a page is referenced, the use bit is set; whenever modified, the modify bit is set. When a page must be replaced, the algorithm begins with the page frame pointed to. If the frame's use bit and modify bit are set, the use bit is cleared and the pointer advanced; if the use bit is set but the modify bit is not, the use bit is cleared and the pointer advanced; if the use bit is clear but the modify bit is set, the modify bit is cleared (and the algorithm notes that the page must be copied out before being replaced) and the pointer is advanced; if both the use and modify bits are clear, the page in that frame is replaced. In the following, assume references 2, 4, and 7 are writes. The two numbers written after each page are the use and modified bits, respectively.) Initially, all pages have been used but none are modified.

time	0	1	2	3	4	5	6	7	8	9	10
ω		c	a*	d	b*	e	b	a*	b	c	d
frame 0	a	→a/10	→a/11	→a/11	→a/11	a/00*	a/00*	a/11	a/11	→a/11	a/00*
frame 1	b	b/10	b/10	b/10	b/11	b/00*	b/10*	b/10*	b/10*	b/10*	d/10
frame 2	c	c/10	c/10	c/10	c/10	e/10	e/10	e/10	e/10	e/10	→e/00
frame 3	d	d/10	d/10	d/10	d/10	→d/00	→d/00	→d/00	→d/00	c/10	c/00
page fault						1				2	3
page(s) loaded						e				c	d
page(s) removed						c				d	b

Variable Number of Frames

We shall demonstrate these algorithms by running them on the reference string ω = ccdbcecead.

Working Set (WS)

This policy tries to keep all pages in a process' working set in memory. This table shows the pages consitiuting the working set at each reference. Here, we take the working set to be that set of pages which has been referenced during the last $t = 4$ units. We also assume that a was referenced at time 0, d at time -1, and e at time -2.

time	0	1	2	3	4	5	6	7	8	9	10
ω		c	c	d	b	c	e	c	e	a	d
Page a	a	a	a	a	—	—	—	—	—	a	a
Page b	—	—	—	—	b	b	b	b	—	—	—
Page c	—	c	c	c	c	c	c	c	c	c	c
Page d	d	d	d	d	d	d	d	—	—	—	d
Page e	e	e	—	—	—	—	e	e	e	e	e
page fault		1			2		3			4	5
page(s) loaded		c			b		e			a	d
page(s) removed		e			a		d				

Page Fault Frequency (PFF)

This approximation to the working set policy tries to keep page faulting to some prespecified range. If the time between the current and the previous page fault exceeds some critical value t, then all pages not referenced during that interval are removed. This table shows the pages resident at each reference. Here, we take t = 2 units and assume that initially, a, d, and e are resident.

time	0	1	2	3	4	5	6	7	8	9	10
ω		c	c	d	b	c	e	c	e	a	d
Page a	a	a	a	a	—	—	—	—	—	a	a
Page b	—	—	—	—	b	b	b	b	b	—	—
Page c	—	c	c	c	c	c	c	c	c	c	c
Page d	d	d	d	d	d	d	d	d	d	—	d
Page e	e	e	e	e	—	—	e	e	e	e	e
page fault		1			2		3			4	5
page(s) loaded		c			b		e			a	d
page(s) removed		?			a,e					b,d	