Nicholas Garrett

Professor Fouda

CS 4461

2/27/2022

Homework 4

Chapter16

16.1

seed	addresses [decimal]	translation (or if seg. violation) [decimal]
0	108	Valid in segment 1 and at 492
0	97	not valid
0	53	not valid
0	33	not valid
0	65	not valid
1	17	valid in segment 0 at location 17
1	108	valid in seg. 1 at location 492
1	97	invalid
1	32	invalid
1	63	invalid
2	122	valid in seg. 1, address 506
2	121	valid in seg 1, address 505
2	7	valid in seg. 0, address 07
2	10	valid in seg. 0, address 10
2	106	invalid

.

16.2

The highest virtual address we can access in segment 0 is 19, and the way we can test this is by running the parameters: python3 segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 2 -A 19 -c.

The highest virtual address we can access in seg. 1 is 107, and we can test this through the parameters: python3 segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 2 -A 107 -c.

Finally, the lowest and hightest values we can reach are 0 and 127 respectively, which can be tested through "python3 segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 2 -A 0 -c" and "python3 segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 2 -A 127 -c" respectively.

16.3

For me, I set the bound for seg. 0 to 10 and its limit to 2, and then the bound of seg. 1 to 10 and its limit to 2. The parameters I used were "python3 segmentation.py -a 16 -p 128 -A 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 –b0 10 –l0 2 –b1 10 –l1 2 -c". This gave the intended results.

16.4

By setting the sum total of the limit register values for seg 0 and 1 to be about $\frac{9}{10}$ of the virtual space, the desired result should occur.

16.5

If you set the limits for the two segments to 0, then no addresses are valid.

Chapter 17

17.1

operation	return	free list state
ptr[0] = alloc(3)	1000	[address: 1003, size 97]
free(ptr[0])	0	[addr. 1000, sz 3],[addr 1003, sz 97]
ptr[1] = alloc(5)	1003	[adr 1000, sz 3],[addr 1008, sz 92]
free(ptr[1])	0	[addr 1000, sz 3][addr 1003, sz 5],[addr 1008, sz 92]
ptr[2] = alloc(8)	1008	[addr 1000, sz 3][addr 1003, sz 5][addr1016, sz 84]
free(ptr[2])	0	[addr 1000, sz 3][addr 1003, sz 5][addr1008, sz 8][addr: 1016, sz: 84]
ptr[3] = alloc(8)	1008	[addr 1000, sz 3][addr 1003, sz 5][addr1016, sz 84]
free(ptr[3])	0	[addr 1000, sz 3][addr 1003, sz 5][addr 1008, sz 8][addr: 1016, sz: 84]
ptr[4] = alloc(2)	1000	[addr 1002, sz 1][addr 1003, sz 5][addr 1008, sz 8][addr: 1016, sz: 84]
ptr[5] = alloc(7)	1008	[addr 1002, sz 1][addr 1003, sz 5][addr1008, sz 8][addr: 1016, sz: 84]

I noticed that as time passed, the free list grew when larger memory allocations were made, but when smaller ones were performed, they simply were added to the front of the list.

17.2

Using the WORST fitting policy, the address space simply grows further at the right end with its final address at 1033.

17.3

Using the FIRST fitting policy, because it only searches for the first chunk large enough to fit the request, it does not necessary need to search through all the nodes. The main differences are that there are fewer nodes and the larger block requests lay further back in the list.

17.4

Under the FIRST policy, the ADDRSORT and SIZESORT+ are the same and under the BEST policy, the ADDSORT and SIZESORT- are the same, just with the ADDSORT starting at a lower address and the SIZESORT- starting at the highest address.

17.5

In this case, when the coalescing mode was on, the list coalesced into a single element on the free list. However, when coalescing was switched off, the free list was much larger. At the end, coalescing's list held a single element, while the non-coalescing list held 31 elements.

17.6

As the percent of allocated space rises, the number of nodes in the free list also rises. Eventually, when the percent allocated is high enough, (about 85% in my testing) it simply fails. Then, when the percentage is lower than 50, the size of free list also rizes, though not as dramaticall until it begins to shrink at around 10%.

17.7

Chapter 18

18.1

As page table size grows, the number of pages in the table increase by about 1025 pages. Then, by increasing the table size, the number of pages decreases. A reason why one would not want to simply use large pages in general is a note about space usage and efficiency.

18.2

As the percentage of pages allocated increases, the number of virtual addresses that are valid also increases.

18.3

The first example from the list is a bit unrealistic, as the parameters indicate that there is only space for four pages in the address space.

18.4