CSCE 611 Operating Systems Summer 2017

825008857 Chaiwei Chang

MP2

Design for contagious frame pool

In This machine project assignment, we are asked to implemented a frame pool that represent the manager to allocate and free collections of memory. Our system has the total amount of is32 MB memory, I was implementing by follow the spec that kernel space from 0 to 4 MB and the process space from 4MB to 32MB. There also is a 1MB hole of inaccessible memory starting at 15 MB. The kernel space will be direct mapped from virtual memory to physical and shared between process page tables. User space memory will be managed. Both the frames and the pages will be 4 KB in size.

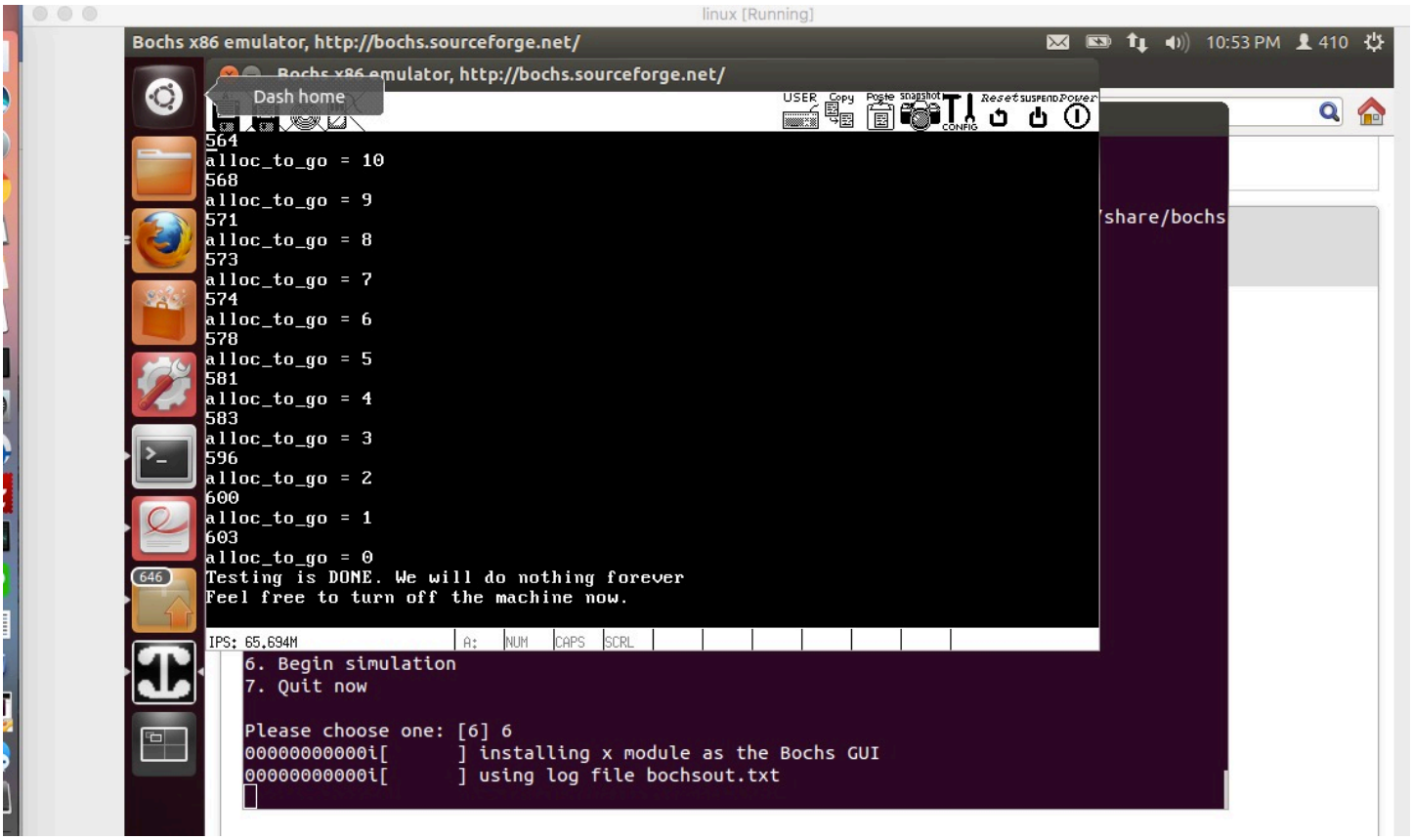
The frame pool manages contiguous frames of memory, memory below 4 MB is managed by our kernel frame pool and memory above 4 MB is managed by our process frame pool. I have created a link list structure to bind all the constructed cont_frame_pool object. With this data structure, I am able to link back to previous or to next frame pool when it does not have the privilege to access that frame. The first nontrivial section is allocating a number of contiguous frames from the frame pool which is get_frame method. Orriginally, I tried to implement by recording one char per entry in the bitmap at a time. However, if these information is going to be stored in kernel frame pool which is 2MB/4KB = 512 frames, it is not enough not even mention memory efficiency. Therefore, I change my strategy into wrap up my mind fighting with bit manipulate. Since now we have 3 stages of usage, FREE, or ALLOCATED, or HEAD-OF-SEQUENCE. Instead of come up with 0xFFFF 16 bits mapping algorithm, I tried to maintain two bitmap one is for use or not the other one is for availability. As shown in the following table, X means not use. Basically, 11 = H, 10 =A and 00 = Free.

Available	Bit_map use	1	0
1	X	HEAD-OF-SEQUENCE (not available)	X
0	X	ALLOCATED	Free(empty)

Finally for the end of this function, if successful, returns the frame number of the first frame; if fails, returns 0. The constructor initializes the frame pool, I have un command the process frame pool constructor for more accurate testing. The info frame number is the frame that will store the management information of the frame pool. If the frame number is 0 the frame pool will store the pool info in the first available frame of the pool. I choose to store management information in the form of 2 bitmaps, each 8 bit and 1024 bits in size (large enough to map our entire memory 32 MB). The reason why I use 2 map is not only to represent more than 2 states. Also, if we only used single bit map and marked inaccessible frames as used initially it would prevent the user from accessing frames, but if a user were to call release frame on an inaccessible frame there would be no way to differentiate between an invalid and valid release.

Upon construction the frame pool adds an this pointer object to a static double linked list that will help us do the release job in the future. In my opinion, I think the most tedious job is to design the algorithm that will consume two bitmaps such as bitmap[5]'s last 3 bits and bitmap[6]'s first several bits. This scenario took my almost 80% of my time, since a lot of edge cases need to be considered. Not only for marking as used but also for releasing is also a tough algorithm. I think this is an interesting homework, if you think deeply into it, you can find a large scales of system based fun point to think of, calculate and implement. I wrote some command on my code, but if grader of TA have any question about my logic feel free to contact me at hunkywei@tamu.edu. Lastly, some of my console out is for debugging, I was running out of time to get rid of

them.


linux [Running]
Bochs x86 emulator, http://bochs.sourceforge.net/
10:53 PM
410
Dash home
Bochs x86 emulator, http://bochs.sourceforge.net/
USER
Copy
Paste
snapshot
CONFIG
Reset
suspend
Power
564
alloc_to_go = 10
568
alloc_to_go = 9
571
alloc_to_go = 8
573
alloc_to_go = 7
574
alloc_to_go = 6
578
alloc_to_go = 5
581
alloc_to_go = 4
583
alloc_to_go = 3
596
alloc_to_go = 2
600
alloc_to_go = 1
603
alloc_to_go = 0
Testing is DONE. We will do nothing forever
Feel free to turn off the machine now.
IPS: 65.694M
A:
NUM
CAPS
SCRL
6. Begin simulation
7. Quit now
Please choose one: [6] 6
00000000000i[ ] installing x module as the Bochs GUI
00000000000i[ ] using log file bochsout.txt
/share/bochs