EE255 Real-Time Embedded Systems

HWK Project #3: Virtual Memory Management

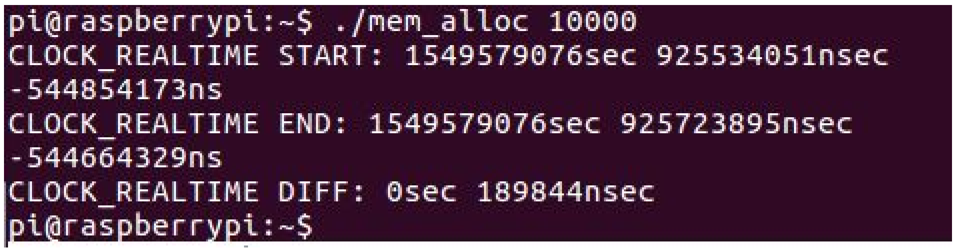
Team05 (Jinwei Zhang & Uriel Escobar)

4.6. Writeup

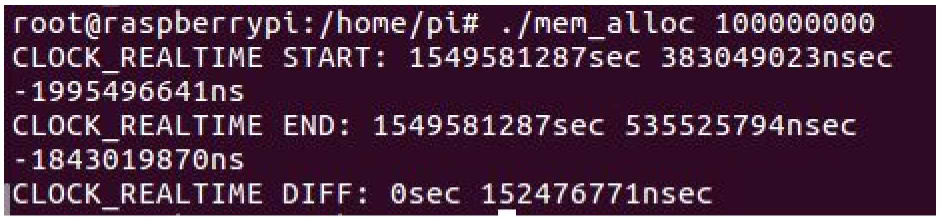
1. What does 'mlockall(MCL_CURRENT | MCL_FUTURE)' do?

By using this call, all the memory of the process in use will be kept in the RAM until it is called unlock. Any new pages that are allocated by the process will also be kept in the RAM. The "mlockall" also protects the process and prevents it leaking information while transferring the message from RAM to peripheral storage.

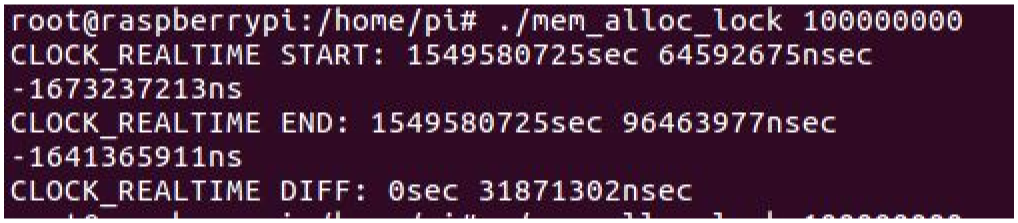
2. Report the memory access time of mem_alloc and mem_alloc_lock (assignment 4.1 and 4.2) on Raspberry Pi 3 with the memory size of 10 KB, 1 MB, and 100MB. Compute average memory access time for each case.



```
pi@raspberrypi:~$ ./mem_alloc 10000
CLOCK_REALTIME START: 1549579076sec 925534051nsec
-544854173ns
CLOCK_REALTIME END: 1549579076sec 925723895nsec
-544664329ns
CLOCK_REALTIME DIFF: 0sec 189844nsec
pi@raspberrypi:~$
```



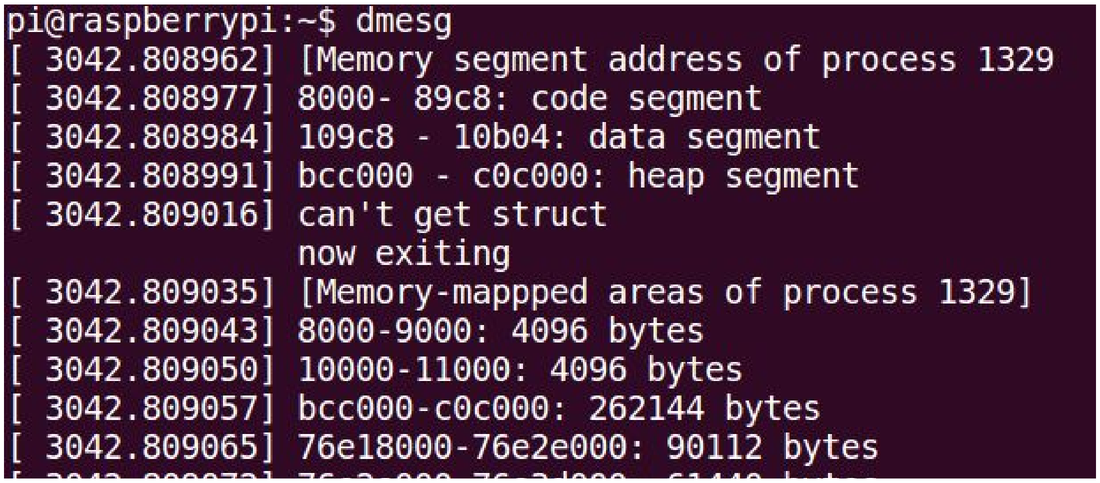
```
root@raspberrypi:/home/pi# ./mem_alloc 100000000
CLOCK_REALTIME START: 1549581287sec 383049023nsec
-1995496641ns
CLOCK_REALTIME END: 1549581287sec 535525794nsec
-1843019870ns
CLOCK_REALTIME DIFF: 0sec 152476771nsec
```


```
root@raspberrypi:/home/pi# ./mem_alloc_lock 100000000
CLOCK_REALTIME START: 1549580725sec 64592675nsec
-1673237213ns
CLOCK_REALTIME END: 1549580725sec 96463977nsec
-1641365911ns
CLOCK_REALTIME DIFF: 0sec 31871302nsec
```

mem_alloc	Exe #1	Exe #2	Exe #3	Avg
10 KB	189844ns	231354ns	190573ns	203924ns
1 MB	2023698ns	1222031ns	1203229ns	1482986ns
100 MB	158262291ns	156317239ns	156349844ns	156976458ns

mem_alloc_lock	Exe #1	Exe #2	Exe #3	Avg
10 KB	296927ns	292813ns	290209ns	293316ns
1 MB	898750ns	992032ns	883229ns	924670ns
100 MB	30817865ns	31871302ns	31023906ns	31237691ns

3. Report the kernel logs of the show_segment_info() syscall (assignment 4.3) for mem_alloc(assignment 4.1) with the memory size of 10 KB and 100MB. Also, report the size of each segment. (*Note*: you will see somewhat counter-intuitive results. Read Section 6. Q&A).

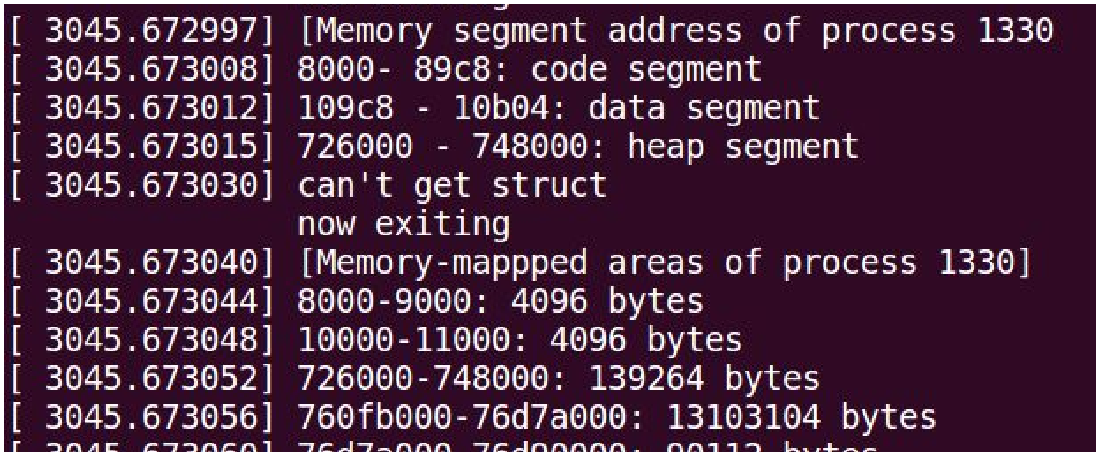


```
pi@raspberrypi:~$ dmesg
[ 3042.808962] [Memory segment address of process 1329
[ 3042.808977] 8000- 89c8: code segment
[ 3042.808984] 109c8 - 10b04: data segment
[ 3042.808991] bcc000 - c0c000: heap segment
[ 3042.809016] can't get struct
               now exiting
[ 3042.809035] [Memory-mappped areas of process 1329]
[ 3042.809043] 8000-9000: 4096 bytes
[ 3042.809050] 10000-11000: 4096 bytes
[ 3042.809057] bcc000-c0c000: 262144 bytes
[ 3042.809065] 76e18000-76e2e000: 90112 bytes
```

The picture above shows the show_segment_info() when we call ./mem_alloc 131000

	code	data	heap
./mem_alloc 131000	4096	4096	262144
./mem_alloc 13100000	4096	4096	139264

The picture above shows the show_segment_info() when we call ./mem_alloc 13100000



```
[ 3045.672997] [Memory segment address of process 1330
[ 3045.673008] 8000- 89c8: code segment
[ 3045.673012] 109c8 - 10b04: data segment
[ 3045.673015] 726000 - 748000: heap segment
[ 3045.673030] can't get struct
               now exiting
[ 3045.673040] [Memory-mappped areas of process 1330]
[ 3045.673044] 8000-9000: 4096 bytes
[ 3045.673048] 10000-11000: 4096 bytes
[ 3045.673052] 726000-748000: 139264 bytes
[ 3045.673056] 760fb000-76d7a000: 13103104 bytes
```

As we can see the heap actually starts to shrinks the more you increase the amount of memory to allocate. This is due to malloc() using brk() when we allocate small amount of memory but when it is big malloc() uses mmap() which stores the heap in another vm area

4. If the system uses virtual memory with demand paging but does not provide mlock-like functions, then what can be a workaround to prevent unpredictable delay in memory access?

Method 1)

Create a child thread that periodically accessing the data so as to keep the data in a "demanding" status.

Method 2)

Check the bitwise valid mark of the page table. Create a child thread, if the page is invalid, then attempt to access the data to page the invalid page into the main memory.

Method 3)

Increasing the levels of paging will reduce the amount of memory each process can take up. So if you have many process running when the level of paging is low you will more likely have more page faults compared to when the level of paging is high, because the each process takes up less space in the

Method 4)

You can copy on write or do page sharing.

5. Give a brief summary of the contributions of each member.

To accommodate the different class schedules, generally, we work on our homework projects individually at the early time after its releasing, then we meet up together to work on the rest and difficult parts. In this way we work both independently and collaboratively so as to maximize practicing and efficiency. We usually meet up 2-3 times for each homework project. Uriel often thinks one step in front of me because he has a better OS background. I (Jinwei) work a little slower but still went through my part of work. We worked on our best together to crack the puzzles and helped each other to understand the hidden knowledge.