REPORT

Aim

To estimate the RSS and Virtual Memory occupied by a process. Also to display the physical addresses of pages which have been physically mapped.

Methodology

Software Walk

1. We initialize page pointer to NULL
2. Given page VA, we can perform a software page table walk doing the following in order,
    - pgd_offset() - returns pgd offset given mm_struct of task and VA
    - pud_offset() - returns pud offset given VA and pgd offset
    - pmd_offset() - returns pmd offset given VA and pud offset
    - pte_offset_map() - returns pte given VA and pmd offset
3. If any of the above returns a none and bad value we exit page pointer as NULL
4. If everything executes succesfully we obtain the struct page using pte_page() and return the same.

Module flow,

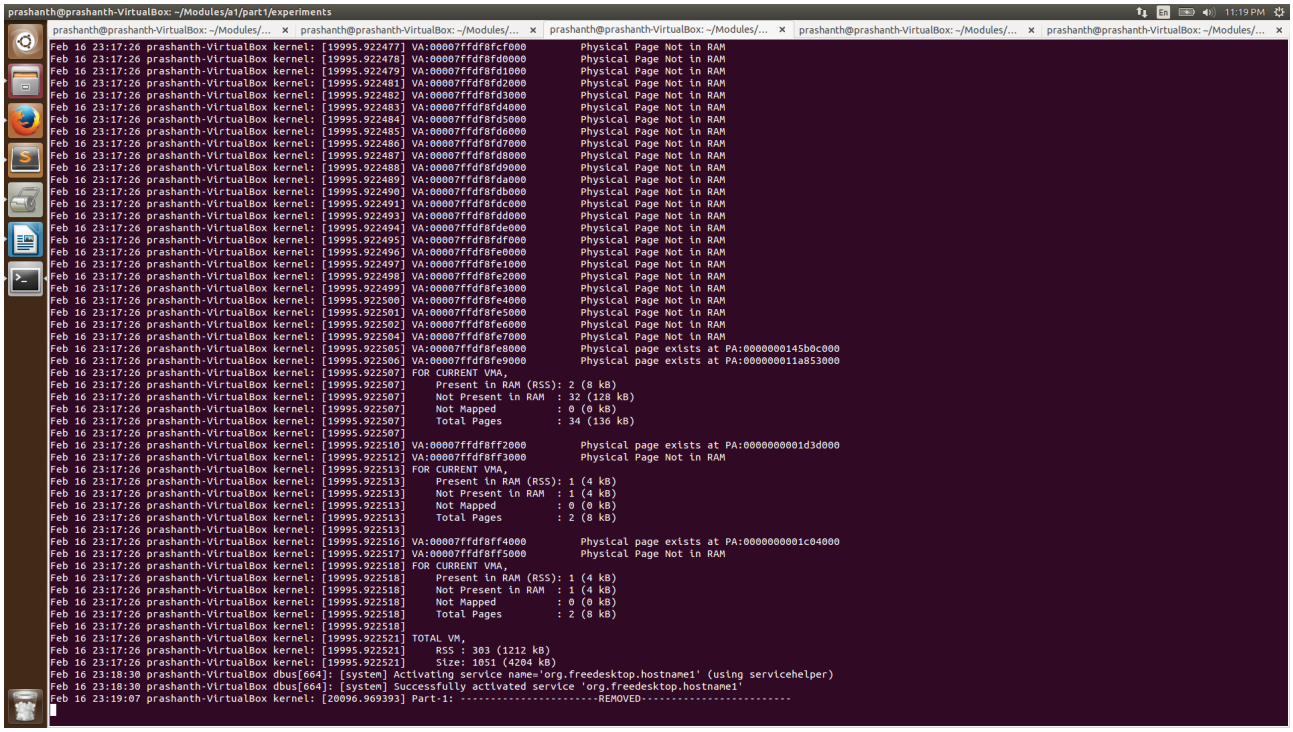
1. Input PID as parameter to the module
2. Extract the task_struct for entered PID using,
    find_vpid() - Returns struct pid for given pid
    pid_task() - Returns task_struct for given struct pid
3. Using the task struct get the first virtual memory area (VMA),
    task_struct -> mm_struct -> vma_struct
4. This struct has the starting and ending virtual addresses (VA) of the current area in,
    vm_start and vm_end
5. Access each page in the area using the start and end by incrementing with PAGE_SIZE
6. Increment page_size for each page reached
7. For each page in VMA perform a software walk of the page table to obtain its page struct pointer

8. If page struct pointer is not NULL and PA returned is not zero, if so we increment RSS and display the PA
9. Display the RSS and size for current VMA
10. Move to the next VMA using vma_next in the current VMA
11. Repeat steps 4-7 until vma_next hits NULL which indicates end of Virtual memory assigned for the process.
12. Display the total RSS and size

Experiments

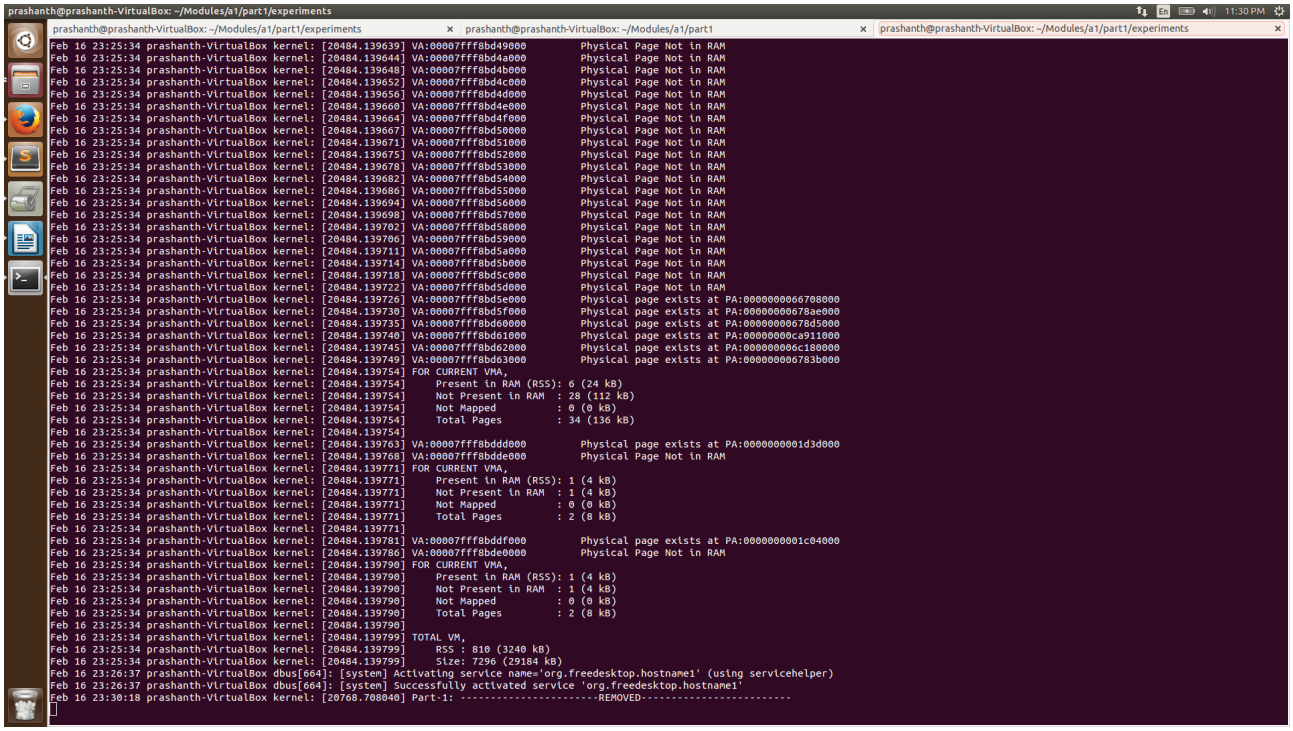
1- Hello World C program

1. A non-terminating hello world program was loaded
2. The VMA tracker module was loaded with hello world's PID
3. Output of log was compared the output of
   /proc/<pid>/status
   /proc/<pid>/smaps
4. It was observed that VM size was equal in all three cases
5. RSS in /proc/<pid>/status didn't match and this was caused due to sharing of pages
6. But module output nearly matched with smaps difference of 2 pages (8kB) - 1212kB and 1204kB



2- Top

1. A top command was invoked
2. The VMA tracker module was loaded with top's PID
3. Output of log was compared the output of
   /proc/<pid>/status
   /proc/<pid>/smaps
4. RSS in /proc/<pid>/status nearly matched with 1 page difference (4kB)
5. The module output also nearly matched with smaps difference of 1 page (4kB)



Observations

1. It was observed that size of VMA output by the module always matched /status.
2. The RSS of module and /status didn't match for some processes - the reason was sharing of pages between processes is accounted differently in /status.
3. RSS nearly matched with /smaps with a minimal difference of 1-3 pages for small processes.
4. Small processes didn't have many unmapped pages, but this number increased as the size of the process increased.
5. Unmapped pages were observed in larger processes like firefox