Design Assignment 3

Group 39

Aayush Ojha- 13009
Amit Kumar - 13094
Shubham Jain- 13683

Process	Number of page faults	Total Number of Ticks
vmtest1	376	1778697
vmtest2	377	834572
queue	118	455249

For calculating the number of page faults, we are increasing the page faults in the following places :

1. When a page fault exception is raised
2. When during the fork the valid but not shared pages are copied into child
3. When we extend the page table to incorporate the additional shared pages during shared memory allocate.

Explanations and Observations:

- **Increasing the OUTER BOUND in vmtest1 will not increase page faults**

All the pages are loaded in the memory after the first iteration of OUTER BOUND and since we are not restricting the number of page frames therefore all the pages once loaded would remain in the memory and therefore further iterations would not affect the result. Also increasing the outer bound increases the total ticks and it would run more iterations of loop.

- **Page faults for vmtest2 is greater than page faults of vmtest1**

On doing experiment we found that

For vmtest1

total numPages	384
number of pages for code	⌈1952/128⌉ = 16
initialised data	16 bytes
number of pages for uninitialised data segment	46080/128 = 360
number of pages for user stack	1024/128 =8

For vmtest2

total numPages	385
number of pages for code	⌈2144/128⌉ = 17

initialised data	16 bytes
number of pages for uninitialised data segment	46080/128 = 360
number of pages for user stack	1024/128 =8

Since vmtest1 and vmtest2 does not use recursive function or stack number of pages for stack will never be used and thus no page faults would be encountered for those. Hence the total number of page faults in vmtest1 is 376 and total number of page faults for vmtest2 is 377.

Queue
In the queue.c we have a large number of pages for code which increases the pagefaults and the shared memory will be counted only once.