Report regarding printing page information and implementing demand paging.

PART 1

To write a system call to print the info regarding the pages

In the syscall.h header file we defined a macro for SYS_pgtPrint as 22.

In the syscall.c file we added the sys_pgtPrint() system call in the system call list so as to enable the calling.

In the sysproc.c file we defined the sys_pgtPrint() system call(what the system call should do using prdefined macros.

In usys.S file we added the system call label "pgtPrint" which would assign the value SYS_pgtPrint to the %eax register and also generate an interuppt.

We created a mypgtPrint.c file to access the system call mypgtPrint.

Finally in the makefile we added "_mypgtPrint/" under the UPROGS which contains the list of programs to be built.

We ran the code for different values of global array and local arrays. The observation is :

For a global array of size 10000 we get 12 page entries from 0 to 12 in which entry number 11 is inaccessible.

For a local array of size 10000 we get 2 page entries from 0 to 2 in which entry number 1 is inaccessible.

After running the program multiple times we observed that the virtual address and the physical address remain same after every time we make the program but running it multiple times after making it gives different values for both the addresses.

PART 2

To implement the demand paging in xv6

In exec.c file we changed the last parameter of the allocuvm() function in the load segment from "ph.vaddr + ph.memsz" to "ph.vaddr + ph.filesz" to accomodate the file size. Also we added the else part to this if condition setting size to "ph.vaddr + ph.memsz".

In trap.c file we added another case to the switch case statement to trap page fault and there we mapped the respective pages to their physical addresses.

We created a mydemandPage.c file to implement the demand paging.

To execute the c program in makefile we added "_mydemandPage/" under the UPROGS which contains the list of programs to be built and in the EXTRAS section we added the "mydemandPage.c" to compile the code.

We ran the code for different sizes of global array. The observation is :

For a global array of size 300 we get 2 page entries from 0 to 2 in which entry number 1 is inaccessible.

For a global array of size 3000 we get 12 page entries from 0 to 5 in sets of 3 for each 1000 size of array in which entry number 2, 3, 4 is inaccessible in part 1, enrty number 3, 4 is inaccessible in part 2, and enrty number 4 is inaccessible in part 3.

For a global array of size 5000 we get 24 page entries from 0 to 7 in sets of 5 for each 1000 size of array in which entry number 2, 3, 4, 5, 6 is inaccessible in part 1, enrty number 3, 4, 5, 6 is inaccessible in part 2, enrty number 4, 5, 6 is inaccessible in part 3, enrty number 5, 6 is inaccessible in part 4, and enrty number 6 is inaccessible in part 5.