Operating Systems - 2 : CS3523
Programming Assignment - 6 : Paging
Vanga Aravind Shounik - CS20BTECH11055

Part - 1:

Here, we are asked to implement a system call named pgtprint
To implement a system call, we have to change the following files.
syscall.h, here, we add a new syscall

```
#define SYS_pgtprint 22
```

syscall.c, here, we have to add the following lines of code

```
extern int sys_pgtprint(void);
[SYS_pgtprint] sys_pgtprint,
```

In their respective places.
Now we add the function in the sysproc.c file.

```
int
sys_pgtprint(void);
```

Now, we have to add the system call in usys.S file

```
SYSCALL(pgtprint)
```

We have to add the following line in user.h

```
int pgtprint(void);
```

Now, we have to define a user program, for that, we have to create a file named pgtprint.c and we have to write the code in it for implementing the system call.
For the user program to execute, we have to make the changes in Makefile. In Makefile, we have to add the name of the file in 2 places, UPROGS and EXTRAS.
Then, we are good to go.
Now, In the function we write in sysproc.c,
Here, we can see that curproc()->pgdir contains the page data and now we store the data in pgdir and then store the entries of the pgdir array in pgdir_entry. If the entry is valid, we check the table and store the table entries in pgtable, pgtable_entry. Now, if the pgtable_entry is valid, then it prints the data of the page.

Here, if we declare a global array, then the whole array is allocated in the main memory, so, the number of pages are more in this case whereas in the case where the array is defined locally, then we can see that extra pages are not allocated in the main memory.

Here, we can see that the virtual and physical addresses change during multiple executions.

Here, to run the xv6 Operating System, we have to use the commands

```
make
make qemu-nox
```

We can see that this gives a shell.
Now, to run the user defined program, we have to use the command

```
pgtprint
```

In the xv6 shell.

Part - 2: Implementation of Demand Paging

Here, in exec.c, the process is allocated a memory of ph.memsize but in this case, here, we are trying to stop the allocation of memory for the dynamic variables and here, we can see that the memory is of the form ph.filesize and then the remaining memory is for the dynamic variables memory. So, in this case, we only allocate the ph.filesize memory to the process.
Now, when the dynamic variables are accessed, the OS gives a page fault. So, we change the trap.c file where the error is caused and if the error is T_PGFLT, then we have to give the process the extra memory it needs. For that, initially, we take the virtual address of the process and we round the address to the nearest multiple of 1024 because each page has 1024B memory in xv6. Here, the rounding off is done by the function PGROUNDDOWN.
Now, if the virtual address is less than the allocated size, then we can say that it needs more memory for the process. So, we allocate the memory to a variable mem using kalloc() and now, we use mappages to map the virtual address to the physical address in pgdir of the process.

Now, we implemented a user program named mydemandPage.c and we can see that the paging is not given to the variables at the start but they are given when they are used in the process. To run the program in terminal use the commands

```
make
make qemu-nox
mydemandPage
```

Here, we can see that the mydemandPage command should be typed in the xv6 shell.