Operating Systems
CS-450
Assignment #2

Sivasenthil Namachivayan
A20391478
March 3,2017

Discussion on the implementation:

Changes done in the following files

sysproc.c

1. Declared a variable syscallnum[23] as int datatype array
2. The count of number of system calls invoked is done by the int systemcall sys_syscallnum(void) where the index of all the system call is stored in the array and it gets incremented when the system call is invoked either by system or by a user program
   Example:
   ```
   cprintf("\n Fork calls-> %d",syscallnum[1]);
   cprintf("\n Exit calls  -> %d",syscallnum[2]);
   ```
3. After the listing of all the system calls present in the xv6, the counter has to be set to zero as per the requirment and so the following syntax is called to make the count in the array to "Zero" and the bitwise representation is known memset is used as below,
   ```
   memset(syscallnum, 0, sizeof(syscallnum));
   ```

syscall.c

1. Declared the user defined
   variable syscallnum[] as extern int array
   function sys_syscallnum(void) as extern
   system call [SYS_syscallnum] sys_syscallnum
2. In the syscall function which is predefined in the xv6 incremented the array syscallnum[0]
3. and if the system declared variable num==each systemcall then the array will be incremented to count the number of times that particular system call is invoked.
   Example:
   ```
   if(num==SYS_fork){
   syscallnum[num]++;
   }
   else if ()// user code in the similar way mentioned above
   ```
4. In the above way 21 predefined system calls along with 1 user defined system call SYS_syscallnum is coded.

assignment2.c – User Program

In this program, the user defined system call SYS_syscallnum is invoked to get the count of all the systemcall invoked.

syscall.h

In this file defined the user system call as SYS_syscallnum with the index value of 22

user.h

In this file declared the system call as int syscallnum(void);

make

In this file mentioned the user program assignment2.c under UPROGS=\ as _assignment2 for the user program to get executed by the xv6

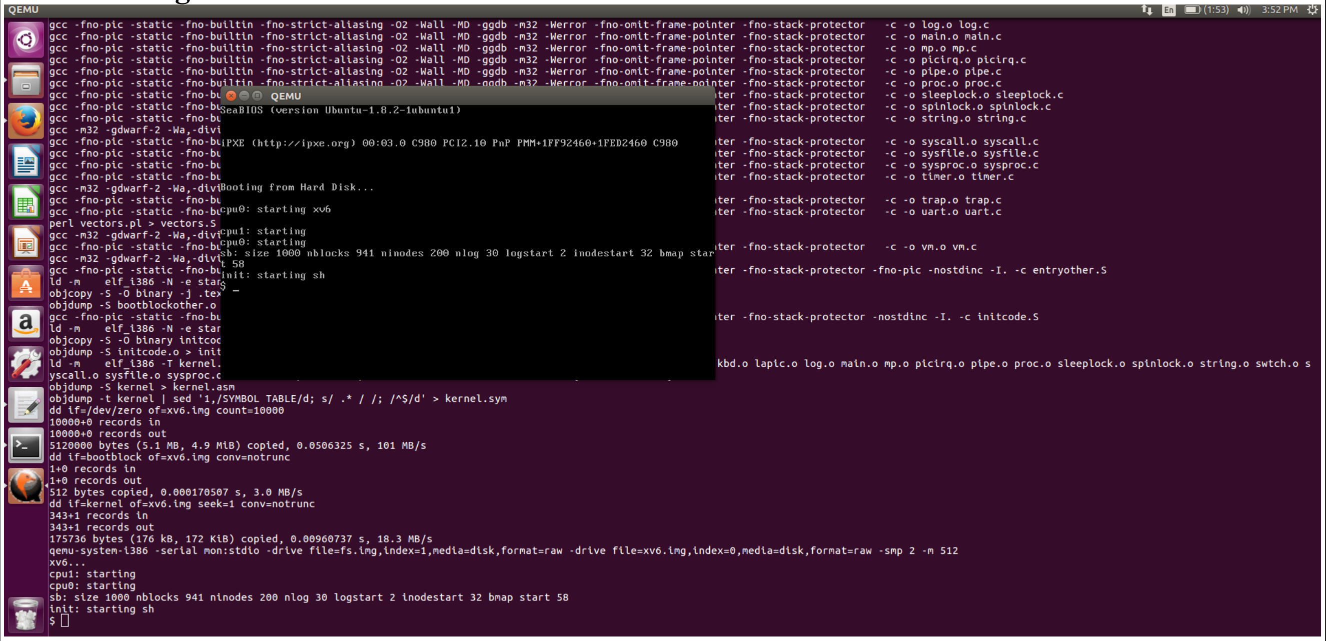
Output:

With all the changes mentioned above, the user defined system call is invoked by the user program and the output is printed in the xv6 kernel.

Details to be mentioned in the output:

1. It lists all the systems calls present in the xv6 along with the one the user done and the number of count it is been invoked till now.
2. The counter of the system calls are again set to zero so that when the program is executed again it will display a consistent number of the system calls count for better understanding.
3. The count of the system call set to zero after the completion of the program is also mentioned in the output.

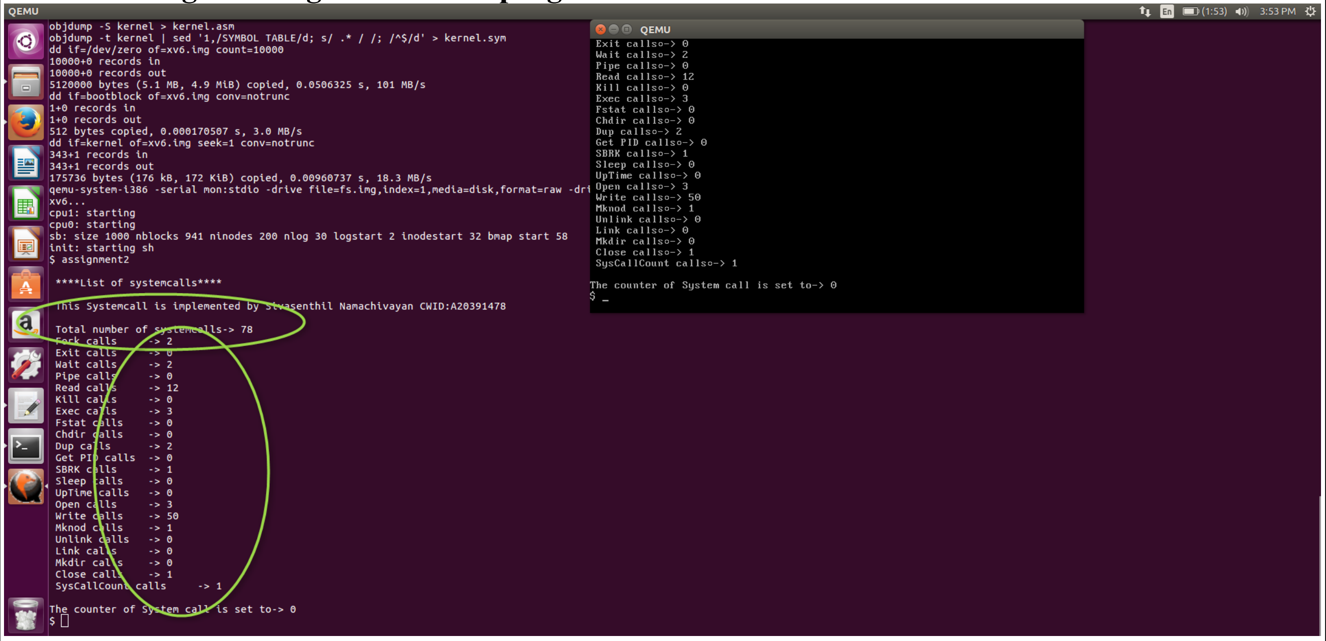
1. On booting xv6



```
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector   -c -o log.o log.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector   -c -o main.o main.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector   -c -o mp.o mp.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector   -c -o picirq.o picirq.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector   -c -o pipe.o pipe.c
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector   -c -o proc.o proc.c
gcc -fno-pic -static -fno-bu                                                                            ter -fno-stack-protector   -c -o sleeplock.o sleeplock.c
gcc -fno-pic -static -fno-bu                                                                            ter -fno-stack-protector   -c -o spinlock.o spinlock.c
gcc -fno-pic -static -fno-bu                                                                            ter -fno-stack-protector   -c -o string.o string.c
gcc -m32 -gdwarf-2 -Wa,-divi
gcc -fno-pic -static -fno-bu                                                                            ter -fno-stack-protector   -c -o syscall.o syscall.c
gcc -fno-pic -static -fno-bu                                                                            ter -fno-stack-protector   -c -o sysfile.o sysfile.c
gcc -fno-pic -static -fno-bu                                                                            ter -fno-stack-protector   -c -o sysproc.o sysproc.c
gcc -fno-pic -static -fno-bu                                                                            ter -fno-stack-protector   -c -o timer.o timer.c
gcc -m32 -gdwarf-2 -Wa,-divi
gcc -fno-pic -static -fno-bu                                                                            ter -fno-stack-protector   -c -o trap.o trap.c
gcc -fno-pic -static -fno-bu                                                                            ter -fno-stack-protector   -c -o uart.o uart.c
perl vectors.pl > vectors.S
gcc -m32 -gdwarf-2 -Wa,-divi
gcc -fno-pic -static -fno-bu                                                                            ter -fno-stack-protector   -c -o vm.o vm.c
gcc -m32 -gdwarf-2 -Wa,-divi
gcc -fno-pic -static -fno-bu                                                                            ter -fno-stack-protector -fno-pic -nostdinc -I. -c entryother.S
ld -m    elf_i386 -N -e star
objcopy -S -O binary -j .tex
objdump -S bootblockother.o
gcc -fno-pic -static -fno-bu                                                                            ter -fno-stack-protector -nostdinc -I. -c initcode.S
ld -m    elf_i386 -N -e star
objcopy -S -O binary initcod
objdump -S initcode.o > init
ld -m    elf_i386 -T kernel.                                                                            kbd.o lapic.o log.o main.o mp.o picirq.o pipe.o proc.o sleeplock.o spinlock.o string.o swtch.o s
yscall.o sysfile.o sysproc.c
objdump -S kernel > kernel.asm
objdump -t kernel | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > kernel.sym
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.0506325 s, 101 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.000170507 s, 3.0 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
343+1 records in
343+1 records out
175736 bytes (176 kB, 172 KiB) copied, 0.00960737 s, 18.3 MB/s
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -drive file=xv6.img,index=0,media=disk,format=raw -smp 2 -m 512
xv6...
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ 
```

```
SeaBIOS (version Ubuntu-1.8.2-1ubuntu1)


iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+1FF92460+1FED2460 C980



Booting from Hard Disk...

cpu0: starting xv6

cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap star
t 58
init: starting sh
$ _
```

2. On running the assignment2 user program on xv6

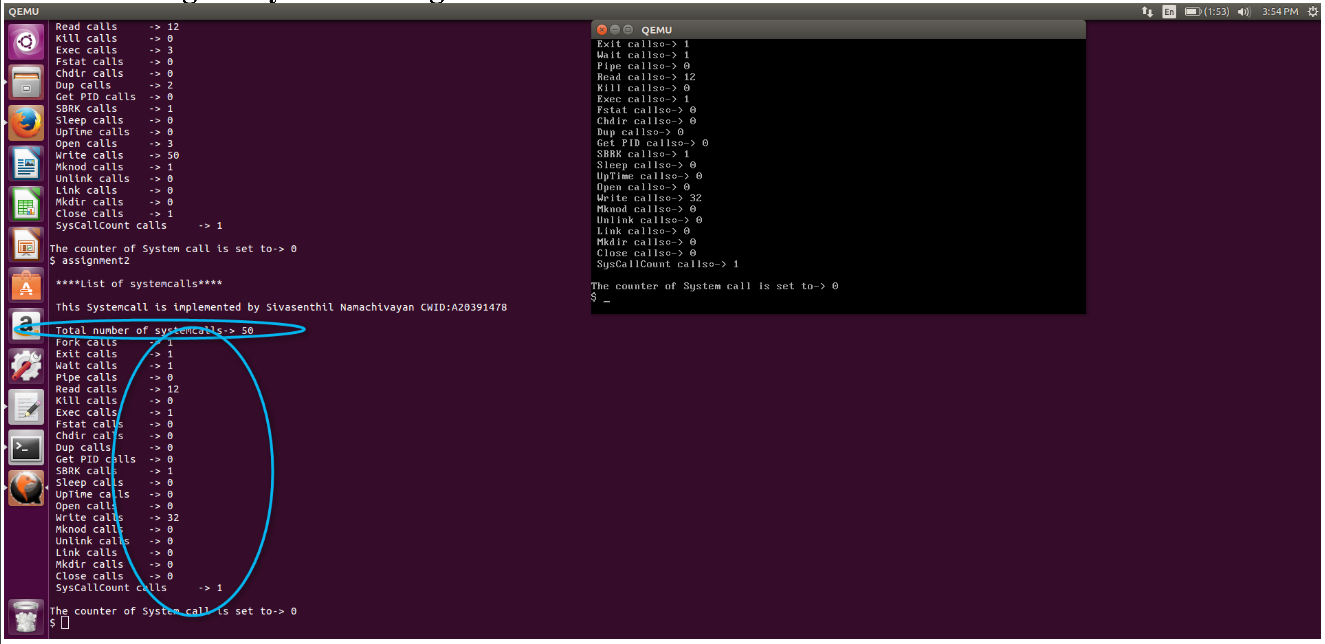


```
objdump -S kernel > kernel.asm
objdump -t kernel | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > kernel.sym
dd if=/dev/zero of=xv6.img count=10000
10000+0 records in
10000+0 records out
5120000 bytes (5.1 MB, 4.9 MiB) copied, 0.0506325 s, 101 MB/s
dd if=bootblock of=xv6.img conv=notrunc
1+0 records in
1+0 records out
512 bytes copied, 0.000170507 s, 3.0 MB/s
dd if=kernel of=xv6.img seek=1 conv=notrunc
343+1 records in
343+1 records out
175736 bytes (176 kB, 172 KiB) copied, 0.00960737 s, 18.3 MB/s
qemu-system-i386 -serial mon:stdio -drive file=fs.img,index=1,media=disk,format=raw -dri
xv6...
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ assignment2

 ****List of systemcalls****

 This Systemcall is implemented by Sivasenthil Namachivayan CWID:A20391478

 Total number of systemcalls-> 78
 Fork calls     -> 2
 Exit calls     -> 0
 Wait calls     -> 2
 Pipe calls     -> 0
 Read calls     -> 12
 Kill calls     -> 0
 Exec calls     -> 3
 Fstat calls    -> 0
 Chdir calls    -> 0
 Dup calls      -> 2
 Get PID calls  -> 0
 SBRK calls     -> 1
 Sleep calls    -> 0
 UpTime calls   -> 0
 Open calls     -> 3
 Write calls    -> 50
 Mknod calls    -> 1
 Unlink calls   -> 0
 Link calls     -> 0
 Mkdir calls    -> 0
 Close calls    -> 1
 SysCallCount calls     -> 1

The counter of System call is set to-> 0
$ 
```

```
 Exit callso-> 0
 Wait callso-> 2
 Pipe callso-> 0
 Read callso-> 12
 Kill callso-> 0
 Exec callso-> 3
 Fstat callso-> 0
 Chdir callso-> 0
 Dup callso-> 2
 Get PID callso-> 0
 SBRK callso-> 1
 Sleep callso-> 0
 UpTime callso-> 0
 Open callso-> 3
 Write callso-> 50
 Mknod callso-> 1
 Unlink callso-> 0
 Link callso-> 0
 Mkdir callso-> 0
 Close callso-> 1
 SysCallCount callso-> 1

The counter of System call is set to-> 0
$ _
```

For the fist time the system call count is 78 which include the boot system calls invoked by kernel

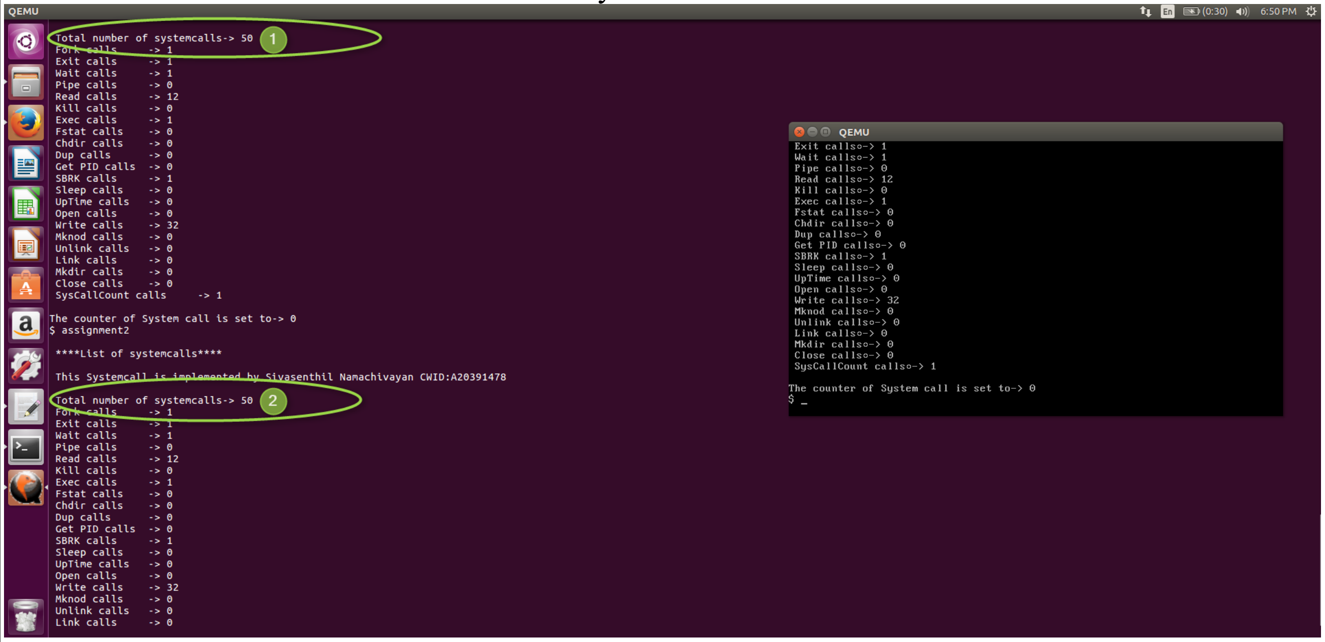
3. On running the system call again after the counter is set to "Zero"



```
 Read calls      -> 12
 Kill calls      -> 0
 Exec calls      -> 3
 Fstat calls     -> 0
 Chdir calls     -> 0
 Dup calls       -> 2
 Get PID calls   -> 0
 SBRK calls      -> 1
 Sleep calls     -> 0
 UpTime calls    -> 0
 Open calls      -> 3
 Write calls     -> 50
 Mknod calls     -> 1
 Unlink calls    -> 0
 Link calls      -> 0
 Mkdir calls     -> 0
 Close calls     -> 1
 SysCallCount calls     -> 1

The counter of System call is set to-> 0
$ assignment2

 ****List of systemcalls****

 This Systemcall is implemented by Sivasenthil Namachivayan CWID:A20391478

 Total number of systemcalls-> 50
 Fork calls      -> 1
 Exit calls      -> 1
 Wait calls      -> 1
 Pipe calls      -> 0
 Read calls      -> 12
 Kill calls      -> 0
 Exec calls      -> 1
 Fstat calls     -> 0
 Chdir calls     -> 0
 Dup calls       -> 0
 Get PID calls   -> 0
 SBRK calls      -> 1
 Sleep calls     -> 0
 UpTime calls    -> 0
 Open calls      -> 0
 Write calls     -> 32
 Mknod calls     -> 0
 Unlink calls    -> 0
 Link calls      -> 0
 Mkdir calls     -> 0
 Close calls     -> 0
 SysCallCount calls     -> 1

The counter of System call is set to-> 0
$
```

```
Exit callso-> 1
Wait callso-> 1
Pipe callso-> 0
Read callso-> 12
Kill callso-> 0
Exec callso-> 1
Fstat callso-> 0
Chdir callso-> 0
Dup callso-> 0
Get PID callso-> 0
SBRK callso-> 1
Sleep callso-> 0
UpTime callso-> 0
Open callso-> 0
Write callso-> 32
Mknod callso-> 0
Unlink callso-> 0
Link callso-> 0
Mkdir callso-> 0
Close callso-> 0
SysCallCount callso-> 1

The counter of System call is set to-> 0
$ _
```

In the above screenshot the value of the total system calls is 50 and the count is set to "zero"



```
 Total number of systemcalls-> 50
 Fork calls      -> 1
 Exit calls      -> 1
 Wait calls      -> 1
 Pipe calls      -> 0
 Read calls      -> 12
 Kill calls      -> 0
 Exec calls      -> 1
 Fstat calls     -> 0
 Chdir calls     -> 0
 Dup calls       -> 0
 Get PID calls   -> 0
 SBRK calls      -> 1
 Sleep calls     -> 0
 UpTime calls    -> 0
 Open calls      -> 0
 Write calls     -> 32
 Mknod calls     -> 0
 Unlink calls    -> 0
 Link calls      -> 0
 Mkdir calls     -> 0
 Close calls     -> 0
 SysCallCount calls     -> 1

The counter of System call is set to-> 0
$ assignment2

 ****List of systemcalls****

 This Systemcall is implemented by Sivasenthil Namachivayan CWID:A20391478

 Total number of systemcalls-> 50
 Fork calls      -> 1
 Exit calls      -> 1
 Wait calls      -> 1
 Pipe calls      -> 0
 Read calls      -> 12
 Kill calls      -> 0
 Exec calls      -> 1
 Fstat calls     -> 0
 Chdir calls     -> 0
 Dup calls       -> 0
 Get PID calls   -> 0
 SBRK calls      -> 1
 Sleep calls     -> 0
 UpTime calls    -> 0
 Open calls      -> 0
 Write calls     -> 32
 Mknod calls     -> 0
 Unlink calls    -> 0
 Link calls      -> 0
```

```
Exit callso-> 1
Wait callso-> 1
Pipe callso-> 0
Read callso-> 12
Kill callso-> 0
Exec callso-> 1
Fstat callso-> 0
Chdir callso-> 0
Dup callso-> 0
Get PID callso-> 0
SBRK callso-> 1
Sleep callso-> 0
UpTime callso-> 0
Open callso-> 0
Write callso-> 32
Mknod callso-> 0
Unlink callso-> 0
Link callso-> 0
Mkdir callso-> 0
Close callso-> 0
SysCallCount callso-> 1

The counter of System call is set to-> 0
$ _
```

When it is executed again the same number of count is printed as mentioned in the screenshot above.

README:

A new xv6 with modified kernel to support the changes done for assignment2-
introduce a new system call that returns the number of total system calls invoked.

To run the program:
1.Copy the rar file attached to the desired location in ubuntu OS
2.Extract the rar file
3.Navigate to the xv6 folder location in the ubuntu terminal using cd command
4.Type make clean in ubuntu terminal to clean the object files
5.Type make to compile the codes in ubuntu terminal.
6.Type make qemu to boot xv6 operating system in ununtu terminal.

Once the boot is done, you can mention the name of my C program which invoke the systemcall to print the output

7.Type assignment2 to run the user program
8.The output will show all the systemcall invoked and its count
9.Type again as assignment2 to run the same program again
10.You will notice that the total number of system call are updated
11.Type again as assignment2 to run the same program again
12.You will notice that the total number of system call is the same as in step 10.