# PROJECT REPORT ON XV6

Submitted By:

Tulika Sureka

01FB15ECS327

Under the guidance of:

Prof. Nitin V Pujari

Department of Computer Science and Engineering

PES University

Bangalore-560085

## Problem Statement

Implement the system call fgproc(), and modify sh.c to use it to track the foreground process. Whenever CTRL + C is pressed, send SIGINT to the foreground process.

## Steps to be followed

### Step 1-

### Adding a user program to Xv6: -

First, we create a C user program called "inf" which runs an infinite loop to test the interrupt simulation. As soon as CTRL+C is pressed, an interrupt should occur to terminate the program's process which is done by implementing the system call fgproc().

Create a new file by typing **"gedit inf.c"** and paste the following code in it.

```
#include "types.h"
#include "stat.h"
#include "user.h"

int
main(void)
{
    while(1)
   {
       printf(1, "Hello");
   }
}
```

Now, we must edit the **Makefile** of the xv6 source code in order to let the compiler know that we have a program like this and we need it to be compiled with other system programs. In the **Makefile**, there are two places in which we need to put entries.

- Add the line below at the end of UPROGS,

  _inf\

- Add the line below after zombie.c\ in EXTRA,

  _inf.c\

## <u>Step 2-</u>

## <u>Adding a System call to xv6</u>

Now we add a system call fgproc() to handle the SIGINT interrupt.

Navigate to the xv6 folder by typing **cd xv6**

1)First we edit **syscall.h** to define the position of the system call vector that connects to our implementation.

Go to the ubuntu Terminal and type gedit syscall.h to edit the file. Go to the end of the file and add a statement as follows:

**#define SYS_fgproc 23**

Note that the 23 here might change depending on the number given before the line we are going to add in the file. That is if the system call number is 22 in the line before the line that we are going to add, our line should have the number 23.

2) Next Go to the ubuntu Terminal and type **gedit syscall.c** to edit the file. Go to the end of lines containing function declarations and add our function declaration. Now add the following lines to the **syscall.c** file.

**extern int sys_fgproc(void);**

and in the array of syscalls, append the following line in.

**[SYS_fgproc] sys_fgproc,**

3)Next we edit the **sysproc.c** , that is where the implementation of our system call goes if it is a system call related to process handling or memory management, Type **gedit sysproc.c** on the ubuntu terminal and put the implementation of our system call as follows

```
int

sys_fgproc(void)

{

  return fgproc();

}
```

4) Next we edit **proc.c** to define our Interrupt handler **fgproc()** . Our core program is defined here.

In this function, we access the process table ptable for pid access of every running process. Meanwhile we lock the process table to prevent any changes during interrupt handling. We iterate over the elements in the process table ptable and once we find our running process, we kill that process (which is the functionality for SIGINT).

Type gedit **proc.c** on the ubuntu terminal to edit the file. At the end of the file, add the function definition as follows:

```
int fgproc(void){
struct proc *p;
for(p = ptable.proc;p<&ptable.proc[NPROC];p++)
{
   if(p!=initproc && p->pid!=2)
  {
        cprintf("\nCtrl + C detected\n");
        p->killed=1;
        kill(p->pid);
        break;
   }
}
  return 23;
}
```

5) Next we edit **defs.h** to declare the function written in **proc.c**.

Type **gedit defs.h** on the ubuntu terminal to edit the file. The declaration is done under the proc.c section as follows :

**int fgproc(void);**

6)Next we edit the file **usys.S**

Type **gedit usys.S** on the ubuntu terminal and add the following statement at the end of the file.

**SYSCALL(fgproc)**

7) Lastly we edit **user.h** file. This is how our user program sees our system call.

Type **gedit user.h** on the ubuntu terminal and add the following statement at the end of the file.

```
int fgproc(void);
```

## <u>Step 3</u>

## <u>System call and Interrupt linking</u>

1)Now we  link the interrupt occurred(Ctrl+C) and interrupt handler (fgproc()) for SIGINT to work.

For this, we work with **console.c** to manage console interrupts. consoleintr() is the function in console.c which handles all console interrupts. We add a new “case” i.e., case C('C') (This indicates that whatever is inside case C('C') will be executed when Ctrl+C is pressed).

Type **gedit console.c** to edit the file.

In the **consoleintr()** method, find the switch control structure. To it, add the following as a case :

```
case C('C'):
   dofgproc=1;
 break;
```

In the same function, after the switch control structure and after the release of process lock (release(&cons.lock)), add the following code :

```
if(dofgproc){
fgproc();
}
```

**<u>Step 4</u>**

**<u>Running the Qemu terminal</u>**

1)In the xv6 directory , first clean the make file by typing

**make clean**

2)Make the changes make to all the files by typing

**make**

Check that no errors occur during the make process.

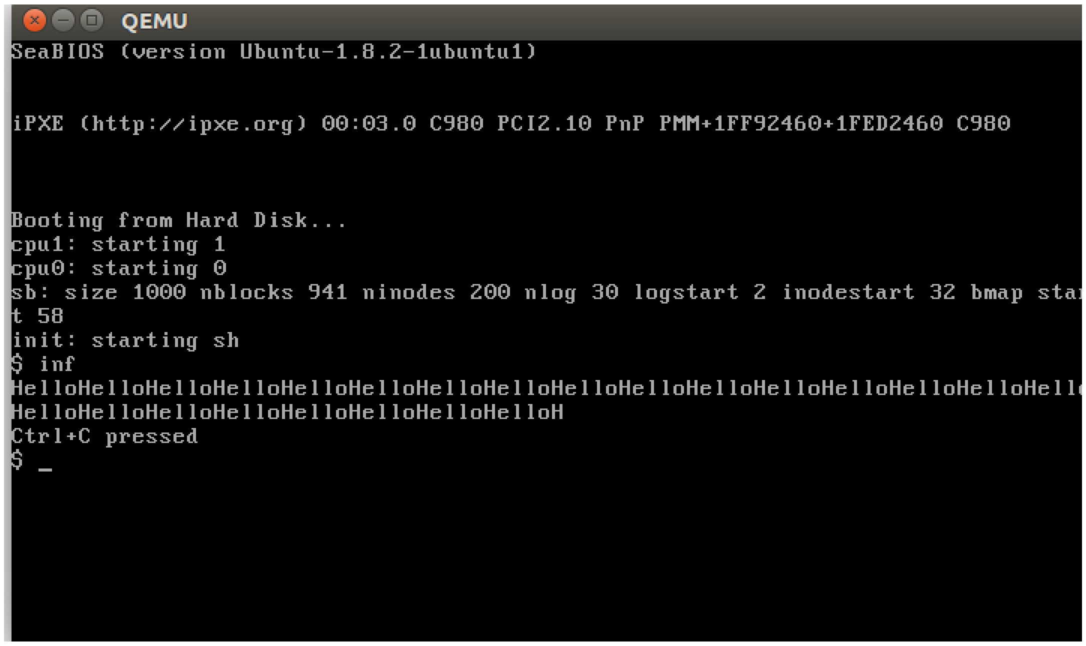
3)Once make is successfully done , Type

**make qemu**

to open the qemu terminal.

4)In the qemu terminal once the $ prompt appears type **inf** (name of our user program to test the interrupt). Once **Ctrl+c** is pressed , the process gets killed.

Output-



```
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...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap sta
t 58
init: starting sh
$ inf
HelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHelloHell
HelloHelloHelloHelloHelloHelloHelloHelloH
Ctrl+C pressed
$ _
```

References:-

http://recolog.blogspot.in/2016/03/adding-user-program-to-xv6.html

https://stackoverflow.com/questions/8021774/how-do-i-add-a-system-call-utility-in-xv6

https://pdos.csail.mit.edu/6.828/2016/xv6/book-rev9.pdf

Pages-9,39,45