CS344 Assignment-1

Sai Ashrritth P, 190101061

Question-1:

For this exercise, the following files were edited :

- **ASCII_image.h:**
  Created a header named *ASCII_image.h* and defined some ascii image strings as macros.
- **sysproc.c:**
  Implemented a function named *sys_draw(void).* If the buffer is too small, it returns -1. If the call succeeds it returns a number bytes copied to the buffer.

```
95 int sys_draw(void)
96 {
97
98   char *buffer;
99   int size;
100
101   // Feches the 1st 32 bit int argument which is the max buffer sizeand assigns it to the size
102   if (argint(1, &size) == -1)
103   {
104     // Invalid address is accessed
105     return -1;
106   }
107
108   // Check that the buffer pointer in first argument
109   // lies within the process address space or not till size bytes, if it does not then return -1.
110   if (argptr(0, (char **)&buffer, size) == -1)
111   {
112     // does not lie in the process address space.
113     return -1;
114   }
115
116   // copying macro wolfi from ASCII_image.h
117   char *draw = wolfi;
118
119   int drawsize = 0;
120   while (draw[drawsize] != '\0')
121   {
122     drawsize++;
123   }
124
125   if (drawsize > size)
126   {
127     //buffer size is insufficient to draw the wolf picture.
128     return -1;
129   }
130
131   //copying the wolf picture into the buffer.
132   for (int i = 0; i < drawsize; i++)
133   {
134     buffer[i] = draw[i];
135   }
136
137   //return the size of draw pictue
138   return drawsize;
139 }
```

  It takes an ascii image string (*wolfi* in line 116) from ASCII_image.h and copies it into the buffer
- **syscall.h:**
  Defined the position of the system call vector as SYS_draw which connected to our implementation.

```
22 #define SYS_close  21
23 // A macro for SYS_draw
24 #define SYS_draw 22
```

- **syscall.c:**
  Added another line *extern int sys_draw(void).* This external function is visible to the whole program, it connects the shell and kernel, and the system call function was added to the system call vector at a position defined in syscall.h.

```
104 extern int sys_write(void);
105 extern int sys_uptime(void);
106 extern int sys_draw(void);
```

```
130 // adding system call vector
131 [SYS_draw] sys_draw,
132 };
```

- **usys.S:**
  Created a user level call definition for the system call sys_draw. Used this to connect the users call to system call to system call function.

```
31 SYSCALL(uptime)
32 SYSCALL(draw)
```

- **user.h:**
  Included the system call *int draw(void *buf, uint size);* in the user header file.

```
26 // system call created which copies the ASCII image of wolf picture
27 int draw(void *buf, uint size);
```

After this the **Drawtest.c** file was created, it is a c code file which included user, types and stat header files and takes the size of the buffer from the user and prints the ascii image if the buffer size is greater than the size of the image or prints an error msg.

k is a void buffer which is used to take input from the user (*line 14).* After getting the input I converted it to an integer value and stored it in *size*. Then I created a buffer named *draw_buffer* with buffer size equal to *size*. Then I called the *draw()* function from user.h in line 30 which returns the bytes copied into the buffer from the kernel.
Now the *draw(void* , uint)* will return -1 if the buffer size is insufficient, then we will print line 36 and if the buffer size is sufficient then we will print the ascii image string which was copied into the *draw_buffer.*

```
11    void* k = malloc(100); int n;
12    // Taking buffer size from the user.
13    printf(1, "ENTER BUFFER SIZE : ");
14    n = read(0, k, 100);
15    if(n < 0) {
16        printf(2, "read error\n");
17        exit();
18    }
19    char* size_str= (char*)k;
20
21    uint size=0, i=0;
22    while(i<n-1){
23        size = 10*size + (int) size_str[i]-48;
24        i++;
25    }
26    // Created a buffer with maximum size as size
27    void *draw_buffer = malloc(size);
28
29    // Called the system call and stored the size of image.
30    int draw_size = draw(draw_buffer,size);
31
32    // If the size of image is in buffer is greater than size then print a error message
33    if (draw_size == -1)
34    {
35        // file descriptor 1 used to print on the standard output i.e (stdout)
36        printf(1, "Buffer size is too small\n");
37    }
38    else
39    {
40        printf(1, "%s\n", (char *)draw_buffer);
41
42    }
43    free(k); // Deallocating buffer which was used to take buffer size from the user.
44
45    return 0;
```

Question-2:

I included *_Drawtest* line in the *USER PROGRAMS (UPROGS)* section in *makefile.* By doing this we make Drawtest.c available for xv6 source code for compilation.
After this is executed the following commands;

```
ashrith@ashrith-VirtualBox:~$ cd xv6*
ashrith@ashrith-VirtualBox:~/xv6-public$ make clean; make ; make qemu-nox
rm -f *.tex *.dvi *.idx *.aux *.log *.ind *.ilg \
*.o *.d *.asm *.sym vectors.S bootblock entryother \
initcode initcode.out kernel xv6.img fs.img kernelmemfs \
xv6memfs.img mkfs .gdbinit \
_cat _echo _forktest _grep _init _kill _ln _ls _mkdir _rm _sh _stressfs _usertests _wc _zombie _Drawtest
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-
```

By doing this we are reloading the qemu terminal.
After entering the xv6 command shell prompt, we checked the contents of fs.img by using *ls.*
And as we can see a system call named *Drawtest* has been added.
And then executed Drawtest and entered the buffer size to get the ascii image from the kernel to print it on console.

```
$ Drawtest
ENTER BUFFER SIZE : 1600
                          ,ood8888booo,
                       ,od8          8bo,
                    ,od                 bo,
                  ,d8                     8b,
                 ,o                         o,    ,a8b
                ,8                           8,,od8  8
                8'                           d8'     8b
                8                           d8'ba     aP'
                Y,                        o8'         aP'
                 Y8,                      YaaaP'    ba
                  Y8o                    Y8'        88
                   `Y8                ,8"          `P
                     Y8o          ,d8P'              ba
                  ooood8888888P"""'                 P'
               ,od                                  8
            ,dP     o88o                           o'
           ,dP          8                          8
         ,d'   oo       8                        ,8
         $    d$"8      8          Y    Y  o     8
        d    d  d8    od  ""booooooooob   d"" 8   8
        $    8  d   ood' ,    8        b  8   '8  b
        $   $  8  8      d  d8          `b  d    '8  b
         $  $ 8   b     Y  d8           8 ,P     '8  b
         `$$  Yb  b      8b 8b          8 8,       '8  o,
              `Y  b       8o  $$o       d  b         b   $o
               8   '$      8$,,$"       $   $o       '$o$$
                $o$$P"                   $$o$
```

```
$ Drawtest
ENTER BUFFER SIZE : 100
Buffer size is too small
```

And we can see that when the buffer size was 1600 we get the image, but when the buffer size was 100 we get an error saying *buffer size is too small.*

```
$ ls
.              1 1 512
..             1 1 512
README         2 2 2286
cat            2 3 16268
echo           2 4 15120
forktest       2 5 9436
grep           2 6 18488
init           2 7 15708
kill           2 8 15152
ln             2 9 15004
ls             2 10 17632
mkdir          2 11 15248
rm             2 12 15224
sh             2 13 27860
stressfs       2 14 16140
usertests      2 15 67244
wc             2 16 17004
zombie         2 17 14816
Drawtest       2 18 15832
console        3 19 0
$
```