

## CS 344: OPERATING SYSTEMS LABS

### GROUP 12

#### ASSIGNMENT\_0B-1

220123029 KAVURI VEDA VARSHA  
220123030 KODIBOYINA LEELA VARSHINI  
220123031 KOJJA VAMSI KRISHNA  
220123052 RAMINENI HARDHIKA

#### EXERCISE 1:

As we have to create a system call we modified these files in xv6 public folder:

- syscall.h
- syscall.c
- sysproc.c
- user.h
- usys.S
- Makefile

#### Syscall.c

```
85 extern int sys_chdir(void);
86 extern int sys_close(void);
87 extern int sys_dup(void);
88 extern int sys_exec(void);
89 extern int sys_exit(void);
90 extern int sys_fork(void);
91 extern int sys_fstat(void);
92 extern int sys_getpid(void);
93 extern int sys_kill(void);
94 extern int sys_link(void);
95 extern int sys_mkdir(void);
96 extern int sys_mknod(void);
97 extern int sys_open(void);
98 extern int sys_pipe(void);
99 extern int sys_read(void);
100 extern int sys_sbrk(void);
101 extern int sys_sleep(void);
102 extern int sys_unlink(void);
103 extern int sys_wait(void);
104 extern int sys_write(void);
105 extern int sys_uptime(void);
106 extern int sys_draw(void);
```

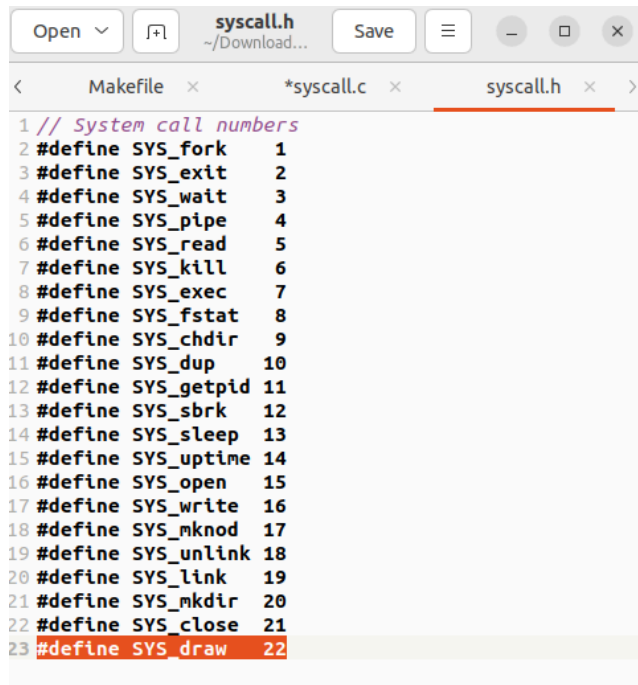
```

107
108 static int (*syscalls[])(void) = {
109 [SYS_fork]      sys_fork,
110 [SYS_exit]      sys_exit,
111 [SYS_wait]      sys_wait,
112 [SYS_pipe]      sys_pipe,
113 [SYS_read]      sys_read,
114 [SYS_kill]      sys_kill,
115 [SYS_exec]      sys_exec,
116 [SYS_fstat]     sys_fstat,
117 [SYS_chdir]     sys_chdir,
118 [SYS_dup]       sys_dup,
119 [SYS_getpid]    sys_getpid,
120 [SYS_sbrk]      sys_sbrk,
121 [SYS_sleep]     sys_sleep,
122 [SYS_uptime]    sys_uptime,
123 [SYS_open]      sys_open,
124 [SYS_write]     sys_write,
125 [SYS_mknod]     sys_mknod,
126 [SYS_unlink]    sys_unlink,
127 [SYS_link]      sys_link,
128 [SYS_mkdir]     sys_mkdir,
129 [SYS_close]     sys_close,
130 [SYS_draw]      sys_draw,
131 };
132
133 void
134 syscall(void)
135 {

```

We modified syscall.c by adding a line “extern int sys\_draw(void);” and “[SYS\_draw] sys\_draw”

## Syscall.h



The screenshot shows a code editor window titled 'syscall.h' with a file path of '~/Download...'. The editor has tabs for 'Makefile', '\*syscall.c', and 'syscall.h'. The 'syscall.h' tab is active, showing the following code:

```

1 // System call numbers
2 #define SYS_fork    1
3 #define SYS_exit    2
4 #define SYS_wait    3
5 #define SYS_pipe    4
6 #define SYS_read    5
7 #define SYS_kill    6
8 #define SYS_exec    7
9 #define SYS_fstat   8
10 #define SYS_chdir   9
11 #define SYS_dup    10
12 #define SYS_getpid  11
13 #define SYS_sbrk   12
14 #define SYS_sleep  13
15 #define SYS_uptime 14
16 #define SYS_open   15
17 #define SYS_write  16
18 #define SYS_mknod  17
19 #define SYS_unlink 18
20 #define SYS_link    19
21 #define SYS_mkdir   20
22 #define SYS_close   21
23 #define SYS_draw    22

```

Here we modified syscall.h by adding “define SYS\_draw 22”

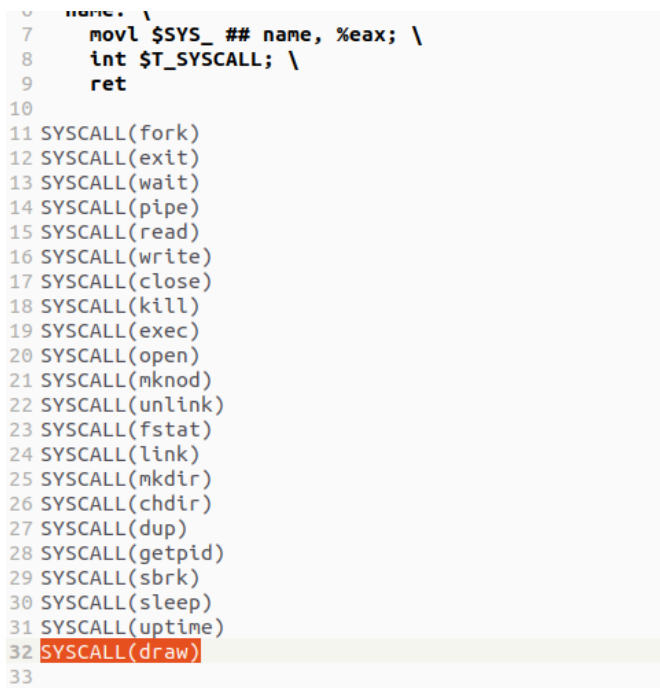
## User.h



```
5 int fork(void);
6 int exit(void) __attribute__((noreturn));
7 int wait(void);
8 int pipe(int*);
9 int write(int, const void*, int);
10 int read(int, void*, int);
11 int close(int);
12 int kill(int);
13 int exec(char*, char**);
14 int open(const char*, int);
15 int mknod(const char*, short, short);
16 int unlink(const char*);
17 int fstat(int fd, struct stat*);
18 int link(const char*, const char*);
19 int mkdir(const char*);
20 int chdir(const char*);
21 int dup(int);
22 int getpid(void);
23 char* sbrk(int);
24 int sleep(int);
25 int uptime(void);
26 int draw(void *buf, uint size);
27 // ulib.c
28 int stat(const char*, struct stat*);
29 char* strcpy(char*, const char*);
30 void *memmove(void*, const void*, int);
```

Here we modified user.h by adding “int draw(void \*buf,uint size);”

## usys.S



```
7     movl $SYS_ ## name, %eax; \
8     int $T_SYSCALL; \
9     ret
10
11 SYSCALL(fork)
12 SYSCALL(exit)
13 SYSCALL(wait)
14 SYSCALL(pipe)
15 SYSCALL(read)
16 SYSCALL(write)
17 SYSCALL(close)
18 SYSCALL(kill)
19 SYSCALL(exec)
20 SYSCALL(open)
21 SYSCALL(mknod)
22 SYSCALL(unlink)
23 SYSCALL(fstat)
24 SYSCALL(link)
25 SYSCALL(mkdir)
26 SYSCALL(chdir)
27 SYSCALL(dup)
28 SYSCALL(getpid)
29 SYSCALL(sbrk)
30 SYSCALL(sleep)
31 SYSCALL(uptime)
32 SYSCALL(draw)
33
```

Here we modified usys.S by adding “SYSCALL(draw)”

## Makefile

```
68 UPROGS=\
69     _cat\
70     _echo\
71     _forktest\
72     _grep\
73     _init\
74     _kill\
75     _ln\
76     _ls\
77     _mkdir\
78     _rm\
79     _sh\
80     _stressfs\
81     _usertests\
82     _wc\
83     _testdraw\
84     _zombie\
85
86 fs.img: mkfs README $(UPROGS)
87     ./mkfs fs.img README $(UPROGS)
88
89 -include *.d
90
91 clean:
92     rm -f *.tex *.dvi *.idx *.aux *.log *.
93     *.ilg \
94     *.o *.d *.asm *.sym vectors.S bootblo
95     entryother \
96     initcode initcode.out kernel xv6.img
97     fs.img kernelmemfs \
```

Here we modified makefile by adding “\_testdraw\”

## Testdraw.c

```
1 |
2 #include "types.h"
3 #include "user.h"
4 #include "stat.h"
5
6 int main(int argc, char *argv[]) {
7     char buf[256];
8     int result;
9
10    result = draw(buf, sizeof(buf));
11
12    if (result < 0) {
13        printf(1, "draw failed\n");
14        exit();
15    }
16
17    printf(1, "draw succeeded, %d bytes copied:\n",
18    result);
19    printf(1, "%s\n", buf);
20    exit();
21 }
22
```

Here we added a new file testdraw.c to xv6 folder

After adding the system call

```
guest@iitg-Vostro-3910: ~/Downloads/xv6/xv6-public-master
stdraw.o testdraw.c
ld -m elf_i386 -N -e main -Ttext 0 -o _testdraw testdraw.o ulib.o usys.o printf.o umalloc.o
objdump -S _testdraw > testdraw.asm
objdump -t _testdraw | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > testdraw.sym
gcc -fno-pic -static -fno-builtin -fno-strict-aliasing -O2 -Wall -MD -ggdb -m32 -Werror -fno-omit-frame-pointer -fno-stack-protector -fno-pie -no-pie -c -o zombie.o zombie.c
ld -m elf_i386 -N -e main -Ttext 0 -o _zombie zombie.o ulib.o usys.o printf.o umalloc.o
objdump -S _zombie > zombie.asm
objdump -t _zombie | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > zombie.sym
./mkfs fs.img README _cat _echo _forktest _grep _init _kill _ln _ls _mkdir _rm _sh _stressfs _usertests _wc _testdraw _zombie
nmeta 59 (boot, super, log blocks 30 inode blocks 26, bitmap blocks 1) blocks 94
1 total 1000
ballocc: first 674 blocks have been allocated
ballocc: write bitmap block at sector 58
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...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ testdraw
draw succeeded, 73 bytes copied:

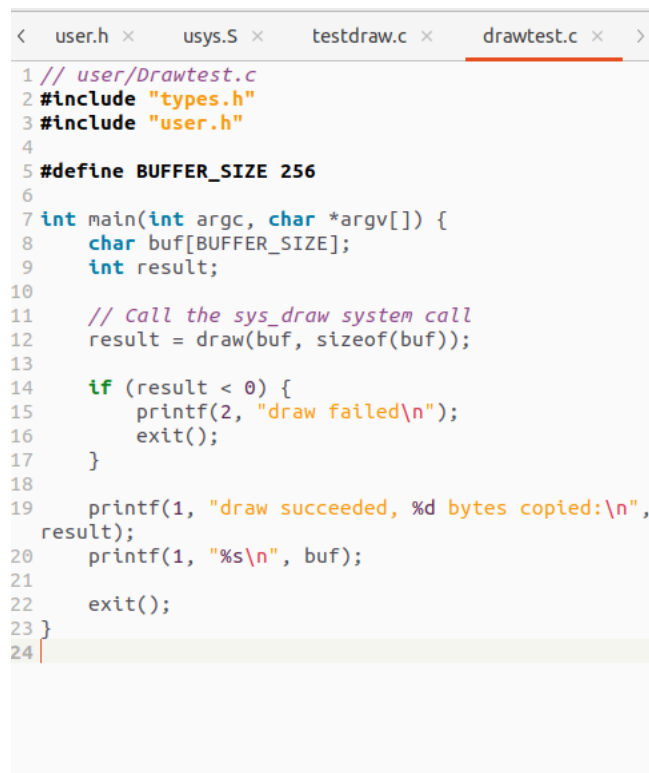
  { }
 | ( ) ( ) |
  ^
 | | | |
 | | | |
 | | | |
 | | | |
$
```

After running the command testdraw it prints the ascii\_art

## EXERCISE 2:

To print the `ascii_art` image in the console we have added the file `drawtest.c` and also modified `makefile`

### Drawtest.c

A screenshot of a code editor window with four tabs: 'user.h', 'usys.S', 'testdraw.c', and 'drawtest.c'. The 'drawtest.c' tab is active and shows the following C code:

```
1 // user/Drawtest.c
2 #include "types.h"
3 #include "user.h"
4
5 #define BUFFER_SIZE 256
6
7 int main(int argc, char *argv[]) {
8     char buf[BUFFER_SIZE];
9     int result;
10
11     // Call the sys_draw system call
12     result = draw(buf, sizeof(buf));
13
14     if (result < 0) {
15         printf(2, "draw failed\n");
16         exit();
17     }
18
19     printf(1, "draw succeeded, %d bytes copied:\n",
20           result);
21     printf(1, "%s\n", buf);
22     exit();
23 }
24
```

Here we added a new file `drawtest.c` to `xv6` folder

### Makefile

