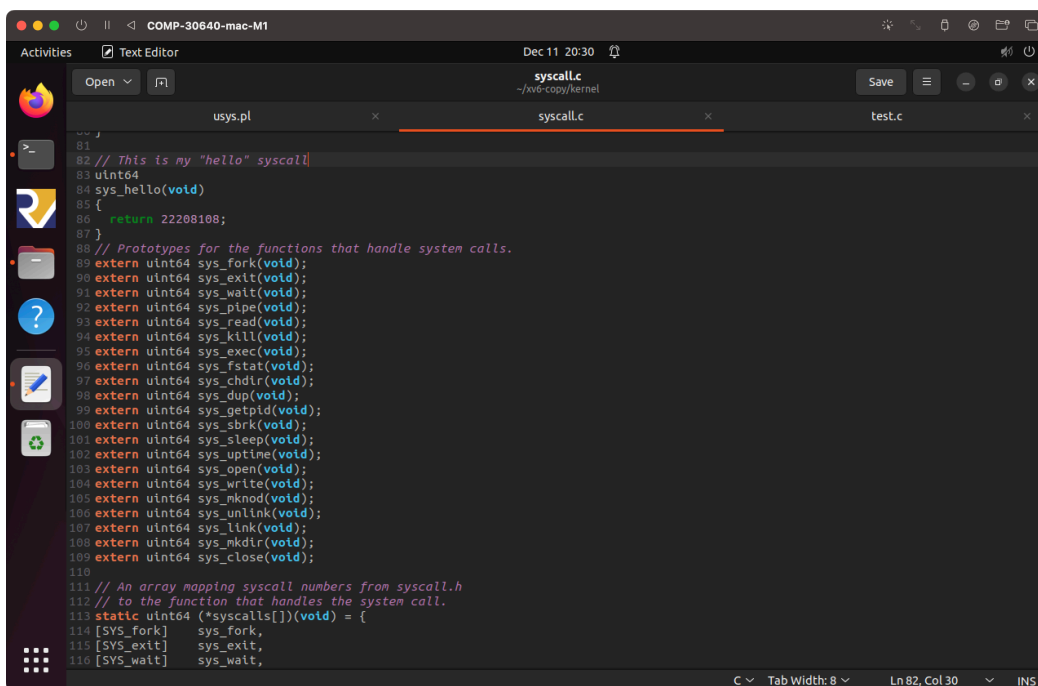


COMP30640: Operating Systems – Finbar Allan

Assignment 4

Question 1

A copy of the xv6-riscv folder was created and named xv6-copy to implement a new system calling for printing Student ID: 22208108. The kernel syscall files were modified to add the `sys_hello()` system call handler function and this function was registered in the syscall array. Figure 1 shows a screenshot of the new system call handler function from the virtual machine; a copy of all files which have been updated to achieve the required output are included in this submission.



```
81
82 // This is my "hello" syscall
83 uint64
84 sys_hello(void)
85 {
86     return 22208108;
87 }
88 // Prototypes for the functions that handle system calls.
89 extern uint64 sys_fork(void);
90 extern uint64 sys_exit(void);
91 extern uint64 sys_wait(void);
92 extern uint64 sys_pipe(void);
93 extern uint64 sys_read(void);
94 extern uint64 sys_kill(void);
95 extern uint64 sys_exec(void);
96 extern uint64 sys_fstat(void);
97 extern uint64 sys_chdir(void);
98 extern uint64 sys_dup(void);
99 extern uint64 sys_getpid(void);
100 extern uint64 sys_sbrk(void);
101 extern uint64 sys_sleep(void);
102 extern uint64 sys_uptime(void);
103 extern uint64 sys_open(void);
104 extern uint64 sys_write(void);
105 extern uint64 sys_mknod(void);
106 extern uint64 sys_unlink(void);
107 extern uint64 sys_link(void);
108 extern uint64 sys_mkdir(void);
109 extern uint64 sys_close(void);
110
111 // An array mapping syscall numbers from syscall.h
112 // to the function that handles the system call.
113 static uint64 (*syscalls[])(void) = {
114     [SYS_fork] sys_fork,
115     [SYS_exit] sys_exit,
116     [SYS_wait] sys_wait,
```

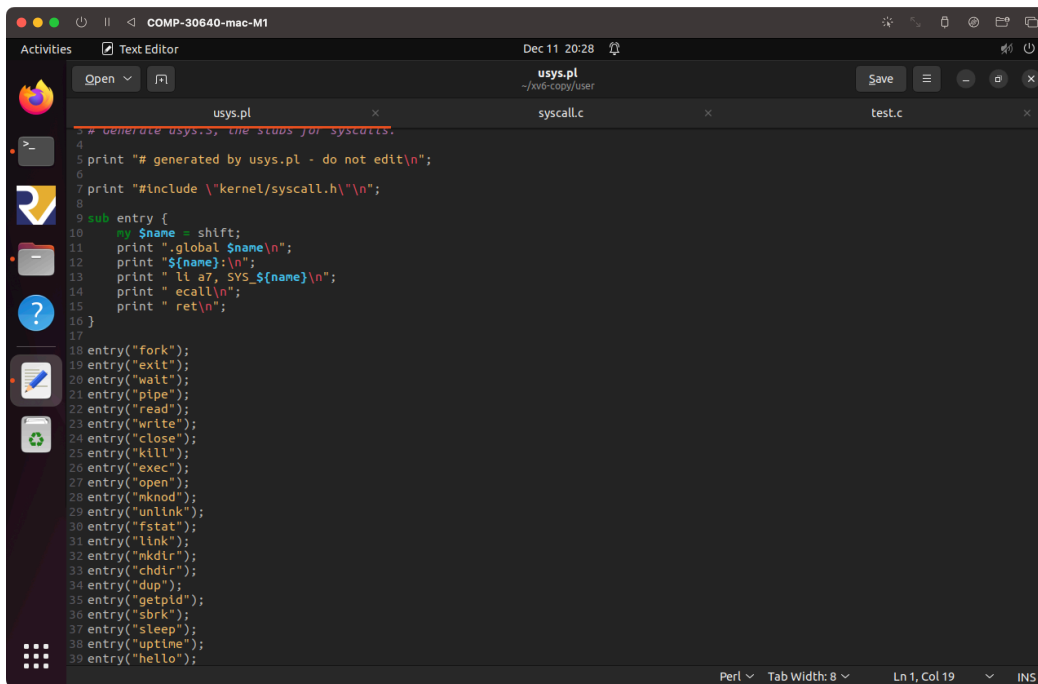
Figure 1: syscall.c file updated to include `sys_hello()` function.

To make the above system call handler function accessible the `usys.py` file must be modified to generate a user-mode C function which is achieved by including the line highlighted below. Figure 2 displays a screenshot of this line in the Perl file from the virtual environment.

```
entry("hello")
```

Similarly, the Makefile must be modified to include the following entry for programs which are accessible to the user (UPROGS):

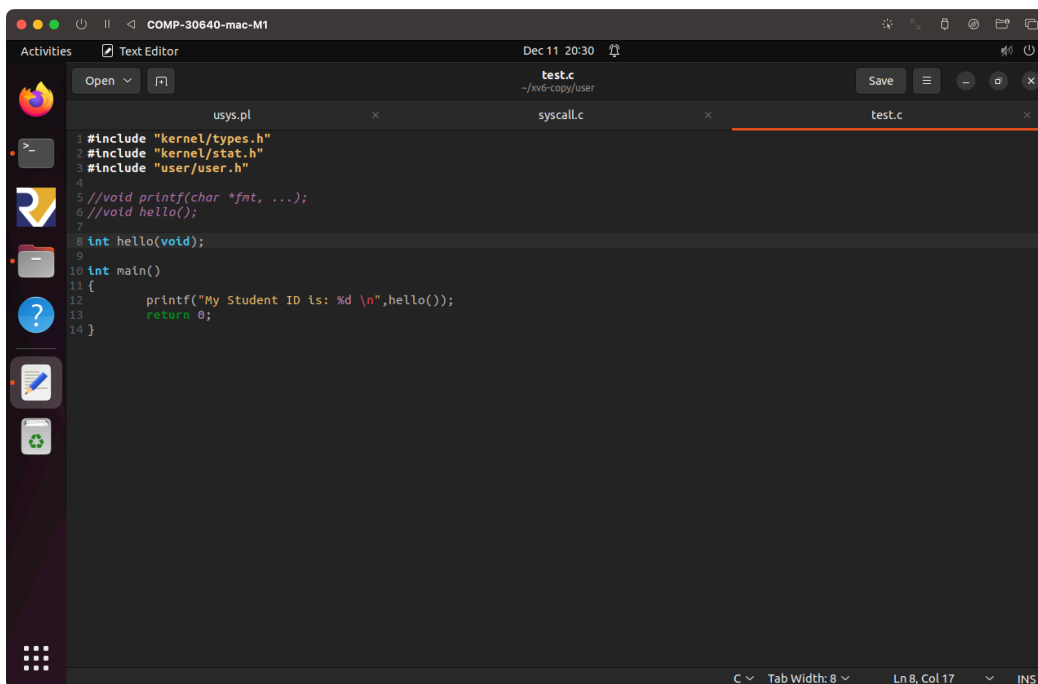
```
$U/_test\
```



```
usys.pl
# Generate usys.s, the stubs for syscalls.
4
5 print "# generated by usys.pl - do not edit\n";
6
7 print "#include \"kernel/syscall.h\"\n";
8
9 sub entry {
10     my $name = shift;
11     print ".global $name\n";
12     print "$name:\n";
13     print "    li a7, SYS_$name\n";
14     print "    ecall\n";
15     print "    ret\n";
16 }
17
18 entry("fork");
19 entry("exit");
20 entry("wait");
21 entry("pipe");
22 entry("read");
23 entry("write");
24 entry("close");
25 entry("kill");
26 entry("exec");
27 entry("open");
28 entry("mknod");
29 entry("unlink");
30 entry("fstat");
31 entry("link");
32 entry("mkdir");
33 entry("chdir");
34 entry("dup");
35 entry("getpid");
36 entry("sbrk");
37 entry("sleep");
38 entry("uptime");
39 entry("hello");
```

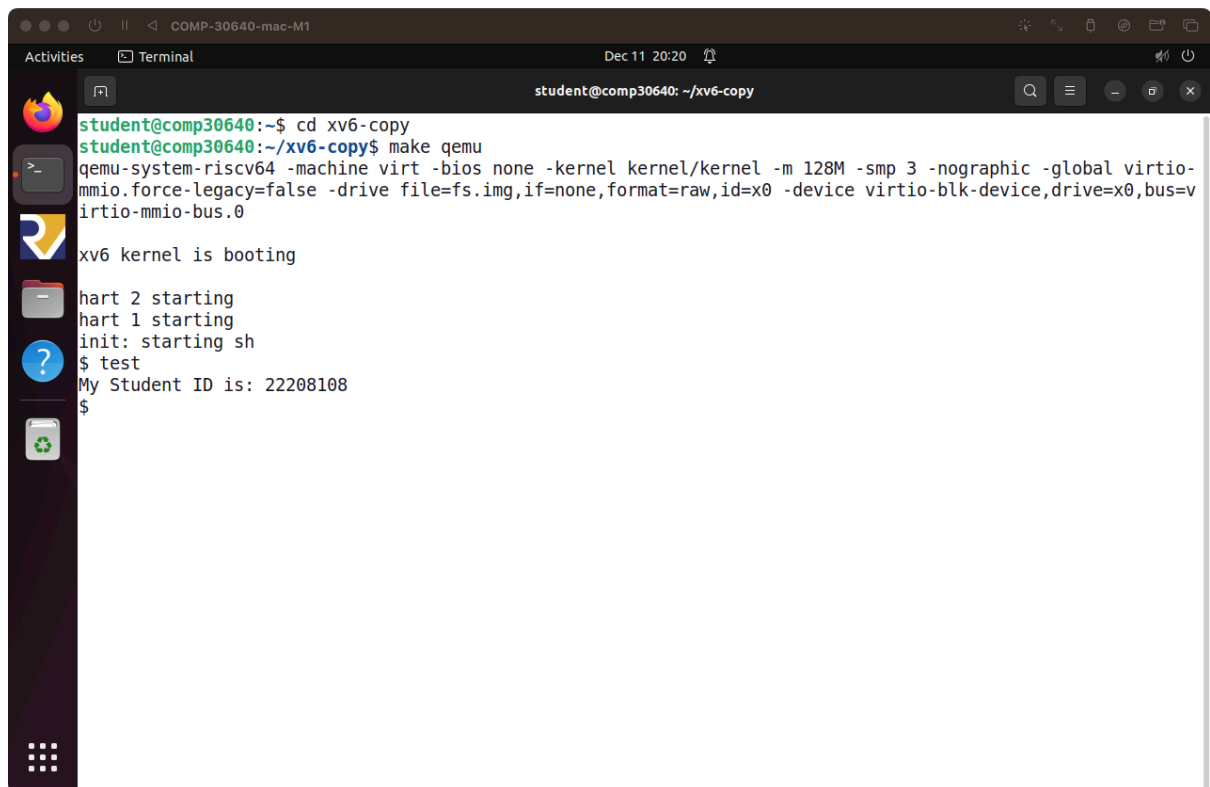
Figure 2: usys.pl file updated to register system call handler.

The test.c utility program invokes a new system call to obtain the Student ID returned by the `sys_hello()` function. As this program is written in C the `printf` function is implemented to display text "My Student ID is:" and the `%d` acts as a placeholder for the returned integer. As xv6-riscv does not have a standard `stdio.h` file the new system call handler function must be declared explicitly as an external function as is displayed in line 8 of Figure 3.



```
test.c
1 #include "kernel/types.h"
2 #include "kernel/stat.h"
3 #include "user/user.h"
4
5 //void printf(char *fmt, ...);
6 //void hello();
7
8 int hello(void);
9
10 int main()
11 {
12     printf("My Student ID is: %d\n",hello());
13     return 0;
14 }
```

Figure 3: test.c system call handler function with explicit declaration.



```
student@comp30640:~$ cd xv6-copy
student@comp30640:~/xv6-copy$ make qemu
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 3 -nographic -global virtio-mmio.force-legacy=false -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0
xv6 kernel is booting
hart 2 starting
hart 1 starting
init: starting sh
$ test
My Student ID is: 22208108
$
```

Figure 4: Output of new system call function in xv6-copy folder.

Question 2

The `downloader.py` python program provided has been updated to include the additional functionality of processing simultaneous downloads by implementing threading. As is the case with the files in Question 1 of this assignment submission the updated python file has been included and an extract which highlights the threading implantation is displayed below:

```
for i in img_urls:
    threads.append(threading.Thread(target=download_image,args=(i,)))
for i in threads:
    i.start()
for i in threads:
    i.join()
```

The first for loop cycles through the images contained in the given array and creates a thread with the image downloaded function as its target and a single image from the array as its argument. These threads are then appended to an empty array allowing them to be held for the second for loop which begins each thread by calling the `.start()` method on all items in the list. Finally the `.join()` method is implemented to make the main thread wait until the newly created threads have executed and links the downloaded images back together.

An elementary script was also developed as a control to download the images individually without implementing threading. Both downloader files were tested five times with the recorded results being displayed in Table 1:

Table 1: Results recorded from testing image downloader programs

(seconds)	Test #1	Test #2	Test #3	Test #4	Test #5	Average
No Threading	20.44	19.14	22.91	15.47	20.86	19.76
With Threading	9.78	11.52	12.05	13.14	17.47	12.79

Question 3

To compose a bash shell script which collects information from the `/etc/passwd` file which contains system information and concatenates the result with the paths to all world readable files on the system the following command is implemented:

```
{ cat /etc/passwd & find $PWD -perm -666; } > report.txt
```

The `cat` command takes the file as an argument and reads its contents. The `find` command searches for files by permission with the `-perm` condition of 666 translation to the binary value representing files which are writable for the owner, group and users of the file system. `$PWD` denotes that the full path name should be returned and the `&` operator concatenates the results of the `cat` and `find` commands. The curly braces containing these commands target a report which is not already on the file system, hence, `report.txt` is created to include the contents of the bash shell script. The execution and result of this expression is displayed in Figure 5. The full `report.txt` file is included as part of the assignment submission.

```

1 root:x:0:0:root:/root:/bin/bash
2 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
3 bin:x:2:2:bin:/bin:/usr/sbin/nologin
4 sys:x:3:3:sys:/dev:/usr/sbin/nologin
5 sync:x:4:65534:sync:/bin:/bin/sync
6 games:x:5:60:games:/usr/games:/usr/sbin/nologin
7 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
8 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
9 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
10 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
11 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
12 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
13 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
14 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
15 list:x:38:38:Mail list Manager:/var/list:/usr/sbin/nologin
16 ircd:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
17 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
18 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
19 _apt:x:100:65534:/nonexistent:/usr/sbin/nologin
20 systemd-networkd:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
21 systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
22 messagebus:x:103:104:/nonexistent:/usr/sbin/nologin
23 systemd-timesyncd:x:104:105:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
24 pollinate:x:105:1:/var/cache/pollinate:/bin/false
25 sshd:x:106:65534:/run/ssh:/usr/sbin/nologin
26 syslog:x:107:113:/home/syslog:/usr/sbin/nologin
27 uidd:x:108:114:/run/uidd:/usr/sbin/nologin
28 tcpdump:x:109:115:/nonexistent:/usr/sbin/nologin
29 tss:x:110:116:TPM software stack,,,:/var/lib/tpm:/bin/false
30 landscape:x:111:117:/var/lib/landscape:/usr/sbin/nologin
31 student:x:1000:1000:student:/home/student:/bin/bash
32 lxd:x:999:100:/var/snap/lxd/common/lxd:/bin/false
33 rtkit:x:112:118:RealtimeKit,,,:/proc:/usr/sbin/nologin
34 dnsmasq:x:113:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
35 kernoops:x:114:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin
36 systemd-oomd:x:115:121:systemd Userspace OOM Killer,,,:/run/systemd:/usr/sbin/nologin
37 whoopsie:x:116:122:/nonexistent:/bin/false
38 avahi-autoipd:x:117:124:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
39 rchmrc:x:118:66:richmrc_daemon:/usr/lib/richmrc:/usr/sbin/nologin

```

Figure 5: Execution and result of the bash shell script command.