

## Screenshots for Project 1

make qemu-nox command to create environment

```
$ QEMU: Terminated  
root@37a99ec8ed20:/xv6-public# make qemu-nox
```

xv6 environment after command

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
iPXE (https://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1EFCAF60+1EF0AF60 CA00  
  
Booting from Hard Disk..xv6...  
cpu0: starting 0  
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58  
init: starting sh  
$
```

## PART 1

hello.c file

```
C hello.c U X  
C hello.c > ⌂ main(int, char * [] )  
1 //Part 1: Writing a Hello World Program in Xv6  
2 //hello.c file that outputs "Hello Xv6!" to terminal when called  
3  
4 #include "types.h"  
5 #include "user.h"  
6  
7  
8 //copied from echo.c with some changes  
9 int  
10 main(int argc, char *argv[] )  
11 {  
12     printf(1, "Hello Xv6!\n");           //1st arg is the file descriptor,1 is the output to stdout  
13  
14     exit();      //no args in xv6  
15 }
```

output of part 1 hello command

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
SeaBIOS (version 1.16.3-debian-1.16.3-2)  
  
iPXE (https://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1EFCAF60+1EF0AF60 CA00  
  
Booting from Hard Disk..xv6...  
cpu0: starting 0  
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58  
init: starting sh  
$ hello  
Hello Xv6!  
$
```

MAKEFILE with hello command

## M Makefile

```
164 # details:
165 # http://www.gnu.org/software/make/manual
166 .PRECIOUS: %.o
167
168 UPROGS=\
169     _cat\
170     _echo\
171     _forktest\
172     _grep\
173     _init\
174     _kill\
175     _ln\
176     _ls\
177     _mkdir\
178     _rm\
179     _sh\
180     _stressfs\
181     _usertests\
182     _wc\
183     _zombie\
184     _hello\
185     _sleep\
```

## PART 2

### ls.c in 3 screenshots

```
C ls.c > ls(char *)
1  /xv6-public/ls.c · Modified
2  #include "stat.h"
3  #include "user.h"
4  #include "fs.h"
5
6  char*
7  fmtname(char *path)
8  {
9      static char buf[DIRSIZ+1];
10     char *p;
11
12     // Find first character after last slash.
13     for(p=path+strlen(path); p >= path && *p != '/'; p--)
14     | ;
15     p++;
16
17     // Return blank-padded name.
18     if(strlen(p) >= DIRSIZ)
19     | return p;
20     memmove(buf, p, strlen(p));
21     memset(buf+strlen(p), ' ', DIRSIZ-strlen(p));
22     return buf;
23 }
24
25 void
26 ls(char *path)
27 {
28     char buf[512], *p;
29     int fd;
30     struct dirent de;
31     struct stat st;
32
33     if((fd = open(path, 0)) < 0){
34         printf(2, "ls: cannot open %s\n", path);
35         return;
36     }
37
38     if(fstat(fd, &st) < 0){
```

```
C ls.c > ls(char *)
26     ls(char *path)
38     if(fstat(fd, &st) < 0){
39         printf(2, "ls: cannot stat %s\n", path);
40         close(fd);
41         return;
42     }
43
44     switch(st.type){
45     case T_FILE:
46         printf(1, "%s %d %d %d\n", fmtname(path), st.type, st.ino, st.size);
47         break;
48
49     case T_DIR: //switch statement checking if st.type matches T_DIR(directory)
50     if(strlen(path) + 1 + DIRSIZ + 1 > sizeof buf){
51         printf(1, "ls: path too long\n");
52         break;
53     }
54     strcpy(buf, path);
55     p = buf+strlen(buf);
56     *p++ = '/';
57     while(read(fd, &de, sizeof(de)) == sizeof(de)){
58         if(de.inum == 0)
59             continue;
60         //edited block
61         if (de.name[0] == '.') //skip if file has '.' as first character in name(ie hidden file)
62             continue;
63         memmove(p, de.name, DIRSIZ);
64         p[DIRSIZ] = 0;
65         if(stat(buf, &st) < 0){
66             printf(1, "ls: cannot stat %s\n", buf);
67             continue;
68         }
69         if (st.type == T_DIR) { //check if file being passed is a directory; if so, add '/' to end of name
70             //did not get to work for now
71             printf(1, "%s %d %d %d\n", fmtname(buf), st.type, st.ino, st.size);
72         }
73         else //else, proceed as usual
74             printf(1, "%s %d %d %d\n", fmtname(buf), st.type, st.ino, st.size);
75     }
76     break;
77 }
78 close(fd);
79 }

80 int
82 main(int argc, char *argv[])
83 {
84     int i;
85
86     if(argc < 2){
87         ls(".");
88         exit();
89     }
90     for(i=1; i<argc; i++)
91         ls(argv[i]);
92     exit();
93 }
94 }
```

```
C ls.c > ls(char *)
26     ls(char *path)
44     switch(st.type){
57     while(read(fd, &de, sizeof(de)) == sizeof(de)){
73         else //else, proceed as usual
74             printf(1, "%s %d %d %d\n", fmtname(buf), st.type, st.ino, st.size);
75     }
76     break;
77 }
78 close(fd);
79 }

80 int
82 main(int argc, char *argv[])
83 {
84     int i;
85
86     if(argc < 2){
87         ls(".");
88         exit();
89     }
90     for(i=1; i<argc; i++)
91         ls(argv[i]);
92     exit();
93 }
94 }
```

output of ls command in environment(could not implement “/” or sorting)

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
SeaBIOS (version 1.16.3-debian-1.16.3-2)

iPXE (https://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1EFCAF60+1EF0AF60 CA00

Booting from Hard Disk..xv6...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodedstart 32 bmap start 58
init: starting sh
$ ls
README      2 2 2286
cat          2 3 15272
echo         2 4 14176
forktest     2 5 8756
grep         2 6 18208
init         2 7 14772
kill          2 8 14228
ln            2 9 14132
ls            2 10 17516
mkdir        2 11 14260
rm            2 12 14240
sh            2 13 28328
stressfs     2 14 15008
usertests    2 15 63152
wc            2 16 15652
zombie       2 17 13808
hello        2 18 13932
```

## PART 3

hello.c after adding in syscall of hello()

```
C hello.c > main(int, char *[])
1 //Part 1: Writing a Hello World Program in Xv6
2 //hello.c file that outputs "Hello Xv6!" to terminal when called
3
4 #include "types.h"
5 #include "user.h"
6
7
8 //copied from echo.c with some changes
9 int
10 main(int argc, char *argv[])
11 {
12     printf(1, "Hello Xv6!\n");           //1st arg is the file descriptor,1 is the output to stdout
13     hello();   //syscall from kernel
14     exit();    //no args in xv6
15 }
```

## sysproc.c

```
C sysproc.c > sys_hello(void)
80 // return how many clock tick interrupts have occurred
82 int
83 sys_uptime(void)
84 {
85     uint xticks;
86
87     acquire(&tickslock);
88     xticks = ticks;
89     release(&tickslock);
90     return xticks;
91 }
92
93 //add syscall hello() such that contents/functionality of this syscall outputs "Hello from Kernel Mode!"
94 int
95 sys_hello(void)
96 {
97     cprintf("Hello from Kernel Mode!\n"); //prints directly to console (doesnt need stdio.h)
98     return 0;
99 }
```

## syscall.h

```
C syscall.h > SYS_hello
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_hello  22 //added number for syscall hello
```

## syscall.c

```
C syscall.c > syscalls
104     extern int sys_write(void);
105     extern int sys_uptime(void);
106     extern int sys_hello(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_hello]    sys_hello //adding in syscall for hello function
131     ];
132
```

## user.h

```
C user.h > hello(void)
1  struct stat;
2  struct rtcdate;
3
4  // system calls
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 user(void);
27 int hello(void);
```

## usys.S

```
asm usys.S
4  #define SYSCALL(name) \
5      .globl name, \
6      name: \
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(hello)
```

## hello syscall kernel output in environment

```
SeaBIOS (version 1.16.3-debian-1.16.3-2)

iPXE (https://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1EFCAF60+1EF0AF60 CA00

Booting from Hard Disk..xv6...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ hello
exec: fail
exec hello failed
$ hello
Hello Xv6!
Hello from Kernel Mode!
$ 
```

## sleep.c

```
C sleep.c > ...
1  #include "types.h"
2  #include "stat.h"
3  #include "user.h"
4  |
5  int
6  main(int argc, char *argv[])
7  {
8      if (argc < 2) {      //gotten from rm.c
9          printf(2, "Error: Pass argument for ticks\n");
10         exit();
11     }
12     int ticks = atoi(argv[1]);    //convert command line arg to an integer using atoi() as shown in ulib.c
13     sleep(ticks);           //syscall hello
14     exit();
15 }
16
```

MAKEFILE with sleep function added

## M Makefile

```
164 # details:
165 # http://www.gnu.org/software/make/manu
166 .PRECIOUS: %.o
167
168 UPROGS=\
169     _cat\
170     _echo\
171     _forktest\
172     _grep\
173     _init\
174     _kill\
175     _ln\
176     _ls\
177     _mkdir\
178     _rm\
179     _sh\
180     _stressfs\
181     _usertests\
182     _wc\
183     _zombie\
184     _hello\
185     _sleep\
```

## sleep function output

The screenshot shows a terminal window with the following content:

```
C sleep.c > ...
1 2 3 4
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
SeaBIOS (version 1.16.3-debian-1.16.3-2)

iPXE (https://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1EFCAF60+1EF0AF60 CA00

Booting from Hard Disk..xv6...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ sleep 100
$ sleep 1000
$ sleep 1000
```

The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. It also shows the SeaBIOS version and iPXE boot information. The command history shows three instances of the sleep command being run.