NAME SHISHU

REG 2020CA089

Operating System Lab Week 3

Question 1. Keep in mind the utility of fork() and exec() system calls in process creation. Exec is

used to load a process and fork() creates a child process.

• In the xv6 directory, type “make qemu-gdb”, to start qemu along with GNU Debugger.

• Open another terminal window, type “gdb” while in the same xv6 directory. Enter

“source .gdbinit”. Now gdb is connected to the xv6 operating system.

• In the same terminal (gdb), add a breakpoint for exec system call by typing “break exec”.

Then type “continue” to reach the first instance where exec() is called. Typing continue

again takes us to the next point where the exec() is invoked. Type continue the third time,

and in the xv6 OS window run any shell command (ls, cat etc.)

Explanation of whole process:

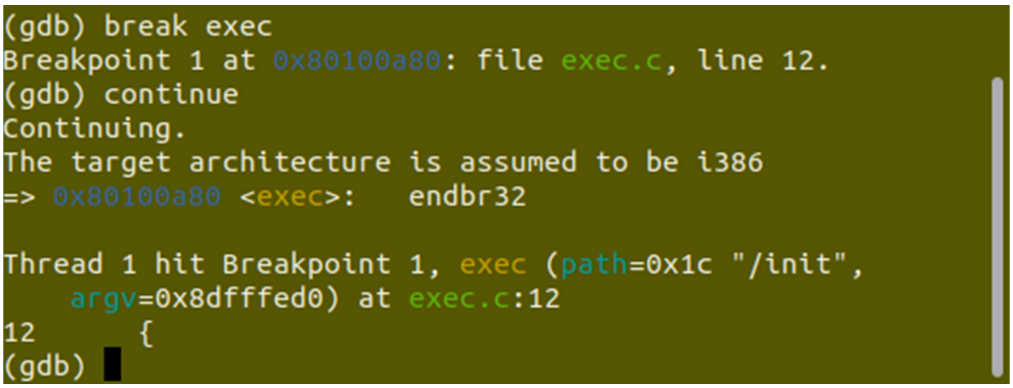
First I executed command make qemu-gdb and after that open another terminal and types gdb

there. In next step, I wrote source .gdbinit, this connected my gdb to xv6 operating system.

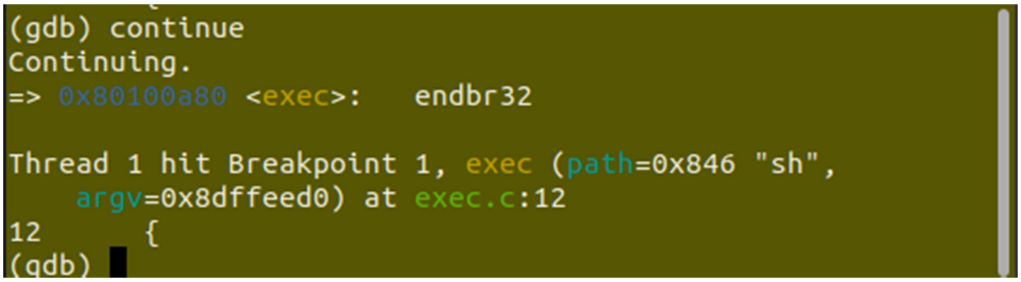
```
(gdb) source .gdbinit
+ target remote localhost:26000
warning: No executable has been specified and target does
not support
determining executable automatically.  Try using the "file
" command.
The target architecture is assumed to be i8086
[f000:fff0]    0xffff0: ljmp   $0x3630,$0xf000e05b
0x0000fff0 in ?? ()
+ symbol-file kernel
warning: A handler for the OS ABI "GNU/Linux" is not built
 into this configuration
of GDB.  Attempting to continue with the default i8086 set
tings.

(gdb)
```

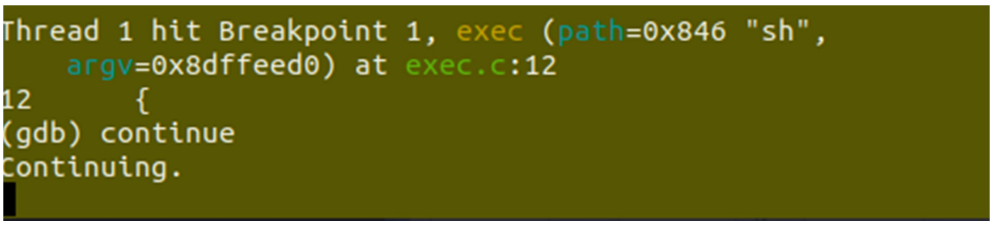
Now, I added breakpoint by typing break exec and then continue. It initiates our operating system and started first process /init as thread 2.



Again after continue, init started a shell process which is the xv6 shell we get when the OS boots.



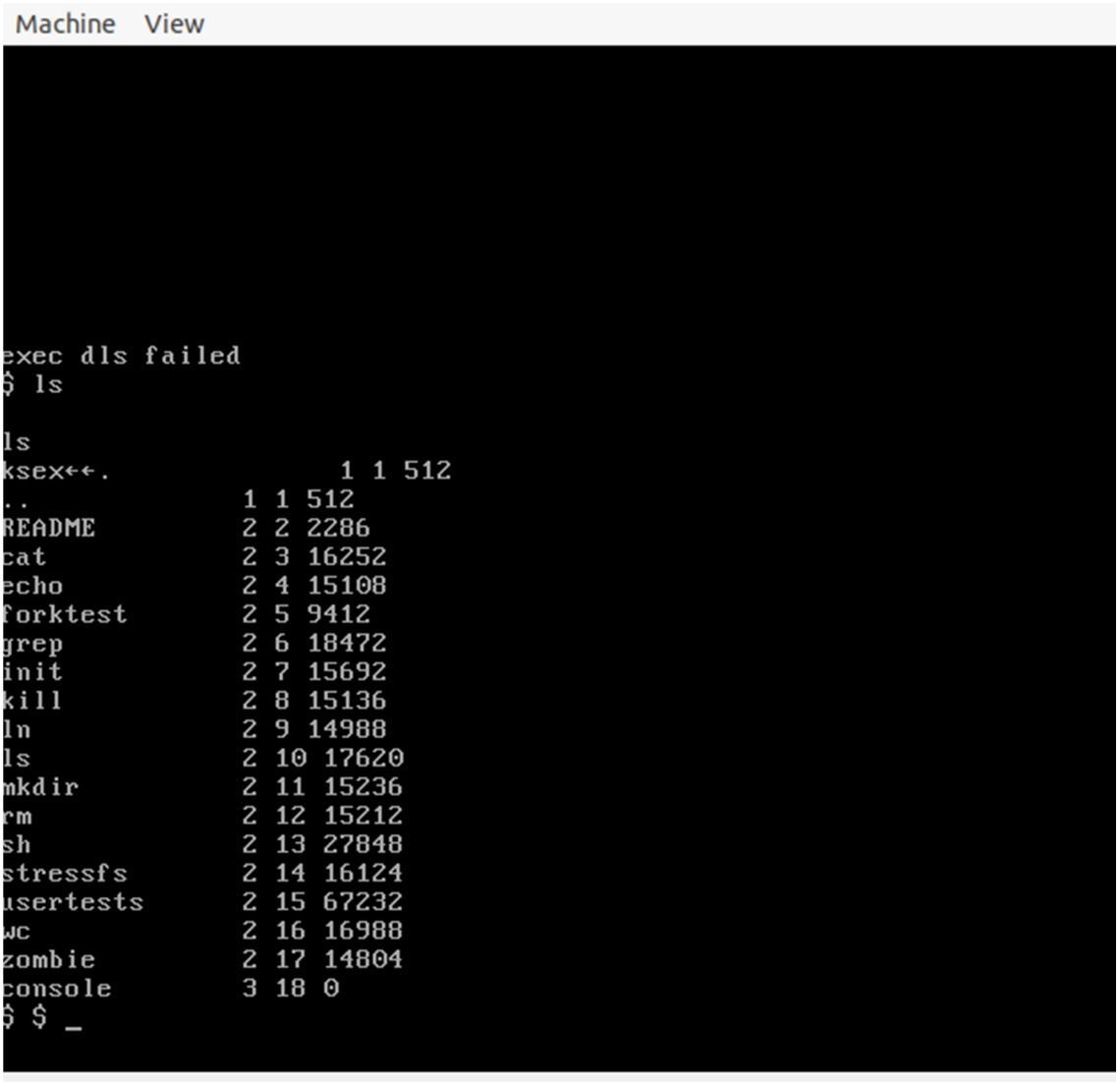
If you continue again, gdb will not return since it is waiting for a command to be started in the shell. Switch to the other window and try typing a command (for example, ls, cat) at which time you will get another break as the shell forks then execs the ls program.



screenshot of other terminal where I executed ls command

```
exec dls failed
$ ls

ls
ksex 1 512
..              1 1 512
README          2 2 2286
cat             2 3 16252
echo            2 4 15108
forktest        2 5 9412
grep            2 6 18472
init            2 7 15692
kill            2 8 15136
ln              2 9 14988
ls              2 10 17620
mkdir           2 11 15236
rm              2 12 15212
sh              2 13 27848
stressfs        2 14 16124
usertests       2 15 67232
wc              2 16 16988
zombie          2 17 14804
console         3 18 0
$ $
```



Question 2. On xv6 shell, type “ls” command. Your task is to make this command more user

friendly, by finding out what the numbers in the second column of the result represent.

The xv6 source code files are well documented using comments.

Edit the C file for this command, find out what the numbers in the second column represent (hint :

you might want to inspect the #included files) and modify the code appropriately so that it prints a

descriptive string in the second column instead of a number, when “ls” is called.

Note: every time you modify xv6, make has to be called again to implement those changes.

Write a document file whatever changes you have made along with the output screenshot

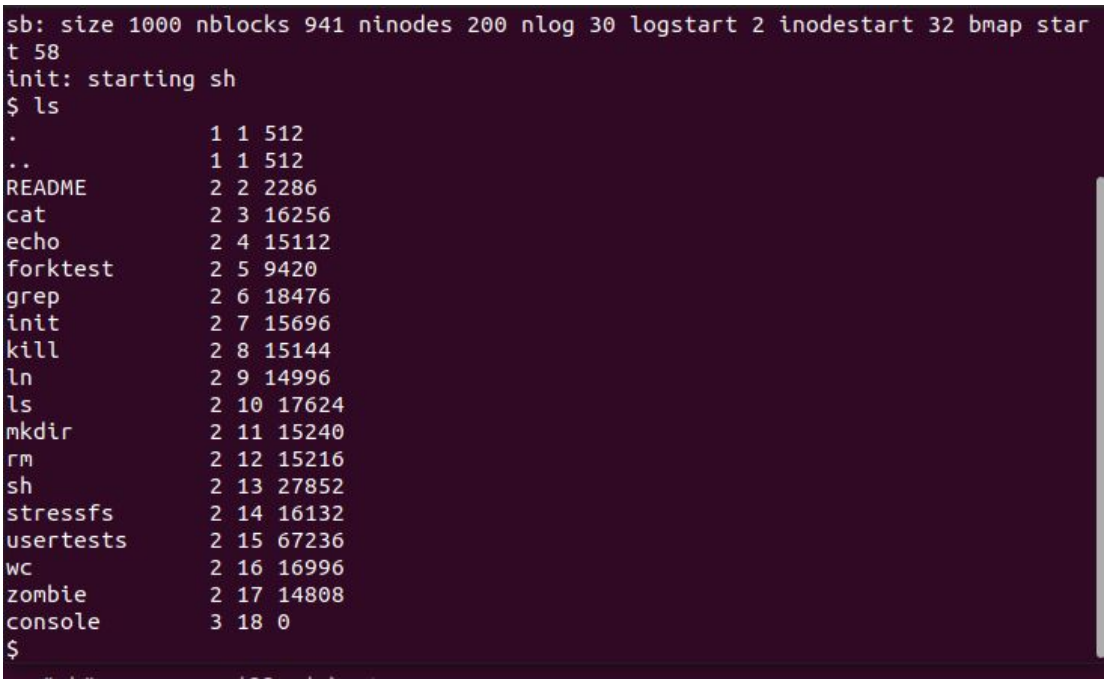
Explanation:

To make second column of ls command more use friendly. I did few changes in ls.c file from line 67 to line 73. Basically, First I find the meaning of status value(1, 2, 3) from stat.h header file. Then I use if-else in ls.c file as you can see in below screenshot from line 67. I changed second format specifier of line 69, 71, 73 with appropriate meaning as I found in stat.h header file.

```
42  }
43
44  switch(st.type){
45  case T_FILE:
46    printf(1, "%s %s %d %d\n", fmtname(path), st.type, st.ino, st.size);
47    break;
48
49  case T_DIR:
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      memmove(p, de.name, DIRSIZ);
61      p[DIRSIZ] = 0;
62      if(stat(buf, &st) < 0){
63        printf(1, "ls: cannot stat %s\n", buf);
64        continue;
65      }
66
67      //Changes done here (Assignment 3)
68      if(st.type == 2)
69        printf(1, "%s %s %d %d\n", fmtname(buf), "File", st.ino, st.size);
70      else if(st.type == 1)
71        printf(1, "%s %s %d %d\n", fmtname(buf), "Directory", st.ino, st.size);
72      else
73        printf(1, "%s %s %d %d\n", fmtname(buf), "Device", st.ino, st.size);
74      //end
75    }
76    break;
```

Then after saving ls.c file. I started qemu by make qemu command, and run ls command there.

Before Changes in ls.c:



After Changes in ls.c