COMP 2560 Fall 2020

Lab 5

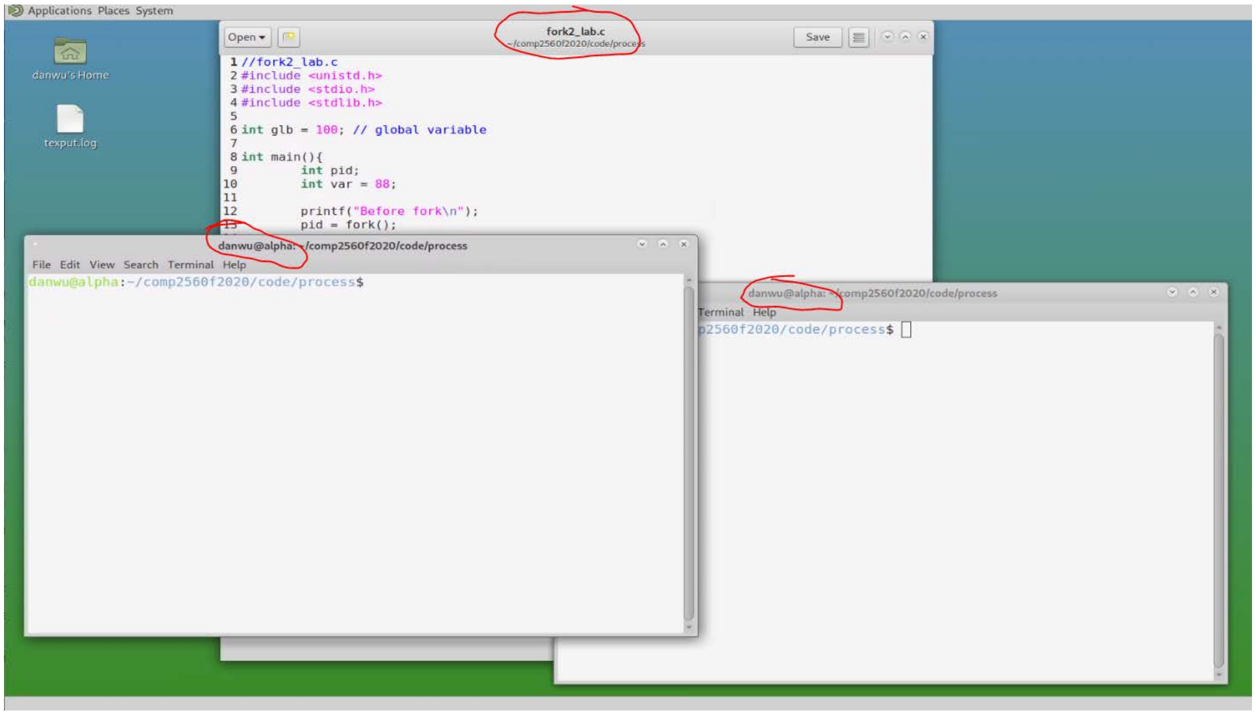
Part 1

GDB is capable of debugging multiple process programs. In this lab, we will study how to debug C programs with fork functions.

You only need to know a few more GDB commands than what you learned before and follow a simple procedure detailed below.

By default, GDB will step into the parent process after the call to fork() and let the child process run unimpeded. To debug both the parent and the child processes, I modified the fork2.c program we discussed in class for this lab, and use this modified program which I call it fork2_lab.c as an example, do the following. (Please study the code of fork2_lab.c and compare it with the original fork2.c to see the difference. A video is also provided to explain the following steps in more detail.)

1. Open two terminal windows. And open the fork2_lab.c file as well.

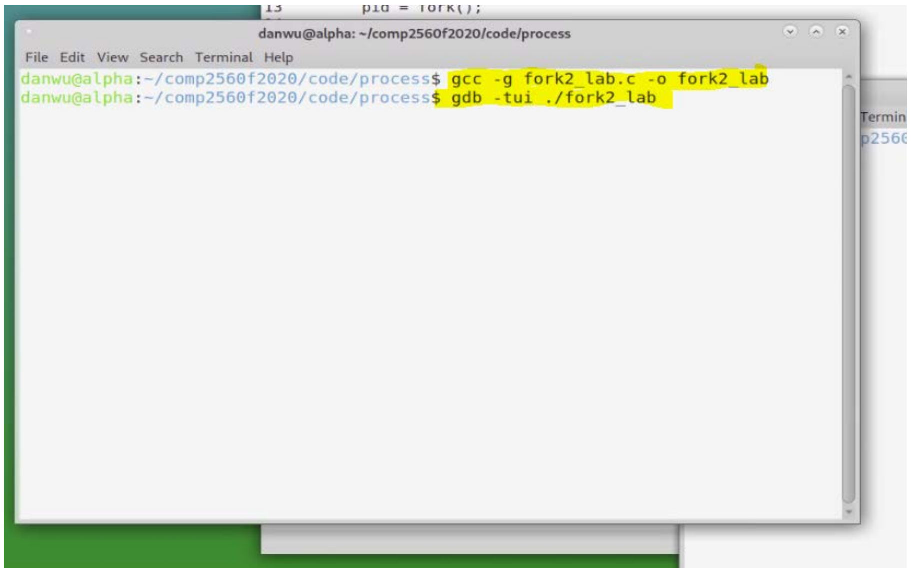

Applications Places System
danwu's Home
texput.log
Open
fork2_lab.c
~/comp2560f2020/code/process
Save
danwu@alpha: ~/comp2560f2020/code/process
File Edit View Search Terminal Help
danwu@alpha: ~/comp2560f2020/code/process
Terminal Help

```
1 //fork2_lab.c
2 #include <unistd.h>
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 int glb = 100; // global variable
7
8 int main(){
9         int pid;
10        int var = 88;
11
12        printf("Before fork\n");
13        pid = fork();
```

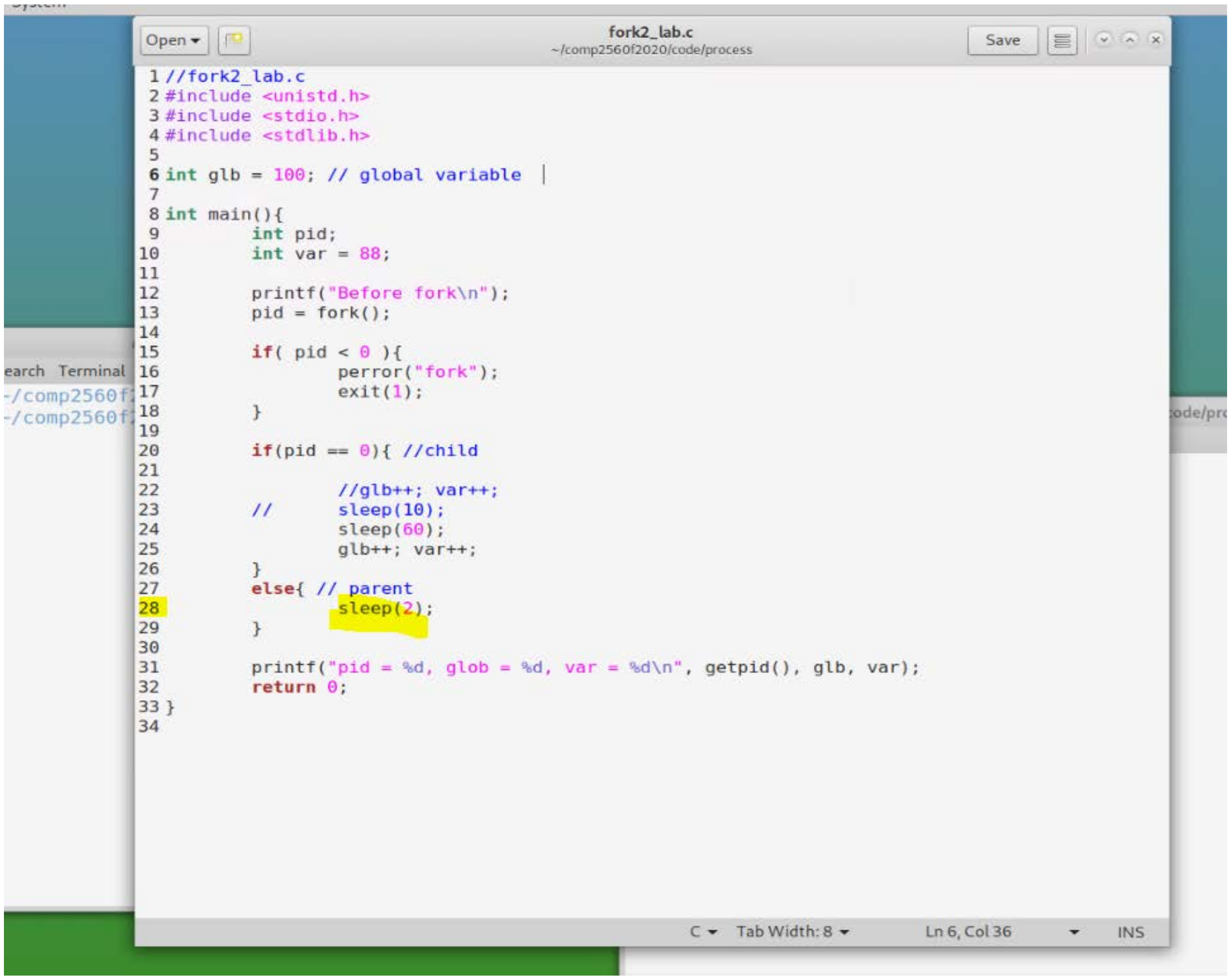
```
danwu@alpha:~/comp2560f2020/code/process$
```

```
p2560f2020/code/process$
```

2. Pick one terminal for parent process and run GDB (gdb -tui ./fork2_lab), set up a breakpoint on a statement after the call to the fork() (but not in any code the child will be executing). For example, in fork2_lab.c (posted online already), I could set a breakpoint at the line 28 “sleep(2);”. See figures below.



```
danwu@alpha:~/comp2560f2020/code/process$ gcc -g fork2_lab.c -o fork2_lab
danwu@alpha:~/comp2560f2020/code/process$ gdb -tui ./fork2_lab
```

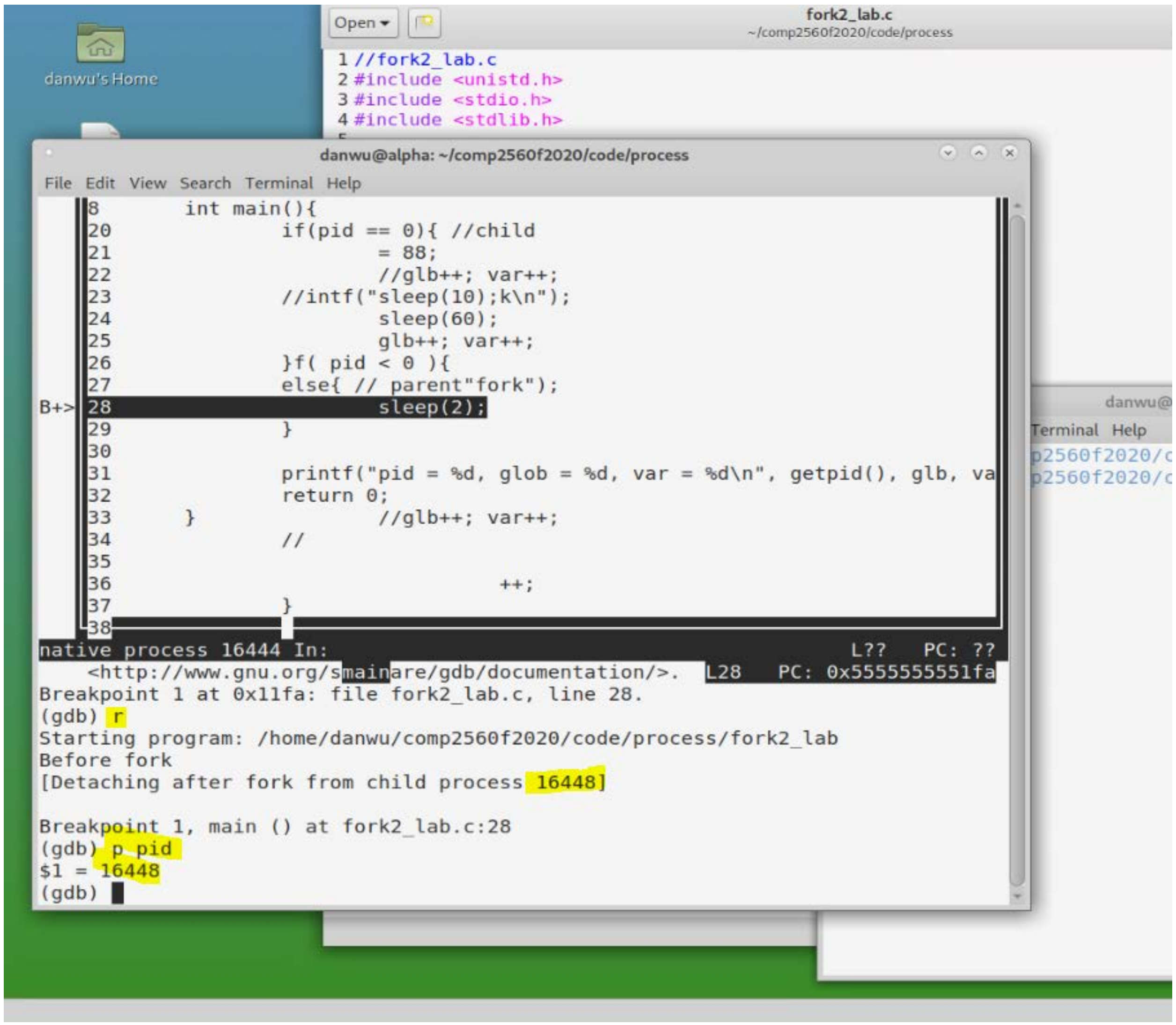


```c
//fork2_lab.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int glb = 100; // global variable

int main(){
	int pid;
	int var = 88;

	printf("Before fork\n");
	pid = fork();

	if( pid < 0 ){
		perror("fork");
		exit(1);
	}

	if(pid == 0){ //child

		//glb++; var++;
	//	sleep(10);
		sleep(60);
		glb++; var++;
	}
	else{ // parent
		sleep(2);
	}

	printf("pid = %d, glob = %d, var = %d\n", getpid(), glb, var);
	return 0;
}
```

Note you need to compile your source code file using -g option for debugging. For example,

gcc **-g** fork2_lab.c -o fork2_lab

3. Run the parent process to the breakpoint. Note the value returned by fork() in the parent process holds the process ID of the child.

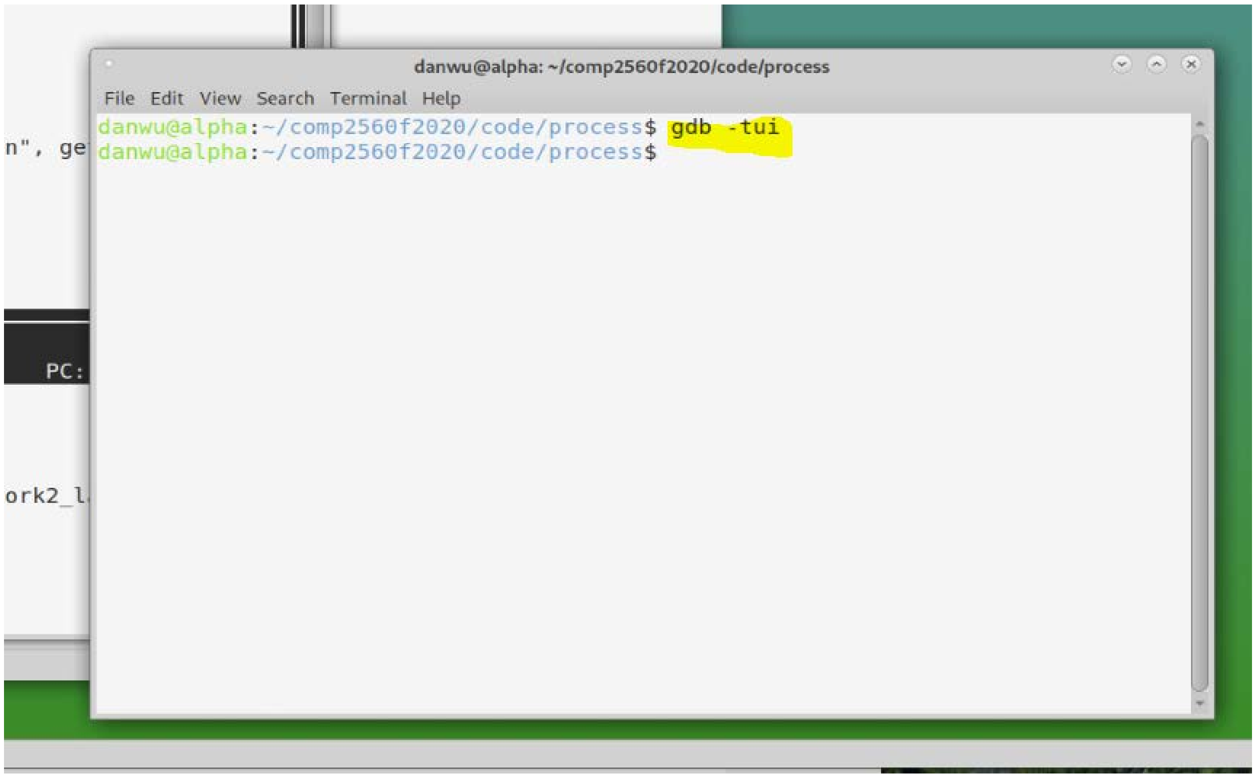


```
//fork2_lab.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
```

```
        int main(){
                if(pid == 0){ //child
                        = 88;
                        //glb++; var++;
                //intf("sleep(10);k\n");
                        sleep(60);
                        glb++; var++;
                }f( pid < 0 ){
                else{ // parent"fork");
B+>                     sleep(2);
                }

                printf("pid = %d, glob = %d, var = %d\n", getpid(), glb, va
                return 0;
        }               //glb++; var++;
                //

                                 ++;
                }
native process 16444 In:                                    L??   PC: ??
    <http://www.gnu.org/smainare/gdb/documentation/>.   L28   PC: 0x5555555551fa
Breakpoint 1 at 0x11fa: file fork2_lab.c, line 28.
(gdb) r
Starting program: /home/danwu/comp2560f2020/code/process/fork2_lab
Before fork
[Detaching after fork from child process 16448]

Breakpoint 1, main () at fork2_lab.c:28
(gdb) p pid
$1 = 16448
(gdb)
```

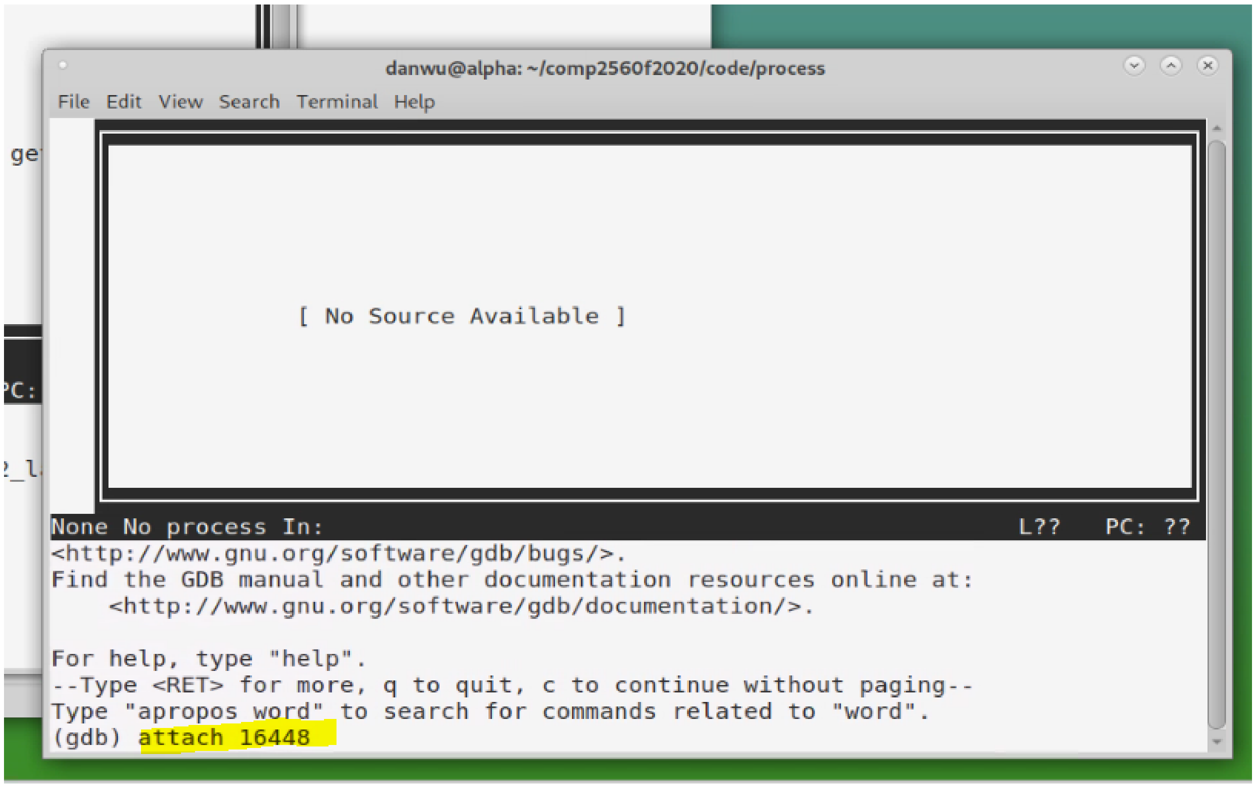
From the above screenshot, you know the pid for the child process is 16448.

4. In the other terminal window, run GDB (without ./a.out) as shown in the Figure below.


danwu@alpha: ~/comp2560f2020/code/process
File Edit View Search Terminal Help

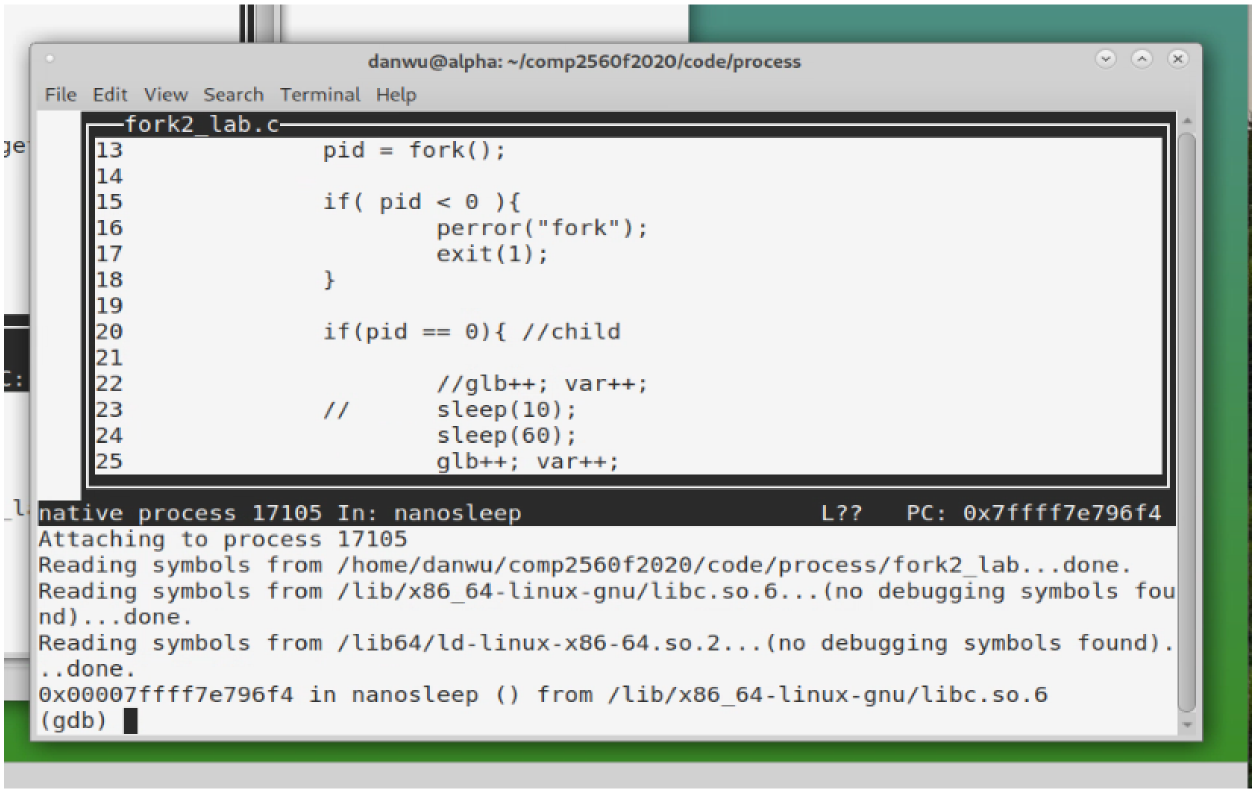
```
danwu@alpha:~/comp2560f2020/code/process$ gdb -tui
danwu@alpha:~/comp2560f2020/code/process$
```

Then type the GDB command “**attach 16448**”, shown below,


danwu@alpha: ~/comp2560f2020/code/process
File Edit View Search Terminal Help

```
                    [ No Source Available ]

None No process In:                                        L??   PC: ??
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
--Type <RET> for more, q to quit, c to continue without paging--
Type "apropos word" to search for commands related to "word".
(gdb) attach 16448
```

Then prese the down arrow, you will see the source code.


danwu@alpha: ~/comp2560f2020/code/process
File Edit View Search Terminal Help

```
fork2_lab.c
13              pid = fork();
14
15              if( pid < 0 ){
16                      perror("fork");
17                      exit(1);
18              }
19
20              if(pid == 0){ //child
21
22                      //glb++; var++;
23              //      sleep(10);
24                      sleep(60);
25                      glb++;  var++;
native process 17105 In: nanosleep                      L??   PC: 0x7ffff7e796f4
Attaching to process 17105
Reading symbols from /home/danwu/comp2560f2020/code/process/fork2_lab...done.
Reading symbols from /lib/x86_64-linux-gnu/libc.so.6...(no debugging symbols fou
nd)...done.
Reading symbols from /lib64/ld-linux-x86-64.so.2...(no debugging symbols found).
..done.
0x00007ffff7e796f4 in nanosleep () from /lib/x86_64-linux-gnu/libc.so.6
(gdb) 
```

5. Set a breakpoint at your desired statement in the child's code, for example, in the above screen shot, I could set it at line 25.

6. Now you can debug both processes using any command you learned from lab 2.

Part 2

0. If I change in the code for child process "sleep(60)" to "sleep (4)" and repeat the steps in Part 1, what would happen? (no submission for this question required)

1. Pick another simple program of your choice with fork () function and practice how to debug both the parent and child processes. You may have to insert a sleep function in the child process to keep it alive while you attach it to GDB and set up breakpoints. (no submission for this question required)

2. After you are familiar with the debugging process, screen recording a session with audio explaining how you debug/trace the program you chose in step 1 above.

3. Submit your video recording before Nov. 8, 11:59 PM.