

# **ISTANBUL TECHNICAL UNIVERSITY**



## **BLG 413E SYSTEM PROGRAMMING**

Ayşe Şima UYAR

### **Project 1**

#### **Group Members**

<b>Name</b>	<b>Surname</b>	<b>Number</b>
Emre	UYSAL	150160510
Cihat	BOSTANCI	150160542
Tugay	ACAR	150160511

## INTRODUCTION

In this project, it is expected to modify kernel functions according to given constraints. For modifying the functions, it is required to implement a task descriptor named as “myFlag”. As a group, we worked on the given prototype system call and according to practice session we implemented all requirements and started our project.

## FIRST STEP

In the first step, we downloaded the Linux-Source-3.13.0.tar.bz2 and extract it to make changes on it. After extracting it we created our directory “/set\_myFlag/” using “mkdir” command in it. In this directory we need to create our system call. So we implemented system call function with using the given template. We returned appropriate errors using <linux/errno.h> .

```
1 #include <linux/syscalls.h>
2 #include <linux/kernel.h>
3 #include <linux/errno.h>
4
5 asmlinkage long sys_set_myFlag(pid_t pid, int flag)
6 {
7     if(capable(CAP_SYS_ADMIN)) //needs root privilage
8     {
9         struct task_struct *process;
10        process = find_task_by_vpid(pid); //find given pid
11        if(process == NULL)
12        {
13            return ESRCH; //no such process
14        }
15        else
16        {
17            if(flag == 1 || flag == 0)
18            {
19                process->myFlag = flag; //set flag value into task_struct
20                return 0; //return success after setting the flag
21            }
22            else
23            {
24                return EINVAL; //invalid argument error
25            }
26        }
27    }
28    else
29    {
30        return EPERM; //operation not permitted error
31    }
32 }
```

After creating the system call we created Makefile. In the Makefile we wrote ;

```
obj-y := set_myFlag.o
```

After that we changed our directory to linux-source-3.13.0 folder and modified the Makefile in it.

```
# Objects we will link into vmlinux / subdirs we need to visit
init-y           := init/
drivers-y       := drivers/ sound/ firmware/ ubuntu/
net-y           := net/
libs-y          := lib/
core-y          := usr/ set_myFlag/
```

## SECOND STEP

In the second step, we created our variable in task\_struct for being capable to use it. The directory is include/linux/sched.h

```
1469 int myFlag;
```

After that we put it into init\_task in the include/linux/init\_task.h ;

```
218     .myFlag = 0, \
```

## THIRD STEP

In the third step, we put our prototype system call into include/linux/syscalls.h ;

---

```
852 asmlinkage long sys_set_myFlag(pid_t pid, int flag);
```

After that we put the required information into arch/x86/syscalls/syscall\_32.tbl for a proper compilation.

```
355 i386 set_myFlag sys_set_myFlag
```

## FOURTH STEP

In this step, we modified the fork system call. When we call the fork() from user space do\_exit() performs in the kernel space. So we modified the do\_exit() function. We used current macro for detecting the current process for forking. After making the controls in an if statement we returned ECHILD error if the flag is 1.

```
1651 struct task_struct *controller = current;
1652 if(controller->myFlag == 1 && task_nice(controller)>10)
1653 {
1654     return ECHILD; //returns error no child processes
1655 }
```

After that an unexpected thing occurred in our fork. So we solved it using the copy process function which creates child processes. In the copy process function we gave child flags 0 for creating an initialized flags in child processes.

```
1290 p->myFlag = 0; //set myFlag 0 to give childs a default flag parameter
```

## FIFTH STEP

In this step, we modified the exit.c in /kernel/ directory. We controlled our constraints with an if block and after that we looked for current process's parent's children list for detecting the siblings. After that we used list\_entry for making the initialization for task\_struct. With the sys\_kill we killed our siblings. All these steps until here in exit included if the flag is one and nice value is greater than ten. If the current process is not suitable to this constraints it makes a normal exit.

```
709     if(current->myFlag == 1 && task_nice(current)>10)
710     {
711         struct list_head *sibs_of_me;
712         struct task_struct *sib;
713         list_for_each(sibs_of_me, &current->parent->children)
714         {
715             sib = list_entry(sibs_of_me, struct task_struct, sibling);
716             sys_kill(sib->pid, SIGKILL);
717         }
718     }
```

## LAST STEP BEFORE COMPILE

We learned that, it is required to modify "arch/x86/boot/compressed/head\_32.S" for a successful compiling.

```
#ifdef CONFIG_EFI_STUB
    .data
efi32_config:
    .fill 11,8,0
    .long efi_call_phys
    .long 0
    .byte 0
#endif
```

After that we need to modify the grub file in our local linux in etc/default/ directory.

```
GRUB_DEFAULT=0
#GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""
```

## TESTING

We used the given tested methods below. Tests are also given below.

### *Error Checking*

- 1- Compile : gcc test3.c -o testError
- 2- Eperm Testing : ./testError <niceVal> <flag>
- 3- Esrch Testing: sudo ./testError <niceVal> <flag>
- 3.1 - Follow the guides on the test code please
- 4- Einval Testing : sudo ./testError <niceVal> <flag>

#### 4.1 - Flag must be given not 1 or 0

##### **Fork Checking**

- 1-Compile: gcc test3.c -o forkTesting
- 2-Normally fork : sudo ./forkTesting <11> <0>
- 3-No child process created (echild) : sudo ./forkTesting <11> <1>

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4 #include    <errno.h>
5 #include <string.h>
6
7 #define set_myFlag 355
8 int main(int argc, char** argv)
9 {
10     if(argc<2 || argc>3)
11     {
12         printf("Give exact two arguments first niceValue Second Flag\n");
13         return 0;
14     }
15
16     int niceValue=atoi(argv[1]);
17     int flag=atoi(argv[2]);
18
19     //try pid manuelly for getting ESRCH error uncomment these lines
20     //int pid;
21     //printf("RANDOM PID: ");
22     //scanf("%d",&pid);
23     //int ERR=syscall(set_myFlag,pid,flag);
24     //after that comment the other error part
25
26     int niceRet =nice(niceValue);
27     int ERR=syscall(set_myFlag,getpid(),flag);
28     if(ERR!=0){
29         printf("Error Control statement:%s\n",strerror(ERR));
30         return 0;
31     }
32     int f=fork();
33
34     if(f==0)
35     {
36         printf("CHild pid:%d ---- Parent pid:%d  niceValue:%d\n",getpid(),getppid(),niceRet);
37     }
38     else
39     {
40         printf("Error Control statement:%s \n",strerror(ERR));
41         printf("Fork return %s created\n",strerror(f));
42     }
43
44 }
45 sleep(1);
46 return 0;
```

##### **Exit Checking**

- 1-Compile: gcc exit\_test2.c -o exitTest
- 2-Compile: gcc flag\_setter.c -o controlExit
- 3-Run: ./exitTest
- 4-Normally Exit: sudo ./controlExit <0> <childpid>
- 5-Kill Siblings : sudo ./controlExit <1> <childpid>

```

9 int main(int argc, char ** argv)
10 {
11     int first = fork();
12     if(first == 0)
13     {
14         printf("Child = %d, parent = %d\n", getpid(),getppid());
15         int second = fork();
16         if(second == 0)
17         {
18             printf("Child %d , parent %d\n", getpid(),getppid());
19             while(1);
20         }
21     }
22     else
23     {
24         int retval = nice(11);
25         int third = fork();
26         if(third == 0)
27         {
28             printf("Child %d , parent %d\n", getpid(),getppid());
29             while(1);
30         }
31     }
32     else
33     {
34         int fifth = fork();
35         if(fifth == 0)
36         {
37             printf("Child %d , parent %d\n", getpid(),getppid());
38             while(1);
39         }
40         while(1);
41     }
42 }
43 else
44 {
45     printf("I am the main process  %d\n",getpid() );
46     while(1);
47 }
48 sleep(1);
49 return 0;
50 }
```

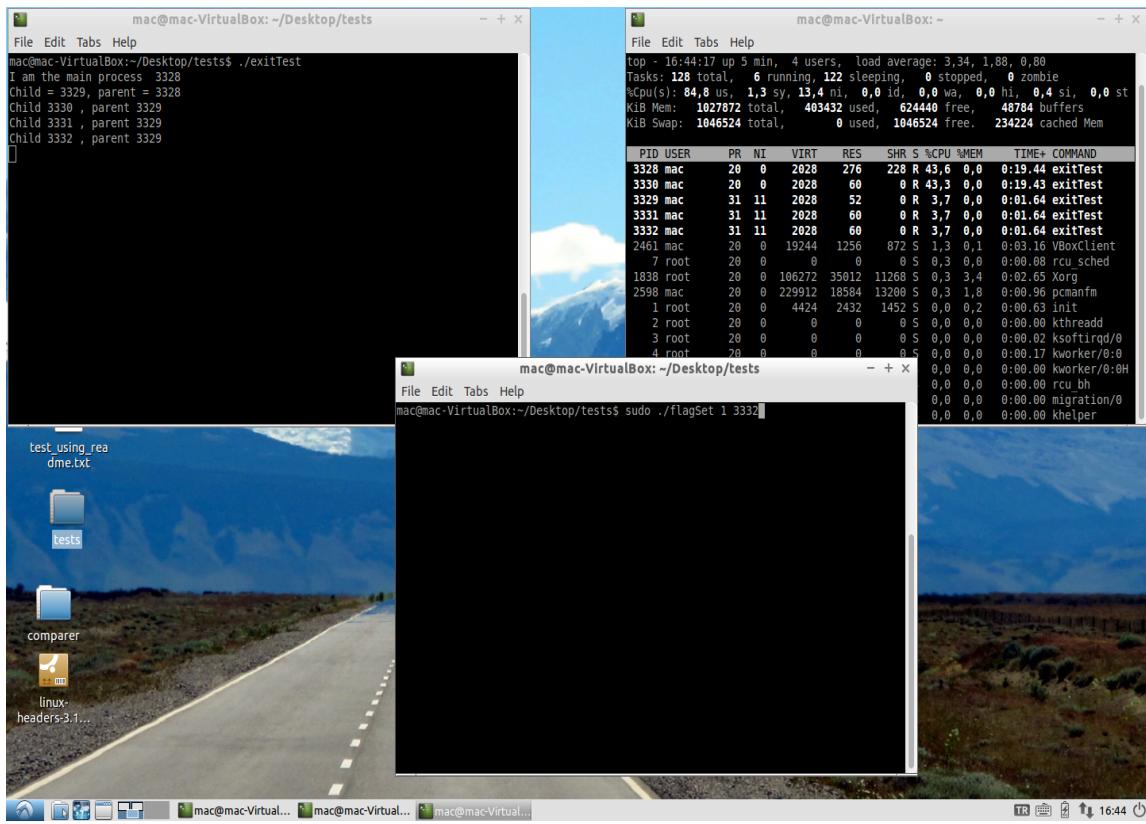
*exit\_test2.c*

```

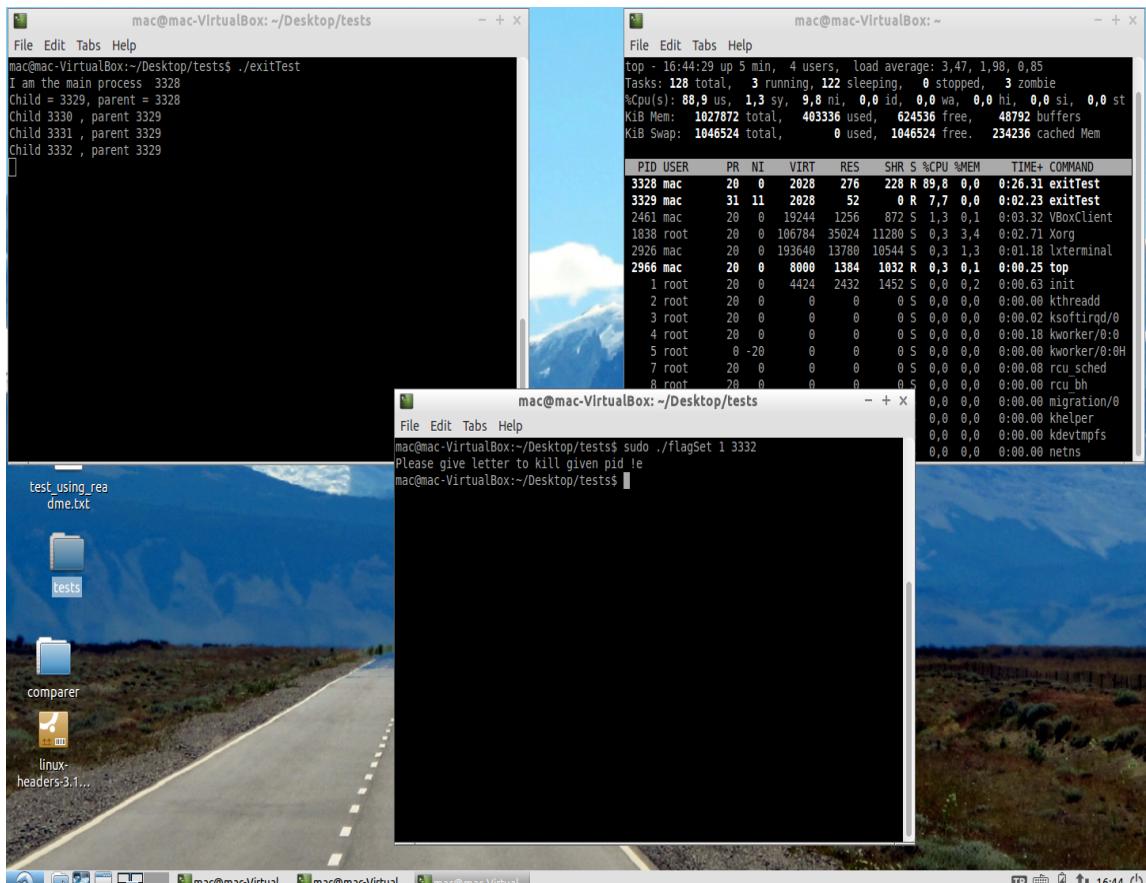
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4 #include <string.h>
5
6
7
8 #define set_flag 355
9
10 int main(int argc, char **argv)
11 {
12
13
14     int flag = atoi(argv[1]);
15     int pid = atoi(argv[2]);
16     int setter = syscall(set_flag, pid,flag);
17     if(setter !=0)
18     {
19         printf("error code : %s ",strerror(setter));
20     }
21
22     printf("Please give letter to kill given pid !");
23     getchar();
24     kill(pid,9);
25     return 0;
26
27 }
```

*flag\_setter.c*

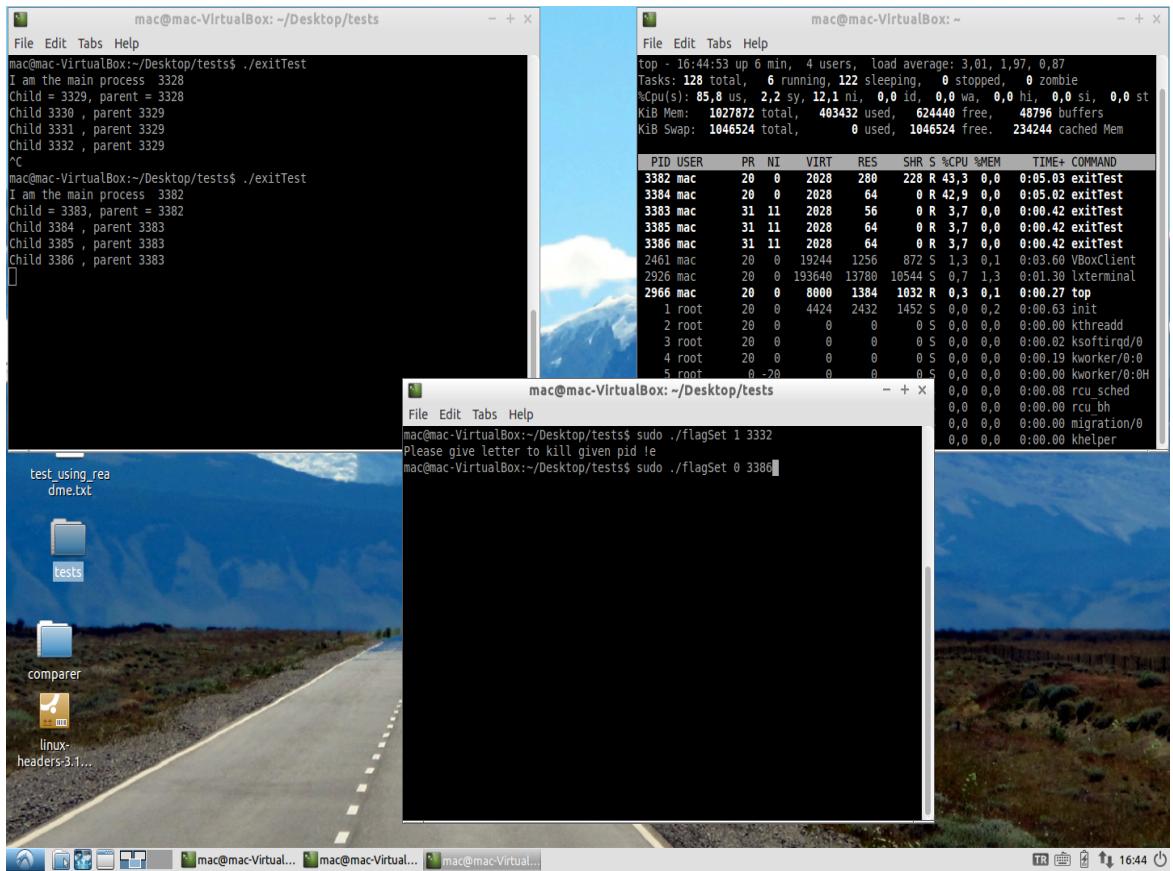
## TEST RESULTS AS SCREENSHOTS



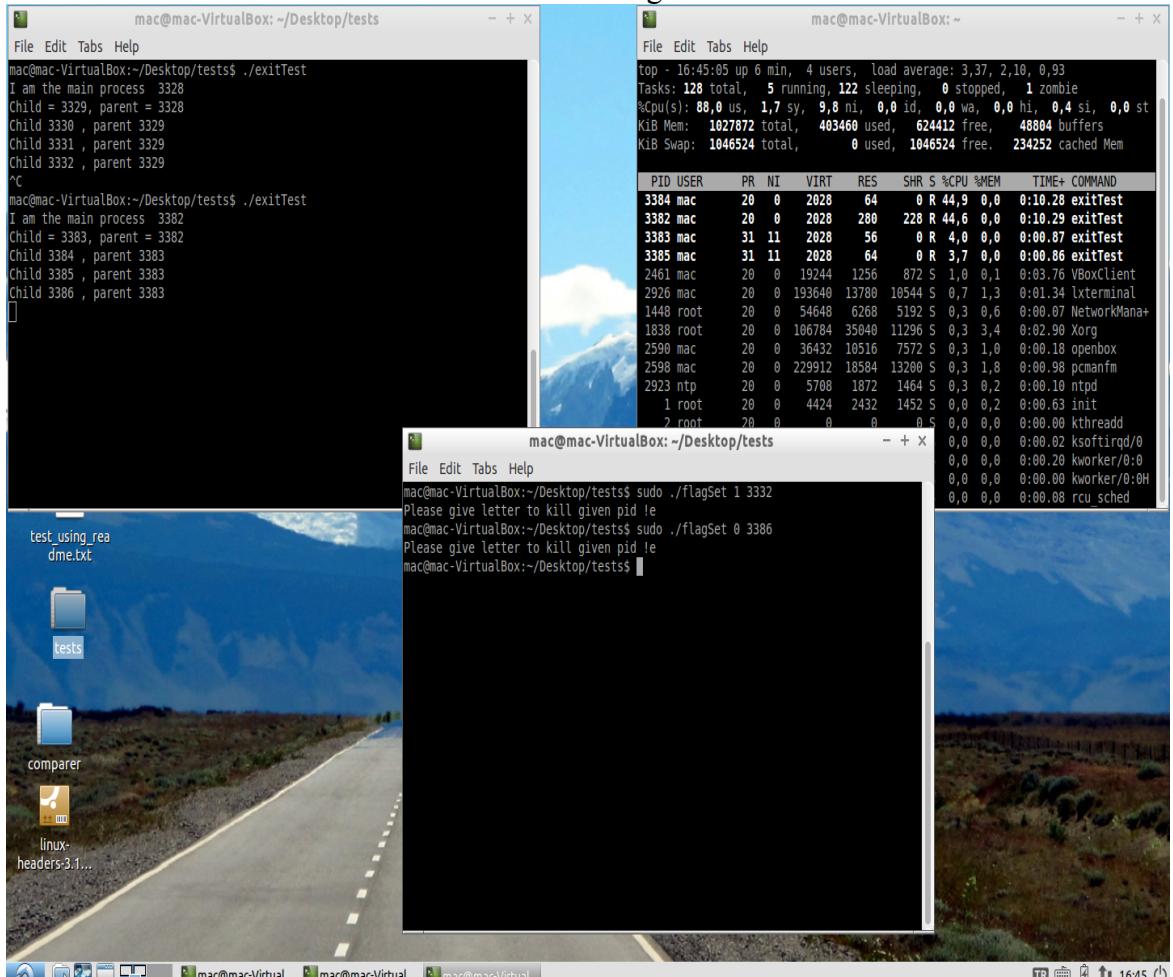
Exit test with flag 1



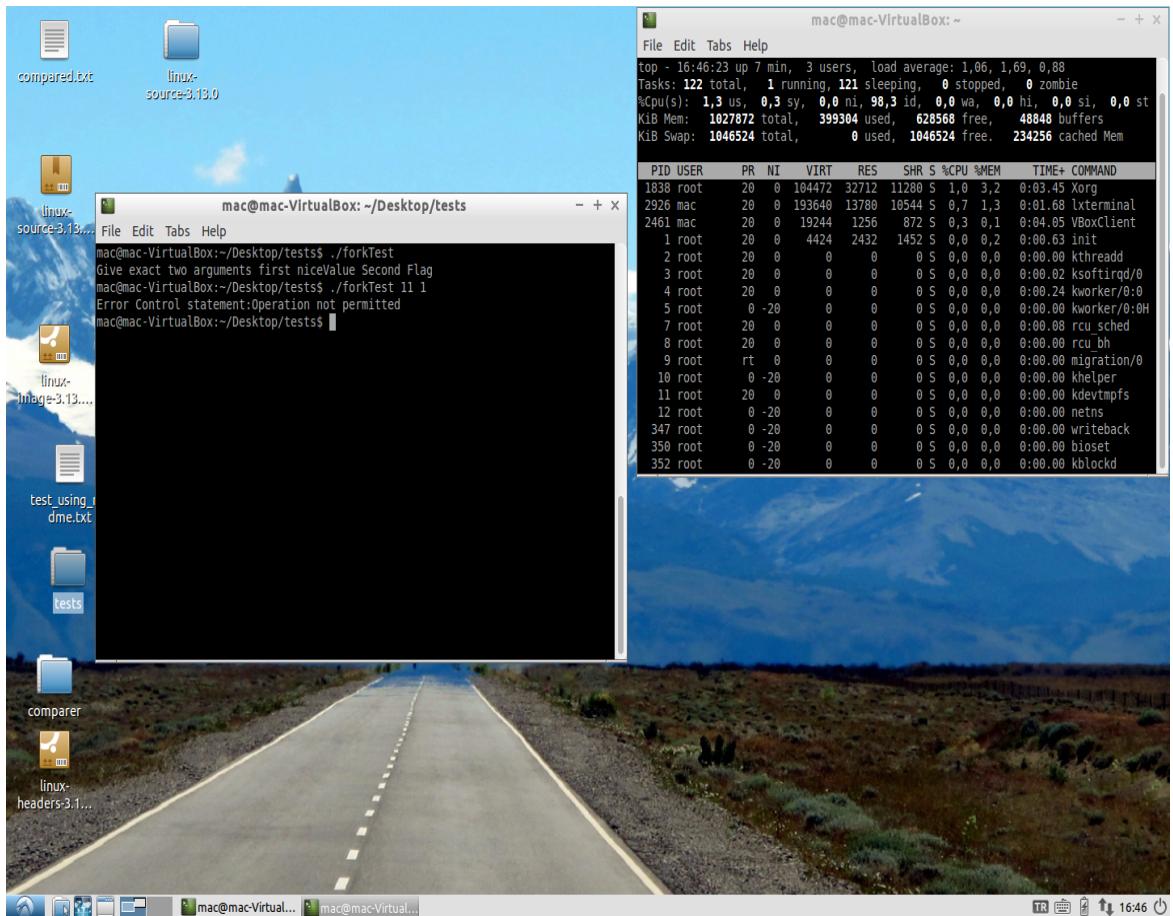
All siblings died



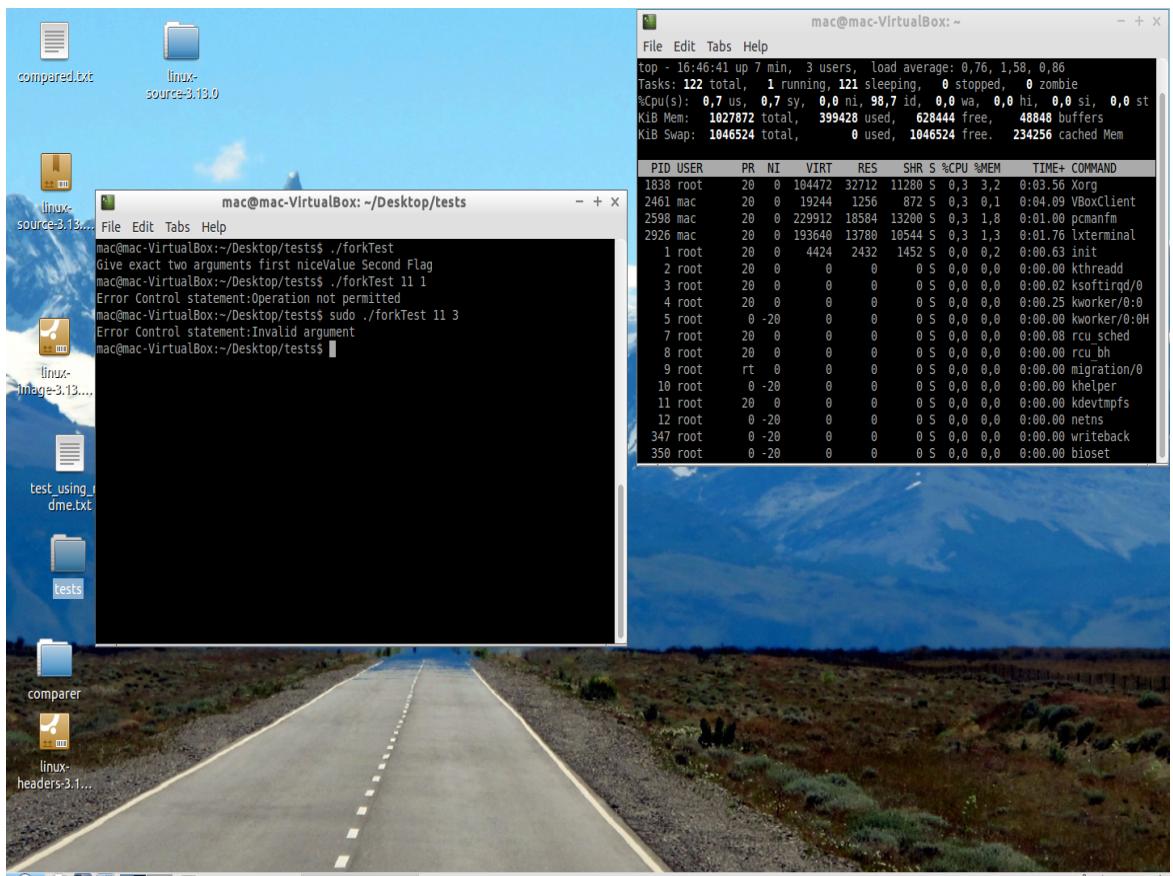
### Exit test with flag 0



Just the given pid died



Eperm error test



Einval error test

```

File Edit Search Options Help
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>

#define set_myFlag 355
int main(int argc, char** argv)
{
    if(argc<2 || argc>3)
    {
        printf("Give exact two arguments\n");
        return 0;
    }

    int niceValue=atoi(argv[1]);
    int flag=atoi(argv[2]);

    //try pid manually for getting ESRCH
    int pid;
    printf("RANDOM PID: ");
    scanf("%d",&pid);
    int ERR=syscall(set_myFlag,pid,flag);
    //After that comment the other error part
    niceRet;

    niceRet=nice(niceValue);
    //int ERR=syscall(set_myFlag,getpid())
    if(ERR!=0)
    {
        printf("Error Control statement:\n");
        return 0;
    }
    int f=fork();
    if(f==0)
    {
        printf("Child pid:%d --- Parent pid:%d niceValue:%d\n",getpid(),getppid(),niceRet);
    }
    else
    {
        printf("Error Control statement:%s\n",strerror(ERR));
        printf("Fork return %s created\n",strerror(f));
    }
    sleep(1);
    return 0;
}

```

```

mac@mac-VirtualBox: ~/Desktop/tests
File Edit Tabs Help
mac@mac-VirtualBox:~/Desktop$ sudo ./forkTest 11 0
Error Control statement:Success
Fork return Unknown error 3478 created
Child pid:3478 -- Parent pid:3477 niceValue:11
mac@mac-VirtualBox:~/Desktop/tests$ gcc test3.c -o forkTest
test3.c: In function 'main':
test3.c:24:2: error: unknown type name 'after'
    after that comment the other error part
^
test3.c:24:13: error: expected '=', ',', ';', 'asm' or '__attribute__' before 'comment'
    after that comment the other error part
^
test3.c:36:79: error: 'niceRet' undeclared (first use in this function)
    printf("Child pid:%d ---- Parent pid:%d niceValue:%d\n",getpid(),getppid(),niceRet);

test3.c:36:79: note: each undeclared identifier is reported only once for each function it appears in
mac@mac-VirtualBox:~/Desktop/tests$ gcc test3.c -o forkTest
mac@mac-VirtualBox:~/Desktop/tests$ sudo ./forkTest 11 0
RANDOM PID: 123123123
Error Control statement:No such process
mac@mac-VirtualBox:~/Desktop/tests$ 

```

```

mac@mac-VirtualBox: ~
File Edit Tabs Help
top - 16:55:57 up 17 min, 3 users, load average: 0.00, 0.25, 0.48
Tasks: 122 total, 1 running, 121 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.0 us, 0.3 sy, 0.0 ni, 98.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 1027872 total, 403136 used, 624736 free, 49276 buffers
KiB Swap: 1046524 total, 0 used, 1046524 free, 234480 cached Mem

```

N	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
0	105668	33184	11444	S	0,3	3,2	0:06.76	Xorg
0	37264	11040	7572	S	0,3	1,1	0:00.31	openbox
0	193640	13780	10544	S	0,3	1,3	0:02.81	xterm
<b>0</b>	<b>8000</b>	<b>1384</b>	<b>1032</b>	<b>R</b>	<b>0,3</b>	<b>0,1</b>	<b>0:00.88</b>	<b>top</b>
0	4424	2432	1452	S	0,0	0,2	0:00.63	init
0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
0	0	0	0	S	0,0	0,0	0:00.54	kworker/0:0
0	0	0	0	S	0,0	0,0	0:00.00	kworker/0:0H
0	0	0	0	S	0,0	0,0	0:00.10	rcu_sched
0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh
0	0	0	0	S	0,0	0,0	0:00.00	migration/0
0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh
0	0	0	0	S	0,0	0,0	0:00.00	helper
0	0	0	0	S	0,0	0,0	0:00.00	kdevtmpfs
0	0	0	0	S	0,0	0,0	0:00.00	netns
0	0	0	0	S	0,0	0,0	0:00.00	writeback
0	0	0	0	S	0,0	0,0	0:00.00	bioset

Esrch error test

```

mac@mac-VirtualBox: ~/Desktop/tests
File Edit Tabs Help
mac@mac-VirtualBox:~/Desktop$ ./forkTest
Give exact two arguments first niceValue Second Flag
mac@mac-VirtualBox:~/Desktop$ ./forkTest 11 1
Error Control statement:Operation not permitted
mac@mac-VirtualBox:~/Desktop$ sudo ./forkTest 11 3
Error Control statement:Invalid argument
mac@mac-VirtualBox:~/Desktop$ sudo ./forkTest 11 1
Error Control statement:Success
Fork return No child processes created
mac@mac-VirtualBox:~/Desktop$ 

```

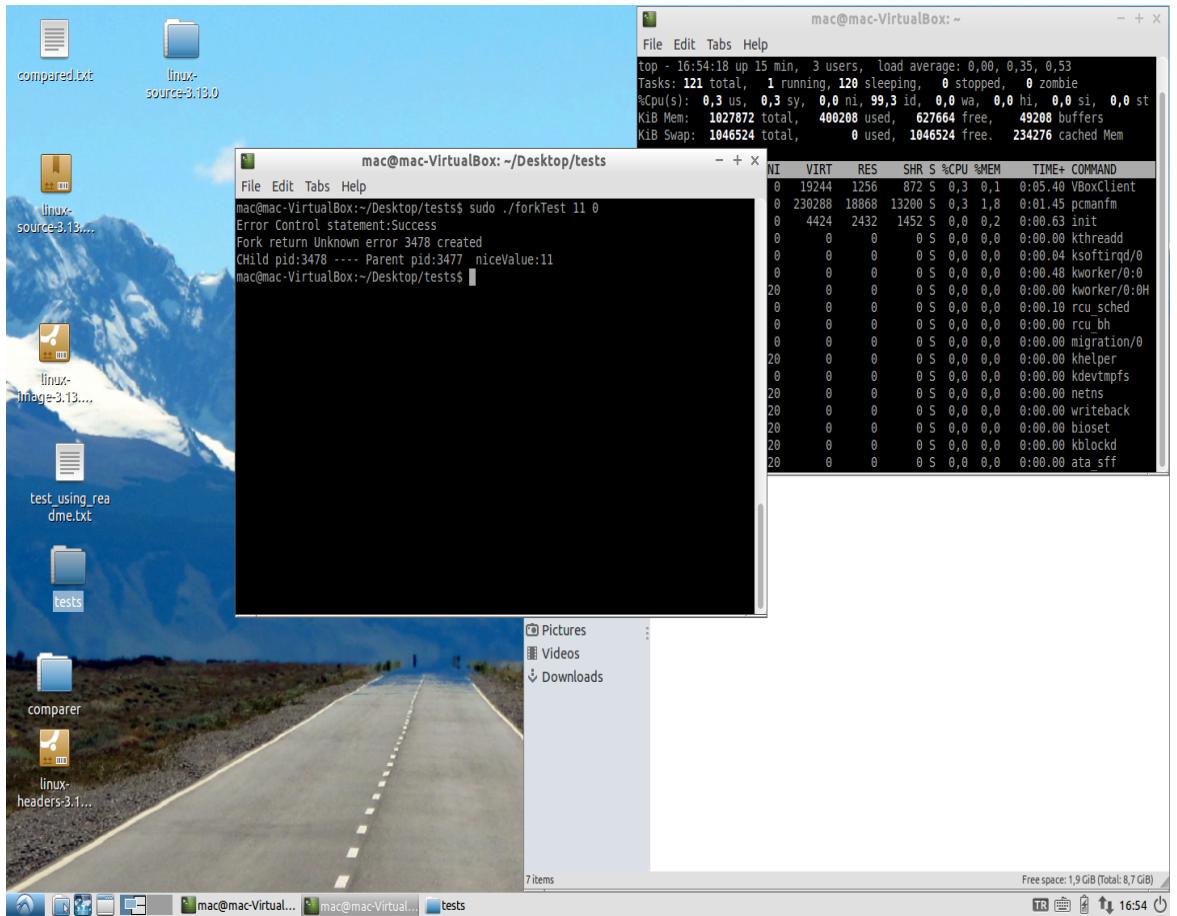
```

mac@mac-VirtualBox: ~
File Edit Tabs Help
top - 16:47:02 up 8 min, 3 users, load average: 0.54, 1.48, 0.84
Tasks: 122 total, 1 running, 121 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.0 us, 0.0 sy, 0.0 ni, 98.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 1027872 total, 399400 used, 628472 free, 48864 buffers
KiB Swap: 1046524 total, 0 used, 1046524 free, 234256 cached Mem

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1838	root	20	0	104472	32712	11280	S	1,0	3,2	0:03.61	Xorg
2926	mac	20	0	193640	13780	10544	S	0,7	1,3	0:01.80	xterm
<b>2966</b>	<b>mac</b>	<b>20</b>	<b>0</b>	<b>8000</b>	<b>1384</b>	<b>1032</b>	<b>R</b>	<b>0,3</b>	<b>0,1</b>	<b>0:00.39</b>	<b>top</b>
1	root	20	0	4424	2432	1452	S	0,0	0,2	0:00.63	init
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.02	ksoftirqd/0
4	root	20	0	0	0	0	S	0,0	0,0	0:00.26	kworker/0:0
5	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/0:0H
7	root	20	0	0	0	0	S	0,0	0,0	0:00.08	rcu_sched
8	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	migration/0
10	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	helper
11	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kdevtmpfs
12	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	netns
347	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	writeback
350	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	bioset
352	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kblockd

Fork test with flag 1(echild error test)



Fork test with flag 0