Project Submitted by – Rahul Pareek.

Project 2 – Operating System

1. Added an entry in table syscall\_64.tbl location –

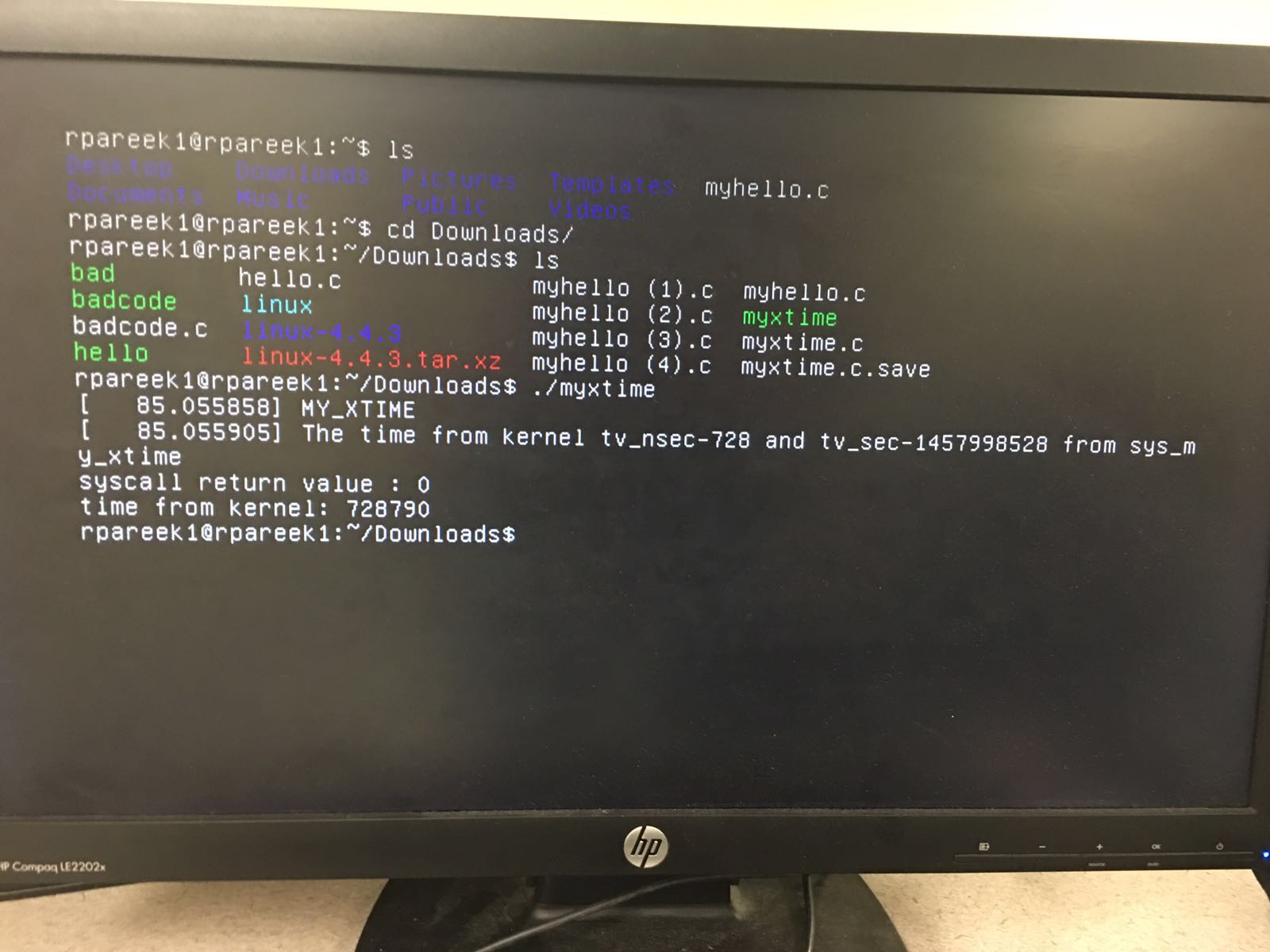
Downloads/linux-4.4.3/arch/x86/syscalls/syscall\_64.tbl

547 common my\_xtime sys\_my\_xtime

1. Added the syscall definition in the file my\_xtime.c in the kernel/
2. #include<linux/init.h>
3. #include<linux/module.h>
4. #include<linux/linkage.h>
5. #include<linux/export.h>
6. #include<linux/time.h>
7. #include<asm/uaccess.h>
8. #include<linux/printk.h>
9. #include<linux/slab.h>
10. #include<linux/errno.h>
11. asmlinkage int sys\_my\_xtime(struct timespec \*current\_time){
12. printk(KERN\_ALERT "MY\_XTIME\n");
13. if(!access\_ok(VERIFY\_WRITE, current\_time, sizeof(struct timespec)))
14. {
15. return -EFAULT;
16. }
18. struct timespec time =current\_kernel\_time();
19. //clock\_gettime(CLOCK\_REALTIME\_COARSE,time);
20. //do\_gettimeofday(time);

23. printk(KERN\_ALERT "The time from kernel %ld from sys\_my\_xtime\n",time.tv\_nsec);
24. if(copy\_to\_user(current\_time,&time,sizeof(struct timespec)))
25. return -EFAULT;
26. return 0;
27. }
28. EXPORT\_SYMBOL(sys\_my\_xtime);
29. Update the makefile. make an entry for .o file.
30. Write a program to call your system call.
31. #include<stdio.h>
32. #include<unistd.h>
33. #include<linux/unistd.h>
34. #include<linux/time.h>
35. int main(){
36. struct timespec tm;
37. int y;
38. y=syscall(547,&tm);
39. printf("syscall return value : %d\n",y);
40. printf("time from kernel: %d\n",tm.tv\_nsec);
41. return 0;
42. }

Observation:- the system call gets called and current time is printed from system call and struct is copied to user.



We have reused the same system call for all the badcodes.

Bad Code 1. Badcode1.c

#include<linux/init.h>

#include<linux/module.h>

#include<linux/linkage.h>

#include<linux/export.h>

#include<linux/time.h>

#include<asm/uaccess.h>

#include<linux/printk.h>

#include<linux/slab.h>

asmlinkage int sys\_myhello(void){

printk(KERN\_ALERT "hello world: myhello\n");

int a=100,i=0,c=0;

c=a/i;

printk(KERN\_ALERT "this should not print %d\n",c);

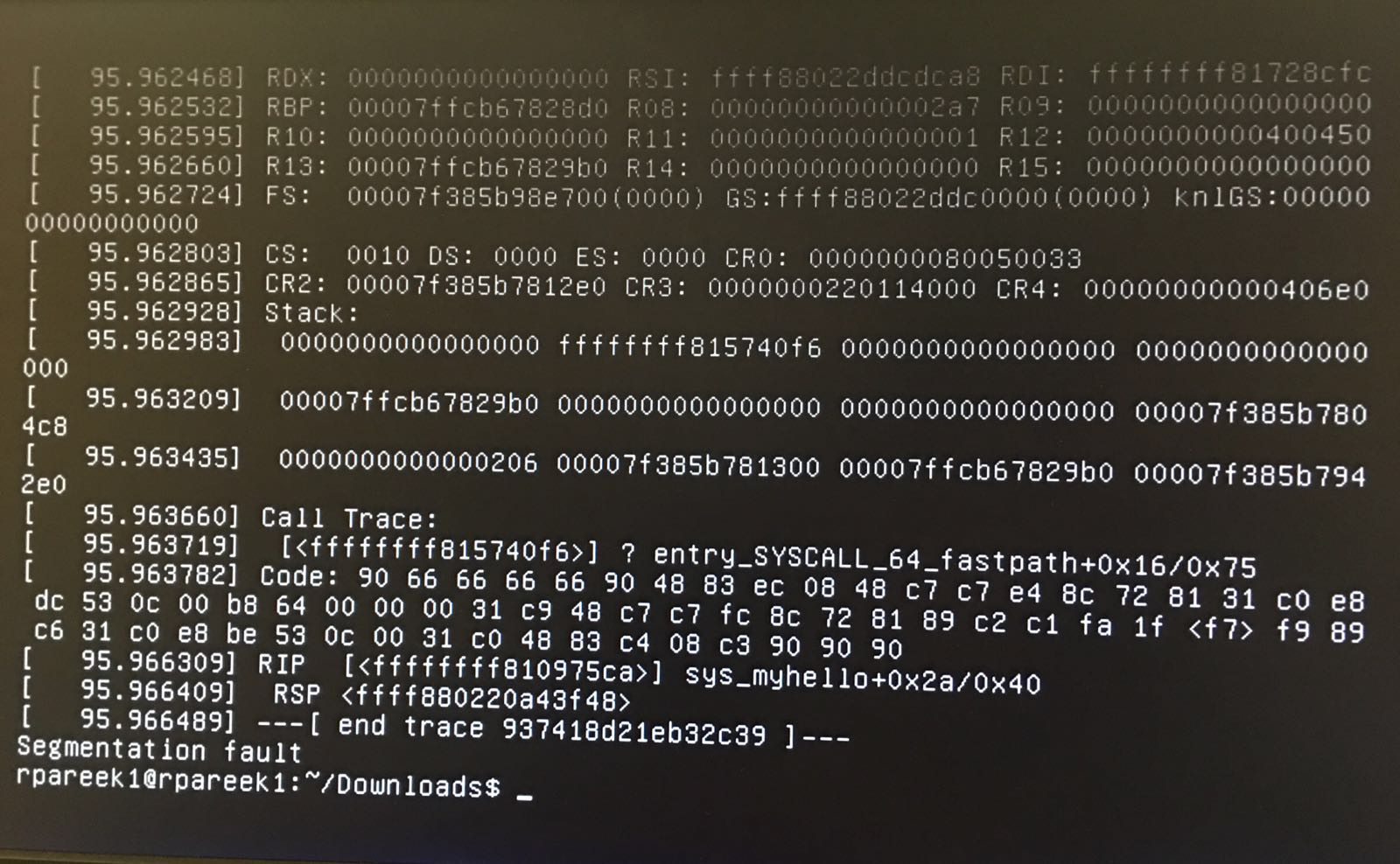
return 0;

}

EXPORT\_SYMBOL(sys\_myhello);

Observation :- bad code for divide by zero gives the seg fault

Attaching the screenshot.



Badcode2.c

#include<linux/init.h>

#include<linux/module.h>

#include<linux/linkage.h>

#include<linux/export.h>

#include<linux/time.h>

#include<asm/uaccess.h>

#include<linux/printk.h>

#include<linux/slab.h>

asmlinkage int sys\_myhello(void){

printk(KERN\_ALERT "hello world: myhello\n");

int \* ptr=5;

printk(KERN\_ALERT "this should not print %d\n",\*ptr);

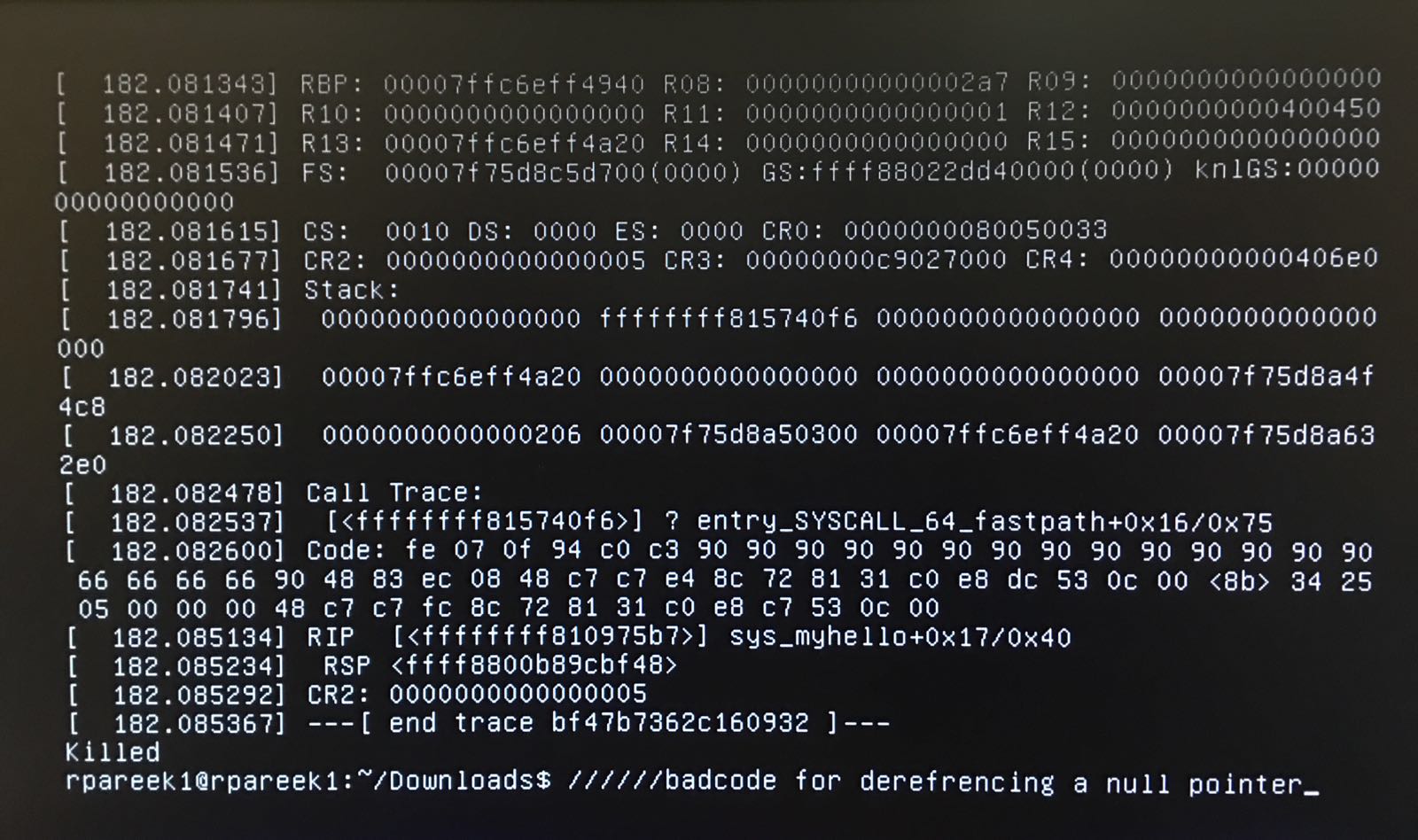
return 0;

}

EXPORT\_SYMBOL(sys\_myhello);

Observation :- bad code for dereferencing a pointer gets killed by kernel.

Attaching the screenshot.



Badcode3.c for not returning a value from system call

#include<linux/init.h>

#include<linux/module.h>

#include<linux/linkage.h>

#include<linux/export.h>

#include<linux/time.h>

#include<asm/uaccess.h>

#include<linux/printk.h>

#include<linux/slab.h>

asmlinkage int sys\_myhello(void){

printk(KERN\_ALERT "hello world: myhello\n");

}

EXPORT\_SYMBOL(sys\_myhello);

Observation:-

The code works fine.

Badcode4.c using a user lib function

#include<linux/init.h>

#include<linux/module.h>

#include<linux/linkage.h>

#include<linux/export.h>

#include<linux/time.h>

#include<asm/uaccess.h>

#include<linux/printk.h>

#include<linux/slab.h>

asmlinkage int sys\_myhello(void){

printk(KERN\_ALERT "hello world: myhello\n");

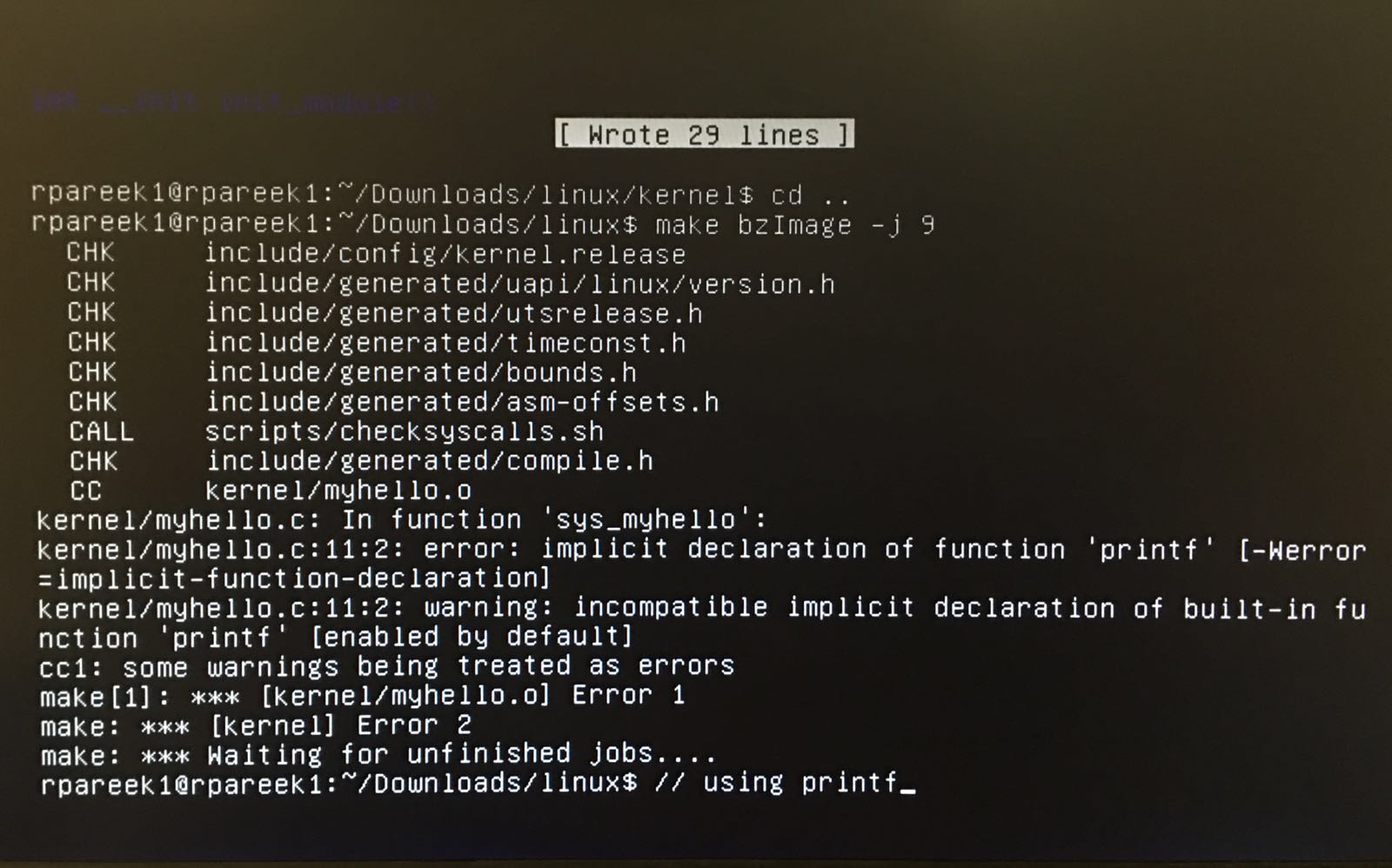
printf("this should not print\n");

return 0;

}

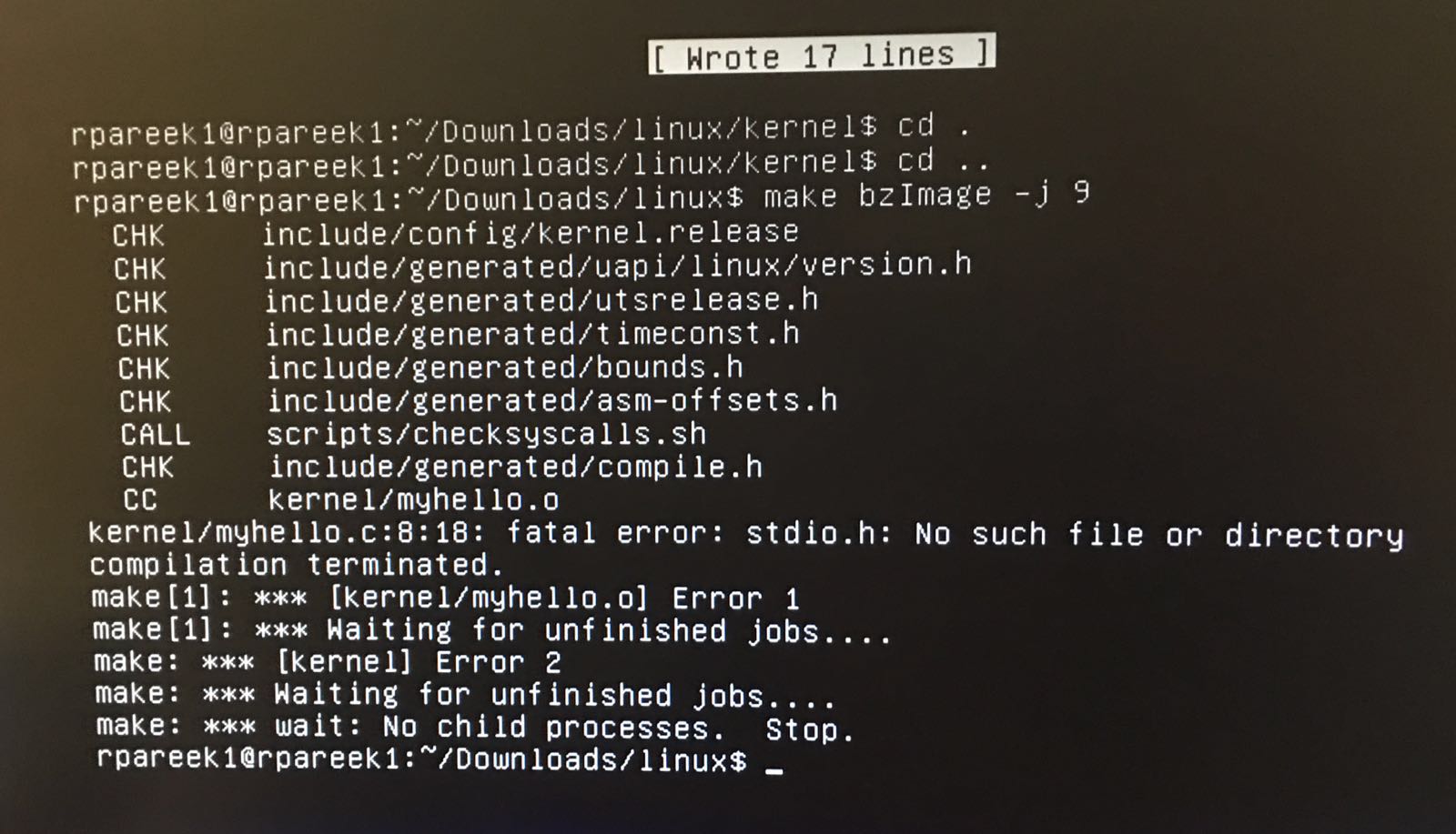
EXPORT\_SYMBOL(sys\_myhello);

Obeservation:- the code does not compile.



Own badcode:-

Used standard library-



Observation: -

Fatal error.

Badcode6.c

Reursinve Called system call from the system call.

#include<linux/init.h>

#include<linux/module.h>

#include<linux/linkage.h>

#include<linux/export.h>

#include<linux/time.h>

#include<asm/uaccess.h>

#include<linux/printk.h>

#include<linux/slab.h>

asmlinkage int sys\_myhello(void){

printk(KERN\_ALERT "hello world: myhello\n");

//again new infinte

sys\_myhello();

return 0;

}

EXPORT\_SYMBOL(sys\_myhello);

Observation:- kernel panic – not syncing.

