A search function to find all the children of a process

We need to find all the children that have the same parent

Once we find the children, we know the parents’

Once we know the parent , we can find the sibling

long cs3013 syscall2(unsigned short \*target pid, struct ancestry \*response)

e \*target pid is a pointer to an unsigned short and response is a

pointer to an ancestor struct, as defined below.

The kernel must print a message about the target

process, the siblings, children, and ancestors in the system log, indicating what relationship the process has

to the targeted process.

struct ancestry {

pid\_t ancestors[10];

pid\_t siblings[100];

pid\_t children[100];

};

So we will need a way to print this

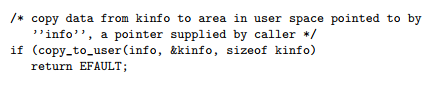
information you need can be found in the structure called task struct, defined in

include/linux/sched.h in the kernel source tree

check the validity of th These functions are copy from user and copy to user, and they are defined

in include/asm-generic/uaccess.h. You will need to use both. See pp. 76-77 in Linux Kernel

Development, 3rd edition.e arguments passed by the caller.



EFAULT is a error code defined in include/asm/errno.h.

You need to intercept the cs3013 syscall2 system call, as you did for cs3013 syscall1 in Part 1 of this project.

Hint: Start out your cs3013 syscall2 system call with getting just a few pieces of information, so as to make sure that you can return it to the caller. After you get this part working, add the functionality to access the task structs of the parent, child, and siblings. Remember that the children and siblings are in linked lists that need to be accessed using the Linux kernel list macros described in Chapter 6 of the Love text.

Step 1: print uid and process id and print it

Step 2: I think parent pid is simple

Step 3: children PID

Step 4: Sibling

Step 5: print it all out