For this program, it user global variables to pass the variable informant between different files inside the kernel. Following are the functions and prompt that user need to enter to get the two list and compare them. Page 5 have an image of overall design of the system.

## Userland function:

- Set functions that call each of the syscall functions listed in kernel folder
- Compare function that compare the two list
- Initial list_number = 0;
- main function:
    1. asking user to run pa aux in command line to get the PID that user wanted to check for. If user forget to run pa aux before use the program. User can get PID by open the system monition program to get the PID.
    2. while loop that run twice to get two list of syscall number
    3. in while loop:
        1. prompt user to enter PID
        2. scan the pid and store in process_id
        3. sent process_id to kernel (later use to compare with kernel process ID so kernel know which procees id's syscall number to log into list.)
        4. Asking user to enter 1 to start log syscall number to list
        5. Store 1 in allow and sent to kernel
        6. System start recording syscall number
        7. Syscall number is store in list

8. Asking user to enter 0 to stop recording
9. Store allow = 0 and sent to kennel to stop the record
10. Get syscall number list from kernel and save as list1 or list2 dependent of list_number.
    - List_number = 0, save to list 1
    - List_number = 1, save to list 2
11. Get the number of system from kernel and save as count
12. Compare if list_number is greater than or equal to 2
    - If yes, end while loop
    - If no, increase list_number by 1 and back to step 1 to record list 2

4. Compare list 1 and list 2
    1. Enter k to see how many you want to compare
    2. Start from end of the list can go back k times
        - Each time compare if they are the same
            - Same = 1
            - Different = 0
                - Error + 1
    3. Compare to see if error is more then 10% of total number of syscall list.
        - If yes, display message “system is attack”
        - If no, display message “system is safe”

## Kernel function:

- Void Sys_set_PID(long PID);

    - set the process id from userland to compare to the kernel to lock with system call number get to store in list
- Long Sys_get_PID(void);
    - let userland to get the PID from kernel
- Void sys_set_syscall_list(void)
    - Set up the list for log syscall number
- Long * sys_get_syscall_list(void)
    - Sent the list back to userland
- Sys_set_allow(long bool)
    - Set when should information get log into list
        - 1 start log information to list
        - 0 is stop log information to list
- Sys_get_allow(void)
    - Sent allow number to your land
- Sys_get_count_process(void)
    - Sent number of syscall number to userland
- Initi_read()
    - Read the file in kernel from userland
- Initi_write()
    - Write the list to kernel file, later can be read out by init_read() function by userland

## Entry_64.S

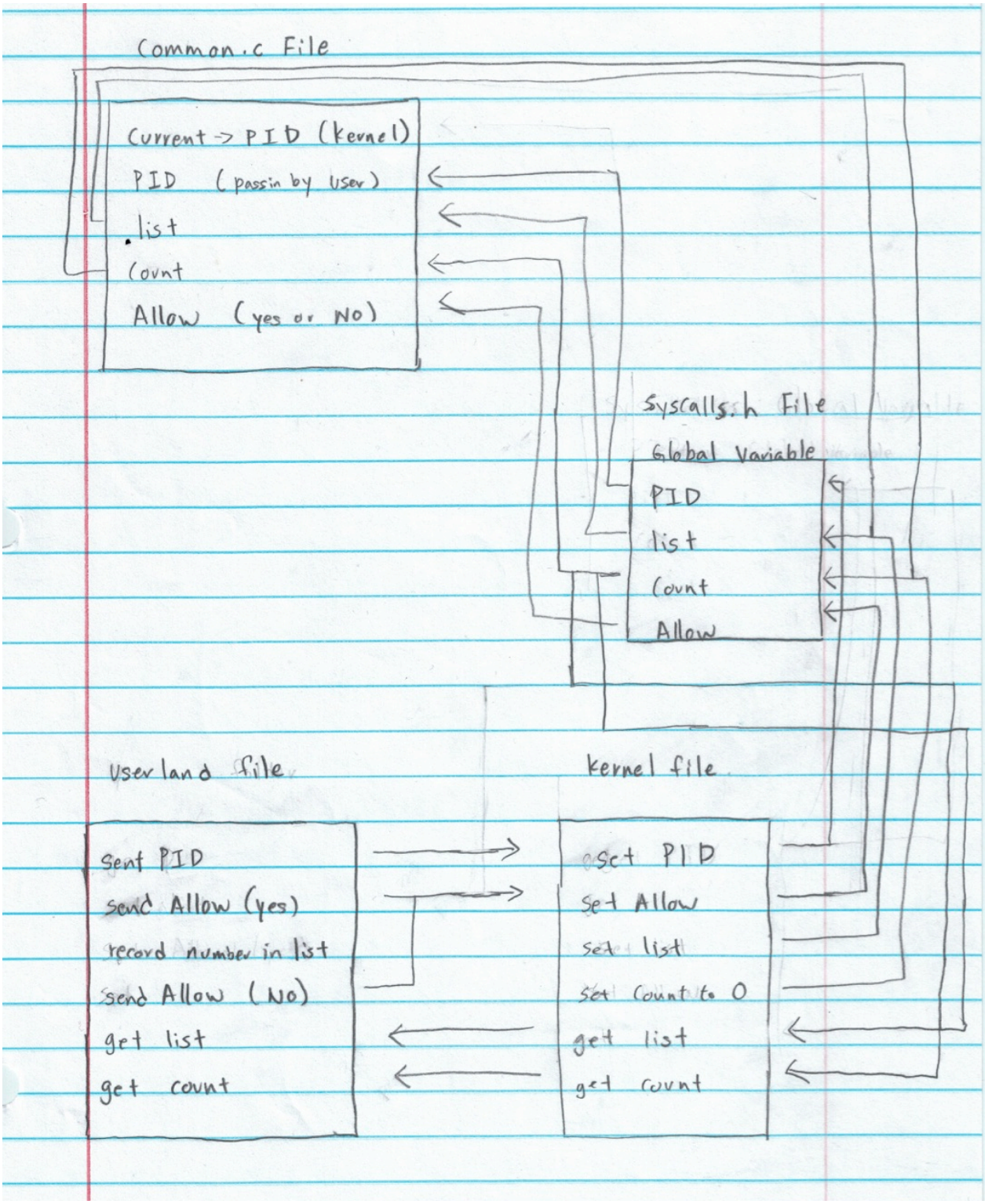
- Change line 250 from jnz to jmp (condition to unconditioned)
- it will just to do_syscall_64 every time it gets called instead of compare it before jump.

## Common.c file

- If statement
    - Compare if allow is equal to 1
    - Compare if current pid is equal to pid pass in by userland
    - If both about statement is true
    - Syscall number is add to list
    - Count_process is increase by 1

## Syscalls.h file Global variable

- Variable that can be usedin any file that include syscalls.h
- Extern long processID
    - Process id that is pass in by userland
- Extern long * syscall_list
    - List that store all the syscall number
- Extern long allows
    - Set 1 on to store record syscall number to list
    - Set 2 to stop record syscall number to list
- Extern long count_process
    - Number of syscall number in the list


Common.c File
Current -> PID (kernel)
PID (passin by User)
list
count
Allow (yes or NO)
Syscalls.h File
Global Variable
PID
list
count
Allow
Userland file
Kernel file
Sent PID
send Allow (yes)
record number in list
send Allow (NO)
get list
get count
Set PID
Set Allow
set list
Set count to 0
get list
get count