Lab1Answers

Haoran Lei

3.1

In the original distributed version, the system 'idles' by running infinite while-loop and nulluser() is at priority 0 which means there is always one process that is waiting. The alternative way is to call halt() in the loop and the system will wait until interrupts happen.

4.1

C type of system call is int32

5.3

in main(), the address of the top of the run-time stack before myProgA() is create/resumed is 0x efc8fd0

in main(), the content of the top of the run-time stack before myProgA() is create/resumed is 0x 0

the address of the top of the run-time stack after myprogA() is created but before myfuncA() is called inside myprogA()is 0x fdeffdc

the content of the top of the run-time stack after myprogA() is created but before myfuncA() is called inside myprogA()is 0x 0

in myfuncA() the address of the top of the run-time stack after myfuncA() is called is 0x fdeffac

in myfuncA() the content of the top of the run-time stack after myfuncA() is called is 0x 0

in main(), the address of the top of the run-time stack after myProgA() is create/resumed is 0x efc8fd0

in main(), the content of the top of the run-time stack after myProgA() is create/resumed is 0x 0

Description: We could find that the top of stack address and content are the same before and after myProA() is created and resumed. The owner of the stack could be seen from the above comments.

6. The strategy to find the return address of myProgA to the last function call is as follows:

The return address is the content of the stack frame 4 bytes above ebp of the current process stack of myProgA. We can find it out by printing. After calling myfunA, the return address of myfuncA is also in the stack frame 4 bytes above ebp of the current process stack. Since the we found that the two stack frames containing the returning address of two different functions are 0x00000400 away from each other due to the already set space allocation of function call in xinu, we can find the return address of myProgA from the ebp of myfuncA and overwrite it with the address of malware().