Design of countTraps()

proc.h

Two new arrays are added into struct proc (per-process state). One array is called *countSyscall*, which contains 22 elements and each element in the array is used to record the numbers of each system call. Another array is called *countTraps*, which contains 20 elements and each element in the array is used to record the numbers of traps that system calls get into.

trap.c

Function *check_trap_func()* takes *struct trapframe* tf* as its input.

If the type of the trap(*tf -> trapno*) is a system call, *tf -> eax* is used to specify the type of the system call using *check_trap_func()* and increase the values of the corresponding element in array *countSyscall(myproc->countSyscall[i])*. If the type of the trap(*tf -> trapno*) is not a system call, then other actions are performed

proc.c

Static arrays *syscallNameCode* and *trapNameCode* are used to store the names of system calls and other types of traps (including exceptions and interruptions) respectively. Function *countTraps()* is implemented at the end of this file. Firstly, the arrays *countSyscall* and *countTraps* are initialized in the function *allocproc()* (call function *assignArr()*), with every element in these two arrays being assigned to 0. System call *exit()* checks if the parent of the current process is not equal to the initproc. If the parent of the current process is not equal to the initproc, then the respective trap of the child process is added into its parent's trap.

In function *countTraps()*, *myproc()* is used to get the user process. The nested for-loops are used to calculate both the numbers of system calls and the total number of all traps. They are also used to print out traps and each trap's name and amount.