Changed code:

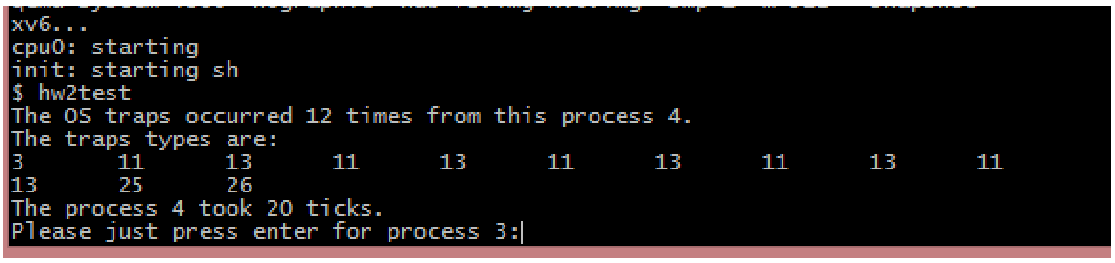
1. Makefile: I add the path “_hw2test” in UPROGS.
2. proc.h: I add some variables in structure proc:
    - start_ticks -- Save the tick time of the start of this process;
    - next_start_tick -- Save the start time when the process return from system call.
    - total_ticks -- Save the total tick time up to now.
    - trap_count -- Save the count times of trap occurrence from this process.
    - int trap_type[100] -- Save the types of top100 occured traps. (just used for test)
3. proc.c: Add code to initial the new variables above, both in allocproc function and exit function.
4. sysproc.c: I add four system call fucntions:
    - sys_sum_ticks: Sum the tick time for process up to now, by adding the tick time in this period. (In order to not contain the trap time) In addition, I use sys_uptime() to get the current tick time.
    - sys_init_start: Initial the start the time when trap finished and the process continues.
    - sys_print_ticks: Print the total tick time of the process.
    - sys_print_trapscount: Print the count number of traps from the process.
    - sys_print_trapstype: Print the types of Top 100 traps from the process.
5. syscall.c: Add the declration of new function above
    Add the mapping in syscalls[] function.
    Set the sys_sum_ticks() and sys_init_start() in syscall() function to record the tick time. Before “proc->tf->eax = syscalls[num]();”, it’s the end of operating for process, so we sum this period of process time by calling sys_sum_ticks(). After “proc->tf->eax = syscalls[num]();”, the process will continue to run, so we call sys_init_start() to set the start time of this new period.
6. syscall.h: Define the correspond call number for those new system calls.
7. trap.c: In trap() function, I add the code to count the trap time and record the trap type when trap is from system call.
8. user.h: Add the declration of the new function.
9. usys.S: Add the new function name in mapping.
10. hw2test: This is the program for testing new system calls. Firstly it generates a child process to compare with parent process. And the loop is simple. The outer loop time is 5 with 5 times sleep() and 5 times getpid(). The inner loop time depends on the pid of process, and the code just literately increases the variable “testuse”. Particularly for parent process, it will wait child process at first, in case of interference between each other. And in order to check whether I/O influence the tick time of the process, the parent process contains a procedure to ask for keyboard input.

The mean traps are from getpid() and sleep(). The parent process will consume less time due to less number in 10000000*getpid(), but it should contains more traps than child process due to the extra wirte (printf) and read (gets) function. At last, each process will display the traps count, trap types and total tick time.

Procedure to execute:

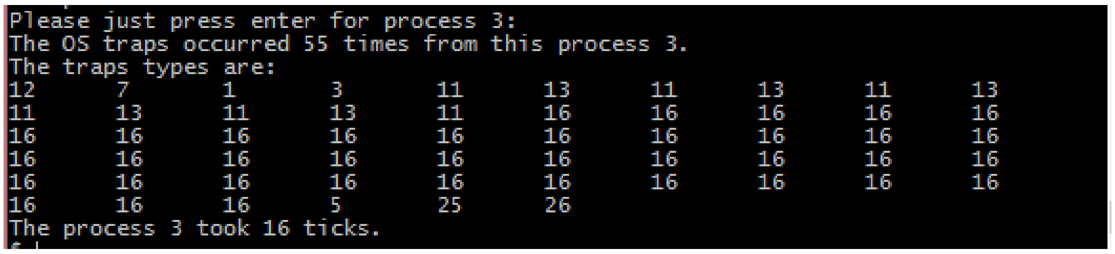
I run the xv6 OS under windows environment by the aid of vagrant and VM Virtual Box. So there may be a difference in xv6 files.

Firstly, I run hw2test command.
As we can see below, the child process (#4) finished and the count time of trap is 12, the total tick time is 20.
Looking at the trap types, there are 5 sets of number 11 which means SYS_getpid and 5 sets of number 13 which means SYS_getpid. Those are correspond to test code. And the number 3 at first means SYS_wait as the code has wait() at the beginning. The number 25, 26 is our new system call for printing. Particularly notice that the system call with number 26 is not included in the count time of traps, since print_trapstype() runs after is has counted trap times. Therefore the count number of 12 is correct.



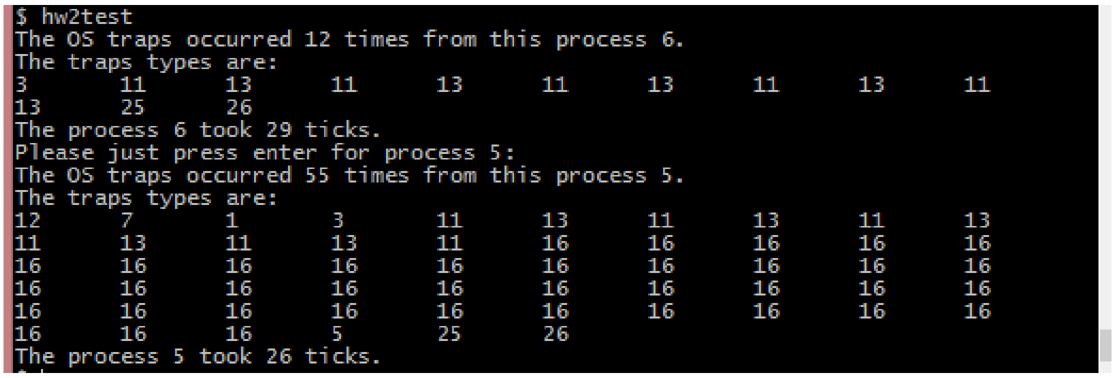
```
xv6...
cpu0: starting
init: starting sh
$ hw2test
The OS traps occurred 12 times from this process 4.
The traps types are:
3       11      13      11      13      11      13      11      13      11
13      25      26
The process 4 took 20 ticks.
Please just press enter for process 3:
```

Now the parent program (#3) wait to get enter notation. We can wait for a long time, for example one minutes. Then let us press enter and the information of parent process will display.
As we can see below, although we wait a long time before entering, the parent process still only takes 16 ticks. It means the program doesn't contain the time consumed by input, as the tick time should concentrate on the CPU time in this process. And checking the traps type, we can find there are extra more traps. At first, the number 12 and 7 means SYS_sbrk and SYS_exec which should belongs to parent process. The importance is we can find that there are many write system calls (number 16) and one read system call (number 5). They correctly correspond to the printf() and gets() function. The count number is also correct.



```
Please just press enter for process 3:
The OS traps occurred 55 times from this process 3.
The traps types are:
12      7       1       3       11      13      11      13      11      13
11      13      11      13      11      16      16      16      16      16
16      16      16      16      16      16      16      16      16      16
16      16      16      16      16      16      16      16      16      16
16      16      16      16      16      16      16      16      16      16
16      16      16      5       25      26
The process 3 took 16 ticks.
```

We can run the hw2test again to make sure it will continuously correct.
Then we can find the similar result. The difference is that they get a longer tick time. This result is reasonable because the pid is larger than before. And process 6 should takes more time than process 5.



```
$ hw2test
The OS traps occurred 12 times from this process 6.
The traps types are:
3       11      13      11      13      11      13      11      13      11
13      25      26
The process 6 took 29 ticks.
Please just press enter for process 5:
The OS traps occurred 55 times from this process 5.
The traps types are:
12      7       1       3       11      13      11      13      11      13
11      13      11      13      11      16      16      16      16      16
16      16      16      16      16      16      16      16      16      16
16      16      16      16      16      16      16      16      16      16
16      16      16      16      16      16      16      16      16      16
16      16      16      5       25      26
The process 5 took 26 ticks.
```

Overall, the print_ticks() and print_trapscount() runs successfully.