Team9 Sprint 1 Report

Sprint Goal: Add system call that gets system-level information (total processes, free memory, system uptime)

Part I. Code Modification

types.h: Added typedef unsigned long uint32.

syscall.c: Added a global int totproc initialized to 0. It tracks how many processes are running at once through the built in xv6 allocproc() and wait() functions available in the proc.c file.

sysproc.c: Created 3 new system calls: sys_freemem(), sys_totproc(), sys_sysinfo().

sys_freemem(): Calculates the number of pages of free memory the kernel has then multiplies that number by 4 since each page is 4KB.

sys_totproc(): Returns the external totproc variable as is.

sys_sysinfo(): Initializes a usrbuff array into the user call's array of system info to be populated (passed through the argptr() function). It creates a local array of uint32 variables (since sys_freemem() needs to be an uint32), and populates it with the return values of sys_uptime(), sys_totproc(), and sys_freemem(). Then, it uses the copyout() function to copy this local array into the process's own memory page. This function returns 1 if successful and -1 if an error occurs.
Note: sys_uptime() is an already existing function

printf.c: Added support for printing uint32 types in order to print the above information.
user.h: Added all above system calls to the file as valid system calls.
usys.S: Added all above system calls to the file as valid system calls.
syscall.h: Added all above system calls to the file as valid system calls.

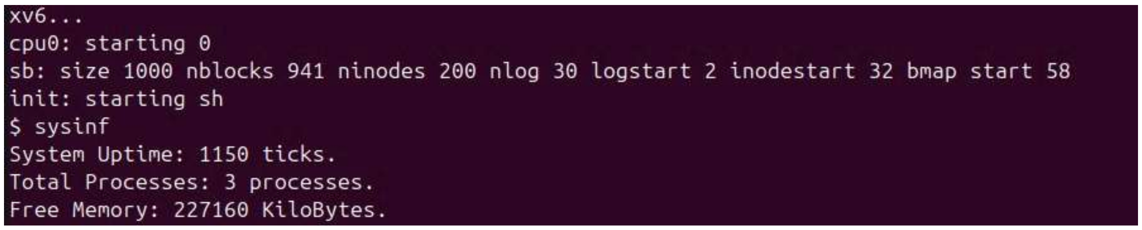
sysinf.c: Creates a uint32 array and passes it to sys_sysinfo(). If an error occurs during the call, printf() to the stderr file descriptor (2) that an error has occurred and exit(). Otherwise, printf() the content of the above array populated by sys_sysinfo() to the stdout file descriptor (1) and exit().

Part II. Elaboration

Regarding Total Processes: We added an external global variable "totproc", initialized at 0, which we added in syscall.c that increments when allocproc() - which creates a new process - is executed and decrements when wait() - which removes a process from the process queue - is executed.

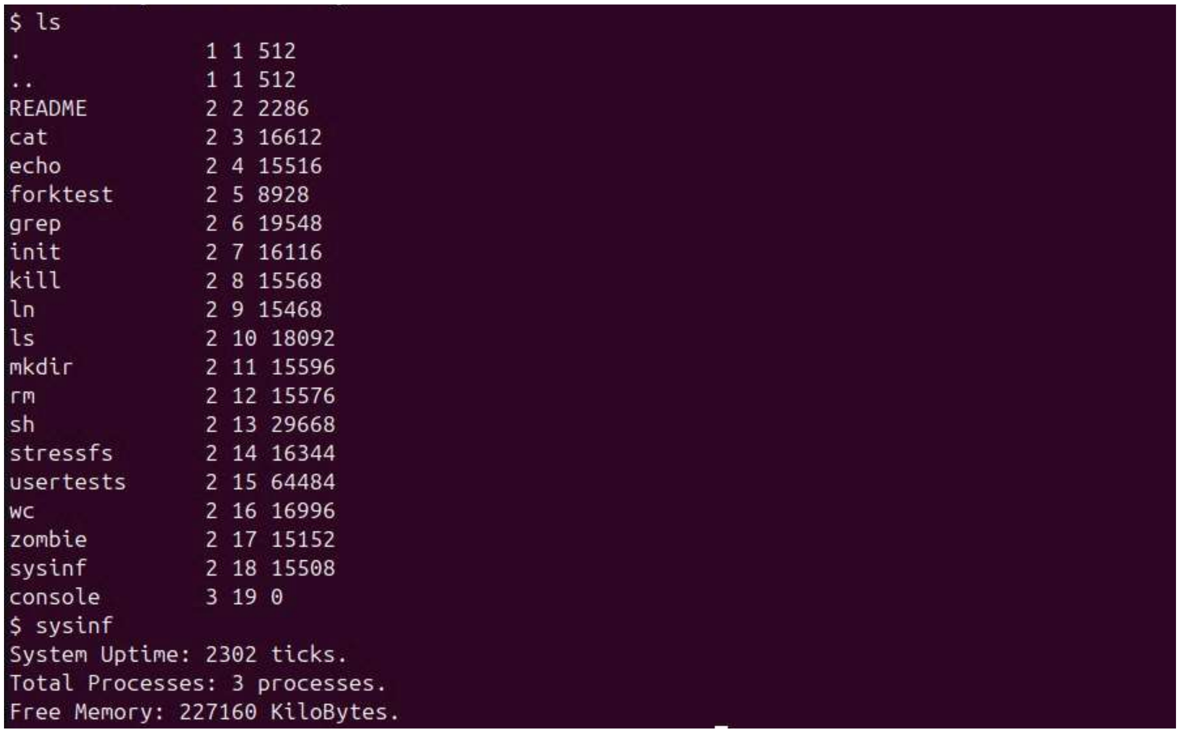
Regarding Free Memory: We used kernel memory to count the number of free pages through a linked list of free memory. Since each page is about 4KB, we simply multiply the total number of pages by 4 in order to get, in KB, the free memory. This value is printed as an uint32.
Regarding System Uptime: Already available, no changes
Regarding sys_sysinfo(): This system call returns all the required information in one call (it calls all the previous functions).
Regarding sysinf.c: This is a user call that prints out the information from sys_sysinfo(). We slightly modified printf.c in order to support the printing of uint32 values.

Part III. Output & Testing



```
xv6...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ sysinf
System Uptime: 1150 ticks.
Total Processes: 3 processes.
Free Memory: 227160 KiloBytes.
```

After the initial launch of the shell, we used sysinf in order to see the initial values of the system uptime, total processes, and free memory. Our test would basically be checking if the values of the total processes and free memory would change even if no changes were made.



```
$ ls
.              1 1 512
..             1 1 512
README         2 2 2286
cat            2 3 16612
echo           2 4 15516
forktest       2 5 8928
grep           2 6 19548
init           2 7 16116
kill           2 8 15568
ln             2 9 15468
ls             2 10 18092
mkdir          2 11 15596
rm             2 12 15576
sh             2 13 29668
stressfs       2 14 16344
usertests      2 15 64484
wc             2 16 16996
zombie         2 17 15152
sysinf         2 18 15508
console        3 19 0
$ sysinf
System Uptime: 2302 ticks.
Total Processes: 3 processes.
Free Memory: 227160 KiloBytes.
```

As we can see here, the only value that changed after only using ls was the system uptime. The number of total processes and the amount of free memory remained the same.