**getCallCount System Call Manual (MAN) Page**

**NAME**

getCallCount – get reference count of system call

**SYNOPSIS**

**#include “user.h”**

**int getCallCount(int** *sys\_call***);**

**DESCRIPTION**

**getCallCount**() indexes into *sys\_call\_counter* (defined in the *proc* struct)

for the system call number *sys\_call*.

*sys\_call\_counter* is a data structure declared for all new processes which

increments a reference counter anytime a system call is invoked.

**RETURN VALUE**

On success, the reference count of the *sys\_call* system call is returned

(zero indicates the system call was never invoked).

On error, -1 is returned.

**ERRORS**

If a non-integer *sys\_call* value or an undefined system call number is passed,

then an error will occur.

**NOTES**

Based on the implementation of the counter, a reference count for a given

system call is only incremented for a successful return of the system call. This

case is rarely noticed except in the instance where the system call number for

*getCallCount* (22) is passed.

It might be expected that

**printf(1, “getCallCount count: %d\n”, getCallCount(22));**

shows 1 as the count since the system call **getCallCount**() reference counter

should have been incremented. However, because the counter is only

incremented after a successful system call return, the value returned would be

0 until after the system call returns.

**BUGS**

Because of xv6’s implementation of *argint()* (found in *syscall.c*), it is possible to pass a floating point value into **getCallCount**() (or any system call) and not receive an error. This happens because *argint()* casts the passed value to an integer, where the system call adopts the “new” integer value.

**getCallCount System Call Implementation**

**xv6/usys.S**



Added line 32 to define the name of the system call for creating the trap vector.

**xv6/syscall.h**



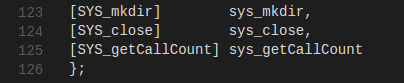
Added line 23 to define the value of the system call, which gets placed into the %eax register.

**xv6/syscall.c**



Added line 101 to declare the function defined in xv6/sysproc.c.

In \*syscalls[]:



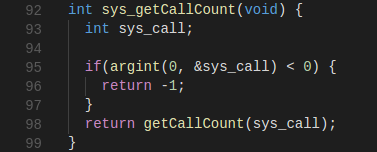
Added line 125 to create the mapping of the system call value to the system call function.

In syscall():



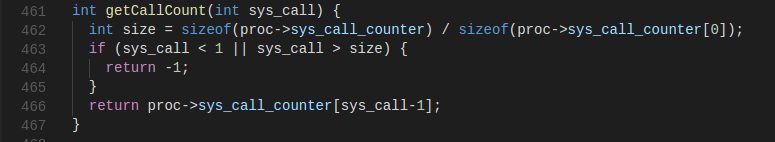
Added line 136 to increment the counter for the invoked system call (num). 1 is subtracted from num since the system calls start from 1, whereas the counter array index starts at 0.

**xv6/sysproc.c**



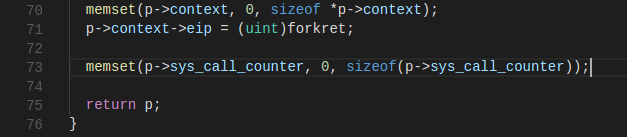
Added the function from lines 92-99 as a wrapper for the system call function. First, the passed parameter (the integer value of a system call) gets fetched from the stack. If the parameter is invalid, the function returns -1 (an error). For a valid integer, the system call function gets called and returns.

**xv6/proc.c**



Added the function from lines 461-467 to return the counter value for a given system call. If an invalid system call is passed (for our modified version of xv6, a valid system call number is 1-22), -1 is returned, indicating an error. For a valid system call number, the indexed counter value is returned. Same as the reasoning above, 1 is subtracted from sys\_call since the system calls start from 1, whereas the counter array index starts at 0.

in allocproc():



Added line 73 to initialize the counter array to all 0’s anytime a new process is allocated.

**xv6/user.h**



Added line 25 to define the system call and its parameter type to be called by the user.