**READ System Call**

**System Call:**

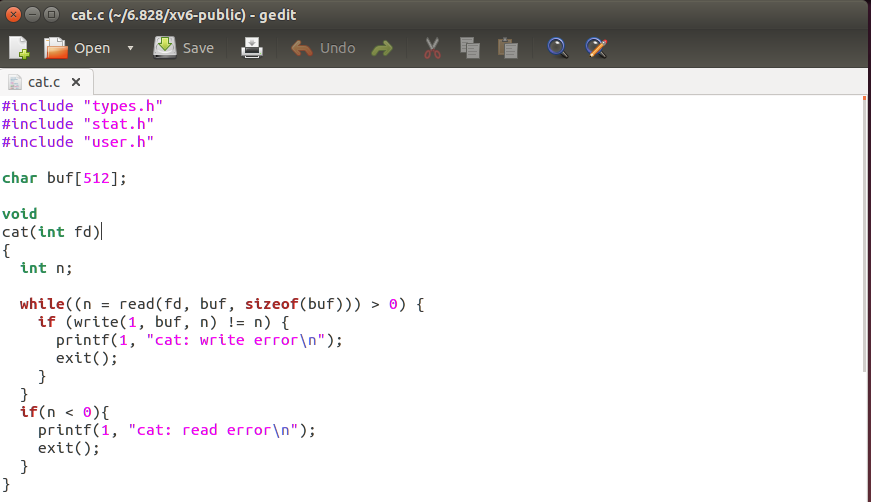
API of the operating system. Function that allows to access the properties of the CPU. Helps in raising the privilege level from the user mode to kernel mode.

**Cat Command:**

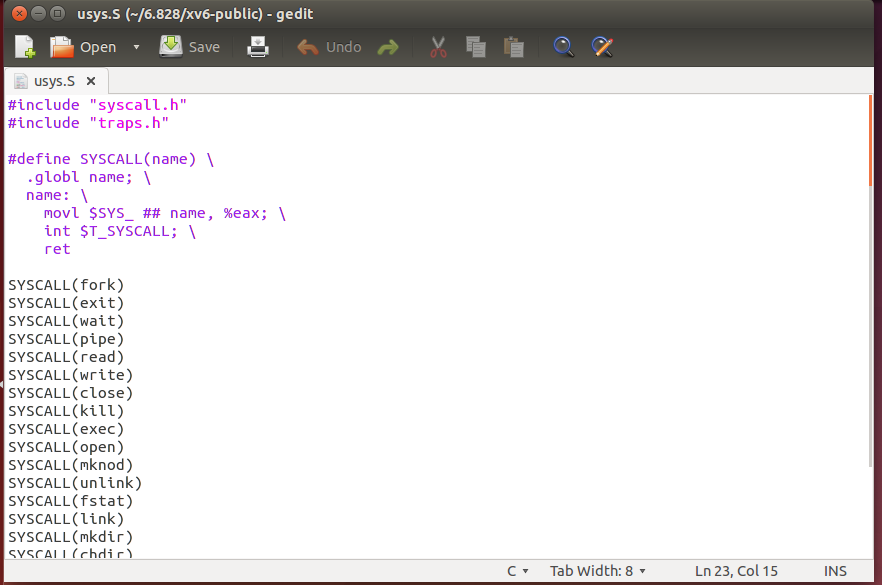
**Syntax :** cat Filename

Cat command followed by filename can be used to read the contents of a file.

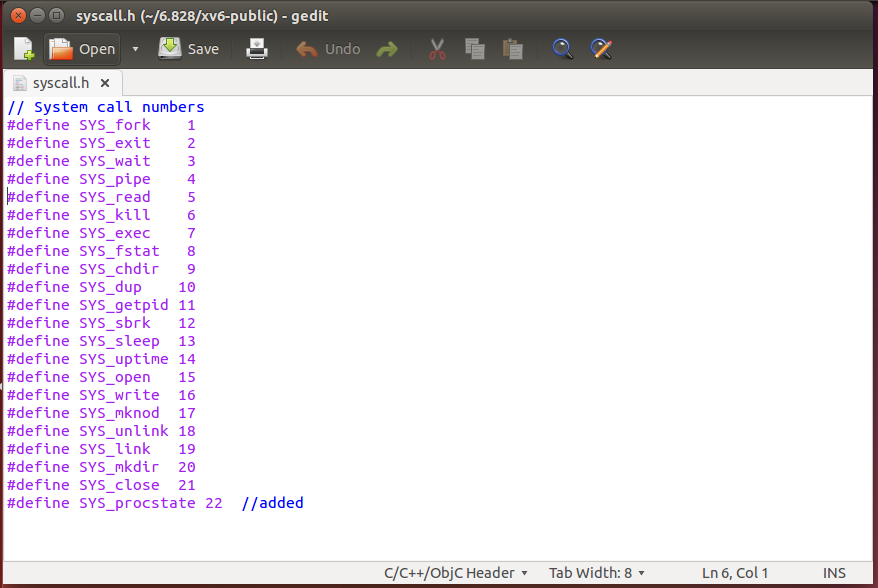
**File name:** cat.c



* Executing the cat command calls the read system call with argument as file descriptor.
* When this read system call is executed the control is transferred to **usys.S** filename.

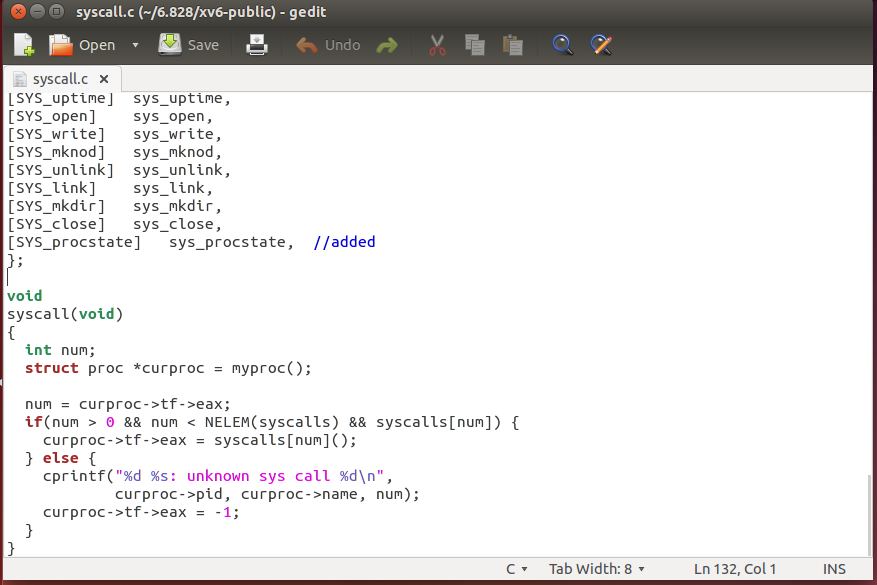


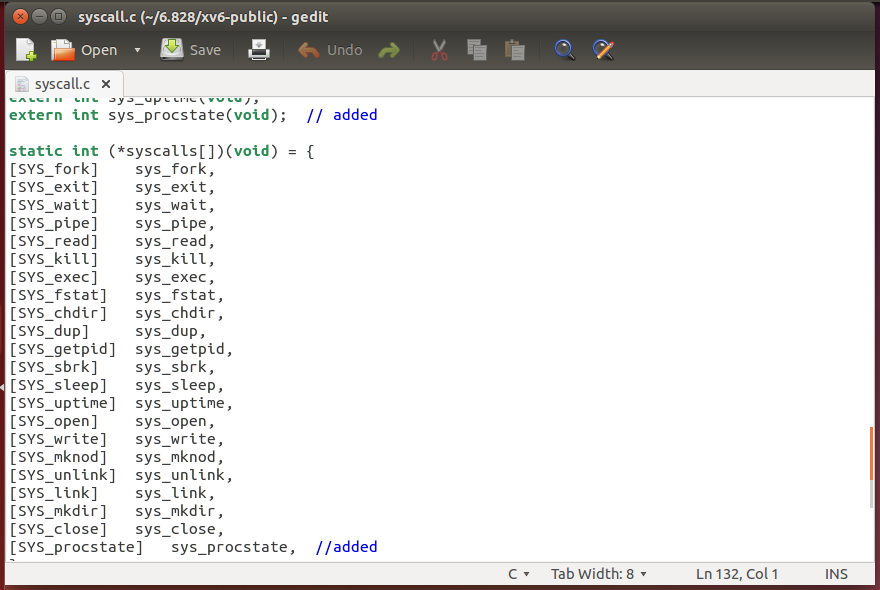
* In the usys.S file #define SYSCALL definition will create (read) as a global variable and moves the SYS\_read to the eax register.
* Int T\_SYSCALL (Global interrupt) is called after moving the contents to the eax register. (Useful in interrupting the processor)



* The SYS\_read system call is defined in the **syscall.h** user file. It is given a unique number as 5.
* So, the value of 5 for the read system call is stored in the eax register.

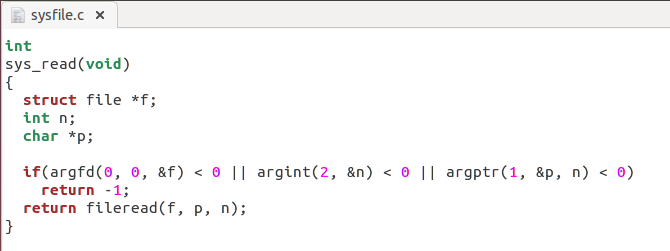
After defining the system call and storing the system call number in the eax register. Now the control is transferred to the syscall() function defined in the **syscall.c** file.



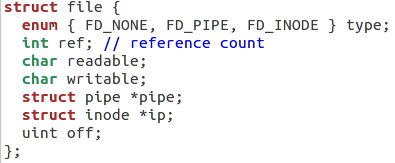


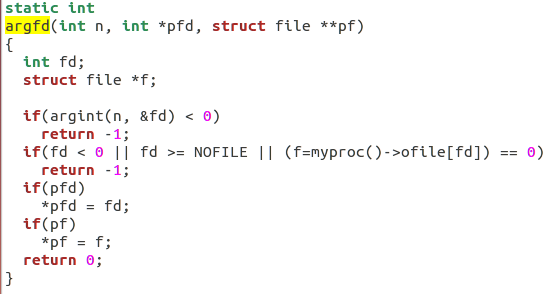
* In syscall.c file syscall(num) function is called with num as 5 for read system call and it puts the contents in the eax register and it is returned to the user. *(curproc->tf->eax=syscalls[num]())*
* Here the system call number is mapped to sys\_read function. *([SYS\_read] sys\_read )*

Implementation of sys\_read function is defined in the **sysfile.c** file.



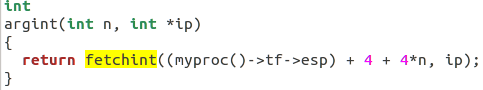
* In the sys\_read system call function before communicating with the kernel a pointer \*f is initialized of type file.



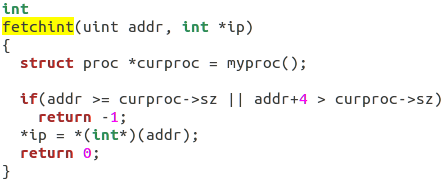
* After pointer creation outcomes of argfd(),argint() and argptr() are validated and based on the results it returns -1 or calls fileread() function.
* Argdf() function fetches the nth word-sized system call argument as a file descriptor

and return both the descriptor and the corresponding struct file.

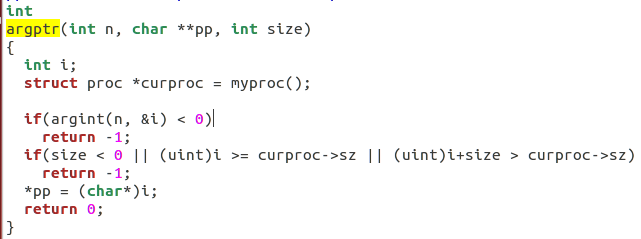
* Argint() function fetches the nth 32-bit system call argument.



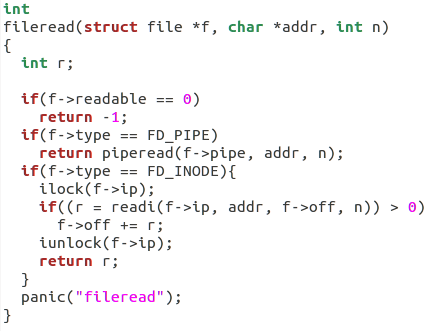
* Fetch the int at address from the current process.



* Argptr() function fetches the nth word-sized system call argument as a pointer to a block of memory of size bytes. Check that the pointer lies within the process address space.



* File read function is used to read the contents from the file. Available in **file.c** file.



Thus the flow of the system call (READ) can be represented as cat.c->usys.S->syscall.h->syscall.c->sysfile.c->file.c.