ASSIGNMENT

HENIL VISESH SHIVAM

October 2022

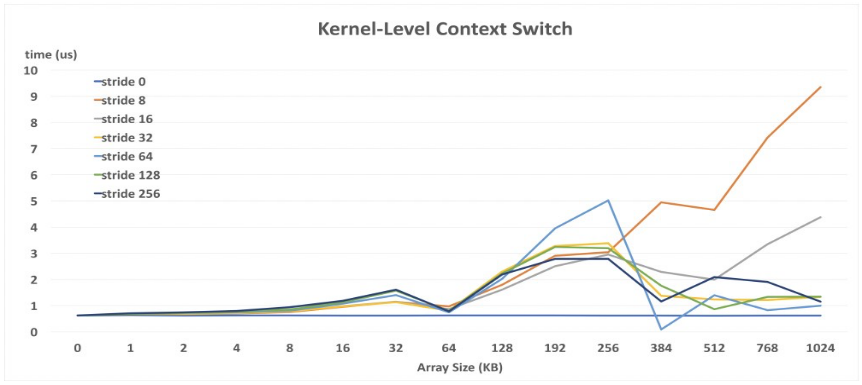
1 Trace of Read and Write System call

(I) Write System Call Tracing : Here I write the Read system call with the parameters, write=(int file Descriptor, void* buffer, size of((buffer)); In the tracing, first it access the file descriptor, here the file descriptor store the int values of the created file, so if the parameter of the file descriptor -1, it means the file maybe crashed or its give an error, so this file descriptor is helpful for the storingthe open system call values, so for the back-tracing it traces the open system call, int open (const char* Path, int flags [, int mode]); in this call based on the given first and second parameters, if the file .txt is not created then this functioncreates the .txt file, and if the file is available then it erase all the data from the file, and in the last parameter based on the permission given by user's the .txt file is created, now using the value of file descriptor, we access the completed file, and here we defined the size of the buffer, this is helpful for the storing the data into buffer, and the last parameter it traced the size of the buffer, it we write the data more than the assigning the value of the buffer, then it gives an error. If we store the value of write into variables, then there it may return the values like if we get the 0, the meaning is that we write the data into a file and reach the end file; also, for the value of -1, we can consider it as an error or signal interrupt. This is the way we can trace the write system call.

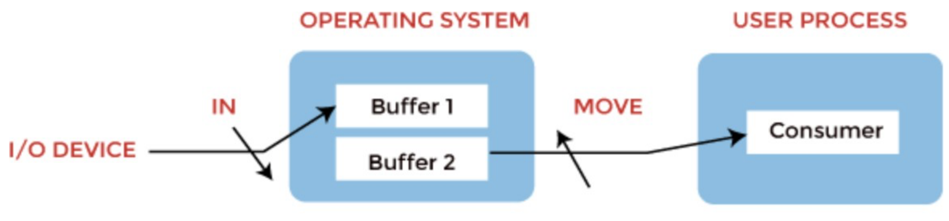
(II) Read System Call Tracing: Here I write the Read system call with the parameter: size_t read (int fd, void* buf, size_t cnt); In the tracing, we need the file descriptor, here the descriptor refers value of variable, in which we write the data into the file, so this file descriptor access the value of write of file, through that we can use the value, now here the read_buffer is present, so this useful for read the data from the buffer, now the last parameter is the sizeof(read_buffer), so this read_buffer is helpful for to the store the accessing value through the file descriptor,so this the way we backtrack the read system call. Here I attach the reference for the above tracing system call https://www.geeksforgeeks.org/input-output-system-calls-c-create-open-close-read-write/

2 How to Improve Read and Write System Call

System calls are used to transfer operations from kernel mode to user mode. Unlike a typical program, this system call requires the application to ask the CPU to enter kernel mode before moving on to the kernel's predetermined lo- cation. The main changes focus on improving read and write system calls while reducing switching. Operating systems, computer architectures, and the num- ber of shared resources between processes all influence the frequency of context flipping. The first is that we need to keep the queue linked properly so that the transition occurs appropriately, and we don't wait for the link to other resources. I started several methods by which we can in- crease the efficiency of the context switch. The second way is to increase the efficiency of the read and-write system calls by using the proper locks to stop competing processes. For example, if the kernel allocates a buffer for a read system call, that system call will block the file until the I/O transmission from the disk has finished. However, switching takes time, and completing processes may still be running during that time. According to a recent study, there are typically 150 cycles, with 79 entering and 71 leaving due to context shifts. The below image has been taken from https://web.eecs.umich.edu/ chshibo/files/microkillerreport.pdf

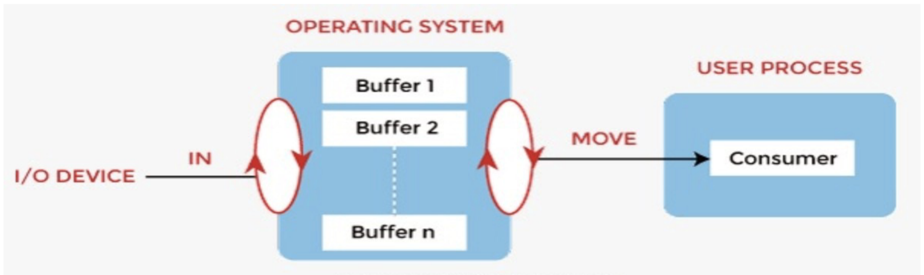


Reducing the frequency with which a process enters, and leaves kernel mode is the quickest and most straightforward technique to resolve this problem. This is yet another tactic for raising system call efficiency. Because we release all the resources needed for the completion of the program when the number of exit calls rises, there is a chance that the program will occasionally crash because it will be challenging to complete the task if the thread still needs to access the resource. Therefore, exit calls sometimes decrease the probability that the program will end and increase the risk of crash. Another method is to use the cache to increase read and write system call efficiency. The cache is used to temporarily store essential data that is needed for subsequent processing and is kept in a computing environment. This method temporarily stores the cached data on a local, cache client-accessible storage medium that is not connected to the primary storage. This method allows clients to easily access data by checking the buffer cache; if the data is located there, this is referred to as a cache hit. If the information is not found in the cache, it is referred to as a cache miss, and the client must access the data from the main memory. To improve the read and write system call, we can use the double buffering method, so in this method, there are two buffers placed in one place, here the one buffer is considered as the producer, here we assumed that write is the producer. The second buffer is consumer, means the read system call, so in this scenario, the data is written into the producer buffer, it is directly used by the consumer buffer, so through that, we do not wait for the buffer filling and empty problem,by using this we can reduce the access time of resources, but the disadvantageis that the process complexity in this situation is increased.



DOUBLE I/O BUFFERING

Another method is the circular buffer technique; in this technique, there are more than two buffers used, so in this technique, the data cannot transfer directly like in the double buffering method; here, the data is overwritten into the buffer. The main advantage of using both techniques, the buffer reduces the traffic of the disk. As a result, it would increase the overall throughput of the execution call, and it may automatically decrease the response time. But the disadvantage is that it is unpredictable about the buffer size because initially, we did not know about the number of elements we required to store in the buffer; if the number of elements size is more than the defined buffer size, then it transfers the larger size for the buffer, and if the number of elements is smaller, then it used for the storing purpose, but there is the waste of memory.



CIRCULAR I/O BUFFERING

https://www.javatpoint.com/buffering-operating-system.