Que es el char \_\_user \*???

As you read through the list of file\_operations methods, you will note that a number of parameters include the string \_\_user. This annotation is a form of documentation, noting that a pointer is a user-space address that cannot be directly dereferenced. For normal compilation, \_\_user has no effect, but it can be used by external checking software to find misuse of user-space addresses.

Why all system calls in Linux passes arguments to kernel using “call by reference”?

Efficiency.

The memory space for user mode and kernel mode are different. When you make a system call, the MMU of the Linux subsystem makes sure that proper memory mapping of the user space process running in their own Virtual address space is done to the Physical address space of the kernel. Variables in the user mode stay in the process' virtual address space. They can't just be passed in system calls and expected to get mapped in the physical address space .

This is what my understanding is. Would love to discuss and clarify if needed.

Many (most?) systems implement function calls by pushing argument values onto a stack. If you pass a struct or any other complex data type by value, you'd need to copy it to the stack. There's no reason to do this, since the kernel has access to the entire memory space of the process. Aside from the copy cost, you'd also increase the stack space needed.

In addition, the kernel will need to copy any data it needs to retain into the kernel memory space. The kernel can't rely on user space code behavior. (It's also not going to free anything obtained from user space, which eliminates some any concerns over mixing up responsibility for reclaiming memory.)

Finally, realistically, coders working in the kernel need to be very comfortable with working with pointers. There's really no advantage to passing by value once you're completely comfortable with pointers.

Parameters are passed to system calls

Second, because a system call runs on its own stack, the number of arguments that can be passed to a system call is limited. The operating system linkage conventions specify that up to eight general purpose registers are used for parameter passing. . Because a system call does not have direct access to the application's stack, all parameters for system calls must fit in eight registers.

Los registros eax ebx ecx y edx se emplean

register vs stack cpu

A **stack register** is a computer central [processor register](https://en.wikipedia.org/wiki/Processor_register) whose purpose is to keep track of a [call stack](https://en.wikipedia.org/wiki/Call_stack). On an [accumulator-based architecture](https://en.wikipedia.org/wiki/Accumulator-based_architecture) machine, this may be a dedicated register such as SP on an [Intel x86](https://en.wikipedia.org/wiki/Intel_x86) machine.  A register is used to store the address of the topmost element of the stack which is known as **Stack pointer (SP)**. In this organisation, ALU operations are performed on stack data. It means both the operands are always required on the stack. After manipulation, the result is placed in the stack.

En arquitectura de ordenadores, un registro es una memoria de alta velocidad y poca capacidad, integrada en el microprocesador, que permite guardar transitoriamente y acceder a valores muy usados,

FORK

**fork** is an operation whereby a [process](https://en.wikipedia.org/wiki/Computer_process) creates a copy of itself.  In multitasking operating systems, processes (running programs) need a way to create new processes, e.g. to run other programs. Fork and its variants are typically the only way of doing so in Unix-like systems. For a process to start the execution of a different program, it first forks to create a copy of itself. Then, the copy, called the "[child process](https://en.wikipedia.org/wiki/Child_process)", calls the [exec](https://en.wikipedia.org/wiki/Exec_(system_call)) system call to overlay itself with the other program: it ceases execution of its former program in favor of the other.  is used to create processes. It takes no arguments and returns a process ID. The purpose of fork() is to create a new process, which becomes ...

MACROS

Los macros son muy utilizados en C y C++. Estos básicamente son un alias que podemos incluir en nuestro código el cual, al momento de compilar, sera reemplazado por lo que hayamos definido. Un macro no es más que otro **define**, pero dado que es capaz, o al menos susceptible de **realizar decisiones lógicas**, o **funciones matemáticas**, una **macro** es una porción de código que se traduce en tiempo de preproceso.

What does asmlinkage mean in the definition of system calls?

The short answer to your question is that asmlinkage tells your compiler to look on the CPU stack for the function parameters, instead of registers.

System calls are *services* that userspace can call to request the kernel to perform something for them (and therefore execute in kernel space). These functions are quite unorthodox in the sense that you cannot expect them to behave like normal functions, where parameters are typically passed by writing to the program stack, but instead they are written to registers. While still in userspace, calling a syscall requires writing certain values to certain registers is translated. The system call number (http://lxr.linux.no/linux+v3.5.4/arch/x86/syscalls/syscall\_64.tbl) will always be written in eax, while the the rest of the parameters will go into ebx, ec

ithout asmlinkage, all the parameters for the syscall will be passed in registers. Since there are only 6 general purpose registers in x86/x86\_64, more than 6 parameters can not be passed without asmlinkage

Why is Linux syscall return type “long”?

Syscalls with fewer than 6 parameters passed in registers – %eax (syscall number), %ebx, %ecx, %esi, %edi, %ebp • If 6 or more arguments – Pass pointer to block structure containing argument list • Maximum size of argument is register size – Larger arguments passed as pointers • Use special routines to fetch pointer arguments – get\_user(), put\_user(), copy\_to\_user(), copy\_from\_user

-EFAULT if you pass an invalid pointer.

Links

Teoria

L10-syscall

http://www.cs.columbia.edu/~jae/4118/L10-syscall.pdf

System Call in OS: Types and Examples

https://www.guru99.com/system-call-operating-system.html

22C:169 Notes, Lecture 9

http://homepage.divms.uiowa.edu/~jones/security/notes/09.shtml

CS124Lec14

<http://users.cms.caltech.edu/~donnie/cs124/lectures-wi2016/CS124Lec14.pdf>

Tutorials

Adding A System Call

https://www.csee.umbc.edu/courses/undergraduate/CMSC421/fall02/burt/projects/howto\_add\_systemcall.html

Linux-kernel tutorial

https://blog.guillaume-gomez.fr/Linux-kernel/1/3

Add Your Own System Calls to the Linux Kernel | William the Grey's Blog

https://williamthegrey.wordpress.com/2014/05/18/add-your-own-system-calls-to-the-linux-kernel/

Adding a System Call Which Can Pass a Userspace String | by 李謙 | Medium

https://medium.com/@lee1003094395/adding-a-system-call-which-can-pass-a-userspace-string-b245105bed38

Lab on Making Your Own System Call

http://www.cs.albany.edu/~sdc/CSI500/Fal13/Labs/OwnSyscallV3plus/OwnSyscallFE.html

c - How to pass parameters to Linux system call? - Stack Overflow

https://stackoverflow.com/questions/53735886/how-to-pass-parameters-to-linux-system-call

How do I get the output of a Linux System Call in C/C++? - Stack Overflow

https://stackoverflow.com/questions/26828461/how-do-i-get-the-output-of-a-linux-system-call-in-c-c

Allocate memory

Kernel Korner - Allocating Memory in the Kernel | Linux Journal

https://www.linuxjournal.com/article/6930

kmalloc()/kfree() include/linux/slab.h

<https://people.netfilter.org/rusty/unreliable-guides/kernel-hacking/routines-kmalloc.html>

Tutoriales C

C Dynamic Memory Allocation Using malloc(), calloc(), free() & realloc()

https://www.programiz.com/c-programming/c-dynamic-memory-allocation

C Data Types

https://www.programiz.com/c-programming/c-data-types

C - Pointers and Strings - C Programming - DYclassroom | Have fun learning :-)

https://dyclassroom.com/c/c-pointers-and-strings#:~:text=Creating%20a%20pointer%20for%20the,point%20at%20the%20string%20str.