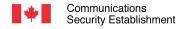
# 32 bits Linux system calls

Hello? who's this?

© Government of Canada

This document is the property of the Government of Canada. It shall not be altered, distributed beyond its intended audience, produced, reproduced or published, in whole or in any substantial part thereof, without the express permission of CSE.





#### Full list for 32 bits is:

/usr/include/i386-linux-gnu/asm/unistd\_32.h

© Government of Canada

This document is the property of the Government of Canada. It shall not be altered, distributed beyond its intended audience, produced, reproduced or published, in whole or in any substantial part thereof, without the express permission of CSE.





#### This file looks somewhat like this

```
#ifndef
         ASM X86 UNISTD 32 H
#define ASM X86 UNISTD 32 H 1
#define
          NR restart syscall 0
#define
          NR exit 1
#define
          NR fork 2
#define ¯
          NR read 3
#define
          NR write 4
#define
          NR open 5
          NR close 6
#define
#define
          NR waitpid 7
#define
          NR creat 8
#define
          NR link 9
#define
          NR unlink 10
#define
          NR execve 11
#define
          NR chdir 12
#define
          NR time 13
#define
          NR mknod 14
#define
          NR chmod 15
```

Each system call has its own number!

You can refer to this file if you are not sure what number is assigned to a specific system call.

#### Extract from:

/usr/include/i386-linux-gnu/asm/unistd\_32.h





# **Documentation for system calls**

man 2 <system\_call\_name>

I also like: https://syscalls.kernelgrok.com/

This document is the property of the Government of Canada. It shall not be altered, distributed beyond its intended audience, produced, reproduced or published, in whole or in any substantial part thereof, without the express permission of CSE.





## Doing a call - Hello system calls

```
global main
section .data
   hello:
       db 'Hello system calls', 0x0a, 0
section .text
main:
       push ebp
       mov ebp, esp
       mov eax, 0x04; Write System Call
       mov ebx, 0x01 ; 0=STDIN, 1=STDOUT, 2=STDERR
       mov ecx, hello ; Message
       mov edx, 0x13; Len
       pop ebp
       ret
```

System call **number** goes into **eax**.

Parameters go into ebx, ecx, edx, esi,edi, ebp from first to sixth parameters.

When dealing with STDIN, STDOUT or STDERR, their respective value is 0, 1, 2.

**0x80** is the **interrupt number** that allows to send a **software interrupt** for a system call into the kernel.

Calling convention details can be found using "man 2 syscall"

## Calling convention based on architecture

arch/ABI	instruction	syscall #	retval	Notes
arm/OABI arm/EABI arm64 blackfin i386 ia64 mips parisc s390 s390x sparc/32 sparc/64 x86_64 x32	swi NR swi 0x0 svc #0 excpt 0x0 int \$0x80 break 0x100000 syscall ble 0x100(%sr2, %r0) svc 0 svc 0 t 0x10 t 0x6d syscall syscall	- r7 x8 P0 eax r15 v0 r20 r1 r1 g1 g1 g1 rax	al r0 x0 R0 eax r8 v0 r28 r2 r2 o0 rax	NR is syscall #  See below See below See below See below See below See below

Extract from: man 2 syscall

# Let's write some code!



