Assembly no Linux

~~~~~~

SYSCALL

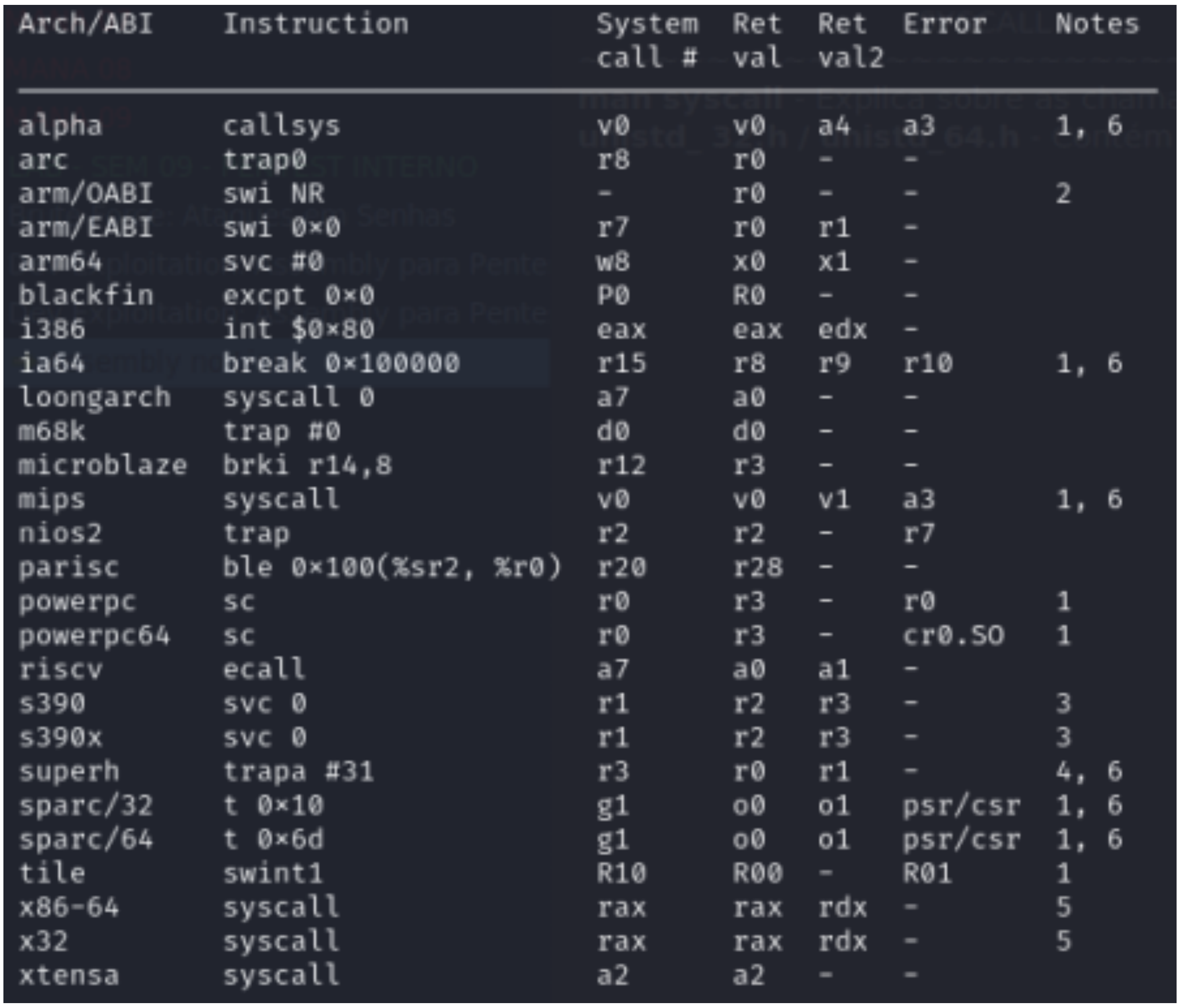
~~~~~~

man syscall - Explica sobre as chamadas no Linux

unistd_ 32.h / unistd_64.h - Contém os nomes das syscall com seus números

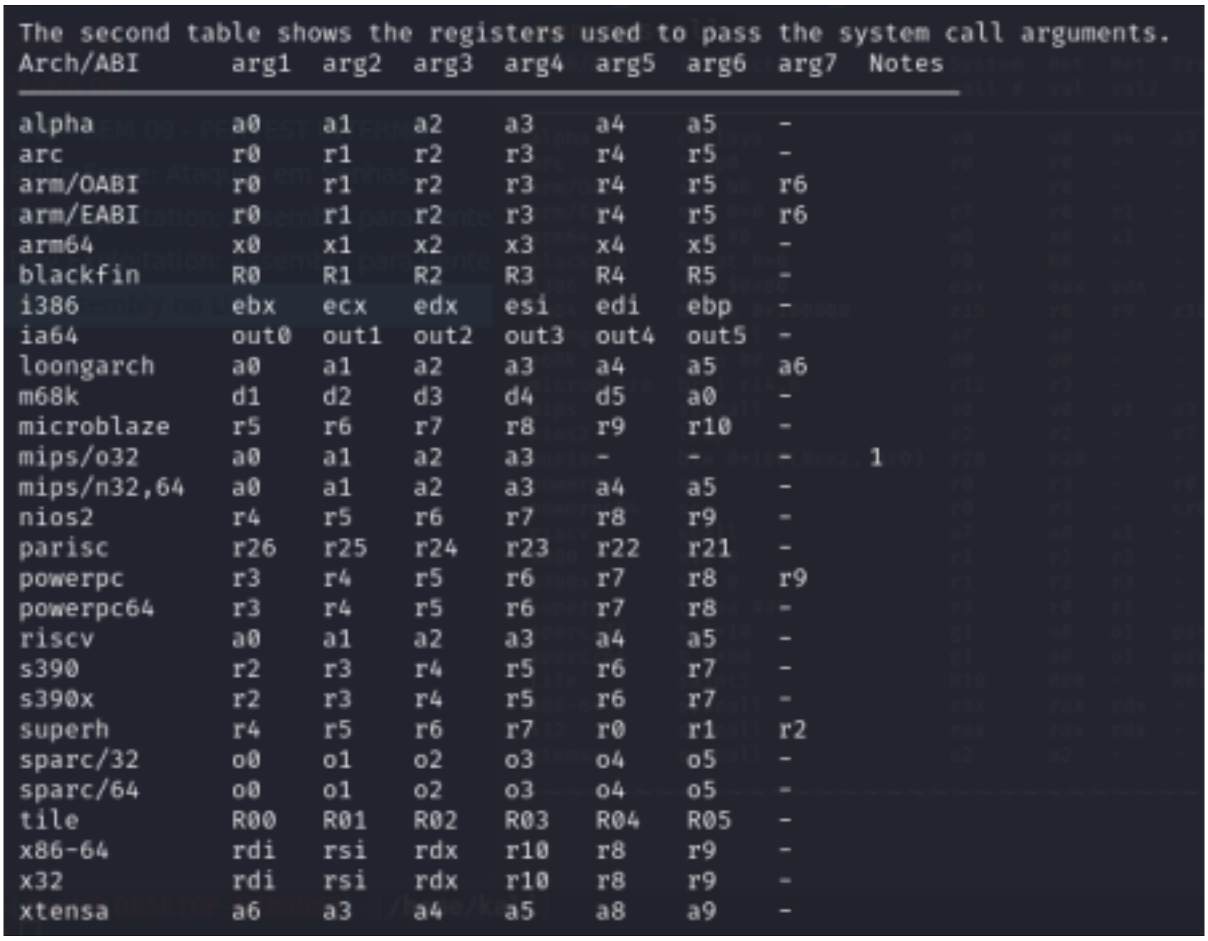
```
man syscall
```

The first table lists the instruction used to transition to kernel mode (which might not be the fastest or best way to transition to the kernel, so you might have to refer to vdso(7)), the register used to indicate the system call number, the register(s) used to return the system call result, and the register used to signal an error.



```
Arch/ABI    Instruction           System  Ret  Ret   Error    Notes
                                  call #  val  val2
───────────────────────────────────────────────────────────────────
alpha       callsys               v0      v0   a4    a3       1, 6
arc         trap0                 r8      r0   -     -
arm/OABI    swi NR                -       r0   -     -        2
arm/EABI    swi 0x0               r7      r0   r1    -
arm64       svc #0                w8      x0   x1    -
blackfin    excpt 0x0             P0      R0   -     -
i386        int $0x80             eax     eax  edx   -
ia64        break 0x100000        r15     r8   r9    r10      1, 6
loongarch   syscall 0             a7      a0   -     -
m68k        trap #0               d0      d0   -     -
microblaze  brki r14,8            r12     r3   -     -
mips        syscall               v0      v0   v1    a3       1, 6
nios2       trap                  r2      r2   -     r7
parisc      ble 0x100(%sr2, %r0)  r20     r28  -     -
powerpc     sc                    r0      r3   -     r0       1
powerpc64   sc                    r0      r3   -     cr0.SO   1
riscv       ecall                 a7      a0   a1    -
s390        svc 0                 r1      r2   r3    -        3
s390x       svc 0                 r1      r2   r3    -        3
superh      trapa #31             r3      r0   r1    -        4, 6
sparc/32    t 0x10                g1      o0   o1    psr/csr  1, 6
sparc/64    t 0x6d                g1      o0   o1    psr/csr  1, 6
tile        swint1                R10     R00  -     R01      1
x86-64      syscall               rax     rax  rdx   -        5
x32         syscall               rax     rax  rdx   -        5
xtensa      syscall               a2      a2   -     -
```

~~~~~~
~~~~~~



The second table shows the registers used to pass the system call arguments.

Arch/ABI	arg1	arg2	arg3	arg4	arg5	arg6	arg7	Notes
alpha	a0	a1	a2	a3	a4	a5	-	
arc	r0	r1	r2	r3	r4	r5	-	
arm/OABI	r0	r1	r2	r3	r4	r5	r6	
arm/EABI	r0	r1	r2	r3	r4	r5	r6	
arm64	x0	x1	x2	x3	x4	x5	-	
blackfin	R0	R1	R2	R3	R4	R5	-	
i386	ebx	ecx	edx	esi	edi	ebp	-	
ia64	out0	out1	out2	out3	out4	out5	-	
loongarch	a0	a1	a2	a3	a4	a5	a6	
m68k	d1	d2	d3	d4	d5	a0	-	
microblaze	r5	r6	r7	r8	r9	r10	-	
mips/o32	a0	a1	a2	a3	-	-	-	1
mips/n32,64	a0	a1	a2	a3	a4	a5	-	
nios2	r4	r5	r6	r7	r8	r9	-	
parisc	r26	r25	r24	r23	r22	r21	-	
powerpc	r3	r4	r5	r6	r7	r8	r9	
powerpc64	r3	r4	r5	r6	r7	r8	-	
riscv	a0	a1	a2	a3	a4	a5	-	
s390	r2	r3	r4	r5	r6	r7	-	
s390x	r2	r3	r4	r5	r6	r7	-	
superh	r4	r5	r6	r7	r0	r1	r2	
sparc/32	o0	o1	o2	o3	o4	o5	-	
sparc/64	o0	o1	o2	o3	o4	o5	-	
tile	R00	R01	R02	R03	R04	R05	-	
x86-64	rdi	rsi	rdx	r10	r8	r9	-	
x32	rdi	rsi	rdx	r10	r8	r9	-	
xtensa	a6	a3	a4	a5	a8	a9	-	

→ A syscall conversa diretamente com o kernel do sistem

→ Essas tabelas são a documentação

https://syscalls.w3challs.com/?arch=x86_64

https://syscalls.w3challs.com/?arch=x86

\~

ASSEMBLER + LINKER

\~

NASM + LD (ELF32)

```
nasm -f elf32 file.asm
```

```
ld -entry _main -m elf_i386 file.o -o file
```

NASM + LD(ELF64)

```
nasm -f elf64 file.asm
```

```
ld -entry _main file.o -o file
```

\~

DEBUGGER PARA LINUX

\~

- GDB TUI
- EDB
- GDB