Power Architecture 64-bit Linux system call ABI

syscall

Invocation

The syscall is made with the sc instruction, and returns with execution continuing at the instruction following the sc instruction.

If PPC_FEATURE2_SCV appears in the AT_HWCAP2 ELF auxiliary vector, the scv 0 instruction is an alternative that may provide better performance, with some differences to calling sequence.

syscall calling sequence[1] matches the Power Architecture 64-bit ELF ABI specification C function calling sequence, including register preservation rules, with the following differences.

[1] Some syscalls (typically low-level management functions) may have different calling sequences (e.g., rt_sigreturn).

Parameters

The system call number is specified in r0.

There is a maximum of 6 integer parameters to a syscall, passed in r3-r8.

Return value

- For the sc instruction, both a value and an error condition are returned. cr0.SO is the error condition, and r3 is the return value. When cr0.SO is clear, the syscall succeeded and r3 is the return value. When cr0.SO is set, the syscall failed and r3 is the error value (that normally corresponds to errno).
- For the scv 0 instruction, the return value indicates failure if it is -4095..-1 (i.e., it is >= -MAX_ERRNO (-4095) as an unsigned comparison), in which case the error value is the negated return value.

Stack

System calls do not modify the caller's stack frame. For example, the caller's stack frame LR and CR save fields are not used.

Register preservation rules

Register preservation rules match the ELF ABI calling sequence with some differences.

For the sc instruction, the differences from the ELF ABI are as follows:

Register	Preservation Rules	Purpose
r0	Volatile	(System call number.)
r3	Volatile	(Parameter 1, and return value.)
r4-r8	Volatile	(Parameters 2-6.)
cr0	Volatile	(cr0.SO is the return error condition.)
cr1, cr5-7	Nonvolatile	
lr	Nonvolatile	

For the scv 0 instruction, the differences from the ELF ABI are as follows:

Register	Preservation Rules	Purpose
r0	Volatile	(System call number.)
r3	Volatile	(Parameter 1, and return value.)
r4-r8	Volatile	(Parameters 2-6.)

All floating point and vector data registers as well as control and status registers are nonvolatile.

Transactional Memory

Syscall behavior can change if the processor is in transactional or suspended transaction state, and the syscall can affect the behavior of the transaction.

If the processor is in suspended state when a syscall is made, the syscall will be performed as normal, and will return as normal. The syscall will be performed in suspended state, so its side effects will be persistent according to the usual transactional memory semantics. A syscall may or may not result in the transaction being doomed by hardware.

If the processor is in transactional state when a syscall is made, then the behavior depends on the presence of PPC_FEATURE2_HTM_NOSC in the AT_HWCAP2 ELF auxiliary vector.

- If present, which is the case for newer kernels, then the syscall will not be performed and the transaction will be doomed by the kernel with the failure code TM_CAUSE_SYSCALL | TM_CAUSE_PERSISTENT in the TEXASR SPR.
- If not present (older kernels), then the kernel will suspend the transactional state and the syscall will proceed as in the case of a

suspended state syscall, and will resume the transactional state before returning to the caller. This case is not well defined or supported, so this behavior should not be relied upon.

scv 0 syscalls will always behave as PPC_FEATURE2_HTM_NOSC.

ptrace

When ptracing system calls (PTRACE_SYSCALL), the pt_regs.trap value contains the system call type that can be used to distinguish between sc and scv 0 system calls, and the different register conventions can be accounted for.

If the value of (pt_regs.trap & 0xfff0) is 0xc00 then the system call was performed with the sc instruction, if it is 0x3000 then the system call was performed with the scv 0 instruction.

vsyscall

vsyscall calling sequence matches the syscall calling sequence, with the following differences. Some vsyscalls may have different calling sequences.

Parameters and return value

r0 is not used as an input. The vsyscall is selected by its address.

Stack

The vsyscall may or may not use the caller's stack frame save areas.

Register preservation rules

r0	Volatile
cr1, cr5-7	Volatile
lr	Volatile

Invocation

The vsyscall is performed with a branch-with-link instruction to the vsyscall function address.

Transactional Memory

vsyscalls will run in the same transactional state as the caller. A vsyscall may or may not result in the transaction being doomed by hardware.