Gravel Kernel API

18–349 Embedded Real-Time Systems

October 21, 2012

1 Kernel API

This document lists all of the syscalls that the Gravel kernel supports. All of the syscalls are to be made using the SWI instruction. The SWI number to be used is the syscall number of your syscall added to a base of 0x900000. Arguments to a syscall are placed in order, in r0, r1, r2, r3. The return value of a syscall is placed in r0. No registers other than r0 are modified during a syscall. Control is transfered to the instruction succeeding the SWI after the syscall is complete. A syscall with an invalid syscall number results in the task being forcibly exited with an exit status of 0x0badc0de. A syscall returning an error returns the negative of the error codes mentioned below.

2 SWI Numbers

SWI Number	Syscall Name
0x900001	exit
0x900003	read
0x900004	write
0x900006	time
0x900007	sleep

3 Syscalls

3.1 exit

Syscall Number 1

Description `exit` immediately terminate the calling program. Any open file descriptors are closed. The exit status provided is returned to U-boot.

Argument 0 `int status`
    The exit status to return.

Return This function does not return.

3.2 read

Syscall Number 3

Description `read` attempts to read a requested number of bytes from the given file descriptor. The read values are written into the provided buffer sequentially.

Argument 0 `int fd`
    The file descriptor to read from — e.g. STDIN_FILENO

Argument 1 `void *buf`
> The target buffer to write any read data to.

Argument 2 `size_t count`
> The number of bytes to read into `buf`. If this value is zero, then zero is returned and nothing else happens. If this value is greater than SSIZE_MAX, the result is unspecified.

Return `ssize_t ret`
> On success, the number of bytes read is returned. This number can be less than `count` if a short read occurs due to an end-of-file or another error. The following errors may be returned.
>
> **EBADF** `fd` was an invalid or unreadable file descriptor.
>
> **EFAULT** `buf` points to region whose bounds lie outside valid address space.
>
> **EIO** An internal I/O error occurred.

3.3 write

Syscall Number 4

Description `write` attempts to write a requested number of bytes to the given file descriptor. The values to be written are read from the provided buffer sequentially.

Argument 0 `int fd`
> The file descriptor to write to — e.g. STDOUT_FILENO

Argument 1 `void *buf`
> The target buffer to read data from.

Argument 2 `size_t count`
> The number of bytes to read from `buf`. If this value is zero, then zero is returned and nothing else happens. If this value is greater than SSIZE_MAX, the result is unspecified. Note that this behavior is subtly different from the linux behavior and is used here to remain consistent with read.

Return `ssize_t ret`
> On success, the number of bytes written is returned. This number can be less than `count` if the underlying medium cannot accept any more bytes at the moment. The following errors may be returned on failure.
>
> **EBADF** `fd` was an invalid or unwritable file descriptor.
>
> **EFAULT** `buf` points to region whose bounds lie outside valid address space.
>
> **EIO** An internal I/O error occurred.

3.4 time

Syscall Number 6

Description `time` returns the time in milliseconds that have elapsed since the kernel booted up. This can be interpreted as the number of milliseconds to ellapse since the kernel timer was turned on.

Return `unsigned long ret`
> The function returns the time elapsed since the kernel booted up in milliseconds. There is no error case. Note that even though the number returned is in milliseconds, the actual resolution may be lower.

3.5 sleep

Syscall Number 7

Description `sleep` suspends the execution of the current task for the given time. The task may continue executing after atleast the given time has elapsed.

Argument 0 `unsigned long millis`
The number of milliseconds to sleep. Note that the kernel may round up the time provided if its timer resolution is lower than a millisecond.

Return `void`
There is no return value. The function always succeeds.