

The Userspace I/O HOWTO

Hans-Jürgen Koch

The Userspace I/O HOWTO

Hans-Jürgen Koch

Publication date 2006-12-11

Copyright © 2006-2008 Hans-Jürgen Koch.

Copyright © 2009 Red Hat Inc, Michael S. Tsirkin (mst@redhat.com)

Abstract

This HOWTO describes concept and usage of Linux kernel's Userspace I/O system.

This documentation is Free Software licensed under the terms of the GPL version 2.

Table of Contents

1. About this document	1
Translations	1
Preface	1
Acknowledgments	1
Feedback	1
2. About UIO	2
How UIO works	2
3. Writing your own kernel module	5
struct uio_info	5
Adding an interrupt handler	6
Using uio_pdrv for platform devices	7
Using uio_pdrv_genirq for platform devices	7
Using uio_dmem_genirq for platform devices	7
4. Writing a driver in userspace	9
Getting information about your UIO device	9
mmap() device memory	9
Waiting for interrupts	9
5. Generic PCI UIO driver	11
Making the driver recognize the device	11
Things to know about uio_pci_generic	11
Writing userspace driver using uio_pci_generic	12
Example code using uio_pci_generic	12
A. Further information	14

- `size`: The number of ports in this region.
- `porttype`: A string describing the type of port.

Set the `.name` element of `struct platform_device` to `"uio_dmem_genirq"` to use this driver.

When using this driver, fill in the `.platform_data` element of `struct platform_device`, which is of type `struct uio_dmem_genirq_pdata` and which contains the following elements:

- `struct uio_info uiainfo`: The same structure used as the `uio_pdrv_genirq` platform data
- `unsigned int *dynamic_region_sizes`: Pointer to list of sizes of dynamic memory regions to be mapped into user space.
- `unsigned int num_dynamic_regions`: Number of elements in `dynamic_region_sizes` array.

The dynamic regions defined in the platform data will be appended to the `mem[]` array after the platform device resources, which implies that the total number of static and dynamic memory regions cannot exceed `MAX_UIO_MAPS`.

The dynamic memory regions will be allocated when the UIO device file, `/dev/uioX` is opened. Similar to static memory resources, the memory region information for dynamic regions is then visible via `sysfs` at `/sys/class/uio/uioX/maps/mapY/*`. The dynamic memory regions will be freed when the UIO device file is closed. When no processes are holding the device file open, the address returned to userspace is `~0`.

other value for `count` causes `read()` to fail. The signed 32 bit integer read is the interrupt count of your device. If the value is one more than the value you read the last time, everything is OK. If the difference is greater than one, you missed interrupts.

You can also use `select()` on `/dev/uioX`.


```
    fprintf(stderr, "Started uio test driver.\n");
else
    fprintf(stderr, "Interrupts: %d\n", icount);

/*****
/* Here we got an interrupt from the
   device. Do something to it. */
*****/

/* Re-enable interrupts. */
err = pwrite(configfd, &command_high, 1, 5);
if (err != 1) {
    perror("config write:");
    break;
}

/* Wait for next interrupt. */
err = read(uiofd, &icount, 4);
if (err != 4) {
    perror("uio read:");
    break;
}
}
return errno;
}
```

Appendix A. Further information

- OSADL homepage. [<http://www.osadl.org>]
- Linutronix homepage. [<http://www.linutronix.de>]