Drivers and Miscellany

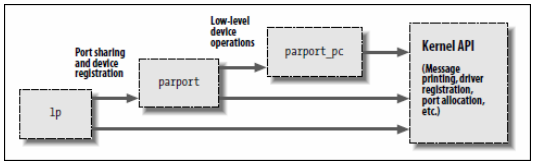
Jonathan Misurda
jmisurda@cs.pitt.edu

Error Handling

```
int __init my_init_function(void)
{
   int err;
   /* registration takes a pointer and a name */
   err = register_this(ptr1, “driver”);
   if (err) goto fail_this;
   err = register_that(ptr2, “driver”);
   if (err) goto fail_that;
   err = register_those(ptr3, “driver”);
   if (err) goto fail_those;
   return 0; /* success */

   fail_those: unregister_that(ptr2, “driver”);
   fail_that: unregister_this(ptr1, “driver”);
   fail_this: return err; /* propagate the error */
}
```

Driver Stacking



Race Conditions

- The kernel will make calls into your module while your initialization function is still running

Module Parameters

```
insmod hellop howmany=10 whom=“Mom”

static char *whom = "world";
static int howmany = 1;
module_param(howmany, int, S_IRUGO);
module_param(whom, charp, S_IRUGO);
```

Parameter Types

- `bool, invbool (int)`
- `charp`
- `int, long, short`
- `uint, ulong, ushort`

- Array parameters, where the values are supplied as a comma-separated list:
  - `module_param_array(name,type,num,perm);`

Permissions

- Module parameters show up as files in the sysfs entry
  - If perm is set to 0, there is no sysfs entry at all
- S_IRUGO – Read Only
- S_IRUGO|S_IWUSR – Writeable by root

Makefile

```
obj-m := hello_dev.o

KDIR  := /u/SysLab/shared/linux-2.6.23.1
PWD   := $(shell pwd)

default:
	$(MAKE) -C $(KDIR) M=$(PWD) modules
```

User Space Drivers

- FUSE – Filesystem in User Space
- Newest Kernels support other user space drivers
- Advantages?

BUFFERING: THE SILENT KILLER

A Mystery

```
#include <stdio.h>

int main()
{
        char a[10];
        char b;

        scanf("%s", a);
        printf("a is %s\n", a);
        b = getchar();
        printf("b is %c\n", b);

        return 0;
}
```

Output

```
(16) thot $ ./a.out
the
a is the
b is

(17) thot $
```

Mystery 2

```c
#include <stdio.h>

int main()
{
        printf("hello world");
        for(;;) ;

        return 0;
}
```

Output

```
(20) thot $ ./a.out

(21) thot $
```

BUFFERING!

fflush()

```c
#include <stdio.h>

int main()
{
        printf("hello world");
        fflush(stdout);
        for(;;) ;

        return 0;
}
```

Output

```
(24) thot $ ./a.out
hello world
(25) thot $
```