Device Drivers

Dr Andrew Scott

a.scott@lancaster.ac.uk

1

Device Drivers

- Extensible model
  - Create new drivers as needed to control hardware/ devices
  - Dynamically loadable modules allow run-time changes
    - Modular Operating System*
- Uniform, defined interface to I/O system
  - Allows third parties to write drivers for their new hardware
- Drivers must typically be re-entrant
  - Can be called again before previous request serviced
  - Can't store state in fixed locations

* See Linux commands: *lsmod*, *insmod*, *rmmod*, *modprobe*

2

Unix: Raw Device Functions

- Unix adopted a common 'file oriented' interface
  - read( ), write( ), ioctl( ), mmap( ), …

Entry Point	Function
open( )	Open the device
close( )	Close the device
ioctl( )	I/O control operations
mmap( )	Map device contents into memory
read( )	Input data
reset( )	Reinitialise device
select( )	Poll device for readiness
stop( )	Stop output on device
write( )	Output data

3

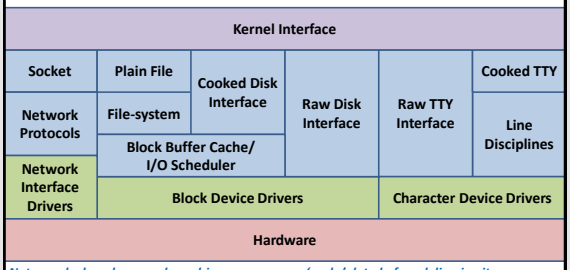
Example Device Driver Functions

- Typically maintain table of device functions
- Devices have NULL entries for functions/ operations that have no meaning for device

	Null	Memory	Keyboard	Terminal/ TTY	Printer
...	...	...	...	...	...
close( )	null	null	k_close	tty_close	lp_close
ioctl( )	null	null	k_ioctl	tty_ioctl	lp_ioctl
open( )	null	null	k_open	tty_open	lp_open
read( )	null	mem_read	k_read	tty_read	error
write( )	null	mem_write	error	tty_write	lp_write
...	...	...	...	...	...

4

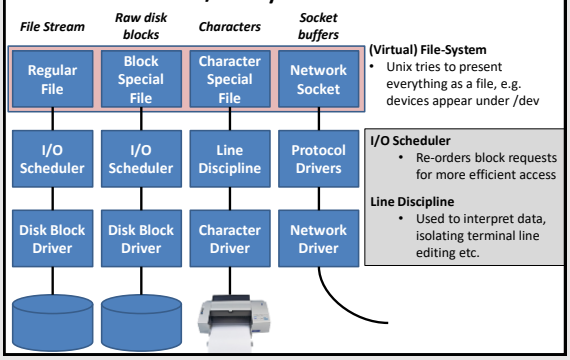
I/O in Traditional Unix Kernels



Note: cooked modes are where driver processes or 'cooks' data before delivering it, for example, line based terminal that interprets character combinations for line editing Raw (non-cooked) modes tend to be character oriented – raw tapes and disks are interesting examples... here, cooked is typically buffered and has byte zero as first user writeable byte (raw allows access from byte zero of the device, i.e., start of disk metadata, labels, etc.)

5

Unix I/O System in Use



6

Device Driver Tasks

- I/O Scheduling
- Buffering, Spooling and Caching
- Error Handling
- I/O Protection

7

Single- and Multi-Instance Devices

- Single instance (e.g. a printer)
  - Devices that can only be used by one process
  - Error if another process attempts to use them
  - Often use *spool files* to sequence requests
- Multi-instance
  - Can handle requests from multiple sources
  - Create another instance of internal data structures

8