CS 134:
Operating Systems
I/O Hardware

Overview

Overview
Patch Peer Review
Drivers
Low-Level
Mid-Level
Upper-Level
Unusual Devices

Patch Peer Review

Drivers

- Low-Level
- Mid-Level
- Upper-Level

Unusual Devices

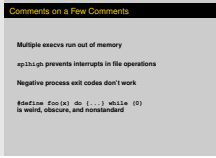
Comments on a Few Comments

Multiple execvs run out of memory

`splhigh` prevents interrupts in file operations

Negative process exit codes don't work

`#define foo(x) do {...} while (0)`
is weird, obscure, and nonstandard



Comments on a Few Comments

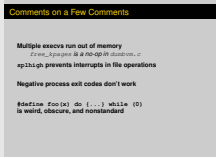
Multiple execvs run out of memory

`free_kpages` is a no-op in `dumbvm.c`

`splhigh` prevents interrupts in file operations

Negative process exit codes don't work

`#define foo(x) do {...} while (0)`
is weird, obscure, and nonstandard



Comments on a Few Comments

Multiple execvs run out of memory

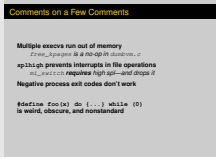
`free_kpages` *is a no-op in* `dumbvm.c`

`splhigh` prevents interrupts in file operations

`mi_switch` ***requires*** *high spl—and drops it*

Negative process exit codes don't work

`#define foo(x) do {...} while (0)`
is weird, obscure, and nonstandard



Comments on a Few Comments

Multiple execvs run out of memory

`free_kpages` is a no-op in `dumbvm.c`

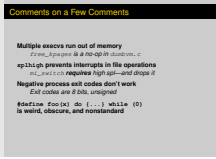
`splhigh` prevents interrupts in file operations

*`mi_switch` **requires** high spl—and drops it*

Negative process exit codes don't work

Exit codes are 8 bits, unsigned

`#define foo(x) do {...} while (0)`
is weird, obscure, and nonstandard



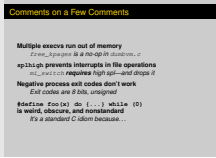
Comments on a Few Comments

Multiple execvs run out of memory
`free_kpages` is a no-op in `dumbvm.c`

`splhigh` prevents interrupts in file operations
*`mi_switch` **requires** high spl—and drops it*

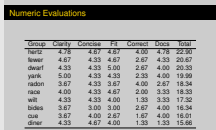
Negative process exit codes don't work
Exit codes are 8 bits, unsigned

`#define foo(x) do {...} while (0)`
is weird, obscure, and nonstandard
It's a standard C idiom because...



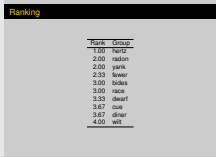
Numeric Evaluations

Group	Clarity	Concise	Fit	Correct	Docs	Total
hertz	4.78	4.67	4.67	4.00	4.78	22.90
fewer	4.67	4.33	4.67	2.67	4.33	20.67
dwarf	4.33	4.33	5.00	2.67	4.00	20.33
yank	5.00	4.33	4.33	2.33	4.00	19.99
radon	3.67	4.33	3.67	4.00	2.67	18.34
race	4.00	4.33	4.67	2.00	3.33	18.33
wilt	4.33	4.33	4.00	1.33	3.33	17.32
bides	3.67	3.00	3.00	2.67	4.00	16.34
cue	3.67	4.00	2.67	1.67	4.00	16.01
diner	4.33	4.67	4.00	1.33	1.33	15.66



Ranking

Rank	Group
1.00	hertz
2.00	radon
2.00	yank
2.33	fewer
3.00	bides
3.00	race
3.33	dwarf
3.67	cue
3.67	diner
4.00	wilt



Device Drivers

Device-access code can run in kernel or user mode (but usually kernel).

Driver must abstract control registers to OS's read/write model:

- Validate request
- Wait for idle
- Issue commands through control registers
- Possibly block waiting for interrupt
- Possibly invoke scheduler

Device Drivers
Device-access code can run in kernel or user mode (but usually kernel).
Driver must abstract control registers to OS's read/write model:
- Validate request
- Wait for idle
- Issue commands through control registers
- Possibly block waiting for interrupt
- Possibly invoke scheduler

Device Abstractions

Device Abstractions
Many devices have common characteristics; e.g., different brands of disk or printer
Makes sense to abstract common parts
Resulting structure is uniform driver sitting above specific one

Many devices have common characteristics; e.g., different brands of disk or printer

Makes sense to abstract common parts

Resulting structure is uniform driver sitting above specific one

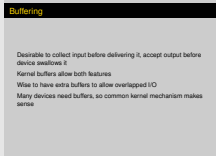
Buffering

Desirable to collect input before delivering it, accept output before device swallows it

Kernel buffers allow both features

Wise to have extra buffers to allow overlapped I/O

Many devices need buffers, so common kernel mechanism makes sense



Error Handling

Best option on errors: retry and hide from upper levels

Alternative: return error code to application & let it handle

Worst option: ask user what to do (user usually has insufficient information to make wise decision)

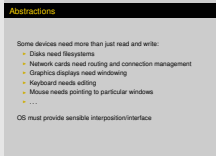
Error Handling
Best option on errors: retry and hide from upper levels
Alternative: return error code to application & let it handle
Worst option: ask user what to do (user usually has insufficient information to make wise decision)

Abstractions

Some devices need more than just read and write:

- Disks need filesystems
- Network cards need routing and connection management
- Graphics displays need windowing
- Keyboard needs editing
- Mouse needs pointing to particular windows
- …

OS must provide sensible interposition/interface



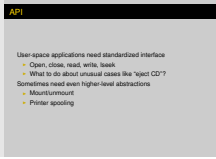
API

User-space applications need standardized interface

- Open, close, read, write, lseek
- What to do about unusual cases like “eject CD”?

Sometimes need even higher-level abstractions

- Mount/unmount
- Printer spooling



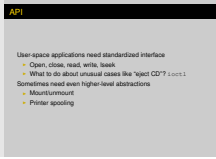
API

User-space applications need standardized interface

- Open, close, read, write, lseek
- What to do about unusual cases like “eject CD”? `ioctl`

Sometimes need even higher-level abstractions

- Mount/unmount
- Printer spooling



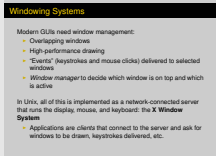
Windowing Systems

Modern GUIs need window management:

- Overlapping windows
- High-performance drawing
- “Events” (keystrokes and mouse clicks) delivered to selected windows
- *Window manager* to decide which window is on top and which is active

In Unix, all of this is implemented as a network-connected server that runs the display, mouse, and keyboard: the **X Window System**

- Applications are *clients* that connect to the server and ask for windows to be drawn, keystrokes delivered, etc.



Clocks

Clocks come in two flavors:

1. Read/write interface (read gives time of day, write sets it)
2. Pure interrupt interface (interrupt every so often)

Typically, first kind is used only at boot time, then periodic interrupts maintain TOD and force process switches

All clocks *drift*; NTP (Network Time Protocol) allows synchronization to GPS or standardized atomic clocks

Clocks

Clocks come in two flavors:
1. Read/write interface (read gives time of day, write sets it)
2. Pure interrupt interface (interrupt every so often)

Typically, first kind is used only at boot time, then periodic interrupts maintain TOD and force process switches

All clocks drift; NTP (Network Time Protocol) allows synchronization to GPS or standardized atomic clocks

Keyboards

Unlike all other devices, humans can't reliably generate input

Keyboard must allow *line editing* to compensate

- Typically supported in driver
- Problem: some programs have own line-editing needs
- Solution: *raw* (as opposed to *cooked*!) mode
- Cooked mode also echoes input

Keyboards
Unlike all other devices, humans can't reliably generate input
Keyboard must allow line editing to compensate
Typically supported in driver
Problem: some programs have own line-editing needs
Solution: raw (as opposed to cooked) mode
Cooked mode also echoes input

Mice

Mouse generates periodic updates: $\Delta x, \Delta y$, buttons

Problem: to whom do mouse events go?

Mice
Mouse generates periodic updates: $\Delta x, \Delta y$, buttons
Problem: to whom do mouse events go?

Mice

Mouse generates periodic updates: $\Delta x, \Delta y$, buttons

Problem: to whom do mouse events go?

Solution: Send to windowing system, let it decide which window is interested

Mice
Mouse generates periodic updates: $\Delta x, \Delta y$, buttons
Problem: to whom do mouse events go?
Solution: Send to windowing system, let it decide which window is interested