ELEC 377 – Operating Systems

Week 8 – Class 3

Last Class

- File System Implementation
- ◊ I/O Hardware

Today

- File System Implementation
- I/O Hardware
- I/O Systems

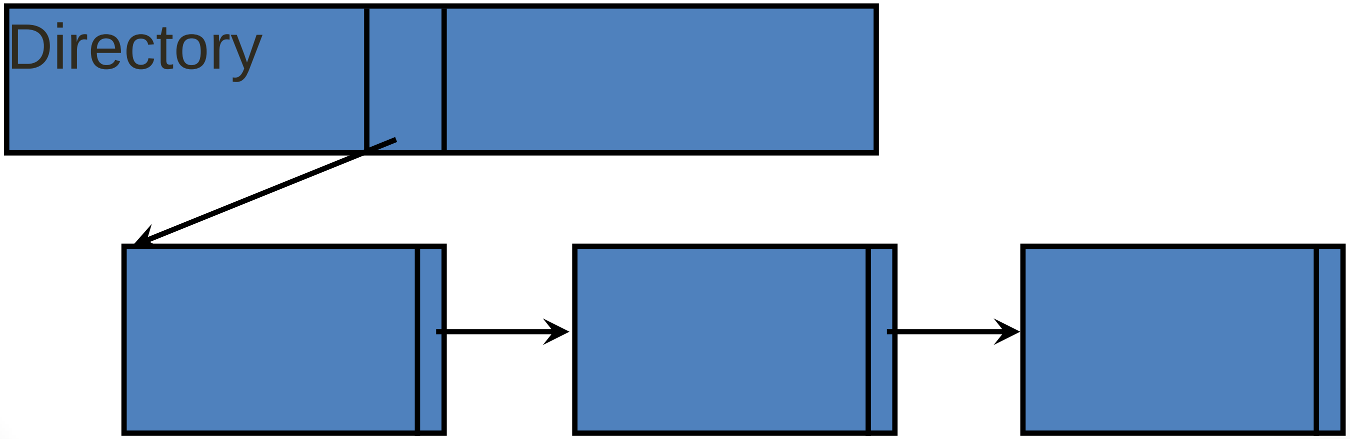
◊ Block and Character Devices

Allocating Disk Blocks to Files

- Contiguous
  - ◊ IBM VM/CMS - Data Set, Partitioned Data Set
  - ◊ blocks for a file are contiguous
  - ◊ directory contains starting block, length
  - ◊ fast for read/write
  - ◊ direct access is easy
  - ◊ problems with size, fragmentation (same as memory)

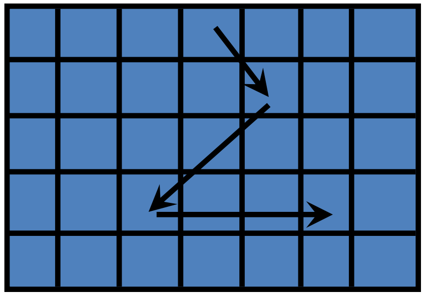
Allocating Disk Blocks to Files

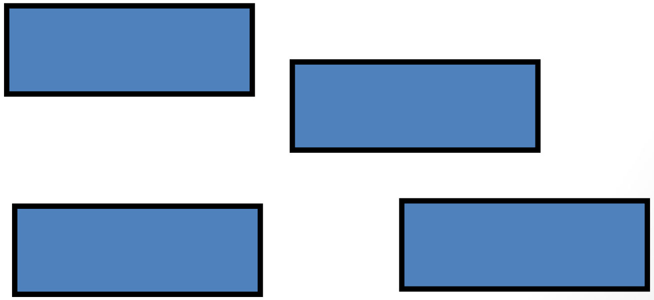
- Linked Allocation
  - ◊ directory gives first and maybe last block in file
  - ◊ each block has a pointer to next block
  - ◊ less data stored in each block (alt. FAT)
  - ◊ direct access is not nearly as easy (follow chain)



Allocating Disk Blocks to Files

- Linked Allocation
  - ◊ Fat Allocation
  - ◊ blocks grouped together into clusters (MS-DOS)
  - ◊ 16 bit - 65536 clusters
  - ◊ 128K reserved for link table (small enough to keep in ram)
  - ◊ table is indexed by block number, and gives the next block in the chain
  - ◊ data block now contains only data

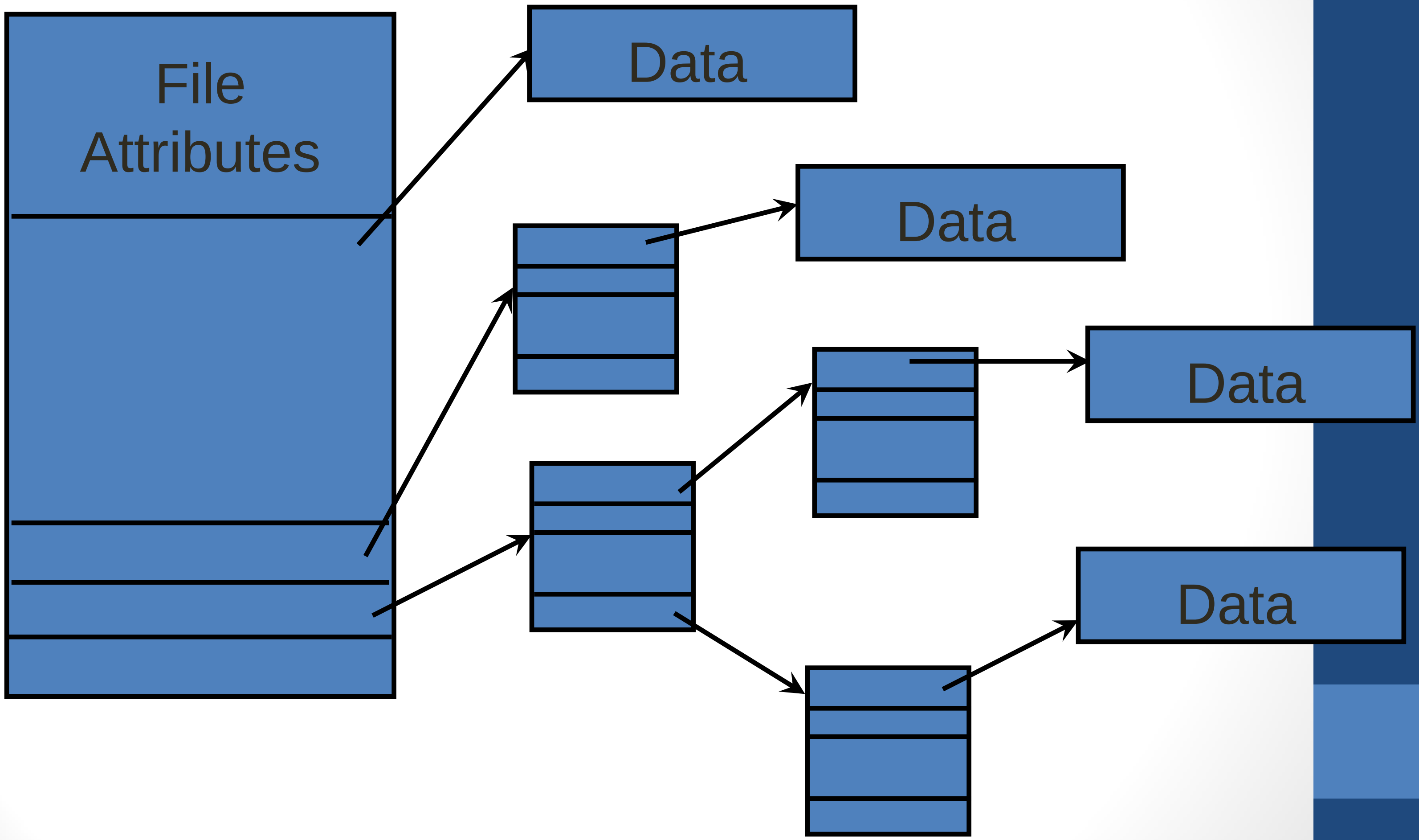




Allocating Disk Blocks to Files

- Indexed Allocation
  - ◊ Use one or more disk block for the file that contains the pointers to the data bocks
  - ◊ more overhead
  - ◊ direct access into file is easy
  - ◊ May need more than one index block
    - linked list
    - tree (internal nodes are index, leafs are data)
    - combined (Unix)
    - 80 - 20 distribution

Allocating Disk Blocks to Files

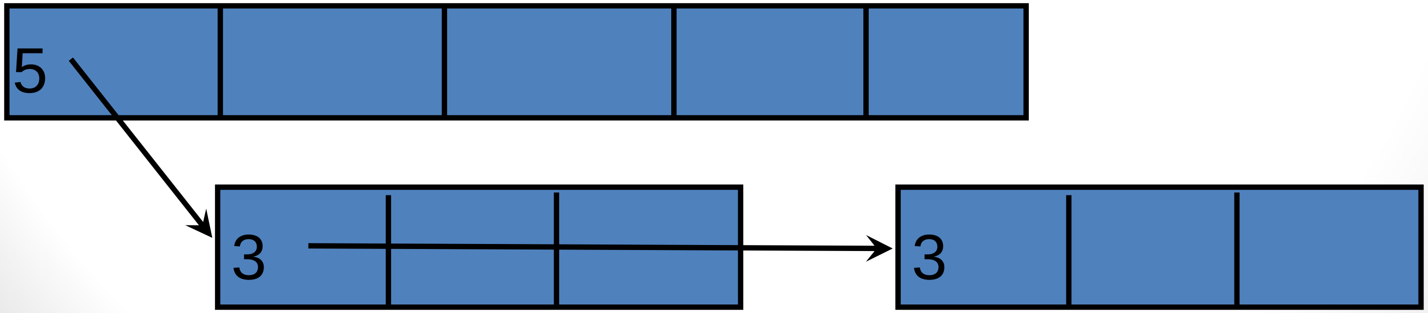


Free Space Management

- Reclaim lost space
  - ◊ Mainframes take easy way out (Track allocation, PDS regeneration)
  - ◊ keep track of free blocks (file alloc, file delete)
- Bit Vector (like FAT link map)
  - ◊ one or more blocks (overhead)
  - ◊ each bit represents a data block on the system
  - ◊ easy to get contiguous space
  - ◊ OK for smaller disks
- Linked List
  - ◊ link all free blocks into a list
  - ◊ no overhead

Free Space Management

- Grouping
  - ◊ lists of pointers (multi level)
  - ◊ link free blocks into a tree, group by disk location
- Counting
  - ◊ blocks often allocated and released in groups
  - ◊ usually several blocks together
  - ◊ First free block contains count of the number of contiguous blocks and a pointer to next group



Efficiency and Performance

- Efficiency
  - ◊ dependent on use of system
  - ◊ 80 - 20
  - ◊ fragmentation (three meanings)
- Performance
  - ◊ disk cache/page cache
    – unified virtual memory
  - ◊ unified buffer cache
  - ◊ sequential read - FIFO buffer replacement
    -free behind - free the page as soon as next page is accessed
    - read ahead - read next page before it is accessed
  - ◊ ram-disk

Recovery

- Consistency checking
  - ◊ fsck on unix
  - ◊ walk file system and make sure no errors
    - - blocks both in file and on free list
    - - inodes with incorrect dates
- Journaled File Systems
  - ◊ transaction based
  - ◊ reliable log is used
    - – usually several data blocks in the file system
    - – actions are written to log and put on disk before action is taken, then removed from log
    - – don't have to check entire disk, just look at files who have entries in the log

I/O Hardware

- Large Varieties of I/O devices
- Common Concepts
  - ◊ ports
  - ◊ Bus
    - daisy chain
    - shared direct access
  - ◊ Controller
- I/O instructions control the devices
- Devices have addresses
  - ◊ direct I/O instructions
  - ◊ memory mapped I/O
  - ◊ mixture

I/O Hardware

- Many buses that interact with each other
  - ◊ PCI bus
  - ◊ SCSI bus
  - ◊ IDE bus
- Communicate with controllers
  - ◊ Registers

    - status register

    - control/command register

    - input register(s)

    - output register(s)

Polling - busy waiting

- Status Register
  - ◊ busy bit - controller is busy
- Command Register
  - ◊ command ready bit

Device driver loops checking busy bit

- ◊ origin of term busy wait
- ◊ only useful if device is fast. If device is slow, then a lot of CPU cycles are wasted

Interrupts

- Interrupt current process on CPU
  - ◊ transfer to interrupt- handler
- Usually multiple interrupt vectors
  - ◊ limited number of interrupt vectors so some devices have to share the vector (handler chaining)
  - ◊ don’t have to check status register of every device
- Multiple levels of interrupts
  - ◊ priority of interrupts
  - ◊ some interrupts can be masked (disabled)
  - ◊ multiple level interrupts (used to divide handlers)
  - Interrupts also used for exceptions (divide by zero) and Traps for system calls

DMA

- Direct Memory Access
  - ◊ some devices (Disk Drives, Network Controllers) transfer data in blocks
- Transferring the data in and out of the controllers one byte at a time is waste of CPU cycles
  - ◊ give the controller access to the memory bus
  - ◊ DMA controller for the bus mediates the transfer
  - ◊ Controller tells DMA controller ready to transfer data, DMA holds memory bus for controller
  - ◊ CPU is interrupted when memory transfer is done
- DMA steals cycle from CPU access
  - ◊ slows down current process
  - ◊ gain from hardware (especially for VM)

DMA

- DMA controller might use physical address
  - ◊ OS does translation of addresses when loading DMA registers
  - ◊ buffers must be contiguous in physical memory
- DMA might use logical address
  - ◊ MMU between DMA controller and memory – buffers might not be contiguous in memory

Device Driver Interface

- View to the application program
  - ◊ view to OS
- Devices are abstracted into several classes of devices
  - ◊ abstract differences in devices
    - – IDE disk vs SCSI disc
    - – many “equivalent” devices
  - ◊ common I/O commands for other parts of OS (e.g. file system) and application programs
  - ◊ extend with mechanism for issuing special commands (ioctl in UNIX)
  - ◊ different approaches between different operating systems

Types of Devices

- Character Devices
  - ◊ byte at a time
  - ◊ USB, modem, keyboard, mouse
- Block devices
  - ◊ minimal unit of transfer is a block
  - ◊ ideal for DMA use
  - ◊ disk drives, tape drives, network interfaces

Other Parameters

- Sharable/Dedicated
  - ◊ more than one process can access at a time

- Device Speed
  - ◊ very wide range of speed

- I/O direction
  - ◊ CD-ROMs are read only.
  - ◊ Other devices are write only (some printers)

Network Interface

- Low level device driver is block driven (packets)
- Application level is character stream driven (TCP/IP)
  - ◊ sockets
  - ◊ look like files
- Application level is also block driven (UDP/IP)
- select()
  - ◊ In UNIX terminals look like files
  - ◊ programs had to read from more than one terminal at a time
  - ◊ extended to networks
- All other sorts of interfaces (FIFO’s, streams, queues, mailboxes, etc.)

Clocks and Timers

- Real Time
  - ◊ current time
  - ◊ usually only read at startup
  - ◊ elapsed time
  - ◊ INTERRUPTS

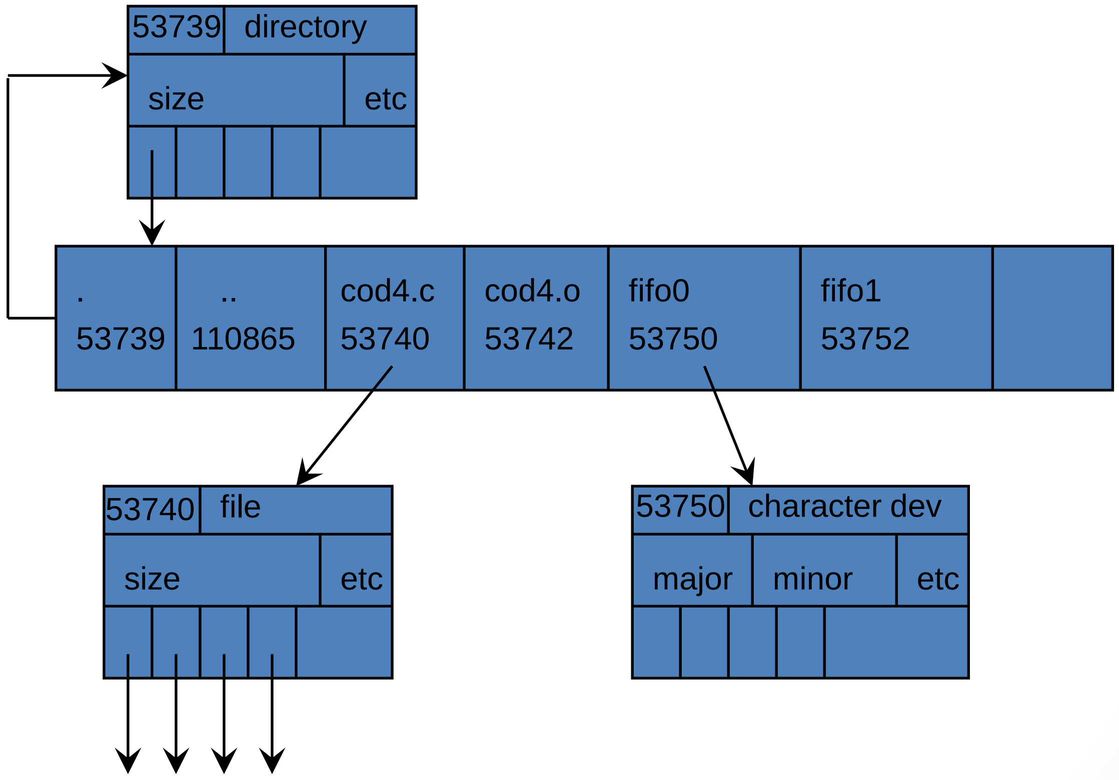
- OS Schedules timer interrupts
  - ◊ limited number of timers
  - ◊ Round Robin Scheduler need the timer as well
  - ◊ Used to run the OS Clock

/dev filesystem

- Direct access to devices under unix.
  - ◊ device access looks like a file
- Everything looks like a file under unix
  - ◊ inode attributes specify device (inode stores information about a file system object)
  - ◊ alternate layout of inode
  - ◊ no data blocks
  - ◊ major and minor numbers
    - major number specifies device driver
    - minor number specifies device
- Traditionally, these files are in /dev, but can be anywhere on the disk. (device id in inode)
  ◊ traditional names (hda - ide, sda- scsi, tty - console and serial, pty - network terminal) , but names can be anything, major number specifies driver

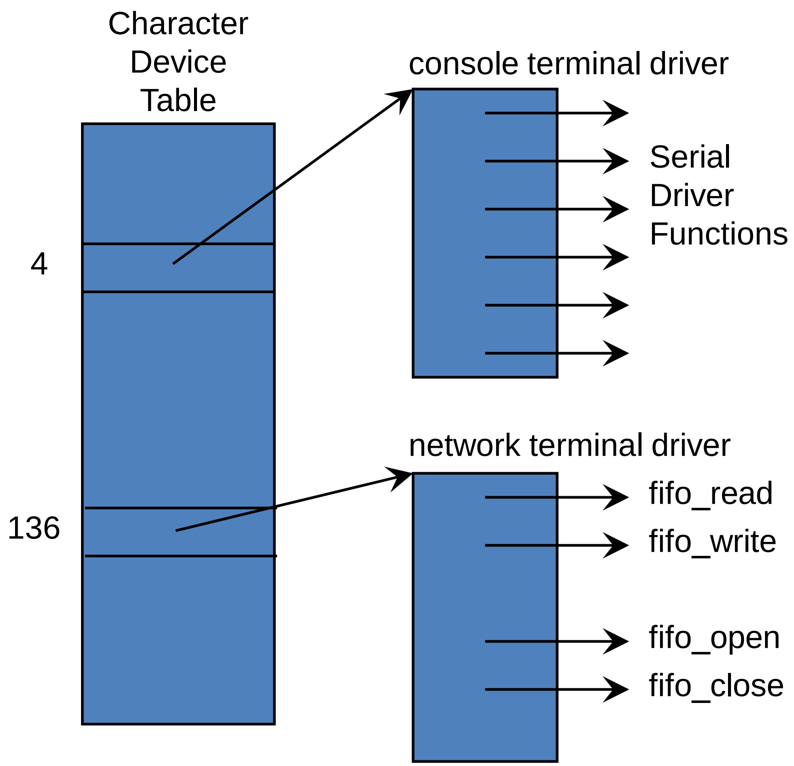
Directories and Files Structures

- inodes are the file control blocks in Unix
  - ◊ directories are files



Data Blocks for cod4.c

Character Device Drivers



- The major number connects the device file to the device driver.

  There can be multiple files with same major, same or different minors, all connect to the device driver

/dev filesystem examples

- Console tty (/dev/tty1 and 2 are Virtual Consoles)

```
crw--w----    1 shannon  tty        4,   1 Nov 10 09:39 /dev/tty1
crw-------    1 root     root       4,   2 Sep  9 10:57 /dev/tty2
```

◊ character device
◊ major = 4, minor = 1
◊ shannon is logged in to the first console
◊ minor specified which console (alt-F*n*)

- network pseudo tty

```
crw--w----    1 stephan tty      136,   2 Nov 10 09:40 /dev/pts/2
```

◊ used to simulate terminal for network connection
◊ console editors such as vi need terminals
◊ major = 136, minor = 2

/dev filesystem examples

- IDE drives

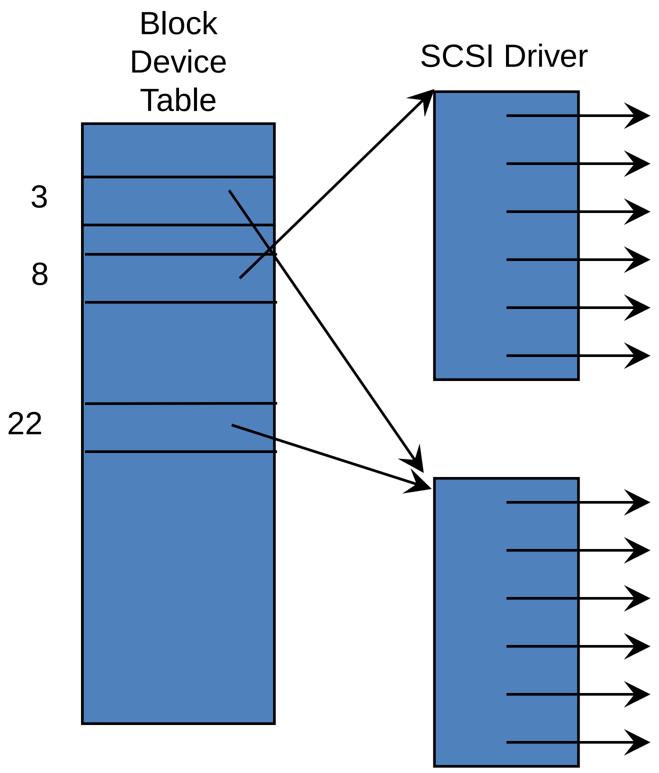
```
brw-rw----    1 root     disk       3,   0 Jan 30  2003 /dev/hda
brw-rw----    1 root     disk       3,   1 Jan 30  2003 /dev/hda1
brw-rw----    1 root     disk       3,  64 Jan 30  2003 /dev/hdb
brw-------    1 shannon  disk      22,  64 Jan 30  2003 /dev/hdd
```

◊ block device
◊ major = 3, minor = 1
◊ different major for second ide bus, but same driver
◊ more than one major may point to the same driver

- SCSI:

```
brwxrwxrwx    1 root     disk       8,   1 Jan 30  2003 /dev/sda1
```

Device Drivers with multiple Majors



- One of the parameters to the device driver is the major number. Thus the same device driver can be used for multiple majors

In the case of the IDE driver, Major 3 = IDE bus 1 and Major 22 = IDE BUS 2.