Scheduling of requests

Some I/O requests get queued waiting for service.
For some types of devices it is more efficient to service the requests out of order.

We may also schedule requests according to other criteria e.g. the priority of the process making the request.

Scheduling is commonly used when dealing with disks.

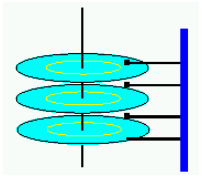
Disks are special devices

Disks are essential to most computing systems.

Disk devices are shared
Lots of requests
Accessed for lots of different reasons:

- Data about the layout and contents of the disk itself
  - size, name
  - partition maps
  - super blocks
  - inodes or MFTs
  - blocks in use
  - blocks free
- Holds the OS
  - e.g. boot information
- Virtual memory
- Communication buffers - spooling space
- Ordinary files

tracks
surfaces
cylinders
sectors
platters
boom
heads



Seek and rotational latency

Disk access

Each different use may use a different access method
e.g. It is usually not a good idea to implement virtual memory as an *ordinary* file. Both Windows and MacOS X use special files. Linux has traditionally used a swap device.

- Virtual memory needs to be as fast as possible.
- The extra overhead in dealing with ordinary files.
- contiguous allocation is faster than scattered allocation
- can then use simpler operations than normal system reads and writes

Similar arguments can be made for treating spooling areas as special

- they are always used sequentially
- they are transitory

Swap space management

We can deal with virtual memory space on the disks in the same way as ordinary files, but

- navigating directory structures takes time and multiple file accesses
- non-contiguous files mean seeking to load a process image (in systems which swap processes)

We can solve some of these problems by allocating contiguous disk space for the swap space and caching the block information to reduce the number of reads to find a particular block.

Separate swap partition (or device)

- there is no file system or directory structure on the partition
- read and write algorithms are optimised for speed, not space efficiency

Disk scheduling algorithms

Designed to minimise the seek time.

FCFS - first come first served

- fair - not efficient

SSTF - shortest seek time first

- more efficient (but not optimal)
- not fair (starvation)

SCAN - elevator algorithm

- like SSTF but only move in until reach the centre and then out, outside tracks are discriminated against
- but starvation can not occur

N-step SCAN

- only services requests which were present at the start of the sweep
  stops new requests slowing down older ones

C-SCAN - circular SCAN

- always goes one way and then zooms back to the beginning
- by the time the head reaches the end most requests are at the beginning
- doesn't discriminate against outside tracks

Which one?

Choosing one - depends on the load on the system.
Also depends on file layout - e.g. directories in the centre.

Some drive controllers can automatically schedule a queue of requests.

There is a trend to move useful algorithms from user-level program control, to device drivers (or the kernel), and finally into hardware.

This provides greater speed and efficiency but it is more expensive (initially) and is more restrictive.

e.g.

- usually dealing with virtual memory should have precedence over ordinary file access
- directories should be updated before files in order to improve robustness in the case of crashes

Other special things about disks

Formatting

Low-level formatting

Painting the lines in which the blocks get written. Done at the factory.

- each sector has a header, data area, and a trailer
- the header and trailer hold the sector number and error-correcting code based on the contents of the data area
- disk controller automatically updates and checks these values
- bad-blocks detected here are mapped out of the logical disk space
- spare sectors are allocated to be used when more blocks go bad over the lifetime of the disk

Partitioning disks

Why?

- because the lousy file system demands it
  - limitations on number of files
  - limitations on size of disk drives
  - forces too much space wasted due to number of logical blocks or clusters

Good reasons

- some disk management can be easier with smaller drives
- safer - a bad partition shouldn't affect files on the other partitions
- a berserk process can only fill up one virtual disk, not all the disk space
- can maintain and check one partition while others are still in use
- back up entire partitions
- want different file systems on the same device

It needs some sort of record which specifies which parts of the device correspond to which logical disk.

Logical formatting

Stores the file systems data structures on the disk.

- super blocks
- free and allocated blocks
- empty directory
- File system info

The blocks which hold information about the file system on the device are critical.

file system parameters

- size
- how many files it can store
- type and version information
- block size of data blocks
- free blocks
- directories
- file information

In UNIX some of this information is stored in a structure called a superblock.

NTFS metadata

Every allocated block on an NTFS volume belongs to a file.

All metadata in NTFS is stored in files.

Master file table	$Mft
Master file table mirror	$MftMirr (first four records)
Log file	$LogFile
Volume	$Volume (label & version)
Attribute definitions	$AttrDef
Root file name index	. (root directory)
Cluster bitmap	$Bitmap (free clusters)
Boot sector	$Boot (bootstrap code)
Bad cluster file	$BadClus
Security file	$Secure (descriptors for files)

etc.

MFT Zone

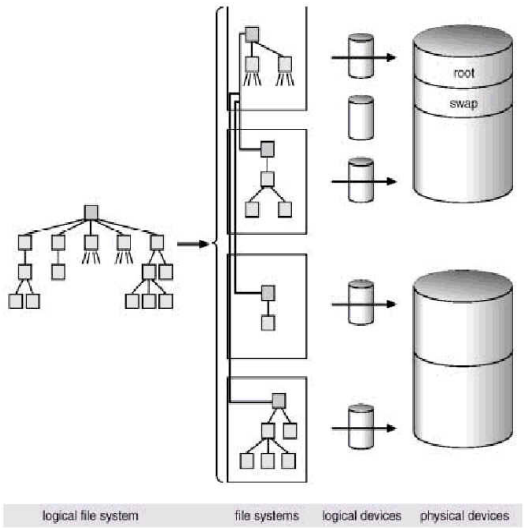
To prevent the MFT from becoming fragmented, NTFS reserves 12.5 percent of volume by default for exclusive use of the MFT. This space, known as the MFT zone, is not used to store data unless the remainder of the volume becomes full.

Depending on the average file size and other variables, as the disk fills to capacity, either the MFT zone or the unreserved space on the disk becomes full first.

Volumes that have a small number of large files exhaust the unreserved space first.

Volumes with a large number of small files exhaust the MFT zone space first.

UNIX disk structure



About the exam

Passing

- Pass mark required in assignments 25%
- Pass mark required in tests/exam ≈ 50%
- Total mark required (20% assignments, 10% test, 70% exam) ≈ 50%

Structure

- 60 marks, 57 questions
- Like the test and exams since 2012
- All Multi-choice questions.

Topics – all sections of the course, but concentrating on:

- OS history/design/implementation [6 marks]
- Virtualization [3 marks]
- Processes, Concurrency & Scheduling [15 marks]
- Deadlock [2 marks]
- File Systems [9 marks]
- Assignment 2 [5 marks]
- Memory Management [7 marks]
- Distributed File Systems & Services [4 marks]
- Protection and Security [8 marks]
- Devices [2 marks]

Before next time (What next time?)

All the best with your study and the exam.