## Filesystems

To wrap up filesystems here are a few remarks

## Filesystems

To wrap up filesystems here are a few remarks

We don’t have time to go into how disks work (SSD or spinning), for example, how data blocks are managed on the medium

## Filesystems

To wrap up filesystems here are a few remarks

We don’t have time to go into how disks work (SSD or spinning), for example, how data blocks are managed on the medium

But be aware this is also a lot of complicated detail!

## Filesystems

Next:

- As disks are relatively slow, there are caching tricks used to help speed things up

## Filesystems

Next:

- As disks are relatively slow, there are caching tricks used to help speed things up
- *Disk caching* is using “spare” memory to keep a cache copy of some of the disk contents

## Filesystems

Next:

- As disks are relatively slow, there are caching tricks used to help speed things up
- *Disk caching* is using “spare” memory to keep a cache copy of some of the disk contents
- The speed benefits are huge as long as you balance the use of memory for cache against the use of memory for everything else

## Filesystems

Next:

- As disks are relatively slow, there are caching tricks used to help speed things up
- *Disk caching* is using “spare” memory to keep a cache copy of some of the disk contents
- The speed benefits are huge as long as you balance the use of memory for cache against the use of memory for everything else
- With a good filesystem a program will use the disk just once or twice and spend most of its time using the disk cache (barely a “file” system at all!)

## Filesystems

Next:

- As disks are relatively slow, there are caching tricks used to help speed things up
- *Disk caching* is using “spare” memory to keep a cache copy of some of the disk contents
- The speed benefits are huge as long as you balance the use of memory for cache against the use of memory for everything else
- With a good filesystem a program will use the disk just once or twice and spend most of its time using the disk cache (barely a “file” system at all!)
- And then there is memory mapped disks as previously mentioned

## Filesystems

### Reliability

- Filesystems *must* be bug free. We can get away with the occasional bug elsewhere, but if the user loses their data this is a big problem

## Filesystems
### Reliability

- Filesystems *must* be bug free. We can get away with the occasional bug elsewhere, but if the user loses their data this is a big problem
- Sometimes we have to put up with a little data loss (e.g., power loss during a write), but we must *never* have data corruption

## Filesystems
### Reliability

- Filesystems *must* be bug free. We can get away with the occasional bug elsewhere, but if the user loses their data this is a big problem
- Sometimes we have to put up with a little data loss (e.g., power loss during a write), but we must *never* have data corruption
- Thus filesystem programmers tend to be very conservative: only well-tested systems should be used

## Filesystems

### Reliability

- Filesystems *must* be bug free. We can get away with the occasional bug elsewhere, but if the user loses their data this is a big problem
- Sometimes we have to put up with a little data loss (e.g., power loss during a write), but we must *never* have data corruption
- Thus filesystem programmers tend to be very conservative: only well-tested systems should be used
- If there is any corruption in the filesystem structure data can be lost

## Filesystems

### Reliability

- Filesystems *must* be bug free. We can get away with the occasional bug elsewhere, but if the user loses their data this is a big problem
- Sometimes we have to put up with a little data loss (e.g., power loss during a write), but we must *never* have data corruption
- Thus filesystem programmers tend to be very conservative: only well-tested systems should be used
- If there is any corruption in the filesystem structure data can be lost
- We can't rely on users making backups!

## Filesystems

### Reliability

- Unfortunately, there can be external influences, such as power loss just as inode pointers are being updated

## Filesystems
### Reliability

- Unfortunately, there can be external influences, such as power loss just as inode pointers are being updated
- This could leave the filesystem in an inconsistent state

## Filesystems
### Reliability

- Unfortunately, there can be external influences, such as power loss just as inode pointers are being updated
- This could leave the filesystem in an inconsistent state
- Modern filesystems try very hard never to let this happen

## Filesystems

### Reliability

- Unfortunately, there can be external influences, such as power loss just as inode pointers are being updated
- This could leave the filesystem in an inconsistent state
- Modern filesystems try very hard never to let this happen
- So things like *transactional*, *log structured* and *journalling* filesystems have been created

## Filesystems
### Reliability

- Unfortunately, there can be external influences, such as power loss just as inode pointers are being updated
- This could leave the filesystem in an inconsistent state
- Modern filesystems try very hard never to let this happen
- So things like *transactional*, *log structured* and *journalling* filesystems have been created
- Similarly, disk technology is very good these days, but disks still have problems

## Filesystems
### Reliability

- Unfortunately, there can be external influences, such as power loss just as inode pointers are being updated
- This could leave the filesystem in an inconsistent state
- Modern filesystems try very hard never to let this happen
- So things like *transactional*, *log structured* and *journalling* filesystems have been created
- Similarly, disk technology is very good these days, but disks still have problems
- So there are hardware monitoring systems like SMART that watch a disk for impending problems

## Filesystems

### Reliability

- And filesystems that check data for errors as it is read

□

## Filesystems

### Reliability

- And filesystems that check data for errors as it is read
- And collections of disks in a *redundant array of inexpensive disks* (RAID) that spreads data across multiple disks so that if one fails the data is retrievable

□

## Filesystems

### Reliability

- And filesystems that check data for errors as it is read
- And collections of disks in a *redundant array of inexpensive disks* (RAID) that spreads data across multiple disks so that if one fails the data is retrievable
- And so on

□

## Filesystems

### Reliability

- And filesystems that check data for errors as it is read
- And collections of disks in a *redundant array of inexpensive disks* (RAID) that spreads data across multiple disks so that if one fails the data is retrievable
- And so on

A lot of research has been put into filesystems: and it's still ongoing

□

## Filesystems

### Reliability

- And filesystems that check data for errors as it is read
- And collections of disks in a *redundant array of inexpensive disks* (RAID) that spreads data across multiple disks so that if one fails the data is retrievable
- And so on

A lot of research has been put into filesystems: and it's still ongoing

Parkinson's Law: data expands to fill the space available

□