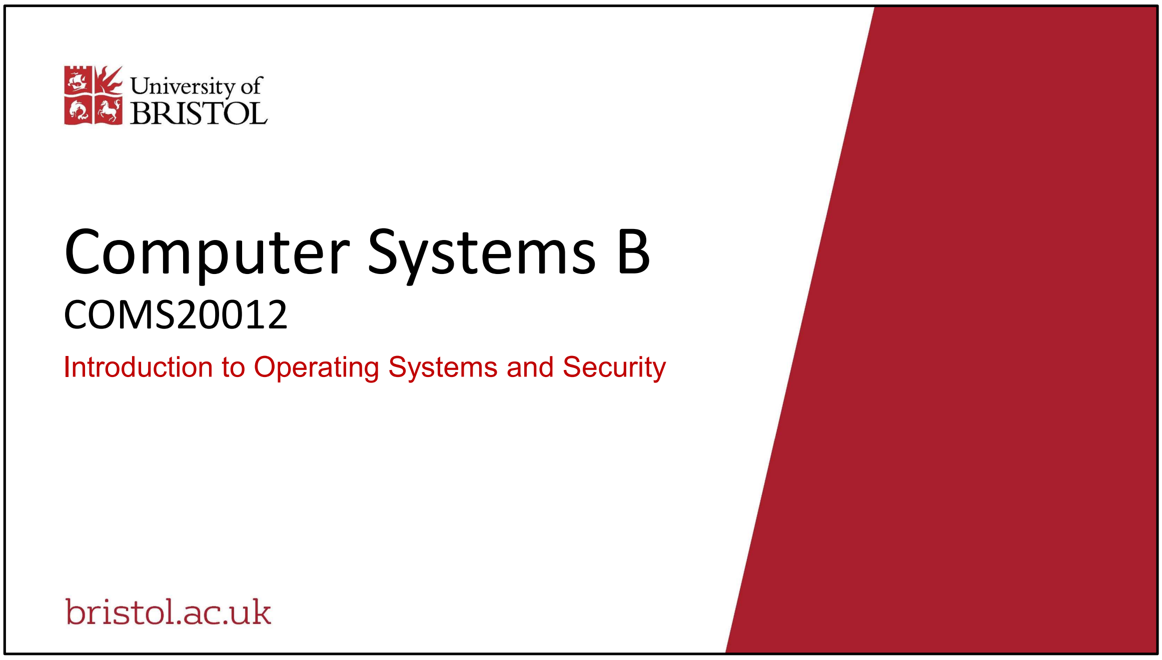


Computer Systems B

COMS20012

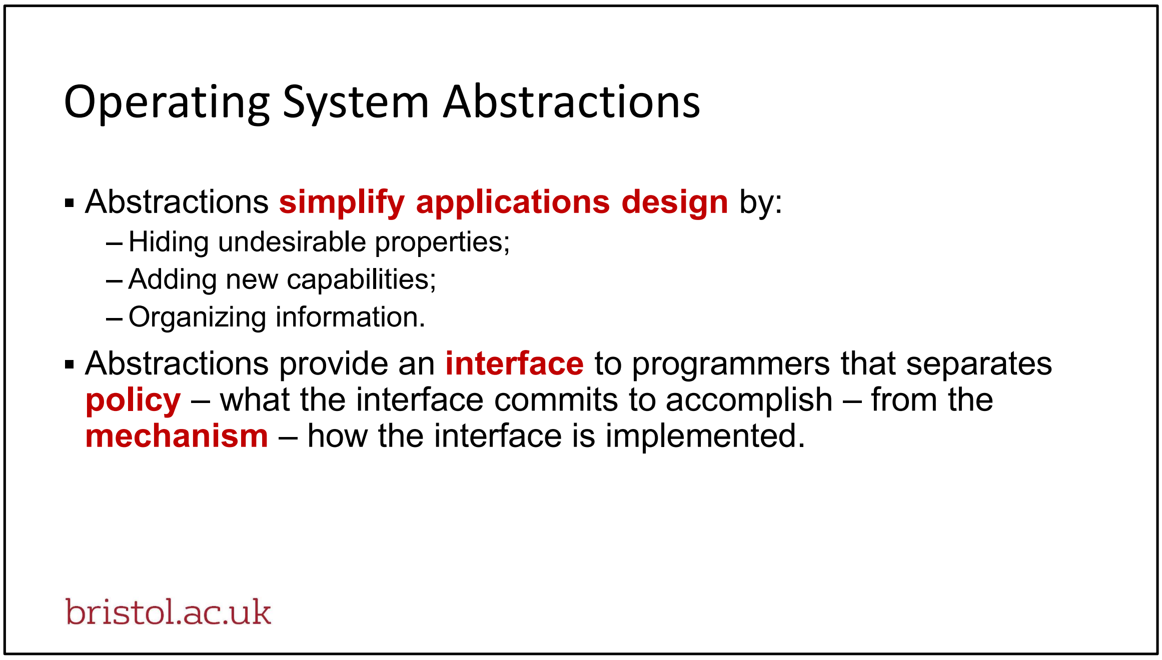
Introduction to Operating Systems and Security

bristol.ac.uk

University of BRISTOL

The process abstraction

bristol.ac.uk

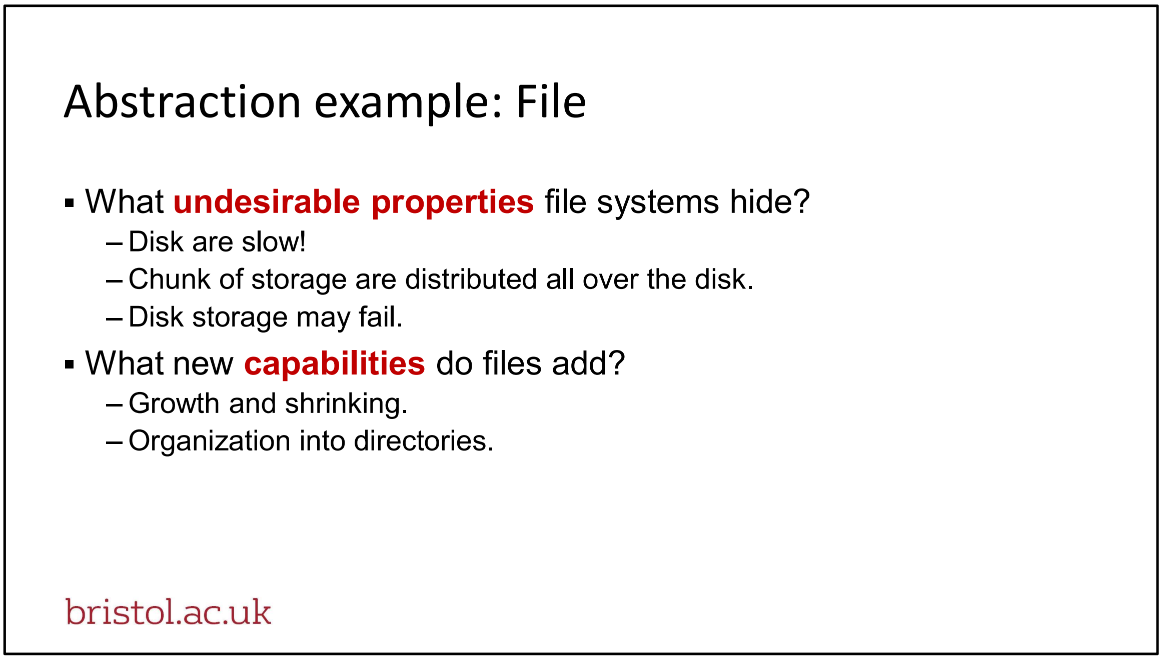


Operating System Abstractions

- Abstractions **simplify applications design** by:
  - Hiding undesirable properties;
  - Adding new capabilities;
  - Organizing information.
- Abstractions provide an **interface** to programmers that separates **policy** – what the interface commits to accomplish – from the **mechanism** – how the interface is implemented.

Abstraction example: File

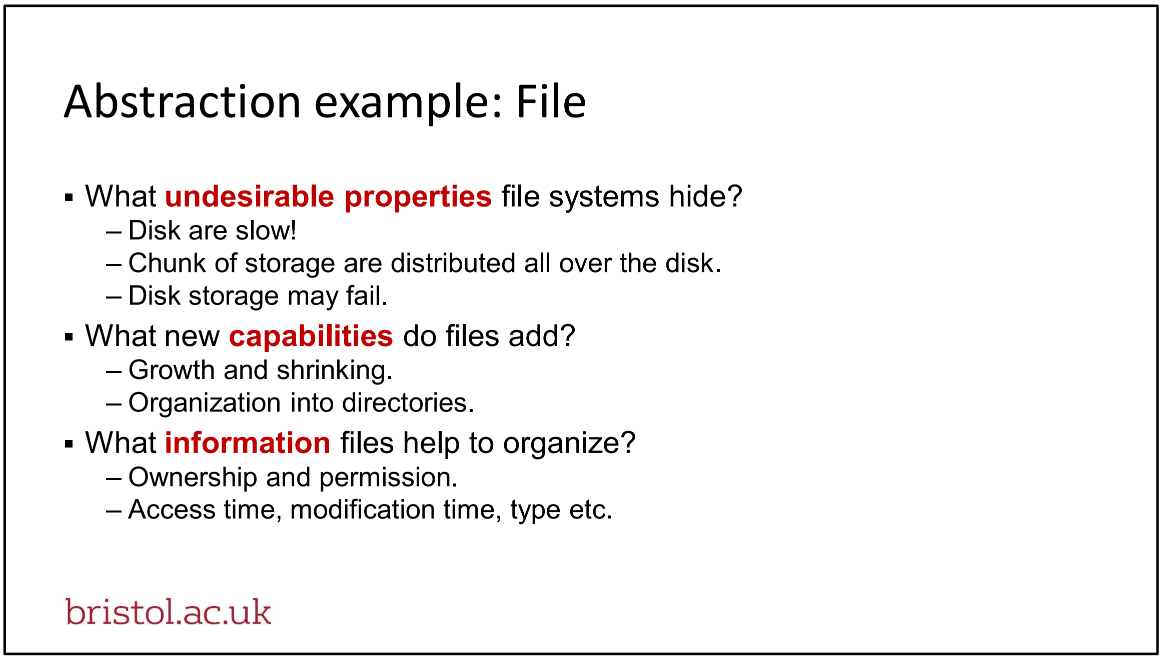
- What **undesirable properties** file systems hide?
  - Disk are slow!
  - Chunk of storage are distributed all over the disk.
  - Disk storage may fail.



Abstraction example: File

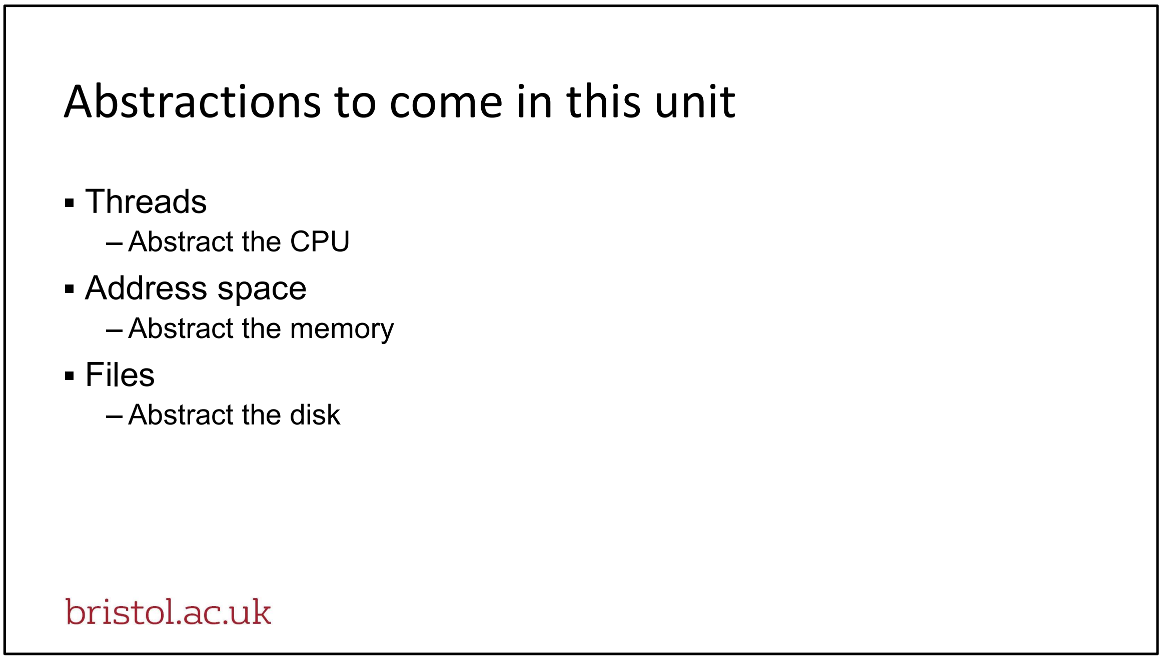
- What **undesirable properties** file systems hide?
  - Disk are slow!
  - Chunk of storage are distributed all over the disk.
  - Disk storage may fail.
- What new **capabilities** do files add?
  - Growth and shrinking.
  - Organization into directories.

bristol.ac.uk



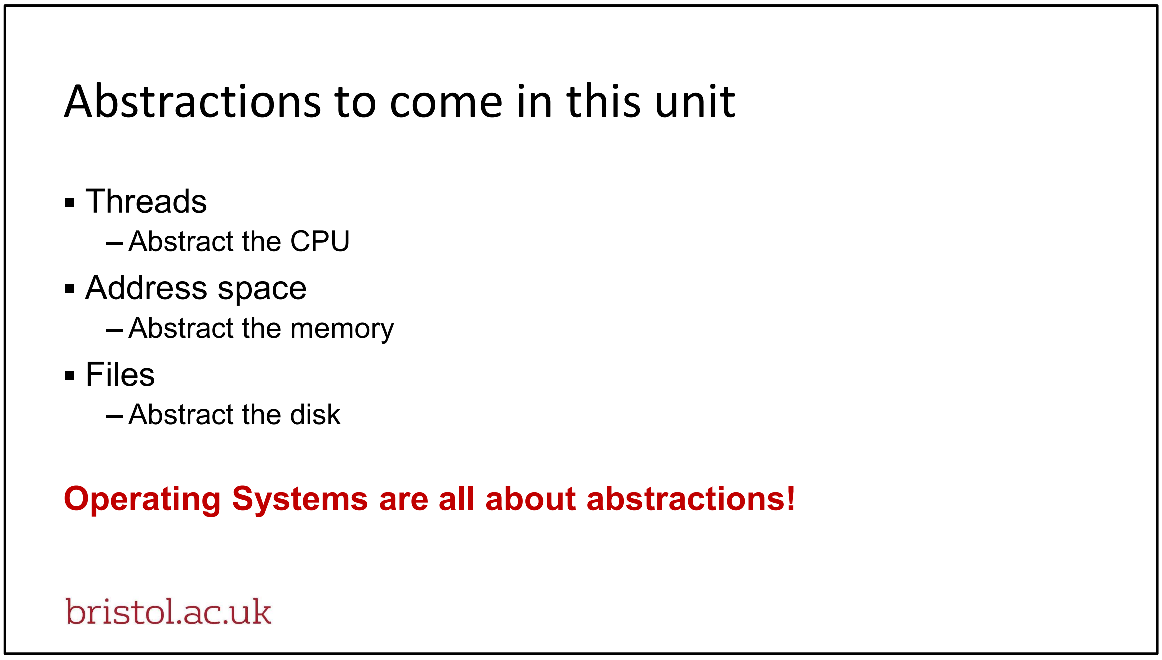
Abstraction example: File

- What **undesirable properties** file systems hide?
  - – Disk are slow!
  - – Chunk of storage are distributed all over the disk.
  - – Disk storage may fail.
- What new **capabilities** do files add?
  - – Growth and shrinking.
  - – Organization into directories.
- What **information** files help to organize?
  - – Ownership and permission.
  - – Access time, modification time, type etc.



Abstractions to come in this unit

- Threads
  - Abstract the CPU
- Address space
  - Abstract the memory
- Files
  - Abstract the disk

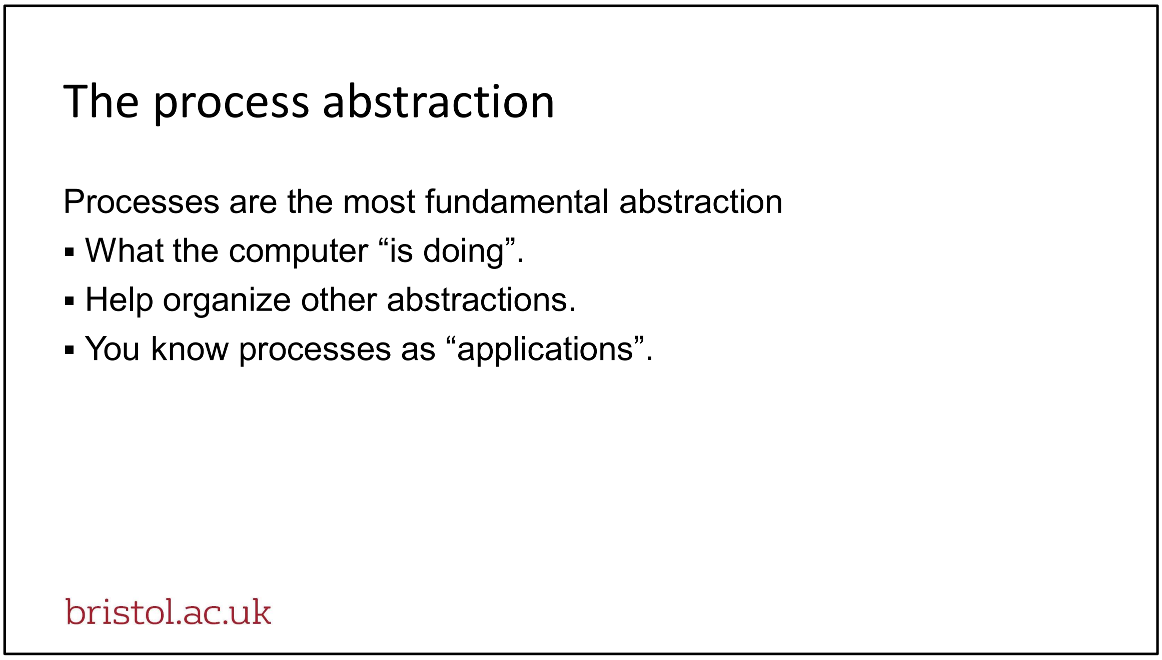


Abstractions to come in this unit

- Threads
  - Abstract the CPU
- Address space
  - Abstract the memory
- Files
  - Abstract the disk

Operating Systems are all about abstractions!

bristol.ac.uk

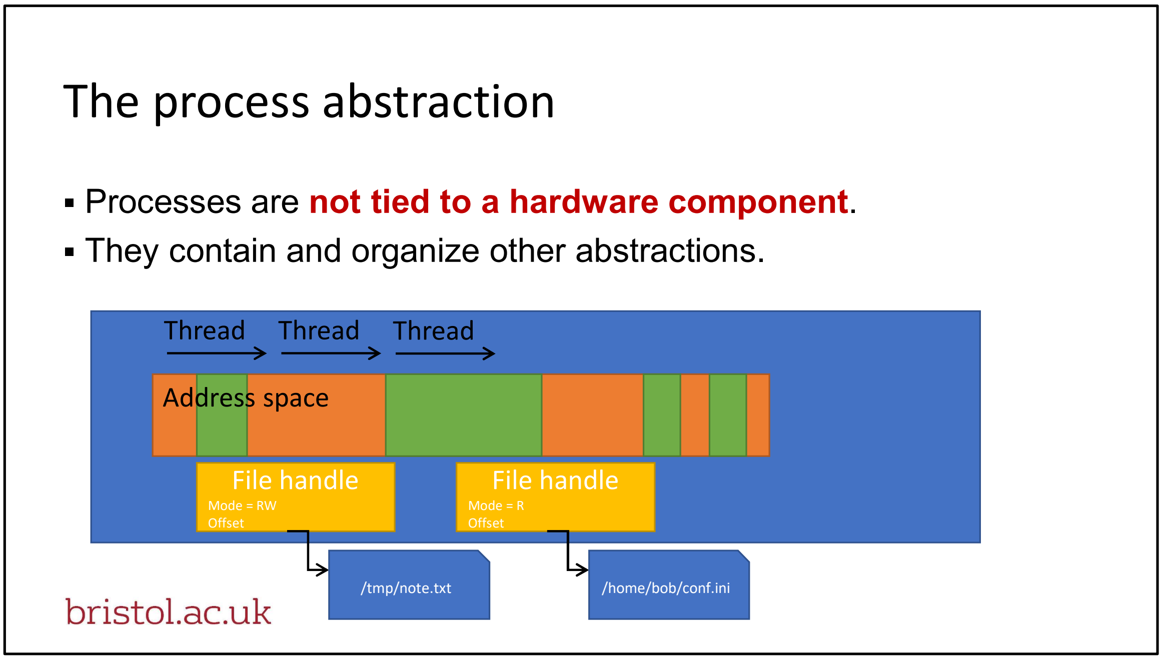


The process abstraction

Processes are the most fundamental abstraction

- What the computer “is doing”.
- Help organize other abstractions.
- You know processes as “applications”.

bristol.ac.uk



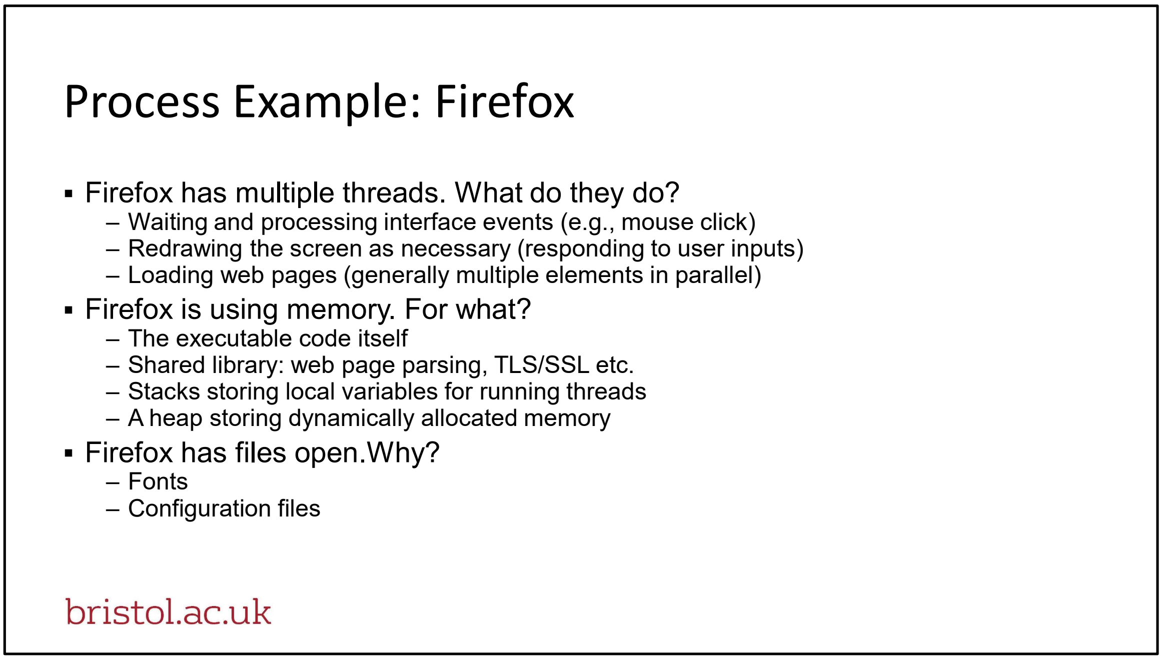
The process abstraction

- Processes are **not tied to a hardware component**.
- They contain and organize other abstractions.

bristol.ac.uk

Processes vs Threads

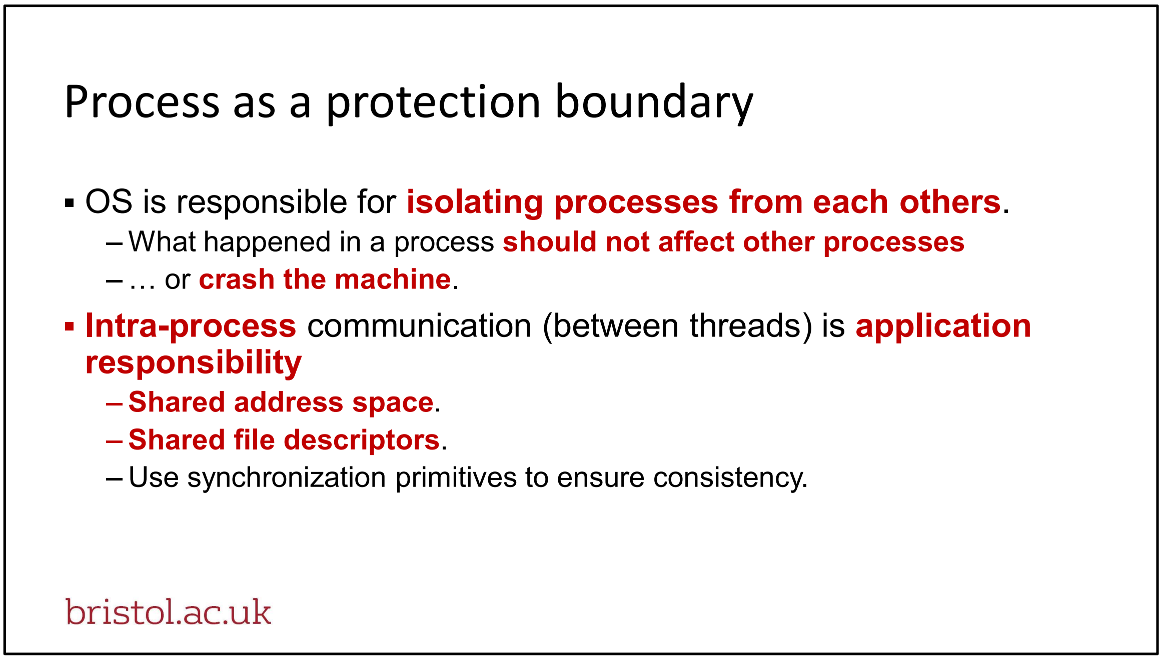
- Potentially confusing due to terminology
  - both described as **running**
- Some terminology useful to remember the distinction
  - Processes require multiple resources: CPU, memory, files
  - Threads abstract the CPU
- A process contains threads, threads belong to a process
  - Except kernel threads who do not belong to a user space process
- A process is running when one or more of its threads are running
- Terminology may vary between OSes but concepts are the same



Process Example: Firefox

- Firefox has multiple threads. What do they do?
  - Waiting and processing interface events (e.g., mouse click)
  - Redrawing the screen as necessary (responding to user inputs)
  - Loading web pages (generally multiple elements in parallel)
- Firefox is using memory. For what?
  - The executable code itself
  - Shared library: web page parsing, TLS/SSL etc.
  - Stacks storing local variables for running threads
  - A heap storing dynamically allocated memory
- Firefox has files open.Why?
  - Fonts
  - Configuration files

bristol.ac.uk

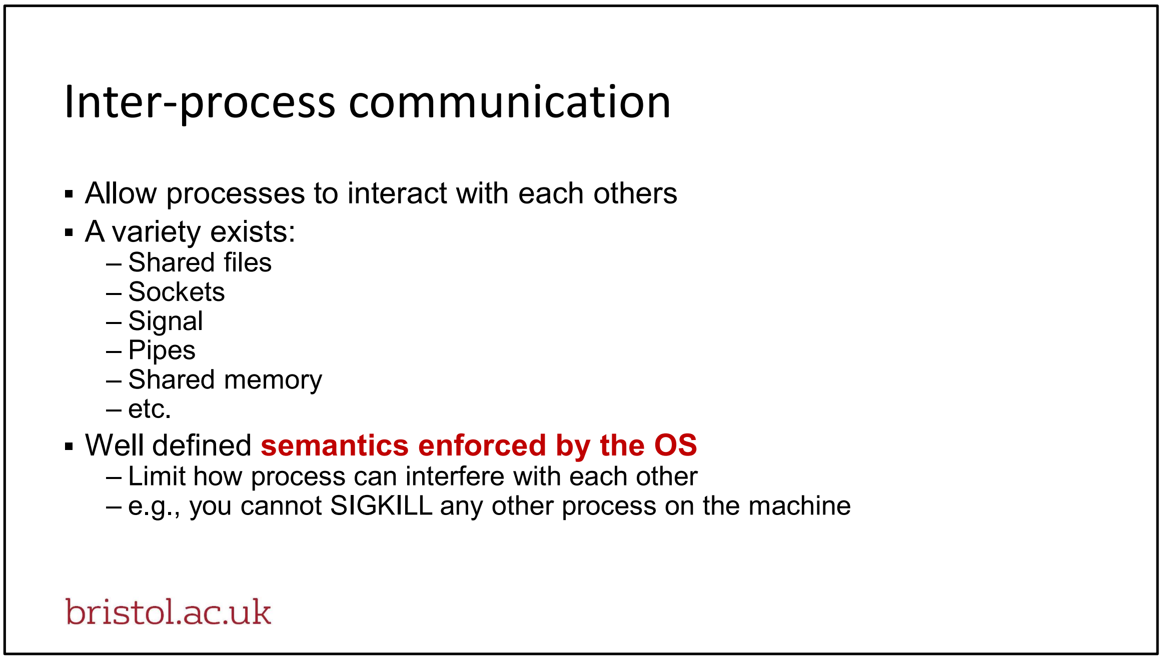


Process as a protection boundary

- OS is responsible for **isolating processes from each others**.
  - What happened in a process **should not affect other processes**
  - … or **crash the machine**.
- **Intra-process** communication (between threads) is **application responsibility**
  - **Shared address space**.
  - **Shared file descriptors**.
  - Use synchronization primitives to ensure consistency.

Process as a protection boundary

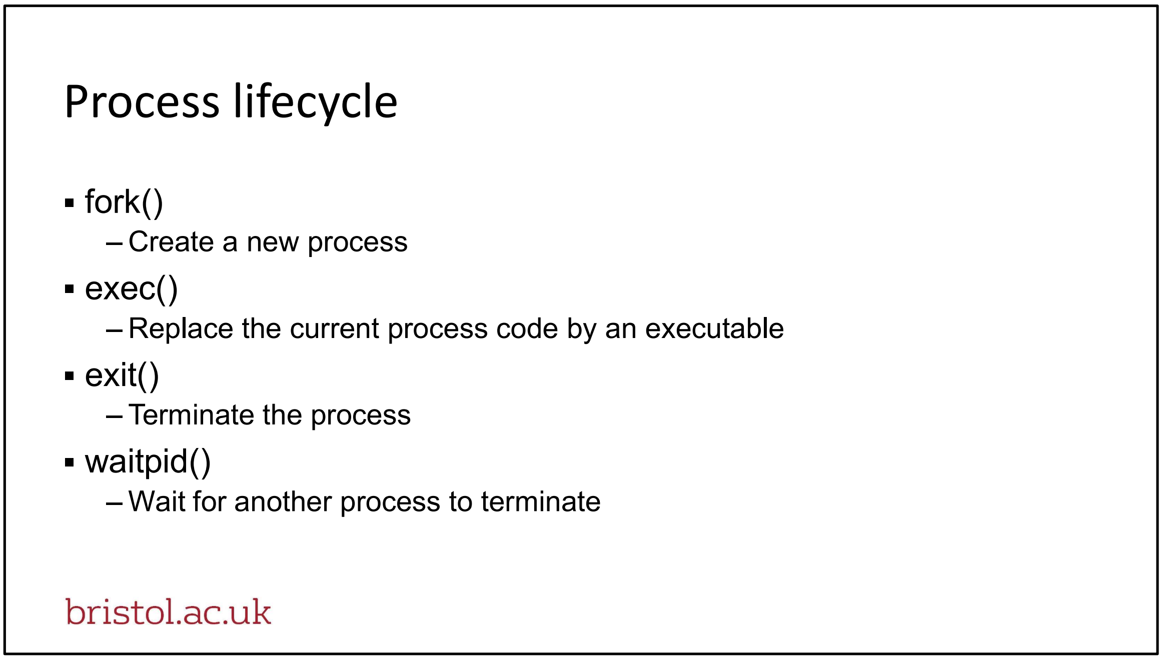
- OS is responsible for **isolating processes from each others**.
  - What happened in a process **should not affect other processes**
  - … or **crash the machine**.
- **Intra-process** communication (between threads) is **application responsibility**
  - **Shared address space**.
  - **Shared file descriptors**.
  - Use synchronization primitives to ensure consistency.
- See Computer Systems A



Inter-process communication

- Allow processes to interact with each others
- A variety exists:
  - Shared files
  - Sockets
  - Signal
  - Pipes
  - Shared memory
  - etc.
- Well defined **semantics enforced by the OS**
  - Limit how process can interfere with each other
  - e.g., you cannot SIGKILL any other process on the machine

bristol.ac.uk



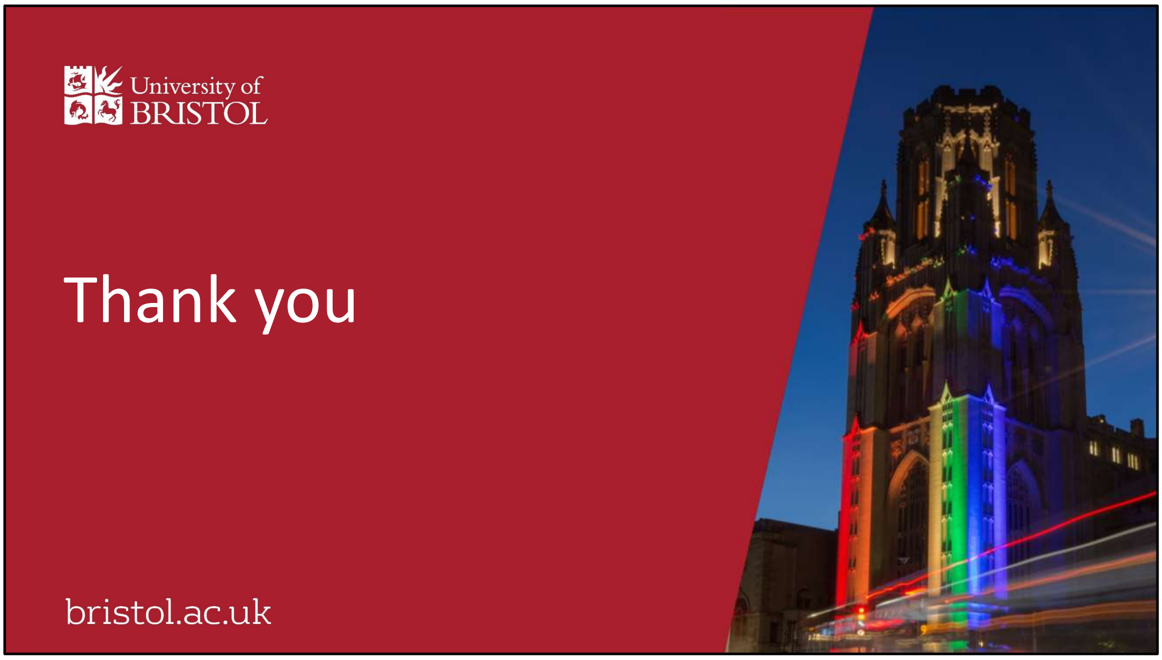
Process lifecycle

- fork()
  - Create a new process
- exec()
  - Replace the current process code by an executable
- exit()
  - Terminate the process
- waitpid()
  - Wait for another process to terminate

Process lifecycle

- fork()
  - Create a new process
- exec()
  - Replace the current process code by an executable
- exit()
  - Terminate the process
- waitpid()
  - Wait for another process to terminate

To be implemented in Lab 7



University of BRISTOL

Thank you

bristol.ac.uk