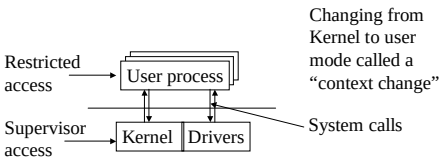
OS Scheduling/Buffering

- Scheduling
  - Kernel/user space
  - Processes/threads
  - Interrupts
  - Non-preemptive/Preemptive
  - Realtime
- Buffering
  - Block based
  - Shared memory/Memory mapping
  - Network smoothing
  - Playback buffers

Kernel/user space



Processes/Threads

- Process
  - a task being run (often simultaneously) with other processes
  - Must take turns on the CPU (timeslicing) unless there are multiple CPUs (multiprocessing – processes may be allocated to CPUs)
  - Managed by the OS (scheduled)
  - Kept apart to reduce chance of system failure (but may communicate by IPC – Inter Process Communication)
- Threads
  - Process splits itself into two or more simultaneously running tasks
  - Share process state (are more dependent)
  - Share memory directly
  - Faster context switching

Interrupts

- Signal (event) from a device to the processor
- Causes the processor to stop what it is doing and execute an "interrupt handler" (context switch)
- Can be from audio device, timer…
- Timer interrupt increments a clock, used by the kernel to switch processes
- Can be disabled (except "non-maskable interrupts") to stop interrupts interrupting other interrupts!
- Interrupt routine should be fast (another interrupt could be on the way)

(Non-)preemptive scheduling

- Non-preemptive multitasking
  - OS does not try to guess when a process has finished (does not pre-empt it)
- Pre-emptive multitasking
  - OS "interrupts" a process (even if it is not complete) and gives control to another process
  - Hardware interrupts can also pre-empt a process
  - Eg Windows, Linux…

Realtime Scheduling

- A Real Time Operating System (RTOS), is an operating system that has been developed for real-time applications.
- Typically used for embedded applications they usually have the following characteristics:
  - Small footprint (doesn't use much memory)
  - Pre-emptable (any hardware event can cause a task to run)
  - Multi-architecture (code ports to another type of CPU)
  - Many have predictable response-times to electronic events
- Minimise interrupt disable period
- Application can request delay, jitter bounds. OS schedules processes according to requests, to meet application requirements.

Buffering

- Block based
  - Device/driver handles data in blocks
  - more efficient than, eg, single bytes: less interrupts/OS calls
  - Bigger blocks = more efficient but more delay
  - Smaller blocks = less efficient but less delay
  - Codecs sometimes restrict choice (eg lpc, mp3…)

Shared memory/Memory mapping

- Shared memory
  - Processes/threads communicate/exchange data via memory accessible to all processes
  - Processes have to manage access (semaphores)
- Memory mapping
  - An area of memory used by the kernel is mapped into user space (user process can access directly)
  - Eg Audio buffer, Framebuffer (video)

Network smoothing

- Variable rate output from a codec (eg I-frames, P-frames, B-frames)
- Constant bitrate output from a network interface
- Network buffers “smooth” the codec output to fit the network capacity
- Causes some delay

Playout buffers

- Receiver receives data with jitter (from network/OS scheduling…)
- Adding a short delay (holding the data in a buffer) may smooth the jitter
- Buffer size depends on type of application (interactive/non-interactive)
- Interactive buffer size=min(“worst case jitter seen so far”, 150ms) (may be too conservative)
- Non-interactive buffer size=0.5s (allows for reasonable speed channel hopping)