# Lecture Summaries

## C:\Users\Hp\Downloads\cricil\lecture notes\input\l6.pdf

\* \*\*Process Creation and Termination:\*\* Processes are created hierarchically, forming a tree. A process terminates via `exit()`, deallocating resources; parents can terminate children via `abort()`. Cascading termination occurs if a parent process ends. Orphaned processes result from parent termination without `wait()`.  
  
\* \*\*Context Switching:\*\* Saving and loading process states (from the Process Control Block, PCB) enables multitasking. Context-switch time is system overhead.  
  
\* \*\*Mobile System Multitasking:\*\* Mobile OSes may support single foreground processes or multiple background processes with limitations on resource use and display.  
  
\* \*\*Multi-process Applications:\*\* Examples like Google Chrome utilize multiple processes (browser, renderer, plugin) for enhanced stability and functionality.  
  
\* \*\*Inter-Process Communication (IPC):\*\* IPC mechanisms (message passing and shared memory) enable cooperating processes to share data and synchronize actions. Message passing involves `send()` and `receive()` operations. Shared memory requires user-level synchronization.  
  
\* \*\*IPC Models:\*\* Direct communication explicitly names processes, while indirect communication uses mailboxes. Both can be unidirectional or bidirectional. Message buffering (zero, bounded, unbounded capacity) affects synchronization.  
  
\* \*\*Synchronization:\*\* Message passing can be blocking (synchronous) or non-blocking (asynchronous). Synchronization primitives manage access to shared resources.  
  
\* \*\*Producer-Consumer Problem:\*\* Illustrates process synchronization challenges; different buffer sizes (unbounded, bounded) introduce varying complexities.