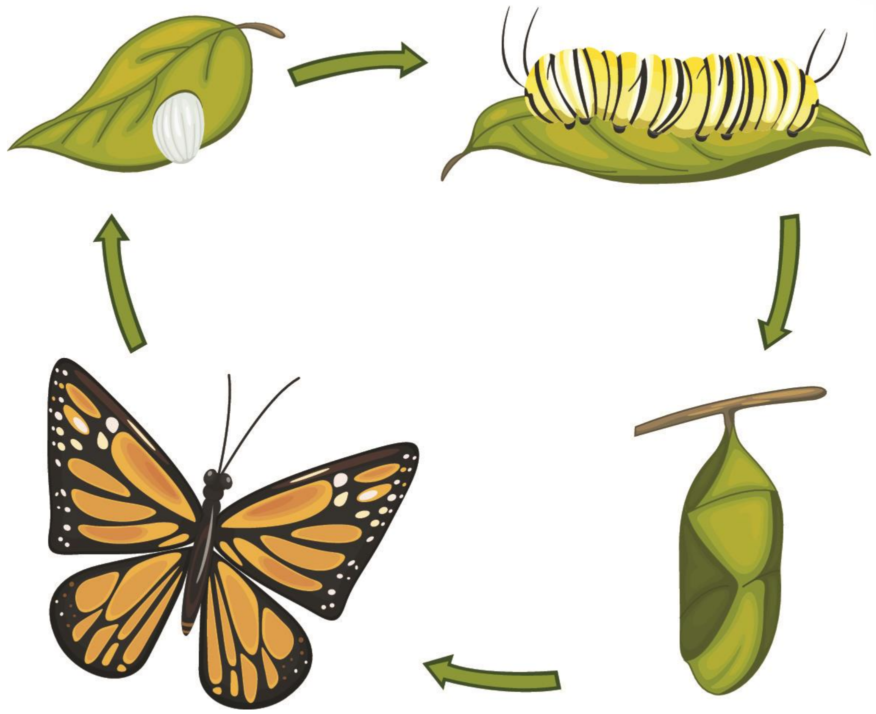
Linux For Embedded Systems

For Arabs

Course 102:
Understanding Linux

Ahmed ElArabawy



Lecture 18:
Process Life-Cycle

PROCESS CREATION

Fork-Exec Procedure

- Creating a new process happens in two steps
  - Fork:

    In this step the new process (child process) is created and inherits its features from its parent
  - Execute:

    In this step the new process (child process) separates from its parent and move on with its functionality

Fork

Execute

Fork

- The Job of "***Fork***" in is to create a new process (child process) and prepare it with what it needs to be able to run….
- This includes,
  - Assign it a new pid
  - Creates associated Structures in the Kernel
  - Sets the ppid, pgid, sid, and any other attributes
  - Add it to the Kernel scheduler queue to give it time slots to run
  - Copy parent memory space (all memory associated with the parent process) to the child process memory space
  - Copy Other parent process resources,
    - Normal file handlers
    - Socket handlers
    - Shared memory handlers
    - ….

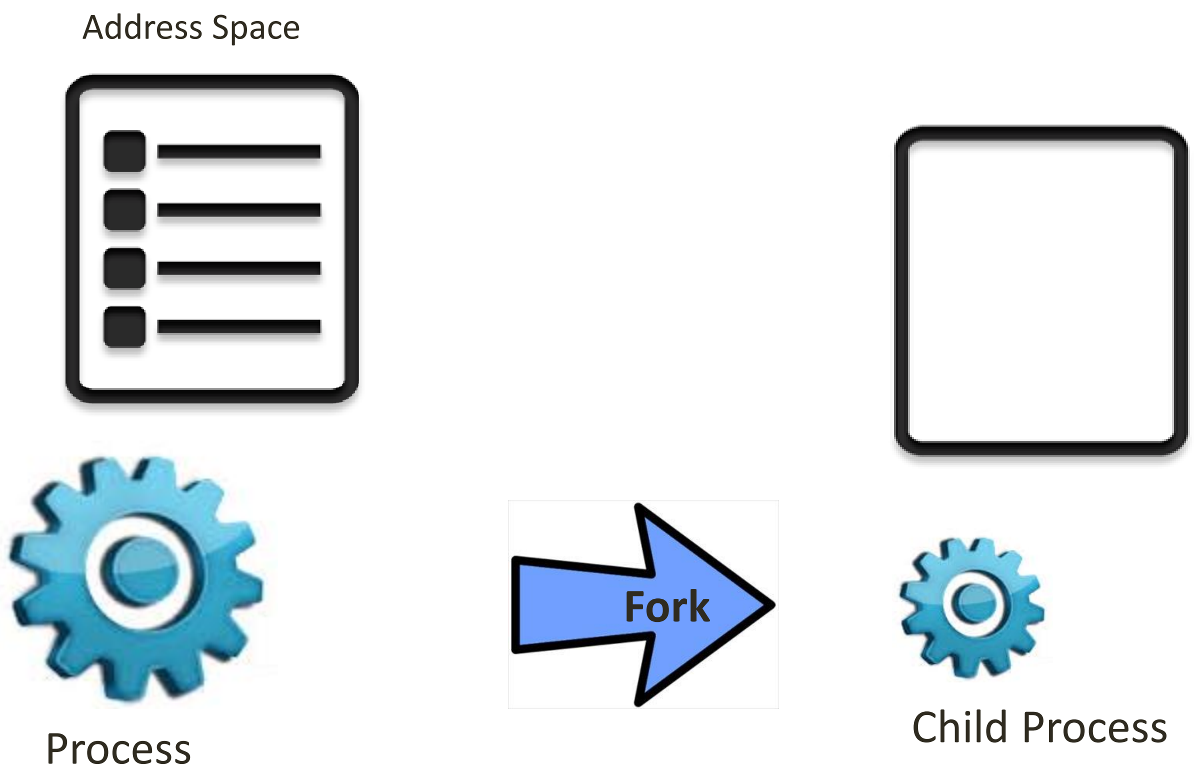
Exec

- The job of “***Exec***” is to send the child process on its own to do its assigned job
- This means the child process will be loaded with,
  - New memory space instead of its parent
  - Will lose access to parent process resources
  - A pointer to the new program to run
- This command separates the child process from its parent, and the child process runs now on its own, using its own memory address space, and running its own executable and creating/reserving its own system resources

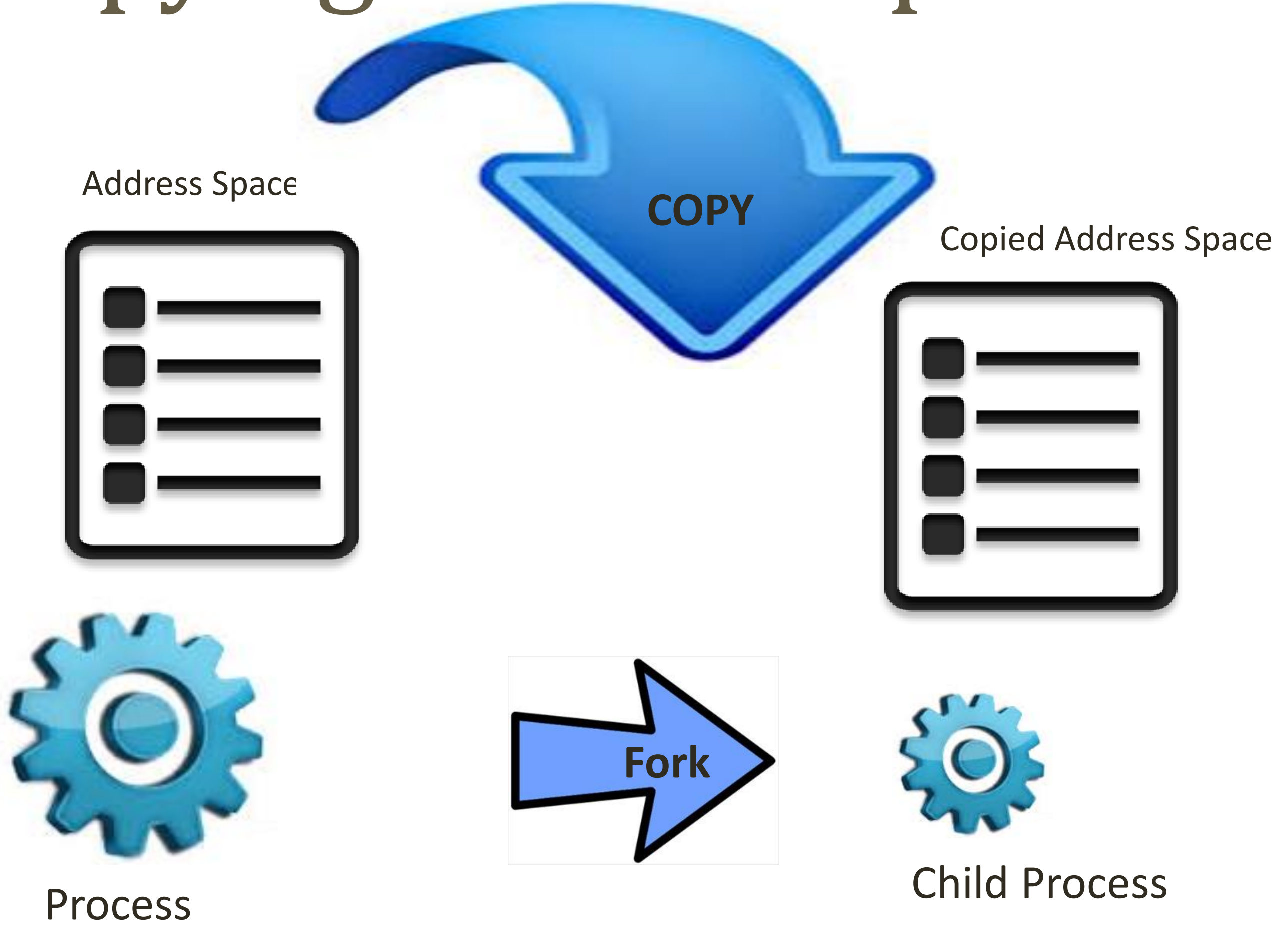
Process Creation: Parent Process

Address Space
Process

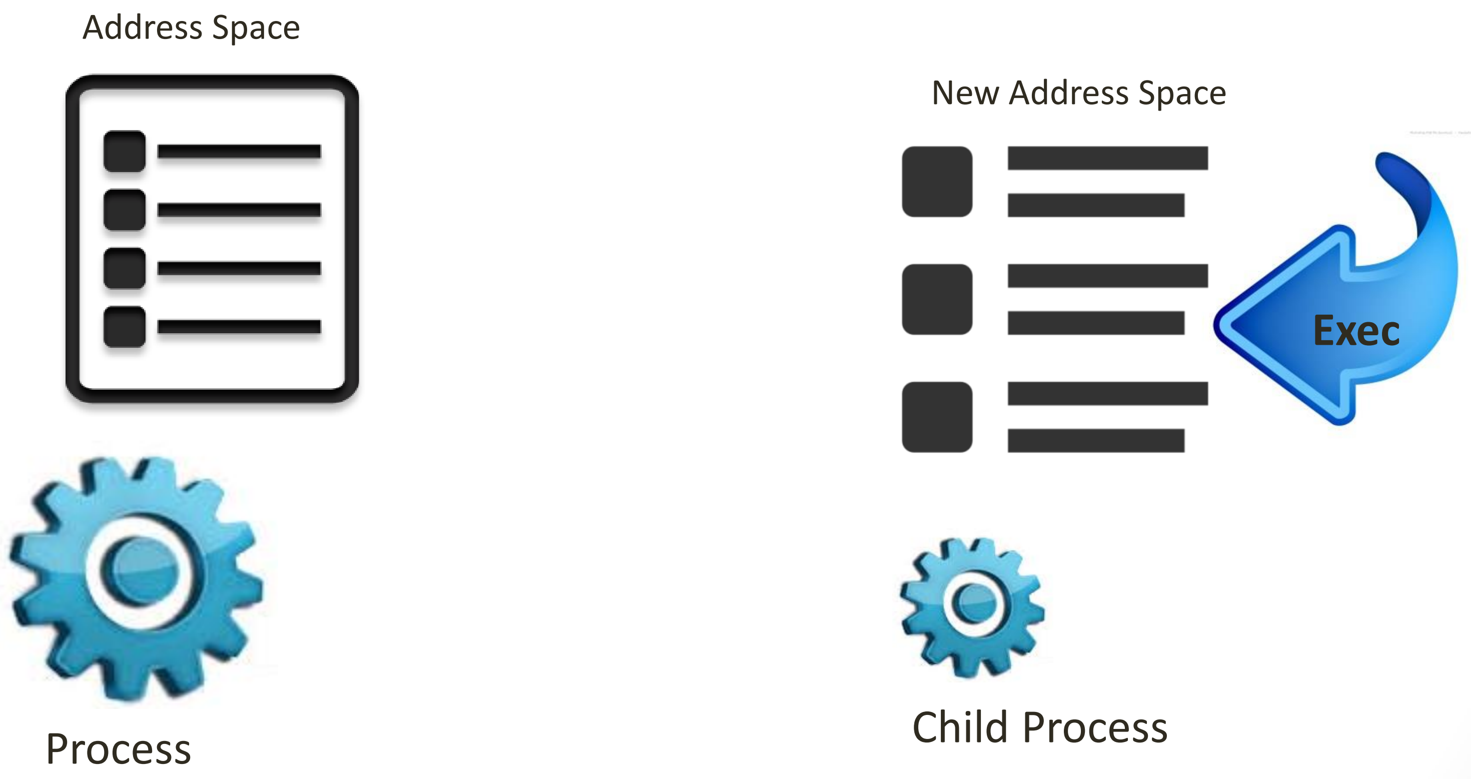
Process Creation: Fork Procedure



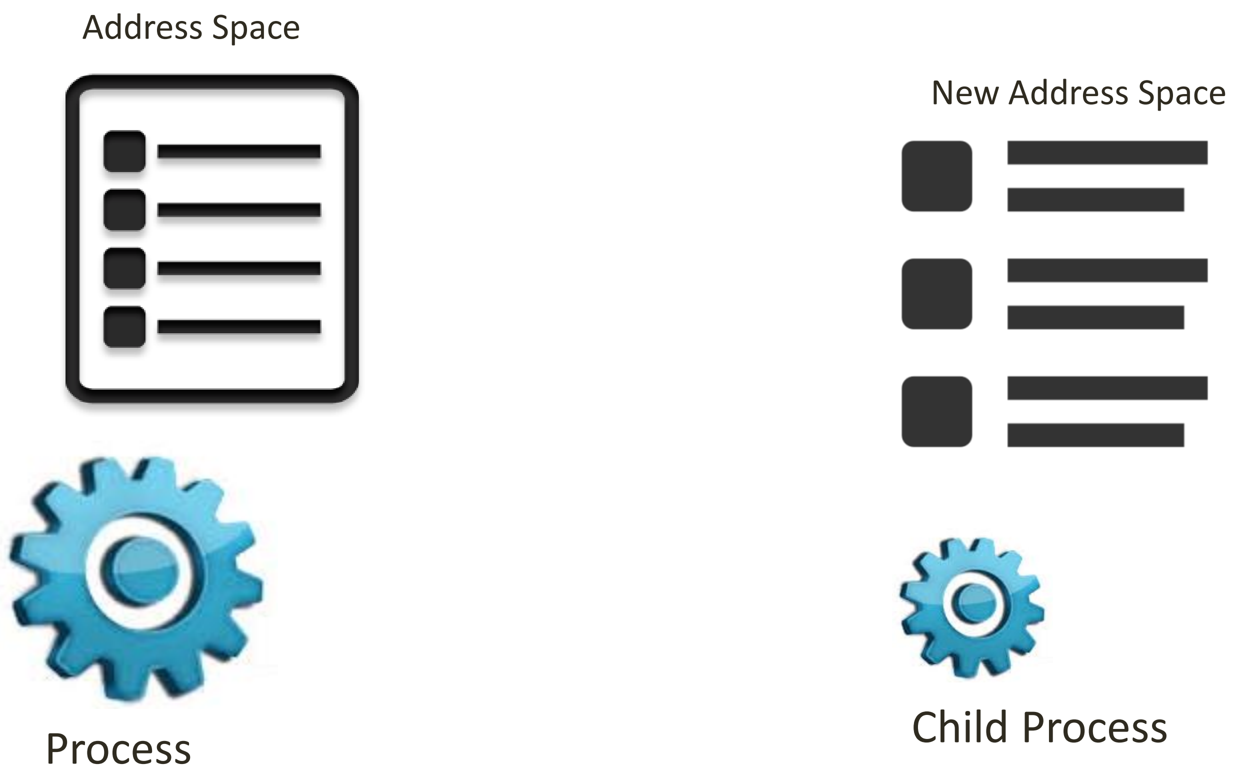
Process Creation: Copying Address Space



Process Creation: Exec Procedure



Process Creation: Parent and Child Processes



Process Vs. Thread

Process Vs. Thread

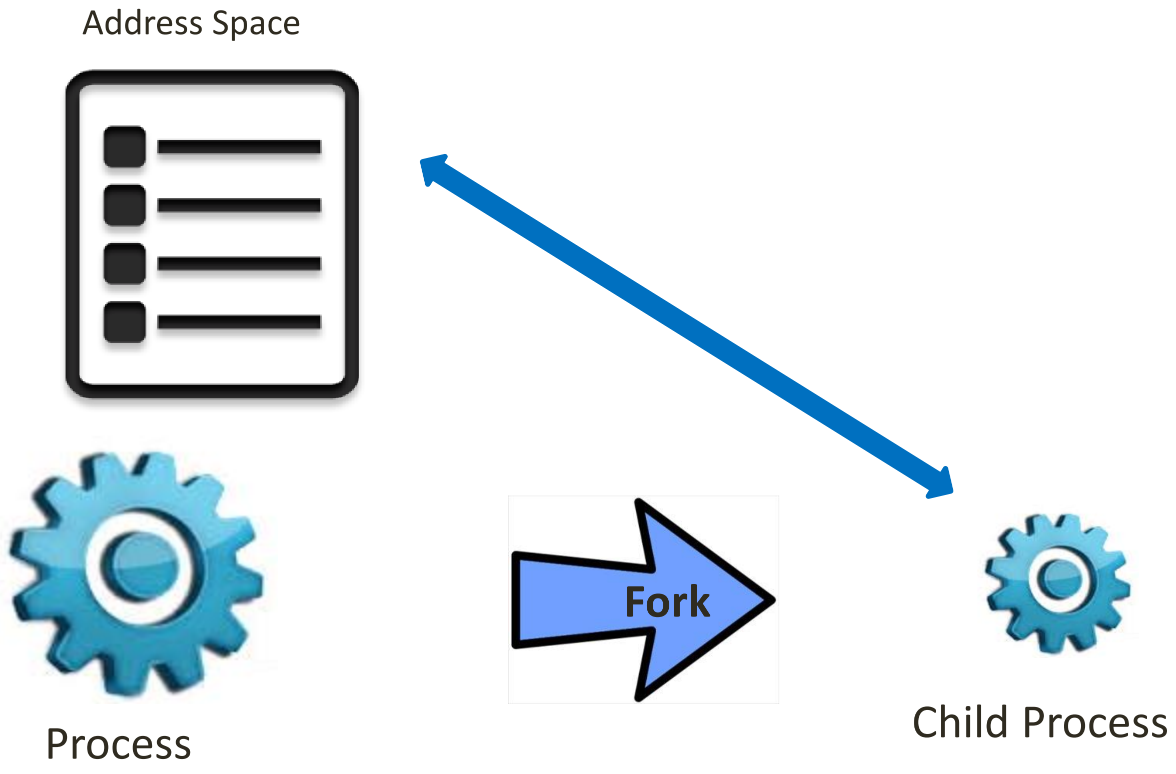
- Both processes and threads are threads of execution that enable parallel operation (multi-tasking) with the help of the scheduler in the kernel
- In both cases, the Linux kernel assigns time slots for the process/thread to execute before another one is assigned

- In Linux a <u>thread</u> is a process with some special differences:
  - When creating a thread, the address space (and some other resources) are not copied from the parent. They are shared with the parent
  - This means threads run within the same address space of its parent and never get a new address space
- Other than that, threads are treated exactly the same as processes
  - For example the kernel scheduler deals with each thread independently
- So, threads created by the same process are multiple processes that share the same address space and resources of their parent process

Creating a Thread: Parent Process

Address Space
Process

Creating a Thread: Fork Procedure



Process Termination

Process Termination

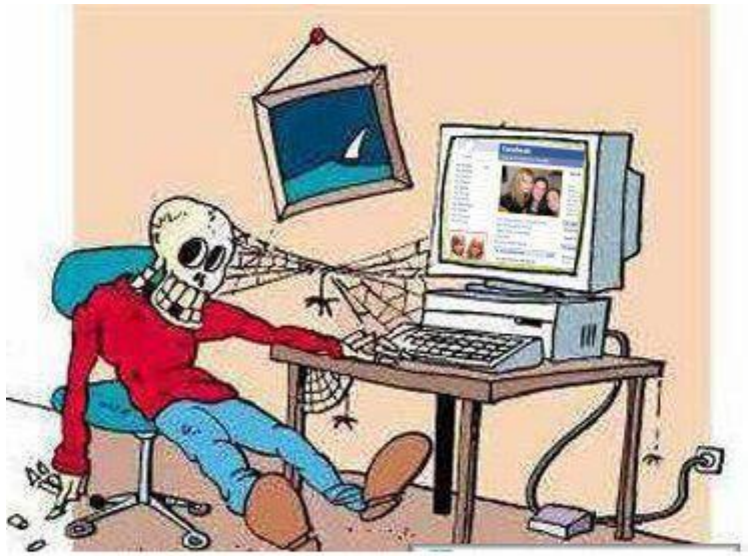
- A process is terminated by
  - Normal termination (reaching the end of its operation)
  - Receiving a Signal that cause it to terminate itself
  - Terminated by the kernel
- When the process is terminated, some of its resources are not yet released, and it changes its state to the ***Zombie*** state.
- The parent process must be waiting for its children processes to perform the required cleanup of resources
- Once the parent process perform the required cleanup, the process state is no longer in ***Zombie***
- This is why every process must have a living parent, and if its parent process terminates, the child processes are re-parented to the ***init*** process

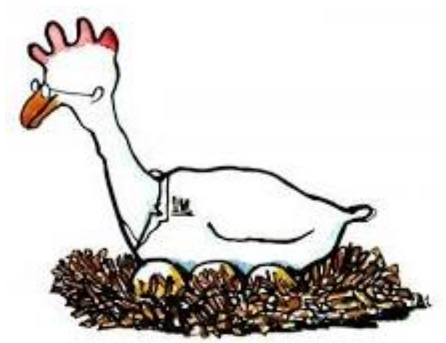
Process States

Running

Stopped

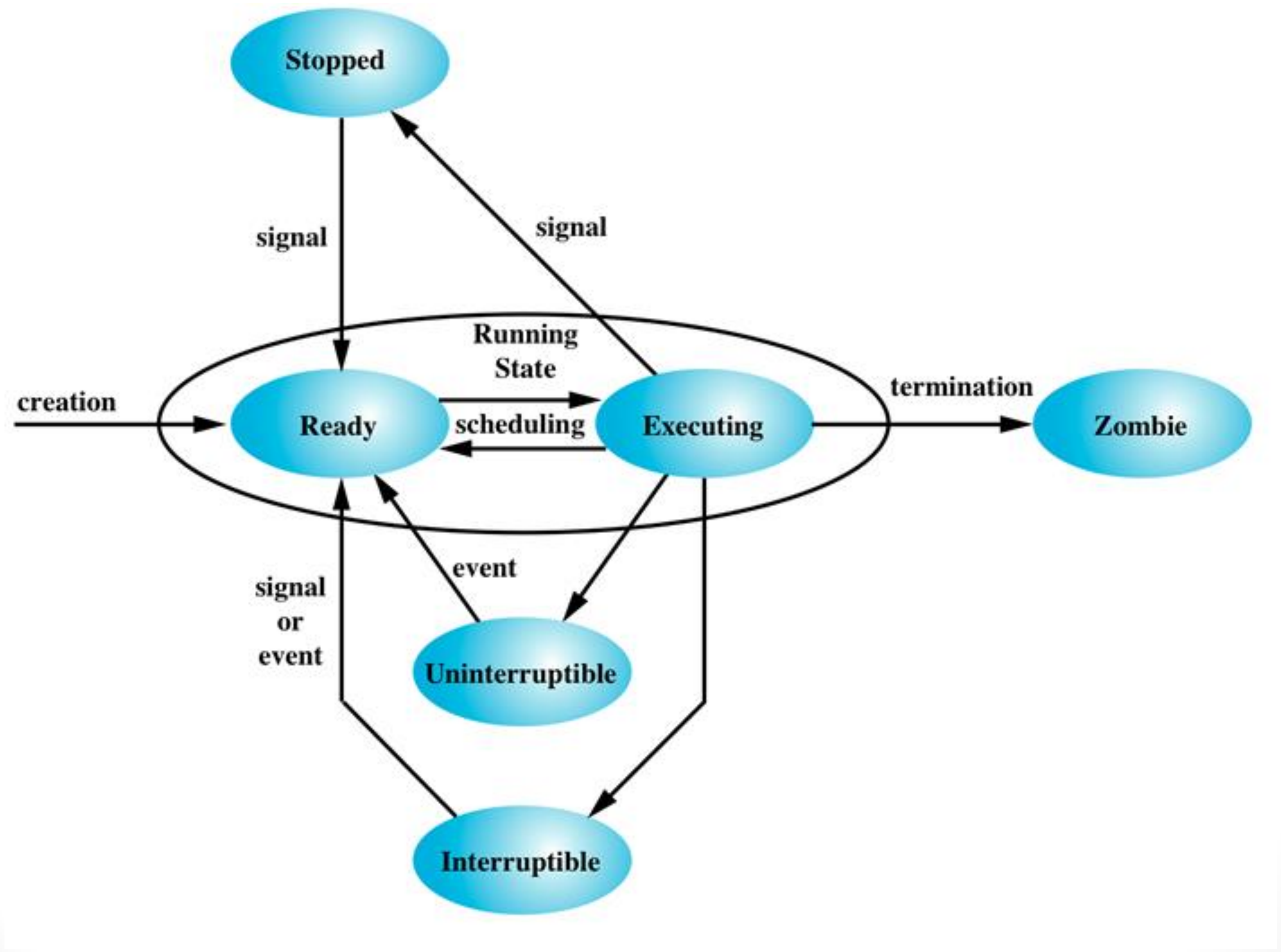
Waiting for some Event





Zombie

Process States



Linux 4

Embedded Systems

http://Linux4EmbeddedSystems.com