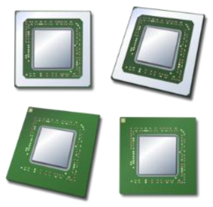
PROCESS CREATION AND CONTROL

Control Primitives: *fork()*, *exec()*, and *wait()*



CONCEPTS

1. A *process* is a ***program in execution***.
2. *Processes* can ***execute concurrently*** in most systems.
3. *Processes* can be ***created*** or ***deleted*** dynamically.
4. *Operating Systems* must provide ***mechanisms*** for ***process creation*** and ***termination***.
5. ***CPU scheduler*** is the first process that is executed ('***sched***' process, PID = 0);
6. '***init***' process (PID = 1) is the first process that is created and serves as the parent root process of all processes in an operating system (UNIX).

CONTROL PRIMITIVES (SYSTEM CALLS)

1. ***fork()*** creates a new process.
   When a process creates a new process, there are two (2) possibilities that exist in terms of execution:
      a. The parent continues to execute concurrently with its children.
      b. The parent waits until some or all of its children have terminated.
   Also, there are two possibilities in terms of the address space of the new process:
      a. The child process is a duplicate of the parent process (same program and data).
      b. The child process has a new program loaded into it.
   Return values:
      a. 0 or -1 (success or failure in creation) is returned to the child process
      b. Nonzero process identifier (PID) of the child process is returned to the parent process.

2. ***exec()*** replaces the child process' memory space with its own binary file and starts its execution

3. ***wait()*** removes a process from the ready queue until termination of its child(ren);

4. ***exit()*** resumes from the call to wait() if there is any, terminates the process, and removes it from the memory.

PROCESS CREATION

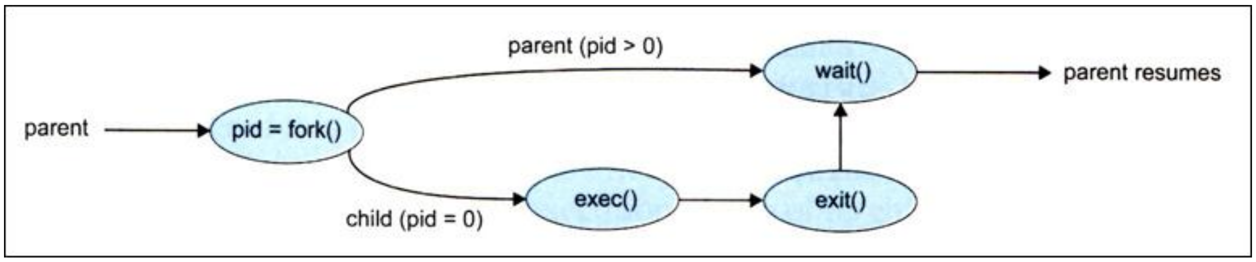


Fig. 1. Process creation using the *fork()* system call