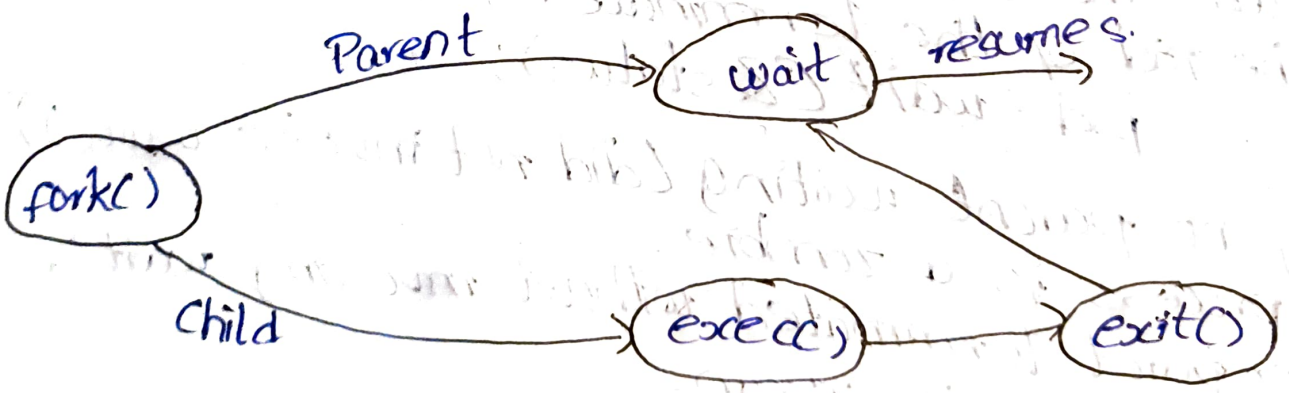
P. Dada Kalander
192424272

Process Creation:-

- Parent process create children processes, which, in turn create other process, forming a tree of processes.
- Generally, process identified and managed via a process identifier.
- Resource sharing options.
  - Parent & children share all resources.
  - Children share subset of parent's resources.
  - Parent waits until children terminate.
- Address space
  - Child duplicate of parent.
  - Child has a program loaded into it.
- UNIX examples.
  - fork() system call creates new process
  - exec() system call used after a fork() to replace the process memory space with a new program.



P.Dada kalander (192,21,24)

Process Termination:

- Paren Process executes last statement and then asks the operating system to delete it using the exit() system call.
    - Returns status data from child to parent
    - Process resources are deallocated by OS.
- Parent may terminate the execution of children processes using the abort() system call.
  Some reasons for doing so:
    - Child has exceeded allocated resources.
    - Task assigned to child is no longer required.
    - The parent is exciting and the OS does not allow a child to continue if its parent terminates.
- Some OS do not allow child to exists if its parent has terminated. If a process terminates, then all its children must also be termined.
    - Cascading termination. All children, grand children, etc. are terminated.
    - The termination is initiated by the OS.
- The parent process may wait for termination of a child process by using the wait() system call. The call returns status information and the pid of the terminated process.

  pid = wait(&status);

- if no parent waiting (did not invoke wait()) process is a zombie.
- if paret terminated without invoking wait, process is an orphan.