Angel Guevara, Jr. 3-05-09

CS340 Assignment 3 – More UNIX Commands

* Commands used in the creation/termination of processes
  + Use **man** to find out about **fork()**, **execve()**, commands in UNIX.
    - **fork**:  
      This command creates a new process, called the child process, while the process calling the fork command is known as the parent process. The child process is either given a new process ID, given a group ID that doesn’t match any other running processes, or the child process is given a different parent ID. However, the child process should be an exact replica of the parent process. After executing the fork command, the parent and child are able to execute independently. The child process may fail if there isn’t enough RAM or storage for the process, and the return value is -1. If everything is successful, the return value is 0 and the child ID is returned to the parent process.
    - **execve**:  
      This command is part of the family functions of exec. This command replaces the current process image with a new one. The new image is executed and will not return any value because the old image is overwritten by the new one. The new process inherits various characteristics from the calling process (similar to the fork command). These characteristics include Process ID, Parent Process ID, Process Group ID, and root directory, amongst others. This command may experience some errors, including E2BIG, which means the number of bytes exceeded the system’s limit, and EINVAL, which means the system doesn’t recognize the format of the file being executed. The errors that may occur will result in a return value of -1.
  + Use **man** to find out about built-in commands: **exec()**, **wait()**, **kill()**.
    - **exec**:  
      This command replaces the current shell process with a specified command. It essentially replaces a shell or becomes a subprocess. Exec overlays the current process with a new one, as opposed to spawning a new process.
    - **wait**:  
      This command will suspend the execution of a calling process until the status of one of its terminated children changes to available, or waits for a specified process to execute and return the status of its termination. This function can take in a parameter n, which is the PID or a job specification. If the parameter is given, then all processes in the pipeline are queued. Otherwise, all of the currently active children are set to wait, and wait() will return 0. If the parameter is incorrect (either an unknown or unfound PID), then the return value is 127.
    - **kill**:  
      This command sends a signal to a process, or a group of processes, which is specified in the arguments (PID, and a SIG (signal) integer), and then returns an integer value. If the PID is greater than 0, the signal will be sent to process IDs that are equal to the PID. If the PID is less than 0, then the signal will be sent to all processes with a group ID equal to the absolute value of PID. If successful, the return value will be 0. Otherwise, the return value is -1 and no signal will be sent.
  + Use Internet sources and give an overview of the command that is used in Windows for creating a process.
  + In a UNIX environment, execute **parent.c**, **child.c**, and **orphan.c** as follows:
    - Child and parent:
      * Compile the child and parent:
        + **gcc parent.c –o parent**
        + **gcc child.c –o child**
      * Run the parent in the current directory (the parent after the fork will call the child). Don’t worry about the warning messages.
        + **./parent**
    - Orphan:
      * Compile and run the orphan:
        + **gcc orphan.c –o orphan**
        + **./orphan**
  + Observe and understand the programs’ output.
    - Extensively comment the output of the programs by relating the theory discussed in class, the meaning of the covered commands and the program listings.  
      *[guan7019@venus ~]$ ./parent  
      Process[17087]: Parent in execution ...  
      Process[17088]: child in execution ...  
      Process[17088}: child terminating ...  
      Process[17087]: Parent detects terminating child   
      Process[17087]: Parent terminating ...  
      [guan7019@venus ~]$ ./orphan  
      I'm the original process with PID 17089 and PPDI 16404.   
      I'm the parent process with PID 17089 and PPID 16404.  
      my child's PID 17090  
      PID 17089 terminates.  
      [guan7019@venus ~]$ I'm the child process with PID 17090 and PPID 1.  
      PID 17090 terminates.*