Week 4 Notes

copy-on-write:

- A memory saving technique for creating "duplicate" resources
- When multiple processes are reading one resource, both process reads the same on copy of it
- When one of the process actually modifies or write on the resources, only then an the resource are copied

Zombie vs Orphaned vs Daemon Process:

- Zombie Process
    - an unintentional orphaned process
    - a process that has ended but still have an entry in the process table
    - usually a parent process would do a clean upon finished child process, but if it doesn't a zombie process is created
- Orphaned Process
    - can be intentional or not
    - a process that its' parent process has ended
    - if a parent process is ended abruptly, usually the child process would be killed to prevent zombie process created
    - if a parent process is ended intentionally, the child process would be "adopted" by the init system, creating what usually is a daemon process
- Daemon Process
    - It's parent is the init process, not always though
    - a process that runs in the background, usually indefinitely
    - It remove its' own file descriptor, to prevent connection with other processes, trully like a "background demon"

exit() vs _exit():

- exit()
    - a library function
    - when it ends its' own process, does a cleanup on buffers and temporary files

- usually not used to end processes, due to potentially cleaning up files and environment variables of its parent process

- usually use to end main functions

- _exit()

- a system call

- only ends its own process, no clean up

- the usuall best-practice to end a process

Dameon creation with double fork():

- a process call fork() twice, creating a child and a grand-child process

- the child process is immediately killed

- since the parent of grand-child, the child, is killed, it would be adopted by the init process

- the grand-child and parent process are decoupled

- next step would be to remove grand-child file descriptors

pipe() with fork():

- a pipe() would create a file that bridges between profiles

- the first process would set its stdout to the pipe file, and the second process would set its' stdin to the pipe file.

- this could also be use with fork(), to create a bridge between parent and child processes.

PS options:

- UNIX options

- arguments always starts with a dash (-)

- usually to monitor current user session and terminal

- BSD options

- arguments DOESN'T starts with a dash (-)

- usually to monitor multiple terminal at once

- GNU long options

- arguments always starts with a double dash (--)

- usually mixed with the other 2 options to create better output