Signals:-

ctrl+c ==> SIGINT ==>terminate
ctrl+\ ==> SIGQUIT ==> terminate
ctrl+z ==> SIGTSTP ==> suspend
div by zero ==> SIGFPE ==> exception
seg fault ==> SIGSEGV
child exit
unblocks parent ==> SIGCHLD
SIGTERM ==> terminate

kill -l ==> list out various signals

send signals:-

kill -2 <pid>
eg:- kill -2 4481

kill -15 <pid>
kill -SIGTERM <pid>
kill <pid> #SIGTERM by default

When a signal is targetted to a process, signal handlers
execute by the processs
Most of signal handlers causes abnormal termination of process

Objective:-
    to ignore signals (or)
    execute custom handler instead of default handler...
        custom handlers can prevent from termination
        or convert to normal termination
Most of the signals can be ignored(masked) or custom handled
but two signals can't be masked or no custom handling
    SIGKILL   ==> sure kill
    SIGSTOP   ==> sure suspend

background vs foreground process
suspend and resume process

command &   ==> run in background
                ==> can't take input from user(stdin)
                ==> typically stores o/p in a log file

fg ==> bring b/g process to f/g
==> resume suspended process in f/g

bg ==> resume suspended process in b/g

ctrl+z ==> suspended active f/g process using SIGTSTP

fg,bg acts on recent process

jobs ==> list out suspended and background processes of current terminal

user can send signals(kill) of owned processes only

pkill ==> send signal by process name
eg:- pkill a.out

killall ===> send signal to all processes with particular name

commands:-
jobs, bg, fg, kill, pkill, killall
command&, ctrl+z

signals:-

SIGINT, SIGQUIT,SIGHUP

SIGTERM, SIGKILL

SIGTSTP,SIGSTOP, SIGCONT

SIGCHLD

SIGFPE, SIGSEGV