Forking

process01.c

Basic example of forking, a child process gets created with the same code, data, stack, user data, the child gets terminated when the parent gets terminated

process02.c

orphan child: why? the parent finishes before the child thus the child becomes an orphan and the os take custody of the child thus when the child print getppid() it prints 1 because that is the process id of the systemd

process03.c

var1 = wait(&var2)
var1 → the id of the terminated child
var2 → 32 bit number with the first three byte = exit code , last byte = 0 if exited normally and 1 otherwise
var2 → stat_loc
wait basically makes the parent wait for the child to terminate

WIFEXITED(stat_loc) → true if the child terminated correctly false if otherwise
WEXITSTATUS(stat_loc) → return the exit code of the child
A Child doesn't get terminated from the process table expect for two case
→ Parent is terminated
→ Parent call wait()
Note: if no exit code is provided the process exits the exit code 0

process04.c

same as process03.c but the only difference is we exited the child with a process code (42)

process05.c

same a process03.c but there is a sleep function excuted by the parent, to let us know that the child is zombie process I guess

process06.c

shows execl which is a function which replace the current process with another process takes a arr of strings first being the path to the program desired the next arguments being the the options the last argument must be NULL,
Note: the parent remains the same and the child doesn’t get a new parent

process07.c

shows how to use signals, use kill -9 pid to terminate the process or ctrl + c
if you terminate the child it become a zombie and it doesn’t go away until the parent get terminated
if you terminate the parent the child gets inherit by systemd and keeps running

process08.c

shows case the priority and niceness, niceness is added to priority, it’s chnaged with the nice function, and takes the value between -20,20.
priority is accessed using getpriority(PRIO_PROCESS, 0)
the less priority the process the more priority the process has (I am sorry)
there might be context switching between child and parent process
system(”ps -l”); → excute a command but doesn’t replace the process unlike the execl.