Monday, October 25, 2021 11:09 AM

What happens to the address space after the fork system call is made ?
----- a copy of the address space is created ---- the child's copy

EXEC system call ---- Exec takes command over existing process

Did you start a C program from another C program ?
TO run a linux command from a C program ?

Exec Family of Functions / System calls

execl ,execlp , execv, execvp

Every Linux command is a process that runs !
The C program we run is also a process

Exec will RUN the ls process within the C process
The Exec process HiJACKS the Current C process
The Exec process takes control of the address space of the C program

As a thumb rule --- The C program calling exec must not write its own code !!!

Usually exec is used along with fork -------

---

execl = give the location of the command as first argument and other command options as comma separated argument list

```
execl( "/bin/ls"," -l", NULL );
```

The exclp , execvp tries to find the ls command in the PATH environment variable

```
execlp ( "ls" , " -l" , NULL );
execvp( "ls" , arr )
```

```
char * arr[] ={ "ls " ,"-l",NULL } ;

execv ("/bin/ls" , arr )
```

---

execl	First arg = name of command with location	Comma seperated options
execv	Same with location	Array of options
execlp	Name of command , location taken from PATH	Comma separated options

execvp	Same	Array of options

____

HW ---

Write a c program that forks and creates 2 child processes

The first process runs the ls - l command } execl

The second child process runs the ps -ef command } execv

The parent should wait (WAIT) for a signal from both the children that they are done.
After that the parent should print GOOD BYE and end

____

Wait = is a way to make the parent wait for child to finish.

This is a good programming practice where programs are using fork !!!
If parents waits till the child finishes then parent cleans up the process tables for terminated children quickly , NO orphans , no zombies !!!!

HW --- read the Linux MAN PAGE for wait()

____

Fork() , getpid, getppid, wait , exec family !!!!

____
__

Signals in linux !!!

Software interrupts / exceptions

Can be sent by Kernel process to user process

By user process to process

Signals are a way of communicating BETWEEN processes ---- Inter process communication

Signals have Signal Handlers !!!

Signal handlers are **functions** that will do something when signal occurs !!!
Default signal handlers for all Signals !!!!!

If we want to change the default signal handler , we can give our own signal handler!!!!

We must register OUR signal handler with the kernel ------ system call **signal ()**

signal(which signal , which handler) system call

____

Kill command can send a signal to the process

Kill -signalnumber pid

Kill pid }} the default signal is 15 SIGTERM Maskable

Kill -9 pid }} SIGKILL - 9 non maskable

Ctrl C =   SIGINT   2   Maskable
Kill   -2 pid

____

Kill   -l   }}   find the list of signals and write handlers for whichever you like !!!
____

We executed the kill command from a C program   !!!!

____