**Fork**

* Processes are created using the fork command
* Basically, the fork duplicates the process

-Creates child process in different address space (duplicate of the parent)

-Parent process is the main process

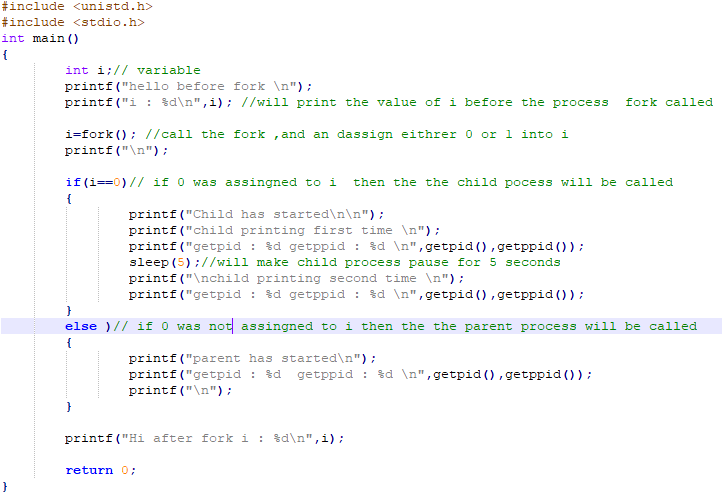
* The fork command will return 1 of 3 options

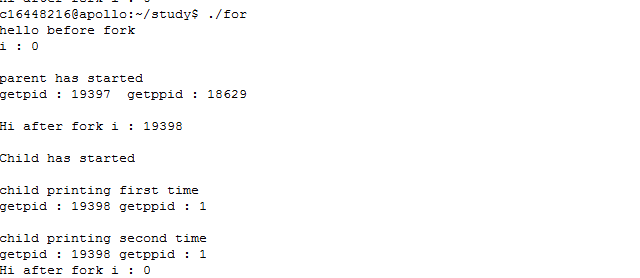
-(0 returned) means the child process

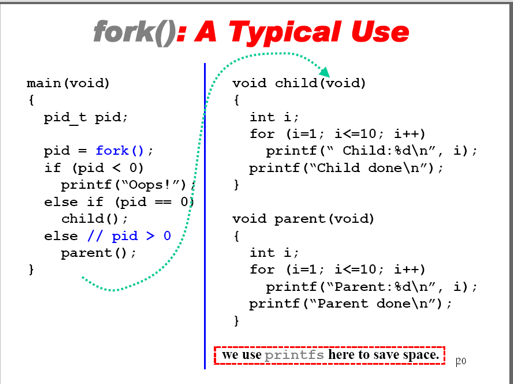
-(1 returned) means the parent process

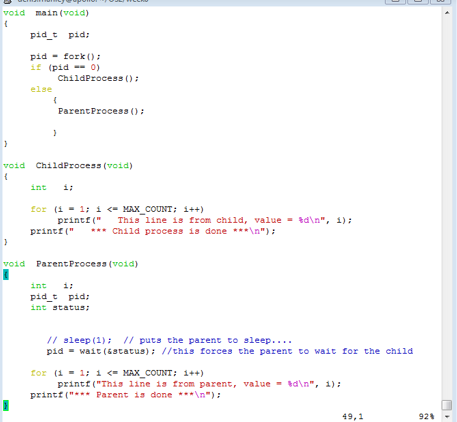
-(-x returned) means the child wasn’t created

* **Getpid()** which return the ID of the calling process(child process)
* **GetPPID()** which returns the ID of the parent of the calling process.
* To create 2 children you can put a for inside a fork as seen in my code (create2children.c)
* Below is my code from fork.c









**Wait**

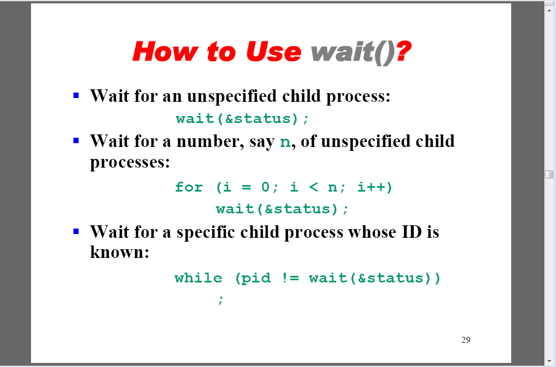
* Wait command is all about the idea of synchronization
* Its used with the fork command
* The fork creates two copies and which ever process has the wait function ,will wait for the other program to complete,and then itself finishes.
* Its a way to synchronise the child and parent.
* Its done with the following line of code

-SLEEP()

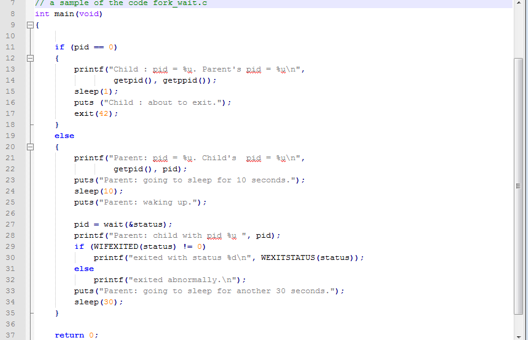
This line of code

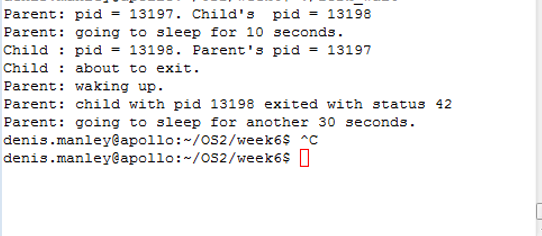
*Int status ;*

*Wait (&status) ! = pid ;*

Will wait for the child process to complete

Fork\_wait.c

**

**

**Exec**

* Basically it’s a way to start the execution of one program from inside another program
* Uses fork and wait commands
* An artificial shell is created and the 2nd program runs inside this shell (similar to the bash window).
* Argv is an array of strings corresponing to the command line arguments
* FOR EXAMPLE: int execvp (char \* prog \*char argv[])

-1.stops current program

-2.loads the program “prog” into address space

-3.Intializes hardware context arguments (args) for the new program

-4.Puts PCB into ready queue

* + Exec is called from the child process, so the os replaces the child process’ memory space with a new program
  + Create parent-child relationship, Call fork, wait, and exec in that order
  + 