Roll No:
17 TEA

10th Assignment

Nikita D Bhosale

GURUKUL Page No
Date

Aim: Implement UNIX system call like fan process management

Problem statement: WAP to implement UNIX sys call like for process management.

Theory :

① CALL :
- When a Prog in user mode requires acess to RAM or i/p/o resource it must ask the kernel to provide acess to that resource. This is done via something called a system call
- when a Prog makes a system call the mode is switched from user mode to kernel mode. This is called a context switch

- In following situation system call are generated.
① ~~Gen~~ Creating, opening, closing, deleting files
② Creating & managing process
③ Creating a connection in n/w, sending & receiving packets.
④ Request access to h/w device like mouse & primeter.

② KERNEL Mode :
- When CPU is in kernel mode the code being executed can access any memory address and any h/w resource
- In user mode if any Prog crashes only that particular Prog hosted.
- That is system will be in safe state even if a Prog in user mode crashes.
- Hence most Prog in an os run in user mode.

③ Basics of system call :-

- since system call are functions we need to includ proper header files e.g for getpid () we need
  # include <sys / type.h>
  # include < unistd.h >
- most system call have a meaningful return value
  ① usually -1 or a negative value indicate an error
  ② A specific error code is Place in a global var called : echo
  ③ To access echo you must declare it extern int err

④ sys calls for processes :-

i) pid-t fork (void)
- create a new child process, which is a copy of current process
- parent return value is the PID of child process
- child return value is 0

ii) int execl (char * name, char * arg () . - - (char *) 0)
- change prog image of current process
- Reset stack and free memory
- start at main()
- Also see other version : exclp (), execv (), etc

iii) pid-ti wait (int * status)
- wait for a child process to complete
- Also see waitpid () to wait for a specific proces

iv) void exit (int status)
- terminate the calling process
- send SIGkill to force termination

⑤ Unix system call :-

i) PS command :-
- used to provide info about currently running process including their process identification
- Every Process assigned a unique PID by the syntax
  syntax : Ps[options]

ii) fork command
- The fork() sys call is used to create Process
- when a Process makes a fork() call an exact copy of Process is created
- Now these are two Process one being the parent Process and the other being the child Process

iii) Join command :-
- The join command in UNIX is a command utility for joining lines of two files a common field
  syntax : join[option] ---- file1 file2

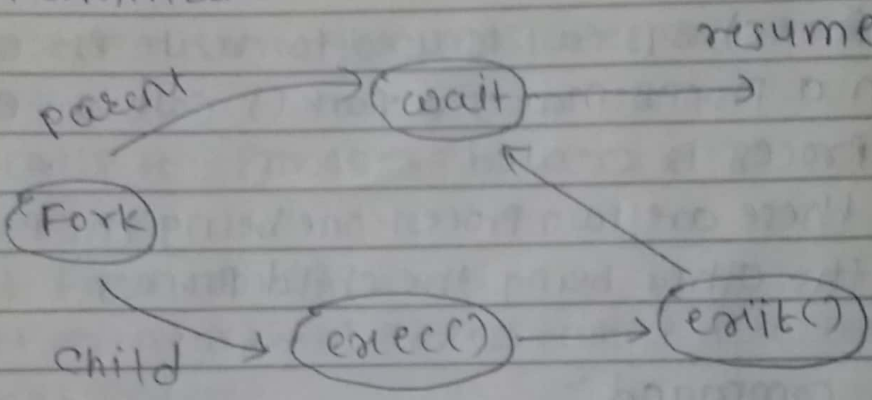
~~iv)~~
iv) exec command :-
- used to create processes
- But there is one big difference betn fork() & exec()
- fork() call creates a new process while preserving the parent process
- exec() replace the address space text reg, Data seg etc of current Process with process

v) wait command ()
- Block the calling Process until one of its child Process exits or a signal is received

GURUKUL Page No.
Date / /

- After child process terminate parent contain continues its execution after wait sys call

child process may terminate due to any 1 of the following reason
- It calls exit()
- It returns (an int) from main
- It receives a signal whose default action is to terminate



Conclusion:
Thus the process sys call prog is implemented and studied various system calls