DPU
Dr. D. Y. Patil Institute of Technology

ASSIGNMENT NO. 3

(*) AIM: Implement UNIX system calls like PS, fork, join, exec family & wait for process management (use shell script/ Java / c prog).

(*) THEORY:

What is UNIX:

The UNIX o.s is a set of program that act as a link b/w computer and user.

The computer program that allocate the system resources & co-ordinate all the detail of computer internals is called OS or kernel.

(*) UNIX system calls:

1) PS - Report a snapshot of the current processes.

Syntax - PS [option]

PS displays information about selection of active processes if you want a repetition update of selection & displayed information use instead.

By default PS seleds all processes with same effective ID as current user & assessed with same terminal as invoke - if display show command args instead of name

EX:

To use every process on the system std syntax.

ps -e
ps -ef
ps -ely.

2) fork() - create child process.

synopsis.
#include <sys/types.h>
#include <unistd.h>.

pid_t fork (void);

Description: fork() creates a child process that differs from the parent process only in its PID & PPID & infact that resources utilization are set to 0.

Return value: The PID the child process is returned in parents thread of execution & 0 is returned in child thread of execution.

ERRORS: fork() can't allocate a suffecient memory to copy parents page tables & allocate a task structure of child.

3) join - Join lines of two files on common field.

syntax Join [option] ------ file 1, file 2.

Description: For each pair of i/p lines with identical.

field write a line to std o/p. The default join field is the first delimited by white space when file 1 or file 2 is read std i/p.

Tag:
-e EMPTY - replace missing i/p field with empty.
-1 FIELD - join on the field of file 1.
-2 FIELD - join on the field of file 2.

4) Exec family - The exec family of function replace the current running process with new process. It comes under header file unistd.h.

i) Execvp - Using this command, the created child program as the parent process doesn't have to run the same program as parent process does.

Syntax:
int execvp (const char * file, char * const argv[]).

ii) execv - This is very similar to execvp() in term of syntax as well
int execv (const char * path, char * const argv[])

iii) execlp & execl - These two also serve the some purpose but syntax of them are different.

Syntax:-
int execlp (const char * file, const char * args, ... / * (char *) NULL * L);
int (const char * path, const char * arg * (char *) NULL * L);

5) wait() - wait for process to change state.

Synopsis -
#include <sys/types.h>
#include <sys/wait.h>
pid_t wait (int * status);
pid_t wait (pid_t pid, int * status, int options);

Description -
Waiting system call suspends execution of current process until one of its children terminate. wait pid (-1, &status, 0);

The waitid() system call provide more precise control over which child state changes to wait state.

(*) CONCLUSION :
Thus, I have studied about UNIX system calls. PS, fork, wait, exec, join. with its syntax.

PWS