Assignment C3

Title: UNIX System Calls

Problem Statement:

Implement Unix System calls like ps, fork, join, exec, family and wait for process management (Use shell script /java/ C)

Objectives:

1) To understand the significance of unix calls
2) To learn about the implementation of system calls

Outcome: I will be able to
1) Implement system calls and their execution
2) Understand the concept behind various system calls

Software and Hardware requirements:
1) 64 bit Fedora 20
2) i5 Core processor
3) memory 4GB
4) Text editor

Theory:

A) Shell Scripting:

A shell provides you with an interface the Unix system. It is an environment in which we can run our commands, program Shell Script. There are two major types of shells
1) Bourne Shell
2) C Shell

Shell script is a list of commands which are listed in order of execution. It is interpreted and not compiled. A good shell Script has commands preceded by # describing the steps:
example – test.sh

```
# Accept User name
echo “What is your name ?”
read Person
echo “Hello, $Person.”
```

Unix System Calls:

A system call is a request for the OS to do something on the behalf of the users program. The system calls are function in the kernel it cell since the system calls execution course in the kernel coma method of process from user mode to kernel mode.

Unix system calls are used to manage the file system, control processes and provide interprocess communication.

Given System Calls:

A] exec() system calls

It transforms and executable binary file into a process.

Prototype of exec() family:

```
int execl(filename, arg0, [arg1,........,argn],NULL)
char *filename, arg0, arg1,..........., argn


int execv(filename,argv)
char *filename, *agrv[]

int execle(filename, arg0,[arg1, arg2,........, argn],NULL,exvp[])
char *filename, arg0, arg1,......, argn, exvp[];

int execvp(filename,argv)
char *filename, *argv[]
```

2] fork() system call

To create a new child process, fork() is executed.

Int fork() is the prototype

It makes UNIX create a new “child process” with a new ID. The content are identical to present process.

3] wait() system call

You can control the execution of a child process by calling wait() in the parent process.

Wait() forces the parent to suspend execution until the child has finished.

It returned PID of the child process that is finished.

```
int wait(status);
int *status;
```

4] <u>ps() system call</u>

It is used to provide information about the currently running processes, including their processes identification numbers.

Syntax:- ps[option]

5] <u>join() system call</u>

The join command is a Unix command line utility for joining lines of two files on the common field. It can be used to join two files by selecting fields within the line and joining the files on them. The result is written to the standard output.

Test Cases :

Description	Input	Expected O/P	Actual O/P	Result
fork	fork()	New process created	New process created	Success
join	Join first.txt second.txt	Text files with matching keys are joined	Text files with matching keys joined	Success
ps	ps	Process id of various process is printed on terminal	Process id of various processes is printed on terminal	Success

Conclusion:

Hence we have completed implemented Unix system calls on a system tested how they can be used to control processes.