Page :
Date : / /

Assignment C3

Title : UNIX System Calls

Problem Statement :
Implement UNIX System Calls like ps, fork, join, exec family & wait for process management ( Use shell script / Java / C )

Date of Completion :

Objectives
To understand the significance of UNIX calls.
To learn about the implementation of system calls

Outcomes : I will be able to :
Implement System Calls & their execution
Understand the concept behind various system calls

Software & Hardware Requirement :
64 bit Fedora 20 OS
i5 core processor
Memory 2 GB
Text Editor - Gedit.

Theory :

A. Shell Scripting

A shell provides you with an interface to the UNIX system. It is an environment in which we can run our commands, program shell scripts.

There are 2 major types of shells:
1) Bourne Shell
2) C Shell

Shell Scripts is a list of commands which are listed in order of execution. It is interpreted & not compiled. A good shell script has comments preceeded 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 behalf of the user's program. The system calls are function in the kernel itself since the system call executes code in the kernel, there must be a mechanism to change mode of a process from user mode to kernel mode.
UNIX System calls are used to manage the file system, control processes & provide interprocess communication.

Given System Calls :-

exec() system calls

It transforms an executable binary file into a process.
Prototypes of the exec family

- int execl (filename, arg0, [arg1, ... argn,] NULL)
  char *filename, *arg0, *arg1, ... *argn.
- int execv (filename, argv)
  char *filename, *argv[]

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

· 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

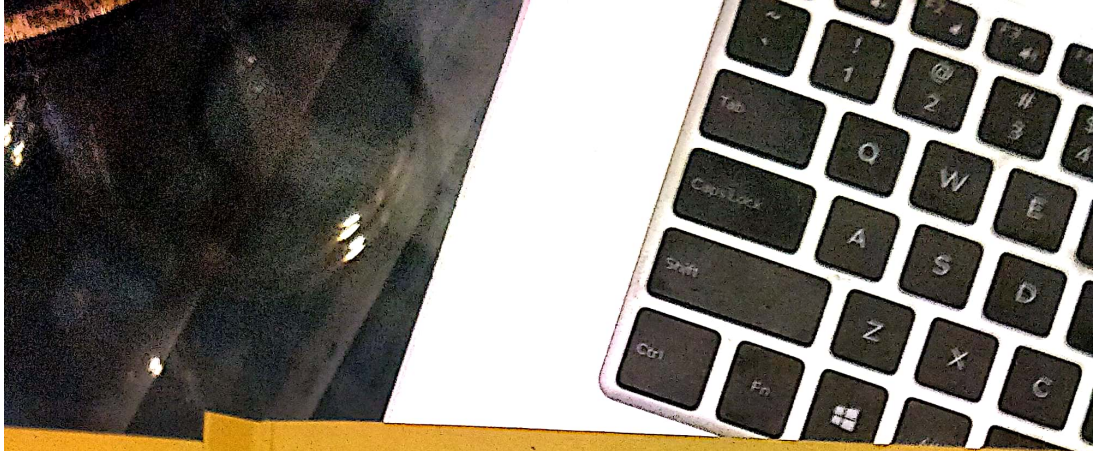
You can control the execution of a child process by calling wait() in the parent process.
wait() forces the parent to suspend execution until the child has finished.
It returns PID of the child process that is finished.
· int wait (status);
int *status;

The
It is
the
their
Syntax

The

The
line
files
If
by
&
The
outp

Conc
Here
Syste
how



Page :
Date :

The ps system call

It is used to provide information about the currently running processes, including their process identification numbers.
Syntax - ps [options]

The join () system call

The join command is a UNIX command line utility for joining lines of 2 files on a common field.
It can be used to join two files by selecting fields within the line & joining the files on them
The result is written to a standard output.

Conclusion :
Hence we have implemented UNIX System calls on our system & studied how they can be used to control processes.

14 Implement algorithmic techniques by virtue of advance progra... Open Source Software (FOSS) tools. ... and Free ...

Page:
Date:

## Test Cases

| | Description | Input | Expected o/p | Actual o/p | Result |
|---|---|---|---|---|---|
| 1) | fork | fork() | New process created | New process created | Succ |
| 2) | join | join first.txt second.txt | Text files with matching keys joined | Text files with matching keys joined | Succ |
| 3) | ps | ps | Process id of various process is printed on terminal | Process id of various process is printed on terminal | Succ |

## Conclusion

Hence we have implemented UNIX system calls on our system & studied how they can be used to control processes.