OS\_LAB\_04\_Assignment

CE\_054

Aim : Study and implementation of ps command’s basic functionality.

Theory :

TASK 1:

**PS (Process Status) :-**

* The ps command, short for Process Status, is a command line utility that is used to display or view information related to the processes running in a Linux system. As we all know, Linux is a multitasking and multiprocessing system. Therefore, multiple processes can run concurrently without affecting each other.
* Linux provides us a utility called **ps** for viewing information related with the processes on a system which stands as abbreviation for **“Process Status”.** ps command is used to list the currently running processes and their PIDs along with some other information depends on different options. It reads the process information from the virtual files in **/proc** file-system. /proc contains virtual files, this is the reason it’s referred as a virtual file system.
* ps provides numerous options for manipulating the output according to our need.

**Some examples of ps command** :

1. **Simple Process selection** :-

Syntax

$ ps

Result contains four columns of information.  
Where,  
**PID –** the unique process ID  
**TTY –** terminal type that the user is logged into  
**TIME –** amount of CPU in minutes and seconds that the process has been running  
**CMD –** name of the command that launched the process.

1. **View Processes not associated with a terminal :-** View all processes

except both session leaders and processes not associated with a terminal.

Syntax :-

$ ps -a

1. **View Processes :**View all the running processes use either of the following

option with

Syntax :-

$ ps -A

$ ps -e

1. **View all processes except those that fulfill the specified conditions**

**(negates the selection) :-** If you want to see only session leader and processes not associated with a terminal. Then, run

**Syntax :-**

**$ ps -a -N**

**$ ps -a -deselect**

1. **View all processes owned by you :**Processes i.e same EUID as ps which

means runner of the ps command, root in this case –

Syntax :-

$ ps -x

1. **View all processes associated with this terminal :-**

**Syntax :-**

**$ ps -T**

1. **$ ps aux :-**

**a**:- This option prints the running processes from all users.

**u**:- This option shows user or owner column in output.

**x**:- This option prints the processes those have not been executed from the terminal.

* + Collectively the options "aux" print all the running process in system regardless from where they have been executed.

1. **$ ps -ejH :-**

**It will gives us PID, PGID, SID, TTY, TIME, CMD.**

1. **$ ps -eLf :-**

**It will gives us UID, PID, PPID, LWP, C, NLWP, STIME, TTY, TIME, CMD.**

**TASK 2:**

**Proc file System :-**

proc is very special in that it is also a virtual filesystem. It's sometimes referred to as a

process information pseudo-file system. It doesn't contain 'real' files but runtime system

information, For this reason it can be regarded as a control and information centre for

the kernel. In fact quite a lot of system utilities are simply calls to files in this directory.

Inside the proc file there is a stat file inside this stat file there are all indormation about

running processes of system.

**Wakekill, Waiting and parked Status :**

1. **Wakekill** :-

A process will be waken up when any fatal or deadly signal is received.

1. **Waiting :-**

To handle concurrent Wake-ups and release the rq-lock we put the process in

this state, its guarantees that nobody will actually run it, and other external

signal can not wake it up.

1. **Parked :-**

The process stays in the parked queue until the read request completes, and sometimes it stays there until the hardware or software error that prevented the request completion is repaired.

PROGRAMS :-

**Write a program to print process id and process name of all current processes in the system.**

**CODE:-**

**/\*Author : Dhruv B Kakadiya**

**\* Implementation of Basic ps command using c language**

**\*/**

**#include<unistd.h>**

**#include<dirent.h>**

**#include<stdio.h>**

**#include<sys/types.h>**

**#include<string.h>**

**#include<stdlib.h>**

**#include<sys/stat.h>**

**#include<ctype.h>**

**#define FORMAT "Process ID : %5d\t Process Name : %30s\t Status : %c\n"**

**void execution\_of\_ps(char \*main\_dir, DIR \*dir, struct dirent \*dirp);**

**// function for ps command**

**void execution\_of\_ps(char \*main\_dir, DIR \*dir, struct dirent \*dirp)**

**{**

**char main\_path[100000];**

**char process\_name[100000];**

**int pid;**

**char status;**

**if((dir = opendir(main\_dir)) == NULL)**

**{**

**fprintf(stderr, "cannot open directory : %s\n", main\_dir);**

**return;**

**}**

**while(dirp = readdir(dir))**

**{**

**if(strcmp(dirp->d\_name, ".") != 0 && strcmp(dirp->d\_name, "..") != 0)**

**{**

**if(dirp->d\_type == DT\_DIR && atoi(dirp->d\_name) > 0)**

**{**

**strcpy(main\_path, main\_dir);**

**strcat(strcat(main\_path, dirp->d\_name), "/stat");**

**FILE \*fp;**

**if (fp = fopen(main\_path, "r"))**

**{**

**fscanf(fp, "%d %s %c", &pid, process\_name, &status);**

**printf(FORMAT, pid, process\_name, status);**

**}**

**else**

**continue;**

**}**

**}**

**}**

**closedir(dir);**

**return;**

**}**

**int main()**

**{**

**DIR \*dir;**

**struct dirent \*dirp;**

**execution\_of\_ps("/proc/", dir, dirp);**

**printf("\n\n");**

**exit(0);**

**}**

**Output :-**



