**LAB 5 ASSIGNMENT**

**Name:** Birva Babaria

**Roll no.:** CE010

**ID:** 19CEUON064

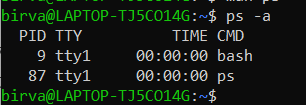
**Aim:** Study and implementation of ps command’s basic functionality.

**Description:**

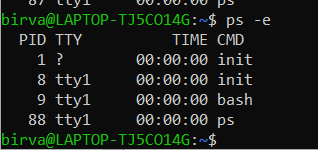
**1). ‘ps’**

ps report a snapshot of the current processes. It displays information about a selection of the active processes.

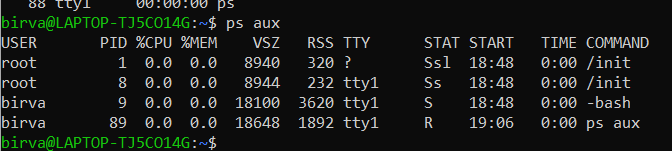
(ps –a) : Select all processes except both session leaders and processes not associated with a terminal.



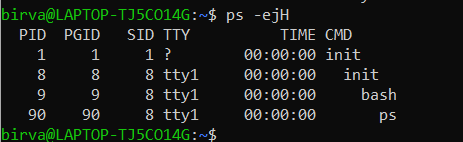
(ps -e) : Select all the processes.



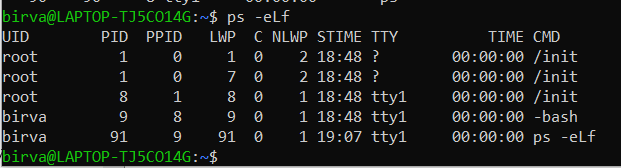
(ps aux) : It shows all the processes for all the users (‘x’ is the user).



(ps -ejH) : It is used to print process tree.



(ps -eLf) : It is used to get information about threads.

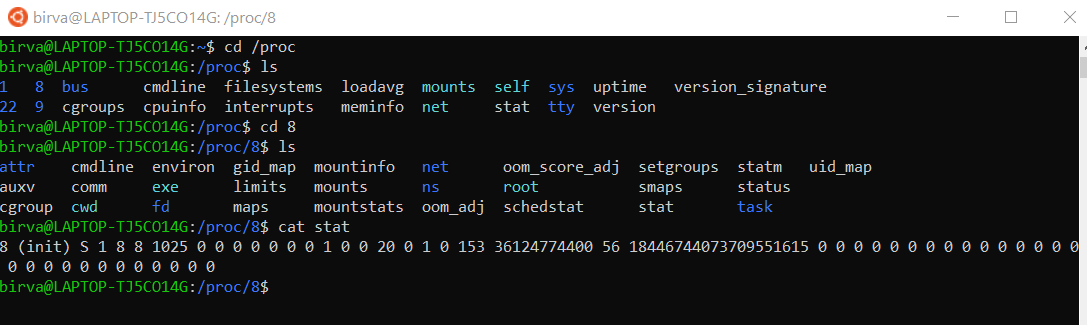


**2). ‘proc’**

The proc filesystem is a pseudo-filesystem which provides an interface to kernel data structures. It is commonly mounted at /proc. Typically, it is mounted automatically by the system, but it can also be mounted manually using a command such as:

mount -t proc proc /proc

Most of the files in the proc filesystem are read-only, but some files are writable, allowing kernel variables to be changed.



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

**CODE:**

#include<stdio.h>

#include<dirent.h>

#include<sys/types.h>

#include<string.h>

#include<stdlib.h>

#include<ctype.h>

#include<unistd.h>

int isNum(char a[])

{

for(int i=0; a[i]!=0; i++)

{

if(isdigit(a[i]) == 0)

{

return 0;

}

}

return 1;

}

void ps(char \*dirname)

{

DIR \*dirp = opendir(dirname);

struct dirent \*\*dir;

struct dirent \*d;

dir = (struct dirent \*\*)malloc(10000 \* sizeof(struct dirent \*));

if(dirp == NULL)

{

printf("Can't access '%s': No such file or Dir OR '%s' is a file",dirname,dirname);

return;

}

int i = 0;

char processinfo[100];

while(d = readdir(dirp))

{

if(d->d\_type == 4 && isNum(d->d\_name))

{

dir[i] = d;

i++;

}

}

char x[100] = "PID";

char y[100] = "P\_NAME";

printf("%5s %15s\n",x,y);

for(int j=0;j<i;j++)

{

char stat\_file[FILENAME\_MAX];

strcat(stat\_file, dirname);

strcat(stat\_file, "/");

strcat(stat\_file, dir[j]->d\_name);

strcat(stat\_file, "/stat");

FILE \*fptr;

fptr = fopen(stat\_file, "r");

if(fptr == NULL)

{

printf("Unable to open file %s\n",stat\_file);

}

fscanf(fptr, "%\*s %s ",processinfo);

printf("%5s %15s\n",dir[j]->d\_name,processinfo);

strcpy(stat\_file, "");

}

closedir(dirp);

free(d);

free(dir);

}

int main()

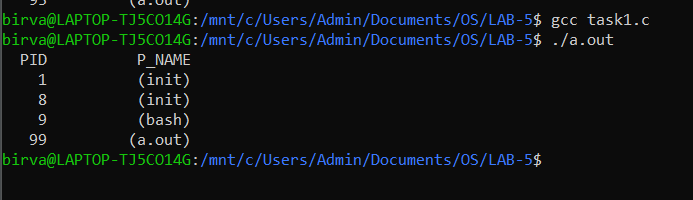
{

ps("/proc");

return 0;

}

**OUTPUT:**

****

**2). Extend the above program to read and display other fields from the stat file.**

**CODE:**

#include<stdio.h>

#include<dirent.h>

#include<sys/types.h>

#include<string.h>

#include<stdlib.h>

#include<ctype.h>

#include<unistd.h>

int isNum(char a[])

{

for(int i=0; a[i]!=0; i++)

{

if(isdigit(a[i]) == 0)

{

return 0;

}

}

return 1;

}

void ps(char \*dirname)

{

DIR \*dirp = opendir(dirname);

struct dirent \*\*dir;

struct dirent \*d;

dir = (struct dirent \*\*)malloc(10000 \* sizeof(struct dirent \*));

if(dirp == NULL)

{

printf("Can't access '%s': No such file or Dir OR '%s' is a file",dirname,dirname);

return;

}

int i = 0;

char processinfo[100];

while(d = readdir(dirp))

{

if(d->d\_type == 4 && isNum(d->d\_name))

{

dir[i] = d;

i++;

}

}

for(int j=0;j<i;j++)

{

char stat\_file[FILENAME\_MAX];

strcat(stat\_file, dirname);

strcat(stat\_file, "/");

strcat(stat\_file, dir[j]->d\_name);

strcat(stat\_file, "/stat");

FILE \*fptr;

fptr = fopen(stat\_file, "r");

if(fptr == NULL)

{

printf("Unable to open file %s\n",stat\_file);

}

printf("PID: %s\n",dir[j]->d\_name);

printf("More information: ");

while(fscanf(fptr, "%s ",processinfo) != EOF)

{

printf("%s ",processinfo);

}

printf("\n");

strcpy(stat\_file, "");

}

closedir(dirp);

free(d);

free(dir);

}

int main()

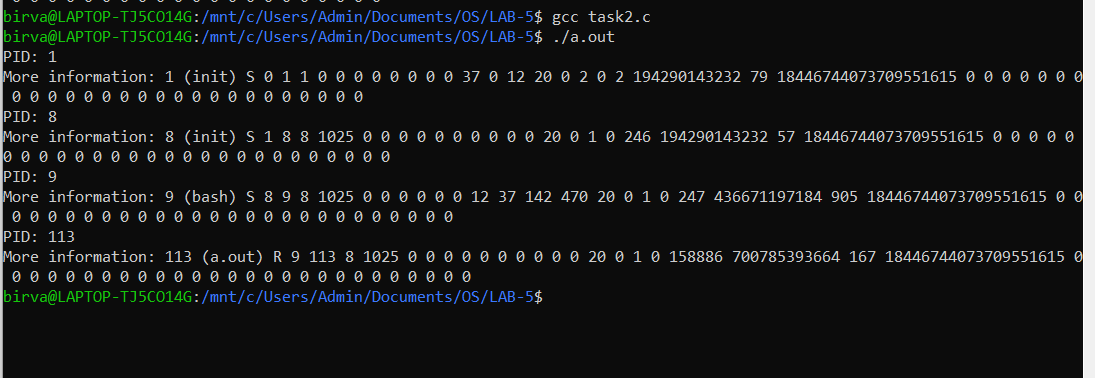
{

ps("/proc");

return 0;

}

**OUTPUT:**

****