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.

```
birva@LAPTOP-TJ5CO14G:~$ ps -a
  PID TTY          TIME CMD
    9 tty1     00:00:00 bash
   87 tty1     00:00:00 ps
birva@LAPTOP-TJ5CO14G:~$
```

(ps -e) : Select all the processes.

```
birva@LAPTOP-TJ5CO14G:~$ ps -e
  PID TTY          TIME CMD
    1 ?        00:00:00 init
    8 tty1     00:00:00 init
    9 tty1     00:00:00 bash
   88 tty1     00:00:00 ps
birva@LAPTOP-TJ5CO14G:~$
```

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

```
birva@LAPTOP-TJ5CO14G:~$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   8940   320 ?        Ssl  18:48   0:00 /init
root         8  0.0  0.0   8944   232 tty1     Ss   18:48   0:00 /init
birva        9  0.0  0.0  18100  3620 tty1     S    18:48   0:00 -bash
birva       89  0.0  0.0  18648  1892 tty1     R    19:06   0:00 ps aux
birva@LAPTOP-TJ5CO14G:~$
```

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

```
birva@LAPTOP-TJ5CO14G:~$ ps -ejH
  PID  PGID   SID TTY          TIME CMD
    1     1     1 ?        00:00:00 init
    8     8     8 tty1     00:00:00   init
    9     9     8 tty1     00:00:00     bash
   90    90     8 tty1     00:00:00       ps
birva@LAPTOP-TJ5CO14G:~$
```

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

```
birva@LAPTOP-TJ5CO14G:~$ ps -eLf
UID        PID  PPID   LWP  C NLWP STIME TTY          TIME CMD
root         1     0     1  0    2 18:48 ?        00:00:00 /init
root         1     0     7  0    2 18:48 ?        00:00:00 /init
root         8     1     8  0    1 18:48 tty1     00:00:00 /init
birva        9     8     9  0    1 18:48 tty1     00:00:00 -bash
birva       91     9    91  0    1 19:07 tty1     00:00:00 ps -eLf
birva@LAPTOP-TJ5CO14G:~$
```

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.

birva@LAPTOP-TJ5CO14G: /proc/8

```
birva@LAPTOP-TJ5CO14G:~$ cd /proc
birva@LAPTOP-TJ5CO14G:/proc$ ls
1   8  bus      cmdline  filesystems  loadavg  mounts  self  sys  uptime   version_signature
22  9  cgroups  cpuinfo  interrupts   meminfo  net     stat  tty  version
birva@LAPTOP-TJ5CO14G:/proc$ cd 8
birva@LAPTOP-TJ5CO14G:/proc/8$ ls
attr    cmdline  environ  gid_map  mountinfo   net      oom_score_adj  setgroups  statm   uid_map
auxv    comm     exe      limits   mounts      ns       root           smaps      status
cgroup  cwd      fd       maps     mountstats  oom_adj  schedstat      stat       task
birva@LAPTOP-TJ5CO14G:/proc/8$ cat stat
8 (init) S 1 8 8 1025 0 0 0 0 0 0 0 1 0 0 20 0 1 0 153 36124774400 56 18446744073709551615 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0
birva@LAPTOP-TJ5CO14G:/proc/8$
```

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:

```
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-5$ gcc task1.c
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-5$ ./a.out
  PID          P_NAME
    1          (init)
    8          (init)
    9          (bash)
   99         (a.out)
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-5$
```

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:

```
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-5$ gcc task2.c
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-5$ ./a.out
PID: 1
More information: 1 (init) S 0 1 1 0 0 0 0 0 0 0 0 37 0 12 20 0 2 0 2 194290143232 79 18446744073709551615 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
PID: 8
More information: 8 (init) S 1 8 8 1025 0 0 0 0 0 0 0 0 0 0 20 0 1 0 246 194290143232 57 18446744073709551615 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
PID: 9
More information: 9 (bash) S 8 9 8 1025 0 0 0 0 0 0 12 37 142 470 20 0 1 0 247 436671197184 905 18446744073709551615 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
PID: 113
More information: 113 (a.out) R 9 113 8 1025 0 0 0 0 0 0 0 0 0 0 20 0 1 0 158886 700785393664 167 18446744073709551615 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
birva@LAPTOP-TJ5CO14G:/mnt/c/Users/Admin/Documents/OS/LAB-5$
```