NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

PROGRAM: SOFTWARE ENGINEERING

OPERATING SYSTEMS LAB

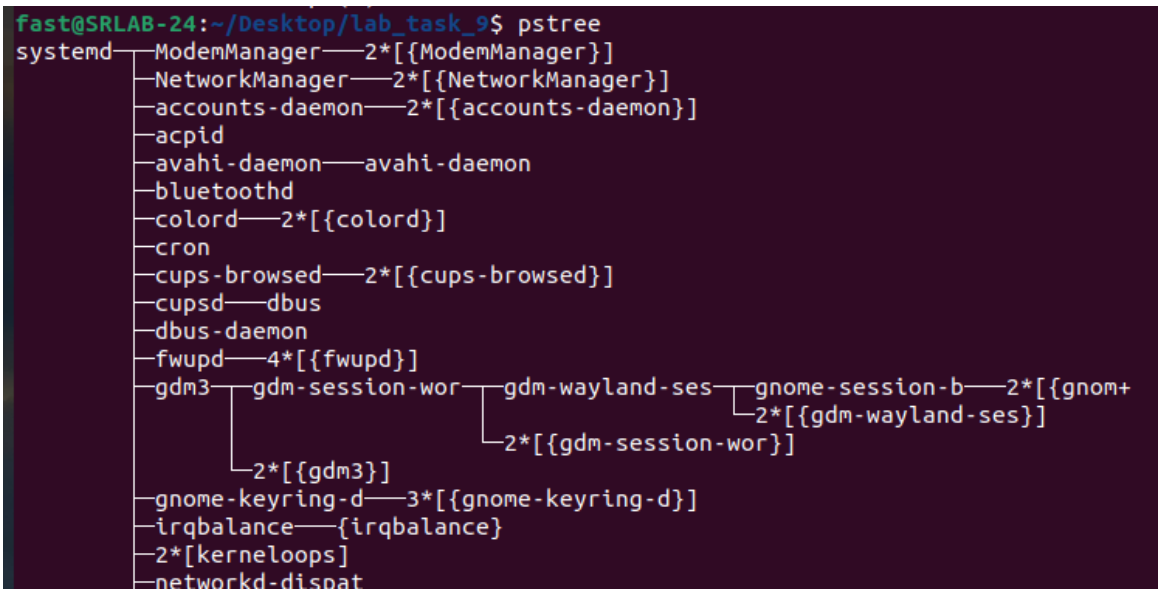
Lab_Task 09

SUBMITTED BY:
Name: Ahmed Ali
Roll No: 22P-9318
Section: BS(SE)-5B

INSTRUCTOR NAME: Sir Saad

A DEPARTMENT OF COMPUTER SCIENCE

Running Command of pstree:



Running Command of pstree | less: Fit the output according to monitor screen

```
        |          |-snap---15*[{snap}]
        |          |-snap-store---14*[{snap-store}]
        |          |-snapd-desktop-i---snapd-desktop-i---4*[{snapd-desktop-i}]
        |          |-tracker-miner-f---5*[{tracker-miner-f}]
        |          |-xdg-desktop-por---5*[{xdg-desktop-por}]
        |          |-2*[xdg-desktop-por---3*[{xdg-desktop-por}]]
        |          |-xdg-document-po-+-fusermount3
        |          |                 `-5*[{xdg-document-po}]
        |          `-xdg-permission----2*[{xdg-permission-}]
        |-systemd-journal
        |-systemd-logind
        |-systemd-oomd
        |-systemd-resolve
        |-systemd-udevd
        |-thermald---3*[{thermald}]
        |-udisksd---4*[{udisksd}]
        |-unattended-upgr---{unattended-upgr}
        |-upowerd---2*[{upowerd}]
        `-wpa_supplicant
(END)
```

Running Command of pstree | grep bash:

```
fast@SRLAB-24:~/Desktop/lab_task_9$ pstree | grep bash
        |          |-gnome-terminal--+-bash-+-grep
fast@SRLAB-24:~/Desktop/lab_task_9$
```

```
                    ├─gnome-control-c──24*[{gnome-control-c}]
                    ├─oosplash─┬─soffice.bin─┬─soffice.bin
                    │          │             └─4*[{soffice.bin}]
                    │          └─{oosplash}
                    └─37*[{gnome-shell}]
├─gnome-shell-cal──5*[{gnome-shell-cal}]
├─gnome-terminal──gnome-terminal.──3*[{gnome-terminal.}]
├─gnome-terminal-─┬─bash──pstree
                  └─3*[{gnome-terminal-}]
├─goa-daemon──3*[{goa-daemon}]
├─goa-identity-se──2*[{goa-identity-se}]
├─gsd-a11y-settin──3*[{gsd-a11y-settin}]
├─gsd-color──3*[{gsd-color}]
├─gsd-datetime──3*[{gsd-datetime}]
├─gsd-housekeepin──3*[{gsd-housekeepin}]
```

grep process receives it, searches for keyword, formats output, and then displays the result.

PIPE SYSTEM CALL:

The code creates a pipe using the pipe() system call. We have passed it the name of the integer array pfd that we have declared

```
fast@SRLAB-24:~/Desktop/lab_task_9$ vim pipe.c
fast@SRLAB-24:~/Desktop/lab_task_9$ cat pipe.c
#include <unistd.h>
int main()
{
int pfd[2];
pipe(pfd);
}
fast@SRLAB-24:~/Desktop/lab_task_9$ ./pipe.out
fast@SRLAB-24:~/Desktop/lab_task_9$
```

Task: Rewrite this code so that you can view the contents of the array using printf arguments. You should see two numbers.
What are these numbers?

```
fast@SRLAB-24:~/Desktop/lab_task_9$ vim new_pipe.c
fast@SRLAB-24:~/Desktop/lab_task_9$ cat new_pipe.c
#include <unistd.h>
#include <stdio.h>

int main()
{
    int pfd[2];
    pipe(pfd);
    printf("Read (pfd[0]): %d\n",pfd[0]);
    printf("(Write pfd[1]): %d\n",pfd[1]);
    return 0;
}
fast@SRLAB-24:~/Desktop/lab_task_9$ ./new_pipe.out
Read (pfd[0]): 3
(Write pfd[1]): 4
fast@SRLAB-24:~/Desktop/lab_task_9$
```

EXTENDED PIPE SYSTEM CALL:

Performing the same operations of open(), close(), read() and write() on the pipe.

```
fast@SRLAB-24:~/Desktop/lab_task_9$ vim extend_pipe.c
fast@SRLAB-24:~/Desktop/lab_task_9$ cat extend_pipe.c
#include<unistd.h>
int main()
{
int pid; // for storing fork() return
int pfd[2]; // for pipe file descriptors
char aString[20]; // Temporary storage
pipe(pfd); // create our pipe
pid = fork(); // create child process
if (pid == 0) // For child
{
write(pfd[1], "Hello", 5); // Write onto pipe
}
else // For parent
{
read(pfd[0], aString, 5); // Read from pipe
}
}
fast@SRLAB-24:~/Desktop/lab_task_9$ gcc extend_pipe.c -o extend_pipe.out
fast@SRLAB-24:~/Desktop/lab_task_9$ ./extend_pipe.out
fast@SRLAB-24:~/Desktop/lab_task_9$
```

Task: Rewrite this code so that you can see the contents of aString in the parent before and after the read() call. What are the contents? You will notice that Hello has been mentioned in the chid process.
Then how is it possible that we are able to see the term Hello in the parent process? The answer is through the pipe mechanism which we just used.

```
fast@SRLAB-24:~/Desktop/lab_task_9$ vim task_extend_pipe.c
fast@SRLAB-24:~/Desktop/lab_task_9$ cat task_extend_pipe.
cat: task_extend_pipe.: No such file or directory
fast@SRLAB-24:~/Desktop/lab_task_9$ cat task_extend_pipe.c
#include<unistd.h>
#include<stdio.h>
int main()
{
    int pid; // for storing fork() return
    int pfd[2]; // for pipe file descriptors
    char aString[20]; // Temporary storage
    pipe(pfd); // create our pipe
    pid = fork(); // create child process
    if (pid == 0)
    { //child
        write(pfd[1], "Hello",5); // Write onto pipe
        close(pfd[1]); //closing write pipe in the child process
    }
    else
    { //parent
        printf("Before read, aString contains: '%s'\n",aString);

        // read from the pipe
        read(pfd[0], aString, 5); // read from pipe into aString
        aString[5] ='\0';

        // After reading, print the contents of aString (it should be "Hello")
        printf("After read, aString contains: '%s'\n", aString);

        close(pfd[0]); // close read end of the pipe in the parent process
    }
    return 0;
}
fast@SRLAB-24:~/Desktop/lab_task_9$ gcc task_extend_pipe.c -o task_extend_pipe.out
fast@SRLAB-24:~/Desktop/lab_task_9$ ./task_extend_pipe.out
Before read, aString contains: '◆m◆◆'
After read, aString contains: 'Hello'
fast@SRLAB-24:~/Desktop/lab_task_9$
```

The parent can see the word **Hello** after reading from the pipe because the pipe allows the parent and child processes to share data. The child writes **Hello** to the pipe, and the parent reads it.

NOW adding close(pfd[0]) in child process and close(pfd[1]) in parent process

```
fast@SRLAB-24:~/Desktop/lab_task_9$ vim close_task_extend_pipe.c
fast@SRLAB-24:~/Desktop/lab_task_9$ cat close_task_extend_pipe.c
#include<unistd.h>
#include<stdio.h>
int main()
{
    int pid; // for storing fork() return
    int pfd[2]; // for pipe file descriptors
    char aString[20]; // Temporary storage
    pipe(pfd); // create our pipe
    pid = fork(); // create child process
    if (pid == 0)
    { //child
        close(pfd[0]);
        write(pfd[1], "Hello",5); // Write onto pipe
        close(pfd[1]); //closing write pipe in the child process
    }
    else
    { //parent
        close(pfd[1]);
        printf("Before read, aString contains: '%s'\n",aString);

        // read from the pipe
        read(pfd[0], aString, 5); // read from pipe into aString
        aString[5] ='\0';

        // After reading, print the contents of aString (it should be "Hello")
        printf("After read, aString contains: '%s'\n", aString);

        close(pfd[0]); // close read end of the pipe in the parent process
    }
    return 0;
}
fast@SRLAB-24:~/Desktop/lab_task_9$ gcc close_task_extend_pipe.c -o close_task_extend_pipe.out
fast@SRLAB-24:~/Desktop/lab_task_9$ ./close_task_extend_pipe.out
Before read, aString contains: '        n◆◆'
After read, aString contains: 'Hello'
fast@SRLAB-24:~/Desktop/lab_task_9$
```

EXAMPLE CODE: OUTPUT

```
fast@SRLAB-24:~/Desktop/lab_task_9$ vim example.c
fast@SRLAB-24:~/Desktop/lab_task_9$ cat example.c
#include <unistd.h>
#include <string.h>
#include <stdio.h>
int main()
{
int pfd[2];
pipe(pfd);
if (fork() == 0)
{
close(pfd[1]);
dup2(pfd[0], 0);
close(pfd[0]);
execlp("wc", "wc", (char *) 0);
}
else
{
close(pfd[0]);
dup2(pfd[1], 1);
close(pfd[1]);
execlp("ls", "ls", (char *) 0);
}
}
fast@SRLAB-24:~/Desktop/lab_task_9$ gcc example.c -o example.out
fast@SRLAB-24:~/Desktop/lab_task_9$ ./example.out
fast@SRLAB-24:~/Desktop/lab_task_9$       12       12      184
as
as
aw



^Z
[1]+  Stopped                 as
fast@SRLAB-24:~/Desktop/lab_task_9$
```