

**Institute of Engineering & Technology (IET)**

**J.K. Lakshmipat University**

**CS1108: Operating System(OS)**

**Instructors:**

Ankur Sharma

**Submitted by:**

*Kushagra Gupta (2022BTech051)*

**Section:**

A

**Assignment**

*Users and Forking*

**1) Create a program that prints numbers 1,2,.... on screen along with the PID of the program. Use usleep to print these numbers slowly. Use a fork before the print statement to create a new process and see what happens. Before starting the printing use getchar and ask the user to press a key to continue.**

**On a separate terminal use ps or pstree to see which processes are active when.**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

int ques1() {

pid\_t pid;

printf("Press any key to start...\n");

getchar();

pid = fork();

int i = 1;

while (1) {

// pid = fork();

printf("PID: %d, %d\n", getpid(), i);

usleep(1000000); // Sleep for 1 seconds

i++;

}

return 0;

}

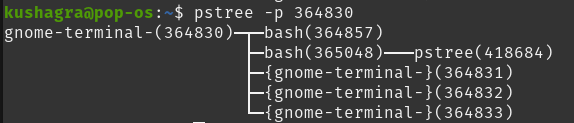
int main() {

ques1();

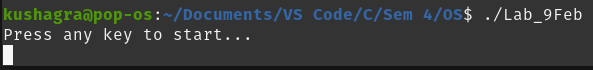
}

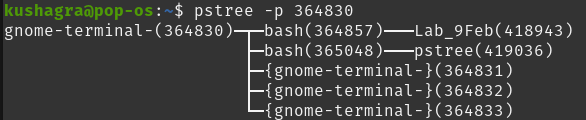
**Output:**

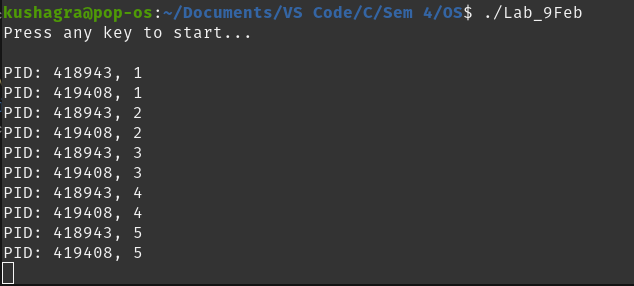
Firstly, the pstree for the terminal is like this:



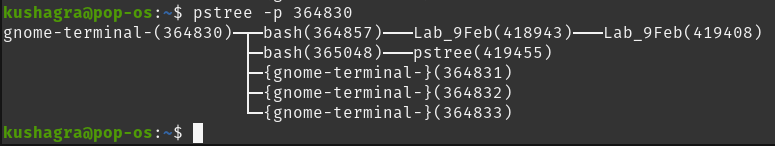
Then after running the C file, the pstree shows the file running:





And when the key is pressed and the fork process also started in the program: 

The tree is shown like this after that:



**2) Define an int x=10 and then fork. Save the return value of fork in a variable called r.**

**Print the value and the address of x. Also print the value and address of r.**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

int ques2() {

pid\_t r;

int x=10;

r = fork();

printf("PID of x: %d, Address of x: %p, x=%d\n", getpid(), &x, x);

printf("Address of r: %p, Value of r=%d\n", &r, r);

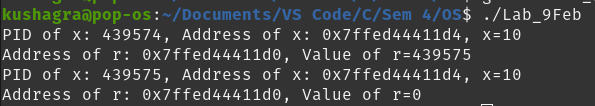
}

int main() {

ques2();

}

**Output:**



**3) Define a structure person containing name and age.**

**Create a person on heap using malloc.**

**Fork. Modify the age in child process. Print the memory address of the this person and the name and age and check what happens.**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

struct person {

char\* name;

int age;

};

int ques3() {

// Create a person on the heap using malloc

struct person \*p = (struct person \*)malloc(sizeof(struct person));

if (p == NULL) {

fprintf(stderr, "Memory allocation failed\n");

return 1;

}

p->name = "Kushagra";

p->age = 20;

pid\_t pid = fork();

if (pid == 0) { // Child process

p->age += 1;

printf("Child Process - PID: %d\n", getpid());

printf("Address of p: %p\n", (void \*)p);

printf("Name: %s, Age: %d\n", p->name, p->age);

}

else { // Parent process

printf("Parent Process - PID: %d\n", getpid());

printf("Address of p: %p\n", (void \*)p);

printf("Name: %s, Age: %d\n", p->name, p->age);

}

free(p);

return 0;

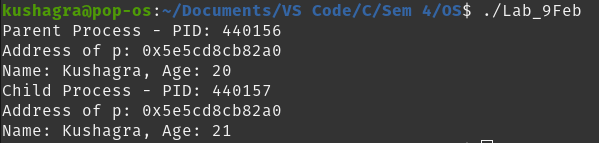
}

int main() {

ques3();

}

**Output:**



**4) Create a user called xyz using the command**

**sudo useradd -m xyz**

**Create its passwd using the command**

**sudo passwd xyz**

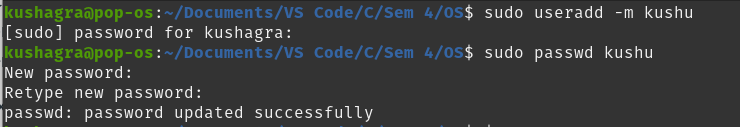
**(As the original user) create a folder called usertest. Go to this folder. Create a program hello that prints "Hello World" on the screen. Create another program bye that prints "Goodbye World" on the screen.**

**Remove the execute permission for others from bye using chmod.**

**Now switch to user xyz and go to the above folder and try to run the programs.**

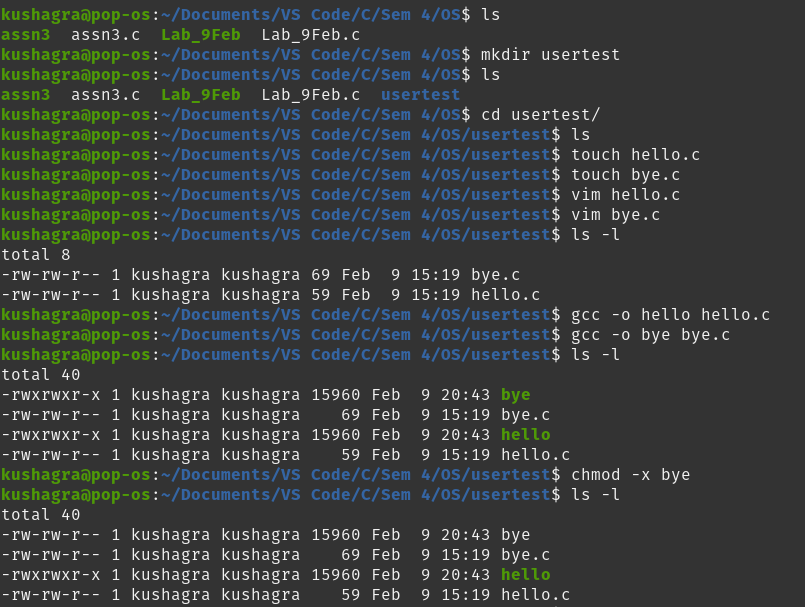
**Output:**

New user called kushu is added and password is set:



A folder usertest is made and in it, 2 programs hello.c and bye.c are made and then compiled.

After that the executions permisson is removed for bye file.



After switching to the “kushu” user, both hello and bye files are executed, from which only hello file is executed and the bye file showed an permission error.

