Write Linux C program to create two processes P1 and P2. P1 takes a string and passes it to P2. P2 concatenates the received string with another string without using string function and sends it back to P1 for printing.
Examples:
Other string is: forgeeks.org
Input : www.geeks
Output : www.geeksforgeeks.org
Input : www.practice.geeks
Output : practice.geeksforgeeks.org
Explanation:
- To create child process we use fork(). fork() returns :
- <0 fail to create child (new) process
- =0 for child process
- >0 i.e process ID of the child process to the parent process. When >0 parent process will execute.

- pipe() is used for passing information from one process to another. pipe() is unidirectional therefore, for two-way communication between processes, two pipes can be set up, one for each direction.
- Example:
- int fd[2];
- pipe(fd);
- fd[0]; //-> for using read end
- fd[1]; //-> for using write end

Inside Parent Process : We firstly close the reading end of first pipe (fd1[0]) then write the string though writing end of the pipe (fd1[1]). Now parent will wait until child process
is finished. After the child process, parent will close the writing end of second pipe(fd2[1]) and read the string through reading end of pipe (fd2[0]).
Inside Child Process : Child reads the first string sent by parent process by closing the writing end of pipe (fd1[1]) and after reading concatenate both string and passes the string to parent process via fd2 pipe and will exit.
Program

```
#include
#include
#include
#include
#include
#include
int main()
{
// We use two pipes
// First pipe to send input string from parent
```

```
// Second pipe to send concatenated string from child
int fd1[2]; // Used to store two ends of first pipe
int fd2[2]; // Used to store two ends of second pipe
char fixed_str[] = "forgeeks.org";
char input_str[100];
pid_t p;
if (pipe(fd1)==-1)
{
fprintf(stderr, "Pipe Failed" );
return 1;
}
if (pipe(fd2)==-1)
{
fprintf(stderr, "Pipe Failed" );
return 1;
}
scanf("%s", input_str);
p = fork();
if (p < 0)
{
fprintf(stderr, "fork Failed" );
return 1;
}
7
// Parent process
else if (p > 0)
{
char concat_str[100];
close(fd1[0]); // Close reading end of first pipe
// Write input string and close writing end of first
// pipe.
write(fd1[1], input_str, strlen(input_str)+1);
close(fd1[1]);
// Wait for child to send a string
wait(NULL);
close(fd2[1]); // Close writing end of second pipe
// Read string from child, print it and close
// reading end.
read(fd2[0], concat_str, 100);
printf("Concatenated string %s\n", concat_str);
close(fd2[0]);
}
// child process
else
```

```
{
close(fd1[1]); // Close writing end of first pipe
// Read a string using first pipe
8
char concat_str[100];
read(fd1[0], concat_str, 100);
// Concatenate a fixed string with it
int k = strlen(concat_str);
int i;
for (i=0; i concat_str[k++] = fixed_str[i];
concat_str[k] = '\0'; // string ends with '\0'
// Close both reading ends
close(fd1[0]);
close(fd2[0]);
// Write concatenated string and close writing end
write(fd2[1], concat_str, strlen(concat_str)+1);
close(fd2[1]);
exit(0);
}
```