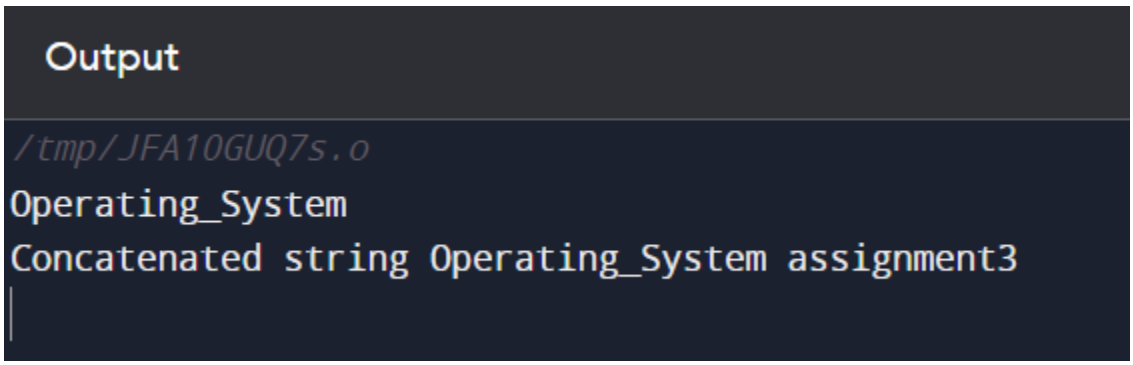
//// C program to demonstrate use of fork() and pipe()

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

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[] = " assignment3";
	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;
	}

	// 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
                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 < strlen(fixed_str); 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);
        }
}
```



//Two-way Communication Using Pipes

```
#include<stdio.h>
#include<unistd.h>
int main() {
int pipefds1[2], pipefds2[2];
int returnstatus1, returnstatus2;
int pid;
char pipe1writemessage[20] = "Hi";
char pipe2writemessage[20] = "Hello";
char readmessage[20];
returnstatus1 = pipe(pipefds1);
if (returnstatus1 == -1) {
printf("Unable to create pipe 1 \n");
return 1;
}
returnstatus2 = pipe(pipefds2);
if (returnstatus2 == -1) {
printf("Unable to create pipe 2 \n");
return 1;
```

```
}
pid = fork();
if (pid != 0) // Parent process
{
close(pipefds1[0]); // Close the unwanted pipe1 read side
close(pipefds2[1]); // Close the unwanted pipe2 write side
printf("In Parent: Writing to pipe 1 – Message is%s\n", pipe1writemessage);
write(pipefds1[1], pipe1writemessage,sizeof(pipe1writemessage));
read(pipefds2[0], readmessage,sizeof(readmessage));

printf("In Parent: Reading from pipe 2 – Message is %s\n", readmessage);
} else { //child process
close(pipefds1[1]); // Close the unwanted pipe1 write side
close(pipefds2[0]); // Close the unwanted pipe2 read side
read(pipefds1[0], readmessage,sizeof(readmessage));
printf("In Child: Reading from pipe 1 – Message is %s\n", readmessage);
printf("In Child: Writing to pipe 2 – Message is %s\n", pipe2writemessage);
write(pipefds2[1], pipe2writemessage,sizeof(pipe2writemessage));
}
return 0;
}
```