Computer Networks
Lab Assignment - 8
CSE 325

4 April 2022

Gyanendra Kumar Shukla
CSE 1
191112040

Department of Computer Science
NIT, Bhopal

Contents

1 Program to write and read two messages using pipe. 2
1.1 CODE 2
1.1.1 OUTPUT 3

2 Program to write and read two messages through the pipe using the parent and the child processes. 4
2.1 CODE 4
2.1.1 OUTPUT 5

1 Program to write and read two messages using pipe.

1.1 CODE

```cpp
// Program to write and read two messages using pipe.
#include <iostream>
#include <unistd.h>
#include <array>

using std::cout;
int main()
{
    std::array<int, 2> pipe_file_descriptors;
    int returnstatus;
    char writemessages[2][20] = {"Message", "To Earth"};
    char readmessage[20];
    returnstatus = pipe(pipe_file_descriptors.data());

    if (returnstatus == -1){
        cout << "Unable to create pipe\n";
        return 1;
    }

    cout<<"Writing to pipe - Message 1 is "<<  writemessages[0] << "\n";
    write(pipe_file_descriptors[1], writemessages[0], sizeof(writemessages[0]));

    read(pipe_file_descriptors[0], readmessage, sizeof(readmessage));
    cout<<"Reading from pipe - Message 1 is " <<readmessage << "\n";

    cout<<"Writing to pipe - Message 2 is " << writemessages[1] <<"\n";
    write(pipe_file_descriptors[1], writemessages[1], sizeof(writemessages[1]));

    read(pipe_file_descriptors[0], readmessage, sizeof(readmessage));
    cout<<"Reading from pipe - Message 2 is "<< readmessage <<"\n";
}
```

1.1.1 OUTPUT

```
Writing to pipe - Message 1 is Message
Reading from pipe - Message 1 is Message
Writing to pipe - Message 2 is To Earth
Reading from pipe - Message 2 is To Earth
```

2 Program to write and read two messages through the pipe using the parent and the child processes.

2.1 CODE

```
// Program to write and read two messages through the pipe using the parent and the
↪  child processes.
#include <iostream>
#include <unistd.h>
#include  <array>

using std::cout;

int main() {
   std::array<int,2> pipe_file_descpritor;
   int returnstatus;
   int pid;

   char writemessages[2][20]={"Message", "To the World"};
   char readmessage[20];

   returnstatus = pipe(pipe_file_descpritor.data());
   if (returnstatus == -1) {
      cout<<"Unable to create pipe\n";
      return 1;
   }
   pid = fork();

   // Child process
   if (pid == 0) {
      read(pipe_file_descpritor[0], readmessage, sizeof(readmessage));
      cout << "Child Process - Reading from pipe - Message 1 is " << readmessage <<
      ↪  "\n";
      read(pipe_file_descpritor[0], readmessage, sizeof(readmessage));
      cout << "Child Process - Reading from pipe - Message 2 is " << readmessage <<
      ↪  "\n";
   } else { //Parent process
      cout << "Parent Process - Writing to pipe - Message 1 is " << writemessages[0]
      ↪  << "\n";
      write(pipe_file_descpritor[1], writemessages[0], sizeof(writemessages[0]));
      cout << "Parent Process - Writing to pipe - Message 2 is " << writemessages[1]
      ↪  << "\n";
      write(pipe_file_descpritor[1], writemessages[1], sizeof(writemessages[1]));
   }
   return 0;
}
```

2.1.1 OUTPUT

```
Parent Process - Writing to pipe - Message 1 is Message
Parent Process - Writing to pipe - Message 2 is To the World
Child Process - Reading from pipe - Message 1 is Message
Child Process - Reading from pipe - Message 2 is To the World
```