

# B.M.S College of Engineering

P.O. Box No.: 1908 Bull Temple Road,

Bangalore-560 019

## DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING



Course – Unix System Programming

Course Code – 19IS4PWUSP

AY 2019-20

Report on Unix System Programming Project  
Simple games Implemented using Named pipe API

Submitted by

Gowrishankar G - 1BM18IS039

Harshit Chelladurai - 1BM18IS040

Submitted to

Shobana T S

# B.M.S College of Engineering

P.O. Box No.: 1908 Bull Temple Road,

Bangalore-560 019

## DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING



## CERTIFICATE

Certified that the Project has been successfully presented at B.M.S College Of Engineering by **Gowrishankar G and Harshit Chelladurai** bearing USN: **1BM18IS039 and 1BM18IS040** in partial fulfilment of the requirements for the IV Semester degree in Bachelor of Engineering in Information Science & Engineering of Visvesvaraya Technological University, Belgaum as a part of project for the course **Unix System Programming and 19IS4PWUSP** during academic year 2019-2020.

Faculty Name – Shobana T S

Designation – Assistant Professor

Department of ISE, BMSCE

## Table of Contents

Sl no	Topics	Page No
1	Abstract	4
2	Introduction	5
3	Problem Statement	6
4	API used in the project	7
5	Explanation of the API	7
6	Implementation	8
7	Results	21
8	Conclusions and Further Enhancements	25
9	References	26

# Abstract

An Operating System is a program that sits between the hardware and the application programs. Like any other program it has a `main()` function and it is built like any other program with a compiler and a linker

UNIX is a computer operating system originally developed by researchers at Bell Laboratories in 1969. The term is also used to refer to later operating systems based in part on its source code. Several of the original UNIX operating system's features, such as its hierarchical file system and multiuser support, became standard in later systems.

A shell script is a computer program designed to be run by the Unix shell, a command-line interpreter. The various dialects of shell scripts are considered to be scripting languages. Typical operations performed by shell scripts include file manipulation, program execution, and printing text.

An application program interface (API) is a set of routines, protocols, and tools for building software applications. Basically, an API specifies how software components should interact. Additionally, APIs are used when programming graphical user interface (GUI) components.

# Introduction

In this project we needed to develop a simple game that changes its output based on user input, we developed two games:

- Hangman
- Word Builder

These two games are made to be two played games. We achieve the two players feature in this project by using an api called named piped which are used for inter-process communication.

The hangman game we have developed is a 3 round game that assigns player 1 to be the game master, meaning player 1 decides on the question and player 2 has to guess. Player 2 has five tries to guess the right answer.

The word builder game is where player 1 starts with a word and player 2 should type a word that begins with the last character of the word player 1 gives then player 1 should type a word begins with the last character of the word player 2 gives. The one to break this chain loses.

# Problem Statement

A simple game that gives different outputs based on user inputs.

We have developed two games that change based on user inputs, they are:

- Hangman
- Word Builder

The API we used here is:

- Named Pipes/FIFO

# API used in the project

We used named pipes in this project to achieve the two-player aspect of the game.

## Explanation of the API used

### Named Pipes:

Named pipes is an api also known as FIFO that is used to achieve inter-process communication through shared memory space. They last as long as the system is up, beyond the life of the process. It can be deleted if no longer used.

Named pipes appear as a file and generally processes attach to it for inter-process communication. It's a special file on the local storage which allows two or more processes to communicate with each other by reading or writing to or from this file. Once we have created this special file, any process can open it for reading or writing, in the same way as an ordinary file. However, it has to be open at both ends simultaneously before you can proceed to do any input or output operations on it.

To create a FIFO file we use the mkfifo function:

```
#include<stdio.h>

#include<sys/stat.h>

#include<unistd.h>

int mkfifo(const char* path_name, mode_t mode);
```

Syntax: mkfifo(pathname, mode);

Example: mkfifo("/tmp/myfifo", 0666);

# Implementation

## Hangman Code:

### (Player 1)

```
#include <bits/stdc++.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#define ngames 3

using namespace std;

int main(){

    int fd, i=0, score=0;

    char myfifo[] = "myfifo";

    mkfifo(myfifo, 0666);

    system("clear");

    char arr1[80]="-", arr2[80];

    while (i<ngames)

    {

        system("clear");

        cout<<"You're playing hangman!!\t\t\tScore:"<<score<<"\n";

        cout<<"\nPrevious Outcome: ";

        printf("%s\n", arr1);

        // Open FIFO for write only

        fd = open(myfifo, O_WRONLY);
```



```

cout<<"\nEnter a sentence: ";
fgets(arr2, 80, stdin);

write(fd, arr2, strlen(arr2)+1);
close(fd);

// Open FIFO for Read only
fd = open(myfifo, O_RDONLY);

// Read from FIFO
read(fd, arr1, sizeof(arr1));

system("clear");

if(arr1[4]=='W') score++;

cout<<"You're playing hangman!!\t\t\tScore:"<<score<<"\n";
cout<<"\nSentence was: "<<arr2<<endl;
cout<<"Previous Outcome: ";

printf("%s\n", arr1);
close(fd);

i++;
}

cout<<"Overall Outcome: ";
if(score==(double)ngames/2) cout<<"Draw\n";
else if(score<(double)ngames/2) cout<<"You Lost\n";
else cout<<"You Won!\n";

system("rm myfifo");
return 0;
}

```

## (Player 2)

```
#include <bits/stdc++.h>
```

```
#include <fcntl.h>
```

```
#include <sys/stat.h>
```

```
#include <sys/types.h>
```

```
#include <unistd.h>
```

```
#define ngames 3
```

```
using namespace std;
```

```
int hangman(char c[], int &score){
```

```
    char b[100], a;
```

```
    map<char, vector<int>> m;
```

```
    set<char> ss;
```

```
    int s, tries=5;
```

```
    int i, j=0;
```

```
    for(i = 0; c[i]!='\0'; i++)
```

```
        c[i] = tolower(c[i]);
```

```
    strcpy(b, c);
```

```
    for(i=0; b[i+1]!='\0'; i++){
```

```
        if(b[i]!=' '){
```

```
            m[b[i]].push_back(i);
```

```
            b[i]='-';
```

```
            j++;
```

```
        }
```

```
    }
```

```
    s=j;
```

```
    while(s!=0 && tries!=0){
```

```

system("clear");

cout<<"You're playing hangman!!\t\t\tScore:"<<score<<"\n\n";

cout<<"Your question: ";

puts(b);

cout<<"Tries Left: "<<tries<<endl;

cout<<"Enter your guess: ";

cin>>a;

if(m.find(a)==m.end()) tries--;

else if(ss.find(a)!=ss.end()) continue;

else{

    for(auto x: m[a]){

        b[x]=a;

    }

    s-=m[a].size();

    ss.insert(a);

}

}

system("clear");

cout<<"You're playing hangman!!\n\n";

cout<<"The answer is: "<<c;

if(s==0){

    score++;

    cout<<"Score:"<<score<<"\n";

    cout<<"\nPrevious Outcome: ";

    cout<<"You Won!!\n";

    return 0;

}

else{

```

```

        cout<<"Score:"<<score<<"\n";
        cout<<"\nPrevious Outcome: ";
        cout<<"You Lost!!\n";
        return 1;
    }
}

int main() {
    int fd1, i=0, score=0, w=-1;

    cout<<"You're playing hangman!!\t\t\tScore:"<<score<<"\n";

    // FIFO file path
    char myfifo [] = "myfifo", a;

    // Creating the named file(FIFO)
    // mkfifo(<pathname>,<permission>)
    mkfifo(myfifo, 0666);

    char str1[80]="-", str2[]="You Won!!", str3[]="You Lost!!", str[80]="-";
    while (i<ngames){

        system("clear");

        cout<<"You're playing hangman!!\t\t\tScore:"<<score<<"\n";
        cout<<"\nPrevious Answer: "<<str1<<"\n";
        cout<<"Previous Outcome: ";

        switch(w){
            case -1: cout<<"-\n"; break;
            case 0: cout<<str3<<endl; break;
            default: cout<<str2<<endl;

```

```

}

// First open in read only and read
fd1 = open(myfifo,O_RDONLY);
read(fd1, str1, 80);


// Print the read string and close
close(fd1);
fd1 = open(myfifo,O_WRONLY);
w=hangman(str1, score)==0;
if(w) write(fd1, str3, strlen(str3)+1);
else write(fd1, str2, strlen(str2)+1);
close(fd1);
i++;
}

cout<<"Overall Outcome: ";
if(score==(double)ngames/2) cout<<"Draw\n";
else if(score<(double)ngames/2) cout<<"You Lost\n";
else cout<<"You Won!\n";
return 0;
}

```

## Word Builder Code:

### (Player 1)

```
#include <bits/stdc++.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#include <string.h>

#include <fstream>

#define ngames 3


using namespace std;


int main()
{
    system("clear");
    cout<<"\n\nNOW PLAYING WORD BUIDING\n\n";


    char wb[] = "wb";
    mkfifo(wb, 0666);
    char fifo2[] = "fifo2";
    mkfifo(fifo2, 0666);
    char word1[20], word2[20];
    char ch1, ch2;
    int flag1=1;
    int index;
    int fp;
    char b,a;
    int sc1=1, j=0;
```

```

int sc2, z=0;

cout<<"Enter the first word\n";

fp = open(wb, O_WRONLY);

cout<<"\nEnter your word: ";

scanf("%s" , word1);

while(word1[z]!='\0'){

    word1[z]=tolower(word1[z]);

    z++;

}

write(fp, word1, strlen(word1)+1);

index = (strlen(word1)-1);

a= word1[index];

close(fp);

while(flag1==1)

{

    z=0;

    fp = open(wb, O_RDONLY);

    read(fp, word2, sizeof(word2));

    cout<<"player2: "<<word2<<endl;

    b= word2[0];

    close(fp);

    if(a==b)

    {

        flag1=1;

        sc1++;

    }

```

```

else if(a!=b){

    cout<<"\n\n\nYOU WON\n\n\n";

    flag1=0;

    break;

}

j++;

system("clear");

cout<<"\n\nNOW PLAYING WORD BUIDING\n\n";

cout<<"Your word: "<<word1<<endl;

cout<<"player2: "<<word2<<endl;

fp = open(wb, O_WRONLY);

cout<<"\nEnter your word: ";

scanf("%s" , word1);

while(word1[z]!='\0'){

    word1[z]=tolower(word1[z]);

    z++;

}

write(fp, word1, strlen(word1)+1);

index = (strlen(word1)-1);

a= word1[index];

close(fp);


if(word1[0]!=word2[strlen(word2)-1]){

    cout<<"\n\n\nYOU LOST\n\n\n";

    break;

}

}

return 0;

}

```



## (Player 2)

```
#include <bits/stdc++.h>
```

```
#include <fcntl.h>
```

```
#include <sys/stat.h>
```

```
#include <sys/types.h>
```

```
#include <unistd.h>
```

```
#include <string.h>
```

```
#include <fstream>
```

```
#define ngames 3
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    system("clear");
```

```
    cout<<"\n\nNOW PLAYING WORD BUIDING\n\n";
```

```
    char wb[] = "wb";
```

```
    mkfifo(wb, 0666);
```

```
    char fifo2[] = "fifo2";
```

```
    mkfifo(fifo2, 0666);
```

```
    int fp1;
```

```
    char w1[20], w2[20];
```

```
    char x,y;
```

```
    int index;
```

```
    int flag2=1;
```

```
    int j=0;
```

```
    int sc2=1;
```

```
    int sc1, z=0;
```

```

fp1 = open(wb, O_RDONLY);
read(fp1, w1, 20);
cout<<"player1: "<<w1<<endl;
index= (strlen(w1)-1);
x= w1[index];
close(fp1);

```

```

fp1 = open(wb, O_WRONLY);
cout<<"\nEnter your word: ";
scanf("%s", w2);
while(w2[z]!='\0'){
    w2[z]=tolower(w2[z]);
    z++;
}
write(fp1, w2, strlen(w2)+1);
y= w2[0];
close(fp1);

```

```

if(x==y)
{
    flag2=1;
}
else if(x!=y)
{
    cout<<"\n\n\nYOU LOST\n\n\n";
    flag2=0;
}

```

```

while(flag2==1)
{
    z=0;

    system("clear");

    cout<<"\n\nNOW PLAYING WORD BUIDING\n\n";

    cout<<"Your word: "<<w2<<endl;


    fp1 = open(wb, O_RDONLY);
    read(fp1, w1, 20);
    cout<<"player1: "<<w1<<endl;
    index= (strlen(w1)-1);
    x= w1[index];
    close(fp1);


    if(w1[0]!=w2[strlen(w2)-1]){
        cout<<"\n\n\nYOU WON!!!\n\n\n";
        break;
    }


    fp1 = open(wb, O_WRONLY);
    cout<<"\nEnter your word: ";
    scanf("%s", w2);
    while(w2[z]!='\0'){
        w2[z]=tolower(w2[z]);
        z++;
    }
    write(fp1, w2, strlen(w2)+1);
    y= w2[0];
    close(fp1);

```

```
    if(x==y)
    {
        flag2=1;
    }
    else if(x!=y)
    {
        cout<<"\n\n\nYOU LOST\n\n\n";
        flag2=0;
        break;
    }
}
return 0;
}
```

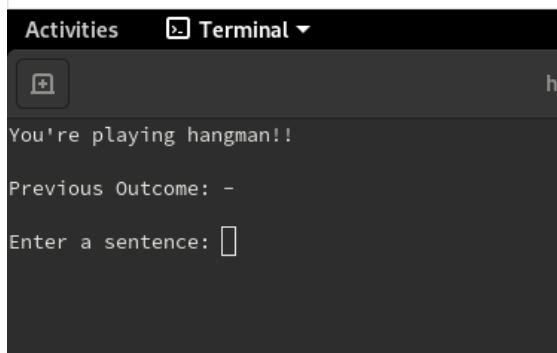
# Results

## Hangman Output:

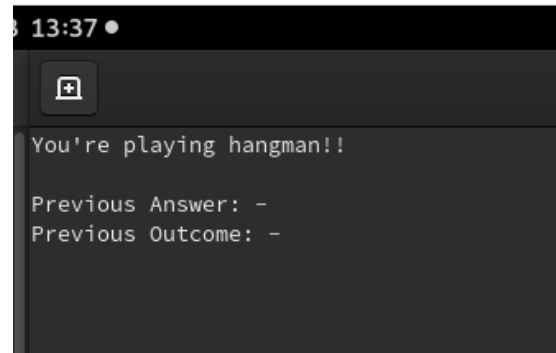
Player 1

Player 2

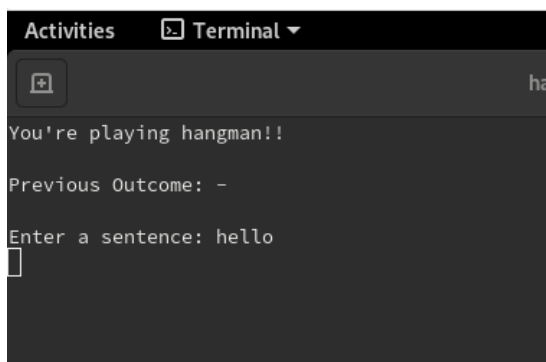
Initially:



```
Activities Terminal
You're playing hangman!!
Previous Outcome: -
Enter a sentence: 
```

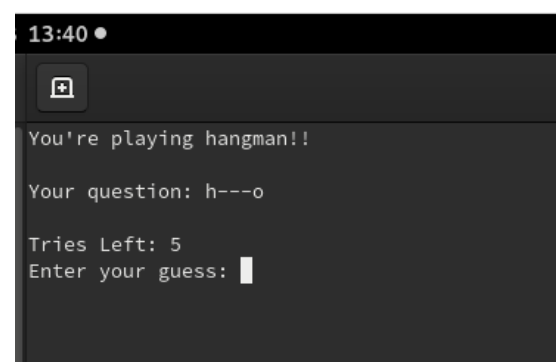


```
13:37
You're playing hangman!!
Previous Answer: -
Previous Outcome: -
```



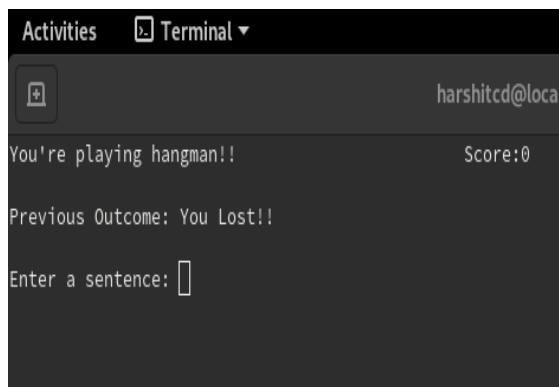
```
Activities Terminal
You're playing hangman!!
Previous Outcome: -
Enter a sentence: hello

```

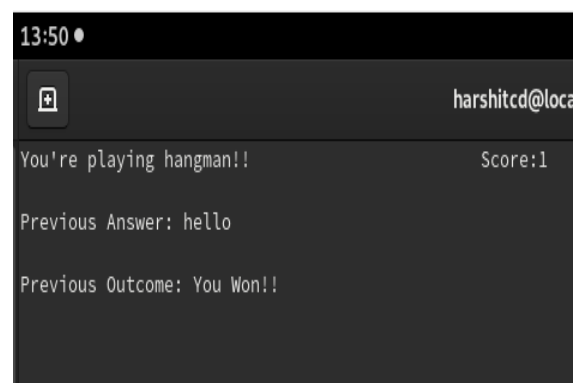


```
13:40
You're playing hangman!!
Your question: h---o
Tries Left: 5
Enter your guess: 
```

End of each round:

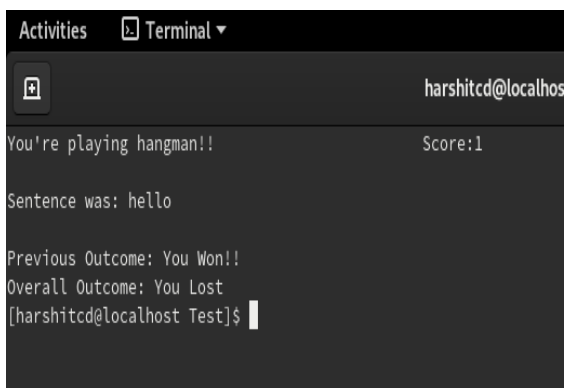


```
Activities Terminal
harshitcd@localhost
You're playing hangman!! Score:0
Previous Outcome: You Lost!!
Enter a sentence: 
```

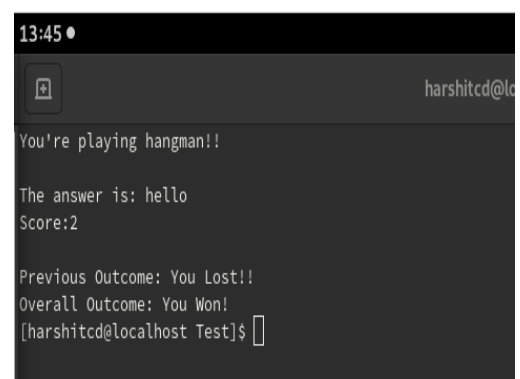


```
13:50
harshitcd@localhost
You're playing hangman!! Score:1
Previous Answer: hello
Previous Outcome: You Won!!
```

End of the game (Final standings):



```
Activities Terminal
harshitcd@localhost
You're playing hangman!! Score:1
Sentence was: hello
Previous Outcome: You Won!!
Overall Outcome: You Lost
[harshitcd@localhost Test]$
```



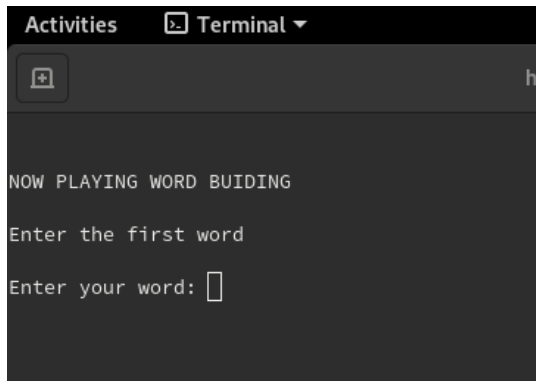
```
13:45
harshitcd@localhost
You're playing hangman!!
The answer is: hello
Score:2
Previous Outcome: You Lost!!
Overall Outcome: You Won!
[harshitcd@localhost Test]$
```

## Word Builder Output:

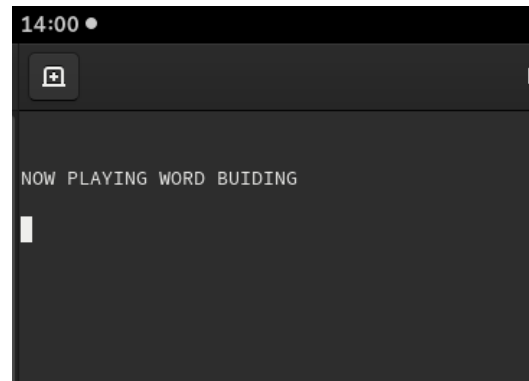
Player 1

Player 2

Initially:

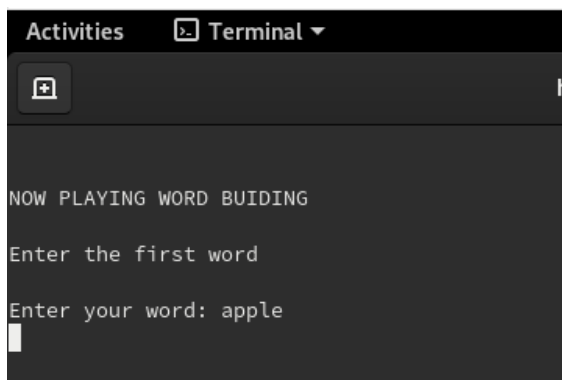


```
Activities Terminal ▾  
[+]  
h  
NOW PLAYING WORD BUIDING  
Enter the first word  
Enter your word: 
```



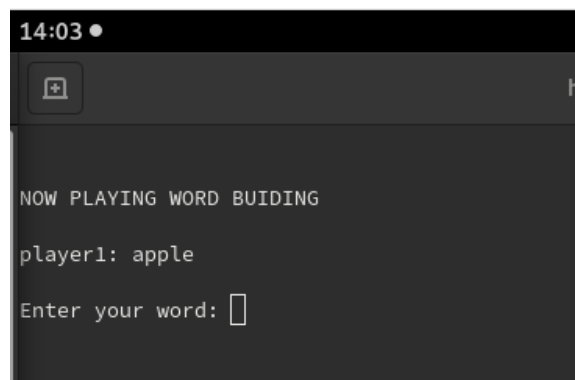
```
14:00 ●  
[+]  
h  
NOW PLAYING WORD BUIDING  

```



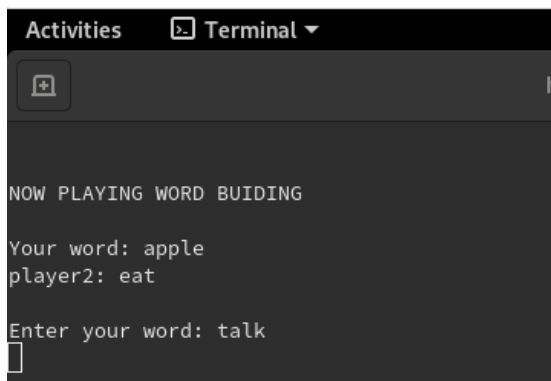
```
Activities Terminal ▾  
[+]  
h  
NOW PLAYING WORD BUIDING  
Enter the first word  
Enter your word: apple  

```



```
14:03 ●  
[+]  
h  
NOW PLAYING WORD BUIDING  
player1: apple  
Enter your word: 
```

In the middle of the game:

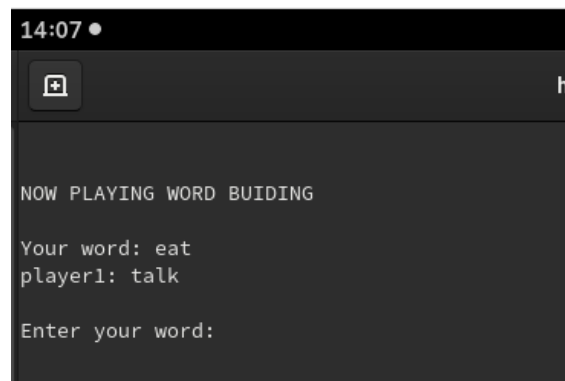


```
Activities Terminal ▾
[+] h

NOW PLAYING WORD BUIDING

Your word: apple
player2: eat

Enter your word: talk
█
```



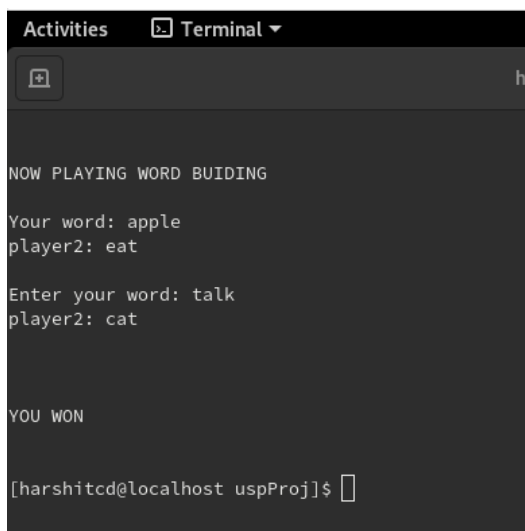
```
14:07 •
[+] h

NOW PLAYING WORD BUIDING

Your word: eat
player1: talk

Enter your word:
█
```

End of the game:



```
Activities Terminal ▾
[+] h

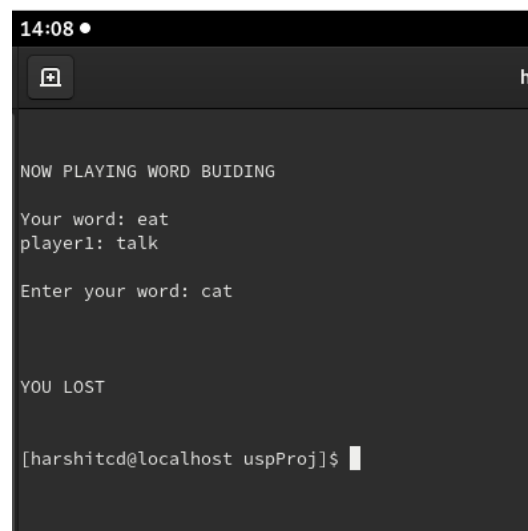
NOW PLAYING WORD BUIDING

Your word: apple
player2: eat

Enter your word: talk
player2: cat

YOU WON

[harshitcd@localhost uspProj]$ █
```



```
14:08 •
[+] h

NOW PLAYING WORD BUIDING

Your word: eat
player1: talk

Enter your word: cat

YOU LOST

[harshitcd@localhost uspProj]$ █
```



## Conclusions and Further Enhancements

- As we were able to see, the Named pipe or FIFO API is a very good way of inter-process communication.
  - Using this technique, we could build dual player games that work as 2 separate processes on 2 terminals.
  - These processes were connected via a shared memory i.e., the FIFO file.
- 
- Further enhancements would be to use this concept to implement the critical section problem using mutex locks and semaphores.
  - We could also look into building CHAT ROOMS that users can access across different devices connected through a network using enhanced versions of named pipes.

## References

- <https://www.geeksforgeeks.org/named-pipe-fifo-example-c-program/>
- [https://en.wikipedia.org/wiki/Named\\_pipe](https://en.wikipedia.org/wiki/Named_pipe)
- [https://www.tutorialspoint.com/inter\\_process\\_communication/inter\\_process\\_communication\\_named\\_pipes.htm](https://www.tutorialspoint.com/inter_process_communication/inter_process_communication_named_pipes.htm)
- Sumitabha Das - Your UNIX Linux, The Ultimate Guide (2012, McGraw-Hill Education)