NAMED PIPES

The concept of named pipes is almost same as anonymous pipes but the anonymous pipe come into existence on run time. Whereas named pipes are already created pipes which can be used by either of the pipes.

A pipe should have to be created first. It should be in the directory where the code is written for example reader and writer. How to distinguish whether anonymous pipes are required or not. For named pipes we must give the name of pipe as an argument. This means that question basically belongs to named pipes

On terminal we have to write *mkfifo pipe*

The command in italic will be creating a pipe in that directory

Functions to be used in writing to a pipe

- **write()**
- **open()**
- **sprintf()**
- **perror()**
- **close()**
- **sleep(1)**
- **#define BUFFERSIZE *PIPE_BUF* for this #include <limits.h>**

Now the header files can be get by using man commands

So now let's talk about the code.

1. **Firstly variables for the right end and data which must be written into the pipe.**
2. **Use open function to open a pipe for write purpose only(because we are just writing into the pipe) if open function returns -1 then perror() this automatically gets an error just like catch in java**
3. **Then we will try to write into the pipe, sir used while(1) so that we will not stop to write and write function will be used but what to write? Before we will use a sprint command. What it does? Basically it stores formatted value to a variable**
    a. **sprintf(var, "Hello guys welcome to the %d video of my %s playlist", I, name); where I is a variable and name is also a variable this sprintf() function returns a value and if it is -1 then give perror after the condition statement**
4. **afterwards outside the loop we will implement closure of pipe using close(fd)**

MUHAMMAD MIKAAL AMJAD

Let we write the code

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <unistd.h>
#include <limits.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/times.h>
#define BSIZE PIPE_BUF
int main(int argc , char *argv[]){
int res,charcount,pip;//pip for pipe identification, res will store the write function value
char buffer[BSIZE+1];
if ((pip=open(argv[1], O_WRONLY)==-1){//linking with pipe
perror("SORRY USTAD ME TERI HARAKATON KI WAJA SE PIPE NA BANA SAKA");
return 1;
}
for (int i=0;i<5;i++){
charcount=sprint(buffer,"This statement is written by the %d iteration of loop into the pipe",i)
if((res=write(pip,buffer, BSIZE))==-1){
perror("Ab kuch nai ho sakta");
}
sleep(1);
}
//at the end
close(pip);
return 0;
}
```

MUHAMMAD MIKAAL AMJAD

Now let come to read side

Just the change is that we will change from write to read, and we will use memset function which will basically clean the buffer

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <unistd.h>
#include <limits.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/times.h>
#define BSIZE PIPE_BUF
int main(int argc , char *argv[]){
int res,charcount,pip;//pip for pipe identification, res will store the write function value
char buffer[BSIZE+1];
memset(buffer, '\0', sizeof(buffer));
if ((pip=open(argv[1], O_WRONLY)==-1){//linking with pipe
perror("SORRY USTAD ME TERI HARAKATON KI WAJA SE PIPE NA BANA SAKA");
return 1;
}
while(1){
if((res=read(pip,buffer, BSIZE))==-1){
perror("Ab kuch nai ho sakta");
}
fprintf(stderr, "The written thing was%s" , buffer);
memset(buffer, '\0', sizeof(buffer));
sleep(1);
```

MUHAMMAD MIKAAL AMJAD

```
}
//at the end
close(pip);
return 0;
}
```

MUHAMMAD MIKAAL AMJAD