What is a named pipe?

A named pipe is a special file that is used to transfer data between unrelated processes. One (or more) processes write to it, while another process reads from it. Named pipes are visible in the file system and may be viewed with `ls' command like any other file. (Named pipes are also called fifos; this term stands for `First In, First Out'.)
Named pipes may be used to pass data between unrelated processes, while normal (unnamed) pipes can only connect parent/child processes (unless you try *very* hard). Named pipes are strictly unidirectional, even on systems where anonymous pipes are bidirectional (full-duplex).

How do I create a named pipe?

To create a named pipe interactively, you'll use `mkfifo`.

To make a named pipe within a C program use include the following C libraries:

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

and to create the named pipe(FIFO) use `mkfifo()`function:

```
if (mkfifo("test_fifo", 0777))
/* permission is for all */
{
    perror("mkfifo");
    exit(1);
}
```

How do I use a named pipe?

To use the pipe, you open it like a normal file, and use `read()` and `write()` just as though it was a plain pipe.
However, the `open()` of the pipe may block! The following rules apply:

- **If you open for both reading and writing (`O_RDWR`), then the open will not block.**
- **If you open for reading (`O_RDONLY`), the open will block until another process opens the FIFO for writing, unless `O_NONBLOCK` is specified, in which case the open succeeds.**
- **If you open for writing `O_WRONLY`, the open will block until another process opens the FIFO for reading, unless `O_NONBLOCK` is specified, in which case the open fails.**

```
open_mode = O_RDONLY or O_WRONLY, or O_NONBLOCK;
res = open("test_fifo", open_mode);
```

Here, open mode is one of the modes that was discussed above.

When reading and writing the FIFO, the same considerations apply as for regular pipes, i.e. `read()` and `write()`;

```
read(res, buffer, BUFFER_SIZE);
write(res, buffer,BUFFER_SIZE);
```

Here is a simple use of pipes, unlike the named pipes, we use two differenct files with out using `fork()`or `execl()`. Offcourse we can use fork and execl in our programs, but they are not necessary.

```
/* fifo1.c*/
/*
This program will read from a fifo and prints the received string to the screen.
*/
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <limits.h>

#define FIFO_NAME "my_fifo"                    /* the named pipe*/
#define BUFFER_SIZE PIPE_BUF
int main()
{
   int res;
   charbuffer[BUFFER_SIZE + 1];

   if (access(FIFO_NAME, F_OK) == -1) {                 /* check if fifo already
exists*/
      res = mkfifo(FIFO_NAME, 0777);            /* if not then, create the fifo*/
      if (res != 0) {
         fprintf(stderr, "Could not create fifo %s\n", FIFO_NAME);
         exit(EXIT_FAILURE);
      }
   }
   memset(buffer, '\0', BUFFER_SIZE + 1);              /* clear the string */
   printf("Process %d opening FIFO for reading\n", getpid());
   res = open(FIFO_NAME, O_RDONLY);                    /* open fifo in read-only
mode */
   read(res, buffer, BUFFER_SIZE);
   printf("Process %d received: %s\n", getpid(), buffer);
   sleep(5);
   if (res != -1) (void)close(res);                    /* close the fifo */
   printf("Process %d finished\n", getpid());          /* make sure you close fifo
after */
   exit(EXIT_SUCCESS);                                 /* using it.                */
}
```

```
/* fifo2.c */
/* This file will write to the fifo a string.*/
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <limits.h>

#define FIFO_NAME "my_fifo"                    /* the fifo name */
#define BUFFER_SIZE PIPE_BUF
int main()
{
   int res;
   char buffer[BUFFER_SIZE + 1];
   if (access(FIFO_NAME, F_OK) == -1) {              /* check if fifo already
exists*/
      res = mkfifo(FIFO_NAME, 0777);           /* if not then, create the fifo*/
      if (res != 0) {
         fprintf(stderr, "Could not create fifo %s\n", FIFO_NAME);
         exit(EXIT_FAILURE);
      }
   }
   printf("Process %d opening FIFO\n", getpid());
   res = open(FIFO_NAME, O_WRONLY);              /* open fifo in write only
mode */
   sprintf(buffer, "hello");                     /* string to be sent */
   write(res, buffer,BUFFER_SIZE);                    /* write the string to fifo */
   printf("Process %d result %d\n", getpid(), res);
   sleep(5);
   if (res != -1) (void)close(res);              /* close the fifo */
   printf("Process %d finished\n", getpid());
   exit(EXIT_SUCCESS);
}
```

result of the programs on the screen:

```
Process 23242 opening FIFO
Process 23242 result 3
Process 23243 opening FIFO for reading
Process 23243 received: hello
Process 23242 finished
Process 23243 finished
```