

# INTER-PROCESS COMMUNICATION AND SYNCHRONISATION:

## **Lesson-15: Pipe Device Functions**

# 1. Pipe Function

# Pipe

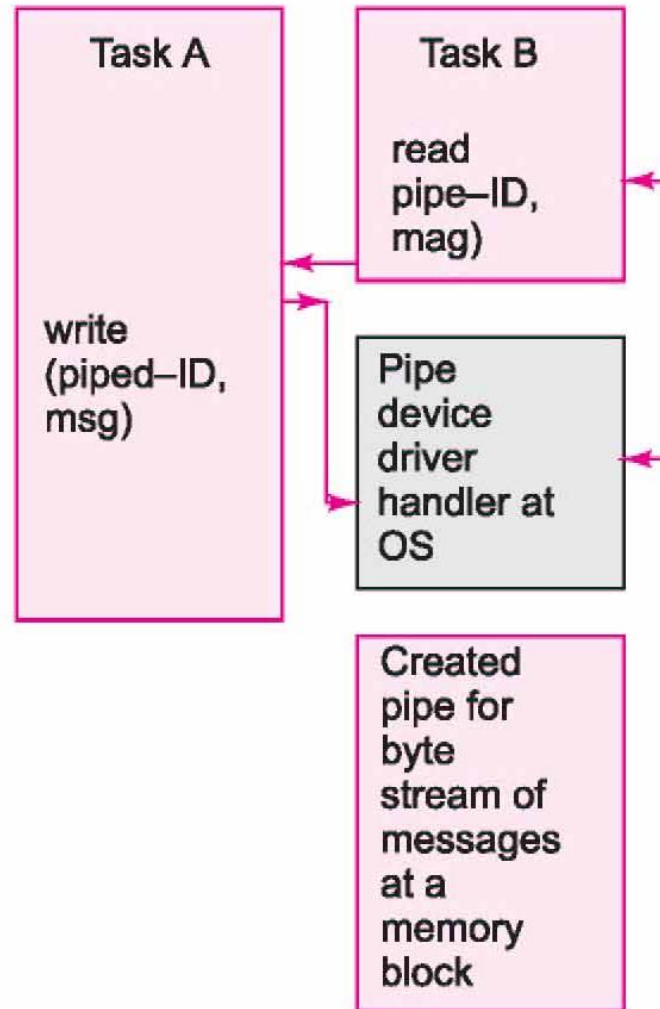
- Pipe is a device used for the inter process communication
- Pipe has the functions create, connect and delete and
- functions similar to a device driver (open, write, read, close)

# Writing and reading a Pipe

- A message-pipe— a device for inserting (writing) and deleting (reading) from that between two given inter-connected tasks or two sets of tasks.
- Writing and reading from a pipe is like using a C command *fwrite with a file name* to write into a named file, and C command *fread with a file name* to read into a named file.

# Pipe-device write and read using device driver

## Functions in a created pipe



# Write and read using Pipe

- 1. One task using the function `fwrite` in a set of tasks can write *to* a pipe at the back pointer address, `*pBACK`.
- 2. One task using the function `fread` in a set of tasks can read from a pipe at the front pointer address, `*pFRONT`

# Pipe as IO Stream

- Pipes are also like Java *PipedInputOutputStreams*. Java defines the classes for the input output streams

# Pipe Messages

- In a pipe there may be no fixed number of bytes per message with an initial pointer for the back and front and there may be a limiting final back pointer.
- A pipe can therefore be limited and have a variable number of bytes per message between the initial and final pointers.



# Unidirectional feature in Pipe

- Pipe is unidirectional. One thread or task inserts into it and other one deletes from it.

## 2. Pipe-device Functions

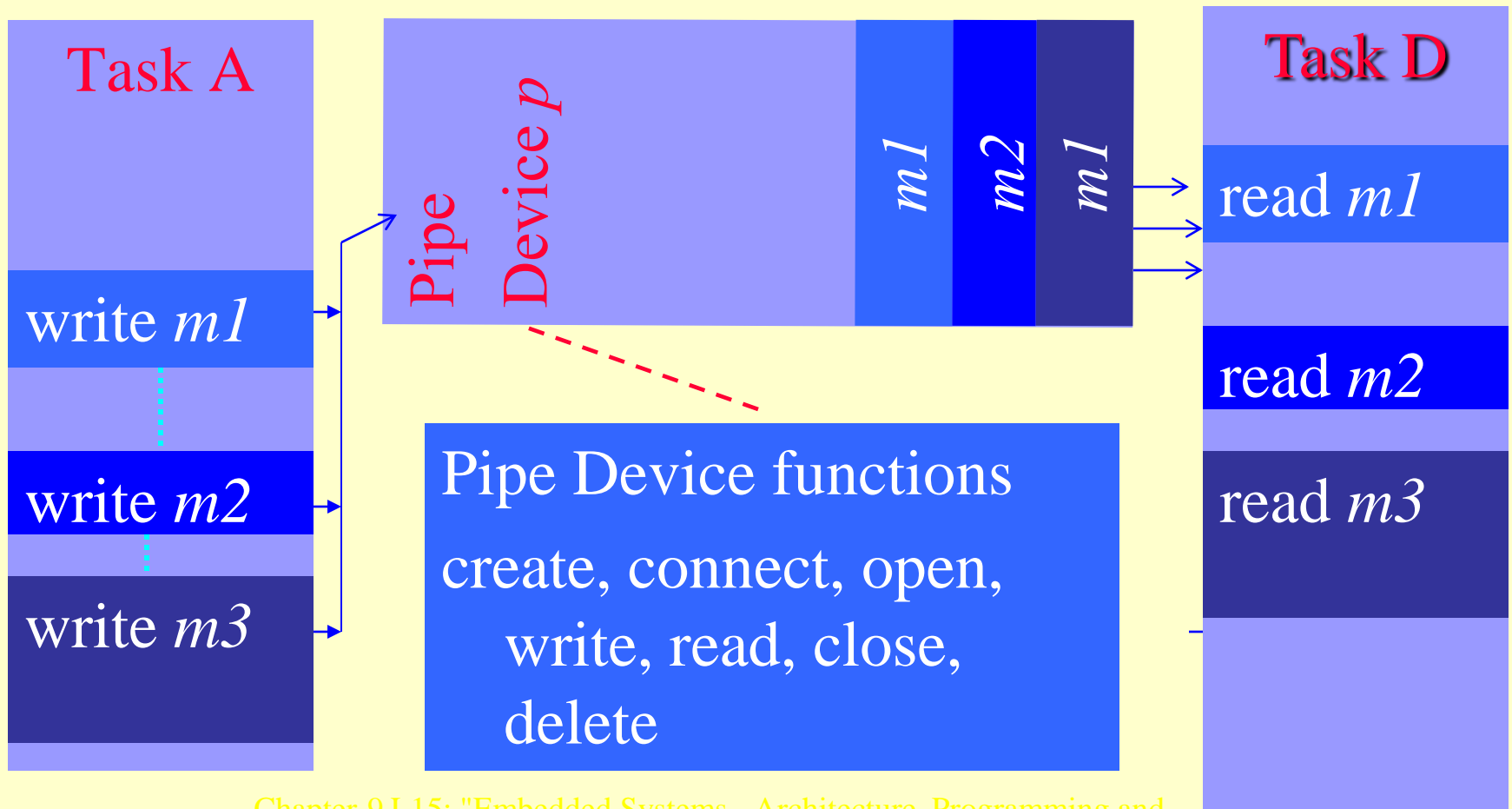
# Pipe-device Functions

1. pipeDevCreate — for creating a device
2. open ( ) — for opening the device to enable its use from beginning of its allocated buffer, its use with option and restrictions or permissions defined at the time of opening.
3. connect ( ) — for connecting a thread or task inserting bytes into the pipe to the thread or task deleting bytes from the pipe.
4. write ( ) — function for inserting (writing) into the pipe from the bottom of the empty memory space in the buffer allotted to it.

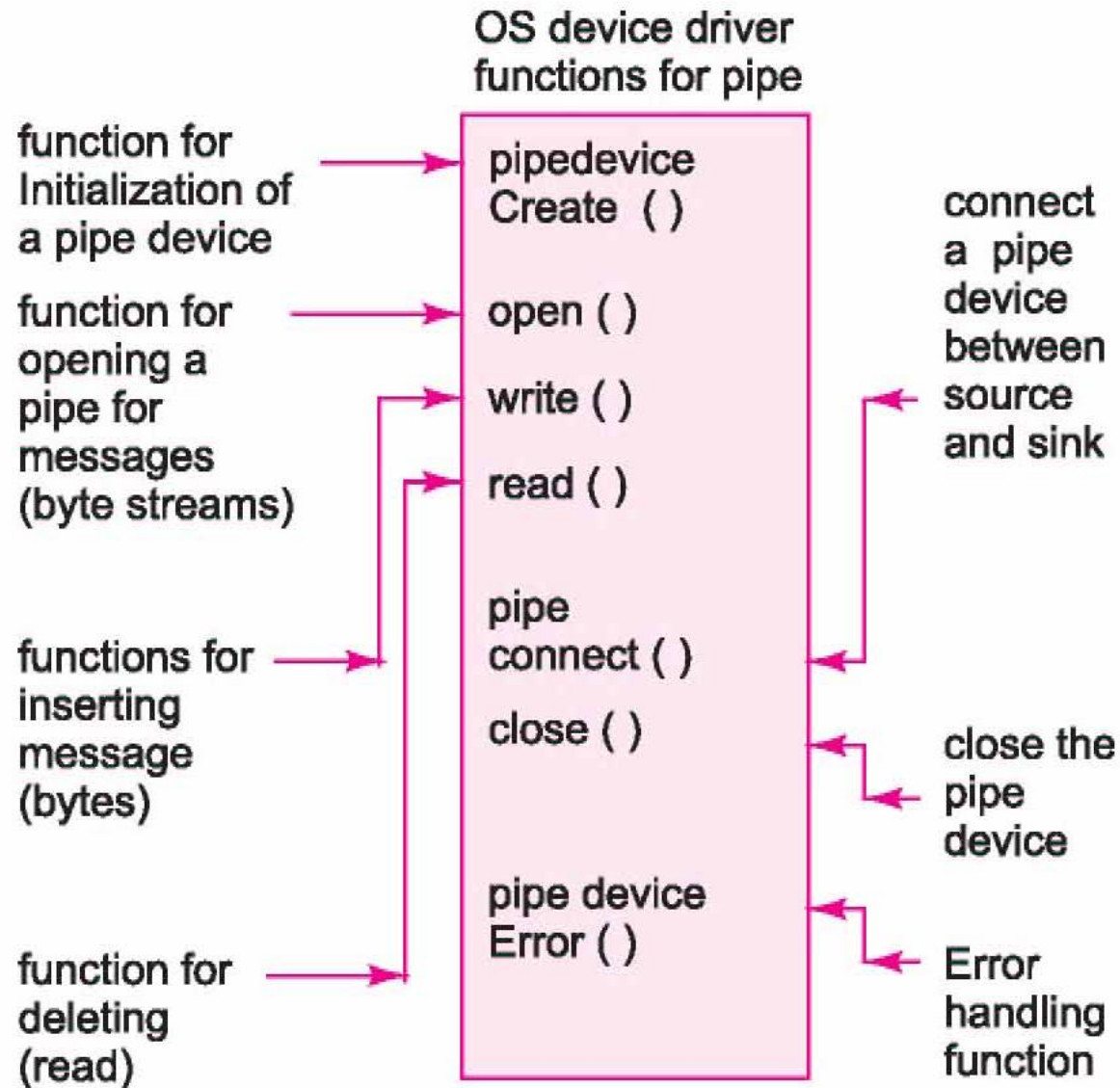
# Pipe-device Functions...

5. `read ( )`— function for deleting (reading) from the pipe from the bottom of the unread memory spaces in the buffer filled after writing into the pipe.
6. `close ( )` — for closing the device to enable its use from beginning of its allocated buffer only after opening it again..

# Task A sending messages into a pipe and task D receiving that

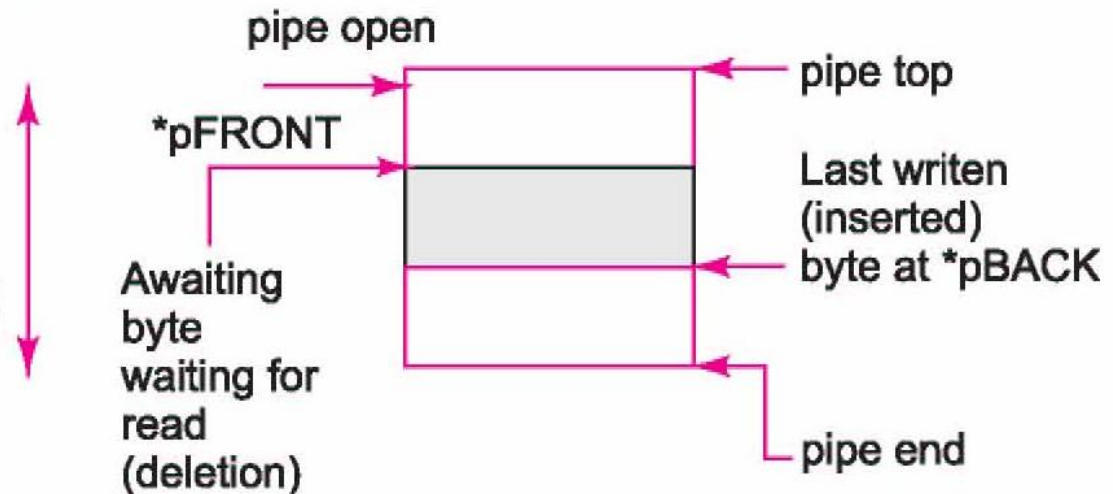


# Pipe-device Functions



# Pipe-messages in a message-block with its top pointed by \*pFRONT and end by \*pBACK

Messages (byte stream) at pipe device memory buffer for writing (insertion) and reading (deletion without wait) as FIFO (first-in first-out)



# 3. IPC Pipe device functions

## Application Example



# pipeCardInfo creation in Smart card

- *pipeDevCreate (“/pipe/pipeCardInfo”, 4, 32) /\* Create a pipe pipeCardInfo, which can save four messages each of 32 bytes maximum \*/*

# pipeCardInfo open in Smart card

- *fd = open (“/pipe/pipeCardInfo”, O\_WR, 0) /\* Open a write only device. First argument is pipe ID /pipe/pipeCardInfo, second argument is option O\_WR for write only and third argument is 0 for unrestricted permission.\*/ .*

# Task\_Send\_Card\_Info in Smart Card

```
static void Task_Send_Card_Info (void  
    *taskPointer) {
```

.

```
while (1) {
```

.

.

```
cardTransactionNum = 0; /* At start of the  
    transactions with the machine*/
```

# **Task\_Send\_Card\_Info sending into pipe**

```
write (fd, cardTransactionNum, 1) /* Write 1  
byte for transaction number after card  
insertion */
```

```
write (fd, cardFabricationkey, 16) /* Write 16  
bytes for fabrication key */
```

## **Task\_Send\_Card\_Info sending into pipe...**

```
write (fd, cardPersonalisationkey, 16) /* Write  
16 bytes for personalisation */
```

```
write (fd, cardPIN, 16) /* Write 16 bytes for  
PIN, personal identification number granted  
by the authoriser bank */
```

```
.
```

```
};
```

# Summary

# We learnt

## OS provides the IPC functions for pipe

- DevCreate, Open, connect, write, read and close functions.
- Limit of the total number of messages and maximum size per message can also be provided when creating a pipe

- Pipe is device for a stream of messages that connects the two tasks and the pipe functions are similar to the device functions and IO stream functions.



# End of Chapter 9 Lesson-15: Pipe Device Functions