INTER-PROCESS COMMUNICATION AND SYNCHRONISATION OF PROCESSES, THREADS and TASKS:

Lesson-16: Mailbox

1. IPC Mailbox functions

Queue and Mailbox

- Some OSes provide the mailbox and queue both IPC functions
- When the IPC functions for mailbox are not provided by an OS, then the OS employs queue for the same purpose.

Mailbox

- Mailbox (for message) is an IPC through a message-block at an OS that can be used only by a single destined task.

Mailbox …

- A task on an OS function call puts (means post and also send) into the mailbox only a pointer to a mailbox message
- Mailbox message may also include a header to identify the message-type specification.]

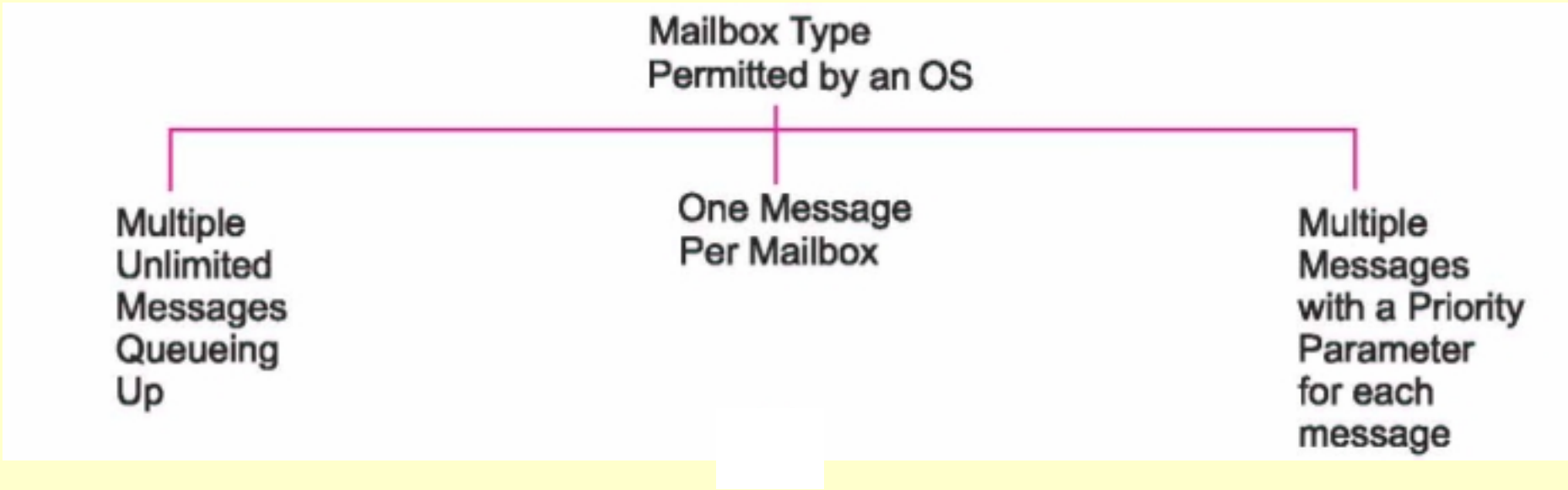
Mailbox IPC features

- OS provides for inserting and deleting message into the mailbox message-pointer. Deleting means message-pointer pointing to Null.
- Each mailbox for a message need initialization (creation) before using the functions in the scheduler for the message queue and message pointer pointing to Null.

Mailbox IPC features…

- There may be a provision for multiple mailboxes for the multiple types or destinations of messages. Each mailbox has an ID.
- Each mailbox usually has one message pointer only, which can point to message.

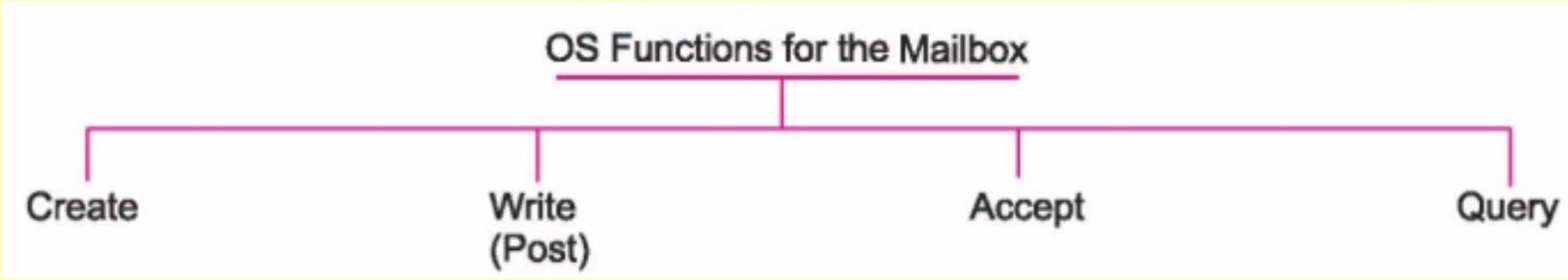
Mailbox Types



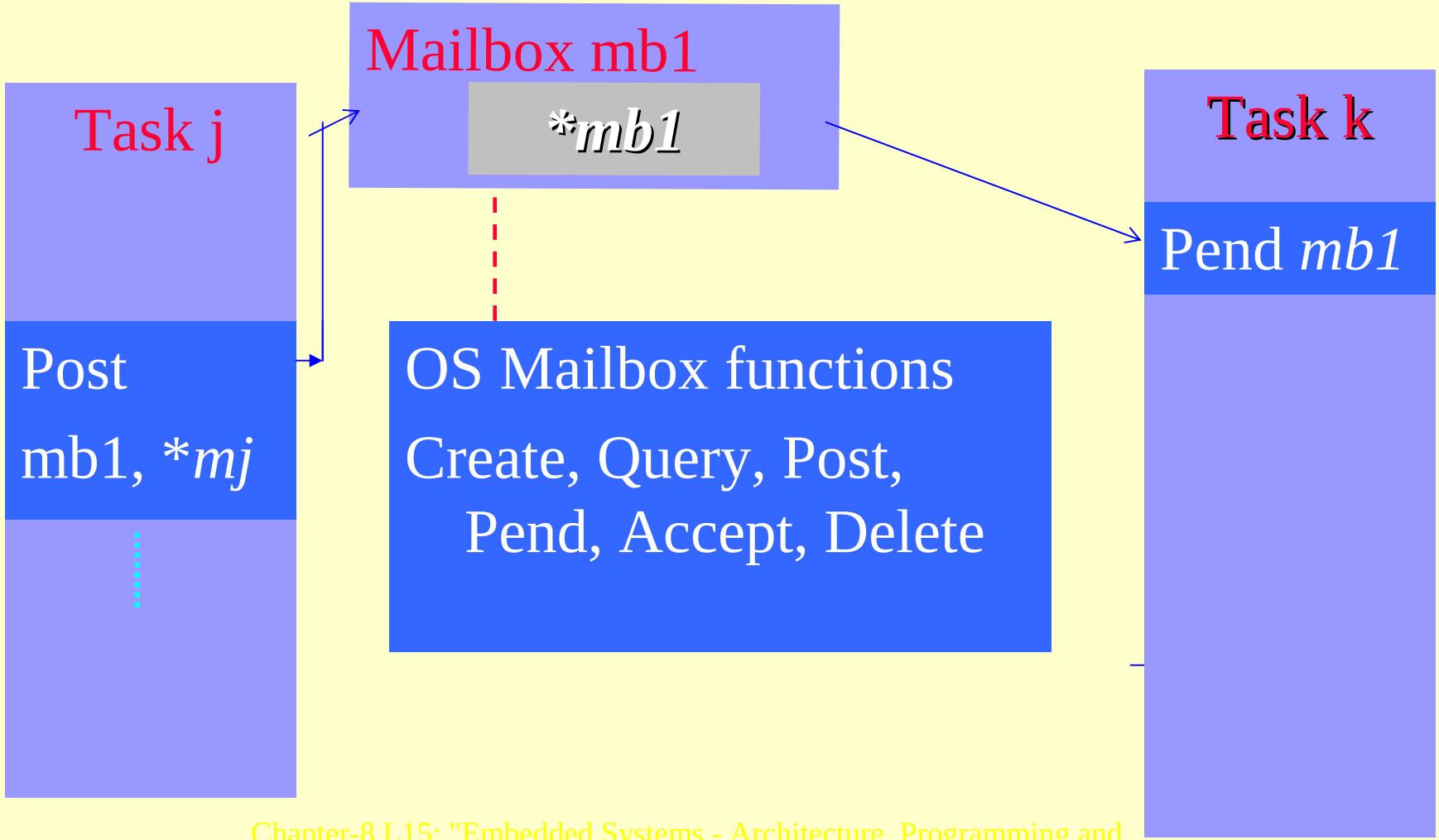
Mailbox IPC features …

- When an OS call is to post into the mailbox, the message bytes are as per the pointed number of bytes by the mailbox message pointer.

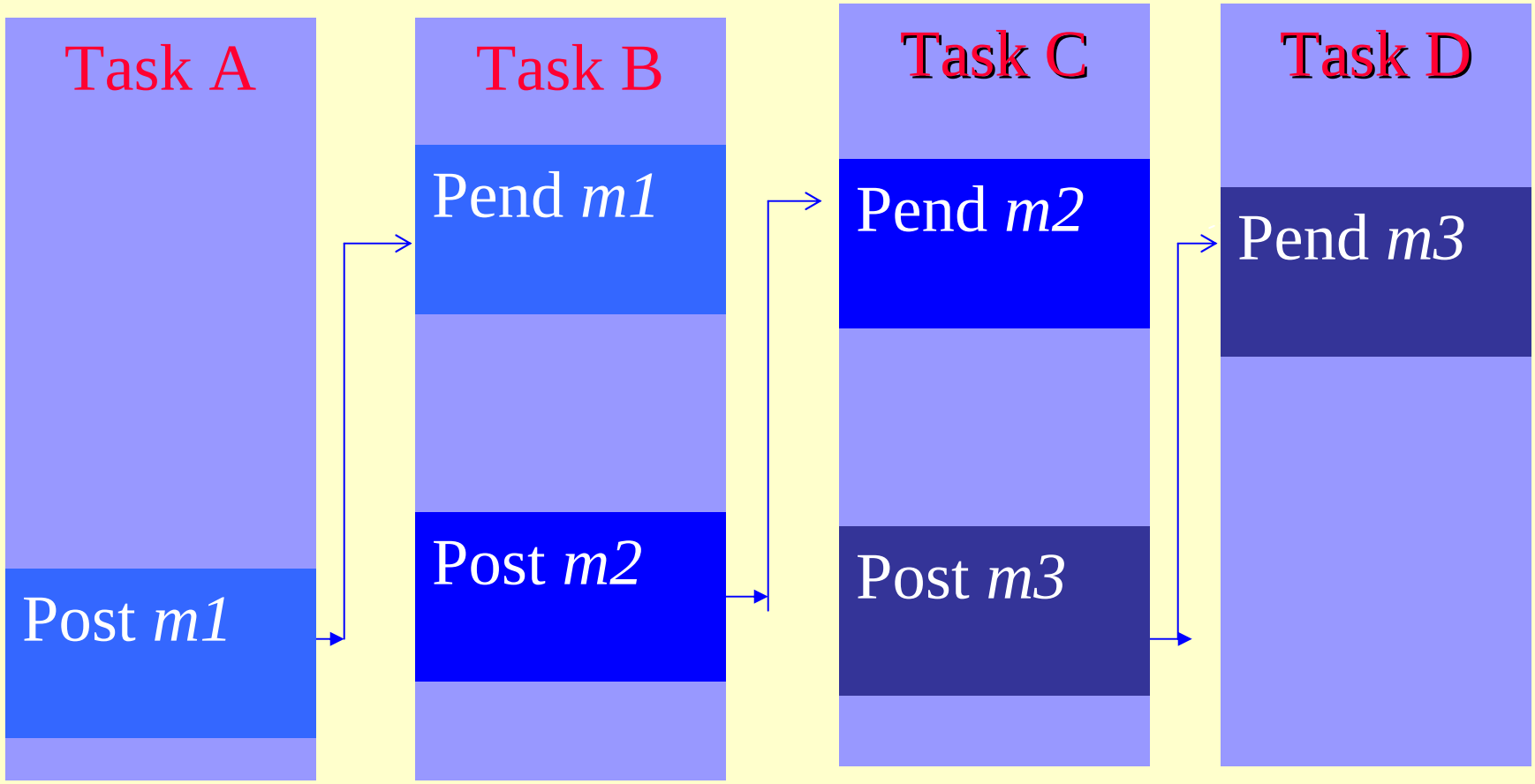
2. Mailbox Related Functions at the OS


OS Functions for the Mailbox
Create
Write
(Post)
Accept
Query

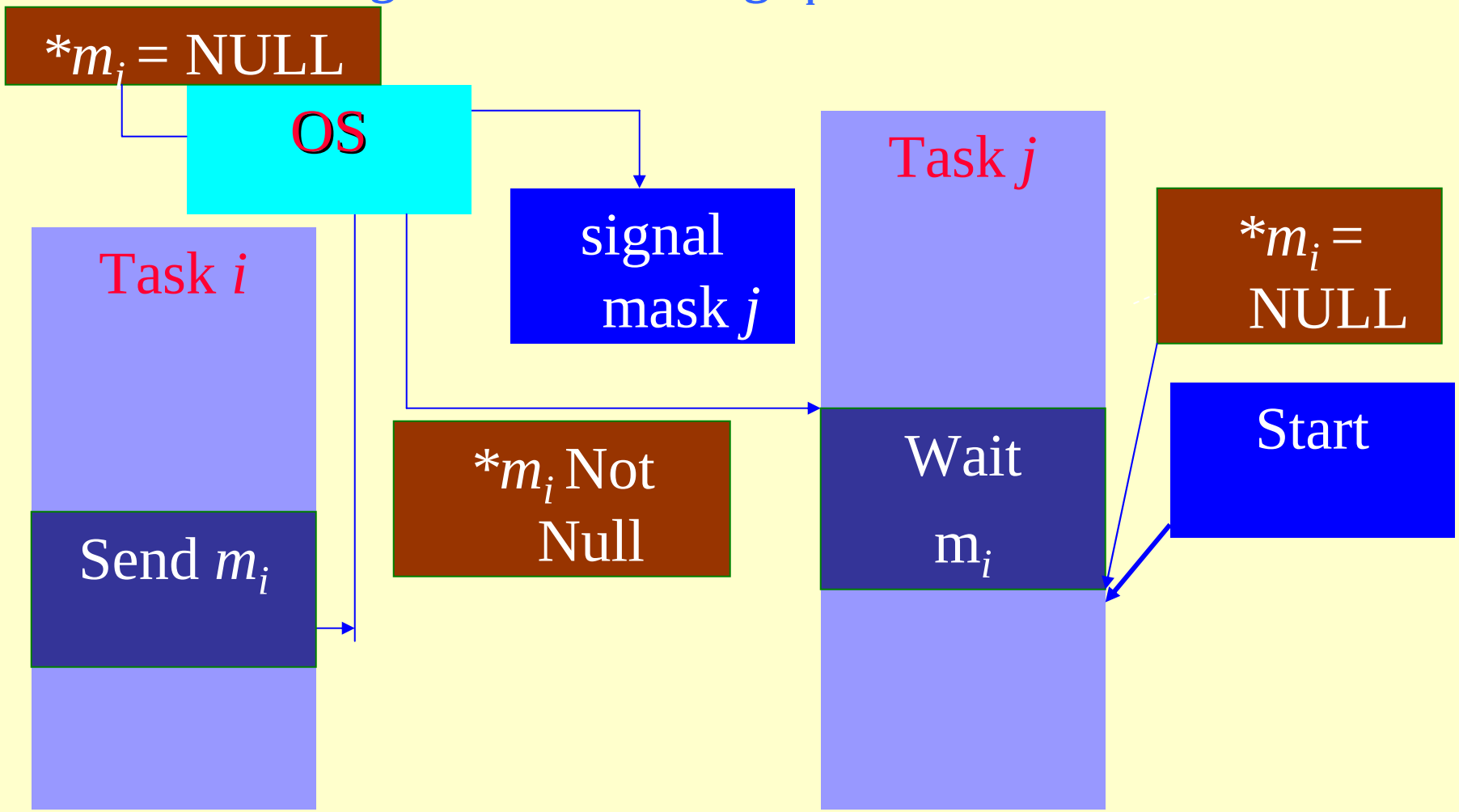
Tasks *j* sending a message-pointer into a mailbox and task *k* receiving that



Section of codes in the tasks B, C, and D waiting for messages *m*1, *m*2 and *m*3 into mailboxes



Task i sending message pointer *m_i to initiate a task section waiting to take a message $_i$ before it could run



Mailbox IPC functions

1. OSMBoxCreate creates a box and initializes the mailbox contents with a NULL pointer at *msg .
2. OSMBoxPost sends at *msg, which now does not point to Null.

- An ISR can also post into mailbox for a task

Mailbox IPC functions…

3. OSMBoxWait (Pend) waits for *msg not Null, which is read when not Null and again *msg points to Null.

- The time out and error handling function can be provided with Pend function argument.
- ISR not permitted to wait for message into mailbox. Only the task can wait

Mailbox IPC functions…

4. OSMBoxAccept reads the message at *msg after checking the presence yes or no [No wait.] Deletes (reads) the mailbox message when read and *msg again points to Null

- An ISR can also accept mailbox message for a task

5. OSMBoxQuery queries the mailbox *msg.

3. IPC Queue functions Application Example

Task_*j* sending an integer value m_j

```
int *m_j; A global variable integer pointer
...; ...; ...;
static void Task_j (void *taskPointer) {...
while (1) {...; ...; ...;
& m_i = 8; ...; ...; ...;
OSMboxPost (m_j);
/* after this instruction executes the next task
section can operate on the m_j */
 ...; ...; ...;}; }
```

Task_k waiting for the m_i

static void Task_k (void *taskPointer) {...
while (1) {...; ...; ...;
OSMboxPend (m_j); /* OSMboxPend waits for mailbox message m_i and when available m_i, reset *m_i = NULL and proceed to next statement */
...; ...; ...;};}

Task Read_Amount in ACVM

```
static void Task Read-Amount (void
    *taskPointer) {
.
while (1) {
.
/* Codes for reading the coins inserted into
    the machine */
```

Task Read_Amount posting amount information in ACVM

/* Codes for writing into the mailbox full amount message if cost of chocolate is received*/

OSMboxPost (mboxAmt, fullAmount) /* Post for the mailbox message and *fullAmount, which equaled null now equals fullAmount message pointer*/*

.

};

Chocolate delivery task in ACVM

*static void Chocolate delivery task (void *taskPointer)* {

.

while (1) {

.

Chocolate delivery task waiting for message in ACVM

/* IPC for requesting full amount message */

fullAmountMsg = OSMboxPend (mboxAmt, 20, *err) /* Wait for the mailbox mboxAmt message for 20 clock ticks and error if message not found. mboxAmt becomes null after message is read.

.

};

Task_*User_Keypad_Input* in ACVM

```
static void Task_User_Keypad_Input (void
   *taskPointer) {
.
while (1) {
.
/* Codes for reading keys pressed by the user
   before the enter key */
```

Task_*User_Keypad_Input* posting amount information in ACVM

/* Codes for writing into the mailbox */

OSMboxPost (mboxUser, *userInput*) /* Post for the mailbox message and *userInput, which equaled null now equals userInput message pointer*/

.

};

Task_Display in ACVM

```
static void Task_Display  (void *taskPointer)
   {
.
while (1) {
.
```

Task_Display waiting for a message in ACVM

/ IPC for waiting for User input message */*

*UserInputMsg = OSMboxPend (mboxUser, 20, *err) /* Wait for the mailbox mboxUser message for 20 clock ticks and error if message not found. mboxUser becomes null after message is read.*

.

/ Code for display of user Input */*

Task_Display waiting for another message in ACVM

TimeDateMsg = OSMboxPend (timeDate, 20, err) / Wait for the mailbox message timeDate.*

/ Code for display TimeDateMsg Time: hr:mm Date: month:date */*

.

};

Summary

We learnt

- OS provides the IPC functions
- Create, Post, PostFront, Pend, Accept, Flush and Query for using message at mailbox.
- The time out and error handling function can be provided with Pend function argument.

End of Lesson-16: Mailbox