CENG 448 — Real Time Operating Systems

Pipes

Dr. Larry D. Pyeatt

South Dakota School of Mines and Technology

Optional Kernel Objects

A good RTOS implements a very small kernel, and provides the deveoper with the option to add components as needed.

Tasks, semaphores, mutexes, and message queues (which we have not covered yet) are essential to an RTOS.

Many RTOS kernels provide additional objects such as

- pipes,
- event registers,
- signals,
- condition variables, and
- other services.

Today we will talk about **pipes.**

Pipes

Pipes were invented as part of the Unix operating system to provide **unstructured** data exchange.

Typically, a pipe is viewed as a **stream of bytes**, and they are usually implemented as part of the file system.

The function to create a pipe returns two descriptors (handles), one for reading and one for writing.

Can be implemented in various ways. In FreeRTOS, there is no “pipe” object, but the **stream buffer** object is essentially a primitive pipe.

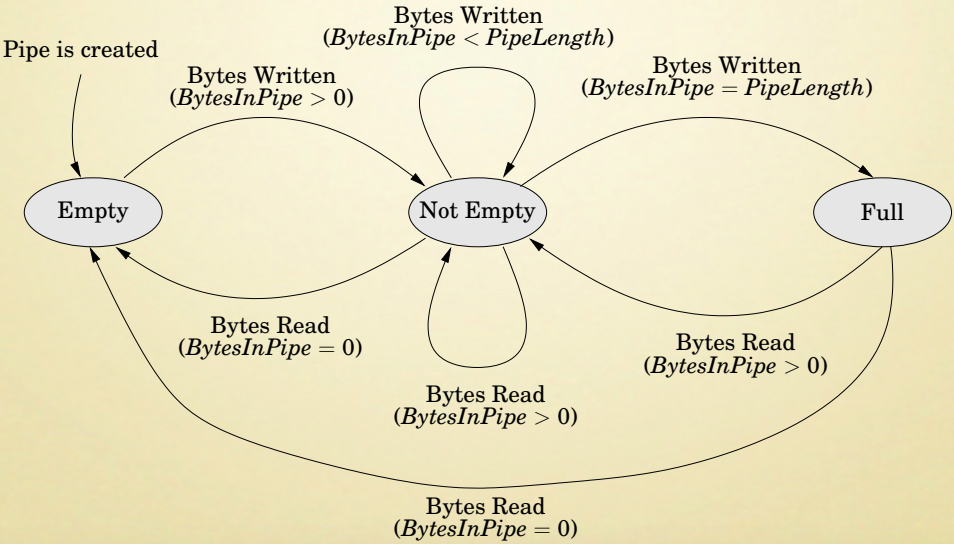
Pipes – Part 2

A pipe has two components:

- a pipe control block, and
- a data buffer.

The OS may provide *named pipes*, which appear as files in the file system. This allows any task to open, read, and write to the pipe as if it were a file.

Pipe States


Pipe is created
Bytes Written
(BytesInPipe < PipeLength)
Bytes Written
(BytesInPipe > 0)
Bytes Written
(BytesInPipe = PipeLength)
Empty
Not Empty
Full
Bytes Read
(BytesInPipe = 0)
Bytes Read
(BytesInPipe > 0)
Bytes Read
(BytesInPipe > 0)
Bytes Read
(BytesInPipe = 0)

Blocking

A task that tries to read from an empty pipe *may* be moved from the running state to the pipe read waiting list (the task becomes blocked for reading).

A task that tries to write to a full pipe *may* be moved from the running state to the pipe write waiting list (blocked for writing).

If the pipe is in the full state, and transitions to the not empty state, then the first task on the write waiting list is moved to the ready queue (unblocked).

If pipe is in the empty state, and transitions to the not empty state, then the first task on the read waiting list is moved to the ready queue (unblocked).

Pipe Operations

Typical pipe operations:

open	Opens or creates a pipe
close	Closes or destroys a pipe
read	read one or more bytes from a pipe
write	write one or more bytes to a pipe
fcntl	allows the pipe to be controlled in various ways, such as setting it for blocking/nonblocking operation, flushing the buffer, etc.
select	wait for specific conditions or wait on multiple pipes