TOPICS:

- The role of tyLib in a serial driver.
- Serial driver initialization
- Serial reads and write
- Serial driver interrupt handling
- Serial driver ioctl’s
Serial Drivers

Introduction

Writing Serial Drivers

TOPICS:

• What is tyLib
• Typical serial driver data flow.
• Typical serial driver control flow.
Writing a Serial Driver

- Use **tyLib** routines to manage data buffering and perform device independent character handling.
- Otherwise works like any other character driver.
- **tyLib** handles:
  - all **selectLib** handling
  - many standard ioctl functions
  - canonical processing

**NOTES:**
- **tyLib** actually uses two ring buffers, one for input (read) and one for output (write).
- **tyLib** functions like line discipline zero in UNIX.
- **tyLib** is a support library (not a driver).
Serial Driver Data Flow

Application

write() read()

tyWrite() tyRead()
tyLib

Output Ring Buffer Input Ring Buffer

tyITx() tyIRd()

Serial Driver

xxStartup() xxRxInt()

Device

NOTES:
NOTES:

- I/O System calls to **tyLib**
  - tyRead
  - tyWrite
- I/O System calls to directly to driver
  - xxOpen
  - xxIoctl
- **tyLib** calls to driver
  - xxStartup
- Driver calls to **tyLib**
  - tyDevInit
  - tyIoctl
  - tyIRd
  - tyITx
First member of driver’s device structure must be TY_DEV:

```c
typedef struct
{
    TY_DEV tyDev;
    ...
} FOO_DEV;
```

**NOTES:**

- The first member of the TY_DEV structure is a DEV_HDR structure.
- In `tyLib.h`, a TY_DEV_ID is typedef’ed as a pointer to a TY_DEV structure.
Serial Drivers

Introduction

Writing Serial Drivers

TOPICS:

• Initializing tyLib
• Handling device independent ioctl’s with tyIoctl()
• Transferring data to and from the input and output ring buffers managed by tyLib
Serial Driver Initialization

- `xxDrv`
  - Install driver -- `iosDrvInstall()`
- `xxDevCreate`
  - Call `tyDevInit()`
  - Initialize any channel specifics
  - Add device to system device list -- `iosDevAdd()`

**NOTES:**

- Typically, the serial drivers installs the following routines into the system driver table:

  ```c
  iosDrvInstall (xxOpen, NULL, xxOpen, NULL, tyRead, 
  tyWrite, xxIoctl);
  ```

  Note that the read and write routines make calls directly to `tyLib`. 
tyLib Initialization

STATUS tyDevInit (pTyDev, rdBufSize, wrtBufSize, txStartup)

- pTyDev Pointer to a TY_DEV structure
- rdBufSize Size in bytes of the input ring buffer
- wrtBufSize Size in bytes of the output ring buffer
- txStartup Address of the driver’s startup routine

- Called by xxDevCreate() to initialize tyLib.

NOTES:

- Creates the input and output ring buffers.
- Creates semaphores for mutual exclusion, read synchronization, and write synchronization.
- Initializes the select wakeup list.
- Flushes input and output.
Serial Driver ioctl’s

- Driver must:
  - Implement the FIOBAUDRATE command.
  - Handle any driver specific commands.
  - Unrecognized commands should be passed to `tyIoctl()` rather than failed.

```
NOTES:
• Example:
  1 int xxIoctl (int devId, command, arg)
  2 {
  3     int status;
  4  
  5     switch (command)
  6     {
  7         case FIOBAUDRATE:
  8             ...  
  9             break;
 10         /* driver specific commands might go here */
 11         default:
 12             status = tyIoctl (devId, command, arg);
 13             break;
 14     }
 15     return (status);
 16 }
```
## ioctl Commands Managed By tyLib

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIONREAD</td>
<td>Returns the number of characters in the input (read) buffer</td>
</tr>
<tr>
<td>FIONWRITE</td>
<td>Returns the number of characters in the output (write) buffer</td>
</tr>
<tr>
<td>FIOFLUSH</td>
<td>Discards all characters in both buffers</td>
</tr>
<tr>
<td>FLOWFLUSH</td>
<td>Discards all characters in output buffer</td>
</tr>
<tr>
<td>FIORFLUSH</td>
<td>Discards all characters in the input buffer</td>
</tr>
</tbody>
</table>
ioctl Commands (cont’d)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIOSETOPTIONS</td>
<td>Set device options</td>
</tr>
<tr>
<td>FIOGETOPTIONS</td>
<td>Returns current device options</td>
</tr>
<tr>
<td>FIOCANCEL</td>
<td>Cancels read or write</td>
</tr>
<tr>
<td>FIOISATTY</td>
<td>Returns TRUE</td>
</tr>
<tr>
<td>FIOSELECT</td>
<td>Add to the select list</td>
</tr>
<tr>
<td>FIOUNSELECT</td>
<td>Remove from the select list</td>
</tr>
</tbody>
</table>

**NOTES:**

- If `tyIoctl()` receives an unknown command, it sets `errno` to `S_ioLib_UNKNOWN_REQUEST` and returns `ERROR`. 

Wind River Systems
Driver Startup Routine

void xxStartup (pSerialDev)

• Called by tyLib to output character to device and initiate a transmit cycle.

• Typically, this driver routine:
  • calls tyITx() to fetch a character from the output ring buffer
  • writes it to the device
  • turns on transmitter interrupts
• May be called at interrupt time by tyIRd().

NOTES:

• Called from tyLib to initiate a data transfer:
  • when data is available to be written
  • to echo characters at interrupt time
  • to handle Xon/Xoff
Fetching Characters from the Output Ring Buffer

STATUS tyITx (pTyDev, pChar)

pTyDev Pointer to device structure
pChar Address to place character fetched

• Returns OK if character available, else ERROR.

• Example xxStartup( ) code:
  
  if (tyITx (pTyDev, &outChar) == OK)
    *deviceDataPort = outChar;
    *controlRegister |= TURN_ON_TRANSMITTER_INTS;

NOTES:

• The tyITx( ) routine may be called at interrupt time.

• The tyITx( ) routine:
  • Unblocks tasks sleeping on a write select()
  • Maps newlines to CR/LF
  • Unblocks tasks sleeping on write()
Serial Write Routine

• Typically, driver installs `tyWrite()` routine in driver table.

• `tyWrite()`:
  • Blocks if ring buffer is full
  • Copies data from user buffer to ring buffer
  • If not undergoing transmit cycle, calls driver startup routine

NOTES:
Serial Read Routine

- Typically, driver just installs `tyRead()` routine in driver table.

- `tyRead()`:
  - Blocks if ring buffer is empty
  - Copies bytes from ring buffer into user buffer
  - Handles Xon/Xoff processing
  - If there are still characters in the input ring buffer, unblocks any task sleeping on it

**NOTES:**

- A driver may supply its own read and write routine and still call `tyRead()` or `tyWrite()`.
Serial Driver ISR

- If interrupt caused by device ready to transmit:
  - Fetch character from output ring buffer by calling \textit{tyITx}( )
  - Write character to device
  - If no more characters available, reset transmit interrupts

- If interrupt caused by data available:
  - Fetch character from device
  - Pass character to \textit{tyIRd}( )
Adding Characters to the Input Ring Buffer

STATUS tyIRd (pTyDev, char)

pTyDev Pointer to device structure
char Character to add to the input ring buffer

• The `tyIRd()` routine
  • Adds character to input buffer
  • Handles canonical processing
  • Releases tasks blocked on empty buffer or blocked on select
  • Fails if input buffer is full.

NOTES:
  • The `tyIRd()` routine may be called at interrupt time.
  • The `tyIRd()` routine is normally called by the serial driver’s receive ISR.
## Other tyLib Routines

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tyBackSpaceSet</td>
<td>Changes the backspace char (^H)</td>
</tr>
<tr>
<td>tyAbortSet</td>
<td>Changes the shell restart char (^C)</td>
</tr>
<tr>
<td>tyDeleteLineSet</td>
<td>Changes the line-delete char (^U)</td>
</tr>
<tr>
<td>tyEOFSet</td>
<td>Changes the end-of-file char (^D)</td>
</tr>
<tr>
<td>tyMonitorTrapSet</td>
<td>Changes the reboot char (^X)</td>
</tr>
</tbody>
</table>

### NOTES:
- These are globals (not defined on a per file descriptor basis).
## Summary

<table>
<thead>
<tr>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tyDevInit()</code></td>
<td>Called by <code>xxDevCreate()</code> to initialize <code>tyLib</code></td>
</tr>
<tr>
<td><code>tyIRd()</code></td>
<td>Called by receive ISR to add char to input buffer</td>
</tr>
<tr>
<td><code>tyITx()</code></td>
<td>Called by transmitter ISR to fetch char from output buffer</td>
</tr>
<tr>
<td><code>xxStartup()</code></td>
<td>Called by <code>tyLib</code> to initiate a transmit cycle</td>
</tr>
<tr>
<td><code>tyRead()</code></td>
<td>Called by I/O system to copy data from input ring buffer to user buffer</td>
</tr>
<tr>
<td><code>tyWrite()</code></td>
<td>Called by I/O system to copy data from user buffer to input ring buffer</td>
</tr>
<tr>
<td><code>tyIoctl()</code></td>
<td>Called by <code>xxIoctl()</code> to implement device independent commands</td>
</tr>
</tbody>
</table>