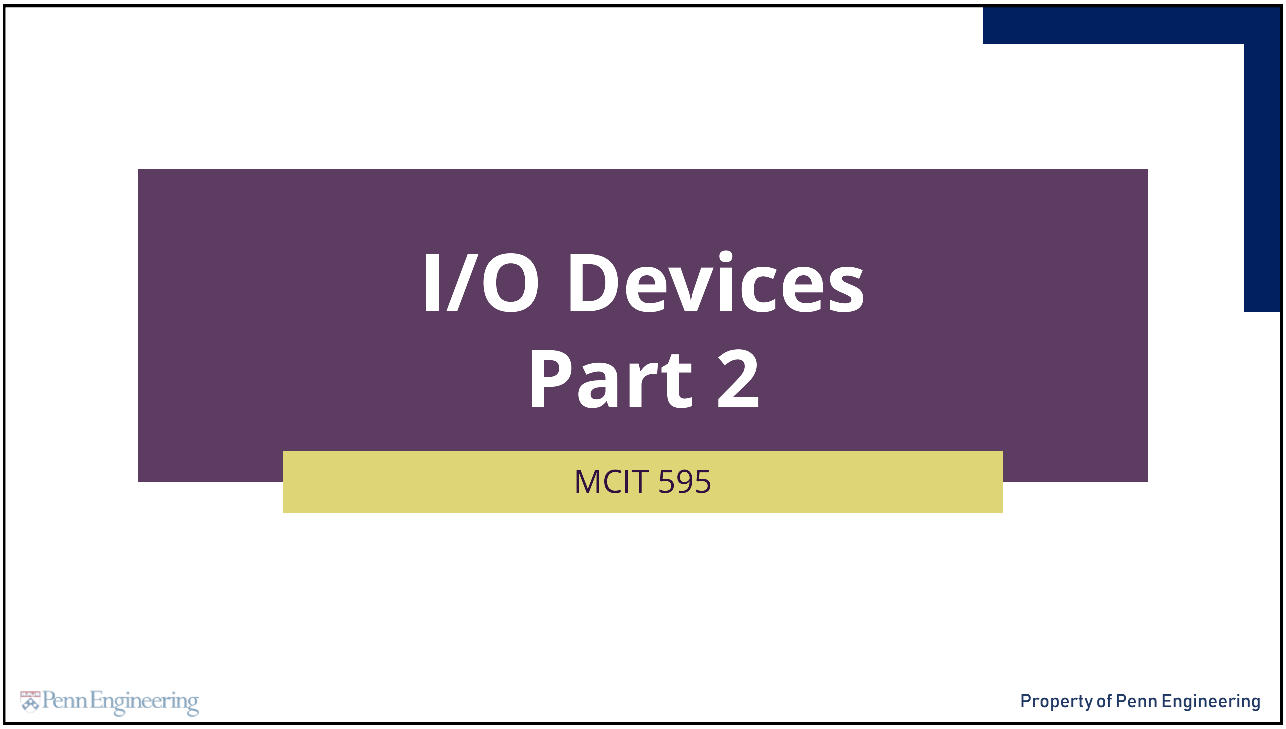
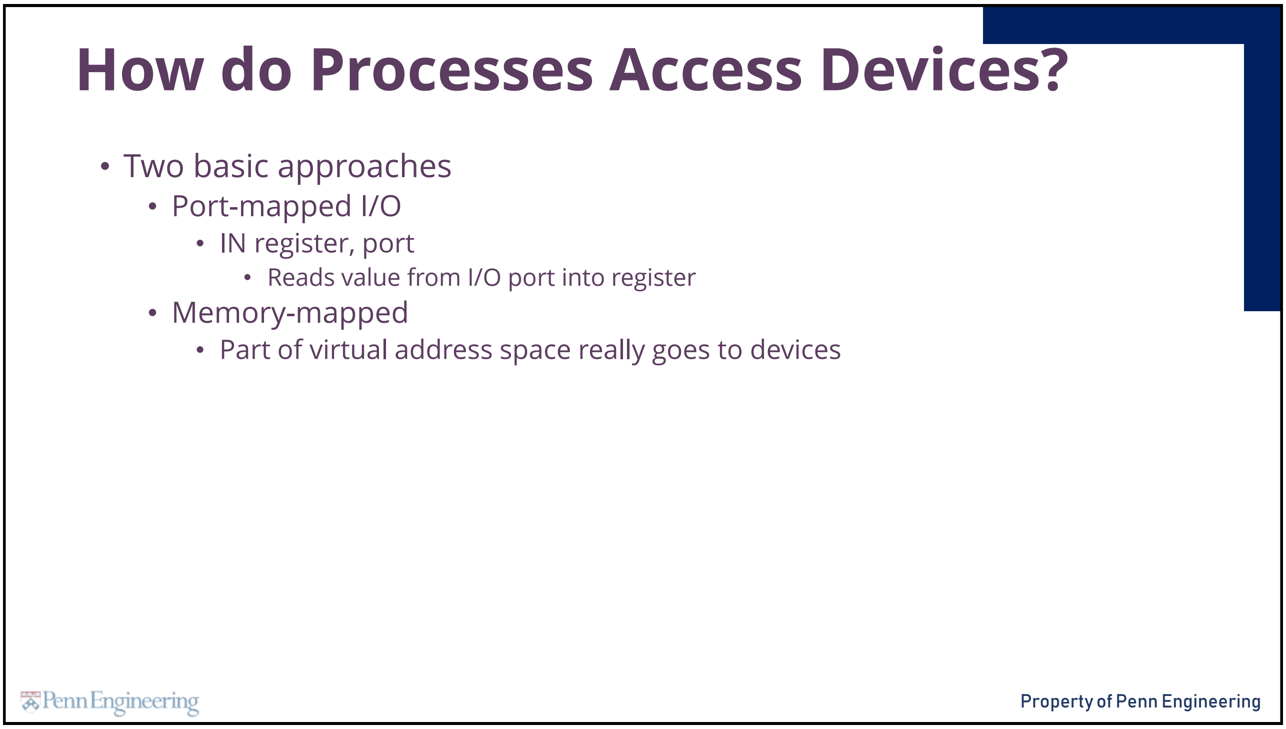


I/O Devices Part 2

MCIT 595



How do Processes Access Devices?

- Two basic approaches
  - Port-mapped I/O
    - IN register, port
      - Reads value from I/O port into register
  - Memory-mapped
    - Part of virtual address space really goes to devices

Memory Mapped I/O

- In a machine instruction “`MOV R, L`”, L can refer to:
  - A location in main memory, or
  - A control register, or address in data buffer, of a specific I/O device
- Memory Mapped I/O:
  - I/O space viewed as part of virtual memory
  - Directly access control registers or data buffers by reading from memory

Memory Mapped I/O

- Advantages
  - Device drivers can be written in high level languages like C
  - Protection managed by pages accessible in each user space
- Disadvantages
  - Caching must be disabled for I/O pages (write-through to devices)
  - MMU need to distinguish I/O requests from legitimate memory accesses

Naïve Busy Wait Device Driver

- Suppose we want to send N byte buffer to some device (say, the printer)
- Device has two registers (mapped as memory locations)
  - *Status*, which we read, to learn if device is ready
  - *Data*, which we write, to send byte to device
- The driver busy wait on *status:*

```
for (i=0; i< N; i++) {
    /* wait for device to be free */
    while (*Status != READY)
        ;
    *Data = buffer[i];
}
```

Device Driver with Interrupts

- Driver and interrupt handler
- Driver copies a single byte into buffer and schedules interrupt:

```
while (*Status != ready)
    ;
*Data = buffer[0]; /* 1 byte */
InvokeScheduler();
```

- Interrupt handler wakes up and reads data:

```
if (N == 0) Wakeup(); /* unblock */
else {
    *Data = buffer[i];
    N--; i++;
}
ResetInterrupt(); /*Notify device (ack) */
```