LSN 16 – Traditional Linux

- **Poor pre-emptiveness for kernel applications**
  - Originally a non-preemptive kernel (kernel locking)
  - Finer grain kernel locking provided in 2.5 Kernel

- **Uses fair-share time-slicing scheduler**
  - The Linux scheduling algorithm works by dividing the CPU time into *epochs*
  - Each process is provided a time quantum
LSN 16 – Linux Scheduling Example

- Develop a Linux application to generate a square wave using the PC parallel port (LPT1)
  - Load `parallel_oscope.c`:

```c
#include <sys/io.h>
#include <stdlib.h>
#define ON 100000
#define OFF ON // OFF set equal to ON

delay( unsigned int i )
{
    while(i--); // loop to decrement i until false (0)
}

int main( void )
{
    iopl(3); // change I/O privilege level of current process
    while(1) // send 50% DC pulse train to parallel port
    {
        outb(0xff, 0x378); // send all 1s to parallel port
        delay( ON ); // wait on time
        outb(0x00, 0x378); // send all 0s to parallel port
        delay( OFF ); // wait off time
    }
    return 0;
}
```
LSN 16 – Linux Scheduling Example

- Compile `parallel_oscope.c`
  - Requires the use of the `–O2` switch for compilation
  - Requires user to be logged in as root or super user

  ```
gcc –O2 –o parallel_oscope parallel_oscope.c
  ```

- Execute program from console window
  - Run program
  - Monitor the D0 – D7 lines on the parallel port with respect to the GND pin using an oscilloscope

- In a separate console window, execute the `yes` command
  - Monitor the results on the scope
LSN 16 – Homework

• Assignment
  – HMWRK8.pdf