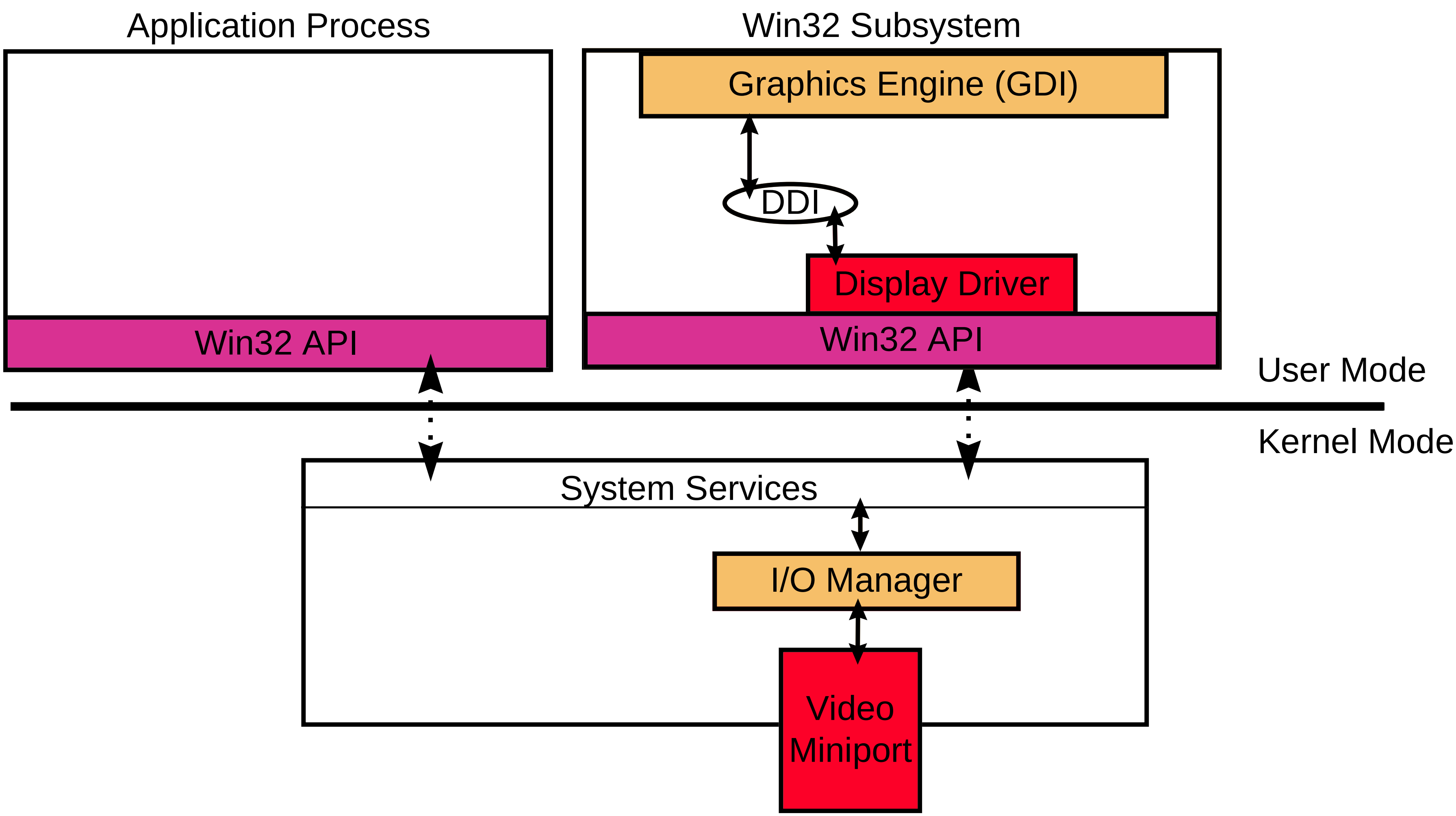
Announcements

- **No class on Thursday**
- Final is May 20, 1996 1:30-3:30 PM
  - in Chemistry Room 115 (same room as lecture)
- Reading: none
- Deadline for midterm #2 re-grades
  - Friday (May 10) at 5:00 P.M.
- Final review
  - Sat May 18 (4:00-6:00)
  - room CLB 104
- Project #5
  - due Monday May 13 (by 5:00 PM)
  - **turn in to Charles in room 1109 (4:30 to 5:30)**

Windows (NT 3.51)

- Kernel exports a mapped device for video
- User Process (Win32) provides
  - screen protection
    - each process has a message queue for its events
  - Win32 API Windows services
    - dialog boxes
    - graphics primitives
  - Programs using API must be on the same machine

NT (3.5) Display Drivers



My Research Interests

- Parallel Computing
  - There are limits to how fast one processor can run
  - solution: use more than one processor
- Issues in parallel computing design
  - do the processors share memory?
    - is the memory “uniform”?
    - how do processors cache memory?
  - if not how do they communicate?
    - message passing
    - what is the latency of message passing

Parallel Processing

- What happens in parallel?
- Several different processing steps
  - pipeline
  - simple example: grep foo | sort > out
  - called: *multiple instruction multiple data* (MIMD)
- The same operation
  - every processor runs the same instruction (or no-instruction)
  - called: *single instruction multiple data* (SIMD)
  - good for image processing
- The same program
  - every processor runs the same program, but not “lock step”
  - called: *single program multiple data* (SPMD)
  - most common model

Issues in effective Parallel Computation

- Load balancing
  - every processor should to have some work to do.
- Latency hiding/avoidance
  - getting data from other processors (or other disks) is slow
  - need to either:
    - hide the latency
      - processes can “pre-fetch” data before they need it
      - block and do something else while waiting
    - avoid the latency
      - use local memory (or cache)
      - use local disk (of file buffer cache)
- Limit communication bandwidth
  - use local data
  - use “near” data (i.e. neighbors)

My Research:

- Given a parallel program and a machine
- Try to answer performance related questions
  - Why is the programming running so slowly?
  - How do I fix it?
- Issues:
  - how to measure a program without changing it?
  - how do you find (and then present) the performance problem, not tons of statistics?
- Techniques:
  - dynamic data collection
  - automated search
  - analysis of process interactions

My Research (I/O):

- Given lots of data to access, and lots of disks
- How do you make effective use of these disks?
- Questions:
  - What should I/O look like?
    - virtual memory
    - file pointer based I/O
    - direct I/O
  - Where should the data be placed?
    - central servers vs. distributed to each node
    - how do improve data locality
  - What information can the application provide?
    - hints about future access patterns?
    - what data is going to be re-used?