## Operating Systems

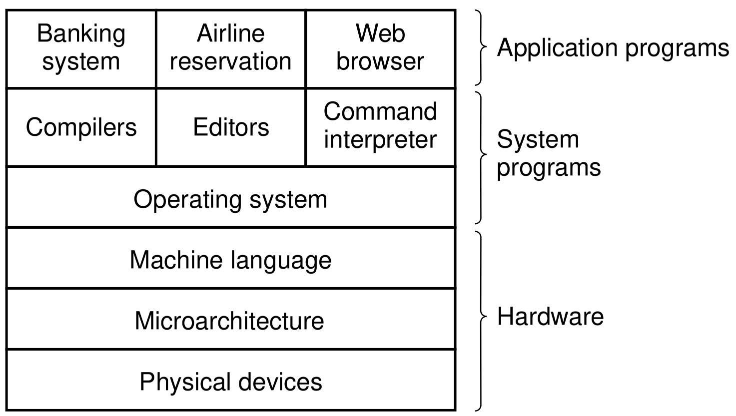
### Purpose

- Standardized basis for utilizing the computer HW-resources

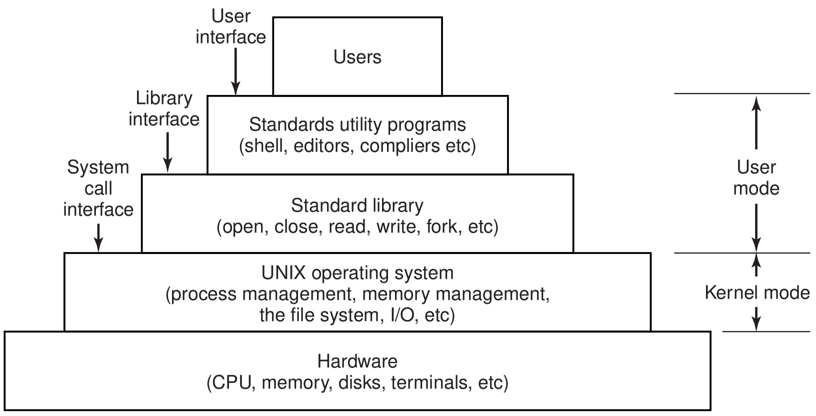
### Functionalities

- Management of activities (processes, threads, IPC)
- Memory and file management
- Input/output: User interface, networking, special devices
- Security
- System administration and control

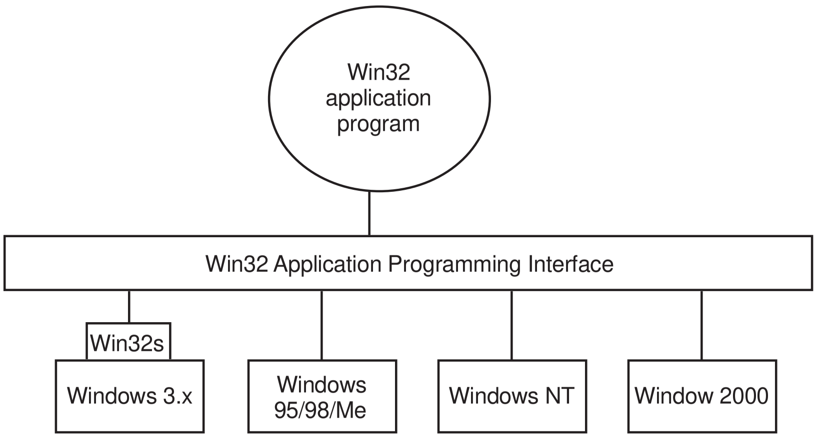
## System Layers



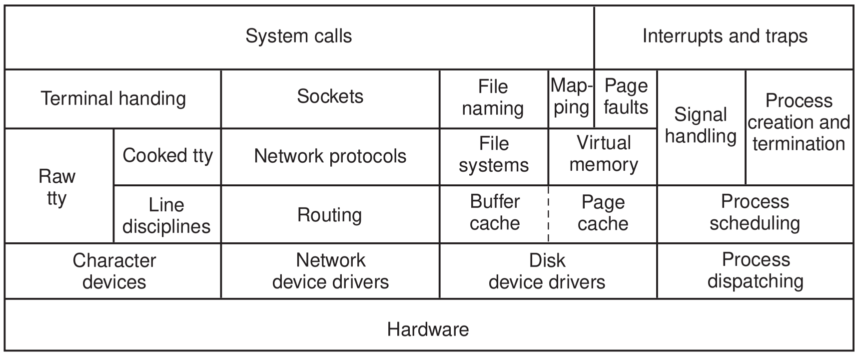
## System Layers (Unix)



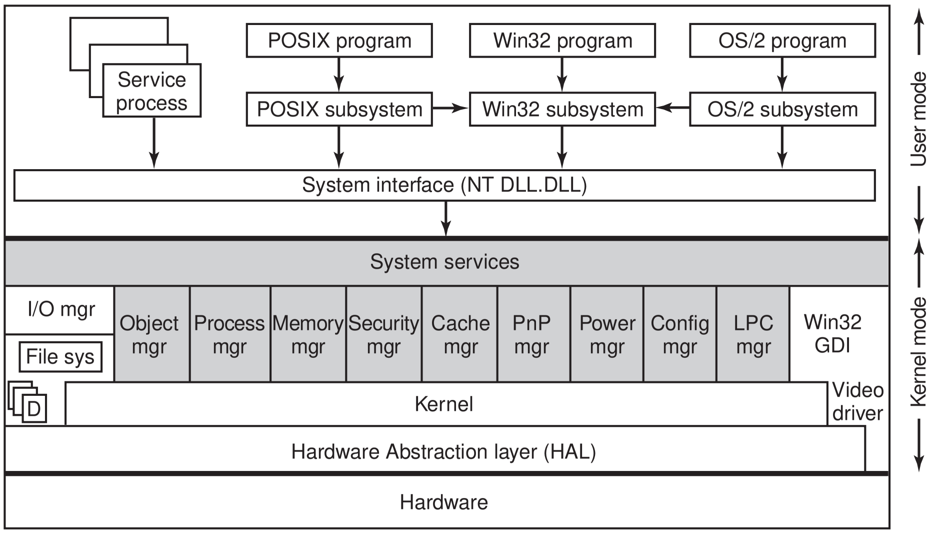

## Role of API (Windows)



## Unix-like systems




## Windows NT/2000/XP/ ...



## Operating System Topics

- Process management
- Memory management
- Synchronization and communication (IPC)
- I/O
- Deadlocks
- File Systems
- Networking
- Security
- Real-time and multimedia
- Distribution

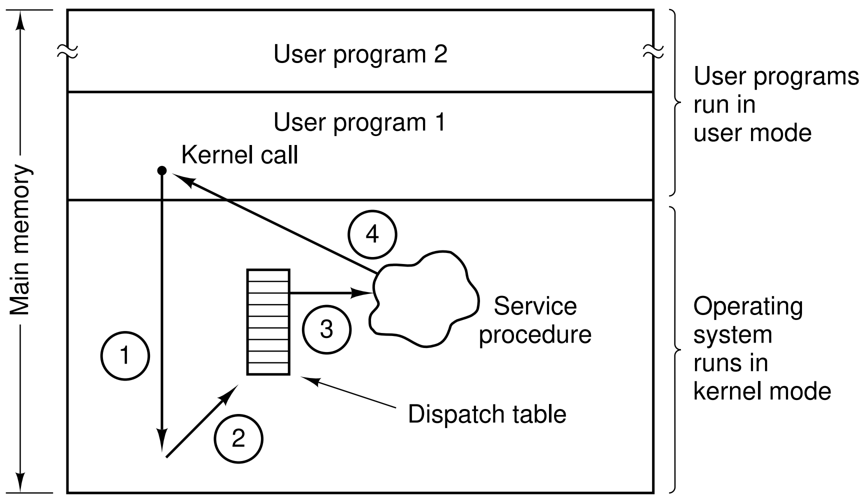
## Process Management

- Provide multiple, *concurrent activities*
- Ensure proper *progress* of activities
- Ensure *protection* between unrelated activities

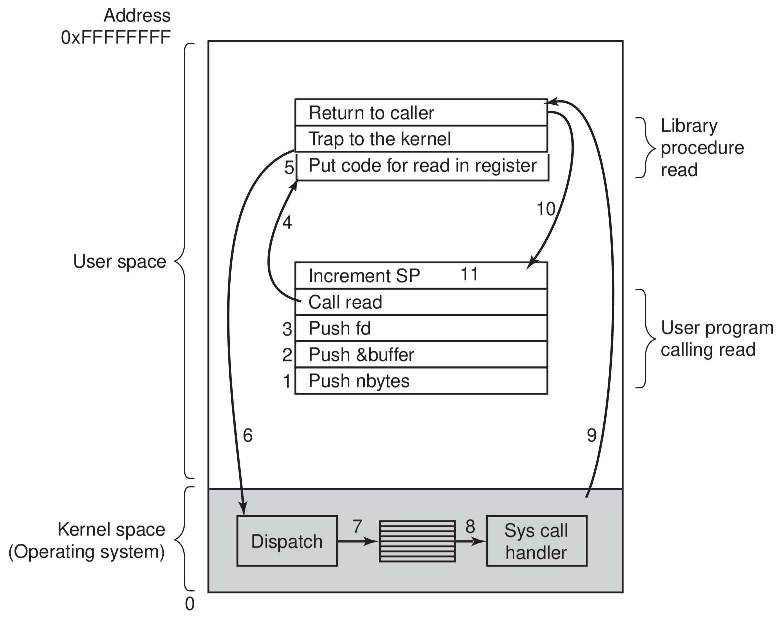
Means

- Resources associated with *program contexts* (processes)
- Activities represented by *execution contexts* (threads)
- *Context switching* between activities
- *Scheduling strategies* for swithcing
- HW support: *Kernel mode* + *memory management unit*

## System Call




## System Call (Elaborate)



## Contexts

### Program (Process)

- Administrative unit holding resources:
  - *Memory address space(s)*
  - *Name space(s)*
  - *Access points* (handles) to other resources, especially files
  - One or more *virtual processors*
  - Security, accounting, and other *information*

### Execution (Thread state)

- State of a virtual processor:
  - *Registers* (including PC and SP)
  - *Stack*
  - Scheduling *information* (thread state, priority, ... )
  - Associated program context

## Process Creation

### Unix

- `fork()`
  Makes an identical copy of calling process except for:
  - Pending signals
  - Return value
- `execve(`*file*`,`*args*`,`*env*`)`
  Loads a new program into current process and starts it

### Windows

- `CreateProcess(`*file*`,`*args*`, ...)` (10 parameters)
  Loads a program into a new process and starts it

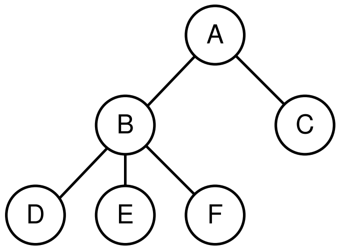
## A Unix Shell

```
while (TRUE) {                          /* repeat forever /*/
      type_prompt( );                   /* display prompt on the screen */
      read_command(command, params);    /* read input line from keyboard */

      pid = fork( );                    /* fork off a child process */
      if (pid < 0) {
            printf("Unable to fork0);   /* error condition */
            continue;                   /* repeat the loop */
      }

      if (pid != 0) {
            waitpid (–1, &status, 0);   /* parent waits for child */
      } else {
            execve(command, params, 0); /* child does the work */
      }
}
```

## Process Relationsships

Unix

- Creator becomes *parent* — only parents can await termination




Windows

- All peers — may pass references