

# OPERATING SYSTEMS II

---

DPL. ING. CIPRIAN PUNGILĂ, PHD.



# User Interfaces

---

A CLOSER LOOK AT THE WAY WE SEE THINGS

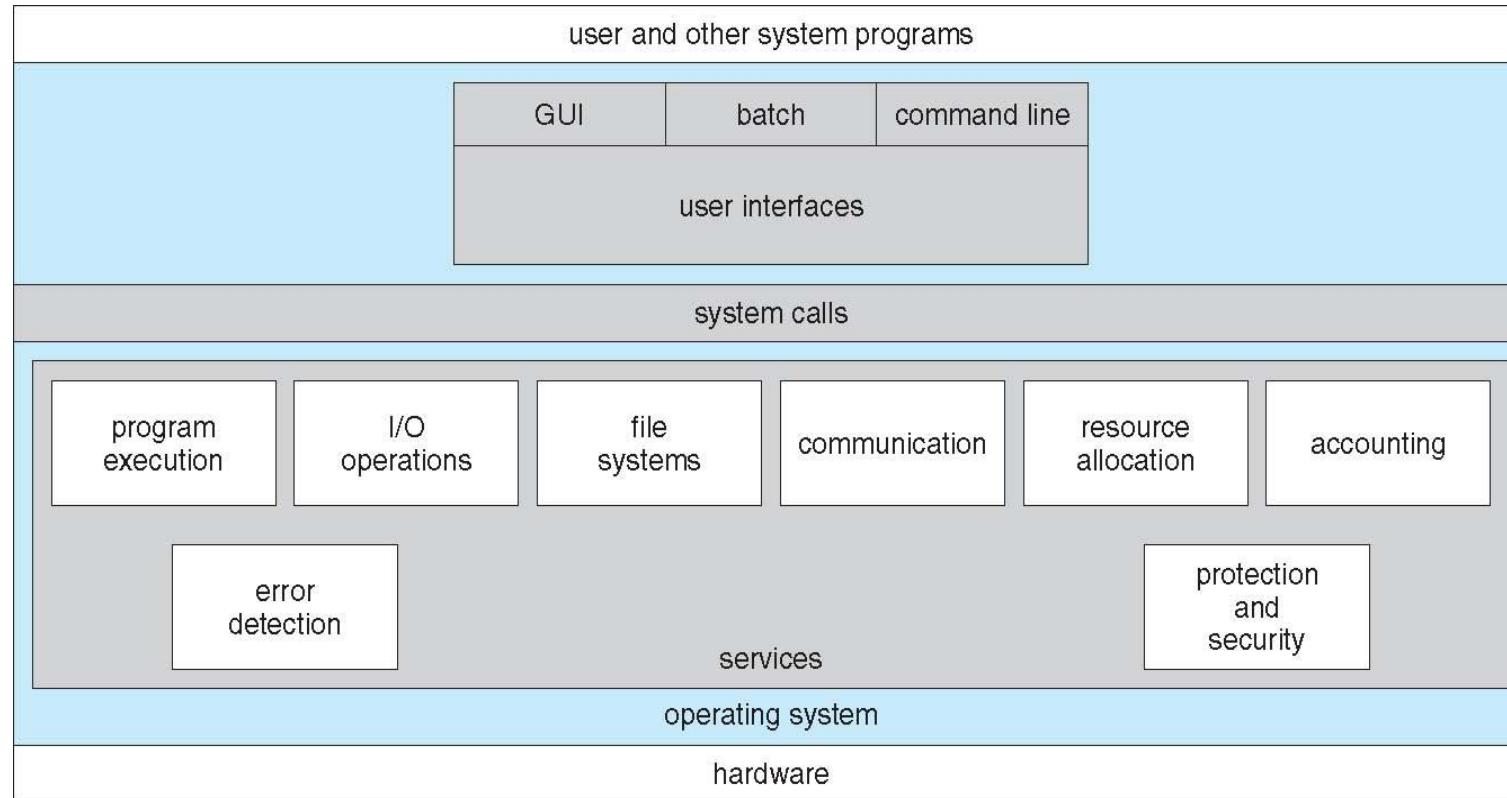
## BIBLIOGRAPHY

1. A. TANNENBAUM, "MODERN OPERATING SYSTEMS", 3<sup>RD</sup> EDITION, 2008, PRENTICE HALL
2. SILBERSCHATZ, GALVIN, AND GAGNE, "OPERATING SYSTEM CONCEPTS", 8TH EDITION, 2009, WILEY

# Graphical Interfaces

## Operating System Services

### □ The general layout



# Graphical Interfaces

## OS Interface - CLI

---

- ❑ Command Line Interface (CLI) or **command interpreter** allows direct command entry
  - ❑ Sometimes implemented in kernel, sometimes by systems program
  - ❑ Sometimes multiple flavors implemented – **shells**
  - ❑ Primarily fetches a command from user and executes it
    - ❑ Sometimes commands built-in, sometimes just names of programs
      - ❑ If the latter, adding new features doesn't require shell modification

# Graphical Interfaces

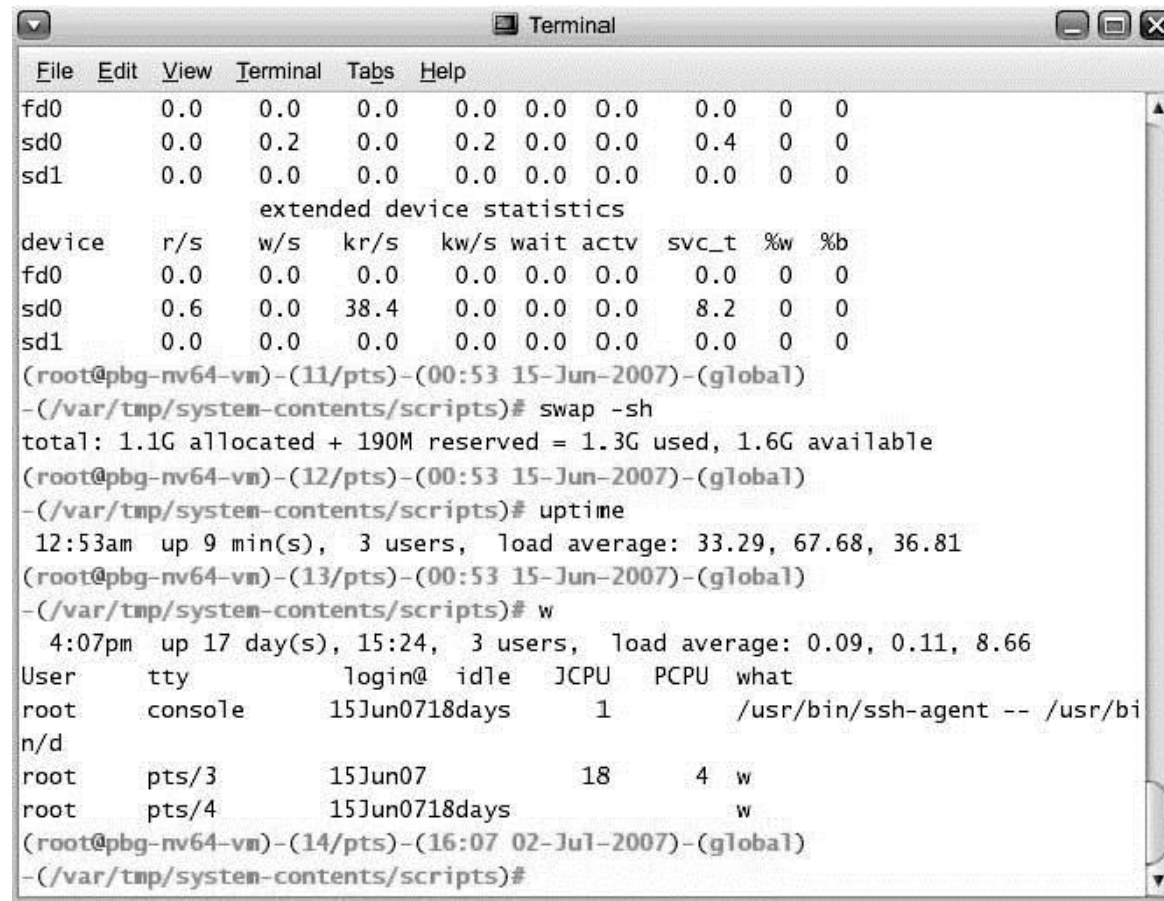
## OS Interface - GUI

---

- ❑ User-friendly **desktop** metaphor interface
  - ❑ Usually mouse, keyboard, and monitor
  - ❑ **Icons** represent files, programs, actions, etc
  - ❑ Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a **folder**))
  - ❑ Invented at Xerox PARC
- ❑ Many systems now include both CLI and GUI interfaces
  - ❑ Microsoft Windows is GUI with CLI “command” shell
  - ❑ Apple Mac OS X as “Aqua” GUI interface with UNIX kernel underneath and shells available
  - ❑ Solaris is CLI with optional GUI interfaces (Java Desktop, KDE)

# Graphical Interfaces

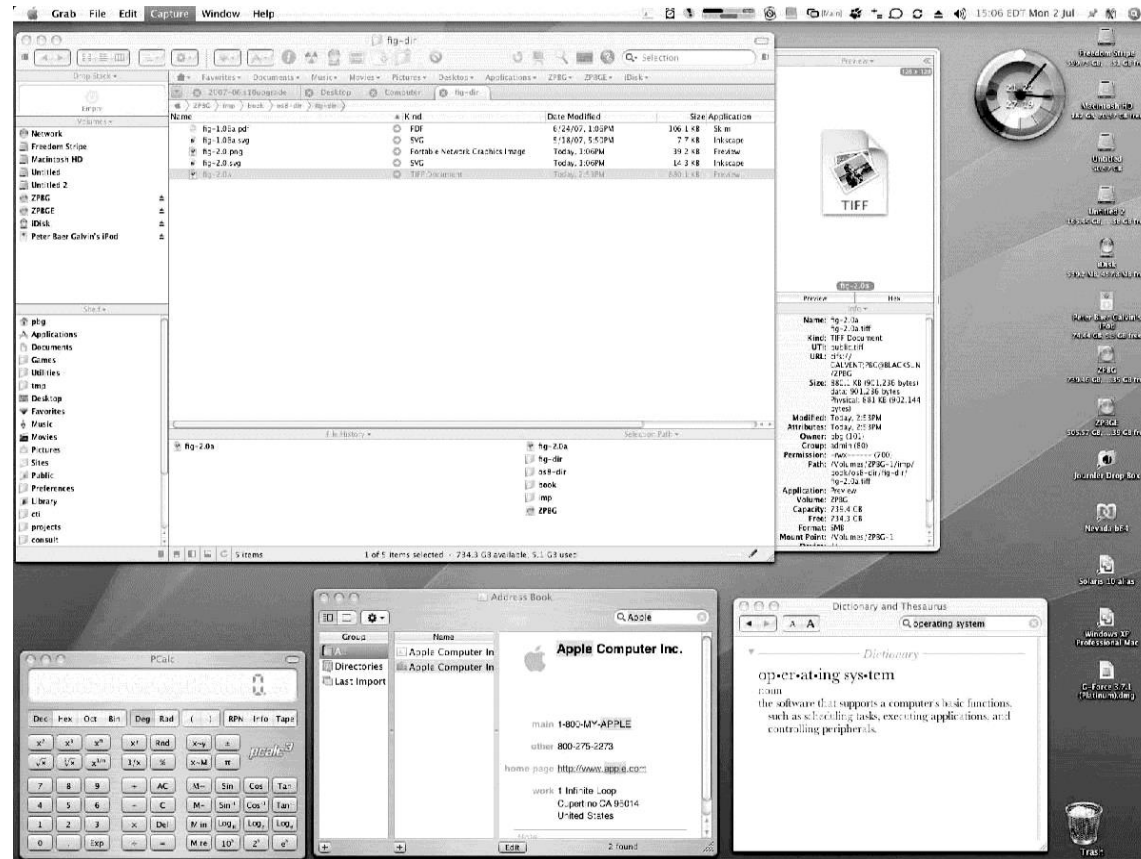
## The Bourne Shell



```
Terminal
File Edit View Terminal Tabs Help
fd0      0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0  0
sd0      0.0  0.2  0.0  0.2  0.0  0.0  0.0  0.4  0  0
sd1      0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0  0
          extended device statistics
device   r/s   w/s   kr/s   kw/s wait actv  svc_t  %w  %b
fd0      0.0   0.0   0.0   0.0  0.0  0.0   0.0  0  0
sd0      0.6   0.0  38.4   0.0  0.0  0.0   8.2  0  0
sd1      0.0   0.0   0.0   0.0  0.0  0.0   0.0  0  0
(root@pbg-nv64-vm)-(11/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# swap -sh
total: 1.1G allocated + 190M reserved = 1.3G used, 1.6G available
(root@pbg-nv64-vm)-(12/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# uptime
12:53am up 9 min(s), 3 users, load average: 33.29, 67.68, 36.81
(root@pbg-nv64-vm)-(13/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# w
4:07pm up 17 day(s), 15:24, 3 users, load average: 0.09, 0.11, 8.66
User      tty          login@ idle   JCPU   PCPU  what
root      console      15Jun0718days   1      /usr/bin/ssh-agent -- /usr/bi
n/d
root      pts/3        15Jun07         18     4 w
root      pts/4        15Jun0718days          w
(root@pbg-nv64-vm)-(14/pts)-(16:07 02-Jul-2007)-(global)
-(/var/tmp/system-contents/scripts)#
```

# Graphical Interfaces

## The MacOS X GUI

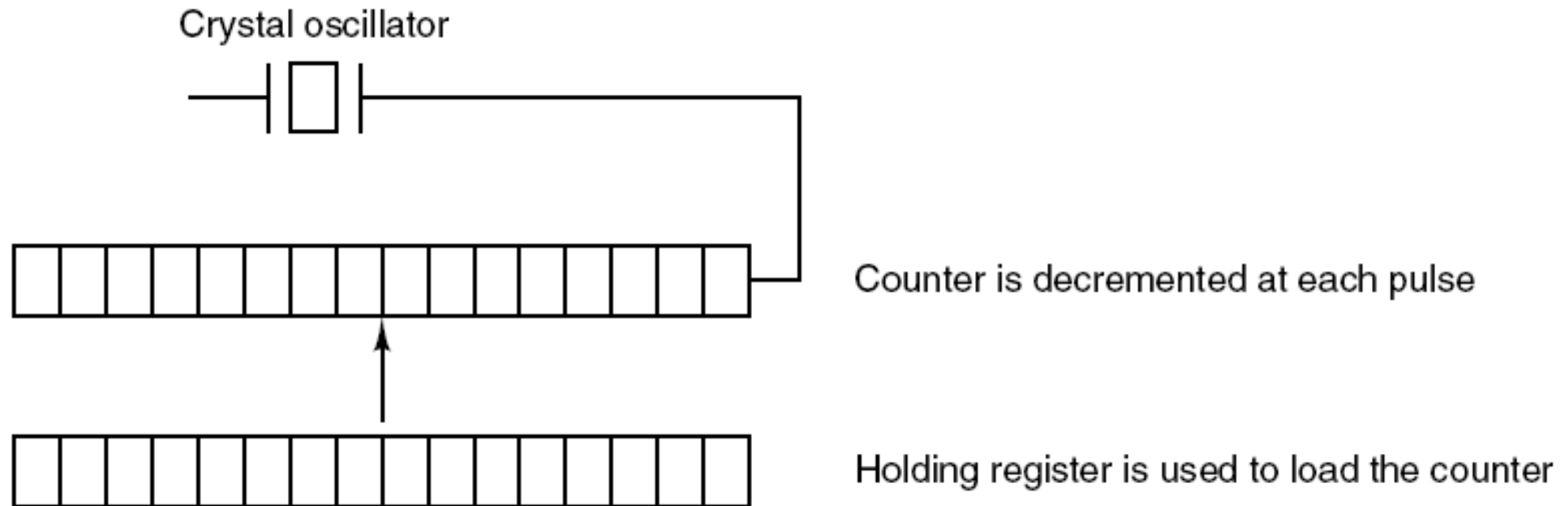


# Graphical Interfaces

## Clocks

---

- A programmable clock





# Graphical Interfaces

## Clock Software

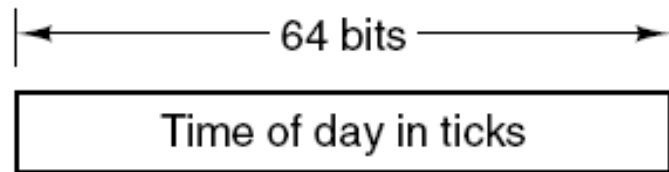
---

- ❑ Typical duties of a clock driver
  - ❑ Maintaining the time of day.
  - ❑ Preventing processes from running longer than they are allowed to.
  - ❑ Accounting for CPU usage.
  - ❑ Handling alarm system call made by user processes.
  - ❑ Providing watchdog timers for parts of the system itself.
  - ❑ Doing profiling, monitoring, statistics gathering.

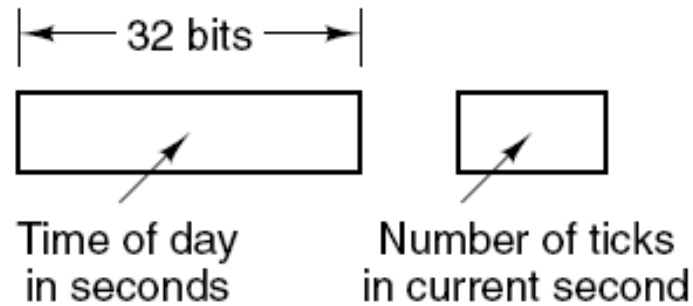
# Graphical Interfaces

## Maintaining Time of the Day

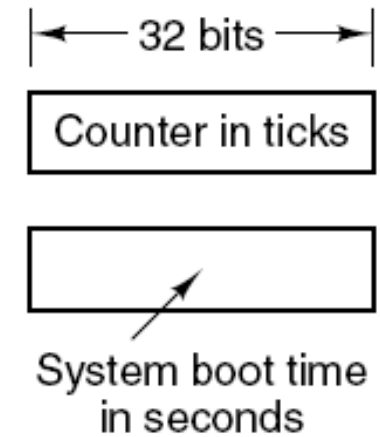
---



(a)



(b)

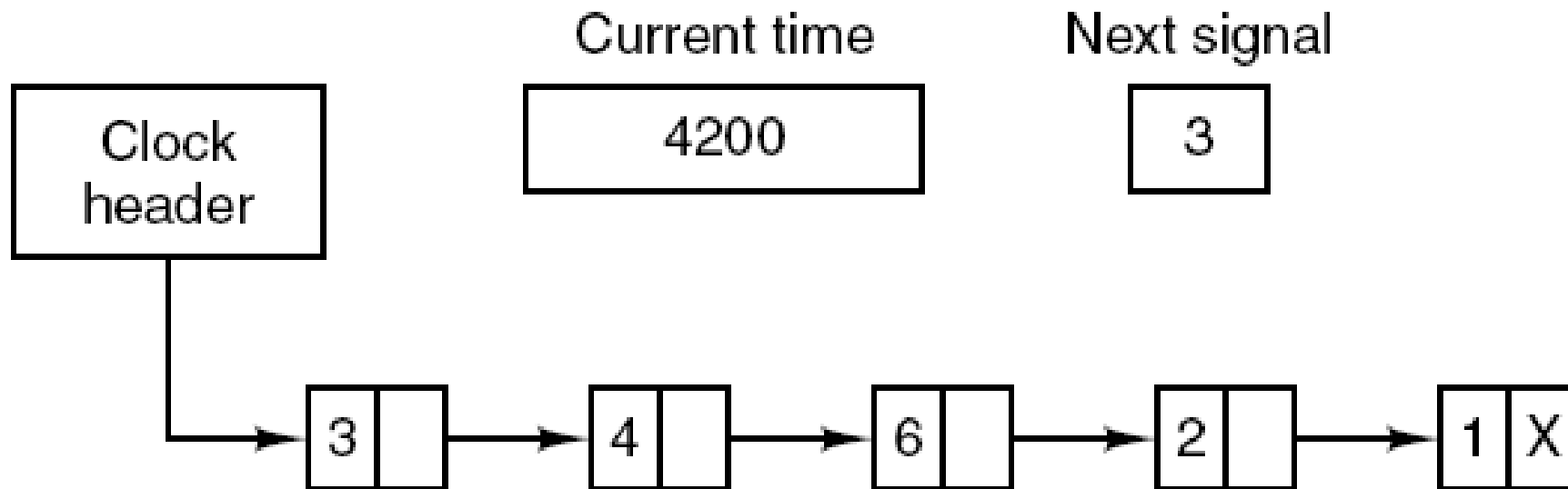


(c)

# Graphical Interfaces

## Simulate Multiple Timers with One Clock

---



# Graphical Interfaces

## Soft Timers

---

- Soft timers succeed according to rate at which kernel entries are made because of:
  - System calls
  - TLB misses
  - Page faults
  - I/O interrupts
  - The CPU going idle

# Graphical Interfaces

## Keyboard Software

---

☐ Characters that are handled specially in canonical mode:

Character	POSIX name	Comment
CTRL-H	ERASE	Backspace one character
CTRL-U	KILL	Erase entire line being typed
CTRL-V	LNEXT	Interpret next character literally
CTRL-S	STOP	Stop output
CTRL-Q	START	Start output
DEL	INTR	Interrupt process (SIGINT)
CTRL-\	QUIT	Force core dump (SIGQUIT)
CTRL-D	EOF	End of file
CTRL-M	CR	Carriage return (unchangeable)
CTRL-J	NL	Linefeed (unchangeable)

# Graphical Interfaces

## The X Window System

---

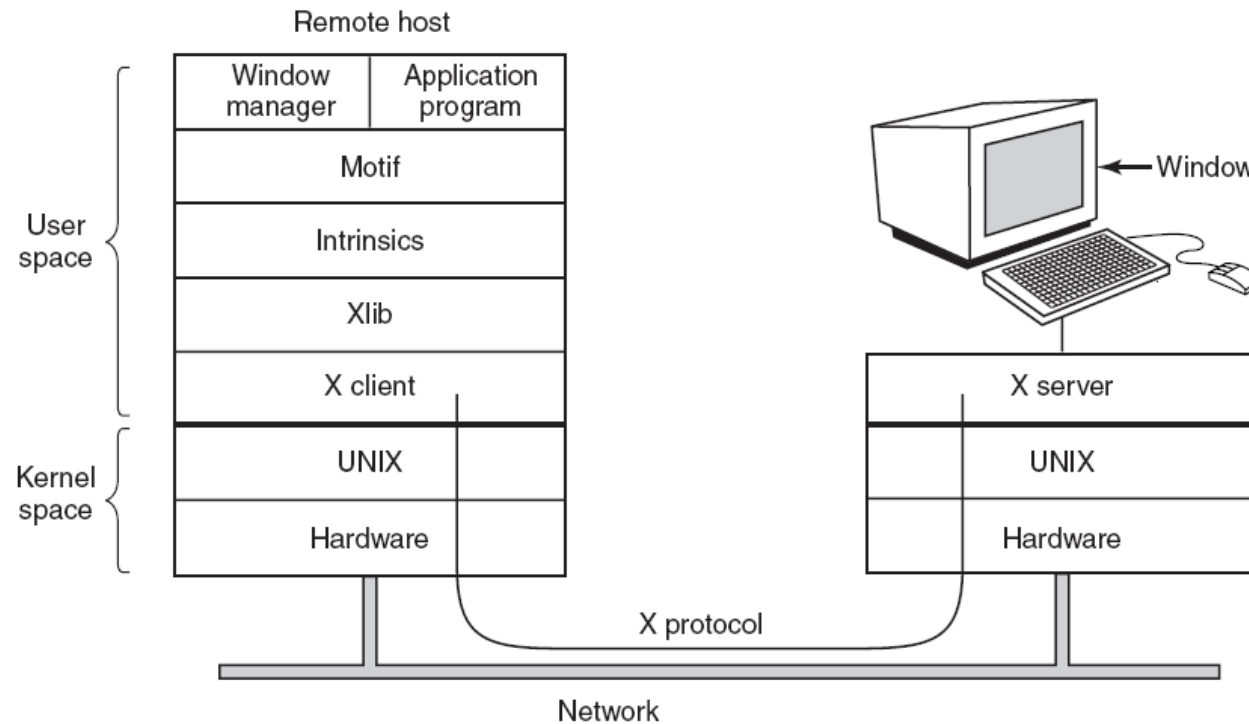
- The ANSI escape sequences accepted by the terminal driver on output. ESC denotes the ASCII escape character (0x1B), and  $n$ ,  $m$ , and  $s$  are optional numeric parameters.

Escape sequence	Meaning
ESC [ $n$ A	Move up $n$ lines
ESC [ $n$ B	Move down $n$ lines
ESC [ $n$ C	Move right $n$ spaces
ESC [ $n$ D	Move left $n$ spaces
ESC [ $m$ ; $n$ H	Move cursor to ( $m$ , $n$ )
ESC [ $s$ J	Clear screen from cursor (0 to end, 1 from start, 2 all)
ESC [ $s$ K	Clear line from cursor (0 to end, 1 from start, 2 all)
ESC [ $n$ L	Insert $n$ lines at cursor
ESC [ $n$ M	Delete $n$ lines at cursor
ESC [ $n$ P	Delete $n$ chars at cursor
ESC [ $n$ @	Insert $n$ chars at cursor
ESC [ $n$ m	Enable rendition $n$ (0=normal, 4=bold, 5=blinking, 7=reverse)
ESC M	Scroll the screen backward if the cursor is on the top line

# Graphical Interfaces

## The X Window System

### □ Clients and servers in the M.I.T. X Window System



# Graphical Interfaces

## The X Window System

---

- ❑ Types of messages between client and server:
  - ❑ Drawing commands from the program to the workstation
  - ❑ Replies by the workstation to program queries
  - ❑ Keyboard, mouse, and other event announcements
  - ❑ Error messages

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>

main(int argc, char *argv[])
{
    Display disp;                /* server identifier */
    Window win;                  /* window identifier */
    GC gc;                       /* graphic context identifier */
    XEvent event;                /* storage for one event */
    int running = 1;

    disp = XOpenDisplay("display_name"); /* connect to the X server */
    win = XCreateSimpleWindow(disp, ... ); /* allocate memory for new window */
    XSetStandardProperties(disp, ... ); /* announces window to window mgr */
    gc = XCreateGC(disp, win, 0, 0); /* create graphic context */
    XSelectInput(disp, win, ButtonPressMask | KeyPressMask | ExposureMask);
    XMapRaised(disp, win); /* display window; send Expose event */
}
```



# Graphical Interfaces

## The X Window System

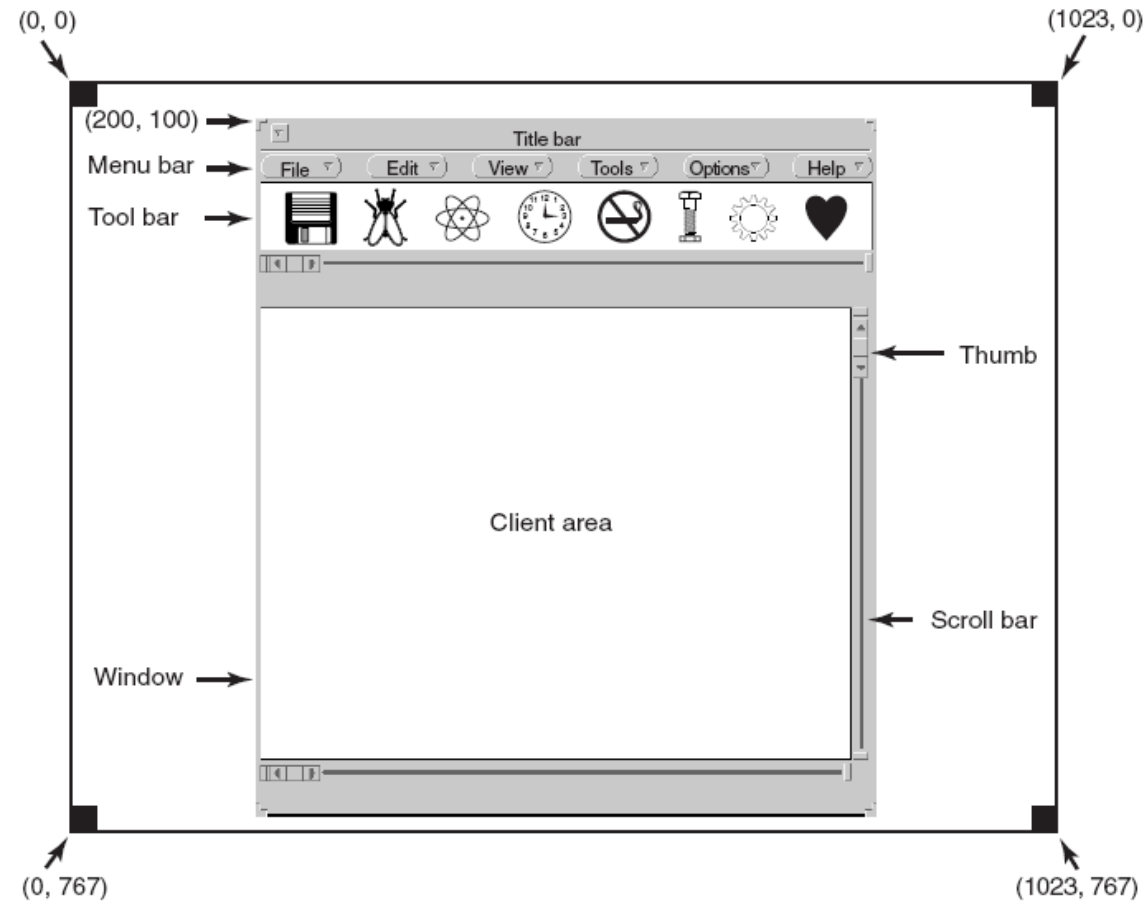
---

```
while (running) {
    XNextEvent(disp, &event);      /* get next event */
    switch (event.type) {
        case Expose:      ...; break;      /* repaint window */
        case ButtonPress: ...; break;     /* process mouse click */
        case Keypress:    ...; break;     /* process keyboard input */
    }
}

XFreeGC(disp, gc);                /* release graphic context */
XDestroyWindow(disp, win);        /* deallocate window's memory space */
XCloseDisplay(disp);              /* tear down network connection */
}
```

# Graphical Interfaces

A sample window at (200,100) on XGA



# Graphical Interfaces

## Skeleton for a Windows GUI

---

```
#include <windows.h>

int WINAPI WinMain(HINSTANCE h, HINSTANCE, hprev, char *szCmd, int iCmdShow)
{
    WNDCLASS wndclass;           /* class object for this window */
    MSG msg;                    /* incoming messages are stored here */
    HWND hwnd;                 /* handle (pointer) to the window object */

    /* Initialize wndclass */
    wndclass.lpszClassName = "Program name"; /* Text for title bar */
    wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION); /* load program icon */
    wndclass.hCursor = LoadCursor(NULL, IDC_ARROW); /* load mouse cursor */

    RegisterClass(&wndclass); /* tell Windows about wndclass */
    hwnd = CreateWindow ( ... ) /* allocate storage for the window */
    ShowWindow(hwnd, iCmdShow); /* display the window on the screen */
    UpdateWindow(hwnd); /* tell the window to paint itself */
}
```

# Graphical Interfaces

## Skeleton for a Windows GUI

---

```
while (GetMessage(&msg, NULL, 0, 0)) {      /* get message from queue */
    TranslateMessage(&msg);                /* translate the message */
    DispatchMessage(&msg);                /* send msg to the appropriate procedure */
}
return(msg.wParam);
}

long CALLBACK WndProc(HWND hwnd, UINT message, UINT wParam, long lParam)
{
    /* Declarations go here. */

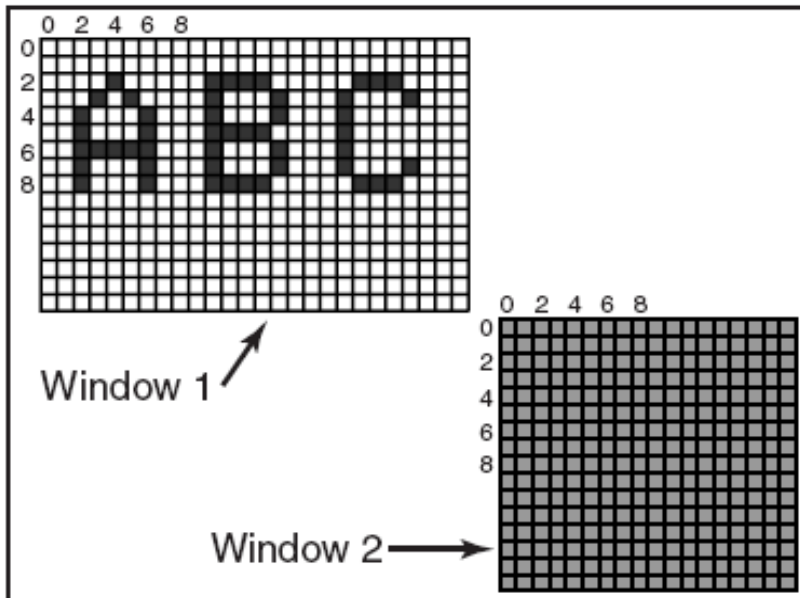
    switch (message) {
        case WM_CREATE:    ... ; return ... ; /* create window */
        case WM_PAINT:    ... ; return ... ; /* repaint contents of window */
        case WM_DESTROY:  ... ; return ... ; /* destroy window */
    }
    return(DefWindowProc(hwnd, message, wParam, lParam)); /* default */
}
```



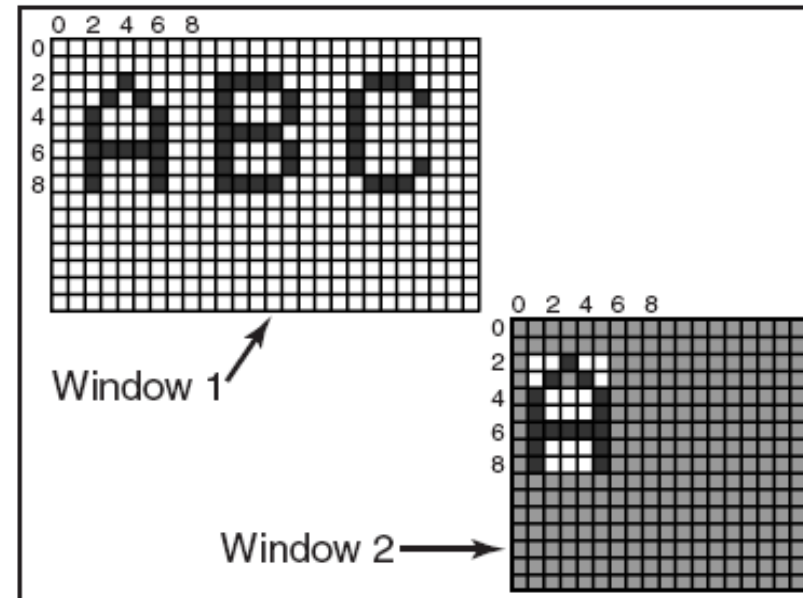
# Graphical Interfaces

## Bitmaps

□ Copying bitmaps using *BitBlt*. (a) Before. (b) After.



(a)



(b)

# Graphical Interfaces

## Fonts

---

- Some examples of character outlines at different point sizes

20 pt: abcdefgh

53 pt: abcdefgh

81 pt: abcdefgh

# Graphical Interfaces

## Thin Clients

---

□ The THINC protocol display commands

Command	Description
Raw	Display raw pixel data at a given location
Copy	Copy frame buffer area to specified coordinates
Sfill	Fill an area with a given pixel color value
Pfill	Fill an area with a given pixel pattern
Bitmap	Fill a region using a bitmap image



# Graphical Interfaces

## Power Management. Hardware Issues.

---

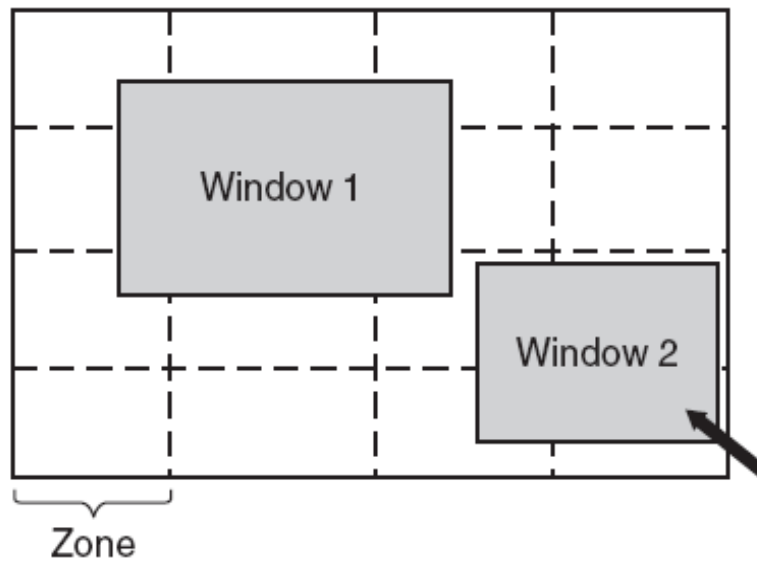
- Power consumption of various parts of a notebook computer

<b>Device</b>	<b>Li et al. (1994)</b>	<b>Lorch and Smith (1998)</b>
Display	68%	39%
CPU	12%	18%
Hard disk	20%	12%
Modem		6%
Sound		2%
Memory	0.5%	1%
Other		22%

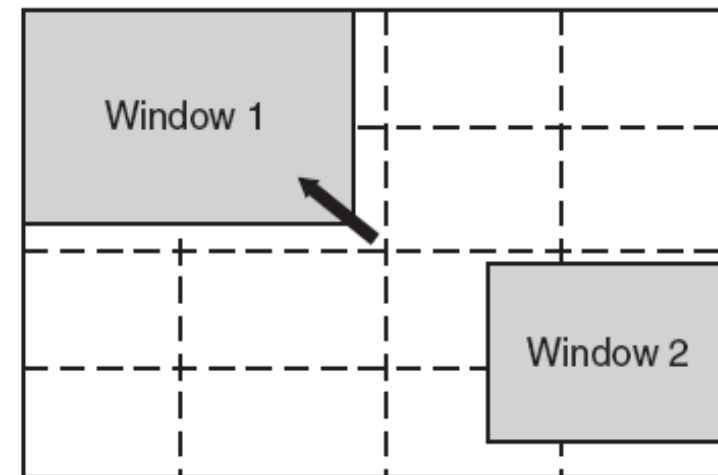
# Graphical Interfaces

## Power Management. The Display.

- The use of zones for backlighting the display
  - (a) When window 2 is selected it is not moved
  - (b) When window 1 is selected, it moves to reduce the number of zones illuminated



(a)



(b)

# Graphical Interfaces

## Power Management. The CPU.

### □ Central Processing Unit

□ (a) Running at full clock speed

□ (b) Cutting voltage by two cuts clock speed by two and power consumption by four

