# CSC 539: Operating Systems Structure and Design

## Spring 2006

Influential operating systems
- Atlas, CTSS, MULTICS, IBM OS/360, UNIX, Alto, Mach

Case studies
- Linux
- Windows XP

Course overview

1

# Influential operating systems

### Atlas (1961), University of Manchester
- batch system with spooling (uses disk as buffer for reading ahead or storing output)
- introduced many OS features: device drivers, demand paging, reference bit

### CTSS (1962), MIT
- first practical timesharing system, supported up to 32 users at terminals
- CPU scheduling involved priorities, multilevel feedback queue to age programs
- swapped entire program out to fast drum

### MULTICS (1965), MIT (+ GE + Bell Labs)
- designed as a general-purpose extension to CTSS (~300,000 lines of code)
- utilized paged segments, multilevel feedback queue
- hierarchical directory structure, protection via access lists
- not fully functional until 1969

### OS/360 (1966), IBM
- designed as a common OS for all IBM computers
- reduced development & maintenance costs, allowed migration of users & programs
- suffered from bloated code, too many complex & interleaved features

2

## Influential operating systems

### UNIX (1972), Bell Labs
- first portable OS, written entirely in C (developed by Thompson & Ritchie)
- open source for academic use, led to BSD UNIX & other variants

### Alto (1978), Xerox PARC
- introduced modern GUI interface, WIMP
- viewed by Jobs, co-opted for Apple Mac OS (and eventually Windows)

### Mach (1986), CMU
- built upon the BSD UNIX kernel
- goals: emulate BSD, support memory models & parallel/distributed, microkernel
- provided support for multiprocessing, threads
- versions of Mach underlie the NeXT OS, Mac OS X

3

## Linux history

### Linux is a modern, free operating system based on UNIX standards
- first developed as a small but self-contained kernel in 1991 by Linus Torvalds, with the major design goal of UNIX compatibility.
- its history has been one of collaboration by many users from all around the world, corresponding almost exclusively over the Internet
- it has been designed to run efficiently and reliably on common PC hardware, but also runs on a variety of other platforms (68000-series, Sun SPARC, PowerMac, …)

### Linux kernel is original, but full system incorporates existing UNIX software
- uses many tools developed as part of Berkeley's BSD operating system, MIT's X Window System, and the Free Software Foundation's GNU project
- Linux kernel is distributed under the *GNU General Public License (GPL)*: free to modify code but cannot make proprietary; also must distribute source code
- many companies (e.g., Slackware, Red Hat, Debian/GNU, Mandrake) market Linux *distributions*: precompiled Linux packages with installation and management utilities

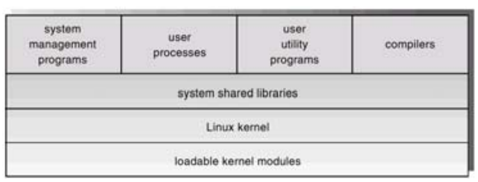
4

## Linux design principles

### Linux is a multiuser, multitasking system with UNIX-compatible tools

- its file system adheres to traditional UNIX semantics, and it fully implements the standard UNIX networking model
- main design goals are speed, efficiency, and standardization
- Linux is designed to be compliant with the relevant POSIX documents

### like most UNIX implementations, Linux is composed of 3 main bodies of code:




1. *system utilities* perform individual specialized management tasks
2. *system libraries* define standard set of functions through which apps interact with the kernel
3. *kernel* is responsible for maintaining the important abstractions of the OS
   - executes in unrestricted kernel mode
   - all kernel code & data in one address space

5

## Process management

### UNIX process management separates the creation/execution of processes

- **fork** system call creates a new process, inherits environment from parent
- a new program is run after a call to **execve**
- a thread is a process that happens to share the same address spaces as its parent (uses **clone** system call instead of **execve**)

### a request for kernel-mode execution can occur in 2 ways

1. program can request OS service (e.g., system call, page fault)
2. device driver may deliver hardware interrupt that triggers interrupt handler

- starting with Linux 2.6, kernel is fully preemptible
- provides spinlocks and semaphores for synchronization

### utilizes 2 different process scheduling algorithms

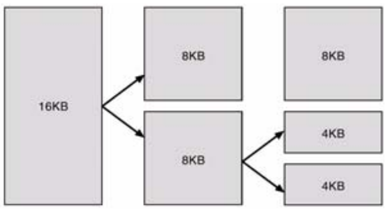
- time-sharing algorithm for preemptive scheduling between multiple processes
  priority based, gives high priority jobs longer time quanta, dynamic priorities
- real-time algorithm for tasks where priorities are more important than fairness
  within priority classes, can use FIFO or round-robin

6

## Memory management

Linux's physical memory-management system deals with allocating and freeing pages, groups of pages, and small blocks of memory

- has additional mechanisms for handling virtual memory, memory mapped into the address space of running processes




page allocator allocates & frees all physical pages

- uses *buddy-heap algorithm* to keep track of physical pages
- each allocatable memory region is paired with an adjacent partner
- whenever buddies are both freed, are combined
- large region may be divided for smaller requests

memory allocations in the kernel occur either statically (drivers reserve contiguous space during system boot) or dynamically (via page allocator)

7

## Virtual memory

the VM system maintains the address space visible to each process

- creates pages of virtual memory on demand, and manages the loading of those pages from disk or their swapping back out to disk as required

the VM manager maintains 2 separate views of a process's address space:

- a *logical view* describing instructions concerning the layout of the address space (address space consists of a set of nonoverlapping regions, each representing a continuous, page-aligned subset of the address space)
- a *physical view* of each address space which is stored in hardware page tables

page replacement algorithm is modified version of second-chance

- each page has age associated with it, measures amount of recent activity
- in effect, implements Least Frequently Used (LFU) approximation of LRU

8

## File & I/O management

### to the user, Linux's file system appears as a hierarchical directory tree

- internally, the kernel hides implementation details and manages the multiple different file systems via an abstraction layer -- the *virtual file system (VFS)*.
- the Linux VFS is designed around object-oriented principles and is composed of two components:
  1. a set of definitions that define what a file object is allowed to look like (inode-object and file-object structures represent individual files)
  2. a layer of software to manipulate those objects

### Linux device drivers appear as objects within the file system

- *block devices* allow random access to independent, fixed size blocks of data
- *character devices* include most other devices; don't need to support the functionality of regular files.
- *network devices* are interfaced via the kernel's networking subsystem

9

## Windows XP history & goals

### Microsoft Windows XP is a 32/64-bit preemptive multitasking OS

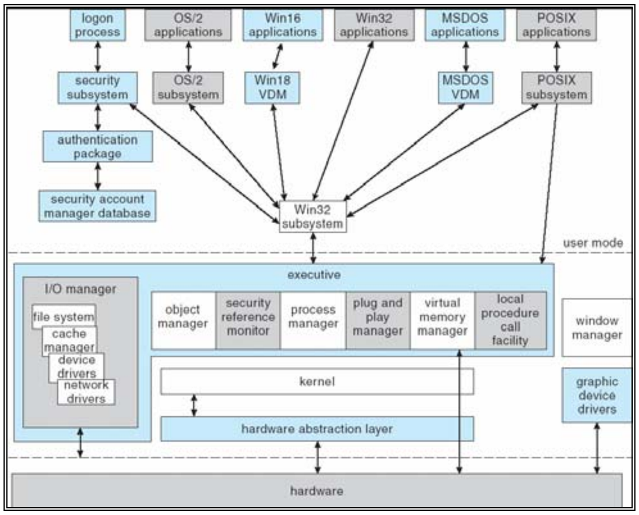
- successor of Windows NT/2000, replaces Windows 95/98
- released in October 2001, server version in 2002
- introduced "better" visual design, simpler menus, easier HCI
- multiuser support through multiple instances of GUI via Windows terminal server

### design goals

- *security:* strict adherence to design standards, extensive code review & testing
  utilizes sophisticated automatic analysis tools to identify security vulnerabilities
- *reliability:* most reliable, stable version of Windows
  extensive manual and automatic code review, driver verification & error-checking
- *compatibility:* introduces compatibility layer to ensure execution of older software
  *also provides POSIX support, so can compile and run most UNIX software*
- *performance:* uses a variety of techniques to build on performance of 2000/NT
- *extensibility:* uses a layered architecture to allow for changes/updates
- *portability:* mostly written in C/C++
  CPU-dependent code isolated in hardware-abstraction layer (HAL)
- *international support:* uses UNICODE, provides support for local languages/formats

10

# XP layered architecture




*hardware abstraction layer (HAL)* hides hardware differences from the OS

*kernel* provides the foundation for the executive and the subsystems

*executive* provides services, including process manager, VM manager, I/O manager

*environmental subsystems* are user-mode processes that enable XP to run processes developed for other OS's (Win32 is base)

11

# Process management

a *process* is an executing instance of an application

a *thread* is a unit of code that can be scheduled by the OS

like NT/2000, XP uses a 32-level priority scheme to determine the order of thread execution

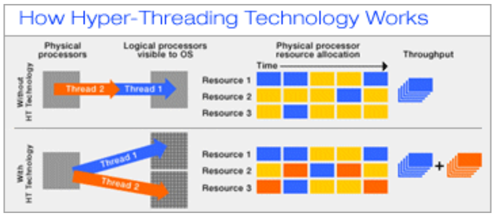
- *real-time class* contains threads with priorities ranging from 16 to 32
- *variable class* contains threads having priorities from 0 to 15
- kernel automatically adjusts priorities based on CPU utilization (I/O-bound ⇑)
  enables I/O-bound threads to keep the I/O devices busy
  CPU-bound threads soak up the spare CPU cycles in the background

12

## Hyper-threading

since 2002, Intel Xeon & Pentium 4 processors have supported Hyper-Threading Technology (HT)

- two architectural states (registers, IC, …) are available on the same processor
- each state can execute an instruction stream (virtually) concurrently
  → a single physical processor behaves as two logical processors



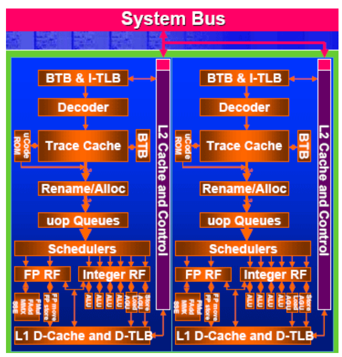

- while it looks like two processors, the execution engine, on-board cache, and system bus interface are shared
- Intel claims 10-30% gain in performance on standard test suites
  (depends upon how much resource sharing goes on between threads)



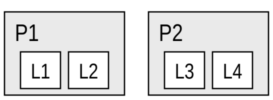
## Dual cores

with the new Intel Core Duo processor, two separate execution cores are embedded in the same chip

- can achieve true parallelism –
  have two threads running concurrently




- with hyper-threading, will appear to the OS as 4 logical processors



# OS support for hyper-threading

## Windows XP & Server 2003 support hyper-threading

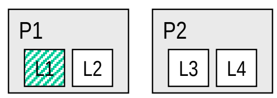
- BIOS reports all logical processors to the OS upon booting
- each logical processor is treated as a separate physical processor by the OS

- threads are scheduled by the OS, which attempts to balance load across processors

  e.g., put slow background job on one logical processor & interactive foreground job on another

  in a dual processor system, will try to balance loads across physical processors

  e.g., if L1 is active, then favor L3 or L4 over L2




HT-enabled applications can set processor affinities for threads to help

15

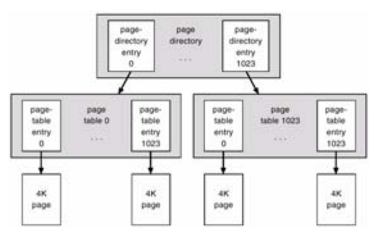
# Memory management

## the VM manager assumes that the underlying hardware supports:

- virtual to physical mapping a paging mechanism
- transparent cache coherence on multiprocessor systems, and
- virtual addressing aliasing

## VM manager uses

- a page-based management scheme with a page size of 4 KB
- multilevel page table, utilizes TLB




- uses per-process FIFO replacement policy
- adjusts number of frames per process by monitoring working-set
- performs prefetching to try to swap in pages before needed

- for performance, allows privileged process to lock select pages in physical memory

16

## File & I/O management

utilizes the NT File System (NTFS)

- provides many modern features: data recovery, fault tolerance, large files, UNICODE names, encryption, compression, …
- a file in NTFS is a structured object consisting of *attributes* (not a simple byte stream, as in MS-DOS or UNIX)
- a directory is stored in a B+ tree structure to provide consistent access time
- NT supports FAT16 to read floppies, FAT32 for Windows 95/98 media

I/O manager is responsible for file systems, cache management , device drivers, network drivers

- keeps track of which installable file systems are loaded, and manages buffers for I/O requests.
- works with VM Manager to provide memory-mapped file I/O.
- controls the 2000 cache manager, which handles caching for the entire I/O system.

17

## FINAL EXAM

Thursday, May 4 8:00 – 9:40

format similar to previous tests

- factual knowledge: TRUE/FALSE
- conceptual understanding: short answer, discussion
- synthesis and application: process scheduling, paging, C++ simulation, …

study advice

- review online lecture notes
- review text
- reference other sources for examples, different perspectives
- look over tests, homeworks, quizzes, online review sheet

18