Slide 1

Week 2 — Overview

- ➜ Operating Systems Overview, continued
- ➜ A Closer Look at System Calls
  - User's perspective
  - Implementation of System Calls
- ➜ Threads and Processes, Part I

Slide 2

Processes

- ➜ Problems occurring in multiprogramming batch systems, time-sharing systems required a closer look at "jobs".
- ➜ What exactly is a Process?

Exact definition is differs from to textbook to textbook:

- ✶ A program in execution
- ✶ An instance of a program running on a computer
- ✶ A unit of execution characterised by
  - a single, sequential thread of execution
  - a current state
  - an associated set of system resources (memory, devices, files)

We define a Process to be an unit of resource ownership

Slide 3

The OS has to

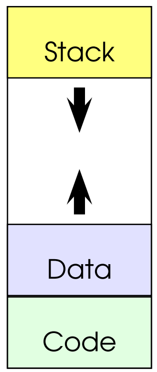
- ➜ Load the executable from hard disk to main memory
- ➜ Keep track of the states of every process currently executed
- ➜ Make sure
  - no process monopolises the CPU
  - no process starves
  - interactive processes are responsive

Slide 4

Process

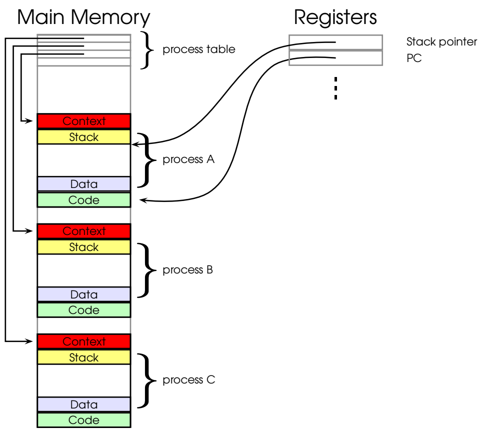
Characterised by:

① An executable program (code)
② Associated data needed by the program (global data, stack)
③ Execution context (or state) of the program, e.g.:
  - contents of data registers
  - program counter, stack pointer
  - state (waiting on an event?)
  - memory allocation
  - status of open files



Slide 5

- ➜ process table keeps track of processes
- ➜ context information stored in Process Control Block (PCB)
- ➜ process suspended: register contents etc stored in PCB
- ➜ process resumed: PCB contents loaded into registers



Slide 6

Dealing with multiple processes is difficult!

- ➜ Synchronization
  - ensure a process waiting for an I/O device receives the signal
  - signals may be lost or duplicated
- ➜ Failed mutual exclusion
  - attempt to use a shared resource at the same time
- ➜ Non-deterministic program operation
  - program should only depend on input to it, not relying on common memory areas
- ➜ Deadlocks

System software is hard to test and practically impossible to prove correct $\Longrightarrow$ Usually buggy

Slide 7

Memory Management

- ➜ Automatic allocation and management:
  - memory hierarchy should be transparent to programmer
  - programmer should not be able to access physical memory directly
- ➜ Process isolation:
  - protect data and memory from other processes
- ➜ Support for modular programming
- ➜ Protection and access control

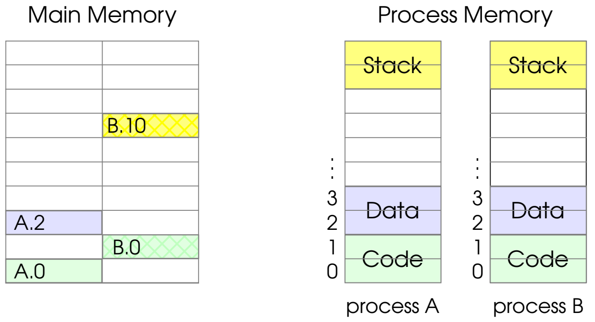
Slide 8

Virtual Memory

Paging and Dynamic Mapping:

- ➜ Process memory is split into equally sized blocks called pages
- ➜ Main memory is also split into blocks of the same size, called frames
- ➜ Pages of a process are dynamically loaded into main memory whenever required

Slide 9



Slide 10

Advantages:

- Reduces start up time of processes
- Reduces fragmentation of main memory
- Possible overlap of execution and loading time of different processes

Slide 11

Virtual Address:

- Virtual address: page number plus offset
- OS maps virtual address to physical address
- From user point of view, every process has its own address space

Advantages:

- Gives applications the illusion to have all RAM to themselves
- Provides an address space for each process which is much larger than actual RAM
- Provides complete isolation of processes from each other

Disadvantages:

- Extra hardware (MMU) is necessary
- Mapping of virtual address to physical address is complicated

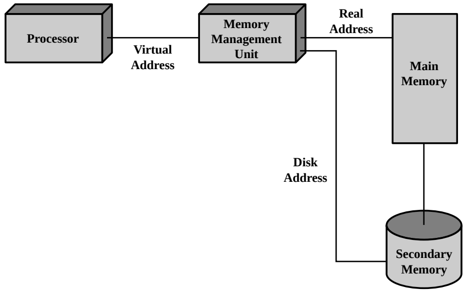
Slide 12

Translation of Virtual Addresses

① Virtual address goes to Memory Management Unit (MMU)
② MMU translates virtual address to physical address
③ causes exception (page fault) if page is not mapped
④ OS (exception handler) fetches page and restarts operation

Slide 13

Virtual Memory Addressing

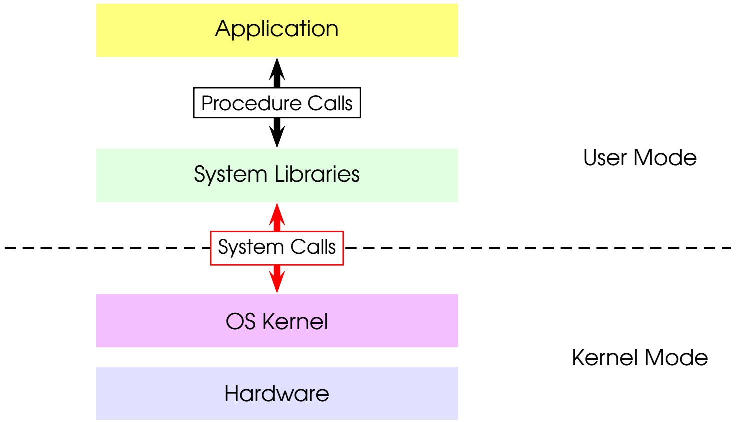


Slide 14

System Calls

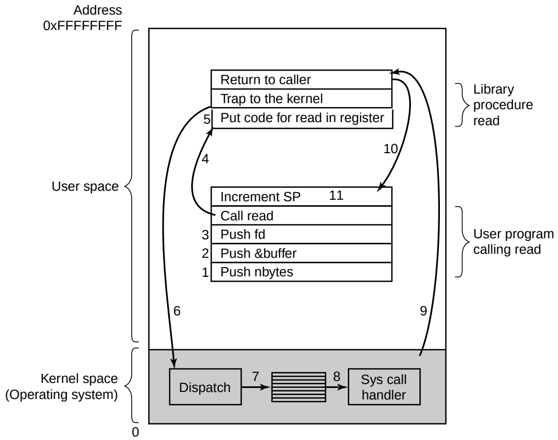
All requests of user level programs for OS services go via system calls:

Slide 15



Slide 16

System Calls



Slide 17

File System

Files and directories (used to group files) provided by the OS to implement a uniform interface to

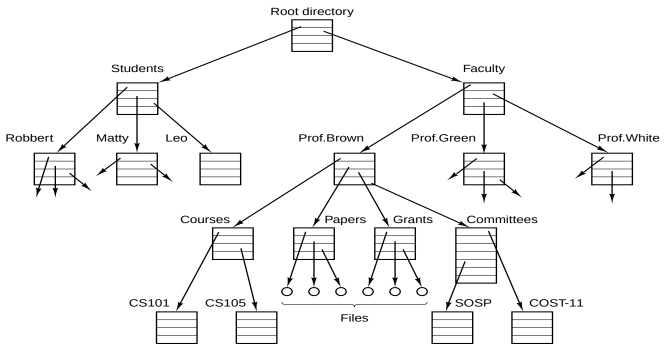
- ➜ disks
- ➜ I/O devices

Provide

- ➜ human-readable name space for data
- ➜ support for exchange of data between systems

Slide 18

Example

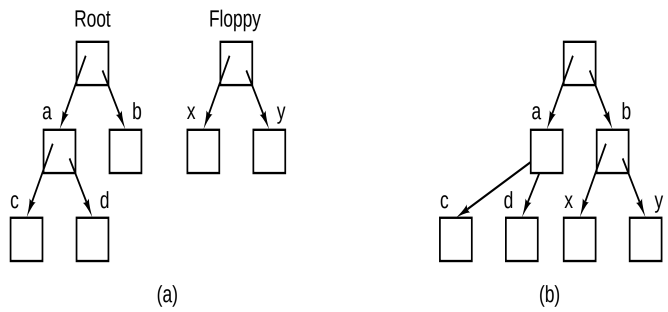


- ➜ Unix-style:`/Faculty/Prof.Brown/Courses/`
- ➜ MS-DOS/Windows style:`\Faculty\Prof.Brown\Courses\`

Slide 19

Mounted File System

- ➜ In Unix-like OS's to provide clean interface to removeable I/O devices



Slide 20

Information Protection and Security

- ➜ Access control
  - regulate user access to the system, e.g.: password protected access
- ➜ Information flow control
  - regulate flow of data within the system and its delivery to users: e.g. Unix file access permissions
- ➜ Certification
  - proving that access and flow control perform according to specifications

Slide 21

Scheduling and Resource Management

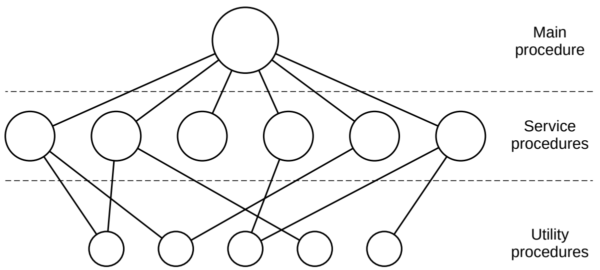
- ➜ Fairness
  - give fair access to all processes
- ➜ Differential responsiveness
  - discriminate between different classes of jobs (interactive, CPU bound)
- ➜ Efficiency
  - maximize throughput, minimize response time, and accommodate as many uses as possible

Slide 22

System Structure

Monolithic Systems:

- ➜ usually evolved from simpler to more complex systems:
  - MS-DOS
  - traditional Unix
- ➜ little internal structure



Slide 23

System Structure

Struggle to cope with the increasing complexity of OS

- ➜ Software Engineering solutions (modular design, clean & simple interfaces) were not sufficient

Hierarchical Layers and Information Abstraction:

- ➜ View the system as a series of levels (lowest may be hardware)
- ➜ Each level performs a related subset of functions
- ➜ Each level relies on the next lower level to perform more primitive functions
- ➜ This decomposes a problem into a number of more manageable subproblems

Slide 24

Examples:

- ➜ THE system, Dijkstra, 1968

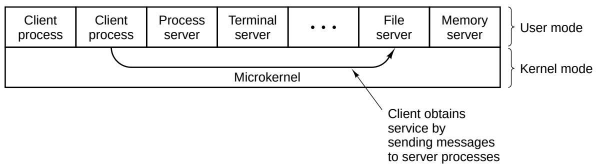
Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

- ➜ MULTICS (M.I.T, Bell, GE)

Slide 25

MICROKERNEL ARCHITECTURE

- assigns only a few essential functions to the kernel
  - address space
  - interprocess communication (IPC)
  - basic scheduling
- other services implemented by user-level servers



Slide 26

MICROKERNEL ARCHITECTURE

➜ Mach, developed mid 80's at CMU

➜ MacOS X based on Mach, many services moved back to kernel

➜ Windows NT partially based on Microkernel architecture

- "modified microkernel architecture"
- OS environments (DOS, Win16, Win32, OS/2, POSIX) run in user mode
- Other services (process manager, vm manager) run in kernel mode

➜ L4 Microkernel Architecture (GMD, IBM)

Slide 27

CHARACTERISTICS OF MODERN OPERATING SYSTEMS

➜ Symmetric multiprocessing

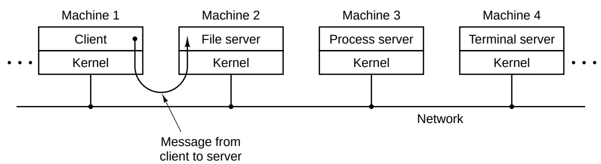
- multiple processors are available
- these processors share same main memory and I/O facilities
- All processors can perform the same functions
- Potential benefits:
  - availability
  - incremental growth
  - performance & scaling

Slide 28

CHARACTERISTICS OF MODERN OPERATING SYSTEMS

➜ Distributed operating systems provide the illusion of a single main memory and single secondary memory space

- distributed file system, distributed shared memory
- microkernel architecture suitable for distributed OS (Cray's Unicos mk)



➜ Object-oriented design

- used for adding modular extensions to a small kernel
- enables programmers to customize os without disrupting system integrity